INTERNATIONAL
HELLENIC
UNIVERSITY

Network management aspects in SDN

Panagiotis Apostolidis

SID: 3301130006

Supervisor: Prof. Andreas Pitsillides

Andreas
Pitsillides

Digitally signed by Andreas Pitsillides
DN: c=GR, st=Thessaloniki, l=Thermi,
o=International Hellenic University, ou=School
of Technology, cn=Andreas Pitsillides,
email=Andreas.Pitsillides@ucy.ac.cy
Date: 2016.02.11 19:06:24 +02'00'

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

DECEMBER 2015

THESSALONIKI – GREECE

Contents

Abstract

This dissertation was written as a part of the MSc in ICT Systems at the International Hellenic University.

SDN is a network approach that separates the control plane from the underlying systems that forward traffic to the selected destination. In SDN networks, management functions have to be effective to ensure performance and availability. In this master thesis, management functions of SDN networks are studied through the perspective of the FCAPS ISO framework. Existing studies, use cases and ONF SDN features are used as sources of information that help in identifying and evaluating management functions. The categorization of the evaluated functions according to the FCAPS model, allows us to argue that, through the perspective of FCAPS model, the Network Management of OpenFlow based SDN networks can be considered effective as soon as it is supported with traditional network management mechanisms and third party OpenFlow-companion solutions. Issues that still exist in limited number of management functions mainly concern carrier-grade networks. Further investigation, programmability effort and continuous development of OpenFlow is required.

I would like to thank my supervisor, Professor Andreas Pitsillides for giving me the opportunity to work with an interesting and emerging technology. His guidance and support played a very significant role to the development and completion of this thesis. I would also like to thank my family for being supportive and patient throughout the thesis work.

Panagiotis Apostolidis

10/12/2015

1 Introduction

SDN (*Software-Defined Networking*) is a network approach that aspires to simplify network design and operation by separating the control plane from the underlying systems, the data plane, that forward network to the destination. In SDN approach, the process of overseeing a network and taking corrective actions to ensure performance and availability, as has always been in traditional networks, continues to be challenging and of great importance.

In this study, the effectiveness of SDN management will be investigated in the context of the ISO network management model. The identified issues will be reported in order to contribute to future improvement of SDN management.

1.1 The problem(s) with Network Management

Since 1980s network management has been continuously evolving. However, certain problems in network management remain unsolved while new challenges in network management are also expected to arise from the deployment of SDN.

Network management of traditional networks is strongly affected by complexity and heterogeneity. Network operators and corporate network administrators quite often have to deal with low-level vendor-specific configuration activities in order to implement higher level network policies. The “closed” and proprietary integration of network devices prevent the adoption of best possible solutions in network management as the network operators remain “locked-in” by vendors. Network operators have to implement *frequent changes to the state or the configuration of a network* in order to enforce various high–level, and sometimes urgent, policies aiming to respond to security threats, data traffic needs or performance issues. Nowadays, network management is not yet able to provide sufficient mechanisms to address the wide range of network events that may occur [1].

Another issue in network management is the lack of network tools that would facilitate network operators to implement complex tasks and sophisticated policies. Most of the complex operations upon network devices, require from network operators the knowledge of vendor-specific *low level configuration commands*. The expertise of the network operators play an important role in implementing such configurations. However, expertise is not enough when network operators have to perform frequent configuration changes that affect the operational state of the network. The risk of misconfiguration increases and operators chose to use ad hoc scripts or external tools.

Today's networks consist of many interconnected, proprietary and vertically integrated devices. Due to vertical integration, vendors strive to fulfill the demand of network operators for deployment of new network protocols and functions. Innovation in network management is then limited resulting in lack of techniques and tools that would allow network operators to analyze and troubleshoot low-level configurations. Obviously, there is a need to improve the *visibility and the tools* that facilitate complex network policies.

1.2 Does SDN add more problems in network management?

In ONF (Open Networking Foundation) SDN architecture the network topology offers centralization which is supported by a relatively new communication standard, the OpenFlow. *OpenFlow with its potential and limitations* is expect to play a determinant role in the future of network management over SDN networks.

Software Defined networking is a new approach in computer networks that simplifies network design and operation by decoupling network control and data planes [2], and and is expected to reduce the cost of managing networks by providing programmable network services [3]. In SDN, network devices (switches, routers and firewalls) are simple packet forwarders while the intelligent part of control logic is implemented in the controllers. Some obvious benefits of this architecture are the centralization of network configuration and the ability to introduce new ideas in the network through software programs [1]. But what could happen *if control plane is misconfigured* while operating? Could this cause unexpected outages in network services?

Although SDN architecture has been discussed in many studies that explore the potential benefits of SDN in computer networks, there is *limited study* of the ways that SDN could affect positively or negatively the network management. The fact that the control plane determines *how the data plane handles and forwards* data traffic might create a need for a new management architecture for SDN networks. Prior to standardizing network management techniques in SDN, certain questions need to be answered:

The study of management functions SDN within the context of existing management model(s) can provide valuable information needed to answer certain questions such as:

- Can SDN protocols and proposed architecture support effectively network management?
- Is there any need for improving or upgrading existing protocols?
- Can existing management model(s) be implemented in SDN?

Today's network management practices needs more effective automation. The features of the SDN architecture, SDN companion protocols and programming effort must be able to support management automation. Also, the *need for facilitating Network Functions Virtualization* (NFV) is a new challenge for modern network management. ISPs need to reduce costs and enhance service delivery. Could this need be met by enabling NFV with OpenFlow-enabled SDN? [5].

1.3 The importance of Network Management

SDN networks or traditional networks that operate in enterprises or in ISPs are complex structures that require attention supported by certain management tasks. Faults in the networks do occur and must be recognized, isolated, corrected and logged. Service levels agreed by customers or users need to be ensured and monitored. Configuration changes on network devices should not affect the network performance. Network security and ability to track network resources need to be ensured as well.

There is close relation between management tasks and the following important factors:

- Cost saving
- Quality (reliability and availability)
- Revenue (ISPs)

As a consequence, network management is at the core of the business for ISPs. Enterprises can also benefit from effective network management. Savings in operational cost along with fast deployment of new services that maintain high quality level, can provide competitive advantage. The way that cost, quality and revenue factors are related to network management will be presented in details in the next paragraphs.

The *total cost of network equipment* ownership (TCO) consists of the equipment cost and the operational cost. Effective network management can reduce TCO by several ways. Management tools enable network operators to detect, isolate and troubleshoot problems or even predict errors (trend analysis). The prevention of network failures or the fast restoration of abnormal behavior ensures minimum impact on offered services. Reporting and analysis of performance enable network operators to achieve better resource allocation. Proper management tools can also reduce the need for higher qualified personnel since it can substitute or minimize human intervention.

Another important aspect of network management concerns the *quality of provided network services*. Some of the most important goals of network management is to ensure availability and reliability of the network. Those two attributes are associated with the network itself. End-to-end provisioning is supported by automated services that take over all the required configuration steps. In this way, the risk for misconfiguration is eliminated, providing an important contribution to higher network availability. Performance analysis can also prevent reliability issues. For example, through bandwidth trend analysis, the network management can predict the future expansion needs of network infrastructure. Actual outages in network availability can also be minimized by analyzing the root cause of failures. Since quality requirements are not met or general quality issues exist in provided network services, affected customers may switch to another network provider to do business with. The total network cost can also be affected by quality issues and poor fault management as the aforementioned factors may result in over-provisioning of network infrastructure.

Network management opens market opportunities to ISPs resulting in *revenue increase*. Service provisioning systems attract new customers that need quick deployment of new network services. In certain cases, network management enable ISPs to offer more competitive services that provide management capabilities. For example SLA monitor or accounting statistics offer to customers more valuable services. The cost savings due to effective network management enables ISPs (Internet Service Providers) or businesses to invest on new services that will offers them competitive advantage.

Summing up, it is clear that effective network management can provide tangible benefits. It plays an important role in reducing cost, and also in providing efficiency and effectiveness in network services. Network providers and ISP could benefit even more, since the network management is vital to increase revenue [4].

1.4 Thesis goal and outline

The goal of this dissertation is to investigate the effectiveness of network management in SDN networks and also identify any existing problems. The study of SDN network management will be carried out within the context of an existing network management model.

The objective of this master thesis is to investigate existing studies, use cases and SDN features in order to isolate and then evaluate network management functions. Prior to evaluation process, the SDN management functions will be categorized according to specific ISO model used for network management. Network management functions that cannot be implemented effectively or cannot be supported by SDN architecture, will be reported. The aim is to contribute to SDN's evolution in respect with the ISO network management model.

The structure of this dissertation is as follows:

- Chapter 2 contains background and related literature review. Important topics such as Network Management, Software-Defined Networking, OpenFlow protocol, OF-Config protocol and Network Functions Virtualization are discussed extensively.
- Chapter 3 contains studies, uses cases and important SDN protocol features that provide valuable information about network management functions. All sources follow a coarse categorization according to the FCAPS (Fault, Configuration, Accounting, Performance, Security) network management model. The investigation takes place in a manner that helps in isolating and later evaluating network management functions.
- Chapter 4 includes the methodology used for the evaluation of network management in Software Defined Networks. The problems that could affect the evaluation processes are discussed as well. This chapter also contains tables that show SDN's con-

tribution in management functions along with the evaluation of effectiveness for each function. All evaluated functions are categorized according to the FCAPS model.

