Waltteri Ylitalo

SOFTWARE-DEFINED NETWORKING, CURRENT STATE, APPLICABILITY AND SECURITY

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2024

ABSTRACT

Ylitalo, Waltteri
Software-defined networking, current state, applicability and security
Jyväskylä: University of Jyväskylä, 2024, 84 pp.
Cyber Security, Master's Thesis
Supervisor: Hämäläinen, Timo

This thesis performs a descriptive literature review on software-defined networking. SD-networks are a new software-based networking approach where the control and data transmission mechanisms are separated. This separation alongside a software-based approach allows the network to become highly flexible and efficient to manage through centralized control features. A key reason behind SDNs flexibility is formed by the emphasis on using software which enables the paradigm to be tailored and customized to fit various different types of networks. This thesis finds that SDN can be applied to various applications such as data centers, internet of things networks, NFVs, and VANETs. These applications offer solutions for both large organizations such as cloud service providers through efficient control tools as well as individual end users through smart home network management. The results of this thesis also find that SDN suffers from multiple security issues that reside in each of the architectures layers. Many of the issues are caused by insufficient authentication and verification features. Additionally, results show that SDN can be prone to service denial attacks through flooding, injections and communication channel overloading. Finally, this thesis finds that a significant portion of existing literature focuses on mainly theoretical settings with models using primarily centralized models and de facto protocols. As a result, future body of research should be expanded by creating a wider range of research settings which are applied to real life environments.

Keywords: software-defined networking, SDN, centralized network control, separated control and data transmission, networking paradigm

TIIVISTELMÄ

Ylitalo, Waltteri
Ohjelmisto-ohjatut verkot, nykytilanne, käyttökohteet ja tietoturva
Jyväskylä: Jyväskylän yliopisto, 2024, 84 s.
Kyberturvallisuus, pro gradu -tutkielma
Ohjaaja: Hämäläinen, Timo

Tämä tutkielma toteuttaa kuvailevan kirjallisuuskatsauksen ohjelmisto-ohjattuihin verkkoihin. Ohjelmisto-ohjatut verkot ovat uusi ohjelmistopohjainen vaihtoehto tietoverkkojen toteuttamiseen. Ohjelmisto-ohjatut verkot erottavat verkon hallinnan ja data siirron toisistaan. Tämä erottelu mahdollistaa verkkojen korkean joustavuuden sekä keskitetyn verkkojen hallinnan. Keskeisenä elementtinä ohjelmisto-ohjattujen verkkojen joustavuuden ja dynaamisuuden mahdollistajana toimii mallin ohjelmistopohjaisuus, mikä mahdollistaa verkkojen räätälöinnin sekä kustomoinnin. Tutkielman tulokset osoittavat, että SDN teknologiaa voidaan hyödyntää palvelinkeskuksissa, esineiden internetin verkoissa, verkon funktioiden virtualisoinnissa sekä ajoneuvojen langattomissa verkoissa. Kyseiset käyttökohteet tarjoavat ratkaisuja niin suurille organisaatioille esimerkiksi pilvipalveluiden tehokkaalle keskitetylle hallinnalle kuin yksittäisille loppukäyttäjille älykotien hallintaan. Tutkielman tulokset osoittavat myös, että mallin arkkitehtuuri sisältään useita haavoittuvuuksia jokaisella mallin kerroksella. Monet haavoittuvuuksista johtuvat puutteellisista tunnistautumis- sekä verifikaatiomenetelmistä. Lisäksi tutkielman tulokset osoittavat, että ohjelmisto-ohjatut verkot voivat olla haavoittuvaisia erilaisilla menetelmillä toteutetuille palvelunestohyökkäyksille. Tutkielman tulokset osoittavat lisäksi, että merkittävä osa aihepiiriä käsittelevästä kirjallisuudesta on keskittynyt käsittelemään aihepiiriä rajallisten tutkimusasetelmien kautta. Merkittävä osa tutkimusasetelmista on toteutettu hyödyntäen keskitettyjä malleja, jotka käyttävät perinteisimpiä protokollia. Tämän seurauksena tulevaisuudessa kirjallisuutta tulisi laajentaa toteuttamalla tutkimuksia useammilla verkkojen malleilla sekä testaamalla malleja käytännön käyttökohteissa.

Asiasanat: Ohjelmisto-ohjatut verkot, keskitetty verkon hallinta, eriytetty hallinta ja datan välitys, verkon paradigma

FIGURES

TABLE OF CONTENTS

1 Introduction

Through the rapid development and emergence of information and communication technologies the world has seen a major shift from an analogical society to a digitalized society. This dramatic change has provided countries and societies with new innovations, opportunities, economic growth, and technologies which have fundamentally changed how the world operates only a daily basis. Today digital technology and the Internet are a key feature in most economic sectors and sections of global infrastructure. In addition to large corporations and countries, most individuals around the globe have access to the Internet, which has also transferred the way day-to-day life is built.

The Internet is everywhere and all-around ranging from mobile devices to large-scale data centers and from websites to smart appliances. Although the Internet is already a key feature and mainstream function of several different aspects of society, it is worth noting that the development is certainly not slowing down but rather quite the contrary. The Internet is still being developed, integrated, and pushed into more and more elements of our daily lives and segments of society. The number of devices with connectivity to the global Internet is rapidly increasing and diversifying through various new network nodes such as smart homes and smart appliances (Keertikumar, Shubham & Banakar, 2015).

Although the Internet and smart technology typically represent new cutting-edge innovations, it should be noted that this does not necessarily apply for all layers of the Internet. Despite popular belief the Internet itself is rather old in terms of its key features and fundamental building blocks. In fact, many of the prominent protocols that sustain and operate the global network are several decades old and have been originally designed to maintain a very different network compared to today's technical needs. For example, key routing protocols such as the Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF) have both been originally designed and implemented in the late 1980s. The same phenomenon can be seen with one of the Internet's most essential architectural models, the TCP/IP architecture which was initially standardized and deployed in 1983. Naturally, the features and models that form the backbone for the global Internet have been revisited, further developed, and improved since their original deployment but nonetheless this does not change the fact, they have been designed to serve very different networking needs compared to today.

Whilst the aforementioned elements of the Internet work and are still a key part of global networking, it is apparent that they contain several restrictions and limitations which cause issues for various network nodes, network managers, and in general to new innovations regarding networking. These issues range from limitations regarding packet sizes and computational power required to operate the TCP/IP architecture (Shang, Yu, Droms & Zhang, 2016) to inefficient and rigid control features for network administrators (Haji ym., 2021). To address the identified issues several different solutions have been proposed ranging from quick fixes and patches to entirely new networking architecture models.

A relatively new approach to improving and mitigating issues caused by traditional networking paradigms are software-defined networks (SDN). By definition these networks are built around various software and make use of programmability as opposed to the typical hardware-based approach implemented by traditional networking paradigms. The most prominent feature of the software-defined networking approach is the ability to differentiate the control and data transmission features from each other. This approach provides several advantages which are discussed in more detail in the latter sections of this thesis, but the most important advantage of this implementation is that SD-networks allow for more dynamic and centralized control of networks. Administrators and network managers are provided with the opportunity to implement and adjust network configurations through a single point of control which is then forwarded to all nodes under the control device.

The SDN approach is not a newly proposed idea as the paradigm of separating data and control planes has been presented initially roughly two decades ago (Feamster, Rexford & Zegura, 2014). However, the technology required to implement a software-defined network has been in many ways lacking causing issues with scalability and reliability especially in large-scale implementations, which has caused the approach to be somewhat ignored. As these voiced issues have been addressed and SDN technology has been further developed, this approach has gained popularity even among major technology and networking organizations such as Google, Cisco, and Microsoft. With further development SDN has the potential to become an even more viable to solutions for individual users as well as organizations to build and manage their networks.

As SDN has been a focus of many companies and scholars this paradigm has produced numerous research papers, architectural approaches, technologies, software, and use cases which have fragmented the topic. As a result, today SDN provides a large and potentially difficult to understand scope of the topic. To address this problem this thesis will conduct a comprehensive literature review regarding the topic SDN in order to compile an up-to-date review of the current state of software-defined networking. In particular, this thesis will focus on four key topics regarding SDN which are the theoretical background of SDN, the current technological options regarding SDN, the current use cases of SDN and finally the prevalent security issues of SDN.

Finally, this thesis will contribute to existing literature as follows. Firstly, the thesis will bring timely relevance to the topic by providing an updated review on the state of SDN. The updated review will bring relevance in particular regarding the security of SDN. Secondly, this thesis will expand previous thesis by providing a more versatile look into the theory of SDN and its possibilities. In many cases the theoretical background regarding SDN is presented through the typical architectural structure of SDN. Whilst this is correct it also excludes other suitable variations and opportunities provided by the SDN model. Thirdly, this thesis will provide a collective review into the actual use cases of SDN in relation to other technical innovations. Lastly, this thesis will also expand other thesis regarding the technical means and applications which can be used to deploy and operate SD-networking. This thesis will also provide a look into protocols which can be deployed to run and maintain a software-defined network.

1.1 Methodology

As previously mentioned, this thesis is conducted as a literature review. By definition, a literature review is research aimed at providing a collective overview of a certain topic or phenomenon (Knopf, 2006). Since the nature of this thesis to provide a collective and up to date view of SDN a literature review is a justified approach. Furthermore, the research questions set for this thesis support this approach. None of the research questions provide a hypothesis which could be tested in a quantitative setting. As for a qualitative approach several issues would also arise.

In terms of type of literature review this thesis opts for a descriptive literature review rather than a systematic review. Whilst the aim of this thesis is to provide a comprehensive review of the topic, it is worth noting that a fully systematic approach to this would be unfeasible in regard to the resources of this thesis. SDN has captioned the interest of numerous researchers and as such the existing body of research regarding the topic is quite extensive. Therefore, attempting a fully systematic review would expand this paper significantly. Secondly, this thesis focuses on two main parts which are applications and security of SDN. Performing a systematic review would require the topic to be further narrowed in order to feasible.

The research material used for this thesis will be collected through free online databases and other databases provided by the university of Jyväskylä. In terms of data this thesis uses and acknowledges primarily research which is published in forums known and graded by the grading tool Julkaisufoorumi (JuFo). This thesis focuses on sources which have received a JuFo grading of at least one or higher. However, certain exceptions are be applied for example whilst showcasing available SDN tools and when studying original documentation of protocols.

The second limitation regarding the research material is made regarding the security of SDN. As previous work and other thesis have addressed the topic of SDN security, it is reasonable to make a limitation on the timely relevance of the material. As such, this thesis will only address SDN security research from 2018 and forward in order to maintain the timely relevance of the security and to avoid repetitiveness.

1.2 Research questions

This thesis answers the following research questions:

1) What are software-defined networks?
2) What are the benefits and disadvantages of SD-networks?
3) Where can software-defined networks be applied?
4) What are the main security issues of software-defined networks?

1.3 Thesis outline

This thesis is structured in the following way. Chapter one provides an introduction to the topic of the thesis as well as presents the research questions that the thesis answers. Additionally, chapter one also addresses the chosen methodology for this thesis and reasons why the approach was chosen. The second chapter acts as the theoretical background of the thesis. Chapter two showcases different variations of the SDN architecture, different layers and interfaces used in SDN and lastly an overview of the benefits and disadvantages of SDN. Chapter three focuses on the current state of available technologies and solutions to implement SDN. Chapter three looks at each layer individually and showcases the different options for implementing each layer. In addition, the third chapter presents various different protocols which can be used for different solutions.

Chapter four focuses on the current use cases of SDN and present different instances where SDN is incorporated or integrated as a part of the technology stack. Chapter five provides an updated view on the security issues regarding SDN. This chapter also approaches security concerns layer by layer and also touches on issues regarding the interfaces. Chapter six provides discussion of the thesis findings, and other notions found during the research process. Finally, chapter seven presents the conclusion and summary of this thesis.

2 Theoretical background of software-defined network-ing

This chapter presents the theoretical background of the topic. This chapter showcases the framework behind software-defined networking and addresses each component in more detail. In addition, this chapter presents multiple different architectural models for setting up SDN, compares them and highlights their differences. Finally, this chapter also showcases the main advantages and disadvantages of the SDN paradigm, primarily from the standpoint of performance and feasibility.

2.1 SDN layers

The architecture of software-defined networks is split into three layers, which each maintain a different function among maintaining the operability of the network. These layers include the top layer which is known as the application layer or application plane. The application plane is the layer which is visible for most of the end users and typically includes various software required within the network (Rawat & Reddy, 2016). Bellow the application plane resides the control plane or control layer, which essentially operates as the brains of operation by managing the networks routing and traffic operations (Xia ym., 2014; Rawat & Reddy, 2016). Furthermore, the control plane also acts as the bridge, which connects both the application layer and the infrastructure layer. Finally, the last layer of the framework is the infrastructure layer or otherwise known as the data plane. As implied by its name the infrastructure layer is the part of the network which hosts and maintains the nodes that form the network. The infrastructure layer topically performs three main functions which are data packet trafficking, implementing the rules and commands given by the control plane and forwarding the aforementioned rules to other nodes (Xia ym., 2014; Haji ym., 2021).

Alongside the three layers SD-networks consist of at least two interfaces which connect the layers together and maintain the operability of the framework. The two required interfaces are typically dubbed as the northbound interface and southbound interface. The northbound interface is an API which handles the connection and traffic between the application and control plane. Consequently, the southbound interface ensures the connection between control plane and the infrastructure plane. The previously addressed interfaces are required in every implementation of SDN but in addition to these interfaces different SDN models can also require additional interfaces, which maintain a horizontal connection rather than a vertical one. With more complicated SDN structures the networks may have for example several controllers within the controller plane which require so called eastbound or westbound interfaces to ensure connectivity. These interfaces can also be used as a link to connect so called legacy systems operating with a traditional networking model to SDN networks.

Figure 1 presents a simplified version of SDN architecture as well as parts of each layer. Next each of the layers and interfaces of SDN will be addressed in more detail.

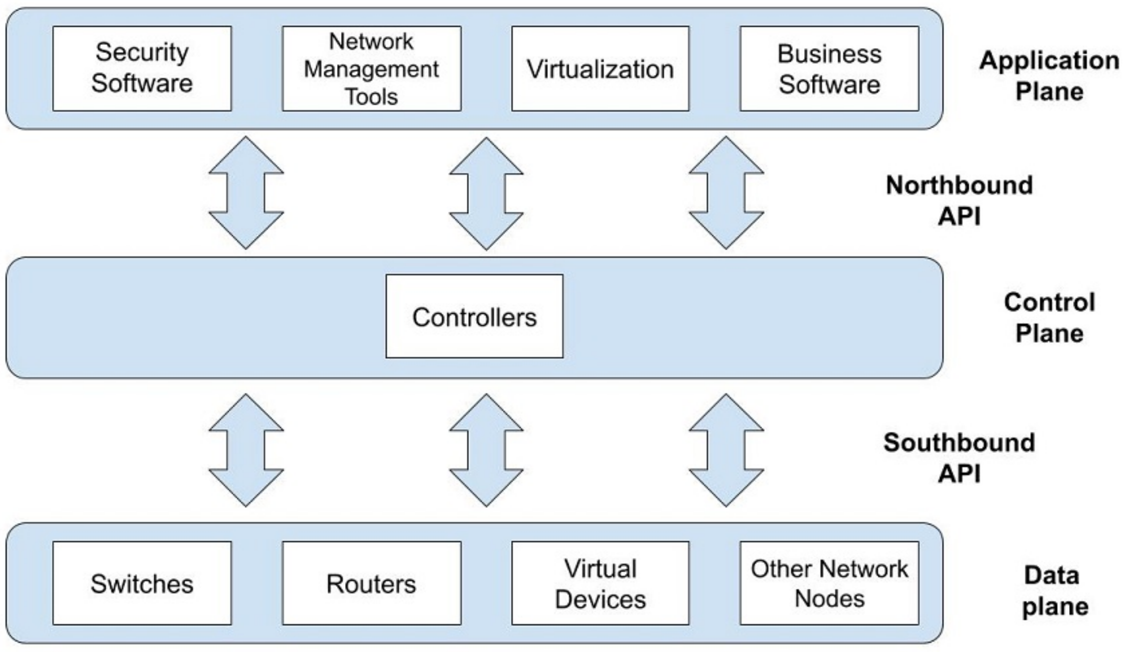

FIGURE 1 Illustration of SDN architecture.

2.1.1 Application Plane

As previously stated, the top layer of SDN models is the application plane which hosts software used within the network alongside operating as the interface connecting the users to the network. Whilst the controllers are responsible for maintaining and implementing network changes the application plane holds an important role in network management. The user interfaces provided by software enable network managers to monitor the status of the network as well as to give the actual commands which are passed to and executed by the controllers (Haji ym., 2021). Although the application plane is fundamentally responsible for giving commands to the control plane, it should be noted that the plane can also perform and incorporate numerous other functionalities.

The application plane is able to host and operate many different software due to its programmability and can execute various other tasks. The most common software include programs which enable network management, routing, as well as traffic monitoring (Nisar ym., 2020). In addition to traffic monitoring, the application plane can also make use of the control planes ability to map out network structures by operating programs designed to display the current topologies of managed networks (Shafique ym., 2020). These functionalities can provide network managers with visual information regarding the current physical state of the network. Other network functionality programs enabled by the application plane can include for example load balancing software, which are designed to help network managers diversify and redirect traffic flows based on occurring circumstances.

Outside network functionality tools it is worth noting that the application plane enables the use of various security related features and software solutions. These solutions include traditional tools such as intrusion detection systems (IDS), access and authentication solutions, firewalls, and other intrusion prevention systems (IPS) (Li, Chen & Fu, 2019; Chica, Imbachi & Vega, 2020). These features can be used to provide security to the network by mitigating attack vectors targeted at the application layer.

2.1.2 Northbound API

With SDN implementations northbound interfaces are tasked with ensuring connectivity between the applications hosted in the application plane and the controllers in the control plane. The requirements placed upon this interface vary based on the needs and implementation of the application plane as the used software dictates data needs. For this reason, there haven't been many efforts to standardize the northbound interface unlike to southbound interfaces (Jarschel ym., 2014). In many cases the protocol deployed to perform the required functionalities is done on an ad hoc basis depending on the chosen controller or the layout of the application layer (Ahmad & Mir, 2021).

Although no official standards are imposed there are certain APIs which are widely used to maintain the interface. One of the most common implementation approaches uses the RESTFUL (also known as REST) API to perform the communication with basic HTTP requests (Zhu ym., 2021). Alternative protocols to maintain the interface include but are not limited to Simple Network Management Protocol (SNMP), Netconf, JSON, Frenetic and Merlin (Shirmarz & Ghaffari, 2020; Ahmad & Mir, 2021; Bannour, Souihi & Mellouk, 2017). These interface implementations also enable using other means of communication as opposed to pure HTTP such as communicating through XML, JSON format and specific commands used by SNMP.

2.1.3 Control Plane

The control plane acts as the middleman in all SDN architectures and is primarily responsible for making sure that the whole architecture is able to operate smoothly. Furthermore, controllers operating in the network also ensure that all of the layers within the model are able communicate through the interfaces maintained. The most fundamental and important task for the controller is to maintain and configure the network based on the policies and instructions provided by the application layer (Ahmad & Mir, 2021). Controllers deployed in the plane contain various different modules with specified functions that can be used to perform each task, such as giving new routing rules to the network nodes. These modules typically consist of at least the link discovery, topology manager, decision-making, storage manager and flow manager modules (Zhu ym., 2021; Ahmad & Mir, 2021). However, the structure of the controller and the hosted modules can vary based on the design of the controller, needs of the network or protocols used to maintain APIs (Ahmad & Mir, 2021).

To control the routing of network nodes within the infrastructure plane, controllers use flow managers to forward rules that are then later implemented by the nodes. Other routing decisions are handled by the decision-making module. In case the network nodes are unsure how to route data packets, they can send questionnaires to their respected controllers which are then tasked to make the final decisions based on how the data packets should be forwarded. Outside of managing and configuring routing rules, the controllers also typically manage features directed at application plane such as maintaining information regarding the state of the network (Ahmad & Mir, 2021). In order for the controllers to be able to make decision regarding the routing choices, controllers also host a feature which can be used to probe the current status and topology of the network. Controllers can use their link discovery modules to send out requests for all network nodes which can then be used to map out the active hosts based on the responses with the topology manager module (Wazirali, Ahmad & Alhiyari, 2021). This information is typically stored within the controller and can be used to make routing decision as well as to be provided to the application layer. In addition, other statistic regarding the networks functionality can be collected, stored, and forwarded (Ahmad & Mir, 2021).

Since the control plane acts as the connecting middle point for the whole architecture, it is also required to provide and host all of the required interfaces connecting the different layers within the system. Naturally, the control plane maintains both the north- and southbound APIs to enable the connectivity and functionality of the network. Depending on the structure and complexity of the SD-network lateral connectivity may also be required (Ahmad & Mir, 2021). In many cases the control plane can contain multiple controllers both for performance and security enhancement reasons. As a result, the controllers are required to communicate with each other through east- or westbound interfaces (Zhu ym., 2021). These interfaces can also be used to connect other SDN systems or legacy systems to the network.

2.1.4 Southbound API

Southbound APIs are interfaces which link together the control plane and data plane ensuring that the network nodes can be managed and configured. Southbound interfaces pass forward routing rules dished out by the controller and pass them forward to the network nodes. Unlike the northbound interface, the southbound interface has a standardized popular de facto protocol called OpenFlow, which is widely implemented in SDN solutions (Haji ym., 2021; Zhu ym., 2021). OpenFlow is an open-source code protocol originally developed in late 2000s (Lara, Kolasani & Ramamurthy, 2013). Due to its open-source and early availability, OpenFlow is widely used as the primary option for maintaining the interface and is sometimes mistakenly even considered as the only option (Shirmarz & Ghaffari, 2020). However, this is not the case as several other protocols have been developed which offer alternative implementation options.

