TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MIKKO KULMALA

IMPROVING NETWORK SECURITY WITH SOFTWARE-DEFINED NETWORKING

Master of Science Thesis

Examiner: Professor Jarmo Harju
Examiner and topic approved by
the Council of the Faculty of
Computing and Electrical Engineering
on 9 December 2015

ABSTRACT

MIKKO KULMALA: Improving network security with software-defined networking
Tampere University of Technology
Master of Science Thesis, 48 pages, 3 Appendix pages
April 2016
Master's Degree Programme in Signal Processing and Communications Engineering
Major: Communication Networks and Protocols
Examiner: Professor Jarmo Harju
Keywords: SDN, software-defined networking, information security, computer networking

Software-defined networking (SDN) is a new technology in computer networks, which enables the management of the network and the development of new network functions in a higher level of abstraction than in traditional networks. In the SDN concept, the management of the network can be centralized to a specific SDN controller instead of managing each network device separately through a vendor-specific interface. This enables new possibilities for designing computer networks and makes the administration easier than before.

In this thesis we are considering the security improvements in computer networks achieved by the software-defined networking. The purpose of the research is to find out if the current maturity of the SDN technology allows traditional networks to be replaced by SDN and what kind of security enhancing network functions can be implemented with the SDN technology. We are also discovering existing SDN applications and solutions presented in former research.

Based on the research, the solutions providing improved network security can be divided to two categories. First is the SDN security applications and second is the solutions that are providing better network management. Many of the proposed solutions are still under development and they will need more research and development contribution before they are ready for the production use. During the research, it became clear that the SDN technology brings new security threats for consideration because of the centralized network management and the management performed by software. In particular the attacks against the management network and the usage of the third party software are possible security threats. Currently, migration from a traditional network to an SDN based network needs still much resources, but in the future the technology will definitely become more common.

TIIVISTELMÄ

MIKKO KULMALA: Verkon tietoturvan kehittäminen ohjelmisto-ohjatuilla verkoilla
Tampereen teknillinen yliopisto
Diplomityö, 48 sivua, 3 liitesivua
Huhtikuu 2016
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma
Pääaine: Tietoliikenneverkot ja protokollat
Tarkastaja: professori Jarmo Harju
Avainsanat: SDN, ohjelmisto-ohjatut verkot, tietoturva, tietoliikenneverkot

Ohjelmisto-ohjatut verkot (software-defined networking, SDN) on uusi tietoliikenneverkoissa hyödynnettävä teknologia, joka mahdollistaa verkkojen hallinnan ja toiminnallisuuksien määrittelyn aiempaa korkeammalla tasolla ohjelmallisesti. Käytettäessä ohjelmisto-ohjattuja verkkoja verkkolaitteita ei enää tarvitse konfiguroida yksittäin valmistajakohtaisen rajapinnan avulla vaan verkkojen hallinta voidaan keskittää erityiselle verkkoon sijoitetulle SDN-ohjaimelle. Tämä luo uusia mahdollisuuksia tietoliikenneverkkojen suunnitteluun ja tekee verkkojen ylläpidosta sekä toiminnallisuuden muuttamisesta aiempaa vaivattomampaa.

Diplomityössä käsitellään ohjelmisto-ohjatuilla verkoilla saavutettavia tietoliikenneverkon toiminnallisuuksia, jotka edistävät verkon tietoturvaa. Työn tarkoituksena on selvittää, onko SDN-teknologia jo tarpeeksi kehittynyttä, jotta sillä voidaan korvata perinteisiä tietoliikenneverkkoja, ja millaisia tietoturvaa edistäviä toimintoja teknologialla voidaan saavuttaa. Työssä tutustutaan myös valmiisiin tietoturvaa edistäviin SDN-sovelluksiin sekä aiemmassa tutkimuksessa esiteltyihin ratkaisuihin.

Tutkimuksen perusteella tietoturvaa edistävät toiminnot voidaan jakaa kahteen osaan, tietoturvaa edistäviin SDN-sovelluksiin ja parempaa verkon hallintaa tarjoaviin ratkaisuihin. Monet esitetyistä ratkaisuista ja tarjolla olevista SDN-sovelluksista ovat kuitenkin vielä kehityksensä alussa ja vaativat lisää tutkimusta ja tuotekehitystä. Tutkimuksessa havaittiin, että ohjelmisto-ohjatut verkot tuovat myös uusia tieoturvauhkia keskitetyn hallinnan sekä verkon ohjelmistopohjaisen hallinnan vuoksi. Etenkin hallintaverkkoon kohdistuvat hyökkäykset sekä kolmannen osapuolen sovellusten käyttö verkon hallintaan ovat esimerkkinä tietoturvauhista. Tällä hetkellä perinteisen verkon päivittäminen ohjelmisto-ohjatuksi verkoksi vaatii vielä paljon resursseja, mutta teknologia yleistyy varmasti tulevaisuudessa.

PREFACE

This thesis was done in the Department of Pervasive Computing at Tampere University of Technology. The topic was provided by Prof. Jarmo Harju.

I would like to thank my supervisor Prof. Jarmo Harju for giving me opportunity to work for the department and guiding me during the work. I would also like to thank M.Sc. Joona Kannisto for giving me valuable feedback and guidance through this process and M.Sc. Markku Vajaranta for giving me advice regarding the experiments.

I am grateful to the staff of the Faculty of Computing and Electrical Engineering for providing me needed guidance during my studies. Special thanks to my friends at TeLE for helping me to maintain a balance between work and social life. Last but not least, I would like to express my gratitude to Laura for supporting me and encouraging me during my studies.

Tampere, 17 April 2016

Mikko Kulmala

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

API	Application programming interface
BGP	Border gateway protocol
DDoS	Distributed denial of service
DoS	Denial of service
IDS	Intrusion detection system
IoT	Internet of things
IPS	Intrusion prevention system
OSPF	Open shortest path first
OVS	Open vSwitch
REST	Representational state transfer
SDK	Software development kit
SDN	Software-defined networking

1. INTRODUCTION

Technologies used to design, build and manage communication networks have stayed unchangeable during last decades. At the same time, the amount of network connected devices has grown exponentially which means that also the size and number of communication networks has increased. Accordingly, the existing networks in companies and data centers have become much more complex and harder to administrate.

While the number of IT services is growing, companies are moving them out from self managed and self hosted infrastructure. Also the building and using of cloud environments have become more popular and the new technologies like Internet of Things (IoT) have created the idea that every single device has to be connected to the Internet. All of these things are bringing the security to essential part of networking and specially in cloud and data center environments the security is an important matter.

The next big movement in communications engineering is to migrate traditional and static Ethernet networks to much more dynamic and easily manageable ones. The new concept, software-defined networking (SDN), transforms the layer 2 Ethernet networks to centrally managed layer 2 clouds, where the network traffic can be controlled using higher level programming language. That allows us to build more efficient and more secure networks when the network resources can be allocated on demand instead of dividing the network to smaller logically separated ones beforehand. In Figure 1.1 there is an example of a small SDN network where the traffic of one network client is considered as untrusted and for that reason is forwarded through firewall.

In this thesis, we are exploring solutions based on SDN network, which are providing better network security. We are also finding out existing SDN applications built specially for enhanced security purposes. The purpose of this work is to find out present state of SDN concept and evaluate it's maturity for wider use on production

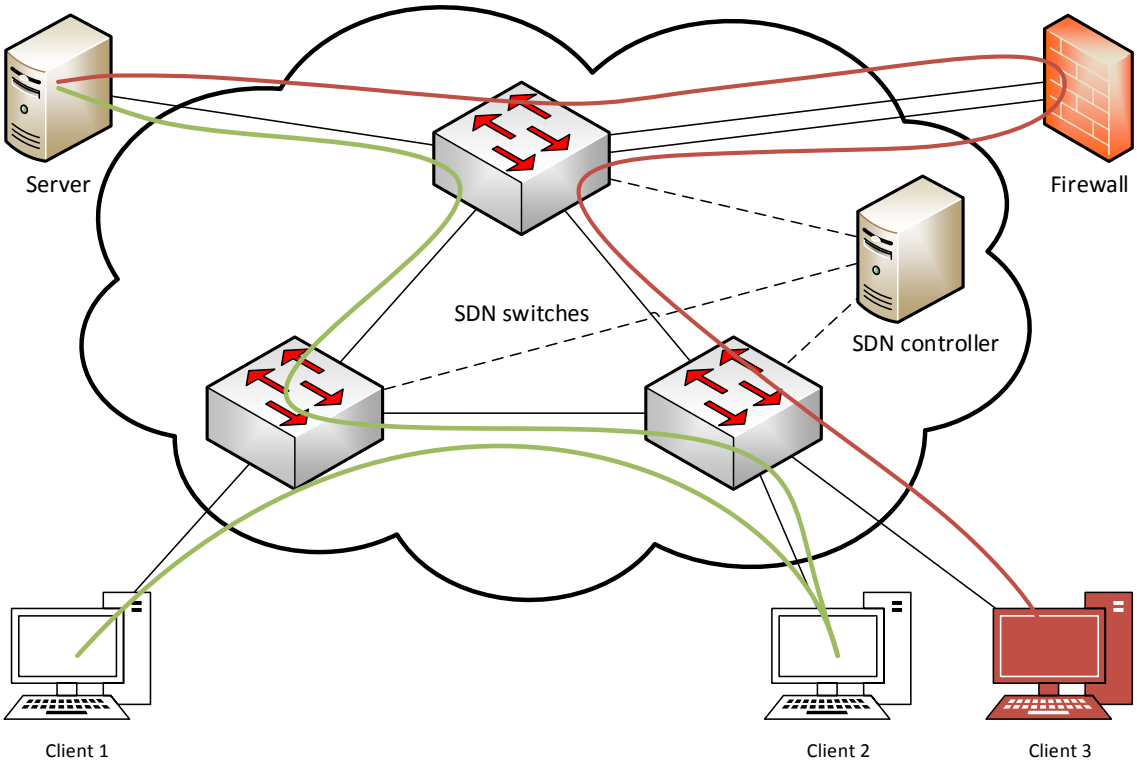

Figure 1.1 *An example of SDN enabled network.*

environments.

The second chapter describes the concept and architecture of software-defined networking. We are also getting familiar with key elements of SDN network and general benefits of SDN. Chapter 3 focuses on network security and security techniques and challenges where SDN could be helpful. Chapter 4 reviews related work on SDN for network security and in Chapter 5 the exploration of existing SDN solutions is performed. The discussion is carried out in Chapter 6 and finally the conclusion is presented on Chapter 7.

2. SOFTWARE-DEFINED NETWORKING

Traditionally we have been using subnetting for dividing large networks to smaller logical ones. Different subnets are combined to each other with routers, and traffic management is done by routers and firewalls between logical networks. Now that different kind of cloud services have become popular, we will need more efficient ways to handle networking problems.

This chapter covers the fundamentals of software-defined networking and compares SDN to traditional networking. We are looking at the architecture of SDN and how the networking elements are communicating with each other. Also SDN controllers and SDN capable networking switches are introduced in this chapter.

2.1 Concept and architecture

While the importance of information networks has been growing in all kind of companies during the last decade, also the work needed for network administration has become more critical and time-consuming. The larger the network grows the more networking devices is needed. Usually each networking device, for example switch, has its own vendor-specific operating system and configuration syntax. Actually we can say that the controller layer and the data layer are both in one specific networking device.

The basic concept of SDN is to separate the control layer and centralize it to one single point of network which means that every single network device need only take care of data layer and move data packets from one point to another based on the forwarding decisions made by the SDN controller [1]. Thus every switch is controlled from one specific controller through application programming interface (API) and the controller is commanded with application layer SDN applications. The basic SDN architecture is shown in Figure 2.1 and the SDN controller is described more accurately in Section 2.2.