- Chapter 5 contains a summary of main points in SDN management, concluding statements and recommendations.

2 Background

An approach of both Network Management and SDN backgrounds can help in the comprehension of SDN management functions that will be investigated in chapter 3. For this purpose, Network Management, the ISO model for network management, SDN architecture, SDN protocols and NVF (Network Functions Virtualization) will be discussed in this chapter.

2.1 Network Management

2.1.1 Definition of Network Management

Network management in general is the process of overseeing a network and supporting it by preventive or corrective actions that ensure performance and availability at a reasonable cost. Network management can also be defined as Operations, Administration, Maintenance and Provisioning (OAMP) of a network.

- *Operations* deals with overseeing procedures that ensure the smooth operation of the network. It also involves monitoring the network and addressing problems.
- *Administration* deals with for keeping records of network infrastructure.
- *Maintenance* is about repairing and upgrading network components, and also implementing measures that optimal operation of network devices
- *Provisioning* is concerned with the assignment of resources needed to support required services and users.

An ISO framework for network management, the FCAPS, categorizes the working objectives of network management into five areas: fault, configuration, accounting, performance and configuration. The FCAPS model will be discussed in section 2.1.3.

2.1.2 History of Network Management

In early 1980s, networking technology and the Internet Protocol which was an open protocol, enabled computers to communicate with each other. At the very beginning, computer networks looked like research artifacts and the “ping” tool was the ultimate tool for troubleshooting in networks [5]. As local area networks and public internet started growing up, the need for managing network equipment grew up. The need for managing networks with multi-vendor compatibility led to the development of open standards for network management. The specifications for SNMP (Simple Network Management Protocol) and CMIP (Computer Management Information Protocol) were issued in late 1980’s. Both management protocols were based on the agent-manager model which is a dominant management architecture until today.

SNMP was an innovative management model which eclipsed CMIP. The management information base (MIB) of SNMP was standardized. The development of standards enabled management systems to understand management information from different equipment.

In early 1990s, a new approach to implement distributed systems, CORBA (Common Object Request Broker Architecture) was introduced. Compared to SNMP, CORBA objects could provide complex management information while SNMP could represent structured information in association with network devices. In CORBA, management product vendors had to define custom specifications. Security issues was a major drawback for SNMP.

During 1990’s Web-Based Enterprise Management (WBEM) was introduced. WBEM uses CIM (Dommon information Model) and XML over HTTP and it is based on object oriented programming that standardizes management information [6].

During the last years, protocols and management information bases have evolved and useful practices from emerging technologies have been adopted. Management information is stored in relational databases and management functions operate and manipulate the databases. Nowadays, many management consoles are browser-based while modern management is generally policy-based and supported by automations. Attempts to adopt new technologies has not been very successful till now. Research continuous on certain emerging technologies such as virtualization.

2.1.3 ISO Functional model for network management (FCAPS)

The International Organization for Standardization (ISO) has defined a functional model for network management. According to this model, there are five functional areas of network management: Performance, Configuration, Accounting, Performance and Security (FCAPS) management.

Fault Management. Fault Management is concerned with detecting, isolating, correcting, and logging fault events in the network. A fault event in the network can be a device hardware failure, a software failure, or a network service failure that affect network normal operation and availability. Network monitoring is of great importance for fault management as it the tool that allows network administrators to detect the fault whenever it occurs and enable reaction. The network monitor includes alarm management and some advanced processing functions for alarm handling. . These alarms are the notifications triggered by the network components at fault occurrences and sent to network management systems. The detection of faults may require fault diagnosis, root cause analysis and troubleshooting. Fault management functionality also includes trouble ticketing and proactive fault management. Fault occurrences trigger network components to send notifications to network management systems. Certain open or proprietary protocols are used to collect information about the network. In traditional networks, the most widely used protocol is SNMP.

Configuration Management. Configuration management is concerned with the configuration information of network equipment, awareness of devices that belong to the managed network and tracking of configuration changes. Configuration information is gathered from all managed network elements and it is stored. In networks, configuration changes made to network elements, software updates, or changes in hardware, are sometimes the cause for network issues. In these cases, configuration management should be able to roll back the network conditions in the previous known and operational state.

Accounting Management. Accounting management deals with gathering information about the usage of network resources or network services and correlating of this stored data with users or devices. The gathered information is used for accounting purposes such as billing. Accounting management is supported by the functions that help organizations or network providers to get credits for the network services they provide. The gathered information (usage data) is usually based on volume, duration and offered quality. For example, accounting measurements may include: traffic (volume), minutes of phone calls (duration), use status of QoS (quality). If there is no charging need in the organization, network managers may need to track usage of network resources by user or by class, in order to ensure smooth operation of the network. Accounting systems can be supported by network standards and protocols such as RADIUS, TACACS+ and SNMP in order to fulfill authentication, authorization, accounting and capacity planning requirements [6].

Performance Management. Performance management aims to ensure that network performance will be maintained at acceptable levels. It deals with the performance of different network components. It quantifies, measures, reports, analyzes and controls network elements such as routers, switches, links, hosts etc. Important measurements (statistics) used by performance management are utilization, throughput, delay, and error ratio. The SNMP plays a key role in the performance management of traditional networks. An SNMP management system provides active monitoring of the network and it also sends alerts to administrator in case of out-of-band performance parameters. Trends in statistics and active monitoring, enable network administrators to prevent future reliability or availability issues.

Security Management. Security Management is concerned with securing the network resources and services from threats, and also with securing the management system itself. The security threats usually target end users (PCs) and IT systems (web servers, databases, etc.). A network must be secure enough to deter such actions that could have negative impact in the areas of confidentiality, integrity and availability. For example, viruses and worms could corrupt or destroy systems or files, denial-of-service attacks could overload portions of a system or network, or hackers could obtain control of systems connected to the network. Securing the network and the management system from threats requires a comprehensive approach of security management. Network managements can use certain techniques that reduce the risk of minimizing the effects of attacks that could be implemented via the network infrastructure. Such techniques may depend on [4]:

- intrusion detection systems
- policies that limit unexpected usage of resources
- blocking of potentially dangerous traffic destinations
- “honeypot” computers.

Security functions have to underpin all other functional areas (Fault, Configuration, Accounting and Performance) to allow security to be effective [9]. The interactions of the FCAPS functions can be seen in figure 1.

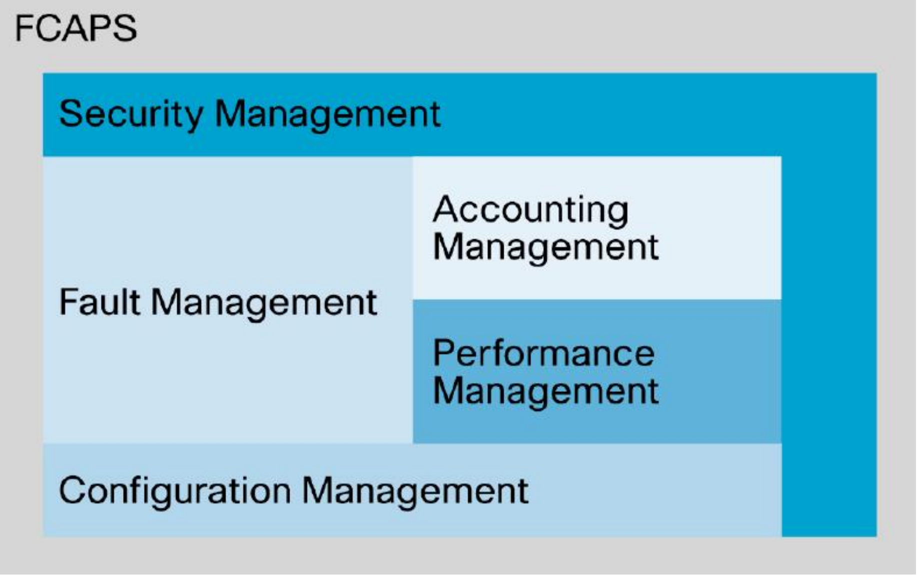


Figure 1: Interactions of the FCAPS functions [9]

2.2 Software Defined Networking

Software-Defined Networking (SDN) is a new approach in networking and it is based on the concept of decoupling the part of the network that makes decisions about data forwarding (control plane) from the underlying systems that forward the data to their destination (data plane).

The migration of control plane into computing devices (SDN controllers) allows not only the abstraction of the underlying infrastructure but also and its use for applications and network services. The Software Defined Network can then be treated as a logical or virtual entity [2].

The SDN can also be considered as set of techniques that facilitate design, operations and delivery of network services in a dynamic, deterministic and scalable manner [10].

2.2.1 History of SDN

The first approach of separating the forwarding and the control planes was followed by ATM switches in late 1990s. A set of open and programmable ATM interfaces were proposed for ATM switches aiming to separate the control software from the hardware. This work was followed by the creation of GSMP (General Switch Management Protocol). GSMP was allowing a central controller to establish connections with ATM switches. The fact that MPLS (Multi-Protocol Label Switching) in late 1990s, was establishing semi-static paths for forwarding flows in traditional routers, can be considered as a small step towards SDN approach.

Network access control methods such as RADIUS and COPS can also be viewed as SDN precursors because they allowed networking attribute changes. Both methods were based on the identity of the computer resource that required connectivity. SNMP orchestration that was used to automatically configure network equipment is also considered as a precursor of SDN (late 2000s). The ForCES work in 2003, was one of the proposals that recommended the decoupling of control and forwarding planes. Another important step towards SDN approach was the network tools that were used in virtualized environments to perform network reconfiguration.