In addition to OpenFlow, several companies and organizations have created alternative protocols to rival OpenFlow and to provide users with protocols enabling new features. Examples of these protocols include Cisco's onePK, IETF's ForCes and LISP, PCEP, PCE and PoF protocols (Shirmarz & Ghaffari, 2020; Jarschel ym., 2014). Each of these protocols provide network managers with alternative options that can be used to tailor make their network based on specific needs. It is however worth noting that unlike OpenFlow all of these solutions are not open-source (Shirmarz & Ghaffari, 2020) and thereby free, which can also be seen as a factor limiting their popularity.

2.1.5 Data Plane

The data plane forms the infrastructure layer of SDN and is responsible for transmitting data between various network nodes (Shirmarz & Ghaffari, 2020). Typically, in SD-networks the data plane consists of switches, routers, gateway points, and virtual devices which under the rules provided by the controller transmit data within the network. Outside of data forwarding the data plane requires networking nodes to host a specified protocol which enables the handling of traffic flow rules guiding the routing (Wazirali, Ahmad & Alhiyari, 2021). Each routing device maintains its own flow table consisting of rules provided by the controller that guide the routing (Shirmarz & Ghaffari, 2020). Devices within the data plane also contain a protocol designed to respond to the controller's topology mapping requests. A commonly used variant for this protocol is the Link Layer Discovery Protocol (LLDP), which is used in OpenFlow protocol implementations (Wazirali, Ahmad & Alhiyari, 2021).

It is worth noting that due to programmability of SDN and the data plane the scope of potential network devices is rather heterogenous. Various sensors and other data producing devices can be incorporated into the network. Examples of these devices can include for example mobile devices, Internet of Things (IoT) nodes and smart vehicles.

2.1.6 Eastbound & Westbound API

Eastbound and westbound interfaces are typically required in more complex SDN architectures such as in distributed models which host multiple controllers. They can also be used to connect separate network domains which both host the SDN approach (Jarschel ym., 2014). Like with the northbound interface no standard for these interfaces exists and the final implementation is often based on the requirements posed by the chosen controllers (Zhu ym., 2021). With architecture models simply using one controller type these interfaces are typically easier to maintain.

Another variant is a hybrid SDN approach which incorporates legacy systems running traditional network structures to an otherwise SDN-based network. Since hybrid models incorporate two fundamentally different networking approaches the connection can be established either directly through the gateway devices or by utilizing specialized middleware as an exchange point (Ahmad & Mir, 2021). Regarding the actual protocols used to maintain connectivity examples include the Border Gateway Protocol (BGP) (Zhu ym., 2021; Ahmad & Mir, 2021) and Multi-Protocol Label Switching (MPLS) (Jarschel ym., 2014).

As a final note regarding the aforementioned interfaces, it is worth noting that neither the east- or westbound interface has a specified function and either one can be used for example as the interface for incorporating a legacy system.

2.2 SDN Architectures

Due to the flexible and programmable nature of SDN network managers are able to configure and setup various different networking topologies based on the needs of the operations. SDN-based networks can be optimized to serve different needs and qualities for different kinds of networks by simply selecting a desired architecture type, which is implemented using the aforementioned elements. For this reason, SDN networks can be used for a wide range of implications ranging from small home networks to large-scale data centers consisting of thousands of nodes. Existing literature regarding SDN, and typical architecture models typically present primarily three models which are centralized models, distributed models, and hybrid models. The key feature separating these architectural models is the implementation of the control planes structure and the connections between the controllers. In essence, the architectural model is determined by the controllers and how they maintain the network.

It is worth noting that although this thesis only presents three main architectures the possible implementations have more variety and options in reality. For example, distributed or hybrid models can be implemented in many different ways or by using different tools to establish connectivity. However, regardless of the intricate details the models are still considered as one of the previously mentioned and will not be addressed in more detail in this thesis. Next up the three architectures are addressed in more detail.

2.2.1 Centralized architecture

The simplest solution to SDN architectures is the centralized controller model showcased below in figure 2. This approach uses a single controller to control the networks routing and flow rules as well as maintaining the connection to the application plane. With the centralized approach the control of the network is simple as the network administrators have to only deal with one single control device and maintain its functionality (Paliwal, Shrimankar & Tembhurne, 2018). Furthermore, with centralized models eastbound and westbound interfaces are not required reducing the workload and requirements posed upon the controller. Centralized models also enable benefits regarding the versatility of the application plane, especially with small network implementations. Since the managed networks are small and the control is tightly restrained, new applications can be integrated in the network more easily (Ahmad & Mir, 2021).

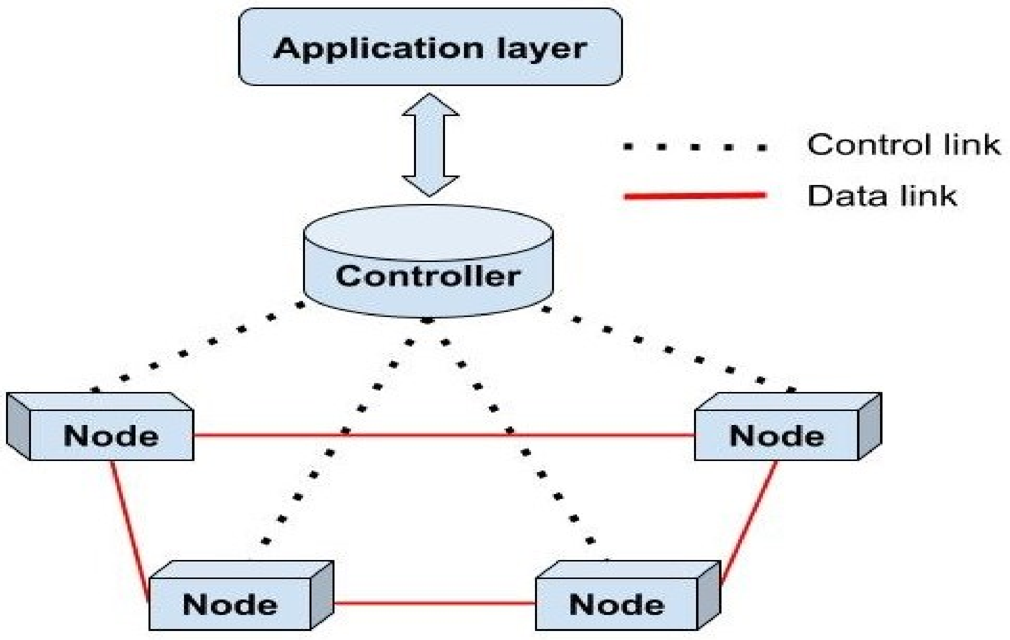

FIGURE 2 Centralized SDN architecture.

However, despite the simplicity centralized controller models do also face difficulties which often limits its applicability in real life use cases. Centralized models are typically most suitable for small network that do not pose too much traffic and workload because a single controller can become a choke point when overwhelmed (Alsaeedi, Mohamad & Al-Roubaiey, 2019). In essence, if the controller becomes overloaded with network traffic or is targeted with a cyberattack the entire network can be rendered useless. As such, centralized models are not the most suitable for network environments that host thousands of nodes.

In addition to actual physical hardware, dynamic networks can also pose limitations on the feasibility of centralized SDN networks. In certain networks dynamic scalability is required which means that based on the workload new network nodes or virtual devices may be needed within the network. With centralized networks this can cause issues as the joining and dropping of network nodes requires the controllers topology manager to keep track of the topology causing an increase in workload (Alsaeedi, Mohamad & Al-Roubaiey, 2019; Ahmad & Mir, 2021). This can be seen as major issue as one of core benefits offered by the SDN paradigm is the promise of increased and cheaper scalability of the network.

2.2.2 Distributed architecture

Distributed architecture models expand the centralized models by incorporating multiple controllers into the control plane. With the distributed approach the workload of the control plane is shared among numerous controllers removing choke points and increasing both the scalability as well as performance of the networks (Paliwal, Shrimankar & Tembhurne, 2018). With distributed models network managers have more possibilities in regard to structuring the networks and domains controlled by each controller. This can be used to share the workloads of the controllers by making use of load balancing which enables upscaling and applicability in dynamic networks. Distributed network models can also be used to increase the resilience and reliability of the network by providing alternative controllers for network nodes in case the primary designated controller becomes unusable (Bannour ym., 2017).

Existing literature presents two distinctive alternatives for structuring and maintaining a distributed SDN approach. These options are dubbed as the flat (also known as horizontal) approach and the hierarchical (also known as vertical) approach (Ahmad & Mir, 2021). With both these approaches the workload of the control plane is shared with numerous controllers but the main difference between the approaches comes with the communication of the controllers. With a flat distributed controller approach there are no layers or hierarchy among the controllers. This means that each controller maintains its designed domain or network segment and coordinates with the other controllers by for example transmitting the current network situation. This is illustrated bellow in figure 3.

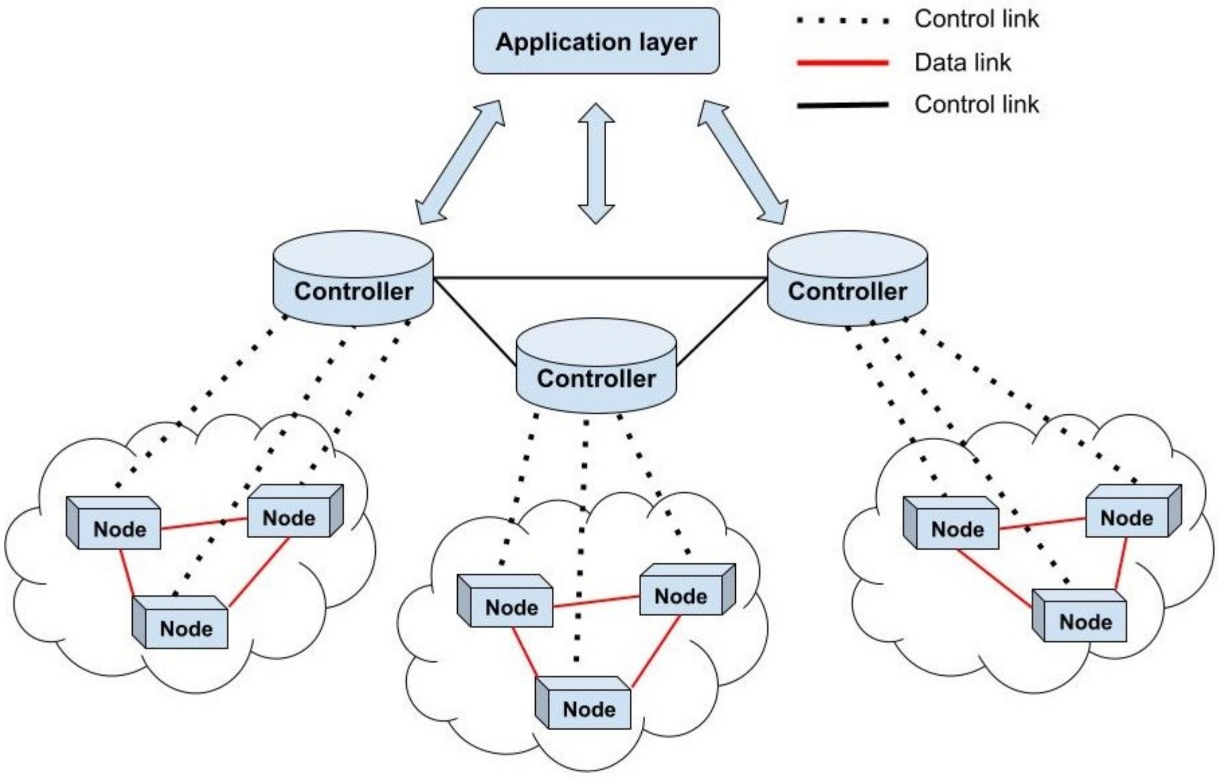

FIGURE 3 Distributed flat SDN architecture

Key benefits with the flat approach include cutting off bottlenecks and good resilience as the network segments can make use of other controllers in the case of failure (Ahmad & Mir, 2021). With a flat distribution model, a key concern is the ability to maintain a constant view of the topology and status of the network. As the controlling duties are shared among the controllers none of the controllers knows the status of the full network on its own. This poses requirements for the controllers to share information about the network between each other regularly and in a synchronized manner. This can cause issues in the form of network workload or if a controller is not working properly (Alsaeedi, Mohamad & Al-Roubaiey, 2019).

With the hierarchical distribution option, the control plane is implemented in a way where the control plane contains additional vertical layers. Essentially, with this approach the control plane is split into two or more layers containing controllers at different layers. These layers have a specific predetermined task that they are responsible for maintaining. For example, in a two-layer approach this could be structured in a way where the bottom layer controllers are responsible for maintaining their appropriated domains whereas the top layer controller is tasked with maintaining control of the entire network (Ahmad & Mir, 2021). This approach is illustrated in figure 4. With this approach the top layer controller only has to worry about guiding the bottom layer controllers and about preserving a view of the global network. In essence, this approach offers more centralized control as the controllers only need to maintain a connection to the top-level controller reducing the network traffic required. In addition, with the hierarchical approach the topology of the network is not fragmented between multiple devices as at least one entity is responsible for sustaining this feature.

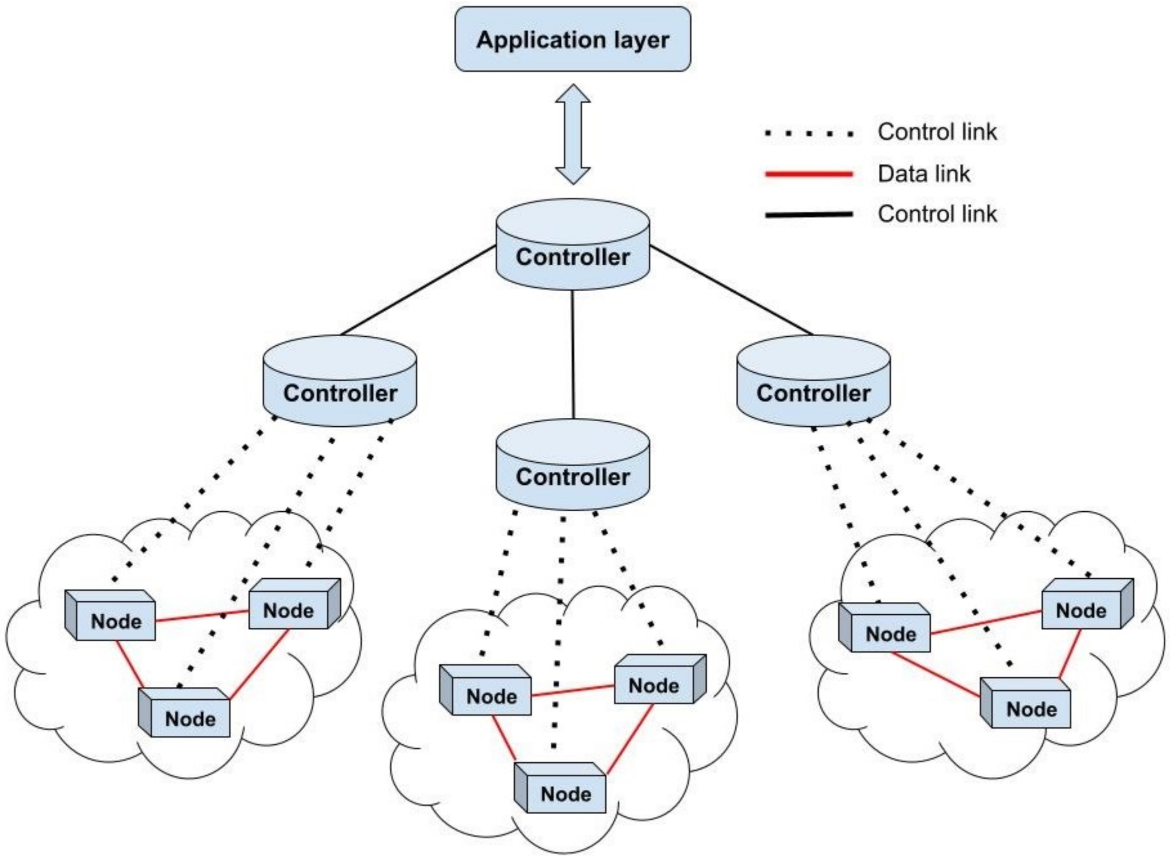

FIGURE 4 Distributed hierarchical SDN architecture

Although distributed SDN models contain numerous benefits and are applicable for medium and large-scale implementations they do also contain certain limitations. First, with the hierarchical model it is possible that the top-layer controllers form bottlenecks if they manage too many controllers at once. With the flat model network administrators on the other hand lose a portion of centralized control as the tasks are divided among many devices. A key issue can also arise with the controller's interoperability in case a variety of different controllers are deployed (Ahmad & Mir, 2021; Bannour ym., 2017).

2.2.3 Hybrid architecture

The third architectural group presented in existing literature are the hybrid SDN models. With hybrid implementations a key aspect in the model is integrating two different networking paradigms together in order to form a cohesive network. Basically, the network is designed to implement and run a SDN-based approach but a legacy system running on traditional networking methodologies is also incorporated through either the eastbound or westbound interface. A key challenge regarding the hybrid model is making sure that not only can the heterogenic systems co-exist but also communicate with each other in a meaningful manner (Sinha & Haribabu, 2017). The level and the depth of the interoperability can vary based on design but simply put hybrid systems can require interoperability with the control plane, data plane, or with both the control and data plane based on the design of the system (Sinha & Haribabu, 2017). As such the limitations and technical means to provide these functionalities vary. A simplified version of hybrid SDN is presented in figure 5.

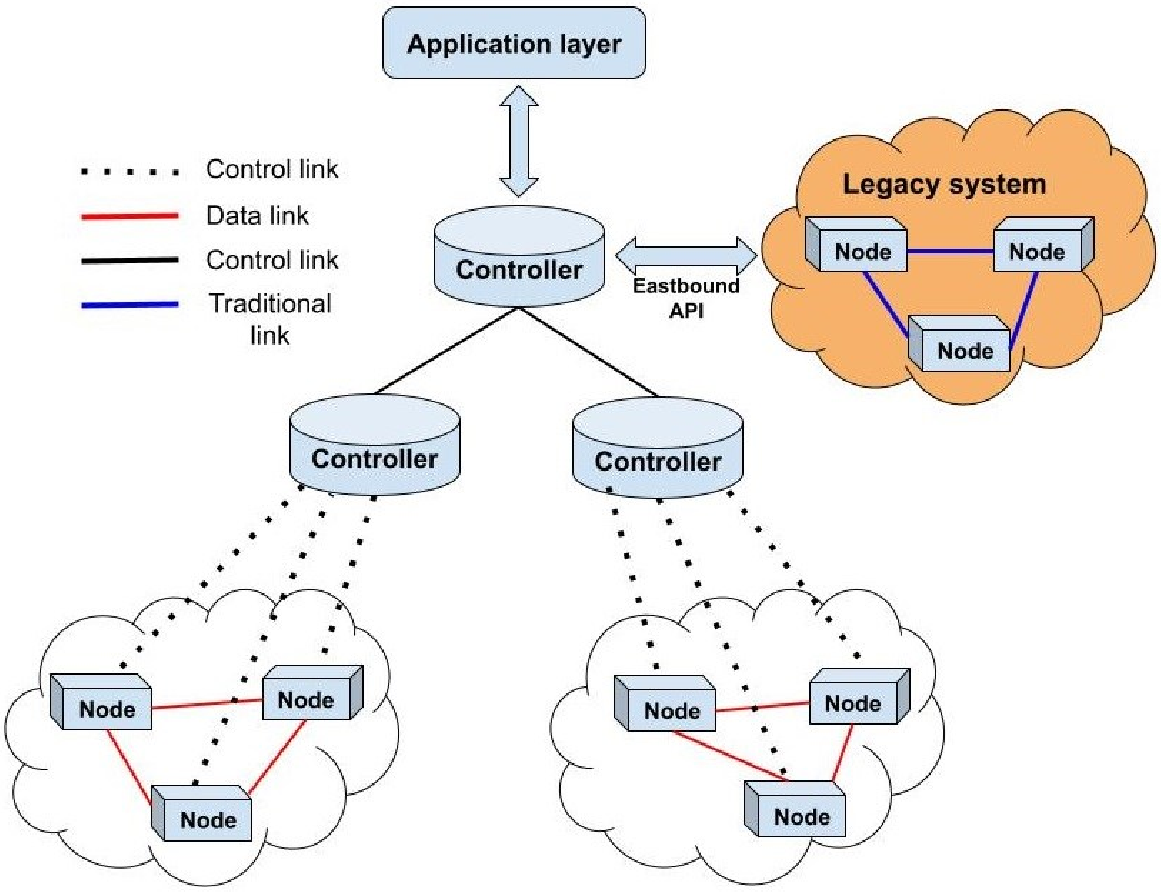

FIGURE 5 Hybrid SDN architecture

Whilst a hybrid SDN network can be most difficult to implement initially due to interoperability requirements, the model does offer benefits to the network administrators. Hybrid SDN models make use of existing traditional networks instead of completely disregarding them. Traditional networks are typically reliable as they have been developed for decades and as such provide stability and reliability to the model (Ahmad & Mir, 2021). Secondly, a hybrid SDN solution can be a cost-effective option in shifting to a new updated networking model. In many cases swapping entirely from a traditional network to a SDN network can be expensive and require new skilled labour (Amin, Reisslein & Shah, 2018). By implementing SDN gradually these requirements can be mitigated lowering the threshold of switching.