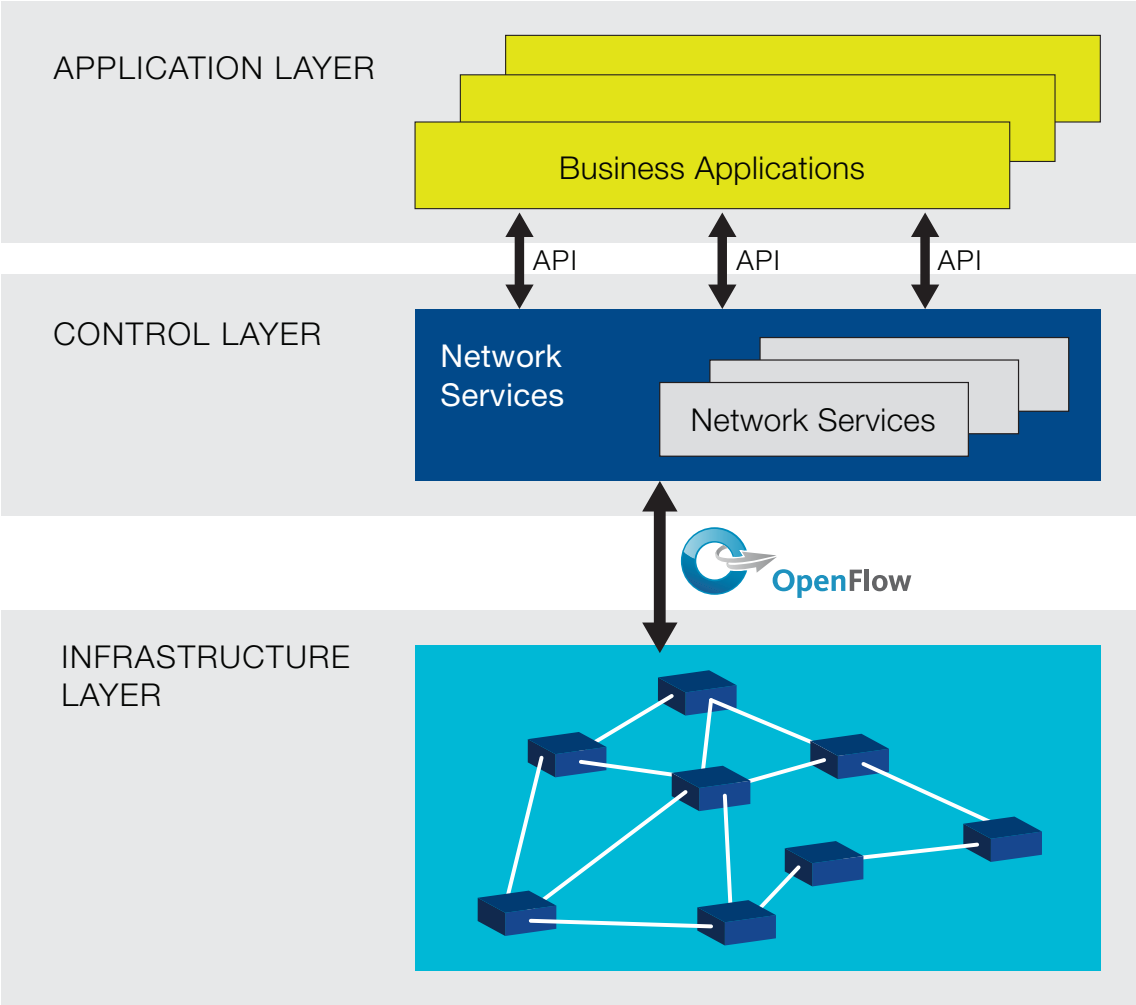

Figure 2.1 *SDN architecture [2].*

The application layer or management plane is the layer above the SDN controller where the whole SDN network is managed. The managing element may be one specific SDN application written for some specific task of a much more complex system. For example OpenStack cloud computing platform can manage network by using its own network management module.

SDN controller configures flows to SDN switches and controls how the nodes connected to the switch can communicate. With flows we can isolate networking devices connected to the switch and control the traffic on OSI layer 2 [3, p. 50], although devices are still connected to the same L3 subnet. That is a much more convenient way to do isolation than we have used before. In traditional network we have used VLANs (Virtual LAN) for L2 separation and subnets for L3 separation of devices. That means we are used to do a lot of work when a new device or a group of de-

vices requiring a separate logical network are connected to an existing network. The difference between switch fabric built by traditional switches and SDN switches is shown in Figure 2.2.

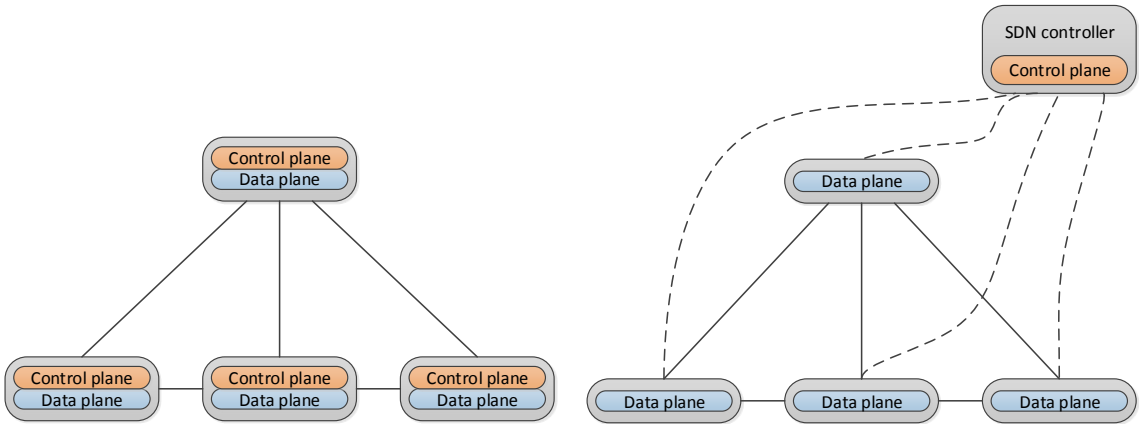

Figure 2.2 *Traditional infrastructure layer and infrastructure layer managed by SDN controller.*

2.2 Controller

SDN controller is the most important part of the SDN network because it is the brain of the network. The function of the controller is to manage and control SDN capable network elements in the network and handle the control plane which is detached from the actual forwarding devices. That means the network elements should only take care of moving traffic between devices according to the rules that the controller has created. Usually SDN controller is only one software running on a standard computer hardware or a virtual machine. Most controllers have two main application programming interfaces which are called northbound API and southbound API. Northbound API is the interface which is used to command the controller with external SDN applications or network orchestrator. Southbound API is respectively the interface which is used to the communication between the SDN controller and a network element. Interfaces are described in more detail in Section 2.3 and 2.4.

Currently there are many commercial and open-source controllers available and there are many differences between them, such as the features of controller and the target use case. Some controllers (e.g. POX [4]) are simple, easy to learn and easy to develop whereas others are much more complex, feature-rich and therefore usually

more harder to get familiar with. The used programming language affects to the efficiency of controller. Diverse features and APIs are needed if we want to build more complex cloud networking system. In this thesis we are using commercial controller from HP for testing purposes. HP's controller is one of the most popular commercial SDN controllers and it is built with Java programming language. It is also quite well documented and community supported which helps developing and testing SDN applications.

2.2.1 HP VAN SDN controller

HP is one of the big companies that have really invested in SDN solutions during the last years. HP VAN SDN controller is probably the best known commercial SDN controller at the moment and therefore we are also using it in this thesis. HP offers also SDN App Store which includes ready-to-go SDN applications and is also a place to discuss and participate to an SDN community. [5]

HP VAN SDN controller is Java-based software which runs on Linux operating system in a server class hardware or a virtual platform. The controller has an open API which allows users to develop 3rd party SDN applications. HP also offers quite extensive documentation of interfaces and also an SDK (Software development kit) for developing applications directly to the SDN controller. [6]

2.2.2 Open source SDN controllers and frameworks

There are multiple open source projects in progress and some of these have a little different approach towards the SDN concept. Floodlight [7] is popular, easy to use and open community controller developed by Big Switch Networks. It is a good starting point for studying SDN and creating SDN environment. Trema [8] is a simple framework that allows one to build an own controller with simple Ruby scripts. The framework also includes a network emulator which helps verifying the functions of the controller. There are also many other alternatives available but as in open source projects usually, the developing of some projects has stopped.

The most complex and enterprise class open source SDN controller is OpenDaylight platform [9]. It is modular and highly scalable controller for building and managing networks of different size. It is also possible to integrate OpenDaylight with OpenStack cloud operating system, which makes it a good choice for cloud environments.

FlowVisor [10] is slightly different than traditional controllers because it acts a transparent proxy between OpenFlow switches and multiple SDN controllers. In this way it can divide the network to slices and delegate control of each slice to different controller. There are also more comprehensive open source projects in progress. Open Network Operating System (ONOS) [11] is introduced as an SDN operating system which is matured and production ready platform for creating applications and services. The mission of ONOS is to produce a system that will allow service providers to build real software-defined networks.

2.3 Northbound API and application layer

The northbound API is a way to manage the controller and the whole SDN network. By developing so called external SDN applications we can modify network structure and policies. Northbound API is usually implemented using restful API (representational state transfer) which makes managing of the controller easy with basic HTTP-methods like POST, GET, PUT and DELETE. Overview of OpenDaylight SDN controller architecture is represented in Figure 2.3.

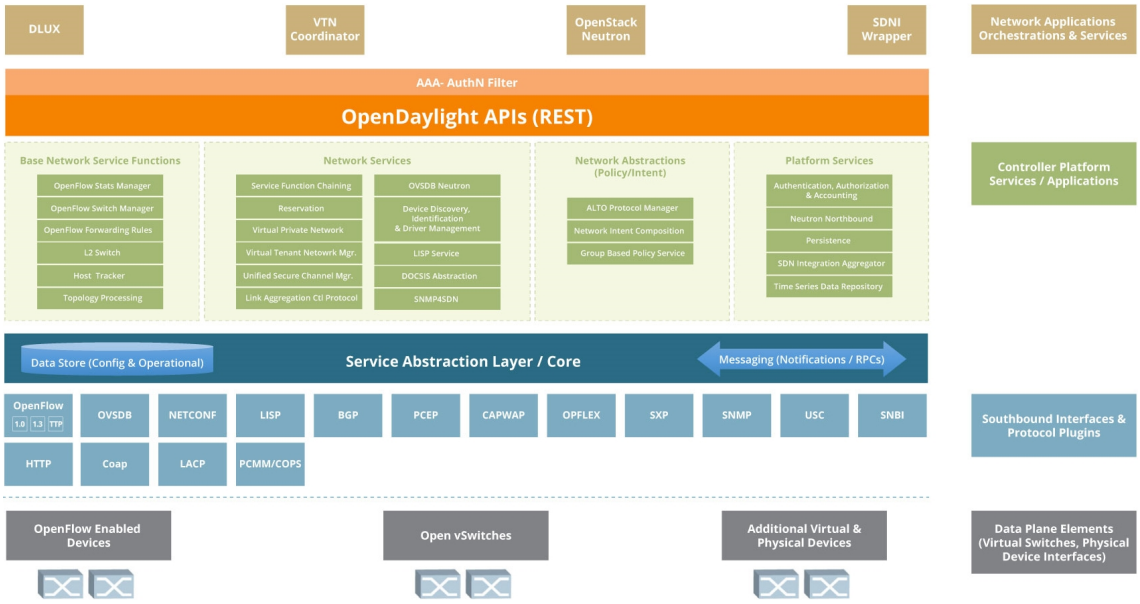

Figure 2.3 *OpenDaylight architecture [12].*

Applications which are controlling SDN network using northbound API are called external SDN applications because they are running outside the controller and are using the programming interface for communication. That kind of applications are

able to manage the controller and send instructions for it. They can not hold the real time status of the controller and operations inside external application can't be triggered by the controller because the interface is one-way communication channel. One use case for external SDN application could be for instance a piece of software which modifies the data path of one specific defective device in network which is generating abnormal traffic. The decision is based on information gathered by the network monitoring appliance.