The Ethane project (2007) and its predecessor SANE utilized a centralized controller in networks where the control plane was decoupled from the data plane. The operational deployment of Ethane project in Stanford University had just triggered the creation of the original OpenFlow SDN API [47].

In 2008, OpenFlow an open standard that supports SDN approach was born. OpenFlow defined how the data plane and the control plane will be decoupled, and the communication protocol between these two planes. The SDN term appeared in a research community and since 2011 SDN begun to affect networking industry.

2.2.2 SDN architecture

During the last years, research communities and industry have both proposed and developed a variety of architectural approaches. The industry-led development is mainly based on proprietary protocols. Research communities support and develop an open-source approach which is documented by the Open Network Foundation (ONF). This dissertation will focus on the Open-SDN approach which is based on the OpenFlow standardization. The reasons for focusing on ONF Open-SDN is the fact that research can be facilitated by a variety of open source software and tools while it is increasingly accepted and implemented by network equipment vendors. For example, in 2013, Google announced the deployment of a hybrid OpenFlow based SDN network. Also, OpenFlow is increasingly supported by most SDN-capable network devices manufactured by HP, Cisco, Alcatel, Juniper and NEC.

The separation of control and data planes within a network is the main characteristic of SDN. In traditional network architecture, both planes are incorporated within the network elements (switches, routers) while in SDN architecture the control plane is abstracted and becomes centralized (figure 2).

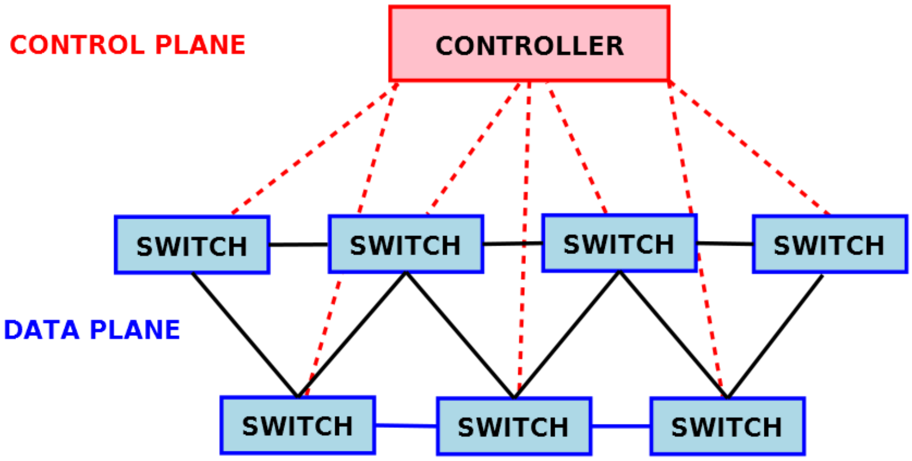


Figure 2: Control plane and Data plane in SDN

The data plane is responsible for forwarding incoming packets. The forwarding functionality is based on flow tables that define actions for the incoming packets considering certain packet characteristics such as source IP address and MAC address. A packet that arrives in the SDN switch may be dropped, forwarded or modified. For example, packets with source IP that doesn't match the address table within an SDN switch will be dropped. A packet that matches the source ip address and the DSCP value will be forwarded to a certain hardware port with queue priority as indicated by the lookup in the tables. In the same way, if the packet is multicast, it will be forwarded to more than one port. Packets that need further processing, are forwarded to the control plane. Network devices that belong to the data (forwarding) plane are the SDN enabled switches, routers and firewalls.

The control plane dictates the data plane behavior by programming or configuring it. Logic, algorithms and protocols needed to configure the data plane are all part of the control plane. The control plane has the knowledge of the network within its domain. It builds the flow tables for the data plane devices, runs the routing and switching protocols, and ensures the synchronization of the flow tables of the data plane devices. In SDN, the control is completely separated from the data plane devices and it resides in SDN controllers.

The communication between control plane and data plane is defined by the OpenFlow standard (Figure 3) which, in the SDN architecture, is considered either as a southbound interface (SBI) or as an application programming interface (API). OpenFlow allows SDN controllers to access and also manipulate the data plane of the network devices. With OpenFlow, the controller is able to add, remove or change the internal flow tables of SDN data plane devices and consequently maintain or change network behavior according to controller’s instructions.

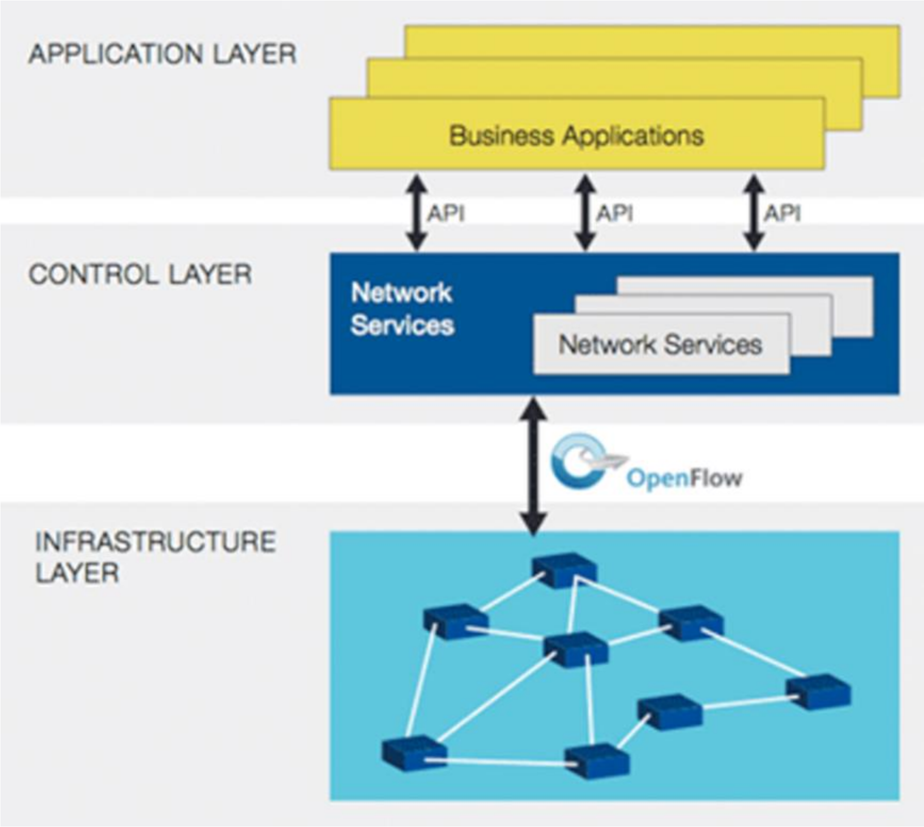


Figure 3: SDN architecture [2]

As can be seen in figure 3, the basic components of SDN architecture are the data plane (infrastructure layer), the control plane (control layer) and the application plane (application layer). The application plane includes applications that run above the control plane (SDN controllers). The applications are programs that communicate all the desired network requirements and behavior via the controller dependent API (Northbound Interface). Each SDN application contains the logic that has to be implemented to the network and the NBI driver(s).

2.3 OpenFlow

The OpenFlow standard is a communications protocol that defines the content and the format of messages exchanged between OpenFlow controllers and the OpenFlow switches. In open SDN networks, OpenFlow is the southbound interface that defines both the behavior of the data plane and the communication between data plane and control plane. The adoption of OpenFlow in SDN networks allows an SDN network to operate with switches manufactured from different vendors.

2.3.1 History of OpenFlow

The first experimental OpenFlow versions were released in 2008. OpenFlow version 1.0 (Dec 2009) was the first OpenFlow release with official vendor support. The specification evolved passing through releases 1.1 (Feb 2011), 1.2 (Dec 2012), 1.3 (June 2012), 1.4 (Oct 2013) up to the current one (1.5 Dec 2014). Each prior version has addressed problems with previous releases and also provided incremental functionality [11]. In this dissertation, OpenFlow specifications study will be focused on the latest OpenFlow release because earlier releases are missing features that can play a significant role in the management of SDN networks.

2.3.2 OpenFlow basic architecture

A basic OpenFlow system is consisted from one or more OpenFlow Switches that communicate with SDN controller. The OpenFlow messages specify the behavior of the switch which responses to the commands sent by the controller. A simplified architecture of the OpenFlow approach can be seen in Figure 4.

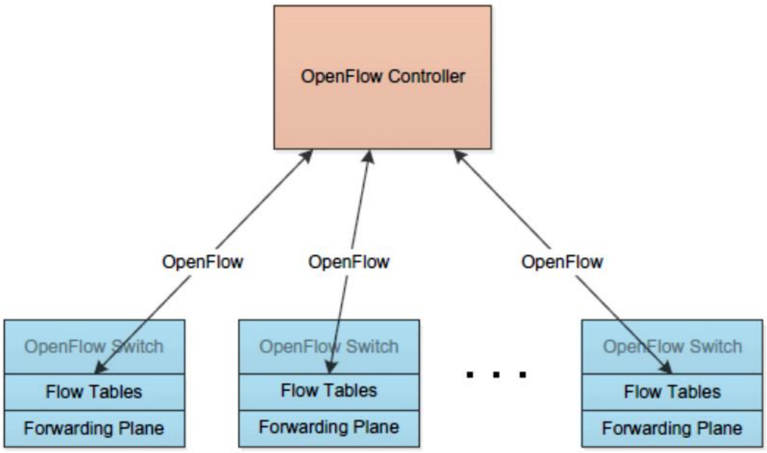


Figure 4: Simplified OpenFlow architecture

The basic operations of OpenFlow can be described as follows:

- The controller maintains the desired flow table entries in the switch.
- The switch matches incoming packets to flows and then treats them according to flow table entries.
- Non-matched packets are sent to the controller and the controller sends back to the switch new entries that allow switch to handle the packet

2.3.3 Open Flow Switch

The main components of the OpenFlow switch (Figure 5) are:

- The OpenFlow communication channel
- The flow tables
- The group table.