Furthermore, hybrid SDN systems can seek to avoid typical transition problems such as unintended downtime by gradually shifting and booting up new systems since the original systems are still left operational (Sinha & Haribabu, 2017). Finally, hybrid models also provide options for network administrators who only need SDN properties for small segments of the network. SDN operations can be incorporated into small segments of the networking domain where some of the paradigms benefits are required without a major transition on networking systems and with reasonable and justifiable costs (Ahmad & Mir, 2021).

2.3 Benefits of SDN paradigm

As stated previously, the new shift in networking paradigm brought up by the software-defined networking approach offers several tempting advantages for both the implementation and management of networks. The most prominent feature of SDN is the separation of the control and data planes which provides network managers with numerous benefits in regard to managing and setting up the networks. The most important benefit of this separation is the centralization of network controls. Unlike with the traditional networking approach networks can be controlled and thereby, managed through a single network node. This centralized control provides managers with more efficient tools to implement their tasks as the configurations and orders are passed to all devices from a single source and are implemented as such to all data forwarding nodes. In addition to pure efficiency, this feature also provides managers with benefits to act dynamically to events within the networks. For example, in the case of cyber-attack, network managers can simply implement routing rules to circumvent a potentially effected route. Secondly, network managers can also optimize the routing in terms of load balancing. In case certain segments of the network are being overloaded with traffic, network managers can simply establish new rules that guide the traffic to new desired routes.

Another advantage brought forward with the separated planes comes in the form of computational requirements of networking devices. With the traditional networking approach each routing device must be equipped with the capability of handling both the routing related traffic as well as the data. As such these devices require sufficient computational power and memory to perform these acts (Haji ym., 2021). With the SDN approach these requirements can be reduced as the infrastructure layer nodes do not require intelligence regarding the routing decisions. As such the burden placed upon the infrastructure layers CPUs and memory can be reduced by delegating the routing intelligence to the controllers (Haji ym., 2021).

This is especially advantageous regarding IoT devices and other low-power network nodes. A key issue regarding IoT is the lack of computational power with devices and the lack of memory, which can limit the feasibility of applying these devices. For example, in many cases the traditional TCP/IP structure causes issues for these nodes as the framework and its packet sizes are not optimal for these low powered devices (Shang ym., 2016).

Outside of data trafficking and control commands a key feature provided by the SD-network is the often built-in topology manager module alongside a graphical user interfaces (GUI) or a web-based interfaces developed for several controllers (Zhu ym., 2021). SDN controllers traditionally maintain the ability to map out the current state and topology of the running network which is used for routing decision but also can be forwarded as a feature for network managers (Zhu ym., 2021). The global topology can be forwarded to the application plane through the northbound interface and be presented through a GUI. This provides network managers with real time information regarding the state of the network in an often easier to comprehend format as opposed to simply viewing router information through terminals or command line interfaces as is done with most configurations (Bwalya & Zimba, 2021). Outside of simply being in a visual format, this feature also helps to maintain a complete view of the network. With traditional networking approaches routers typically do not know the status of the network outside its own routes and as such cannot provide network managers information outside their own routing information, or the information provided by its links.

The second prominent advantage provided by the SDN approach is the emphasis on software over hardware. With the traditional networking approach, the majority of networking functionalities are implemented through hardware whereas with SDN features can be run through programs. Favouring software over hardware comes with several benefits of which the flexibility of the network is essential. With traditional networking the needs of the hardware dictate to a large extend how the network must be configured and implemented. Because of this in many cases the chosen devices pose vendor specific requirements for the commands and policies used to manage the network causing limitations and potentially incompatibilities with other devices (Chica ym., 2020). However, with SDN chosen software handles the configurations and management reducing these dependencies. This also enables the developers of systems to opt out for higher level programming languages which can ease development and provide more options for choosing the language used to code the software (Chica ym., 2020).

Outside of simply managing the network, the software-based approach also provides the option to further develop the networks with new innovations or extensions as they can be simply coded and integrated into the existing system. As software is not tied to the physical hardware, developers can develop new modules or functionalities to be integrated within the application plane (Haji ym, 2021). As a result of this programmability SD-networks holster various different options for all of the architectures elements and provides network managers with numerous options when creating the network structures. For example, the final shape of the network can vary greatly based on the chosen controllers, interfaces, or architecture models. Since these variants contain different advantages, networks can be tailored to specific needs of organizations or use cases. This customizing can provide increased efficiency on predetermined segments of the network as well as better focus on using networking resources.

In addition to developing new features, the programmability and software-based approach also provides network managers advantages in regard to automating and making use of scripts. With traditional networking the routers communicate with each other and update their routing tables automatically, but the actual configurations needed for each device are typically done manually (Karakus & Duressi, 2017; Bwalya & Zimba, 2021). As such, this process can be time consuming and error prone as the configuration settings are prone to human errors such as typos or other mistakes that effect the actual routing process (Bwalya & Zimba, 2021). With SDN these processes can be automated through code or other scripts. In essence, setting up a new device to the network is easier as the required settings for the flow tables and other configurations can be run through code, allowing for faster setup times and reducing the workload of network administrators (Bannour ym., 2017).

With the automated configuration capabilities also comes benefits regarding the scalability of the system. With old networking paradigms scalability is fundamentally achieved through increasing the amount physical hardware that can serve users. In order to achieve this increased scaling organizations must implement expensive devices as well as deploy manual labour to configure the nodes (Karakus & Duressi, 2017). This can be both costly and time-consuming which may cause issues when dynamic scaling features are needed. With SDN these issues can be circumvented as the setup of a new router can be done rapidly through an automated configuration script. In addition, the problems regarding the cost of new devices can be mitigated by making use of virtual devices hosted within network (Chica ym., 2020; Rana, Dhondiyal & Chamoli, 2019; Karakus & Duressi, 2017).

Finally, SDN also offers network managers benefits in terms of security as the fundamentally different approach to routing and data trafficking in many cases mitigates well-known risks and attack vectors. For example, SDN-based networks allow for intricate traffic monitoring which can be analysed to find suspicious activities (Chica ym., 2020). SDN also provides more options for implementing security software such as IDS systems to different layers where these programs can monitor traffic flows directly (Chica ym., 2020). However, it is important to acknowledge that whilst SDN provides security enhancement features through its concept, it also generates new threat vectors which need to be taken into account.

2.4 Performance-based issues of SDN

While SDN-based solutions provide network managers with several benefits they also generate new concerns and issues which must be addressed in order to make the network paradigm shift feasible. Existing literature highlights these issues often focusing on three distinctive elements limiting the applicability of

SDN. These issues include the scalability, reliability, and security of SDN (Bannour ym., 2017). Security questions regarding SDN pose major potential issues and are present within all the layers of SDN. As such, these security concerns have been a major focus point of researchers. Therefore, this thesis addresses these issues on their own in more detail in chapter 5.

Scalability of the network is feature which represents a networks ability to handle situations where the networks traffic reduces or increases dynamically. This is more prevalent usually in cases where the network sees a rapid increase in traffic. With traditional networks these issues typically arise when the existing group of devices is unable to handle the influx of data packets leading to a need for more devices, which as previously presented is slow to setup. Software-defined networks are able to bypass these issues as the deployment of new networking devices can be done more efficiently through software. However, the issue of scalability is not completely solved as fundamental root cause of is not addressed.

With SDN these questions regarding scalability are usually directed at controllers. This is a direct result of SDNs centralized network control logic. In essence, if the controller is not able to handle the new networking devices and their traffic, the whole network will become unusable as a result of the controller's overload (Bannour ym., 2017). In order to solve issues regarding the scalability network managers must implement various solutions to ensure SDNs feasibility to large-scale solutions. This can be done through architectural solutions or by opting to use controllers that have specifically been designed and built to provide high performance (Shirmarz & Ghaffari, 2020; Haji ym., 2021; Ahmad & Mir, 2021).

In addition to scalability another concern regarding SDN is the reliability of the networks. With traditional networks one key advantage is the fact that the networking structures are reliable due to years of development and fine tuning (Ahmad & Mir, 2021). However, with SDN reliability can become an issue in many ways. For example, in use cases where network nodes are geographically sparsely populated, the distance between the network nodes and the location of controller can cause issues regarding latency. In cases where the latency becomes too high this can cause issues regarding the timeliness of routing rules which in turn causes limitations regarding the updating of flow rules leading to issues regarding the reliability and scalability of the network (Karakus & Duressi, 2017).

Outside geographical issues another key factor regarding the reliability of the network is formed by the controllers design and its ability to handle and tolerate unexpected failures. In certain cases, the design of the controllers has been built around increasing the controller performance thus increasing its scalability but at the expense of ensuring reliability and security (Ahmad & Mir, 2021). As such, issues can arise from incorrect packet header information which can cause controllers to malfunction leading errors or even crashes (Ahmad & Mir, 2021). In these cases, the networks reliability is compromised which leads to issues when maintaining a resilient and fault-tolerant network is a necessity.

Another issue regarding SDN can also arise when using distributed SDN architectures, which alleviate a single controller's workload. With distributed SDN models various controllers are deployed simultaneously providing resilience, more computational power and as such more scalability. However, these controllers must communicate and operate in a synchronized manner in order to fulfil the required tasks. This communication taking place in the control plane can cause issues for both the scalability and reliability. Depending on the amount controllers deployed the communication traffic needed to coordinate the controllers can limit the networks ability to serve the data plane and as a result reduce the amount traffic that can be forwarded (Karakus & Duressi, 2017; Ahmad & Mir, 2021). Furthermore, with distributed models the synchronization requirement can cause reliability and resilience issues in case the controllers are not able to efficiently communicate with each other, regarding the current status of their respected network segments (Benzekki, El Fergougui & Elbelrhiti Elalaoui, 2016).

Finally, performance related problems can also arrive from SDN applications making use of multiple different controllers or through integrating legacy systems. With SDN implementations an advantage is the fact that the networking devices are typically not vendor specific as they can be programmed based on the needs of the network. However, with this flexibility the cost is the lack of standardization which can cause issues with interoperability with devices limiting the options of controllers that can be utilized (Bannour ym., 2017). For example, if a network manager wants to implement a distributed model using both performance and reliability optimized controllers, they might have to developed software that can ensure mutual communications. Whilst this is not necessarily a technical challenge, the issue arises from the increased need for the controller to execute these codes leading more workload on the controllers (Karakus & Duressi, 2017).

Similar interoperability related issues can arise from deploying a hybrid SDN model. Like with different controllers the mutual communication with SDN and legacy systems needs to be established through dedicated software or interfaces. Based on the complexity of the hybrid format this can pose additional workload for the controllers involved in the case that they must also configure and control the legacy systems (Ahmad & Mir, 2021). This can lead to scalability issues if the controllers need to dedicate copious amount of computational power to these tasks. It can also lead to reliability issues in the case that the functioning SDN network portion and legacy domain are not synchronized in an adequate manner (Ahmad & Mir, 2021).

3 Current state of available SDN technology

This chapter presents the current state of available SDN technology and showcases potential solutions for implementing SD-networks. The chapter focuses on layers individually as well as showcases the protocols which can be used to implement APIs. Applications are approached from a more generic view as they are more versatile and can be carried out in a more ad hoc basis. Emphasis is placed on the controllers as they are the most prominent technology addressed in existing literature. Finally, individual protocols are showcased particularly for implementing the southbound interface as these protocols are SDN specific and primarily developed with the paradigm in mind.

3.1 Applications

With the development and wider adaptation of the concept of software-defined networking, applications and services have also increased. Originally SDN was largely based on open-source solutions which offered users the chance to introduce the technology freely into action, however today SDN-based solutions are also offered commercially by both major industrial organizations as well as by smaller companies. Because of the versatile nature of SDN it can be applied to many concrete use cases and therefore, pose different needs for different implementations. As such the chosen or required programs can typically be chosen on an ad hoc basis from either commercial or open-source avenues. They can also be developed by the users with adequate programming and networking knowledge, which enables them to be refined to perfectly fit the needs of the solution.

Commercial applications are offered to users in both large bundles and through smaller service packages. Several networking industry leaders such as Cisco and Juniper develop and offer SDN-based networking solutions that host by default several tools and applications that provide users with both security and management features. An example of a commercial SDN solution is Ciscos

Meraki SD-WAN, which offers an SDN-based approach to managing Wireless Area Networks (WAN). This approach includes several security features such as Cisco Snort IDS, Cisco Umbrella SIG cloud security, and Cisco AMP end-point threat Grid, which are integrated into the network (Cisco SD-WAN, 2023). In addition to the security software SD-WAN contains business and management tools such as programs dedicated to monitoring the status and health of the network, VoIP and web-applications (Cisco, 2023; Cisco SD-WAN, 2023). A key emphasis has also been placed in making sure that the information and functionalities provided by these solutions is presented in a visual manner through dashboards and other graphical approaches.

Another commercial bundle solution is offered by Juniper Contrail Networking which offers users a versatile cloud-based hybrid-SDN network. In this solution emphasis is placed in making use of virtualization technology as well as existing cloud technologies such as Kubernetes and OpenStack. Junipers SDN bundle also includes various applications to ensure security such as regular and virtual firewalls, network traffic monitoring, and other policy-based filters and rules (Juniper, 2023). Furthermore, additional security is offered through Junipers Cloud Workload Protection shields, which are applications intended to manage zero-day threats that cannot be picked up by traditional signature-based systems (Juniper, 2023).

In addition to security, Junipers offered applications include features designed to visualize the status of the network such as dashboards and software management tools. Graphical tools can also include debugging tools that can be integrated to provide tools for solving issues or bugs that may arise within the network. Finally, this approach also provides embedded routing services such as BGP as a service, which improves the routing through reducing the costs and complexity of routing through regular means (Juniper, 2023).

Naturally, the aforementioned SDN bundles provide network managers with a lot of prebuilt tools which can be integrated and put into action rather quickly. This reduces the need for inhouse development and also provides reliability through tested software. However, it is worth noting that these solutions can be expensive and include unnecessary features for smaller networking implementations. Furthermore, as the applications are prebuilt this reduces the flexibility of the tools as they are provided by the vendors.

Alongside commercial bundles SDN applications can also be acquired in smaller quantities by simply purchasing individual software designs to perform specific actions. These can include for example security features that are designed to mitigate a prominent attack vector. DefenceFlow is security program developed and offered by Radware and which is designed to mitigate distributed denial of services (DDoS) attacks against cloud and SDN networks (Radware, 2023). DefenceFlow is designed to provide automated and centralized protection against DDoS attacks through a single management point and by providing automated responses to noticed anomalies (Radware, 2023).

Another venue for acquiring SDN software is by making use of the open-source projects still developed to date. Many controllers are developed through a crowdsourced effort and include modules and applications coded by other users which are available in public repositories. For example, projects such as Floodlight, OpenDayLight and Ryu are all open-source developments with existing free to use applications. These applications include firewalls, topology managers, authentication and access controls and IDS prevention systems such as SNORT to name a few (Ryu, 2023; OpenDayLight, 2023). These applications can be used to secure and manage networks without purchasing commercial solutions.

The aforementioned projects also provide users with prebuilt ways of further developing personalized apps. For example, Ryu, a Python-based SDN solution offers an application manager which can be used to manage newly developed features (Ryu, 2023). A similar approach is offered by the Java-based OpenDayLight which enables further developing software and integrating it through the use of Maven-repositories (OpenDayLight, 2023).

Naturally, it is worth noting that open-source implementations do maintain their issues. Firstly, whilst open-source projects consist of freely available repositories there is no guarantee of the quality and reliability of the applications. Whilst the repositories do attempt to prevent malicious or harmful code from being deposited, occasionally malware is stored to the list of applications. These supply-chain attacks can cause issues for the security of the network. Furthermore, these applications may require more technical capabilities in order to be booted and configured initially to the system as there is no commercial expert teams to help with this process. Finally, keeping the code up-to-date can be an issue as security requires constant progression to be achieved. In the case of open-source code, if the development of the code is halted, then it is up to the users to maintain and fix issues that can arise from for example zero-day vulnerabilities.

3.2 Controllers

Controllers are the key element which enable the SDN paradigms feasibility which is why they have received a lot of attention among scholars. Existing literature presents several different versions of controllers that are more often than not tailored to provide features to distinct SDN architectures or to resolve existing challenges. Because existing controllers exist in double digit numbers it is not feasible to address all of them in this thesis. Therefore, this thesis will showcase controller options for each of the aforementioned SDN architecture types to provide a general idea of the popular available options and their general approach. However, for more comprehensive listings this thesis recommends papers by Ahmad & Mir (2021) and Zhu ym., (2021).

Centralized SDN implementations are often recommended for smaller scale networks such as small WANs for university campuses or other small networks, which is why the existing controllers often lack computational power (Ahmad & Mir, 2021). One of the first variants of centralized controllers is the

NOX controller initially showcased in the late 2000s. Whilst still a potential solution for small networks, NOX has also served as a steppingstone for newer more powerful and better suited solutions, which have been developed based on the previous features of NOX. These controllers include the more powerful NOX-MT, the developer-oriented POX and NOX-Verity (Ahmad & Mir, 2021; Zhu ym., 2021).

In addition to the NOX-based controllers existing literature also provides several centralized open-source solutions such as Ryu, Floodlight, and Maestro (Bannour ym., 2017). These controllers are developed mainly through the use Java with the exception being the Python-based Ryu. Whilst these controllers are still applicable and usable, they lack the required features for large-scale commercial applications, which is why they are often suggested to be used for academic purposes or other testing (Ahmad & Mir, 2021). However, due to their older nature, existing documentation (Zhu ym., 2021) and relatively simple design they can also be used as an option for small scale networks, which require further development.

Newer and more unique options for centralized controllers represent controllers such as ParaFlow (Song, Liu, Liu & Qian, 2017) and Ravel (Wang ym., 2016). ParaFlow is a controller presented in 2017 with the idea of providing a highly scalable and efficient centralized controller which could be deployed to manage large-scale networks. The primary idea behind ParaFlow is making use of multiple cores that can be utilized more optimally by making use of fine-grained parallelism with both the controller event handlers and task handlers (Song ym., 2017). Ravel on the other hand is a controller which seeks to integrate the SDN paradigm with databases and in particular SQL language. In essence Ravel is incorporated into a database which provides SDN data through SQL (Wang ym., 2016). The main benefits suggested by this approach are received from having a high-level and well-known language, SQL, which is used to control and further develop the controller (Wang ym., 2016).

For distributed SDN architectures network managers can choose from a variety of options based on the structure of control plane. Overall, existing research suggests that the flat distribution seems to be the more popular option in terms of controllers as it hosts considerably more variety. Two existing options include the Onix controller and Onos controller, which make use of instances to provide cohesive distributed control (Ahmad & Mir, 2021). With Onix the general idea is to create multiple instances of Onix controllers, which share information between each to maintain the global view of the network (Ahmad & Mir, 2021; Koponen ym., 2010). The use of instances enables Onix to be run on a single physical device with several instanced controllers providing scalability to the network (Koponen ym, 2010).

Onos is another instance-based controller designed to carry out distributed SDN networks. Unlike Onix, Onos is a fully open-source approach making it a more applicable and feasible alternative for network managers to implement (Zhu ym., 2021). Whilst Onos also follows an instanced model, it differs from Onix since Onos itself makes use of a centralized controller Floodlight to perform the actual SDN functions. The Onos controller deploys several centralized Floodlight controllers that it then connects to provide the required distribution architecture (Berde ym., 2014).

An alternative to the aforementioned controllers is the HyperFlow controller, which offers distribution and scalability through running several centralized controllers. The design of HyperFlow is built upon enhancing the NOX controller by utilizing the existing design on NOX and by adding the application features of HyperFlow (Tootoonchian & Ganjali, 2010). In essence, each of the controllers sees the global network status reducing the need for lateral communications, but the actual management is divided by HyperFlow. HyperFlow is also responsible for maintaining the required information for each of the physical controllers (Tootoonchian & Ganjali, 2010).

As for the hierarchical distribution approach popular controllers include Kandoo and Orion. Kandoo is controller suggested initially in 2012 that creates a two-level control plane to provide hierarchical control. The idea behind Kandoo is quite typical for hierarchical control as the controllers are split into local and root controllers (Hassas Yeganeh & Ganjali, 2012). Local controllers are designed to manage and handle request that are frequent whereas unusual or rarer instances are passed forward to the root controllers (Hassas Yeganeh & Ganjali, 2012). With Kandoo all of the controllers are the same, meaning that all of the controllers can act as either the root or local controllers. To determine which layer needs to make the decision is done through an extra flag among to data packets (Hassas Yeganeh & Ganjali, 2012).

Orion controllers provide large versatility as they can actually form both a flat and hierarchical control plane. Unlike Kandoo, Orion controller deploys a three-layer control plane consisting of the bottom, middle, and upper layers (Ahmad & Mir, 2021). With Orion the bottom layer consists of typical networking nodes forming a sub-domain, which is controlled by the middle layer consisting of controllers with a view of the global network (Fu ym., 2014). Finally, the upper layer consists of another set of controllers that parse together information passed by the middle layer (Fu ym., 2014).

Finally, as the last showcased distributed controller acts the controller Beehive. Beehive is hierarchical controller model initially presented in 2014. Beehive is developed with the Go programming language and consists of two key components being the programming model and the control platform (Yeganeh & Ganjali, 2016). Beehive seeks to provide network managers with controller that is both simple to program as well as a controller that produces good results in terms of efficiency.