The element sending instructions to SDN controller can be also a much more complex system than only an application built for deploying specific function to network. OpenStack cloud operating system is a large and modular open source software platform which can manage the whole cloud environment including compute, storage and networking resources [13]. Another example of using northbound REST API is the integration of OpenDaylight controller and OpenStack cloud operating system. OpenStack Neutron communicates with OpenDaylight controller through northbound API using OpenStack ML2 plugin. That means we can use OpenDaylight SDN controller to manage the whole cloud network of OpenStack environment.

2.4 Southbound API and infrastructure layer

The main goal that we are trying to achieve with SDN is to make the infrastructure layer easier to manage. Traditionally every switch and router on specific networking domain have to learn their own environment with routing protocols (e.g. OSPF and BGB) and keep their own forwarding table up to date. When a new device is connected to the network, it will advertise itself and existing devices should update their forwarding table accordingly. [1]

In software-defined networking this environment learning is centralized to SDN controller and switches are managed by the controller through southbound API. The controller is keeping up the view of the network and it is configuring necessary flows to every switch under the controller. Switches will keep up flow tables which tell where to forward packets. If switch can't make decision based on beforehand programmed flows, it will execute the configured default action that can be for example sending the packet to the controller. The relation between the controller and OpenFlow switches is illustrated in Figure 2.4.

SDN controller can handle the managing of forwarding devices by both proactive and

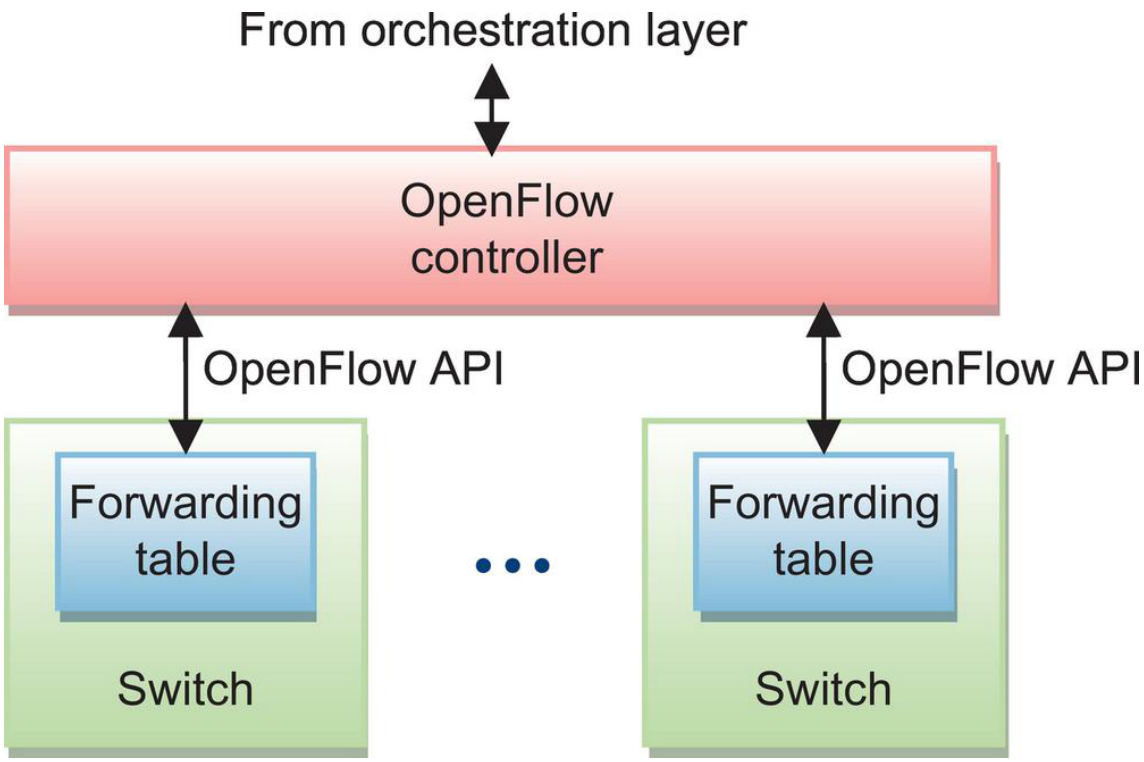

Figure 2.4 *Infrastructure layer in SDN network [1].*

reactive way. Proactive method needs the flow configuration before the traffic enters the network, so the forwarding switch has the needed forwarding rules installed already when the first packet arrives to the device. Instead of proactive method, reactive method needs the first packet before the needed flows will take place in the switch. When the switch notices that it does not have any matching flow for incoming traffic, it will send the packet to the controller through a secure channel. The controller then generates the needed forwarding flow and sends the packet back to the switch and installs the flow in to the switch. When the next packet of the same data stream arrives to the switch, it will match to the newly installed flow and the switch can forward it right away.

While OpenFlow is a widely used protocol in SDN solutions, it is not the only choice for communication between SDN controller and forwarding elements. SDN concept does not care how the communication is implemented. ForCES [14] is proposed to one possible southbound protocol and also for example NETCONF [15] and LISP [16] have their own use cases.

2.4.1 OpenFlow protocol

OpenFlow is an open standard and currently the most popular protocol for communication between the SDN controller and OpenFlow enabled switches. OpenFlow was designed at the Stanford University [17] on 2008 and currently it is managed by the Open Networking Foundation (ONF) [18]. The basic concept of OpenFlow is that switches have forwarding tables and an open API which OpenFlow controller is using. Since the controller knows the view of the network, it will populate the needed forwarding rules to the switch. The open standard allows connecting different devices from multiple vendors to one OpenFlow capable controller. [1, 19]

The specification of an OpenFlow switch is shown in Figure 2.5. OpenFlow protocol defines the specification for communication between an OpenFlow controller device and an OpenFlow switch and the behaviour of data plane function in an OpenFlow switch. The communication is performed through a secure channel where both the OpenFlow control messages and the transferred data packets are moving from the controller to a switch and vice versa. The switch includes pipeline processing including multiple flow tables where the incoming data packet is processed. [19]

One OpenFlow entry in a forwarding table consists of match fields, statistics fields and a set of instructions to apply to matching packets. When a new packet arrives to the switch, it will be checked against match fields of the flow tables. If the packet matches to some flow table entry, the actions are performed. Basic actions for incoming packet are:

- forwarding the packet to one or multiple switch ports
- encapsulate the packet and send it to OpenFlow controller through secure channel
- drop the packet
- modify fields of the packet
- send the packet to normal processing pipeline.

Possible match fields in initial OpenFlow V.1.0 specification are switch input port, VLAN ID, VLAN priority, Ethernet source address, Ethernet destination address,

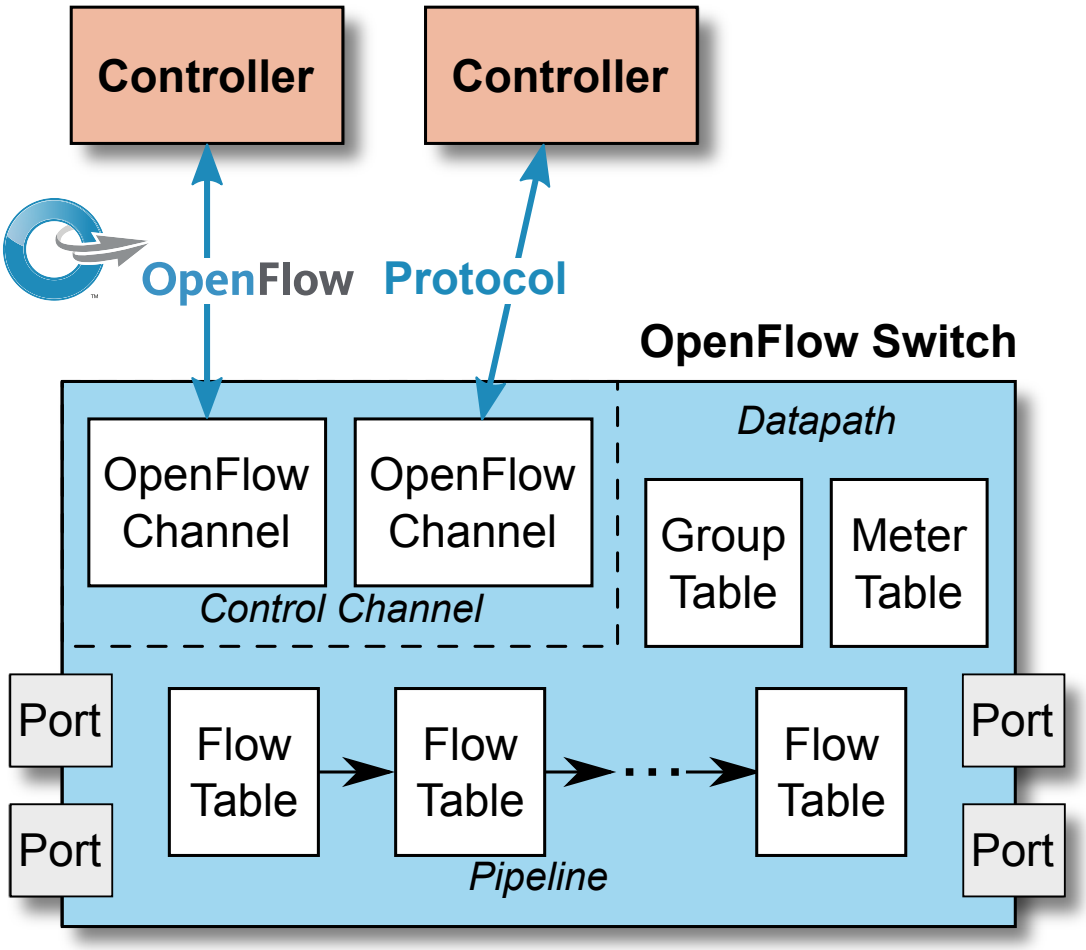

Figure 2.5 *OpenFlow switch specification [20].*

Ethernet frame type, IP source address, IP destination address, IP protocol, IP Type of Service (ToS), TCP/UDP source port and TCP/UDP destination port. Instead of just forwarding the packet to the specific out port or port group, the OpenFlow protocol also allows the switch to modify packet headers quite widely. Even the first version of OpenFlow protocol enabled modification of MAC address, IP address, VLAN ID and port of Ethernet packet and the next widely used version 1.3 brought the ability to modify for example IPv6 headers, ARP headers and ICMP headers. [19]

An example of OpenFlow rules in JSON format is shown in Figure 2.6. The example rule matches to a packet which is coming in to the switch through port 4 and which has source IPv4 address 10.0.1.1 and destination IPv4 address 10.0.2.2. The matched packet is then forwarded out from switch port 1. Idle timeout field has a value 30

which means that the flow is automatically deleted 30 seconds after the last matched packet. Hard timeout is 300 seconds which means that the flow will get deleted after 5 minutes even though there is matching traffic passing the switch all the time. The flow entry has also table id and priority fields which will define the processing order of all OpenFlow rules in a network switch. [19]

```
{
    "flow": {
        "cookie": "0x2031987",
        "hard_timeout": 300,
        "idle_timeout": 30,
        "instructions": [
            {
                "apply_actions": [
                    {
                        "output": 1
                    }
                ]
            }
        ],
        "match": [
            {
                "in_port": 4
            },
            {
                "ipv4_src": "10.0.1.1"
            },
            {
                "ipv4_dst": "10.0.2.2"
            },
            {
                "eth_type": "ipv4"
            }
        ],
        "priority": 59000,
        "table_id": 0
    }
}
```

Figure 2.6 *An example of OpenFlow rule.*

The journey of OpenFlow have started on 2007 when Ethane was presented in Stanford University [21] and the first OpenFlow version 1.0 released on 2009. The biggest improvement that the next version, OpenFlow V.1.1, brought, was the support for multiple flow tables and group table for action buckets. After the version 1.1 the management of OpenFlow was moved to Open Networking Foundation. While versions 1.2 and 1.3 added some functionality to OpenFlow, the version 1.3 has become

the last major milestone release and it is now widely used. [19] Currently the newest version of OpenFlow specification is V.1.5 [20].