OpenFlow messages are exchanged between the controller and the switch through the OpenFlow Channel. The OpenFlow protocol allows the controller to modify proactively or reactively the content of the flow tables (add, update, delete). The records within the Flow Tables are called Flow Entries and they consist of *matching fields*, *counters* and sets of *instructions* for matching packets.

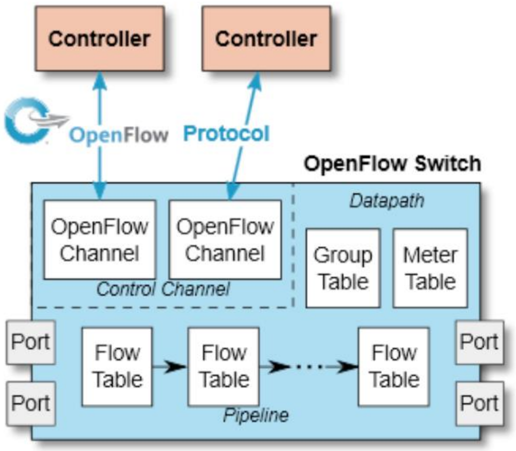


Figure 5: The main components of the OpenFlow Switch [12]

Each time a packet arrives to the switch, the matching procedure is prioritized according to the order of the flow tables. Within flow tables, the matching procedure starts by checking the first flow entry and then the next ones. When a packet can be matched with a flow entry, the instruction defined in the specific entry is executed. The outcome for non-matching packets may vary depending on the table-miss entry. For example, the packet will be dropped, it will follow the matching procedure in the next flow table, or it will be forwarded to the controller. The instructions are associated with flow entries and define either the modification of the pipeline processing or the actions that should be taken each time a packet is matched with a flow entry. Such actions could be packet modification, packet forwarding or further processing by the group table. The Group Table entries are associated with certain number of actions (Action Buckets) such as flooding, link aggregation, multipath and fast-reroute. The main components of entries in Group Tables are: The Group Identifier, The Group Type, The Counters and the Action Buckets.

Some additional important terms frequently used in OpenFlow 1.5 are the following:

- *Control Channel:* The components of an OpenFlow switch used for communication with controllers.
- *Counter:* A major element of statistics. Their typical role is the counting of bytes and packets passing through OpenFlow elements.
- *Data path:* The Aggregation of all components involved in traffic manipulation.
- *Header:* The information contained to a packet that is used for matching purposes.

- *Metadata:* Maskable register for transferring information between tables
- *Meter:* A switch element used for measuring and controlling the rate of packets.
- *Pipeline:* The aggregation of flow tables within an OpenFlow switch, used in packet matching, forwarding and modification processes.
- *Port:* Ingress or egress of OpenFlow pipeline. Ports can be physical, logical or reserved ones.
- *Queue:* Being used for scheduling packet forwarding and provide Quality of Service.

The packet flow through the OpenFlow pipeline can be seen in figure 6. The flow tables are sequentially numbered and the pipeline processing is implemented in two stages. The first stage is the ingress processing and starts at the first flow table. The outcome of this first matching attempt will determine whether the packet will be further processed with the next flow tables or will be forwarded to an egress port. The second stage which is optional, is the egress processing. Since egress flow table(s) exist, the packet will be processed by the outgoing port.

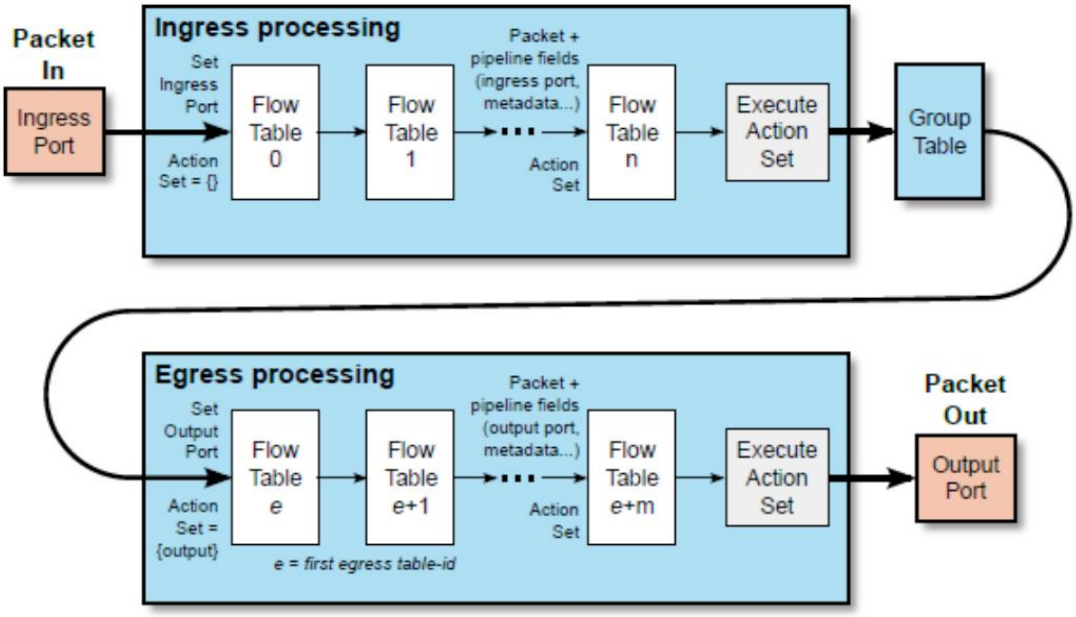
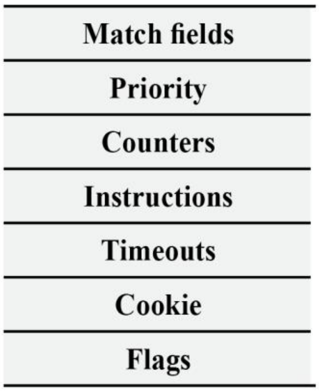


Figure 6: Pipeline processing [12]

The main components (Table 1) of flow entries contained in flow tables are:

- *Match fields:* Contain incoming port and packet headers. Optionally, metadata may be contained as well.
- *Priority:* Flow entry precedence.
- *Counters:* Change according to matching status.
- *Instructions:* Affect action sets or pipeline process.
- *Timeouts:* Time thresholds that specify the duration of a switch flow.
- *Cookie:* Data value used by controller to alter flow entries being affects by statistics or flow modifications.
- *Flags:* Used for changing the way of managing flow entries

Table 1
Components of entries within flow tables.



Match fields
Priority
Counters
Instructions
Timeouts
Cookie
Flags

As OpenFlow evolves, the ability of an OpenFlow switch to match the incoming packets against various fields is improving. The latest OpenFlow Switch Specification (version 1.5.0), refers to 40 header fields that can be matched with flow entries. For example certain header fields include:

- Ingress – Egress switch ports
- Source and destination Ethernet addresses
- IP protocol type
- Source and destinations IP addresses
- VLAN parameters
- DSCP / ECN / ToS values
- MPLS information

Matching fields have a varying length of up to 128 bits. As can be seen in Appendix – Table 1, almost all matching fields except for ingress/egress port headers, are related to Layer 2, Layer 2.5 (MPLS), Layer 3 and Layer 4 of network protocol stack. All OpenFlow switches must support at least the match fields being noted as "required" in APPENDIX – Table 1. The value "ANY" can be used as a field type that matches all the header fields.

2.3.4 SDN OpenFlow Controller

The SDN controller plays the role of the operating system (OS) for the network and implements the network control plane. The controller implements the network policy, controls the SDN devices that belong to the network and provides the northbound API for use by applications [11]. The southbound API is OpenFLow and it allows the interfacing between the controller and the OpenFlow switches.

The main components of the SDN controller can be seen in Figure 7. The modules section shows the main functions which are discovery of devices and topology, device management, statistics and flow management.

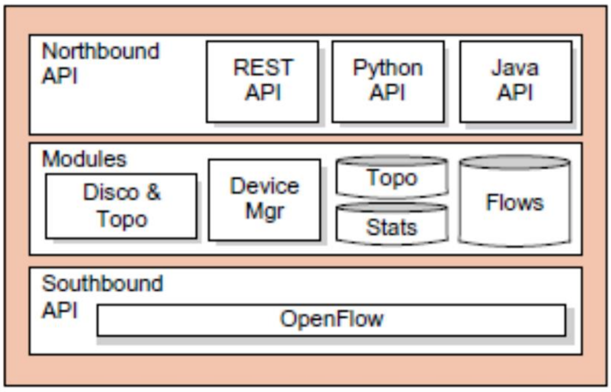


Figure 7: OpenFlow Controller components [11]

Although the southbound API has been standardized, the northbound API varies depending on the controller used. Such northbound APIs are: Python (Ryu Controller, POX Controller), Java (Floodlight Controller), C++ (NOX Controller) and REST (Floodlight Controller). Depending on the controller type, the northbound API can be a low level interface that allows developers to gain access to network devices, or it can be a high level API that provides developers with an abstraction of the network.