With hybrid SDN controllers' availability is somewhat more limited in comparison to other architectural approaches, but nonetheless existing literature provides options for network implementations. Older variants of hybrid controllers include RouteFlow and HybNET presented in the early 2010s at the pioneering stages of the SDN paradigm. RouteFlow is a hybrid architecture controller making use of OpenFlow and virtualization, by deploying a RouteFlow server dedicated to maintaining a virtual routing environment

(Nascimento ym., 2011). The basic idea behind RouteFlow is that the application plane of the model contains the virtual routers hosted by RouteFlows server and the northbound interface is established through the same server (Nascimento ym., 2011). The RouteFlow server communicates with controllers in the control plane whilst the physical infrastructure plane runs on older legacy networking protocols such as BGP and OSPF (Nascimento ym., 2011).

Another older solution is HybNET which unlike RouteFlow is built upon a specifically designed controller to ensure interoperability between SDN and legacy nodes. With HybNET networks are built with at least one pair of a regular SDN controller supporting OpenFlow and with a specific HybNET controller (Lu ym., 2013). The first step in the networking process is done by the HybNET controller which collects the networking request from both the legacy and SDN nodes. The HybNET controller then parses all of the requests to the required format and then passes the requests to the actual controller which then makes the required decisions regarding the network functionality (Lu ym., 2013). This approach enables both the legacy and SDN nodes to be hosted and run within a single network domain.

Lastly, a newer option for creating hybrid SDN networks is the HybridFlow controller. HybridFlow is a controller that makes use of two popular routing protocols OpenFlow and OSPF and existing network nodes that are equipped to be able to run both routing protocols simultaneously (Guo ym., 2021). HybridFlow controller is built upon the idea that in many cases legacy networking solutions such OSPF are sufficient in managing the network and can as such be used. However, in cases where more dynamic and specific networking is needed HybridFlow also provides the ability to calculate special crucial routes with the use of SDN (Guo ym., 2021). With this approach much of the networking can be achieved making use of traditional methods and the advantages and abilities of SDN are tapped into when required.

Due to SDNs development and increased attention among scholars, also the amount of controller options has also increased rapidly. As such network managers have a wide range of implementations to choose from. However, it is also worth noting that although several controllers are available through open-source channels, there are also several factors which effect the applicability of individual controllers. Firstly, different controllers pose limitations on operating systems that can be used by the controller. Most of the controllers' support Linux and Windows-based systems, but in particular with MacOs the options are more limited (Ahmad & Mir, 2021). As such when choosing potential solutions this can prove to be a contributing factor.

Secondly, the built-in functionalities and API support varies between controllers. For example, most of the controllers are designed to support using the de facto southbound API OpenFlow, but there is a lot of variation between which versions are supported. The previously addressed POX controller only offers inbuilt support for OpenFlows 1.0 version whereas Ryu comes with support for any of the versions between 1.0 -1.5 (Ahmad & Mir, 2021). Naturally, these issues can be circumvented with further development, but this is something that should be noted in case ready to go solutions are desired.

Continuing with further development another factor that should be considered is the chosen coding language. A large portion of the available controllers have been developed through either Java or Python, with Java being the more popular language (Ahmad & Mir, 2021). If further development and customization is expected this should be noted when choosing the controllers to ensure the required knowhow.

Finally, a factor that may be important to consider is the amount of documentation of the controllers. Existing literature highlights that although several of the controllers are freely available through open-source code, the documentation is insufficient or even lacking (Zhu ym., 2021). As such if the paradigm of SDN is unfamiliar or the chosen controller is unfamiliar, adequate documentation may be significant factor in the choosing process.

3.3 Protocols

As SDN consists of three layers different protocols are required to connect each layer and possibly lateral movement through EB or WB interfaces. Since there are no mandated solutions and the paradigm relies on software, network administrators have a wide range of options which can be utilized. The northbound interface is often suggested to be implemented through more traditional protocols such as REST, a version of JSON or NetConf. These are not SDN specific and will therefore not be addressed in more technical detail in this thesis.

On the contrary the SBI is often suggested to be implemented by protocols which are specifically designed for the architecture and are such unique to the technology stack. The most well-known and often the de facto standard protocol for implementing SBIs is a version OpenFlow. However, despite its popularity it is not the only available option as other protocols are also developed by reputable and significant actors in the networking field. Commercial organizations such as Cisco have developed their own competing communication protocol called LISP that is deployed for the SBI. Other options include ForCes which has been created by the Internet Engineering Task Force (IETF).

The aforementioned will be addressed in more detail below to showcase their key characteristics. Naturally, this is by no means an exhaustive listing as new protocols can be created and deployed when necessary, but the main goal is to highlight options from currently existing protocols produced by major contributors and which are already applicable to networks.

3.3.1 OpenFlow

OpenFlow is the most widely applied SBI protocol and is originally developed and still maintained by the Open Networking Foundation. An OpenFlow switch consists of two key elements which are the control channel and flow tables. Each device contains at least one flow table and group table which are used to store cached information about previously issued flow rules that are delivered by the controller through the control channel (Open Networking Foundation, 2015). These rules are stored to flow tables as flow entries. Each flow entry is further broken down into smaller metadata components which are match fields, counters, instructions, priority, timeouts, cookies, and flags (Open Networking Foundation, 2015).

When an OpenFlow switch receives a flow entry it begins a matching process where it compares the newly received flow matching information to previously cached entries. If the device finds a flow entry where all the match fields match the flow, a new query is performed to the instructions component which tells the devices what actions should performed on the packets (Open Networking Foundation, 2015). For example, when a matching flow rule is met the instructions may direct the switch on where to forward the data packets or how to modify them. Conversely, packets may also be dropped if they are not allowed to traverse to a certain location or if none of the flow tables contain a matching entry.

Because SD-networks are designed to be dynamic it is expected that packets that do not match the flow table will be detected. As such OpenFlow devices typically contain a so-called table-miss entry. When a switch receives a packet that it cannot process it can send a Packet-IN message to the controller via the table-miss entry to ask for guidelines on what actions to perform to the packets (Open Networking Foundation, 2015). The controller will send the switch either a Packet-OUT response or new rules for a flow entry through Flow-MOD packets which tells which actions to apply to packets (Open Networking Foundation, 2015).

Flow-MOD packets can also be applied to generally add, modify, group, or delete existing flows to cause changes to the network and how data is being forwarded (Open Networking Foundation, 2015). For example, each flow has a dedicated priority field which contains a predetermined value. This priority dictates which flow entry is used if there are multiple matches. A network manager can alter the routing by sending a Flow-MOD packet which changes these priority values, thereby also changing which entry is used.

3.3.2 ForCES

ForCES (Forwarding and Control Element Separation) is an older protocol which predates SDN by over a decade. The protocol is originally developed and conceptualized by the IEFT in late 2003 as an early conceptualization of separating packet forwarding and network controls. The protocol is actually an element in a larger framework called the ForCES framework which is intended to act as a unifying force for the forwarding elements and control elements enabling them to communicate via a standardized protocol (Haleplidis ym., 2015). ForCES protocols role as standardized communication format enables the framework to be more dynamic by allowing a wider range of customization for the forwarding elements (Haleplidis ym., 2015).

Typical to the SDN paradigm ForCES framework adopts a format where the forwarding elements which are responsible for packets transmission can asks the controlling elements on guidance in case, they are unable to make the routing decision on their own (Haleplidis ym., 2015). Although the framework differs from traditional networking with the separation features, the control elements can still make use of traditional routing algorithms such as RIPE or BGP which provide a routing decision that is then relayed by the ForCES protocol (Haleplidis ym., 2015).

The ForCES protocol actually consists of two protocol layers which have their set of requirements and roles (Haleplidis ym., 2015). These are the protocol layer and the transport layer. As the name suggests the transport layer is responsible for the actual transmission of the communication between the control and forwarding elements. The chosen protocol for this task is Stream Control Transmission Protocol (SCTP) (Haleplidis ym., 2015). The protocol layer is implemented using ForCES (Haleplidis ym., 2015). Much like traditional tcp-connections before transmitting data, a connection needs to be established between the control element and forwarding element (Haleplidis ym., 2015). After this connection has been achieved control messages can be transmitted.

ForCES protocol specifications are designed to be simple to ensure that the protocol is viable regardless of the forwarding element designs. The protocol contains ten different message types of which three are for setting up the initial SCTP connection, one being a heartbeat message, and the last six being control messages (Haleplidis ym., 2015). These control message types are config and config response messages, queries and query responses, event notifications, and packet redirection messages (Haleplidis ym., 2015). The message types also have predefined operations that include functions such as get, set and delete which are directed at the targeted devices (Haleplidis ym., 2015). Because of the designs light nature, the protocol and its functionalities do not impose significant limitations for other elements of network.

3.3.3 LISP

LISP or Location Identifier Separation Protocol is a protocol that uses a decoupling design to perform its routing (Rodriguez-Natal ym., 2015). Unlike traditional SDN, LISP is designed to separate the location identifier and the hosts rather than control and data packets. LISP maintains a set of Endpoint identifiers (EID) and routing locators (RLOC) that enable the protocols functionality (Rodriguez-Natal ym., 2015). To connect various EID through the use of RLOCs LISP also maintains a mapping system which acts in a similar fashion as SDN controllers (Rodriguez-Natal ym., 2015).

A simplified description of the protocols functionality can be presented as such. Data needs to be transmitted from one EID to another but the router in charge of doing this does not know the routing locator needed. Rather than waiting for typical routing advertisements or using a default route the router sends a control query to the mapping system (Rodriguez-Natal ym., 2015). The mapping system as the name suggests has mapped out all of the RLOCs that connect the different EID and responds with the required RLOC (Rodriguez-Natal ym., 2015). To avoid causing unnecessary overhead communication the router then updates its routing cache to contain the information that it has received which in turn means that it can directly use this information for future transport (Rodriguez-Natal ym., 2015).

LISP contains different versions to its detailed implementation. The protocol has both open-source and commercial variations (Rodriguez-Natal ym., 2015). For example, Cisco has adopted the protocol and tailored it into their own specific needs to support their own operating system software (Cisco, 2024). The protocol is also developed by the IEFT and general guidelines for maintaining the protocol adaptability are maintained.

4 Applications for SDN technology

Chapter four offers a dive into currently existing or outlined concepts that integrate SD-networks as a solution for enabling technical architectures. This chapter presents four different technologies which utilize SDN to either improve existing technical solutions or as a solution to resolve identified issues ensuring the feasibility of the suggested frameworks. Naturally, the applications presented in this chapter are not an exhaustive listing of solutions for SDN, however the aim is to provide a general sense of the current state as well to showcase the wide range of possibilities regarding the future of this networking paradigm.

4.1 Internet of Things

A common use case for SDN technology is formed by networks making of use smart devices otherwise known as IoT devices. Internet of Things devices are often commonly used items such as watches, televisions, sensors, and other household appliances which are connected online. They are commonly designed to enhance the quality of these devices by offering digital features such as automatization, remote usability, and optimization. These devices maintain two distinctive features which are that the pool of devices is generally heterogenous and that the devices are often equipped with limited computational power and resources. Whilst these characteristics ensure that the IoT devices costs remain small and that the devices are widely available for a large number of use cases, they also create problems in particular for networking.

Since IoT devices lack computational power, they can often struggle with running traditional networking protocols leading to issues with routing optimization and efficiency. Furthermore, the heterogeneity of individual devices can cause issues in ensuring the connectivity of each device as traditional networks are typically rigid and company specific. Software-defined networks enable circumventing both of these issues due to the decoupling of routing and data transmission as well as having a less rigid programmable approach.

A survey conducted by Younus, Islam, Ali, Khan and Khan (2019) provides insights into smart home designs built around SD-networks. According to their survey smart homes have been an important and largely supported development for the future of cities and housing. However, despite the promising benefits that smart living accommodations can provide for the environment and residents, the quality of service in these solutions is often limited due to issues with traditional networking (Younus ym., 2019). The survey notes that smart homes primarily comprise of several individual smart devices which combine their efforts into creating a smart home.

Although the survey does not present a direct implementation of a smart home architecture the scholars note key features that are present in other bodies of research and which they present as fundamental advantages of SDN enabled smart home networks. In their perception of SDN smart homes Younus ym., (2019) approach the network structure through centralized network models where the communication through the SBI is handled through OpenFlow. Smart devices which act as the infrastructure layer nodes are connected to so called sink nodes which in turn communicate with the controller of the network. The connectivity between the sink nodes and IoT devices can be established through either WiFi, Bluetooth, Zigbee, and LiFi (Younus ym., 2019).

The generic model presented in the survey places importance on programmability of the control plane by suggesting the use of existing data analytic tools to gather information regarding the state of the network and its nodes. The authors suggest placing Python-based data analytics, API management and analysis tools on the centralized controller to provide efficient resource management and processing as well as to optimize the flow decision algorithms (Younus ym., 2019).

Finally based on their literature survey they present seven key benefits to smart homes enabled by SDN. These are more flexible network management, improved reliability, optimized bandwidth performance, improved quality of services, enhanced security, reduced latency, and improved energy efficiency (Younus ym., 2019). However, it should be noted that although these presented advantages are based on existing research, they are conditional regarding the implementation of the network. This means that the benefits might not be feasible based on purely the most bare bones architectural models but rather through making use of further developed technologies that SDN devices are able to maintain and utilize in their operations.

A second approach to incorporating SDN into IoT device routing is present by Rabet ym., (2022) in their paper regarding highly dynamic and mobile IoT networks. Highly dynamic and mobile IoT networks are networks where the connected devices are moved inside rapidly within the network causing sudden changes in the network topology. As showcased by Rabet ym., (2022) in their work, these networks have dedicated protocols which have been designed to service the needs of lower powered IoT device routing. However, these protocols excel in stable networks where the topology of the network remains mostly static and begin to suffer serious performance issues if the topology changes rapidly and continuously (Rabet ym., 2022). To address this Rabet ym., (2022) conceptualize and showcase their own protocol for mobile IoT networks called SDMob (Software Defined Mobility management) which through the use of a controller improves existing IoT protocols by offering a solution for rapidly changing topologies.

In their model Rabet ym., (2022) make use of all the layers present in SDN to implement existing technical solutions to cater to needs of a dynamic mobile IoT network. The infrastructure layer of the model consists of two different types of nodes which are dubbed mobile nodes and static nodes. As the names suggest, these nodes serve a different role in the layer to provide the controller required information of the network status. Static nodes perform the actions of a sink which acts as the mediator between the mobile nodes and controller allowing the controller to collect localization information (Rabet ym., 2022). The control plane consists of two pieces, a controller and a border router, which are hosted as Linux processes. The border router is used as processes which is responsible for parsing and processing the network traffic into a suitable format for the controller to handle. Finally, the application layer consists of several particle filters which are software designed to run existing algorithms used to predict the most likely future locations of the mobile nodes (Rabet ym., 2022). It is worth noting that the solution presented by Rabet ym., (2022) makes use of the flexibility of SDN as the interfaces are not implemented by the most common protocols such as OpenFlow but rather by existing solutions for IoT networks.

The flexibility of the paradigm is also present in additional features integrated into SDmob as several quality-of-service features can be implemented to mitigate known issues. These include collision avoidance which is used to mitigate unnecessary overhead caused by MAC transmissions caused by mobile nodes, stronger links caused by controllers preferring non-mobile static nodes, more rapid topology updates by having the ability to make use of downward links and using parallel filters and buffers to handle major increases in network traffic caused by rapid mobile node movement (Rabet ym., 2022).

Advantages regarding the model include making use of existing IoT network tools such as already defined algorithms in the application plane, allowing for a lightweight deployment of the architecture as the controller and the required border router are hosted as processes rather than as physical hardware, and finally by providing increased reliability and quality by circumventing known issues of common IoT routing protocols (Rabet ym., 2022). Identified issues include maintaining a centralized SDN model which can limit the number of mobile nodes that can be hosted in the network, as the controller may form a chokepoint if the congestion control mechanisms are overloaded by overhead traffic (Rabet ym., 2022).

Lastly, another SDN-based model for smart homes is presented by Gilani ym., (2024). The framework consists of a distributed hierarchical two-layer SDN implementation, where the network is built by using a parent-child connection. This approach is intended to provide depth to the framework whilst mitigating some of the issues faced by distributed models, such as synchronization requirements. In essence, the model consists of a primary SDN controller as well as a local SDN controller. The primary controller resides outside of the local area network and is responsible for dealing with external services and networking whilst the local SDN controller handles the LAN functions (Gilani ym., 2024). Furthermore, the framework also includes a switch running OpenFlow which is connected to a local controller as well as the network nodes, in this case primarily smart sensors (Gilani ym., 2024).

Whilst the local SDN controllers are intended to host less demanding computational actions, the primary controller is equipped with more resources and power by making use of cloud platforms (Gilani ym., 2024). This ensures that the network does not begin to experience reduced quality of service even in resource heavy functions.

The local controller is responsible for keeping an up-to-date view of the smart home network and the status of its devices. To achieve this the framework makes use of state request queries which are broadcasted regularly to services and nodes hosted in the network (Gilani ym., 2024). Based on the received response controllers update their topology to both account for offline devices as well as potential new nodes (Gilani ym., 2024). The framework also suggests automated responses and debugging for cases where issues are identified (Gilani ym., 2024). To manage the system Gilani ym., (2024) showcase a custom-made user interface containing a dashboard, settings, service summary, and statics (Gilani ym., 2024). This enables the end user to survey the status of the network from a centralized unit, showcases the collected and processed data in the form of statistics, and the ability to administer the network and services.

4.2 Network function virtualization

Another concept born from virtualization technology is network function virtualization (NFV) which as its name suggests performs various network functions through software. Much like virtual machines, virtualized network functions are performed on top of a virtualized layer that uses its hosts hardware resources (Li & Chen, 2015). Much like SDN, NFVs provide its user with more flexibility as the functions of the network are longer bound tightly to the device but are rather an implemented piece of software. SDN and NFV can be used to complement each other creating a network which is both controlled through software, and which has its functions virtualized through software (Li & Chen, 2015). This creates a highly flexible and automatable solution for network managers.

A framework of SDN enabled NFV is presented by Li & Chen (2015) who showcase a general architecture utilizing both technologies. Their solution consists of three components, a control module, forwarding devices, and a NFV platform, which are used to create the architecture (Li & Chen, 2015). The NFV platform consists of various pieces of hardware that are equipped with the capability to run hypervisors enabling virtual machines. These virtual machines run software-based network functions such as firewalls and proxies that enable the network to operate (Li & Chen, 2015). The NFV platform is also connected to the forwarding plane which hosts numerous forwarding devices that forward data through the network (Li & Chen, 2015).

To control the aforementioned pieces of the framework a control module is implemented consisting of a SDN controller and NFV orchestration mechanisms. Essentially, the orchestration system is responsible for controlling the network function while the controller is responsible for making network routing decisions (Li & Chen, 2015). This approach enables combining NFVs for multiple devices inside a single network as the orchestration system is able to tell the controller what needs to be done for the network to operate optimally whilst the controller is able calculate an optimal routing for these functions inside network (Li & Chen, 2015).

Another combination model of NFV and SDN is presented by Boubendir, Bertin & Simoni (2016) who showcase a framework for dynamic Network-as-a-Service (NaaS) models. Like other more traditional as-a-service models, NaaS seeks to provide its users with paid networking services that can be acquired from an external provider. In essence, a customer is able to purchase networking features such as routing and the required infrastructure from a NaaS-provider, enabling them to cut off their own in-house infrastructure and operations. Similar to Li & Chen (2015) Boubendir ym., (2016) unite virtualized NFV features with SDN networking in an attempt to provide highly dynamic NaaS services that can be tailored to a customers needs.

Boubendir ym., (2016) NaaS architecture consists of four key components which are the SDN layer, NFV layer, Network Exposition Layer and a user interface. The SDN layer is responsible for maintaining a view of the networks state and topology. Controllers manage network traffic inside the service infrastructure as well as chain together various network function services that a customer might choose to purchase. On top of the SDN layer resides the NFV layer which consists of predetermined network functions that the NaaS providers is willing to provide (Boubendir ym., 2016). These functions are run through virtualization which enables them to be dynamic, isolated, as well as easy to boot or shutdown if required. These virtualized network functions are connected through an interface to the network exposition layer which is responsible for managing the offered services as well as acquiring the requests from customers (Boubendir ym., 2016).

This layer consists of two key components which are the service description module and the service broker. The service broker is exposed to the customers and acts as the interface that services customer orders from a user interface (Boubendir ym., 2016). This module enables the customers to browse the offered and exposed services and to create requests which are passed on to the service description module (Boubendir ym., 2016). The service description module is implemented as a registry which holds information about the underlying services that are exposed and available to request.

As the NaaS architecture is required to be dynamic in nature, Boubendir ym., (2016) suggest that the model could be built in three different ways to provide the required services. The first option would be to host the NFV-services as a constantly running service which are either used or not. The infrastructure is constantly running the services meaning that they do not need to be booted up after a request is made by the customer. The second showcased option is an event-based model were by default no services are run but rather booted up after an event or a request is submitted (Boubendir ym., 2016). This approach leverages the dynamic and rapid nature of both virtualized solutions as well as SD-networks to provide on-demand services. Lastly Boubendir ym., (2016) suggest a customized deployment model. In this model standard features are offered through the constantly running model but for certain clients' customizable options are offered. In this model specialized deployments are not run by default but rather built and booted when required (Boubendir ym., 2016).

4.3 Cloud services and data centers

A key feature of many modern network solutions is making use of large-scale data centers and cloud services to setup the required networking infrastructure and platforms. Large-scale tech companies provide various services to satisfy the technical needs of the network by offering their networks, servers, and communication channels to customers, often removing the need to host in-house networking infrastructure. Data centers are often massive in terms of infrastructure in order to cater to the needs of potentially thousands of customers, creating a need for efficient network management and routing to ensure a sustainable upkeep as well as to keep to the quality of service high. Furthermore, on top of simply infrastructure and platforms, many cloud services also provide application solutions creating further requirements for flexible networks. As such, SDN can be viewed as lucrative option for managing cloud networks inside data centers due its flexible nature, centralized command features, as well as its complete view of the networks topology.