2.4.2 SDN capable networking device

The last key element of SDN enabled network in addition to SDN applications and SDN controller is SDN capable networking elements. Usually we are speaking about OpenFlow enabled network switches, which are able to communicate with SDN controller by using OpenFlow protocol. The switch can be either physical or virtual one depending on network clients and use case. Physical ones are used for instance in office environment as an access switch where client computers are connected. Virtual ones are very useful in cloud and data center environments where plenty of virtualized networking or computing instances are connected to each other. There are a number of SDN switches available, both commercial and open source ones. [19]

Open vSwitch (OVS) is one of the most popular production quality open source virtual switch and it supports OpenFlow protocol. It is originally designed by Nicira which is now acquired by VMware. OVS software can be run on the top of the hypervisor or in physical switch device. It is also videly used in virtualization and cloud platforms such as XenServer and OpenStack. [22]

Multiple vendors are investing in OpenFlow enabled physical switches. In this thesis we are using switches from HP because these are known as good production quality devices and it is presumable that those will work nicely together with HP's SDN controller.

An OpenFlow enabled switch can operate in one of the two possible modes: pure OpenFlow or hybrid mode. On pure OpenFlow mode, forwarding decisions are made by OpenFlow pipeline processing and managed by OpenFlow controller. On hybrid mode, the switch will operate simultaneously as a traditional network switch and as an OpenFlow enabled one. [19]

2.5 Network function virtualization

Network function virtualization (NFV) is a concept where the functionality of a physical network element is provided by virtualized software running on top of the

hypervisor. The network element can be for example a switch, router, firewall or an IDS appliance. The main idea of virtualization of network functions is to make the network more flexible and reduce development costs. Because the CPU power of the traditional server hardware is growing all the time, it is much more convenient to implement a network function on top of the server class hardware instead of buying new network hardware from vendors. NFV is very useful in data center environment where a service provider can easily implement the needed virtualized network function based on request from a tenant. [1]

When considering SDN networks, NFV is very close to SDN concept and it is useful to be included in SDN environments. While it makes doing different network related tasks for certain traffic quite straightforward when the network appliance and the computing resource are both running on top of the hypervisor, it will also make it quite simple to deploy new network related functions like load balancer, VPN server or network monitoring instance. If the amount of network traffic is growing, it is straightforward to allocate more CPU time for virtualized network function instance.[19]

3. NETWORK SECURITY: PRINCIPLES, TECHNIQUES AND CHALLENGES

While the number of devices connected to the Internet has been growing all the time and more valuable information is being transferred over the internet, network security has become a very important question in all kind of business and companies. Some companies are moving their services to the cloud and the others are providing this environment where multiple business critical systems are running on the top of the same physical server hardware. The amount of network traffic is growing and service providers are forced to offer more effective and scalable network resources without bypassing any security goals.

This chapter covers the basic background of network security. First we are describing the basic principles of information and network security and common threats on network security. After that we are focusing on the basic security techniques and services in traditional communications network. Eventually, we consider the security challenges in different kind of circumstances where the SDN concept could prove to be better than the traditional solutions.

3.1 Principles of information security

Historically, information security has been divided to three different sections and interaction between them: confidentiality, integrity and availability. This approach is also called as the CIA triad. Achieving the needed level on one of these sections is not a problem but usually it reduces security from the other perspectives. In other words, reaching a good level of information security is always balancing between these three main goals. [23]

Confidentiality denotes that only authorized parties are able to get access to information. While information has always a value, it is important to take care of privacy

which is quite close to confidentiality. Common mechanisms to reach this goal are encryption of data and authentication with user ID and password.

Integrity means the consistency of information. Unauthorized people should not be able to modify information and information must be also protected against external threats such as system crash or network break. The integrity of information can be ensured for example with checksums.

Availability or planned availability is reached when authorized people can access the information whenever they need to. Keeping hardware and software up to date and in good condition is the key thing when ensuring availability. Capacity of hardware and throughput, backups and redundant systems are issues that have to be considered. Availability can be threatened by nonhuman threats such as power interrupts or by human caused threats like denial of service attacks.

Specially in the field of communication networks there are two more important properties that we have to take into account.

Authentication is the ability to verify users identity. The real life example of authentication could be a customer who is going to bank and is intended to withdraw money from a bank account. Before the customer can get money, he will tell his claimed identity by showing an ID card and a bank officer will verify his identity by comparing the picture on the ID card and the face of the customer.

Non-repudiation means that sender cannot deny having sent information and also that the receiver cannot deny having received the information.

3.2 Network security

In this thesis we are focusing on network security, which is only a little part of larger information security domain. In a computer network, there are many threats and vulnerabilities that network administrators have to consider. When we are aware of those, we have to protect the network and defend against them. In networking there are multiple technical ways to protect networks, detect vulnerabilities and inspect traffic.

3.2.1 Security threats

A security treat is a potential harm that may threaten our asset and threats can be divided to multiple different sections. Threats can be caused by human and by nonhuman events. A nonhuman event can be for example flood or fire, loss of electrical power or failure of an important component such as hard disk. Threats caused by humans can be either malicious or not. Benign threats are usually human error that are common for all people. That kind of harms will come true if user accidentally types wrong command to Unix shell, carelessly sends information to a wrong receiver or drops his laptop on to the floor. [23]

Nevertheless, most of the security threats are human caused and malicious when someone particularly wants to cause harm to vulnerable asset. CIA triad can be analyzed also from the perspective of harm that a malicious threat will cause to our asset. Harms can be categorized to four different types:

- interception
- interruption
- modification
- fabrication.

3.2.2 Layered security

Layered security (or defense in depth) is a security strategy where we have multiple security layers around the secured device, data, an application or a network. That means the possible attacker has to break multiple different security layers to get in touch with the protected asset and every layer should slow the attacker down. Layered security also means that every layer of security uses a different security mechanism. The selection of different layers will depend on the possible threat, vulnerabilities and the type of asset that we are protecting. [24]

One proposal of the layered security model adapted to network security is to divide the security to four layers (Figure 3.1): passive protection, active protection, passive monitoring and active monitoring. By using this kind of segmentation we can place

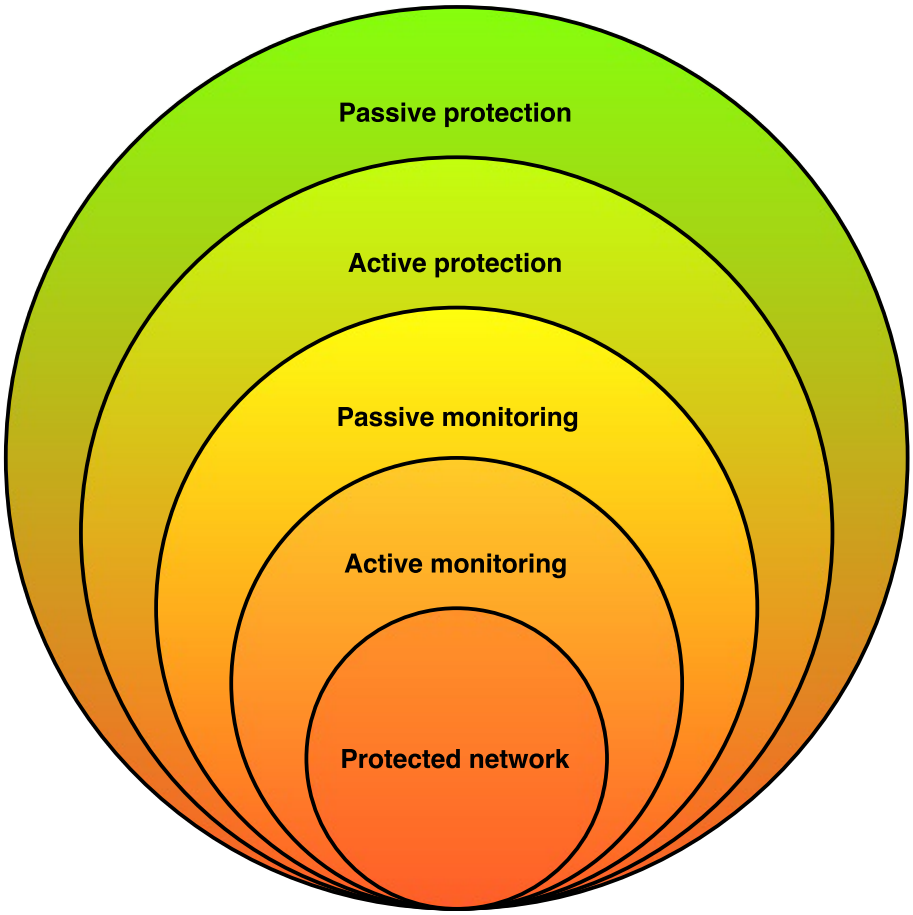

Figure 3.1 *A model of layered security.*

different network security techniques to different security layers of this proposed model.

Passive protection is the most outer layer of the model and it includes protection mechanisms that don't need active concern. An example of that is the simple packet filter firewall which implements the basic firewall rules. Second layer, active protection, includes techniques that are actively changing their functionality depending on the current state of the network. Stateful firewall, application firewall and intrusion detection and prevention systems are on this layer. Passive monitoring is for example gathering logs from network devices and traffic. If the gathered logs are inspected manually, we have already moved to the layer of active monitoring. Active monitoring includes also continuously performed analysis of the network, vulnerability scans and for example penetration testing. Firewalls are discussed in Section 3.3.1, intrusion detection systems in Section 3.3.2 and networks management and monitoring in Section 3.3.3.

3.2.3 Security policies

When we have identified security threats that threaten our secured asset or system, we have to define proper security policies. A security policy is a rule in a high level of abstraction, which is describing security goals of an organization. Policies are based on the threat analysis and security needs. A policy can be for example the rule that a user can access and read the data on a host but can't modify it. In the case of network security, security policies are usually defining the network devices or logical networks that another device or network can communicate with. [23]

When the security policies are defined properly, we have to implement multiple security mechanisms to a network which executes these policies, for example a network monitoring appliance and a firewall. Policies are also defining how the network administrator would develop and manage the network in a proper way. A security policy can be enforced for example by implementing a firewall rule to a network gateway which denies the all traffic from the Internet to a private network except the traffic which is destined to a web server's port 80. [23]

3.3 Security techniques

In this section, we are discussing of the basic security techniques in the field of network security. We are discussing how to monitor and detect malicious traffic in communication network and when such traffic is detected, how to stop the attack and prevent the attacker to cause any damage against the protected assets.