2.3.5 Communication between OpenFlow devices

There are three types messages exchanged between OpenFlow Switches and OpenFlow Controllers

- *Symmetric messages:* These messages are “hello” and echo messages between switch and controller, used for presence indication and monitoring communication latency.
- *Asynchronous messages:* These messages are sent by the switch in case a packet cannot be matched to a flow table entry or if the switch has to inform the controller of a change related to port, error or flow removal.
- *Controller-to-switch messages:* These messages are sent by the controller whenever the controller needs specific information from the switch or the switch has to modify flow tables.

2.4 OF-Config

The OpenFlow Management and Configuration Protocol (OF-Config) is a companion protocol to OpenFlow that allows OpenFlow switches to be configured remotely. Compared to OpenFlow’s time-scale operations (e.g. flow modification), OF-Config performs operations such as disabling a switch port on a slower time-scale.

Since OF-Config’s implementation is directly related with configuration management, it will be discussed in 3.2 (Configuration Management in SDN networks).

2.5 SDN and NFV

The Network Functions Virtualization (NFV) is the concept of implementing the functionality of network devices with software. This means that generalized components of a network such as routers, firewalls, switches or intrusion detection systems (IDS) can be implemented in software that runs in multi-purpose server appliances. The servers used for NFV purposes include multicore processors and network controllers. They were initially deployed to support virtual machines and virtual switches in data centers.

Decoupling the network functions from proprietary dedicated hardware can result in reducing capital and operational expenses (CAPex & OPex) in several ways:

- No need to purchase dedicated hardware. Standard x86 servers can be used for NFV implementation.
- No need for network administrators to over-provision datacenters. Virtual machines can be run on different physical servers according to required needs (e.g. bandwidth expansion).

The NFV was first presented at the SDN and OpenFLOW Congress in 2012. NFV and SDN can be each one complimentary to the other one or sometimes they may be overlapping. NFV can be partially be implemented by SDN and other non-SDN technologies [11].

3 SDN management and FCAPS

The study of management functions in SDN networks through the perspective of the FCAPS framework can provide strong indication whether SDN networks can be managed effectively. Since, OpenFlow is a standard that defines the features and the behavior of the network equipment used by the data plane, its study can reveal strengths and weaknesses related to network management. The study of OpenFlow based SDN within the context of the FCAPS model can provide valuable information needed to answer certain questions such as:

- Can OpenFlow support or leverage network management in SDN networks?
- Does SDN network management need OpenFlow accompanying protocols?
- Can FCAPS information be queried by the network management software in SDN networks?

Existing literature, SDN use cases and OpenFlow Switch specifications can be studied through the FCAPS perspective aiming to collect all this information that will help in answering the aforementioned questions.

3.1 Fault management in SDN

The fault management in SDN network must be able to detect, isolate, correct and log fault events in the network. Typically, a link failure can result in loss of certain hosts, loss of network services or network separation. A peculiarity in SDN networks is the fact that a link failure can also affect the communication between the SDN controller and the SDN switches. The study of scenarios where the failed link is not just affecting network data flows but also the OpenFlow control channel, can help in evaluating a critical aspect of fault management.

For in-band OpenFlow networks, one physical connection is used for both control channels. Link connections may carry data traffic and communication messages being exchanged between controllers and switches (Figure 8a). In the case of the out-of-band OpenFlow networks the control channel and the data traffic channel operate through separate physical paths (Figure 8b).

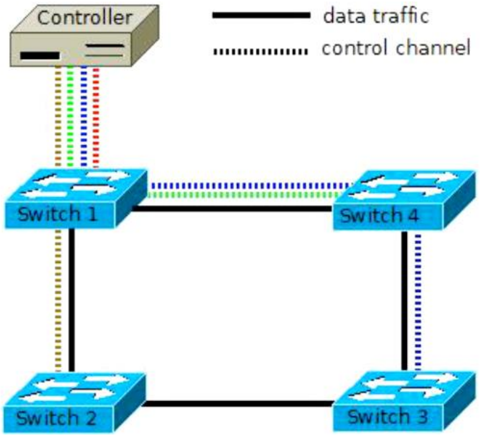


Figure 8a: In-band OpenFlow control.

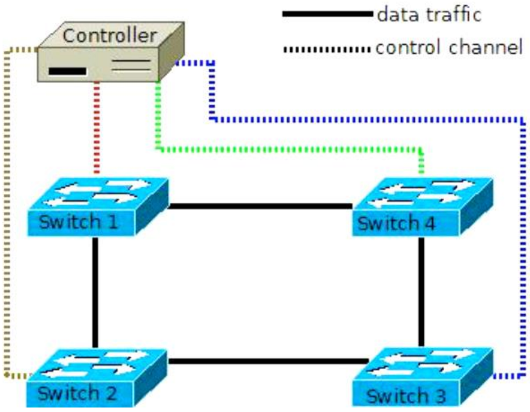


Figure 8b: Out-of-band OpenFlow control.

For both topologies, certain protection and restoration techniques have been proposed to address link-fail events. The network protections techniques deal with reserving networks resources (e.g. paths) prior to link-failures. Restoration techniques have to do with the selection of certain replacement resources that can be allocated after a failure occurs. Such protection and restoration approaches will be discussed for both OpenFlow control topologies.

The failure recovery problem for *in-band* OpenFlow topology is addressed by the work of [13] which studies the recovery of failed links through a carrier-grade perspective. In the network of figure 8a it is assumed that the controller communicates with switch-2 through switch-1. A link failure between switch-1 and switch-2 (Figure 9a) will disconnect switch 1 from the controller. According to the proposed *restoration* technique, the controller can detect the loss of communication with switch 2 either through port status messages on neighboring switches (switch-1 and switch-3) or through Echo request timeout. Then, the network controller can update the flow tables of switches 1, 4 and 3, to reroute the control traffic to switch-2 (Figure 9a). However, the controller still cannot communicate with switch 2 because the control channel is associated with the port on the failed link. The authors of [13] propose that switch-2 should be configured to flood the control traffic in order to achieve communication with the controller after the link failure through switches 3-4-1.

Concerning the proposed *protection* technique, an OpenFlow feature, the Group Table can be used to enable the switches themselves to react in the case of a link failure. By choosing the fast failover Group Type in the Group Table (2.3.3), the switch is enabled to change forwarding [12] without involving the controller. Each “fast failover” Group entry contains at least two action buckets in defined order. The first action describes the behavior (packet treatment) under normal operation while the second action defines the behavior when a state–change occurs. For example, when the status of the monitored port changes from “up” to “down”, the second action bucket is enabled. In the case of network shown in Figure 9a, the switch-2 will detect the “down” status of the port connected to the failed link and it will forward the traffic through switch-3 and the predefined ports of involved switches. Once the communication between the controller and the switch is restored through switches 3-4-2, the controller will update the flow tables in switch-2 and data traffic communication will be restored as well. The protection of data traffic can be achieved by the use of the Group Table feature that already has been discussed in control channel protection technique.

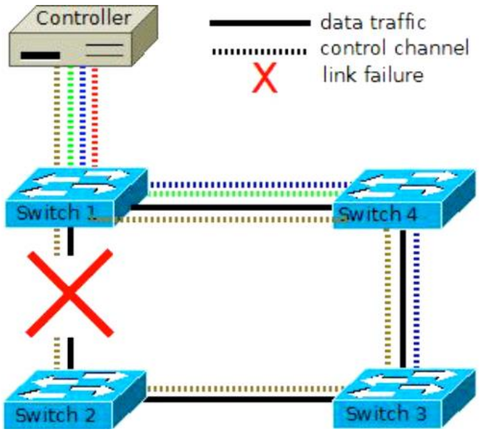


Figure 9a: In-band link failure

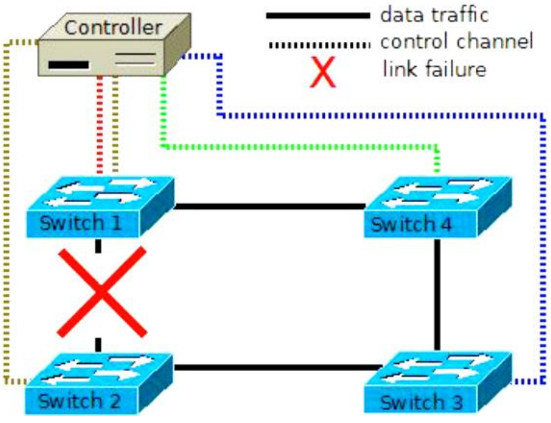


Figure 9b: Out-of-band data link failure

In the case of *out-of-band* control, protection and restoration techniques for OpenFlow networks are discussed in [14] and [15]. Since the controller remains connected with the switches involved in link failure events, the *data recovery* procedure is simplified if compared to the case of link failure in in-band topology. As soon as the link between switch-1 and switch-2 fails (Figure 9b), the affected OpenFlow switch notifies the controller about the link status change and the controller reacts immediately aiming to achieve fast flow *restoration*. Flow entries being related to failed link are deleted from switch-1 and switch-2 and new flow entries are installed to switches along the restoration path. In any case the controller have to be aware of the network topology or the spanning tree protocol must be used in the network.