The first solution making use of SDN in cloud service implementations is presented by Conti, Kaliyar & Lal (2019). This solution dubbed CENSOR is a framework for heterogenous IoT networks that use cloud services for computational power as well as storage. CENSOR consists of all of the standard SDN layers, an application layer which hosts IoT applications that are used to control the sensors, the control layer which hosts several centralized controllers that also have an API to a cloud provider, and the infrastructure layer which has the switches, sink nodes, and IoT sensors (Conti ym., 2019). CENSOR makes use of OpenFlow as the SBI and utilizes OpenFlow switches which connect the sensors as well as the infrastructure plane and controller. These switches are also equipped with fog computing capabilities in order to provide computing power close the data sources, in this case the sensors, and to reduce unnecessary traffic on the SBI which would happen if all the heavy computing would be done through cloud services (Conti ym., 2019).

The framework places emphasis in data collection and utilization which is present through the fog computing capabilities, cloud APIs, and application tools residing on the application layer. CENSOR makes use of the cloud services by storing the required data inside a cloud, reducing the need of building storage capabilities inside the network (Conti ym., 2019). Furthermore, a key issue for many SDN implementations is the scalability and resilience of the network under large-scale data flows causing issues for many popular controllers. CENSOR attempts to circumvent this with fog computing on the infrastructure layer, reducing traffic on the controller and by also making use computing power inside cloud (Conti ym., 2019). This reduces the stress on the controllers as support can be acquired from bigger network providers and also increased scalability as many of the cloud services are quick to boot up and shutdown if there are significant changes in the networks traffic quantities.

On top of utilizing data collection and analysis, CENSOR also makes use of SDNs programmability in its security implementation by performing periodical multi-layer attestation procedures. In essence, all of the nodes residing in the infrastructure layer have so-called IoT agent modules installed inside the device. This agent is inspected periodically in accordance with the set time interval by a device placed higher in the architecture. In practice this means that IoT nodes are inspected by their local switches and these switches also hosting the agent module are checked by their respective controllers (Conti ym., 2019). To avoid foul play or other interference that an adversary might attempt, verification is calculated through hashing algorithms stored inside the read-only memory of the device. Furthermore, a secret input is stored inside the verifying device that is passed as a parameter for the hashing algorithm. After the software of the device has been hashed with the source code and the secret parameter, the formulated hash will be sent back to the verifier which compares the result to a previously stored valid hash (Conti ym., 2019). If these hashes match, the integrity of the software can be validated.

A second framework is presented by Cziva, Jouet, Stapleton, Tso & Pezaros (2016), who showcase a solution for more efficient management of virtual machine-based data centers. Their work makes use of an existing framework S-Core which is further developed to more accurately measure network loads and to offer quick links for flows to circumvent bottlenecks (Cziva ym., 2016). The framework consists of two key capabilities which are the migration features of virtual machines between independent hypervisors and SDNs efficient network control mechanisms that can be distributed rapidly to numerous devices.

The underlying feature of S-Core is an algorithm which is designed to calculate communication costs between data links inside a large network or data center. In essence, this algorithm calculates a cost for each data link and based on those figures derives an optimized suggestion on which hypervisor the target virtual machine should be transferred to ensure a better cost for the data link and to optimize the resource utilization (Cziva ym., 2016). Cziva ym., (2016) use this algorithm as a foundation for their framework, which extends the algorithm by incorporating SDN to mitigate known flaws such as a static topology view and required manual operations for moving the virtual machines.

The architecture follows a typical model for SD-networks. The application plane consists of key software that is used by the data centers management such as management applications for the machines. The controller layer consists of controllers that use key SDN modules such as a topology manager and link discovery to provide the S-Core algorithm an on-demand view of the network. Lastly, the infrastructure layer consists of the virtual machines, hypervisors above them, and software-based switches which connect these machines to the control plane. The NBI is implement using REST and the SBI through OpenFlow (Cziva ym., 2016).

The framework provides several benefits that are brought forward by SDN. Firstly, since the controller is required to maintain an up-to-date view of the networks topology, the required modules are already implemented in existing solutions. Furthermore, because the networks routing and control features are centralized individual virtual machines do not need to main a view of the network. This makes S-Core more dynamic as the topology changes can be updated from a single source rather than having to be implemented in each of the machines (Cziva ym., 2016). Secondly, when S-Core derives a result on a data link that warrants a VMs migration to another hypervisor this information can be propagated more efficiently through SDN flows to other devices (Cziva ym., 2016). Thirdly, the added advantage of SDN is also present through decoupling the routing and data transfer as the controller and application plane are tasked with running S-Core and propagating migration decisions (Cziva ym., 2016).

Finally, as the last cloud-based SDN solution presented in this thesis is a security model SeArch developed by Nguyen, Phan, Nguen, So-In, Baig and Sanguanpong (2019). SeArch is an architectural model intended to provide security to IoT networks utilizing software-defined cloud networks. The core idea behind SeArch is to implement various intelligent intrusion detection systems in various devices in the network. This is done by running IDS inside the gateway devices to catch suspicious activities rapidly and further emphasised by deploying IDS in multiple layers to create a hierarchical security solution with depth (Nguyen ym., 2019). Key components in enabling the IDS solutions in SeArch are machine learning algorithms that are used as protection against malicious network traffic.

Unlike in typical SDN architectures SeArch does not follow a strict application, control, and infrastructure plane approach but rather views the architecture through edge, fog, and cloud computing layers. The edge computing layer hosts switches as gateway devices which are responsible for maintaining connectivity with the controllers residing in the fog computing layer. The cloud computing layer acts as the application layer. Unlike many SDN implementations, in SeArch the majority of generated data is stored in cloud devices as they offer the most computational power and resources. Essentially in SeArch data generated by IoT sensors is both stored and processed in the application or in this case cloud layer (Nguyen ym., 2019). Naturally the required applications are also run on this layer as well.

The technical implementation of SeArch is formed through creating additional layers within the three aforementioned. The lowest level IDS is simply placed on the gateway device connecting the edge and fog layer. This IDS implementation does the initial inspection to the traffic that is forwarded to the controller inside the fog computing layer. On the fog layer an additional link is created between the controllers and a SDN application. This application is used to host and run the IDS process. Finally, on the cloud layer the IDS is run directly with the clouds resources (Nguyen ym., 2019). On top of maintaining a hierarchical model SeArch also provides horizontal support for the edge computing systems. Essentially, the gateway nodes can also run with a distributed approach providing additional shared computing resources and thereby forming a hybrid of hierarchical and distributed SDN paradigms.

This hierarchical approach comes with several recognized advantages that seek to provide additional security as well as resilience against denial-of-service attacks. SeArch uses several different machine learning algorithms which are distinct in each of the layers (Nguyen ym., 2019). This provides more security as algorithms can be chosen to complement each other and to emphasize different inspection parameters. Furthermore, this enables running less resource intensive machine learning algorithms on the lower levels reducing bottlenecks and disturbance on the edge layer nodes (Nguyen ym., 2019). On top of simply analysing traffic SeArch also provides mitigation tools for countering malicious activity. The edge layer IDS is running the lightest algorithm is intended to simply block traffic that is suspected to be malicious (Nguyen ym., 2019). This provides an initial solution. Further tools are available on the fog layer which is designed to create evasive manoeuvres such as changing flow routes or policies (Nguyen ym., 2019). These provide the network with several layers where security and response actions can be performed.

4.4 Software-defined vehicular networks

Outside of traditional IoT devices another rapidly developed sector of Internet connectivity is formed by vehicles that host increasing amounts of software and networking protocols. For example, modern cars are typically are equipped with various software with direct Internet access providing the owners with tools such as voice-controlled assistance and inbuilt navigation features. However, due to their dependencies for wireless technology and mobile nature these vehicles create new requirements for network infrastructure. To accommodate these requirements various solutions have been presented to create so-called vehicular ad hoc networks (VANETs). Much like traditional networking VANETs are also facing an increasing amount of issues in providing quality and reliable networking due to increased data loads, increased demands of data processing and requirements for timely traffic information (Barolli, Spaho, Qafzezi & Bylykbashi, 2019). Whilst data processing and storage related issues have been partially circumvented by fog and cloud services the fundamental issues regarding network management and controls still prevail (Barolli ym., 2019).

SDN provides a solution to resolve issues regarding network management and controls through its programmability and efficiency thus making it a lucrative option to complement existing VANET architectures. SDN features such as controllers are incorporated into to the network to provide centralized control mechanisms and to enable rapid topology shifts that alongside with cloud and fog services provide end users with data driven services (Barolli ym., 2019).

The first solution regarding SDN enabled VANETs is presented by Barolli ym (2019). Their solution called Fuzzy-based System for Resource Management (FSRM) integrates SDN controllers into an existing architecture for VANETs. FSRM makes use of cloud computing services to perform resource heavy computational operations and fog computing to bring beefed up computing resources near the user of the vehicle. This provides three layers of data processing when counting the initial processing performed by the integrated computer inside vehicle (Barolli ym., 2019). The actual management of the network is performed by SDN modules which are incorporated in various sections of model.

FSRM implements SDN through two key component which are a SDN module that is placed inside a vehicle and SDN controllers which are placed in the fog layer of architecture (Barolli ym., 2019). In their model Barolli ym (2019) use SDN both vertically and horizontally. The vertical features are conducted through typical SDN means where the vehicles module sends a request to a controller inside the fog layer if it needs decisions that the module cannot handle. On top of this horizontal connections are made when two or more vehicles are equipped with SDN modules in order to share information and to provide close proximity computing and reduced latency (Barolli ym., 2019). Among shared information are details such as the vehicles current position, speed, and direction which are used to gather intel for the vehicles applications.

A key detail regarding VANETs is the mobility and speed of the vehicles using the services provided by the network. Because of high speeds and constant movement of the vehicles it is critically important for the network and security applications to produce rapid and timely information for the vehicles systems. To ensure this FSRM employs features that use the provided SDN modules information to determine where the computed data will be processed (Barolli ym., 2019). For the most urgent computing lower layer computing e.g. module and fog services are also used to reduce latency and to provide timeliness. For other events the speed of the vehicle is used to determine where the data will be forwarded for further processing (Barolli ym., 2019).

More in-depth analysis regarding VANETs controller requirements are provided by Rotermund, Häckel, Meyer, Korf & Schmidt (2020) in their work assessing fundamental requirements for SDN controllers to enable VANET functionality. Whilst SDN requirements in particular for controllers have been studied previously regarding data centers, less work has been done for VANETs (Rotermund ym., 2020). Work done by Rotermund ym., (2020) provide a standardized baseline for minimum requirements that SDN controllers must meet in order to be both safe and viable options for performing on-board networking inside VANETs.

The analysis creates a fundamental presumption that each vehicle is equipped with its own controller creating a local on-board network for every vehicle (Rotermund ym., 2020). This expands the concept of Barolli ym., (2019) which depends more on external servers to make networking decisions. This approach brings more computing capabilities to an individual vehicle as both the data forwarding and network control can be performed locally reducing otherwise mandatory latency.

Rotermund ym., (2020) divide their findings into three primary groups in terms of minimal requirements. These include real-time requirements, safety requirements, as well as security requirements (Rotermund ym., 2020). Although safety and security requirements can be seen as overlapping in many ways, in this classification safety requirements address on-board safety of the vehicle whereas security is addressed through the lens of cyber security. The real-time requirements posed for the controller revolve primarily around ensuring rapid services for the vehicles to provide adequate service quality and availability (Rotermund ym., 2020). These include demands such as rapid boot times for network devices and ability to host existing standards for VANET extensions. Essentially, the network devices are not continuously operated unlike in traditional networks but are powered off when the vehicle is shutdown. As such when the vehicle is started the on-board controller needs to be ready to go almost instantaneously.

On top of timeliness several safety requirements are also presented which are designed to ensure that vehicle and its control devices have built-in fail-safe mechanisms in case the network experiences issues. These include flow state validation checks, hard-coded paths, link failure detection capabilities, and alternative controllers points for cases where the primary controller malfunctions (Rotermund ym., 2020). These requirements highlight the need for reliability and fault-tolerance for VANETs and SDN network equipment. Essentially, a SDN controller must have safety features such as emergency links, backup controllers and fault detection systems that can act if the primary system encounters issues (Rotermund ym., 2020).

Lastly are the security features which are required to mitigate and defend against malicious activities against vehicles or VANETs in general. These features are described quite vaguely but revolve primarily around integrity and validation checks. Rotermund ym., (2020) place emphasis on providing secure communication channels for both SBI and NBI. Furthermore, they suggest that the flows guiding the network should be locked for each application to ensure that only the validated application can interact and change established flows. Finally, adequate authentication mechanisms are required to ensure sufficient access controls (Rotermund ym., 2020).

After analysing existing controller solutions with these set requirements Rotermund ym., (2020) conclude that none of the widely used existing solutions met with these standards. These findings highlight that although SDN is marketed as a highly versatile technology a lot of its implementations are centralized around the same use cases. Furthermore, Rotermund ym., (2020) note that their analysis suggests that the despite existing security features that have been implemented into SDN controllers, they are still too insecure for VANET use cases in particular with insecure NBIs.

Finally, as the last implementation of SD-VANETs is presented by Islam, Khan, Saad and Kim (2021) who survey existing solutions for conducting routing inside SD-VANET with varying SDN models. In their survey Islam ym., (2021) showcase various implementations of VANETs which make use of all of the SDN models, centralized, distributed, and hybrid architectures. These seek to provide solutions to existing issues present in the SD-VANET solutions. In regard to routing, Islam ym., (2021) note that traditional routing protocols are not well suited for VANETs as calculating optimal routes is quite slow and requires stable and static paths. As vehicles naturally move around, they pop in and out of certain network segments creating a situation where traditional protocols are simply not able to keep up. Because SDN requires an up-to-date view of the topology these issues can be circumvented as a controller should constantly have a full view of the network and can therefore propagate control messages to moving nodes (Islam ym., 2021).

To resolve the routing protocol related issues, existing solutions are presented and grouped into three categories being cluster-based routing, single path routing and multi-path routing (Islam ym., 2021). These categories are combined with a chosen algorithm to provide the actual routing protocol approach (Islam ym., 2021). With the cluster-based solutions a fundamental idea is to group vehicles inside a certain network area into a cluster with a designed "head" node. This head node acts as a link between the controller and other vehicles in case other vehicles inside the cluster fall outside the connective zone (Islam ym., 2021). This ensures that connection is not lost even after moving from one zone to another and helps reduce unnecessary traffic that is needed to maintain new connections and links. As the clusters head node is tasked with keeping the network connected it is important that the cluster-based routing solutions use efficient algorithms to decide which nodes are denoted.

The second part of the routing protocol is formed by the path decision format which Islam ym., (2021) categorise into single and multi-path groups. The fundamental difference between these approaches is how complex the decision process is. With single path protocols the algorithm chooses a single link based on several parameters such as distance and density and then uses this link to forward data. With multi-path approaches this is changed as before transmitting data the source vehicle has several paths where it can choose from. This means that an algorithm must also be deployed to choose the link which data is transmitted from.

Of the surveyed VANET protocols the majority comprised of a solution using single path routing and clustering features (Islam ym., 2021). The chosen routing algorithm presents more variety as several different algorithms are deployed. A key finding of the survey is that a large portion of the proposed protocols are classified as highly complex and are primarily tested through simulation platforms (Islam ym., 2021). These findings highlight that although SDN can be used to resolve underlaying issues for VANETs, the solutions can be complex and difficult manage.

5 Security concerns of SDN

This chapter addresses currently known security issues regarding SDN as a whole and provides a more detailed view of security concerns on each component of SDN. Although SDN resolves several issues regarding current network and routing security, it is worth noting that attack vectors and vulnerabilities are also generated with a software-based solution. As such in order to be a feasible option for future large-scale networking solutions, these concerns require identification and mitigation. Security concerns are present in each of the layers creating several challenges which need to be addressed. Furthermore, issues are also generated with the architectural model of SDN as the functionality and availability of all the layers is required for the paradigm to operate successfully.

5.1 General security issues

To preface security concerns, it is worth addressing the entire architectural stack of SDN. As the previous chapters have demonstrated, the paradigm consists of three layers and their respected interfaces which enable the model to exist and operate smoothly. Whilst this provides acknowledged advantages it also generates severe issues in terms of the security of the networking model. In essence, in order for SD-networks to function, each of the components needs to run as intended. This presumption can lead to several scenarios where a single malfunctioning component can essentially lead to a complete failure of the network structure. The most notable example of this can be showcased through a centralized SDN-model, where the network is managed through a singular controller. If the controller experiences issues whether through malicious or non-malicious origins, it can completely hinder the networks' ability function leading to the administrators to be unable to command the controller through the application plane and simultaneously rendering the data planes nodes unable to update their flow tables.

The previous example showcases a critical issue in the functionality of the network, however this issue can also arise from more trivial impediments such as issues in an interface and can be a serious security concern for environments which require absolute reliability. As such these security issues can cause network managers to opt for traditional networks which provide reliance through having more fault tolerance in routing as a single routers failure does not necessarily impact the other devices ability to provide routes (Maleh, Qasmaoui, El Gholami, Sadqi & Mournir, 2023).

Another potential cause for security concerns brought forward by SDN architectures is the importance of controllers in the model. Since the controllers are mainly responsible for the intelligence of the network and for executing the commands issued by the administrators, they are also a prime target for any malicious actor looking the attack the network. Since the controllers act as a mediator between all the layers, they are also to a certain degree accessible by every component or layer (Chica ym., 2020). This can cause several security issues directed at the controller even if the actual device or layer is properly secured.

For example, a poorly configured node in residing within the data plane can send harmfully crafted networking packets causing the controller to suffer from wasted computing resources leading to a dip in performance or even a crash in the worst-case scenario. Also, the interfaces connecting the different layers may cause issues if for instance the interface alters the integrity of the data packets or drops the packets as a result of a man-in-the-middle (MitM) attack. As such the architecture of SDN should be carefully studied and understood in order for network managers to be able to grasp the full extent of the increased attack vectors existing with the new network paradigm.

Other common security issues brought forward by the general architecture of SDN include misconfigurations and weak authentication mechanisms that reside in all of the present layers. Although these issues are not per say SDN specific, they pose various security challenges that may result in security breaches through unauthorized access. Weak authentication mechanisms typically refer to security features which do not enforce secure authentication procedures such as strong passwords or multi-factor authentication and as a such potentially compromise the system or devices protected by the measure. With traditional routing models, servers and routers are protected by some authentication mechanisms such as login credentials, that ensure sufficient confidentiality. With this approach if the routers authentication credentials are secure the integrity of the routing device is quite secure.

However, with SDN the network is pieced together by all three layers which all can host applications and devices used within the chain. This means that potential attackers have a wider attack surface which they can seek to utilize to gain access to a component inside the SDN model. For example, weak passwords or even worse default login credentials can be present in poorly configured and maintained networks, which can be used by attackers to gain access to the controller, through either security software placed running inside the application layer or by data nodes operating within the data layer. As a result, SDN networks can be more vulnerable to exploitation even with adequately protected controllers, if the other components of the model are not up to par regarding their authentication controls. Therefore, it is very important that network managers and other users with access to tools controlling the network to withhold strong security procedures and to ensure that the authentication mechanisms are both enforced and strong in terms of their configuration.

Outside of authentication, other misconfigurations pose several vulnerabilities for SD-networks. As showcased in previous chapters, SD-networks offer network managers much greater variety and options for the architecture, technology, protocols and design of the network as opposed to traditional TCP/IP models. However, with this freedom comes the responsibility of adequately configuring and maintaining the chosen stack of technology running the network. This means that network managers must be equipped with the skills to deal with and understand the chosen components that are used to create the network structure. Alongside simply understanding the solution, network managers must also be aware of constantly discovered vulnerabilities and implement the required security solutions or patches that are rolled out.

The list of security issues emerging from misconfigurations is quite broad and dependent on the technology used, but these can for example be vulnerable encryption ciphers, outdated software versions running code enabling the execution of known exploits, or a general lack of enforced and mandated security features. For example, with the most common implementation of the southbound interface OpenFlow, it is up to the network managers to setup the required public key infrastructure (PKI) to maintain the standard TLS-encryption model that encrypts data transmission between the controllers and data layer nodes (Maleh ym., 2023). This essentially means that network managers must be able to correctly setup up to PKI structure, generate the necessary public-private key pair, choose an up-to-date encryption algorithm, and configure model.

Other misconfigurations examples can include running outdated versions of protocols and software. For example, with older versions of OpenFlow or certain controllers, TLS encryption is not supported or mandated to begin with (Maleh ym., 2023; Chica ym., 2020) meaning that in order to provide secure means of communication network managers need to be able to identify and utilize a correct version to reach the desired outcomes. Security issues can also be found with outdated code containing bugs that enable for example injections. An example of this could be with the northbound interface that is implemented with the REST-API. If a vulnerability is found within this API, it is up to the network administrator to make sure that they are aware of this finding and that they download and install the required patches to mitigate the found vulnerability.

Besides technical security issues the paradigm of SDN also contains general security related issues due its relative novelty which can result in security issues within the design process and lack of standardization. As previously addressed, SD-networks are still quite young which has resulted in the technology needing time to mature and becoming more fault tolerant and secure. As such, existing research and development has resulted in considerable effort being directed at the feasibility and performance of model. Whilst this has greatly improved the aforementioned features, it has often been done somewhat at the expense of security leading severe vulnerabilities that have later been patched or mitigated. However, this approach has led to several key SDN technologies which have been originally designed with a lack of security features, which have been attempted to be resolved through existing security frameworks or extensions. Although these suggested extensions showcase potential, they are still fundamentally bandages to underlying issues that should be taken into account for future designs.