3.3.1 Firewall

A Firewall is the most important security device for protecting network by preventing unwanted traffic and attacks. It is a device that filters network traffic between a trusted network and an untrusted network. The filtering is done based on beforehand created firewall rules where the firewall compares a network packet going through it. Usually the untrusted network is the Internet and the trusted and protected network is a private network of a company, campus or user's home. Firewall can locate also between two different logical networks that have different security policies. For example if a company has a separate logical network which includes

services that will process highly confidential data, it is desirable to separate that network from the others by using a firewall. [23]

Because a firewall is often the single point through which traffic is channeled between two logical networks, the performance aspect is really important. Firewall is usually a dedicated device which is specially designed to filter traffic effectively but it is also possible to run a virtualized firewall instance on top of traditional server hardware. Firewall can be also implemented as an application that runs on the top of the operating system like a personal firewall in a home computer. [23]

Firewalls can be divided to three different main categories and each type of firewall has its own specific use case. Different threats will also need different types of firewalls. A *packet filter firewall* is the simplest possible firewall type. It is operating on OSI layer 3 so it doesn't care the data inside IP packet. Packet filter will check only the packet headers and make the decision to forward or drop the packet based on these. Usually the packet filter rules are made to match specific source or destination IP address, port number or internet protocol type and packets are processed one by one. If the matching firewall rule is not found, the default action, which can be deny or permit, is performed. Packet filter firewall is also called as a stateless firewall, because it does not keep up any kind of information regarding current network connections. [23]

A *stateful firewall* is a firewall appliance, which will maintain some kind of state concerning current traffic flows and network connections. With the information of traffic the stateful firewall can make a bit diverse traffic filtering than the traditional packet filtering firewall and it is therefore operating on OSI layer 4. One example of stateful firewall could be ongoing port scan attack where the attacker is trying to establish a connection to multiple randomly selected ports on a short time window and the connection will fail. Stateful firewall can recognize and categorize this as unwanted traffic and block all traffic from this specific source address. [23]

An *Application firewall* or an application gateway is operating on OSI layer 7 or application layer and it is able to inspect passing traffic in more detail than a packet filter. With application firewall it is possible to filter network traffic based on content inside IP packets, so for example accessing some websites from one specific subnet can be denied while the rest of HTTP traffic is working normally. Application firewall allows content modification too from which the spam filtering proxy is a one feasible practical example. [23]

3.3.2 Intrusion detection system

The fact is that even though we have carefully protected our network, implemented firewalls to correct positions and fulfilled the other needs of security policies, there is always the risk that some malicious attacker is able to intrude into our network. Or the malicious actor can be found among people, who already have access to our internal and protected network. A system, which monitors network traffic and is trying to detect malicious activity, is called an intrusion detection system (IDS). When IDS system is detecting some abnormal actions, it will trigger alert for network administrator and upgrade the event to security logs. IDS works in communication networks like a fire alarming system on building which triggers alarm when it detects smoke. [23]

IDS system can be either a device or a piece of software running on a traditional workstation. It has one or multiple sensors which could be monitoring traffic, auditing vulnerabilities or misconfigurations, checking the integrity or data or systems, and doing other kind of things that will help detecting unusual actions in the network. There are two main types of intrusion detection systems available: a signature based system and an anomaly detection system, which both have their own particular use case. [23]

A signature based intrusion detection system is looking for known patterns from network traffic and actions which it will compare to signatures from already known attack types. It is keeping the signature database up to date like a virus scanner. Another commonly known type of IDS system is based on anomaly detection. That kind of IDS system is using artificial intelligence and learning the usual and traditional transactions happening in the monitored network and when it notices some abnormal or suspicious activity, for example in packet streams, it will trigger an alarm. In other words, it is investigating the statistical probabilities of current activities and generating alarms based on probabilities. The anomaly detection system can use also things like a neural network to help decisions. [23]

An intrusion prevention system (IPS) is a system which has the functionality of IDS system added to automatic response capability. It will automatically trigger some needed operation to stop the detected intrusion. Usually it is desirable to have IDS functionality on some level, because it is possible that the network administrator is not able to solve security incidents immediately. Sometimes the term next generation firewall is used to mean a device which will combine the functionality from IDS, IPS

and application firewalls. [23]

3.3.3 Network management and monitoring

Network management is a continuously performed action starting from the network planning and continuing during the whole life of the network. Because the network traffic is dynamically changing, the network needs to be dynamically changing also to meet the needs. Network management is a diverse subject and it includes things from capacity planning, network addressing and load balancing to actively performed network monitoring and event management. Different kind of network management tools can simplify the network administrator's job significantly. For example a network configuration management system can be used to help the configuration of multiple network devices.

Network monitoring is a crucial part of the network management, which allows us to manage other security functions such as IDS of IPS. It is also ensuring us to reach our main security goals. The simplest possible way to perform network monitoring is the gathering of logs from network's services and network devices in our networks. We can for example look at the authentication log of our SSH server and analyze the frequency of brute force login attempts or we can analyze the health status of our network device or server hardware by exploring the system log files. And again, the amount of malicious and blocked network traffic is revealed from the packet log of a firewall.

In small and non-critical environments manually made log analysis can be enough but when we are managing more critical networks, log analysis should be automated. The next level of automation could include the automatically sent e-mail to an administrator when the hard disk breaks down in our RAID array of SDN controller. Or possibly an automatically triggered alarm, if some network device appears to be broken, for example, when it does not answer to ping.

When the managed network is big enough, some more convenient way to handle security events is required. Security information and event management (SIEM) is a piece of software or an appliance which collects data from security devices and services in a network. It can analyze the data in a real-time and generate security alerts to network administrators. Usually the data analysis is compared to the history of security events when it is possible to detect unusual patterns, which may

be indicators of an security incident. [23]

3.4 Challenges

As we have stated, the size and number of computer networks have grown during the last years and is growing all the time. We have seen IoT emerging and cloud services become everyday even in small companies which are moving services from physical server hardware in to the cloud. At the same time the amount of malicious traffic naturally has increased. This constant change will address more challenges to maintain a high level of network security specially in the field of network management and intrusion detection and prevention. With software-defined networking, we can try to answer these challenges.

4. SDN AND NETWORK SECURITY

During the last years the software-defined networking has received a lot of attention from the research community. In this chapter we are exploring the related work and discovering how the network security is proposed to be improved with the SDN concept. Also possible security trade-offs and new issues are considered.

This thesis divides SDN related security improvements to two categories. The first includes solutions that are clearly security related and are implementing some security functions to a computer network with SDN related techniques. The other includes suggestions which are providing better network security by means of better network management which is improved with SDN solutions.

4.1 Security improvement with SDN

Security improvement using SDN based solutions can be achieved by developing new security functions which are answering to current security challenges better than the old ones. Security improvement can be achieved also by implementing existing security functions more effectively so that the operational costs of the network administration and execution of current security policies are being reduced.

Network monitoring and traffic analyzing are security functions that can be performed using SDN based solutions. Network monitoring and implementing security functions in SDN network is analyzed in a survey by J. François et al. [25]. One example solution of traffic analysis and control in small office and home environment is proposed by I. Hafeez et al. [26]. The solution consists of a secure box and a cloud based traffic analyzer. The secure box is an enhanced SDN controller which controls multiple access points and OpenFlow switches, and it replaces home gateway by allowing hosts to connect to The Internet through it. The cloud-based traffic analyzer runs virtual firewall and IDS functionality and can analyze the network traffic and populate new security rules to defend against attacks.

FlowCover, a low-cost and high-accuracy monitoring scheme, is proposed in [27]. In FlowCover the flow statistics is collected from the SDN controller which is connected to every OpenFlow switch in the network and which is also performing the network controlling and monitoring tasks. Flow statistics have been used for anomaly detection also in [28] by using an sFlow-based mechanism. sFlow is a protocol for gathering flow statistics from switches and it is being used as a network monitoring mechanism also in OrchSec [29], which is an orchestrator-based architecture for enhancing network security in SDN networks.

Implementing of security services and functions is needed in addition to network monitoring and traffic analyzing. FRESCO [30] is an OpenFlow security application development framework. It is an OpenFlow application which is designed to help in developing and designing of security services to OpenFlow networks. FRESCO framework includes a library of 16 reusable modules and it is trying to simplify the developing of security applications by using its own FRESCO scripting language. Much more security solutions relying on SDN technology is explored in a survey [31].

Using software-defined networking for security enhancement in wireless mobile systems is discussed in [32]. The article proposes a security enhancement framework for wireless mobile networks which is designed to help designing of IDS systems, DoS prevention and secure handoff. The research advocates that the maturity and the fast development of SDN makes it a good way to improve security in mobile networks.

Moving target defense solution called *OpenFlow random host mutation* is introduced and evaluated in [33]. The idea of moving target technology is to mutate end-hosts' IP addresses randomly and frequently which can defend them against different scanning-based attacks. In this technique the IP address is unchangeable from the perspective of end host and the randomly generated and frequently changing virtual IP address is translated to the real IP right before the host by using OpenFlow rules.

Denial of service (DoS) or distributed denial of service (DDoS) have become a common way to attack against computer networks. Using SDN for recognizing that kind of attacks in particular is considered in [34] and defending against them in [28].

4.2 Network management benefits

Because the computer networks are continuously growing, the managing of the networks has become a factor which directly affects the level of network security. Because the SDN concept makes the L2 cloud of switches an entity which is managed by a centralized controller, it definitely makes the network more secure also, at least if the management has performed well. SDN will also simplify the operation in a multitenant cloud and data center environment.

Hyajoon and Feamster are identifying key problems in a current network infrastructure and they are proposing improvements for network management [35]. According to the proposal, three aspects can be improved:

- frequent changes on network condition and state,
- support for network configuration in a higher level programming language and
- perform network diagnostics and troubleshooting.

Hyajoon and Feamster have designed Procera which is an SDN based network management framework that helps network operators to manage the network infrastructure. The system defines northbound interface which allows creating event driven and high-level reactive policies to the network by a high level programming language. Procera has been deployed on campus and on home network.

Network security management problem is considered in [36]. The article is examining effects that separation of a control and a data plane in SDN network has in terms of network security compared to traditional network architecture. It is also considering the infrastructure of an SDN network and a question of which services should be left to the data plane and which should be taken to the separated controller plane.

Network slicing is discussed in [37]. Slice abstraction mechanism and algorithms for compiling slices are also developed and a prototype of slicing is designed. Another proposition of slicing an SDN network to multiple logical networks, a network virtualization layer called FlowVisor, is presented in [38]. The network slicing is a solution where a network administrator has to build logically separated networks inside a one SDN network. That is needed for example in a multitenant data center environment where multiple customers are running their own network and services

on top of the same hardware with other customers. Traditionally the network slicing is performed by using VLANs in computer networks but in SDN networks, using simultaneously the old and the new technology only creates another unnecessary layer of complexity. In the solution proposed in [37] the network slicing is performed by programming while the FlowVisor acts as a proxy controller between SDN network and multiple SDN controllers.

Overheads and costs that SDN technology is causing are analyzed and a model called DevoFlow is proposed in [39]. The aim of DevoFlow is to pull some of the control back to switches so that the control of the most important flows still remains to the SDN controller. This reduces the load of the controller. DevoFlow is introducing two new mechanisms: rule cloning and local actions which will accomplish the proposal of transferring part of the control to the switches

Network hypervisor concept is proposed in [40]. The proposition is based on the fact that different actors in the SDN ecosystem are thinking the current SDN concept in a different way. That heterogeneity makes it difficult to develop SDN networks which are connecting the network resources from multiple service providers. The new abstraction layer, network hypervisor concept, is trying to solve that problem by providing a new high level application programming interface which allows building SDN applications which are running on top of multiple networks around the world. The future vision is that there are multiple service providers who are offering the base network and on top of the network hypervisor is being built virtual networks called HyperNets which are then connecting users.

4.3 Security problems, drawbacks and challenges of SDN

Because many SDN solutions rely on centralized management of the whole network, there are also many new security issues and new possible threats to take into account [41]. OpenFlow supports using TLS (Transport Layer Security) for securing the connection between the controller and OpenFlow switches. Especially when operating in environments where the physical security of OpenFlow enabled devices cannot be guaranteed, this is certainly a necessary layer of security in OpenFlow networks. In addition to new security risks, SDN has some drawbacks and it also brings some new challenges to the design of computer networks.