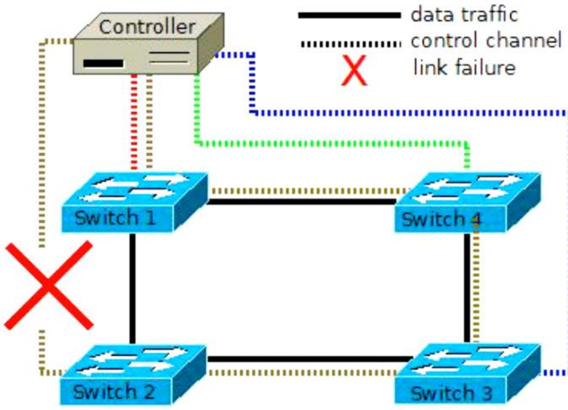


Figure 10: Out-of-band control channel failure. The new path is in-bound.

According to the out-of-band data *protection* scheme, there is no need for the failure-affected switch to notify the controller in order to establish a restoration path. The alternative data path is precomputed and no further flow entry deletions, modifications or additions are required after the failure event. Fast recovery requires an end-to-end continuous monitor of the path. The protection mechanism can depend on the BFD (Bidirectional Forwarding Detection) network protocol which provides fault detection on lower levels of network stack layers.

Concerning the *out-of-band* control plane resiliency there are two proposed solutions [15]. The rationale behind the first solution is the same with the one discussed in the in-bound recovery techniques. The control channel between the controller and the switch is re-established through a separate data path between the controller and the switch-2 implementing an in-bound topology (Figure 10). The second proposed solution is based on the use of two controllers. Once the control channel between a switch and the first controller fails, the pre-configured control channel between the second controller and the switch is activated.

A critical aspect of failure recovery is the time needed for a network to restore the operational state. In carrier networks, 50ms is considered a tolerable value of time needed for network recovery [16]. Experiments with OpenFlow recovery mechanisms [13] [14] have shown that only the protection schemes could allow network recovery within an interval of 50ms in carrier-grade networks. Recovery with restoration techniques can meet the target of less than 50ms only in smaller networks [15].

OpenFlow and SDN are able to provide flexibility for programming networks without being constrained by distributed network protocols. However, reactive response to network events must always depend on the OpenFlow controller. Time delays needed for controller notification and updating of forwarding rules, along with signaling overheads, make the implementation of network resiliency neither efficient, nor easy [17]. The OpenFlow switch specification 1.4 introduced the fast-failover mechanism that allow switches to react locally using more than one action bucket within a flow entry. However, in complex network topologies the protective scheme requires complicated scenarios with path reservations. This approach then becomes inefficient from the aspect of resource allocation.

The OpenFlow native mechanism for detecting failures is a LOS (loss of signal) indication that operates at the data plane. This mechanism doesn’t provide any information to switches located along the affected data path. Since the path protection needs an end-to-end mechanism to detect path failures, the BFD protocol can be used in the whole OpenFlow network to provide fast failure detection [15]. Another study [19] shows the benefits of implementing the BFD protocol in SDN networks. A proposed failover scheme with precomputed link-paths by the controllers and BFD sessions in every link can reduce significantly the recovery time while it is not affected by network size and path length.

An alternative way to implement fast failure detection with OpenFlow networks is the implementation of Link Layer Discovery Protocol (LLDP). The LLDP allows an OpenFlow SDN controller to detect failed nodes or link failures and trigger path restoration actions. A study [18] have shown that this approach has scalability limitations caused by the volume of network’s LLDP messages that reach to the controller and must be processed requiring CPU resources. The proposal of [18] for overcoming this issue contradicts the SDN concept. The authors propose a slightly relaxed separation of control plane and data plane operations. This OpenFlow proposed extension, places general message generators and functions on switches, and provides failure recovery within 50ms in a scalable way.

The need for logging flow-related fault events in SDN networks is addressed by the application plane. Northbound APIs provide syslog handlers to allow developers processing fault messages sent by the OpenFlow controllers and develop applications that provide:

- Monitoring of topology-based faults through failed link and device status.
- List of affected hosts and the corresponding flow entries.
- Analysis of flow paths for root-cause process and determination of problems.

Concerning the monitoring of non-flow data, the SDN OpenFlow is not able to deal with equipment-centric data. For example, fault management or even performance management must be aware of the operational state of submodules contained in data plane devices (power supplies, resilient CPU unit, fans). Such data, related to the performance of hardware that doesn’t have an immediate impact on data plane performance, doesn’t belong either to the data plane or the control plane. Till today, there are not any studies that attempt to address this issue of monitoring and managing device-centric data. The latest OpenFlow switch specification [12] doesn’t deal with non-flow data. For the moment, only existing mechanisms used for monitoring device health-status such as SNMP can be used. SNMP has been successfully used in network monitoring and could fit to the SDN approach until a new OpenFlow accompanying protocol is implemented.

3.2 Configuration Management in SDN

The traditional network architecture is based on distributed and autonomous operation of network elements. Each network device contains an operating system and runs its own processes while implementing various network protocols. Processes related to network functions (e.g. routing) run within the device and an internal API is used to allow the operating system to program the forwarding hardware. The discovery or the programming of traditional network elements require the management of a distributed control plane (Figure 11a).   According to the SDN architecture (Figure 11b), a centralized Network Operating System is responsible for certain network needs and functions that support configuration management. The responsibilities of the Network Operating System have to do with [20]:

- Maintaining network topology
- Being aware of network state
- Handling changes in network topology and network state
- Transferring network changes to network applications and hardware

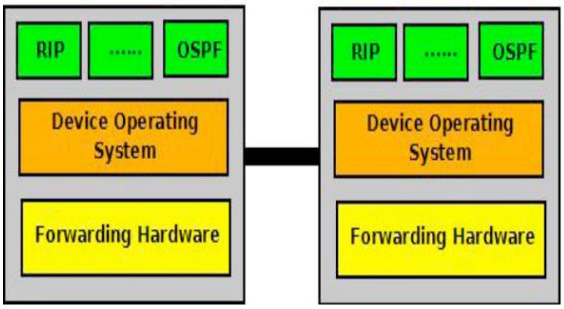


Figure 11a: Distributed Control Plane

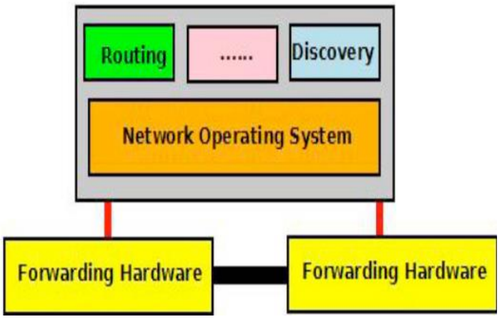


Figure 11b: Centralized Control Plane

The responsibilities mentioned above can be supported by:

- Techniques that have been used in section 3.1.1 to provide the control plane with information about link failures and link failure-affected hosts.
- Network applications which are software packages that implement particular functionalities of the network. They utilize the northbound API interface provided by the SDN controllers and they can be responsible for implementing network updates.

Considering the FCAPS model with the configuration management tasks that have been discussed in 2.1.3, it becomes clear that the centralized approach of the control plane with the Network Operating System plays a key role in configuration management. However, OpenFlow which is used as the southbound API is not able to perform certain important functions required by configuration management. For example, the assignment of the IP address to an OpenFlow switch cannot be done by OpenFlow. The need for various network management functionalities led to the proposal of a companion protocol to OpenFlow, the OpenFlow Configuration and Management Protocol which is also referred as OF-Config. The service used to communicate OF-Config messages to an OpenFlow Switch is called Configuration Point. The OpenFlow configuration port exchanges OF-Config messages with the operational context that includes an OpenFlow Switch (Figure 12) [21]. The OF-Config protocol provides various functionalities to configuration management [21]:

- IP address configuration of the controller
- IP address configuration of the OpenFlow switch
- Configuration of transport layer used in communication (TCP/TLS)
- Configuration of queues (rates) and ports
- Enable or disable ports
- Adjust speed on ports

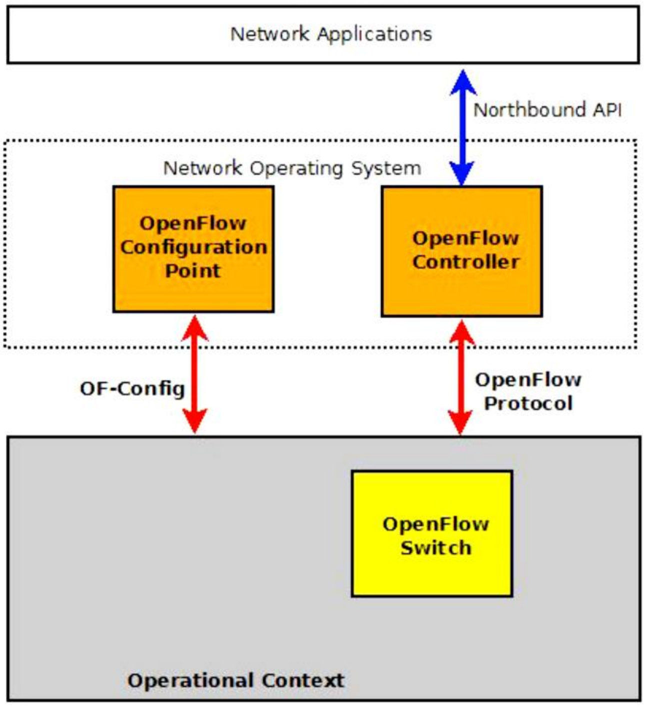


Figure 12: OpenFlow and OF-Config

The OF-Config requires the implementation of NETCONF (RFC4741), a protocol that implements remote procedure calls via an SSL channel. The NETCONF protocol used by OF-Config is extended with YANG models (RFC6020).The work of [20] has focused on network configuration by analyzing the ONF SDN model and its implementation on virtualized network environment of a network operator. This use case takes in to account the fact that various virtual network operators (VNOs) share the same physical infrastructure used in carrier-grade networks while the provided services can be from a point-to-point service up to a virtualized SDN environment (figure 13). The virtual connections are implemented by tunnels and can be monitored with methods already discussed in 3.1.1 (e.g. BFD). Each VNO controls the network offered to its customers via its own Network Operating System (NOS).