In addition, key protocols such as OpenFlow offer the option to run communications through simply uncyphered TCP-channels meaning that even built-in secure features are optional (Chica ym., 2020). Thereby, even critical features ensuring confidentially can be circumvented and are not mandated whilst this it would be technically feasible.

A key issue in terms of security is also the lack of standardization which is a result of the flexibility with SD-networks. Existing standardizations and best practices can optimally act as a guideline for security procedures that can be used to enhance the security of the network. They can also provide assistance to network managers who for instance adopt new devices or software in their systems helping with the initial configurations and setups of the devices. With SDN the amount of existing documentation is often limited and can be completely lacking in particular with non-commercial solutions. Even with established technology such as OpenFlow protocol, documentation regarding best security practices and principles is quite limited and does not provide an extensive step-by-step approach to security (Deb & Roy, 2022, Nisar ym., 2020).

5.2 Application layer security

The application layer maintains the applications used by network administrators to control and command the network through controllers. Since the layer is fundamentally dedicated towards enabling the use of various applications, it is no surprise that many of the security issues regarding the layer are focused on issues and vulnerabilities that the applications can possess. Firstly, as the applications can be used to issues commands towards the networks controllers it is critically important to secure these applications behind strong authentication and identification processes. This is particularly important for networks which can be maintained remotely through the Internet. Weak credentials or default access credentials can lead to compromised software which potential adversaries can use to gain a foothold within the network compromising the availability of the system (Chica ym., 2020).

Another major group of security issues concerning the application layer resides in the code used in the software. Coding related security issues are a large group of vulnerabilities which may arise from poorly designed or written code that cause security issues in any of the major security elements such as confidentiality, integrity, or availability of the code (Deb & Roy, 2022). Vulnerabilities brought forward by bad coding practices can be both accidental or due to lack of better knowledge which result in security issues such as injections, resource hogging, denial of service, or other unexpected bugs. An example of bad quality coding arises from injection vulnerabilities, which are a result of insufficient sanitation of user inputs and often a lack of structure within the code. Injection attacks are attacks where the attackers are able to pass their own code, operating system (OS) commands, or queries through input fields which require user interaction. These injections can allow attackers to for example access file systems through inputting OS commands within the URL field of browsers. In SD-networks system injection vulnerabilities can allow attackers to inject their own flow rules or temper with existing rules guiding the controller's decision making (Maleh ym., 2023). However, it is also worth noting that SD-networks are also vulnerable to traditional OS injections if the application allow OS commands to be injected in the machines hosting the control applications.

Outside of injections, another security issue regarding code is the optimization and limitations of computational resource distribution for hosted applications (Maleh ym., 2023). Excluding the most significant tech giants such as Google or Microsoft, most networks operate with a finite number of computational resources, which must be distributed amongst the hosted applications. Resource management must ensure that each of the required applications has access to a sufficient amount of computational power or otherwise decide which applications are given priority in conflicting situations. If this is neglected or resource greedy software is run, the availability and reliability of the network can be compromised (Maleh ym., 2023). These addressed resource-based vulnerabilities and potential security issues can be caused by poorly coded in-house or third-party software in which the efficiency of the code is not noted, by malware designed to hog resources, or by simply misconfigurations (Shaghaghi, Kaafar, Buyya & Jha, 2020). These vulnerabilities can also be triggered by bugs with are not caught during testing which use infinite loops or other resource heavy processes that get stuck in the application.

In addition to the previously described issues, third party and opensource SDN software can also provide network managers additional issues regarding the validity and integrity of the code used within software. With traditional networking nodes purchased from well-known manufacturers and vendors, a certain level of trust can be placed in the internal validation process of the products security. Also, additional security features such as hard coded repository addresses and checksums are often present which can help administrators to verify that the intended updates or features are downloaded from the correct source and that the code has maintained its integrity throughout the process of downloading and installation. However, with opensource products this is not necessarily the case as the products can lack these features altogether. Thereby it is significantly more important to determine who is behind the development of the SDN applications and what their dedication to quality is. As such, more emphasis is placed on the network managers responsible for choosing and implementing the SD-network components.

With third party and opensource software administrators have to inspect the source code more thoroughly to establish that the code is legit and not containing unintended malicious segments (Maleh ym., 2023). This is further hampered by the fact that most source code comprises of several pieces including imported libraries that can also be prone to security threats. Therefore, even if the source code used to create SDN application is valid and secure, security issues can arise from the supply chain if malicious software has successfully found its way towards public repositories. For example, a popular opensource Java-based controller OpenDayLight is available through Javas public Maven repository. Albeit rare there have been instances where malware has been able to avoid security checks and has been instated into the Maven repository. This can cause issues if the controller uses for example components or libraries from the repository that are later discovered to be malicious.

Other issues of a similar nature may arise if the codes supply chains are tricked into downloading code that has been masked to resemble the original code. In these cases, the malware contained in the repositories can be accidently downloaded and integrated into the applications causing security issues. Although the previously presented example addressed the supply chain of a controller, this can also be applied into other applications such as various control applications.

Lastly, application related security issues can also be generated from a general lack of standardization present with SDN. Since SD-networks operate through software rather than hardware, the networks are considerably more flexible to serve the needs of users. However, with this flexibility also comes the added risk of having to potentially integrate various traditionally incompatible software or protocols into a single framework. As such network managers might have to rely on either publicly available or self-made solutions to combine the software. This can again result in several vulnerabilities if the code unifying the pieces is insecure and vulnerable to exploits. This again highlights the need for qualified operators and managers who are aware of proper security practices and that can assess potential dangers before proceeding to integrating new features.

5.3 Controller layer security

Controllers are the key pieces in ensuring the applicability of the SDN paradigm and largely determine the networks functionality. Because the controllers hold such an important role in the architecture, they are also a popular topic among cybersecurity scholars as well as potential adversaries and attackers

(Han ym., 2020). Existing literature has identified multiple security limitations regarding popular controller models which can cause several potential issues ranging from man-in-the-middle attacks to complete service denials attacks.

To begin with, a lucrative threat vector for potential attackers is formed by the controllers tools for routing. As the controllers are tasked with making routing decisions based on given flow rules and network topologies, adversaries can attempt to make an impact on the network by manipulating either of the aforementioned. An example of such attack can be performed by dynamic flow tunnelling. Dynamic flow tunnelling is an attack which occurs when malicious actors are able to pass flow rules to the controller either through vulnerable applications, injections or otherwise unrestricted methods (Chica ym., 2020). The vulnerability allows attackers to pass forward their own flows rules which can alter existing flow table entries by replacing them or create new rules leading to new unintended routing links (Chica ym., 2020). In addition to routing changes dynamic flow routing can also be used as a denial-of-service attack if attackers are able pass flows that are conflicting with existing rules. If the controller is not equipped with the ability to resolve the contradiction, it might become unable to continue its routing duties (Chica ym., 2020).

Another security issue regarding the controllers routing tools is known as Spanning Tree poisoning (Chica ym., 2020). In this vulnerability malicious parties attempt to disturb the controller's topology manager module by abusing inbuilt link discovery features. Since only controllers are allowed to perform routing related decisions it is paramount that they possess a full and up-to-date view of the hosted paths and devices within the network. This is done typically through the use of LLDP packets which inform controllers of the node's status. However, if an attacker has access to for example a rogue network node residing within the infrastructure layer, they can attempt to influence the topology by sending purposefully crafted LLDP packets. In essence, an attacker can create malicious packets that mislead controllers to think that new more lucrative links are present leading to controllers performing new flow rules that are then dispatched to other nodes under the controllers influence (Chica ym., 2020). This attack can thereby create security issues as legitimate links between devices can be cut leading to availability issues as well as confidentiality risks if the manipulated links enable man-in-middle actions.

Similarly to Spanning Tree poisoning, the networks links can also be vulnerable to path modifications if attackers are able to enumerate information regarding the links metadata. In networks using OpenFlow as the SBI, different links between network nodes are distinguished by datapath identifiers (DPID) which are unique to each link (Chica ym., 2020). If attackers are able to gather information regarding the datapath identifiers through for instance unencrypted network traffic and have access to a rogue device inside the network, they can craft packets with the modified header information in an attempt to spoof existing datapaths. If these spoofing attempts are successful and the controllers are tricked by the should be unique DPID forgeries, existing links are disconnected leading forced disconnections of legitimate devices (Chica ym., 2020).

Outside of attacking data links and paths another major security issue regarding controllers is spoofing and other controller poisoning attacks (Han ym., 2020; Chica ym, 2020). Similarly to the previous addressed vulnerabilities attackers can also attempt to craft network packets that seek to trick controllers into thinking that devices outside the setup network are in fact a part of the managed network. A technique dubbed controller poisoning seeks to achieve this again by making use of forged LLDP packets that are designed to trick controllers into adopting the rogue devices into its switch table and also into the networks topology (Chica ym, 2020). If the network and controller lack sufficient authentication and integrity checks attackers can seek to use this security problem to gain a foothold inside the network.

It is worth noting that although previously discussed vulnerabilities often use forged LLDP packets it is not the only protocol which can be spoofed. Similarly other layer 2 and 3 protocols can be abused to create spoofed tracks. Researchers have shown that also protocols related to traditional networking such as IP and Address Resolution Proctol (ARP) can be faked in attempt to hijack traffic (Han ym., 2020). For example, Han ym., (2020) note in their literature review regarding SDN controller security issues and mitigation solutions that popular open-source controllers such as OpenDayLight and Floodlight have been shown to be prone to spoofing attacks highlighting their risks.

Although many of the spoofing threat vectors target controllers from other layers of the architecture they can also be initiated from inside the control plane. More complex and larger network models make use of distributed SDN models which means that more than one controller is applied to provide resilience, additional computation power as well as to mitigate bottlenecks. As such it is to be expected that the network contains multiple simultaneous functioning controllers that communicate with each other. However, this can be weaponized by attackers if they are able to add a new rogue controller into the network. Without proper security implementations adversaries can hijack controller locations and instate their own rogue controllers into the network (Rahouti ym., 2022). If the attackers manage to successfully integrate their own controller into the network, they can then seek to disrupt or modify routing decisions.

Another security issue present in the control plane are the EBI and WBI links between the controllers in distributed SDN architectures (Maleh ym., 2022). In distributed models each controller is only in responsible for their designated segment of the network, which mean that in order to have access to the entire network topology or flow information in other segments, they need to communicate with other controllers. This communication can therefore become a target for attackers that seek to cause service denials. If attackers are able to cause issues or block communication between controller links, the controllers are not able to synchronously share information between each other causing functionality issues (Maleh ym., 2022).

Finally, as a last security concern regarding controllers routing decisions is an attack known as switch table flooding. As the name suggests, this attack involves an attacker having access to a device, for example a compromised switch which is then used to spam traffic to its controller (Chica ym., 2020). This can be achieved by making use of network traffic that seeks to inform the designated controller that a new device has entered the network and is in need of routing guidelines. This attack is primarily applicable in networks which lack authentication mechanism for new devices (Chica ym., 2020), as the absence of these security features leaves the controller unable to recognize the difference between legitimate devices and rogue devices. As a result of this attack, targeted controllers must constantly dedicate computational resources to processing the forged traffic and managing its internal switch tables to account for the so-called new nodes. This can lead to issues with availability in particular with small, centralized implementations as well as in the routing decisions being altered. Also, if the attack is successful enough controllers might become unable to add and process new switches that are later added into the network as the switch table components have become full.

Outside of security issues plaguing controllers, another key vulnerability is formed by denial-of-service attacks. As previously discussed, a fundamental issue for SDN models has been a lack in reliability with in particular larger scale implementations. Although, several controller designs have attempted to increase the power and efficiency among the devices, this does not necessarily eliminate the risks brought forward by network traffic that is purposefully and maliciously produced. As such, DoS attacks are still a major security concern due to their relative ease in execution as well as variety in attack vectors.

Possibly the simplest variation of DoS attacks against controllers can be performed in the form of packet-in flooding. This attack can be performed by making use of the SBI and a device that has been compromised in the infrastructure plane. To exploit this vulnerability an attacker in essence only has to program the compromised device to begin sending request to the controller in the form of packet-in format (Chica ym., 2020). Since these requests are naturally occurring in a SD-network the controller will begin to resolve them in order to provide routing decisions. However, as each of the request needs a certain amount of computing resources, less efficiency optimized and centralized controllers can become overrun quickly with the never-ending stream of requests. This leads to a DoS for other devices in the network as the controller is unable to store and process their requests (Chica ym., 2022).

Another vulnerability regarding DoS attacks is caused by the possibility of reverse or infinite loops (Rahouti ym., 2022). In these technique malicious actors attempt to modify the routing links in a manner that forms a never-ending loop in an attempt to make sure that the network traffic is never delivered and whilst making sure that the controllers need to process the request multiple times as a result. Because network routing needs to be dynamic to account for new devices entering or existing devices dropping, controllers typically have inbuilt mechanisms that seek to identify potential infinite loops that might form as a result (Rahouti ym., 2022). However, as Rahouti ym., (2022) present in their paper it has been shown that this security mechanism can be bypassed meaning that the security threat is still relevant. Attackers with the right technical capabilities and access to compromised devices can attempt to perform this attack leading to a denial of service if successful.

A third major vulnerability group is formed by operating systems used to run controllers in the control plane. The programmable and flexible nature of SDN is also applied and present in the control plane meaning that controllers used by the network also provide the option for running applications on top of the operating system (Deb & Roy, 2022). This is not inherently a security issue on its own as it is done on purpose and enables network administrators to for example setup security software such as additional firewalls, IDSs, packet inspection tools, and other security frameworks also on the controllers. However, it can lead to major issues if known vulnerabilities are not accounted for and actively mitigated.

As with any operating system a critically important security threat is formed with injection attacks. Operating system injections are attacks where a malicious user is able to pass OS commands as a part of the data payload which is processed and afterwards executed in the operating system. Os injections can cause several issues for the controller depending on the format of the injections. These can include for example attacks that seek to modify the data contents of the controller such as altering the topology managers data, attacks that seek to remove or destroy data such as flow rule lists, or simply leak additional information about the system to the perpetrator for further enumeration (Chica ym., 2020). In addition, common security issues not specific to SDN also include installing backdoors and deploying rootkits which can be used to gain a more permanent foothold inside the system for later exploitation (Chica ym., 2020).

Outside of injections controllers are also vulnerable to similar code and malware related issues as addressed previously in the application layer security section. Controllers can be improved and modified due to their programmability which means that the risk for buggy or dangerous code is always present (Deb & Roy, 2022). Furthermore, network managers can install additional applications and other third-party code that is processed and run by the operations system. Code downloaded from untrustworthy sources can contain serious vulnerabilities and might not be updated with regular security patches (Deb & Roy, 2022). On top of this, it should also be noted that available codes can also contain malware that is hidden among the other components (Deb & Roy, 2022). As the controller is a critical piece of the network with significant access to making decisions, they are a tempting target for bad actors with the capability of producing said malware.

A less often addressed issue that is present for controllers is the physical security of the devices. Although controllers and SDN can largely be run on virtual devices, there are still physical devices that run the software. This means that security issues targeted towards the physical hardware used to host the controllers are also directly linked to the controller's security (Han ym., 2020). This is particularly true for centralized controllers where a single controller can be solely responsible for maintaining the network. In the worst-case scenario, unsustainable and recoverable damage to the physical server can also take out the controller and thereby, the entire network. Because of SDNs relative newness and design it is not as robust as traditional networks to recovering from physical device failures. Traditional networks are decentralized meaning that if a single router in a certain region fails others will simply calculate new routes if possible. Because this is not case in smaller SD-networks the physical threats should be treated with sufficient importance.

Physical threats to SDN infrastructure can be both intentional and unintentional. For unintentional security threats issues such as natural disasters or fires could cause significant issues. Also, other issues to the physical devices that are completely unrelated to SDN could in the worst-case crash servers leading to simultaneously crashing the controller. As for intentional attacks, attacker can seek to purposefully destroy the physical devices or infect them with malware through portable means such as USB-sticks if they manage gain access to the infrastructure.

Finally, as the last of set security threats addressed in this thesis are the miscellaneous threats that concern controllers. As discussed earlier in this chapter injections can pose significant issues to the controller. Although a primary concern is often placed on OS commands, other common injection-based attack vectors can also be used by malicious parties. If the network makes use of an incorporated database, SQL injections can also become a significant threat (Han ym., 2020). With SQL injections attackers create traffic where the data payload contains SQL operators that are mistaken as legitimate database queries. With controllers SQL injections can be used by attackers to leak valuable information from for example small in-memory databases that the controllers are running (Han ym., 2020).

Another less often addressed security threat is formed again by the topology manager of the controller. Like previously showcased the topology manager can be targeted by various spoofing and flooding attacks which mess with integrity of the data. However, as discussed by Rahouti ym., (2022) the topology manager can also be targeted with the polar opposite approach. As SD-networks are designed to be flexible it is expected that the network topology changes due to natural circumstances, which is noted and acknowledged by the controller. However, attackers can also target this process through an attack called topology freezing, where the main goal is to disrupt the controller topology module leaving it in a state where implementing the updates within the network becomes impossible leaving the controllers network view essentially in a static state (Rahouti ym., 2022). Although this attack does not necessarily cause immediate issues it can cause problems if new nodes are not able to join the network or when existing once are dropped.

Lastly, the final controller security issue discussed in this thesis are side-channel attacks and data leaks. As the controller acts as the brains of the network it also processes and stores a lot of information interesting to adversaries. As such attacks targeted at simply enhancing enumeration findings can cause severe issues. This is particularly true with centralized SD-networks where a single controller manages the network and thereby, knows almost everything about the network. If the attacker is able to gain access to the controller either directly or by through for example rogue nodes within the infrastructure plane, they can in the worst case extract the entire architecture of the network from the topology manager or all of the maintained links stored in the flow table (Deb & Roy, 2022). This can naturally be leveraged for future attacks by determining the other valuable targets or by finding links that form chokepoints for potential DoS attacks.

Other threats regarding enumeration and fingerprinting controllers are also present through more traditional side-channel leak attacks. As a network operates certain things can be measured even without direct unauthorized access. Attackers can seek to measure various features such as latency, response times, routing paths and other information regarding devices operating in the network (Maleh ym., 2023). Whilst side-channel attacks are not specific to SDN and are typically difficult to mitigate due to their nature of being a normal by-product of an operating network, they are worth addressing especially in SD-networks which require high levels of resilience and reliability.

5.4 Infrastructure layer security

The infrastructure layer is dedicated to transmitting data inside the network which is why a large portion of identified security threats address threats that seek to interfere and disturb this transmission. Existing literature consists of several security threats that can be leveraged to cause denial of service attacks on the network itself or more specifically individual nodes or switches that reside in the network. These security issues often attempt to utilize the lack of intelligence in regard to routing that the devices have or simply seek to overwhelm the limited and often rigid computational resources that infrastructure layers nodes typically have. Furthermore, as popular protocols that are deployed in the southbound interface and infrastructure layer lack mandated, built-in and standardized encryption and integrity features, malicious actors can seek to exploit existing SDN tools to achieve the exploitation of the aforementioned security issues.

One of the most common security issues are denial of service attacks that are targeted specifically towards the infrastructure layers devices. These can be both individual network nodes that an attacker seeks to isolate or disconnect entirely from the network or switches connecting multiple nodes if larger service denials are attempted. Several scholars (Rahouti ym., 2022; Maleh ym., 2023; Chica ym., 2020) present in their body of work that service denials can be achieved through both traditional methods such as through TCP-packet flooding attacks or through SDN specific methods such as flow table or path link attacks.

TCP packet flooding attacks are a form DoS where the attacker abuses the TCP handshake to exhaust the devices computational resources by continuously sending TCP packets that either seek to establish new connections with SYN packets or never send the required ACK-packets to acknowledge the connection (Rahouti ym., 2022). As a result, the devices will attempt to finish the required TCP-handshake by resending the unacknowledged packets leading to an exhaustion of available connections and wasted network bandwidth. In addition to wasting available connection and bandwidth, TCP flooding attacks can also be leveraged against switches running OpenFlow by filling up the limited buffers that reside in them (Rahouti ym., 2022). If the network becomes clogged, switches awaiting rules from controllers can store data inside their buffers. However, if the DoS attack continues these buffers can runout of memory (Rahouti ym., 2022) leading to either packets having to be dropped or congestion avoidance features having to be deployed. This limits the amount data that can be transmitted leading to service denials in worst case scenarios.

Another attack vector for performing denial of service attacks is formed by the flow tables and flow rules installed inside the switches. Because SDN architecture separates control and data transmission the switches operating in the infrastructure layer are fully reliant on the controller on instructions how to function. As such they typically lack the ability resolve a potential conflict of two flow rules or the ability manage flow rules that lack required headers. This coupled with the fact that OpenFlow-based solutions do not require any integrity and validation checks causes severe vulnerabilities that can be exploited by attackers (Rahouti ym., 2022; Deb & Roy, 2022).

An example of exploiting these vulnerabilities comes in the form of forged flow rules, which can be utilized in a wide variety of attacks. Attackers can send flow rules that are designed cause conflict into switches that due to a lack authentication and validation install new rules in their flow tables causing them to become unable to transmit data to its nodes (Deb & Roy, 2022). However simply causing conflictions is not the only way this vector can be leveraged, as attackers can also seek to cause service denials by sending forged commands that instruct the switches to drop their flow tables altogether, by filling the flow tables with nonsense rules that have no real purpose, or by installing rules that cause the traffic to be directed to new malicious switch causing man-in-the-middle attacks (Rahouti ym,, 2022; Maleh ym., 2023). These man-in-the-middle attacks can be used to perform service denials if the switches are instructed to drop the captured network traffic, causing in essence a black hole effect.