The security threats on SDN are discussed generally in [42]. They are arguing that

security needs to be considered when designing software-defined networks and they are describing new fault and attack planes that SDN has and which are opening new possible threats. They are also pointing out that traditional networks have kind of a natural protection following from heterogeneity, closed devices, and decentralization of control plane. The following seven potential new threat vectors are described in the article and also possible solutions are proposed:

1. Forged or faked traffic flows.
2. Attacks on vulnerabilities in switches.
3. Attacks on control plane communications.
4. Attacks on and vulnerabilities in controllers.
5. Lack of mechanism to ensure trust between the controller and management applications.
6. Attacks on and vulnerabilities in administrative stations.
7. Lack of trusted resources for forensics and remediation.

Two types of denial of service attacks in OpenFlow enabled networks are considered in [43]. In the first one the attacker tries to exhaust the control plane bandwidth with carefully selected packets which a switch has to redirect to the controller. The other one aims at filling the memory of the switch with forwarding rules. Also the DoS mitigation strategies are proposed. They claim that the first type of attack can be prevented by rate limiting the number of packets sent to the controller. The second type can be prevented by using optimal flow idle timeout value which is small enough or by using a technique called flow aggregation.

The SDN concept brings a new aspect to security planning which is trusting to third party applications. An automated framework for static analysis of SDN applications called SHIELD is presented and evaluated in [44]. Also the malicious behavior of SDN applications is defined and categorized. SHIELD consists of three main components. Source code tracer, control flow analyzer and result manager. As a result, 10 malicious behaviors are detected and categorized by the target which are control channel, controller, other apps and system flow.

The key challenges that prevent building a carrier grade SDN network are discussed in [45]. Specially performance, scalability, security and interoperability issues are considered along with the proposed solutions. The performance of OpenFlow enabled networks is examined in [46] by measuring the switching times of current OpenFlow hardware. Also the forwarding speed and blocking probability model of an OpenFlow network is derived and validated by simulation.

5. SDN SECURITY APPLICATIONS AND SOLUTIONS

In this chapter we are discovering how network security could be improved with applications and solutions based on the SDN concept. We are getting to know existing security appliances built by the Internet community and vendors, and we are also describing ways to develop own SDN applications to carry out needed networking and security functions.

5.1 SDN test environment

We have built a little test environment to the security laboratory for testing different SDN solutions and help SDN related research in the future. The environment consist of three HP 3800 series switches and one HP 5900 series switch and a few desktop computers and rack servers acting as network clients and servers. We selected HP VAN SDN Controller to our test environment mainly because it has quite comprehensive documentation and we intended to explore SDN applications on HP SDN application store later during this research. Products from the same vendor are also supposed to co-operate well. The network diagram of the test environment is shown in Figure 5.1.

In this thesis, we are using the portion of network which is located physically in the security laboratory. In other words, we are using three HP 3800 switches connected to each other and controlled by HP SDN controller. The internet access is provided via Juniper SRX 220 router. The SDN controller is installed on the virtual machine running on top of the VMware ESXi cluster. Technical specifications of the controller are shown in Table 5.1. Because the main parts of the network are physically located in the security laboratory, the network structure is easily changeable during experiments.

On some of the experiments we are using software called Mininet instead of physical

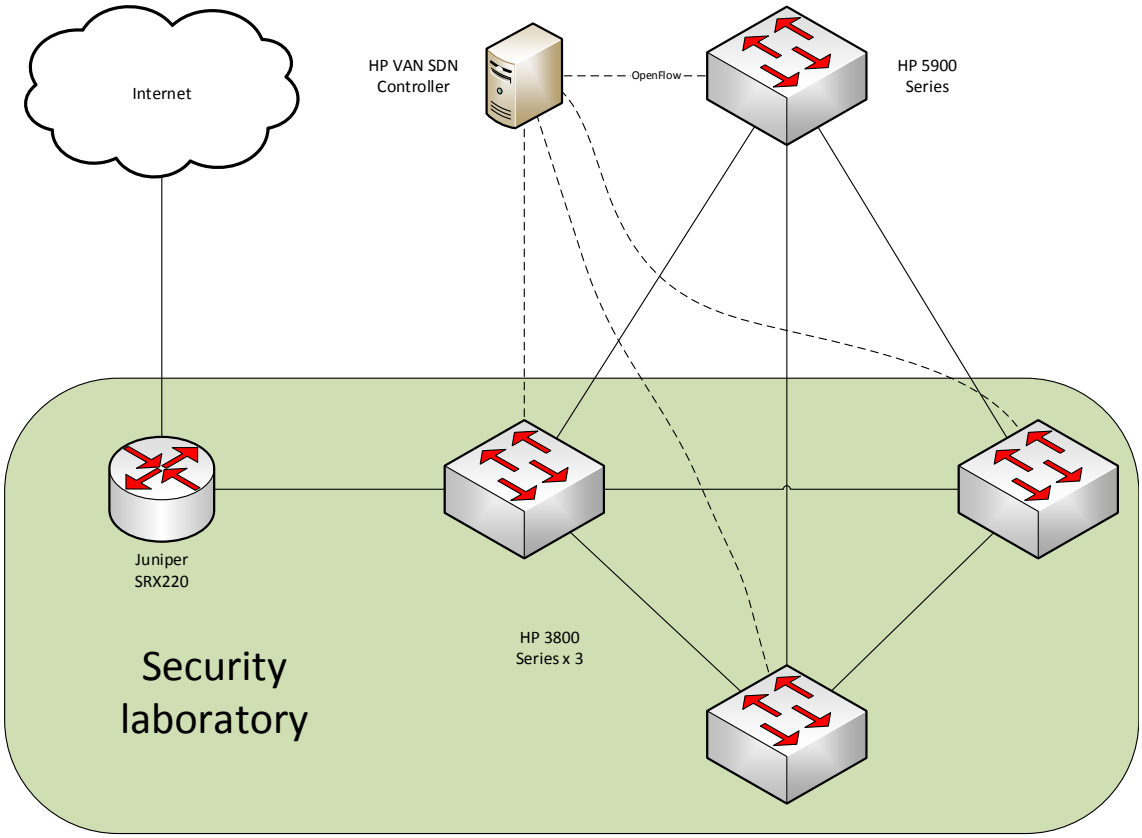

Figure 5.1 *SDN test environment.*

Table 5.1 *Technical specifications of the installed SDN controller*

HP VAN SDN Controller	
Controller version	2.6.8.0433
Operating system	Ubuntu 14.04.3 LTS
Memory	32 GB
CPU	16 cores
HDD	32 GB

switch fabric. Mininet is an application that creates a realistic, easily customizable virtual network including switches, hosts and SDN controller and it is meant specially to research, development and learning purposes. It is a really convenient way to start become acquainted with SDN solutions. [47]

5.2 Developing SDN applications

External SDN applications are programs that are built outside of the SDN controller. They are performing needed functions by instructing the controller through a specific application interface. Usually this interface is a RESTful API which is an easy way to use the controller without a considerable understanding of the internal function of the existing controller.

5.2.1 Example of using RESTapi

The simplest possible example of network modification using REST interface is the manual flow configuration with a simple Python script. In Figure 5.2 we have four OpenFlow switches connected to each other and one host connected to the bottom left and to the bottom right switch. The leftmost host (red) is sending ICMP echo request packets to the rightmost one (blue) and it is answering with ICMP echo reply packets. As we can see, the graphical user interface of HP's controller includes a nice OpenFlow topology feature which shows the physical structure of the whole OpenFlow network managed by the controller. In the topology view, we can select the source and the destination hosts and the controller will colorize the flow path showing the route that packets are using when moving from the source host to the destination host.

The reason why the flow is going through the top left switch and is not using the shortest possible path is that we have manually added two flow table entries to three switches and these will manipulate the flow's path. The default operation of the controller is to implement the basic L2 switch forwarding function and in this particular case it means that when clients are starting to communicate with each other, the controller is programming forwarding flows to the two switches on the bottom of the topology and the traffic is being forwarded along the shortest path. Now we have manually added new flows with higher priority and made the desired flow modification. The example Python code performing this modification is presented in Appendix A.

5.2.2 Nodecutter SDN application

The Nodecutter SDN application is a good starting point to get familiar with the HP VAN SDN controller and specially the REST API of the controller. It is an

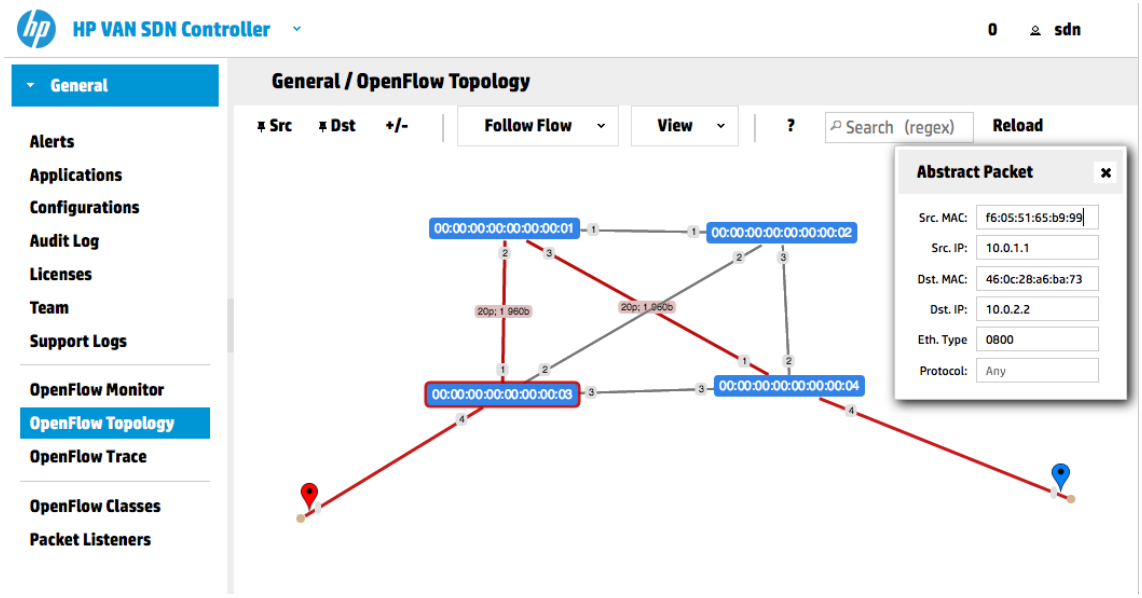

Figure 5.2 *Modified flow shown in topology view of HP VAN Controller.*

example application that is able to list all switches and devices in the current SDN network and allows the network administrator to block traffic of one specific node in the network. The application is written in Perl and it provides a simple web interface. The operation of Nodecutter is presented in a comprehensive way in [48] starting from producing a REST command with Curl.

5.3 Comparison of security applications in HP SDN App Store

HP is one of the big network device vendors which is seriously investing in SDN solutions. It is also one of the biggest companies offering commercial SDN solutions including physical devices, SDN controller and applications. In this section we are investigating the HP SDN App Store [49] and SDN applications found from there.

Currently there are around 35 applications promoted in the app store which are divided to different categories according to their use case. Unfortunately, after a quick exploration in the app store, it is obvious that the most of the applications seem to be more like a presentation of proposed application than a real working solution under continuous development. Also the documentations of the applications were quite compact in most cases.

The most interesting and finished SDN applications found in the app store and designed to improve network security by implementing a security function or providing

a better network management are listed in Table 5.2. When selecting apps for further examination, we observed the purpose of an app, the level of the documentation and the vitality of the application.

Table 5.2 *Security-minded applications in HP SDN App Store*

Application	Category
BlueCat DNS Director	Security
HPE Network Protector	Security
HPE Network Visualizer	Orchestration and Visualization

The products developed by HP Enterprise are clearly the most production ready solutions in the whole store. The BlueCat DNS Director is designed to work with other security solutions from BlueCat. There are also a couple of little community developed applications in the store which are implementing for example a network monitoring function for one specific network path. However, we did not select them for further examination mostly because the security relevance of them is so minor.