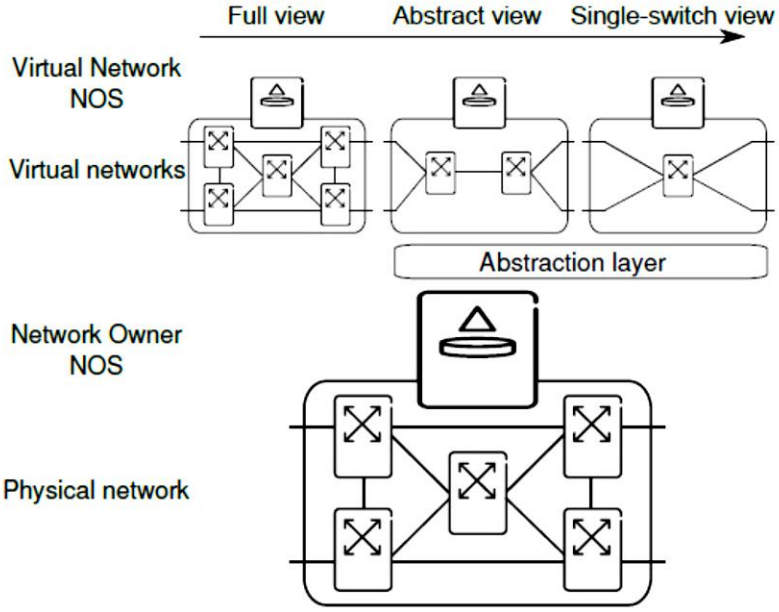


Figure 13: Carrier-grade SDN virtualization [20]

According to the aforementioned use case [20], the configuration management physical and virtual network infrastructure involves *device configuration*, *network bootstrapping, physical network configuration and virtual network setup*.

The *device configuration* procedure, for newly connected switches, can be initiated with OF-Config and DHCP (Dynamic Host Configuration Protocol) protocols. The basic connection identifiers required by the OpenFlow Configuration Point are the IP addresses of the switch and the IP address of the Configuration point. The IP assignment can be done automatically via DHCP. If any authentication certificates are required, certain identifiers can be provided as well. The credentials needed for a secure connection to the OpenFlow Controller and the OpenFlow Configuration Point can be preconfigured on a network device. The network's *bootstrapping* process can also be supported by the Authentication and Authorization (AA) service which enables an initial trust. The trust allows the OpenFlow switches to obtain certificates such as SSH keys, for securing the configuration channel. Once the encryption is ensured, the OpenFlow switch establishes an OF-Config connection with Configuration Point.

The *physical network configuration* initially involves the discovery of physical resources and capabilities. The OF-Configuration Point discovers physical resources (e.g. switch memory, CPU, ports) and capabilities (e.g. OpenFlow actions, supported tools) via the established OF-Config session. Prior to connecting a controller with a switch, the OF Configuration Point needs to represent as instance a logical switch with access to all of its physical ports. The logical switch is configured with minimum CPU and bandwidth allocation, and high traffic priority to ensure control under heavy traffic conditions. Then, the OpenFlow Configuration Point provides credentials and identifiers to a central controller, and finally establishes an OpenFlow session. Once the OpenFlow session is established, the central OpenFlow Controller which is connected to the logical switch uses discovery mechanisms such as LLDP in order to map the physical topology of the OF switches. The information obtained by discovery procedure is also sent to the central OF Configuration Point. Alternatively, OpenFlow Switches can contribute to the discovery of the topology by allowing switches to use discovery mechanisms (LLDP, Spanning Tree), switch-local topologies can be retrieved in a distributed manner. The OF Configuration Point then retrieves the distributed topology information in order to create the physical topology of the network.

The *setup of the virtual network* and specifically the virtual network topology is determined by the discovered topology, the discovered switch capabilities, and customer's requirements that affect network topology. Involving OpenFlow to configure flow tables, the central OpenFlow Configuration Point creates tunnels (virtual links) between the endpoints of the virtual network. Parts of physical resources and assigned capabilities, define logical switches operating within the virtual network.

While SDN has the ability to introduce programmatically new functionalities into networks, the configuration consistency depends on applications that enable controllers to implement the network. Since any software bugs increase the risk of triggering network outages or internet outages in carrier-grade networks, the centralized programming approach should minimize the likelihood of any bugs. OpenFlow applications must consider the fact that the system is asynchronous, with simultaneous events at multiple switches and delays that may affect control channel communication. An example can be seen in figure 14 where a packet is not reaching its destination due to delayed installation of flow entry in the second switch.

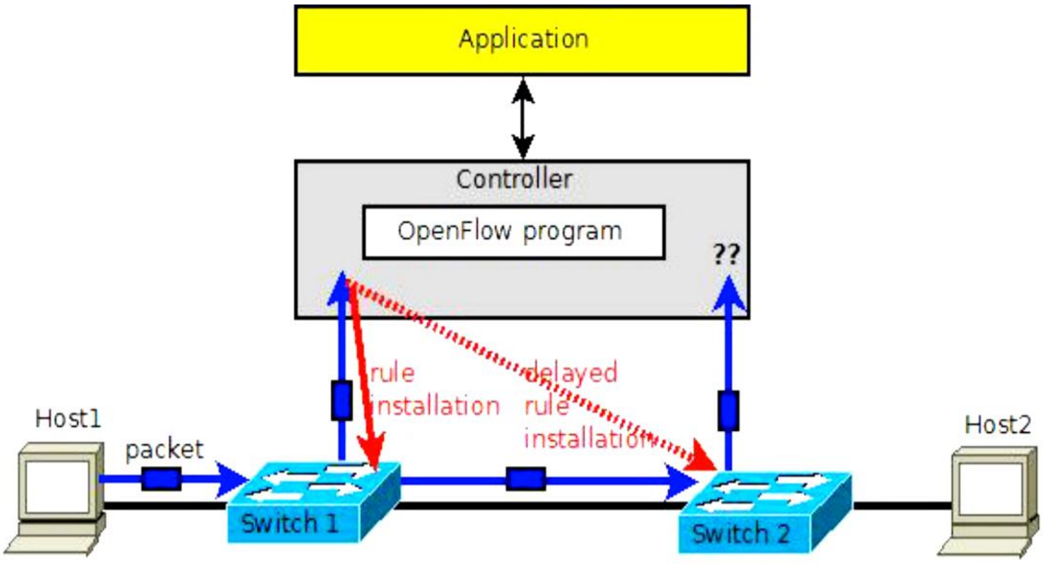


Figure 14: Delay in installation of flow entry

According to the work of [22] the wide system state, the variety of data packets and the many event driven actions contribute to scalability issue in OpenFlow networks. A proposed tool (NICE) can effectively *automate the testing of OpenFlow applications* written for NOX controllers by using symbolic execution with event handlers [22].

Instead of using low level interfaces to implement network changes, the study of [23] presents a different approach of higher level abstract operation that allows SDN programmers to apply the network configuration in one step. The study considers two abstractions:

- *Per-packet consistency:* This is the main abstraction assuming that each packet being forwarded within the network is treated according to one consistent network configuration. The packet may be processed according to the old configuration or the new but never as a mixture of the two.
- *Per-flow consistency:* This is a generalization of the per-packet consistency that assume packets of the same flow are forwarded according to the rules of the same configuration

The study of [23] led to the built of a system (Kinetic) that is implemented to the top of NOX controllers. A two-phase mechanism applies configuration updates using versioning in order to isolate the traffic and the old configuration from the new configuration. The proposed technique uses configuration version stamping in incoming packets by storing configuration version in VLAN tags while every configuration is modified so that only packets with specific version are processed. New configurations are enabled by installing rules at network's perimeter only after intermediary switches have been populated with newer version configurations. Provided libraries are mitigating this heavyweight programming approach.

3.3 Accounting Management in SDN

Accounting management for SDN networks can be related to the questions of charging customers in carrier-grade networks or just tracking the usage of network resources in corporate networks. Some examples of accounting management are:

- Once a customer exceeds a predefined bandwidth threshold of no-charge, the Network Operator charges its customers according to the consumed bandwidth.
- A network administrator maintains an archive with user associated bandwidth consumption.

Thus, SDN network should be able to identify users and determine the resources consumed by each user. This information should be sent via the NBI to the application layer for further processing. Generated reports of throughput or transactions logs can then be used by billing department or network administrators.