Another DoS attack vector available for advisories to exploit is provided through ARP protocol. ARP is a layer 2 protocol that is used in the infrastructure plane to determine the MAC address each device. In essence ARP is used to determine which MAC address corresponds to which IP address or in other words which IP address is connected to a specific device. Due to a of lack authentication mechanisms an attacker can attempt to spoof an ARP address by simple sending out forged responses for broadcasted ARP queries (Rahouti ym., 2022). If the attacker is successful, the device broadcasting the queries, most likely a switch, will append its ARP cache with the spoofed MAC address, leading to a scenario where traffic is directed to a malicious node. Since the traffic is routed to a rogue device controlled by the attacker, they again have the option of stopping the traffic by simply refusing to forward it to the legitimate holder of the queried MAC address, causing a service denial.

Finally, as the last showcased attack vector for causing DoS attacks is taking advantage of network latency. As previously mentioned SDN switches and nodes rely on rules and instructions from controllers to function and make decisions. As such it is vital that the instructions are received in the first place and that they delivered in a timely manner. If attackers can cause traffic or congestion inside the network, they can seek to increase the latency of the responses from the controller. If this latency can be sufficiently increased switches can be left in a state where they again have to begin buffering data as they wait for instructions from the controller (Maleh ym., 2023). This can used to create service denials if the attackers are able to gather enough data into the buffer whilst simultaneously ensuring that the switch does not receive the required rules to start forwarding data and thereby emptying its buffers.

Although DoS attacks form a considerable threat to the infrastructure layer they are by no means the only attack that malicious actors can seek to utilize. Another major attack that can be performed in various ways is the previously showcased man-in-the-middle attack which can be also used as a tool for eavesdropping the network and to gather information passing through. Because encryption and tools for verifying data integrity are not mandated by default (Deb & Roy 2022; Jimenez ym., 2021) attackers can have access to a wide range malicious tools when performing a MitM. Firstly, if TLS or other encryption methods are not deployed an attacker with a successful MitM setup can simply capture and read all the data as it is transmitted in plane text format. This can be used to gather valuable information about the target, to steal information that should be private, or simply monitor the activity of the network.

Furthermore, attackers can utilize the lack integrity validation to monitor the data before passing it forward. Because there aren't any mandated integrity checks attacker who have access to a rogue device in the middle of the connection can simply change the transmitted data as they please before forwarding it towards the rightful recipient of the data. However, as no tools for ensuring integrity such as hashes are used, the recipient node has no chance of validating the data leading to a situation of simply having to believe that the received data is intact (Deb & Roy, 2022). In networks that require high levels of confidentiality and data integrity results can be catastrophic.

The aforementioned issues can also be compounded in the future when encryption tools are being setup in the form of an TLS-based PKI system. If an attacker has gained access to a malicious device and has successfully created a MitM setting, they can simply interfere with the key forming and exchange process essentially completely negating the security feature. In essence, the attacker can simply use the MitM device to generate its own set of PKI keys which it then uses to decrypt the traffic from the sender and then to re-encrypt it is own keys before passing the data to the real target.

Lastly, on top of simply eavesdropping on data, a lack of integrity and validation can also be used by attackers to perform injections on other devices in the layer. Because the switches do not contain tools for validating data sources attackers can also create malicious payloads that contain various commands that are designed to cause the target device to execute various tasks. These can include tasks such as disconnections, shutdowns and other operating system commands depending on the system that the device runs on (Chica ym., 2020). Injections offer the attackers several attack vectors to follow but they can include for example service denial by shutting down key switches that form chokepoints, redirecting traffic to desired nodes by removing existing links between nodes and switches, and enumeration by gathering information about devices through OS commands.

5.5 Interface security

Existing security research often devotes rather limited resources towards studying and addressing vulnerabilities within communication links or interfaces connecting the network. As such existing work often does not go into as much detail regarding interface security compared to for example security concerns in controllers leading to limited in-depth analysis. These generalized and condensed findings typically address security issues within the northbound and southbound interfaces neglecting the interfaces used to handle communications between controllers in distributed and hybrid SDN models. Furthermore, although SDN doesn't contain any enforced standards regarding the protocols used to implement these interfaces, much of the existing research is conducted using the most commonly applied protocols such as OpenFlow for the southbound interface and REST for the northbound interface leading to limited view of potential issues.

Since the interfaces are used to connect different layers of SDN and are designed to handle the communication between these layers, it is no surprise that many of the most commonly presented security issues revolve around these features. Two of the most common showcased and discussed topics regarding interface security are denial of service attacks and uncyphered communication channels which are present in all the security sources used in this thesis. A major problem regarding existing solutions used for implementing the interfaces is a lack proper authentication and verification mechanisms (Jimenez ym., 2021), which limit the devices ability pick up on fraudulent and malicious activities. Because there are no standardized and mandatory authentication mechanisms, attackers who have a foothold in the network can simply generate overwhelming amounts of traffic that cause severe overhead in the network leading to at minimum a loss in efficiency and in the worst-case total denial of service (Jimenez ym., 2021).

Another major security threat is brought forth by a lack of verification regarding data integrity and its origin. As with the previously discussed authentication, existing SDN models typically do not enforce and require strict verification regarding data passed among the network by default. As such if network managers do not implement their own vetted solutions attackers can take advantage of this vulnerability and cause numerous forms of damage. Firstly, attackers can seek to take advantage of OpenFlows lack of integrity checks and simply forward spoofed data through the southbound interface (Deb & Roy, 2022). This can cause various issues such as passing unauthorized network commands to nodes residing on the infrastructure plane.

Secondly, another major issue can be present with larger scale distributed SDN models. Like showcased earlier in chapter two, distributed SDN models host multiple controllers that share the responsibilities of managing the network. Depending on the model, these controllers must communicate with each other to ensure that topology of the network stays up-to-date and in a synchronized state. Because the eastbound and westbound interfaces also lack integrity tools, attackers with a foothold in the controller plane can seek to interfere with other controllers as well. Attackers can simply attempt to disrupt the overall topology view of the network by sending forged information that is accepted and updated by other controllers (Jimenez ym., 2021). This can later be leveraged to perform other actions. Attackers can also attempt to cause service denials by impeding the synchronization process in flat distributed networks. In flat distributed SD-networks none of the controllers deployed have a full view of the network topology and as such rely on regular communication with other controllers. If a controller does not respond to its peers or is simply out of sync with the other controllers, the network cannot maintain an up-to-date view of the current network state. This can also result in other controllers having to start managing other segments of the network, which can lead to unintended consequences such as a lack of efficiency or unplanned reroutes.

The second major concern prominent in most research is the lack of encryption regarding the communication channels used to operate the interfaces. As discussed many times prior in this thesis, most common SDN models do not provide or require mandated encryption and ciphering to protect their communication. As such maintaining and instating sufficient security features falls into the hands of network managers leaving potential attack surfaces for attackers to exploit. In SD-networks that do not maintain up to date and reliable encryption formats potential adversaries can monitor the traffic data gathering information about the networks structure, topology, control commands and management (Chica ym., 2020; Rahouti ym., 2022; Maleh ym., 2023).

Another form of data leaking can be caused by exploiting a listener mode feature present in SD-networks which have implemented the southbound interface through an older version of OpenFlow switches (Deb & Roy, 2022; Jimenez ym., 2021). Listener mode enables attackers to gather additional information about the network and its configurations due to an emphasized lack of authentication present in this mode (Jimenez ym., 2021). Because strong authentication is not required and listening mode is enabled, attackers can attempt to open connections to the devices to gain a foothold within the network and to gather further reconnaissance (Deb & Roy, 2022; Jimenez ym., 2021).

Lastly, on top of the most commonly presented security concerns existing literature also identifies vulnerabilities regarding the functionality of interfaces. As SDN is by its nature intended to utilize programmability its interfaces are also built to support these features. However, this also leaves them vulnerable to attacks such as injections (Jimenez ym., 2021; Chica ym., 2022; Rahouti ym., 2022). These injections can include SQL injections targeted at breaching databases residing in controllers (Jimenez ym., 2021), data extraction through passed commands (Rahouti ym., 2022), destruction of network information such as flow rules or other stored information (Rahouti ym., 2022) and shutting down legitimate devices through operating system commands (Chica ym., 2022). It is worth noting that this list could be expanded based on the actual implementation of the interface as different protocols have their own known weaknesses and special features. However, the aforementioned should be seen as common issues that are often causes for concern in many of the chosen implementations.

6 Discussion

This chapter analyses the results of this study and discusses observations made during the literature review process. The observations address the overall state of SDN, feasibility of various existing frameworks, and potential improvements that could be made to the existing body of research. This discussion addresses the observations from a general point of view and from a more direct viewpoint of applications and security research. Finally, this chapter also presents direct recommendations for future research regarding SDN.

6.1 Discussion of findings

The results of this thesis show that SDN is technological paradigm which has drawn the attention of scholars both from the standpoint of providing new innovative frameworks for future technical infrastructure as well as from the standpoint of cybersecurity. Although, this thesis set strict timely limitations to ensure the timeliness of the source material, searches to databases provided sufficient results. In addition, these databases and libraries show that literature regarding the topic has significantly increased in the past few years showing the topic is relevant. This can be seen as natural development as numerous research articles studied during this process highlight the need for innovation regarding the development of future networking.

SDN is typically advertised as a solution which is flexible due to its programmability. This is apparent when reviewing the search results of database queries as the results showcased that the architecture has been widely applied to many frameworks and applications. It is also worth noting that albeit the paradigm is a large component on its own, several research papers have integrated it as a relatively small background component to support another framework. This highlights that SDN can be applied widely as a supporting component to larger technical architectures rather than acting as the key component of the model.

Although this thesis approached the topic from various angles certain commonalities were present in the body of research which caused limitations to existing work. Since the topic is quite recent from the standpoint of theoretical background, a lot of the research is still recent and has a large focus in ensuring the overall feasibility of the paradigm. This can be seen as a natural step in the evolution of this research direction as a working foundation needs to be established before implementing and widely adopting SDN. However, this also results in the research being directed towards similar research angles and causes limitations on more directed and specific research targets. This can be seen clearly when analysing the setup of SDN models present in both the studies regarding the applications of SDN as well as security research.

For example, most research papers addressing purely theoretical SDN as well as this thesis present multiple architectures for implementing SDN including centralized, distributed, and hybrid models. On the other hand, the majority of studies discussed in this thesis adopt a centralized model excluding the other options entirely. This can be seen as a justified choice from the viewpoint of the research as centralized models are easier to implement and test through emulators. However, this leaves a wide gap in the body of research as centralized models cannot and should not be the only choice for future SDN implementations. Although, controllers have been developed and their performance has been enhanced it can be seen as unlikely that a single centralized controller could be used to administer large networks such as data centers or widely spread networks such as VANETs. This is because a centralized controller is always to a certain extent a bottleneck and because it is unlikely that a single controller could possess sufficient computing power, regardless of the level development and optimization. As such it is important that research is also performed and tested through the other models to test their feasibility and to discover potential unique limitations or opportunities that they may offer.

Another way this issue is emphasized is through the selection of protocols used to implement in particular the southbound interface of models. A key advantage that many scholars recognize is the ability to avoid been caught up in vendor and specifications related limitations due to the flexible nature of SDN. This in essence means that network administrators and architects can freely choose which protocols are used to connect each layer. Although several protocols exist, an overwhelming majority of proposed models opt to use OpenFlow to manage the SBI link. This once again has its advantages as OpenFlow is often considered as the de facto protocol for the SBI and it is the most widely applied protocol, but this causes a lack of research for the other options and somewhat unintentionally locks OpenFlow as the primary option for SDN.

Furthermore, this also has an impact on the development of network nodes and applications as OpenFlows heavy impact guides developers towards creating software which is intended primarily to support OpenFlow. This limits the flexibility of the paradigm and pushes it back towards a state where end-users become locked with existing solutions that dominate the field. More effort should be placed on both research and development of SDN which attempts to provide more options for implementing the SBI to avoid getting caught with a singular choice and its potential vulnerabilities and weaknesses.

Another issue regarding existing literature comes in the form hybrid models and the interfaces that are used integrate legacy and SDN systems. Since a large portion of research opts to use centralized models this as a consequence excludes EBI and WBI implementations out of the scope of the work. This causes several issues regarding security, adaption of SDN, and knowledge of available options. One major hurdle regarding transitioning to SDN systems is the initial entry cost of building a new system. In addition, many potential end-users may have old and highly reliable implementations of legacy systems. This can cause managers to be reluctant to undergo a shift to newer models. Hybrid models reduce this initial step as the network can be partitioned to only use SDN for a segment of the network which most benefits from its features.

Because hybrid models are not often included and researched, this can limit knowledge regarding the possibilities and thereby prevent the shift. Furthermore, this creates a lack particularly in security research that is specifically aimed at understanding the security implications and vulnerabilities specific to hybrid SDN models. Although, this thesis made a distinct decision to choose only recent publications and several vulnerabilities were discussed, security issues for hybrid models were not really present in the literature.

Another theme raised in many publications is the general lack of standards and standardizations for SDN layers, protocols, or models. This can be seen as expected mainly because SDN is still a new paradigm, versatile, and not really solely developed by any dedicated organization. Generally, the only recognized set of standardized is created and maintained by Open Networking Foundations OpenFlow which might also contribute to its popularity. Creating a predetermined and maintained set of best practices or standards is likely difficult, as without creating restrictions the scope of the project expands and even then, it is unlikely that these standards would cover all implementations.

However, having at least some form of official standards and best practices could be seen as beneficial as this would provide help for network operators who are less familiar with SDN and also create a baseline of rules that system architects could expect to encounter. Also, it is likely necessary to agree upon a general set of rules for larger scale implementations which require collaboration between several manufacturers. This could include for example smart home sensors that need to support a certain communication channel or posses a predetermined number of computing resources. This should not be considered lightly though, as creating these guiding baselines is balancing between creating a common ground for collaboration but at the same time unnecessary restrictions and limitations should also be avoided. One option could be to tailor these best practices and guidelines to match a smaller setup. This would naturally limit their applicability but also enable them to be less restrictive and targeted.

Outside of general observation it should be noted that both the application focused, and security focused research contain features that are worth noting. Firstly, when processing SDN applications it become clear that the topic was applied to timely relevant technological research. Topics such as VANETs and IoT networks are a general research target which has also transitioned into practical applications. This highlights that SDN provides functionalities and features that can create an impact on already existing solutions and can be used as a foundation for future development in these fields. Furthermore, as SDN matures and gains more practical implementations it can be presumed that it will become more well-known and viable to a further range of applications.

This thesis addressed four different applications for SDN which can be grouped into use cases for services and use cases for individual end-uses. The use cases intended for service providers include data centers and cloud applications as well as NFVs. For these cases designed SDN solutions present solutions and frameworks to known existing limitations and can therefore be seen as lucrative options. The solutions follow a general trend of utilizing virtualized services alongside providing services to reduce the requirements for in-house solutions. They also provide the user, in this case larger service providers with modern tools to improve their services through both efficiency and automatization, as well as through improved visibility for network management.

As the targeted end users are presumably large organizations with significant resources and technical capabilities, it can be assumed that the solutions are feasible to implement. However, work regarding these use cases is still largely theoretical and also tested on primarily simulated environments leading to a need in both testing in real world environments as well as further research regarding the practical applicability.

As for the use cases regarding individual end-users a common theme among the research is that many of the proposed models are rather ambitions and their realistic feasibility can be somewhat questionable. For example, models regarding IoT infused smart homes provide frameworks which integrate various network nodes that are connected to cloud and fog servers that are then used to process the user data for various applications (Conti ym., 2019). Whilst this model might be technically possible, it is worth considering is it really worth it for an individual homeowner from the standpoint usage, privacy, cost, and maintainability. Although an increase in sensors connected to an IoT network will naturally increase generated data, it is still unlikely that the collected data masses will require cloud storage and processing power. Furthermore, integrating these features will provide additional costs to the users on a consistent basis as the required services need to be purchased from the cloud providers. Even if these issues are circumvented the question of privacy should not be ignored. Transmitting personal data that is continuously gathered into a remote processing facility to be stored can be seen as an issue which could put off potential users.

Another example of potential feasibility issues comes in the form of VANET implementations. Existing research has proposed solutions where all of the vehicles would form a constantly evolving network and where the resources of these vehicles are jointly shared. Whilst this approach appears reasonable and even supportable to a certain extent, it should be noted that this is still a far reality. Firstly, as previously addressed there are currently very few actual standards when it come to SDN which means that existing manufactures have little to base their designs on. As such for the proposed model to work all of the major automobile manufacturers would have to agree to a universal standard that would have to followed. Alternatively, a certain producer of the networking technology could be chosen to create the required equipment, but this would again constrict the design heavily.

Secondly, forming a network of all traffic participants could form significant issues from a security standpoint. Like discussed in chapter five rogue devices are a security issue that plague SDN. If VANETs would be built with a mindset where new vehicles could simply pop into the network, infiltrating a rogue device into the network would become significantly easier. As safety is a top concern when it comes to vehicles, a lack of security and possible disturbances could have even fatal consequences. Also, personal privacy could become an issue as data would have to be shared with other cars.

Lastly, proposed models also require additional infrastructure to be built in order to serve the networks needs. These will require additional investments and would require a set of ground rules to operate efficiently between potential car manufacturers.

In general, the applications designed for individual users should carefully assess the frameworks viability in particular from the standpoint of an average user. If SDN is to become a primary source of network management for an average household, the technology cannot be overly complicated and overwhelming to shift into. This would reduce the initial step and help bring the technology closer to a larger userbase. One option to achieve this is to make use of SDNs ability run preinstalled scripts and software, which could be leveraged into automating the initial setup process to a large extent. If retailers are able sell essentially preinstalled and configured devices, even less tech savvy users could make use of SDN.

Finally, in regards to security literature certain patterns are also present in existing literature. Similarly to a heavy bias towards utilizing centralized SDN models existing security research has also been predominantly directed towards studying controllers and their security flaws. This can be seen as a natural step as the controller is the most important single device in the architecture and possesses the ability to manage the network, but nonetheless it is not the only possibly vulnerable element. With the emphasis being so heavily on controllers, interfaces connecting the layers are often left quite untouched leading to a lack in literature regarding their vulnerabilities and potential solutions to mitigate their vulnerabilities. This is particularly true for eastbound and westbound interfaces which were addressed very rarely. Furthermore, even for the southbound and northbound interfaces research is quite narrow and the solutions often revolve around applying TLS encryption to at least provide confidentiality.

Another important aspect visible in many literature pieces regarding security is the mindset used to design the controllers. In many cases it would appear that the primary focus around the topic has been to ensure that the controller is operational and that certain computational thresholds are met. Naturally, this is important as a controller that does not meet the demands of the network is not useful and leaves security redundant if the whole project is scrapped. However, this can also create serious vulnerabilities in case performance is prioritised over everything else. This approach can be seen in many papers reviewed for this thesis as despite the fact that security related work was only a couple years old, older existing problems were still noted and present. In addition, rather than combating the identified vulnerabilities the typical approach to mitigating them was applying an extension or other software to cover vulnerability.

It is reasonable to expect that additional security software will be necessary regardless of the security design as the threat landscape constantly evolves and because SDN enables the ability to host additional software. However, it is also worth considering whether old and known vulnerabilities should be mitigated already during the design of SD-network nodes and controllers. For example, older OpenFlow versions and SDN nodes typically lack authentication and encryption mechanisms for data transmission between the infrastructure plane and control plane. Rather than integrating a machine learning framework to expose malicious activity, controller designers could attempt to create a model which integrates these features into to the design by default. This could be done through various ways with existing authentication tools and encryption ciphers to create a stronger foundation. Also, security could be buffed by simply creating mandatory security features such as using TLS from the start.

Generally, the added security solutions could prove to be effective and worth the investment but simply relying on patching vulnerabilities as they are discovered with security tools seems quite difficult manage and unsecure. With this approach the worst-case scenario is that the network is running dozens of security frameworks and software which are all intended act as bandages for known issues.

Another observation found in the literature surveyed was that for many papers which performed a survey on available controller solutions, the primary advantage that was highlight focused on the computational capability of the controller. None of the surveys emphasized or highlighted the security of the controller. This can be considered odd as one of the key concerns regarding SDN is its ability to match traditional networks is reliability. Many researchers approach reliability through the lens of the networks and often the controllers ability to function under abnormal or high traffic conditions but this is too sparse of a viewpoint. A fundamental element of cybersecurity is the availability of resources which directly ties into the reliability of SDN. If the availability of the network or its devices is compromised it directly affects the reliability of the system. For example, if an attacker overflows a controllers flow rule table this can render it unable to operate and issue new commands, essentially crashing the network and causing a loss availability in the process. As such, security elements should be more carefully and thoroughly included when thinking about a controllers reliability and suitability.

6.2 Future research suggestions

This thesis recognizes several research avenues which should be advanced in the future to both further the existing body of research as well as to cover for identified gaps. Firstly, for future research applied SDN models should make more use of distributed models as well as hybrid models. If SDN is to be applied on a larger or even global scale, making use of these models is likely mandatory as centralized controllers will not be sufficient enough and will risk being bottlenecks hindering the reliability and security of network. Larger networks and infrastructure hubs will require distributed control which creates a need for these architectures to be studied in more detail.