5.3.1 BlueCat DNS Director

BlueCat DNS Director [50] is an SDN application which is trying to enhance security by controlling DNS queries. The application is intercepting DNS queries which are targeted outside the corporate DNS server and the intercepted queries are redirected to BlueCat DNS server which will apply needed security policies. After that the DNS response is restructured so that it seems to come from the originally targeted DNS server. The structuring and redirection is done by SDN flow rules and the client cannot detect the interception.

That application allows the network administrator to take a control of all DNS traffic in a company network. According to the developer of the application, this is enhancing network security significantly specially in cases where company is accepting new device policies like bring your own device and the administrators cannot manage all the devices connected to the company network. That allows users to use any reachable public DNS server and for example DNS tunneling which is a security risk.

We deployed the application to HP VAN SDN controller and it seems to do what it proposes. It is configuring needed flow rules to the OpenFlow switches when

a computer connected to SDN network is sending a DNS request to a destination which is not classified as a trusted DNS server or the company's own DNS server. BlueCat DNS Director is designed to work together with BlueCat DNS server and BlueCat Threat Protection solution which is doing the actual policy processing to DNS queries. Because we don't have these services available, the experiment of the whole solution is not possible during this research.

5.3.2 HPE Network Protector

The HPE Network Protector is an SDN solution provided by HP Enterprise which is proposing to implement multiple automated security functions to SDN networks. It is an internal SDN application and it is installed directly into the HP VAN controller. The HPE Network Protector has following four main features:

- simple security for bring your own device (BYOD)
- enables automated network-posture assessment
- provides real-time threat detection across enterprise campus networks
- proactive IT management of threats.

The network protector application is designed to be used in all kind of networks from cloud to campus. It promotes to increase the visibility of threats specially in environments where bring your own device policy is allowed. It also allows prioritizing specific DNS traffic which is providing better availability to business critical applications while it is decreasing the priority of non-critical DNS traffic. The real-time threat detection is based on threat characterization using cloud based data base of malicious applications.The proactive IP management is performed with dynamic flow-based access control lists. [51]

The Network Protector is trying to push security to L2 access switches without additional security hardware when it is possible to block malicious traffic as early as it is possible. The application is operating as a reactive security application meaning that flow rules are created when there is active traffic arriving to a controlled network element. That needs the SDN application to be implemented as near to the devices as possible to minimize latency. The Network Protector is possible to scale to manage up to 2000 OpenFlow devices or clients. [51]

5.3.3 HPE Network Visualizer

The HP Network Visualizer is also an SDN application developed by HP Enterprise. Where as the Network Protector is implementing network functions to SDN network, the Network Visualizer is focusing to network monitoring problems. It is providing dynamic traffic capture and real-time monitoring allowing better network diagnosis. The main features of the Network Visualizer are:

- real-time visibility and diagnosis
- low cost, simple and automated troubleshooting
- fast transition from incident to fix and
- enhanced security.

HP claims that the Network Visualizer helps network troubleshooting, reduces the operational expenses and provides better network security. It is able to monitor multiple switches simultaneously using the leverage of SDN while in traditional network the monitoring of single switch has to be done individually. The typical use case for the Network Visualizer is in an office network when a network administrator starts to solve for example a connection problem. The Network Visualizer can be integrated with Active Directory, and the administrator can analyze the network traffic for example with the user name of the user who made the support ticket. [52]

5.4 Network operating systems

In the open source SDN community, there has been a tendency to develop more comprehensive solutions for managing the OpenFlow network instead of the SDN controller. That is probably because many of the first open source SDN controllers have been destined more or less for research and development purposes while there are also many production quality commercial controllers and SDN solutions available.

The Open Network Operating System (ONOS) [11] is one of the most exciting new solutions in the SDN ecosystem. It is an open source project lead by The Open Networking Lab and supported by multiple commercial organizations. The ONOS

is a software-defined networking operating system for service providers and it aims to provide high availability, high performance and scalability. ONOS is a distributed system, which provides abstractions to the northbound and the southbound interfaces, allowing to manage network in a high abstraction level.

The main concept in ONOS is to manage the communication between hosts by installing intents instead of low level forwarding rules. The high-level intents are compiled to the low-level paths between hosts which are then programmed to the switches. The ONOS is monitoring the status of the forwarding elements in the network in a real time. When ONOS detects that the link on the path is broken, it immediately calculates the another path for the traffic and configures network devices according that new route. An example of the intent configured between two hosts is shown in Figure 5.3. [53]

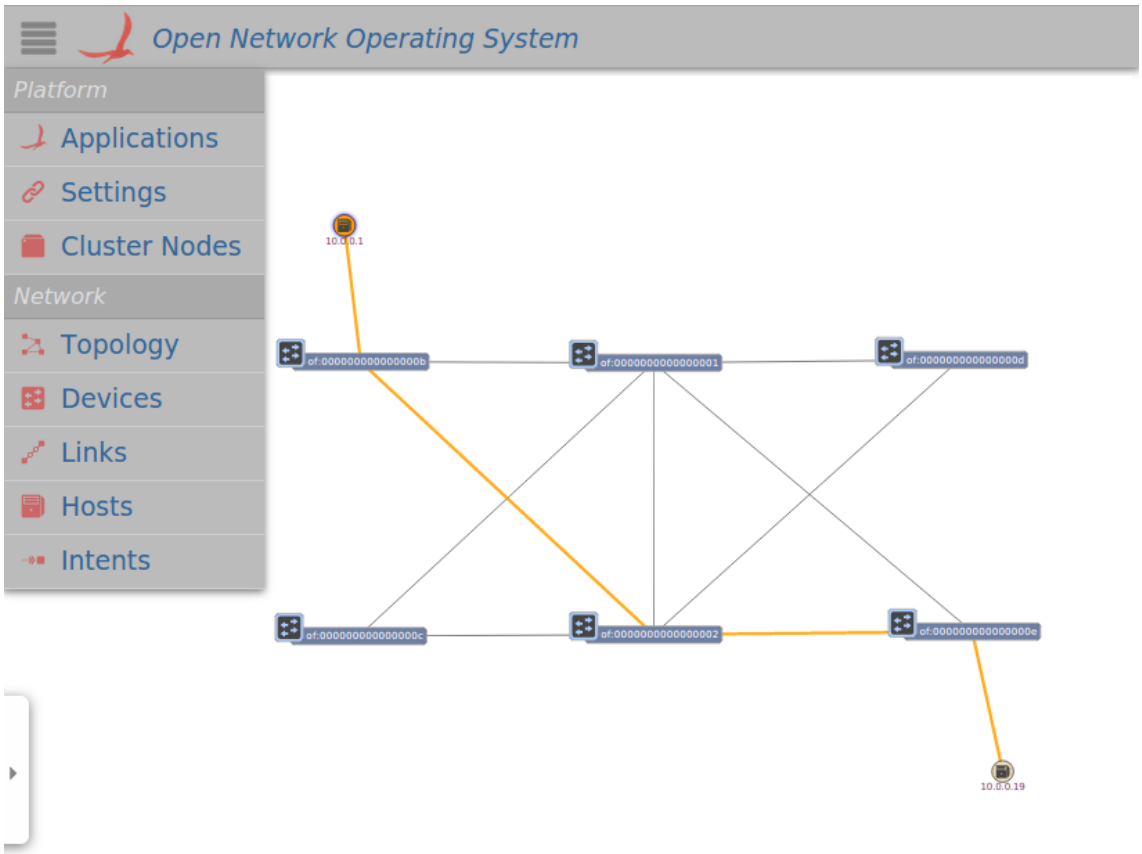

Figure 5.3 *Topology view of ONOS network operating system.*

We did some straightforward experiments with ONOS and the network build by Mininet. The ONOS seems to work as intended and it calculates the patch between hosts rapidly. The communication intents between hosts are working as promised

and when the link on the way of the current path goes down, ONOS immediately programs another path for the traffic. Even though ONOS is not presented as a SDN controller, the basic functionality of it seems to be quite identical to existing SDN controllers. All in all, the ONOS is a very good competitor for the current open source controllers like OpenDaylight.

6. DISCUSSION

In this thesis we have examined the SDN solutions and applications that are aimed to improve network security directly or by means of better network management and administration. We have also investigated the former research carried out in this topic and considered the current maturity of the SDN technology. In this chapter, we are discussing of the results of this thesis and the current status of the SDN technology.

6.1 Security enhancement with software-defined networking

As we have shown in Chapter 4, a lot research has been done around the SDN concept. Research community has developed frameworks, applications and solutions which are aimed to address different kind of security threats in computer networks. Nevertheless, many of the solutions are still more like a concept or a proposal than a concrete and finished solution and it is clear that the SDN concept needs more research during the following years.

The network management is another aspect which could be improved using SDN based network architecture. According to the research, centralized management and a better view of the network also improves the security of the network which is obvious. The management aspect is relevant specially in the big and complex networks. Also the highly dynamic and multitenant cloud environments are a good place to consider the SDN architecture.

When the management of the network needs less effort, the operational expenses are lower also. That allows a company to invest more to the network security than it has invested before.

6.2 Challenges in software-defined networking

Even though the SDN concept brings a lot of possibilities to improving the security of the network, it also brings some security issues that we have to consider. Because the SDN network usually relies heavily on the centralized management of the network, the management network between the controller and network forwarding devices is a security risk that needs attention. The traffic in the management network can be encrypted using TLS but it is still vulnerable for malicious actions. One possible threat against the management network is a denial of service attack and that is considered also in former research.

Another security issue in SDN network is the SDN controller. The controller is managing the whole network and it makes the forwarding decisions for the packets which do not mach to any of the existing flow rules in the switch. If it is possible and the network is critical, the high availability on the controller layer should be considered. For example the HP VAN SDN controller allows team configuration where multiple separate controllers are operating in the same team and one of them is the team manager. That increases the fault tolerance on the controller layer.

Because the network functions can be implemented with network applications in the SDN concept, there is yet another layer of security threats present. Specially when we are using a third party applications, we have to ensure that the applications are secure. The application security is present during the whole life of the network and possible security vulnerabilities need regular inspection and update of the applications.

6.3 Current state of the SDN concept

One of the targets of the thesis was to clarify the maturity of the SDN technology and consider if it is high enough to take the SDN concept into the production environment.

We have also been exploring the HP SDN App Store in this thesis. The purpose was to examine the SDN applications provided in the store and try to determine if these are finished enough to being implemented into the real production network. The result was that only a few of the applications are on the level that they provide something really remarkable to the SDN networks.

The status of the SDN concept itself, including the leading management protocol OpenFlow, is good enough to start utilizing the SDN network and it is a fact that the SDN is the de facto technology in future networks. Currently migrating from the conventional network architecture to the SDN concept needs sufficiently resources from the company to develop also the management layer functionality. Only in that way the company can really benefit from the SDN concept.

7. CONCLUSIONS

Software-defined networking is a new concept to manage and configure computer networks. The main idea of the SDN concept is to separate the controlling layer of the computer network from the network switches and centralize it to the SDN controller. That concept differs from the traditional networks where the network controlling and forwarding decisions are handled in the network switches where also the forwarding function is performed. The centralized management brings a new way to control network functionality with one single application instead of configuring tens or hundreds of devices independently.

In this thesis the SDN concept is considered specially from the perspective of network security and the security improvements are explored. There are a lot of research done in this topic and also a lot of the concepts, frameworks or solutions are proposed for an enhanced security but still more research needs to be done. Also the network vendors should invest more into the SDN development, so there will be more comprehensive solutions available in the market.

We also explored the SDN applications provided in HP SDN App Store and tried to find out if there are some interesting and promising SDN solutions available. Unfortunately, the market of the SDN applications is still quite small and the deployment of the SDN technology needs software development resources from the company.

The SDN will be the leading networking technology in the future but it still needs a little time to evolve. Currently the SDN controllers and the OpenFlow protocol seem to be quite ready but the development of the SDN applications and solutions needs research and development contribution.

BIBLIOGRAPHY

[1] G. Lee, *Cloud Networking: Understanding Cloud-based Data Center Networks*. Elsevier Science and Technology Books, Inc, 2014.

[2] Software-defined networking (sdn) definition. [Online]. [Accessed 16.4.2016]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition.

[3] J. Kurose and K. Ross, *Computer Networking: A Top-down Approach*, ser. Always learning. Pearson, 2013.

[4] The POX controller. [Online]. [Accessed 16.4.2016]. Available: https://github.com/noxrepo/pox.

[5] Hewlett packard enterprise: Software defined networking. [Online]. [Accessed 16.4.2016]. Available: http://hp.com/networking/sdn.

[6] HP VAN SDN controller. [Online]. [Accessed 16.4.2016]. Available: http://www8.hp.com/us/en/products/oas/product-detail.html?oid=5443917.

[7] Floodlight. [Online]. [Accessed 16.4.2016]. Available: http://www.projectfloodlight.org/.

[8] Trema. [Online]. [Accessed 16.4.2016]. Available: https://trema.github.io/trema/.

[9] OpenDaylight. [Online]. [Accessed 16.4.2016]. Available: https://www.opendaylight.org/.

[10] FlowVisor. [Online]. [Accessed 16.4.2016]. Available: https://github.com/opennetworkinglab/flowvisor/wiki.

[11] Open network operating system. [Online]. [Accessed 16.4.2016]. Available: http://onosproject.org/.

[12] OpenDaylight lithium overview. [Online]. [Accessed 16.4.2016]. Available: https://www.opendaylight.org/lithium.

[13] OpenStack. [Online]. [Accessed 16.4.2016]. Available: https://www.openstack.org/software/.

[14] E. Haleplidis, J. H. Salim, J. M. Halpern, S. Hares, K. Pentikousis, K. Ogawa, W. Wang, S. Denazis, and O. Koufopavlou, "Network programmability with forces," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1423–1440, thirdquarter 2015.

[15] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, June 2014, pp. 1–6.

[16] OpenDaylight summit: LISP for SDN and NFV. [Online]. [Accessed 16.4.2016]. Available: http://events.linuxfoundation.org/sites/events/files/slides/LISP_ODLSummit_2014.pdf.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. Available: http://doi.acm.org/10.1145/1355734.1355746

[18] Open networking foundation. [Online]. [Accessed 16.4.2016]. Available: https://www.opennetworking.org/.

[19] P. Goransson and C. Black, *Software Defined Networks: A Comprehensive Approach*. Elsevier Science and Technology Books, Inc, 2014.

[20] *OpenFlow Switch Specification Version 1.5.1*, Open Networking Foundation, March 2015, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[21] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, Aug. 2007. Available: http://doi.acm.org/10.1145/1282427.1282382

[22] Open vSwitch. [Online]. [Accessed 16.4.2016]. Available: http://openvswitch.org/.

[23] C. Pfleeger, S. Pfleeger, and J. Margulies, *Security in Computing*. Pearson Education, 2015.

[24] *Whitepaper: Layered Security: Why It Works*, SANS Institute, 2013, [Online]. Available: https://www.sans.org/reading-room/whitepapers/analyst/layered-security-works-34805.

[25] J. François, L. Dolberg, O. Festor, and T. Engel, "Network security through software defined networking: A survey," in *Proceedings of the Conference on Principles, Systems and Applications of IP Telecommunications*, ser. IPTComm '14. New York, NY, USA: ACM, 2014, pp. 6:1–6:8. Available: http://doi.acm.org/10.1145/2670386.2670390

[26] I. Hafeez, A. Y. Ding, L. Suomalainen, S. Hätönen, V. Niemi, and S. Tarkoma, "Demo: Cloud-based security as a service for smart iot environments," in *Proceedings of the 2015 Workshop on Wireless of the Students, by the Students, & for the Students*, ser. S3 '15. New York, NY, USA: ACM, 2015, pp. 20–20. Available: http://doi.acm.org/10.1145/2801694.2802140

[27] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Flowcover: Low-cost flow monitoring scheme in software defined networks," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, Dec 2014, pp. 1956–1961.

[28] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, pp. 122 – 136, 2014. Available: http://www.sciencedirect.com/science/article/pii/S1389128613004003

[29] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, May 2014, pp. 1–9.

[30] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks." in *Network and Distributed System Security Symposium*, 2013.

[31] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn security: A survey," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, Nov 2013, pp. 1–7.

[32] A. Y. Ding, J. Crowcroft, S. Tarkoma, and H. Flinck, "Software defined networking for security enhancement in wireless mobile networks," *Computer Networks*, vol. 66, pp. 94 – 101, 2014, leonard Kleinrock Tribute Issue: A Collection of Papers by his Students. Available: http://www.sciencedirect.com/science/article/pii/S1389128614001133

[33] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 127–132. Available: http://doi.acm.org/10.1145/2342441.2342467

[34] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, Oct 2010, pp. 408–415.

[35] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, February 2013.

[36] R. L. Smeliansky, "Sdn for network security," in *Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014 International*, Oct 2014, pp. 1–5.

[37] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 79–84. Available: http://doi.acm.org/10.1145/2342441.2342458

[38] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, pp. 1–13, 2009, [Online]. Available: http://sb.tmit.bme.hu/mediawiki/images/c/c0/FlowVisor.pdf.

[39] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "Devoflow: Cost-effective flow management for high performance enterprise networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 1:1–1:6. Available: http://doi.acm.org/10.1145/1868447.1868448

[40] S. Huang and J. Griffioen, "Network hypervisors: Managing the emerging sdn chaos," in *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, July 2013, pp. 1–7.

[41] *Principles and Practices for Securing Software-Defined Networks*, Open Networking Foundation, January 2015, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Principles_and_Practices_for_Securing_Software-Defined_Networks_applied_to_OFv1.3.4_V1.0.pdf.

[42] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 55–60. Available: http://doi.acm.org/10.1145/2491185.2491199

[43] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May 2015, pp. 1322–1326.

[44] C. Lee and S. Shin, "Shield: An automated framework for static analysis of sdn applications," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, ser. SDN-NFV Security '16. New York, NY, USA: ACM, 2016, pp. 29–34. Available: http://doi.acm.org/10.1145/2876019.2876026

[45] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, July 2013.

[46] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Teletraffic Congress (ITC), 2011 23rd International*, Sept 2011, pp. 1–7.

[47] Mininet. [Online]. [Accessed 16.4.2016]. Available: http://mininet.org/.

[48] Tutorial for creating first external SDN application for HP SDN VAN controller. [Online]. [Accessed 16.4.2016]. Available: http://networkgeekstuff.com/networking/tutorial-for-creating-first-external-sdn-application-for-hp-

sdn-van-controller-part-33-node-cutter-sdn-application-in-perl-with-web-interface/.

[49] Hewlett packard enterprise: SDN dev center. [Online]. [Accessed 16.4.2016]. Available: http://www8.hp.com/us/en/networking/sdn/devcenter-index.html.

[50] BlueCat DNS Director, SDN App Store, HPE. [Online]. [Accessed 16.4.2016]. Available: https://saas.hpe.com/marketplace/sdn/bluecat-dns-director-beta.

[51] *Technical white paper: HP Network Protector SDN Application*, Hewlett Packard Enterprise, 2015, [Online]. Available: http://h20195.www2.hp.com/V2/GetDocument.aspx?docname=4AA5-7852ENW&cc=us&lc=en.

[52] *Technical white paper: Being innovative with HP SDN Network Visualizer application*, Hewlett Packard Enterprise, 2016, [Online]. Available: http://www8.hp.com/h20195/v2/GetPDF.aspx/4AA6-3816ENW.pdf.

[53] *White paper: Introducing ONOS - a SDN network operating system for Service Providers*, ON.LAB, 2014, [Online]. Available: http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf.

APPENDIX A. EXAMPLE OF USING RESTAPI FOR FLOW MODIFICATION

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import json
import hmac
import json
import requests
login={"login":{"user":"sdn","password":"skyline","domain":"sdn"}}
host = "https://130.230.115.203:8443"
dpid1="00:00:00:00:00:00:00:01"
dpid2="00:00:00:00:00:00:00:02"
dpid3="00:00:00:00:00:00:00:03"
dpid4="00:00:00:00:00:00:00:04"

def qheader(token):
    return {'Content-Type': 'application/json', 'X-Auth-Token' : token}

def get_token(logindata):
    headers = {'Content-Type': 'application/json'}
    req = requests.post(host+'/sdn/v2.0/auth', headers=headers, data=json.dumps(
        logindata), verify='sdncertti')
    # make an exectpion if not 200 ok
    req.raise_for_status()
    # it loads the json from the string
    authtoken = json.loads(req.text)
    token = authtoken["record"]["token"]
    return token

def addflow(flow,dp,token):
    req = requests.post(host+'/sdn/v2.0/of/datapaths/'+dp+'/flows', headers=qheader
        (token), data=flow, verify='sdncertti')
    req.raise_for_status()
    return req

flow31 ="""{
        "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 59000,
                "idle_timeout": 30,
                "hard_timeout": 30,
                "match": [{
                        "in_port": 4
                }, {
                        "ipv4_src": "10.0.1.1"
                }, {
                        "ipv4_dst": "10.0.2.2"
                }, {
                        "eth_type": "ipv4"
                }],
                "instructions": [{
                        "apply_actions": [{
                                "output": 1
                        }]
                }]
        }
}"""

flow32 ="""{
        "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 59000,
                "idle_timeout": 30,
                "hard_timeout": 30,
                "match": [{
                        "in_port": 1
                }, {
                        "ipv4_src": "10.0.2.2"
                }, {
```

```
                        "ipv4_dst": "10.0.1.1"
                }, {
                        "eth_type": "ipv4"
                }],
                "instructions": [{
                        "apply_actions": [{
                                "output": 4
                        }]
                }]
        }
}"""

flow11 ="""{
        "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 59000,
                "idle_timeout": 30,
                "hard_timeout": 30,
                "match": [{
                        "in_port": 2
                }, {
                        "ipv4_src": "10.0.1.1"
                }, {
                        "ipv4_dst": "10.0.2.2"
                }, {
                        "eth_type": "ipv4"
                }],
                "instructions": [{
                        "apply_actions": [{
                                "output": 3
                        }]
                }]
        }
}"""

flow12 ="""{
        "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 59000,
                "idle_timeout": 30,
                "hard_timeout": 30,
                "match": [{
                        "in_port": 3
                }, {
                        "ipv4_src": "10.0.2.2"
                }, {
                        "ipv4_dst": "10.0.1.1"
                }, {
                        "eth_type": "ipv4"
                }],
                "instructions": [{
                        "apply_actions": [{
                                "output": 2
                        }]
                }]
        }
}"""

flow41 ="""{
        "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 59000,
                "idle_timeout": 30,
                "hard_timeout": 30,
                "match": [{
                        "in_port": 1
                }, {
                        "ipv4_src": "10.0.1.1"
                }, {
                        "ipv4_dst": "10.0.2.2"
                }, {
                        "eth_type": "ipv4"
                }],
```

```
                "instructions": [{
                        "apply_actions": [{
                                "output": 4
                        }]
                }]
        }
}"""

flow42 ="""{
        "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 59000,
                "idle_timeout": 30,
                "hard_timeout": 30,
                "match": [{
                        "in_port": 4
                }, {
                        "ipv4_src": "10.0.2.2"
                }, {
                        "ipv4_dst": "10.0.1.1"
                }, {
                        "eth_type": "ipv4"
                }],
                "instructions": [{
                        "apply_actions": [{
                                "output": 1
                        }]
                }]
        }
}"""

token = get_token(login)

addflow(flow31,dpid3,token)
addflow(flow32,dpid3,token)
addflow(flow11,dpid1,token)
addflow(flow12,dpid1,token)
addflow(flow41,dpid4,token)
addflow(flow42,dpid4,token)
```