Future research should consider using a distributed model even for research where the network is simulated to better understand the additional limitations that the shared control causes. This can also be used to measure additional overhead that the traffic inside the control plane while generate and to identify how much of an impact this will have on the network as a whole. Furthermore, this setting can be used to study how easily potential adversaries could disrupt the synchronization process and cause serious security issues. For hybrid models it would be advantageous to study practical examples of organizations who have shifted to using this approach. This practical research approach would give researchers data of real-world applications and realistic models of the model to study.

Another research avenue which should receive more attention in the future is practical research applications using SDN. As addressed in this thesis a large portion of literature regarding the topic is still theoretical and lacks testing in a real-world setting. Future researchers should continue their work by applying their models or frameworks to actual networks. This would provide proof-of-concept to their models as well as expand knowledge about SD-networks applicability to real-life settings. For many of the applications this step may be harder to perform as many of the models would require sizeable investments to facilitate the models needs but for security applications this should be a natural progression.

As various security extensions and frameworks are offered as a solution for acknowledged security issues researchers should attempt to apply these solutions to real controllers to test their effectiveness. This would once again provide proof-of-concept and act as a relatively low entry point to test actual real-world applicability.

Research could also be expanded from the standpoint of security. Although SDN security has received attention in particular for controller security, much of the research is focused either around identifying vulnerabilities or design research for security extensions. This should be expanded through at least two different approaches. Firstly, researchers should study creating SDN equipment and software with a security first approach. Much of the work regarding SD-networks has revolved around making the paradigm work at the expense of security. Because existing literature presents several acknowledged vulnerabilities which have not been mitigated, researchers should conduct design research that would attempt to create a solution that has built-in features to secure for example known controller vulnerabilities. Focusing on a security-oriented controller could provide end-users with a more secure option that is more resilient to common threats and thereby more reliable.

The second direction that should be explored would be comparative research specifically targeting SDN controllers and applications security qualities and features. Existing research has performed comparative research particularly towards SDN controllers and their performance capabilities. A similar approach should be applied to security features as well as. This would have several benefits. A comparative analysis between the most common and well-known controllers would expand the users knowledge about each controllers security features and how they compare. This would assist them in choosing which controller to apply into their network. Furthermore, this would also provide a clear body of literature regarding the overall state of current options. While existing literature showcases vulnerabilities it can be hard to put them into to perspective as they are universal and not device specific. As such a comparative analysis would showcase how commonly these vulnerabilities are present in the most popular controllers and applications as well as create a generalized view of the currently most pressing security issues.

Finally, as the last recommendation showcased in this thesis is research targeted towards creating informative guidelines or even general standards regarding future SDN implementations. Like discussed earlier in this thesis creating standards for SDN is not a simple process as creating mandated restrictions could limit the flexibility of the paradigm and thereby create more harm then benefits. However, certain guidelines should be considered to ensure that future networks and frameworks remain compatible. For example, VANETs should have some set of general principles which designers and engineers developing the technology can rely on and expect.

One approach to creating adequate and non-restrictive standards would be to limit the scope of the standards. This would mean that instead of attempting to create a universal set of guidelines for the whole SDN paradigm researchers should simply narrow down the scope to a predetermined target, for example a set of standards for southbound interfaces, VANETs, smart homes and such. This approach would be more manageable as the scope narrows down the requirements and reduces the number of factors that need to be considered. In addition, this approach would avoid causing restrictions that might negatively affect other applications as they could operate under a different set of guidelines.

Lastly, researchers and developers should also continue maintaining the few existing standards that already exist. For example, the standards for Open-

Flow and ForCES act currently as a general set of rules due to the protocols wide adaptation. Continuing the development of these guidelines will have a large impact in also guiding the development the paradigm.

7 Conclusion

This thesis performed a descriptive literature review on research regarding software-defined networking. Software-defined networking is a relatively new paradigm which seeks to improve and resolve issues facing traditional networking models, by separating the control and data transmission features of networking. This separation alongside transitioning from hardware-based networking to software-based networking attempts to create more sufficient networking through centralized control mechanisms and through flexibility. This thesis looked at the current state of SDN from the standpoint of available technologies, potential applications avenues, and currently existing security issues and as such the research questions guiding the thesis were the following:

1) What are software-defined networks?
2) What are the benefits and disadvantages of SD-networks?
3) Where can software-defined networks be applied?
4) What are the main security issues for software-defined networks?

This thesis finds that the main benefits showcased in existing literature revolve around the flexible and programmable nature of SDN. Due to its software-based approach, SD-networks are able to break free of previously imposed restrictions caused by hardware. SD-networks allow for large customization of the network as new features can be integrated simply through further development and programming. In addition, network administering can be enhanced as further automatization can be achieved with better scripting capabilities. Another key benefit enabled by a SD-network is a centralized and efficient control point inside the network. As opposed to traditional networking, SDN enables network administrators to manage the network and routing decisions through a single controller which propagates flow rules rapidly to nodes residing in the infrastructure layer.

The main identified disadvantages of SD-networks are the paradigms reliability and scalability in particular for centralized models. As the network can be controlled through a single control device it can form a bottleneck hindering the networks reliability and limiting its scalability. Since the control and decision are limited to a finite number of predetermined control devices if these controllers face issues the whole network can become unusable. Furthermore, scalability can become an issue if large numbers of nodes are incorporated into the network, as it is difficult to create a controller with sufficient computing resources to handle all the new devices.

For the applicability of SDN technology showcased literature typically uses the paradigm to mitigate previously identified issues inside a larger framework or technical architecture. SDN is applied for IoT networks, VANETs, NFVs and data centres. These applications typically make use of two SDNs prominent features which are the ability to integrate heterogenous devices, often present in IoT networks, to a single network as well as the ability to manage a often and rapidly shifting network topology. Furthermore, SDN is used for managing larger networks present in data centres which require efficient tools to manage numerous devices simultaneously and efficiently.

As for the security issues regarding SDN, existing literature raises multiple vulnerabilities, exploits, and issues which are present in each layer and component of the architecture. Since SDN consists of three distinct layers as well as the connecting interfaces it is reliant in all of the components working. As such the paradigm can be prone to significant security issues if a key component like the controller is compromised. Existing literature places heavy emphasis in studying and understanding security issues regarding the control plane and centralized controllers in particular. Whilst this provides important understanding for this security aspect, it also leaves gaps in literature regarding the safety of other components.

Chapter five of thesis provides a comprehensive look at identified security concerns. These can be grouped into three main categories which are a lack of authentication and integrity checks, denial of service issues, and potential confidentiality issues. Because many SDN devices and protocols do not require and impose authentication and integrity checks, potential adversaries can attempt various injection attacks which can attempt to alter flow rules or tables that are stored inside networking nodes as well as controllers. Furthermore, this lack of security features can also enable malicious nodes to enter a network. Another key issue is formed by denial of service attacks which can be used to cripple the availability of the network. Malicious actors can simply flood the network with various packet types or flow rules which in turn exhaust a controllers computing resources or a network nodes flow table. These issues can be tied to a lack of authentication as potential rogue devices can be difficult identify as a result of lacking security features.

Based on the results of the literature review this thesis highlights emerged general observations regarding the body research in the discussion chapter. SDN is considered and advertised as a highly flexible and adaptable solution for networking. However, this not often present in chosen research settings as much of the literature opts to use a narrow scope of the technology using primarily centralized models with de facto protocols. This trend of opting for a similar research design setting limits work on other SDN implementations and architectures leading to a gap in knowledge regarding their applicability and also guides users to adopt this approach causing restrictions on the paradigm.

Another theme present in the literature is a general lack of research applied to real-life settings. The majority of literature utilizes simulated environments to conduct measurements for their SDN-based frameworks or designs. This should be further expanded in the future to also include follow up research to assess the true feasibility of the proposed models. Conducting additional research with real-life settings would provide information about the models reliability and feasibility in experimental settings that are not as controlled and potentially highlight issues which might emerge from overly ambitious designs.

Finally, much of the work regarding security research focuses on controller security and generally favours applying frameworks or extensions to fix identified issues. For future research this should be accounted for by expanding research to cover less studied areas such as interface security or security of hybrid and distributed models.

SOURCES

Ahmad, S., & Mir, A. H. (2021). Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers. *Journal of network and systems management*, 29(1), . https://doi.org/10.1007/s10922-020-09575-4

Alsaeedi, M., Mohamad, M. M., & Al-Roubaiey, A. A. (2019). Toward adaptive and scalable OpenFlow-SDN flow control: A survey. *IEEE Access*, 7, 107346-107379.

Amin, R., Reisslein, M., & Shah, N. (2018). Hybrid SDN networks: A survey of existing approaches. *IEEE Communications Surveys & Tutorials*, 20(4), 3259-3306

Bannour, F., Souihi, S., & Mellouk, A. (2017). Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Communications surveys and tutorials,* 20(1), 333-354. https://doi.org/10.1109/COMST.2017.2782482

Barolli, L., Spaho, E., Qafzezi, E., & Bylykbashi, K. (2019). A New Fuzzy-Based Resource Management System for SDN-VANETs. *International journal of mobile computing and multimedia communications, 10(4), 1-12.* https://doi.org/10.4018/IJMCMC.2019100101

Benzekki, K., El Fergougui, A., & Elbelrhiti Elalaoui, A. (2016). Software-defined networking (SDN): a survey. *Security and communication networks*, 9(18), 5803-5833.

Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., ... & Parulkar, G. (2014). ONOS: towards an open, distributed SDN OS. *In Proceedings of the third workshop on Hot topics in software defined networking (pp. 1-6).*

Boubendir, A., Bertin, E., & Simoni, N. (2016). NaaS architecture through SDN-enabled NFV: Network openness towards web communication service providers.

Bwalya, B., & Zimba, A. (2021). An SDN approach to mitigating network management challenges in traditional networks. *International Journal on Information Technologies and Security*, 13(4), 3-14.

Chica, J. C. C., Imbachi, J. C., & Vega, J. F. B. (2020). Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, *159*, 102595.

Cisco. 2023. "A smarter WAN in second" Retrieved 6.5.2023 from: https://meraki.cisco.com/sdwhat/en

Cisco. 2024. "Cisco Locator/ID Separation Protocol (LISP)" Retrieved 19.7.2023 from: https://www.cisco.com/c/en/us/products/ios-nx-os-software/locator-id-separation-protocol-lisp/index.html

Cisco SD-WAN. 2023. "Cisco SD-WAN powered by Meraki". Retrieved 6.5.2023 from: https://meraki.cisco.com/product-collateral/meraki-sd-wan-overview/?file

Conti, M., Kaliyar, P., & Lal, C. (2019). CENSOR: Cloud-enabled secure IoT architecture over SDN paradigm. *Concurrency and computation*, 31(8), e4978-n/a.

Cziva, R., Jouet, S., Stapleton, D., Tso, F. P., & Pezaros, D. P. (2016). SDN-Based Virtual Machine Management for Cloud Data Centers. *IEEE eTransactions on network and service management, 13(2), 212-225*. https://doi.org/10.1109/TNSM.2016.2528220

Deb, R., & Roy, S. (2022). A comprehensive survey of vulnerability and information security in SDN. *Computer networks* (Amsterdam, Netherlands : 1999), 206, 108802. https://doi.org/10.1016/j.comnet.2022.108802

Feamster, N., Rexford, J., & Zegura, E. (2014). The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2), 87-98.

Fu, Y., Bi, J., Gao, K., Chen, Z., Wu, J., & Hao, B. (2014). Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks. *In 2014 IEEE 22nd International Conference on Network Protocols (pp. 569-576). IEEE.*

Gilani, S. M. M., Usman, M., Daud, S., Kabir, A., Nawaz, Q., & Judit, O. (2024). SDN-based multi-level framework for smart home services. Multimedia tools and applications, 83(1), 327-347. https://doi.org/10.1007/s11042-023-15678-2

Guo, Z., Dou, S., Wang, Y., Liu, S., Feng, W., & Xu, Y. (2021). HybridFlow: Achieving Load Balancing in Software-Defined WANs With Scalable Routing. *IEEE transactions on communications, 69(8),* 5255-5268. https://doi.org/10.1109/TCOMM.2021.3074500

Haji, S. H., Zeebaree, S. R., Saeed, R. H., Ameen, S. Y., Shukur, H. M., Omar, N., Sadeeq, M.A.M., Ageed, Z.S., Ibrahim, I.M., & Yasin, H. M. (2021).

Comparison of software defined networking with traditional networking. *Asian Journal of Research in Computer Science*, 1-18.

Haleplidis, E., Salim, J. H., Halpern, J. M., Hares, S., Pentikousis, K., Ogawa, K., . . . Koufopavlou, O. (2015). Network Programmability With ForCES. *IEEE Communications surveys and tutorials, 17(3), 1423-1440.* https://doi.org/10.1109/COMST.2015.2439033

Han, T., Jan, S. R. U., Tan, Z., Usman, M., Jan, M. A., Khan, R., & Xu, Y. (2020). A comprehensive survey of security threats and their mitigation techniques for next - generation SDN controllers. *Concurrency and computation*, 32(16)

Hassas Yeganeh, S., & Ganjali, Y. (2012). Kandoo: a framework for efficient and scalable offloading of control applications. *In Proceedings of the first workshop on Hot topics in software defined networks (pp. 19-24).* Hassas Yeganeh & Ganjali, 2012

Islam, M. M., Khan, M. T. R., Saad, M. M., & Kim, D. (2021). Software-defined vehicular network (SDVN): A survey on architecture and routing. *Journal of systems architecture, 114,* 101961. https://doi.org/10.1016/j.sysarc.2020.101961

Jarschel, M., Zinner, T., Hoßfeld, T., Tran-Gia, P., & Kellerer, W. (2014). Interfaces, attributes, and use cases: A compass for SDN. *IEEE Communications Magazine*, 52(6),

Jimenez, M. B., Fernandez, D., Rivadeneira, J. E., Bellido, L., & Cardenas, A. (2021). A survey of the main security issues and solutions for the SDN architecture. *IEEE Access*, *9*, 122016-122038

Juniper. 2023. “Cloud-Native Contrail Networking Datasheet”. Retrieved 6.5.2023 from: https://www.juniper.net/us/en/products/sdn-and-orchestration/contrail/cloud-native-contrail-networking-datasheet.html

Karakus, M., & Durresi, A. (2017). A survey: Control plane scalability issues and approaches in software-defined networking (SDN). *Computer Networks*, 112, 279-293.

Keertikumar, M., Shubham, M., & Banakar, R. M. (2015). Evolution of IoT in smart vehicles: An overview. *In 2015 International Conference on Green Computing and Internet of Things (ICGCIoT)* (pp. 804-809). IEEE.

Knopf, J. W. (2006). Doing a literature review. *PS: Political Science & Politics*, 39(1), 127-132.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., ... & Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. *In OSDI (Vol. 10, No. 1, p. 6).*

Lara, A., Kolasani, A., & Ramamurthy, B. (2013). Network innovation using openflow: A survey. *IEEE communications surveys & tutorials*, 16(1), 493-512.

Li, T., Chen, J., & Fu, H. (2019). Application scenarios based on SDN: an overview. In *Journal of Physics: Conference Series* (Vol. 1187, No. 5, p. 052067). IOP Publishing.

Li, Y., & Chen, M. (2015). Software-defined network function virtualization: A survey. *IEEE Access*, *3*, 2542-2553.

Lu, H., Arora, N., Zhang, H., Lumezanu, C., Rhee, J., & Jiang, G. (2013). Hybnet: Network manager for a hybrid network infrastructure. *In Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference (pp. 1-6).*

Maleh, Y., Qasmaoui, Y., El Gholami, K., Sadqi, Y., & Mounir, S. (2023). A comprehensive survey on SDN security: Threats, mitigations, and future directions. *Journal of reliable intelligent environments*, 9(2), 201-239. https://doi.org/10.1007/s40860-022-00171-8

Nascimento, M. R., Rothenberg, C. E., Salvador, M. R., Corrêa, C. N., De Lucena, S. C., & Magalhães, M. F. (2011). Virtual routers as a service: the routeflow approach leveraging software-defined networks. *In Proceedings of the 6th International Conference on Future Internet Technologies (pp. 34-37).*

Nisar, K., Jimson, E. R., Hijazi, M. H. A., Welch, I., Hassan, R., Aman, A. H. M., . . . Khan, S. (2020). A survey on the architecture, application, and security of software defined networking: Challenges and open issues. *INTERNET OF THINGS, 12*, 100289. https://doi.org/10.1016/j.iot.2020.100289

Nguyen, T. G., Phan, T. V., Nguyen, B. T., So-In, C., Baig, Z. A., & Sanguanpong, S. (2019). SeArch: A Collaborative and Intelligent NIDS Architecture for SDN-Based Cloud IoT Networks. *IEEE access, 7*, 107678-107694. https://doi.org/10.1109/ACCESS.2019.2932438

OpenDaylight. 2023. “Welcome to OpenDaylight Documentatio”. Retrieved 7.6.2023 from: https://docs.opendaylight.org/en/stable-argon/

Open Networking Foundation. 2015. “OpenFlow Switch Specification” Retrieved 19.7.2024 from: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

Paliwal, M., Shrimankar, D., & Tembhurne, O. (2018). Controllers in SDN: A review report. *IEEE access*, 6, 36256-36270.

Rabet, I., Selvaraju, S. P., Fotouhi, H., Alves, M., Vahabi, M., Balador, A., & Björkman, M. (2022). SDMob: SDN-Based Mobility Management for IoT Networks. *Journal of sensor and actuator networks, 11*(1), 8. https://doi.org/10.3390/jsan11010008

Radware. 2023. "DefenseFlow Network-Wide DDoS Attack Defense and Centralized Cyber Control". Retrieved (7.5.2023) from: https://www.radware.com/products/defenseflow/

Rahouti, M., Xiong, K., Xin, Y., Jagatheesaperumal, S. K., Ayyash, M., & Shaheed, M. (2022). SDN Security Review: Threat Taxonomy, Implications, and Open Challenges. *IEEE access*, 10, 45820-45854. https://doi.org/10.1109/ACCESS.2022.3168972

Rana, D. S., Dhondiyal, S. A., & Chamoli, S. K. (2019). Software defined networking (SDN) challenges, issues and solution. *International journal of computer sciences and engineering,* 7(1), 884-889.

Rawat, D. B., & Reddy, S. R. (2016). Software defined networking architecture, security and energy efficiency: A survey. *IEEE Communications Surveys & Tutorials*, 19(1), 325-346.

Rodriguez-Natal, A., Portoles-Comeras, M., Ermagan, V., Lewis, D., Farinacci, D., Maino, F., & Cabellos-Aparicio, A. (2015). LISP: A southbound SDN protocol? *IEEE communications magazine, 53(7), 201-207.* https://doi.org/10.1109/MCOM.2015.7158286

Rotermund, R., Häckel, T., Meyer, P., Korf, F., & Schmidt, T. C. (2020). Requirements analysis and performance evaluation of SDN controllers for automotive use cases. *In 2020 IEEE Vehicular Networking Conference (VNC) (pp. 1-8). IEEE*

Ryu. 2023. "Welcome to RYU the Network Operating System(NOS)" Retrieved 7.6.2023 from: https://ryu.readthedocs.io/en/latest/index.html

Shafique, A., Cao, G., Aslam, M., Asad, M., & Ye, D. (2020). Application-Aware SDN-Based Iterative Reconfigurable Routing Protocol for Internet of Things (IoT). *Sensors*, 20(12), 3521. https://doi.org/10.3390/s20123521

Shaghaghi, A., Kaafar, M. A., Buyya, R., & Jha, S. (2020). Software-defined network (SDN) data plane security: issues, solutions, and future directions. *Handbook of Computer Networks and Cyber Security: Principles and Paradigms*, 341-387.

Shang, W., Yu, Y., Droms, R., & Zhang, L. (2016). Challenges in IoT networking via TCP/IP architecture. *NDN Project*.

Shirmarz, A., & Ghaffari, A. (2020). Performance issues and solutions in SDN-based data center: a survey. *The Journal of Supercomputing*, 76(10), 7545-7593

Sinha, Y., & Haribabu, K. (2017). A survey: Hybrid sdn. *Journal of Network and Computer Applications*, 100, 35-55.

Song, P., Liu, Y., Liu, C., & Qian, D. (2017). ParaFlow: Fine-grained parallel SDN controller for large-scale networks. *Journal of network and computer applications, 87, 46-59*. https://doi.org/10.1016/j.jnca.2017.03.009

Tootoonchian, A., & Ganjali, Y. (2010). Hyperflow: A distributed control plane for openflow. *In Proceedings of the 2010 internet network management conference on Research on enterprise networking* (Vol. 3, pp. 10-5555).

Wang, A., Mei, X., Croft, J., Caesar, M., & Godfrey, B. (2016). Ravel: A database-defined network. *In Proceedings of the Symposium on SDN Research* (pp. 1-7).

Wazirali, R., Ahmad, R., & Alhiyari, S. (2021). SDN-OpenFlow Topology Discovery: An Overview of Performance Issues. *Applied sciences*, 11(15), 6999. https://doi.org/10.3390/app11156999

Yeganeh, S. H., & Ganjali, Y. (2016). Beehive: Simple distributed programming in software-defined networks. *In Proceedings of the Symposium on SDN Research (pp. 1-12).*

Younus, M. U., ul Islam, S., Ali, I., Khan, S., & Khan, M. K. (2019). A survey on software defined networking enabled smart buildings: Architecture, challenges and use cases. *Journal of Network and Computer Applications, 137,* 62-77

Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2014). A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, 17(1), 27-51.

Zhu, L., Karim, M. M., Sharif, K., Xu, C., Li, F., Du, X., & Guizani, M. (2021). SDN Controllers: A Comprehensive Analysis and Performance Evaluation Study. *ACM computing surveys*, 53(6), 1-40. https://doi.org/10.1145/3421764