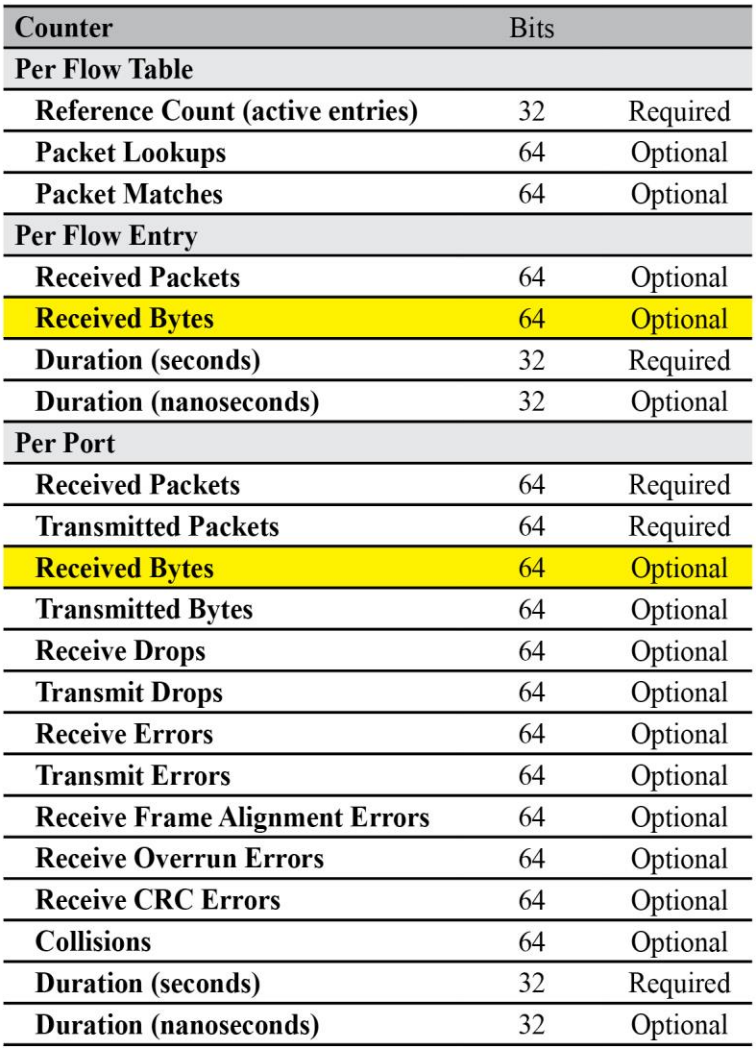
According to the OpenFlow switch specification 1.5.0, the flow table entries can match packets according to ingress port, source IP, egress port, destination IP etc. [12] [Appendix – Table A1]. The "match field" is the component of the flow entry used for distinguishing traffic according to the source IP which can later be associated with particular user.

Counters such as Bytes Received that could be used for accounting management purposes, are marked as optional even in latest OpenFlow Switch Specification (Table 2).

OpenFlow enabled switch vendors are not required to support the particular counters that would allow the gathering of accounting statistics.

Flow entry statistics can be requested by the OpenFlow controller taking advantage of the “Counters” component (Table 1). Counters being maintained separately for each flow are considered as hardware counters and have limited ranges. In order to resolve the limited-range issue, OpenFlow-compatible counters (software defined counters) can be maintained by polling the hardware counters. Depending on accounting needs, a variety of counters can be used to log user statistics such as Received Bytes, Transmitted Bytes, Duration (Seconds) etc.

Table 2
A partial list of OpenFlow Switch counters [12].



Counter	Bits	
Per Flow Table		
Reference Count (active entries)	32	Required
Packet Lookups	64	Optional
Packet Matches	64	Optional
Per Flow Entry		
Received Packets	64	Optional
Received Bytes	64	Optional
Duration (seconds)	32	Required
Duration (nanoseconds)	32	Optional
Per Port		
Received Packets	64	Required
Transmitted Packets	64	Required
Received Bytes	64	Optional
Transmitted Bytes	64	Optional
Receive Drops	64	Optional
Transmit Drops	64	Optional
Receive Errors	64	Optional
Transmit Errors	64	Optional
Receive Frame Alignment Errors	64	Optional
Receive Overrun Errors	64	Optional
Receive CRC Errors	64	Optional
Collisions	64	Optional
Duration (seconds)	32	Required
Duration (nanoseconds)	32	Optional

As already seen, OpenFlow controllers can be aware of network statistics supported by the data plane equipment. Accounting application should then be able to correlate statistics (generated by specific source IP addresses or ingress switch ports) with network users. The management of network users except for security management, is also closely related to accounting management as it allows the accounting applications to share a user database, aiming to assign network usage statistics to particular users. Therefore, users of networks resources must be recognized and associated with particular data flows in SDN networks. Since it is impossible in most cases to map ingress switch ports to particular users, user authentication could be used prior to correlating users' id, ip addresses and flow entries.

Accounting and billing for usage can also be supported by sFlow-RT engines [41], [42] that uses OpenFlow and sFlow protocols to associate real time statistics with users.

The work of [24] shows how Single Sign-On authentication technique can be implemented in OpenFlow SDN networks. The rationale behind this proposal is the creation of flow entries in OpenFlow switches only after a user is authenticated. A simplified OpenFlow SDN network implementing user authentication can be seen in Figure 15.

Assuming that a host (Host 1) has just been attached to the OpenFlow switch obtaining an IP address by DHCP server, then it will attempt to establish communication with a web server (WebServer 1). The Sign-on operation can be seen in the following steps:

- Host 1 attempts to send an HTTP request to Webserver 1.
- The OpenFlow Switch checks its flow table for a matching flow entry with source Host's 1 IP. As such entry doesn't exist, the packet is send to the controller for further processing.
- A packet control module installed in OpenFlow controller, checks permission status for source IP. Since permission hasn't been granted yet, the destination of HTTP packet request is modified in order to allow redirection of the packet to WebServer Auth.
- Through the Webserver Auth page, the user of client 1 is forced to authenticate.
- Upon successful authentication, the database is updated with a new entry containing: user ID, permission status, client IP and client MAC address. A new flow entry is also installed in the OpenFlow switch allowing Host's IP to reach WebServer 1.

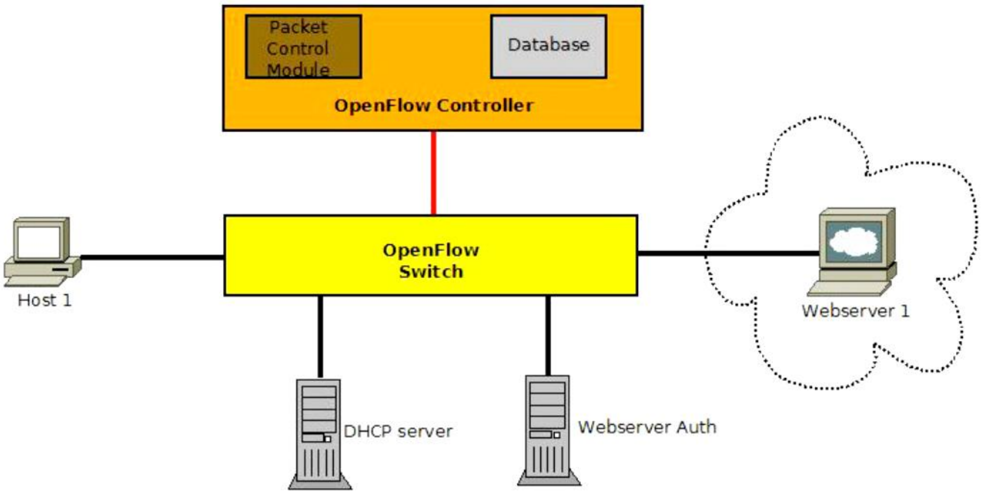


Figure 15: Host authentication in OpenFlow SDN network

3.4 Performance Management in SDN

Performance management can be based either on passive or active measurement methods [25]. The passive methods measure certain characteristics of existing data traffic while active methods inject additional network traffic to be used exclusively for measuring purposes. Monitoring network traffic and collecting network statistics can be supported by both methods. However, each method has its pros and cons. The passive method has the advantage of not generating additional overhead in network but they relies on pre-installed network monitor points. The passive measurement method cannot be implemented in all networks and would increase deployment cost. The active measurement method increases network overhead but it can support performance management by measurements without increasing infrastructure cost. For example, ICMP packets can be used for round-trip delay measurements. The problem with the active method is the fact that network monitor traffic may affect the performance of the “measured” network. Measurements can be performed on both network and application OSI layers depending on the requirements of performance management.

As already seen, through OpenFlow based data plane management, the SDN controller is able to request counter statistics for particular flows and also retransmit captured packets. According to OpenFlow switch specification, the switch sends an unmatched packet (PacketIn message) to the controller which installs a new path (FlowMod message). The controller orders the switch to send (PacketOut message) the previously unmatched packet via the newly installed flow. The OpenFlow message notifications (PacketIn, FlowRemoved) allow the controller to be aware of existing active flows. The FlowRemoved message informs the controller of the duration and byte values of the removed flow. The work of [25] uses the above information along with periodical request of statistics (StatsRequest messages) to acquire information from flows (figures 16a, 16b) and also monitor the end-to-end delay by injecting packets into the network.

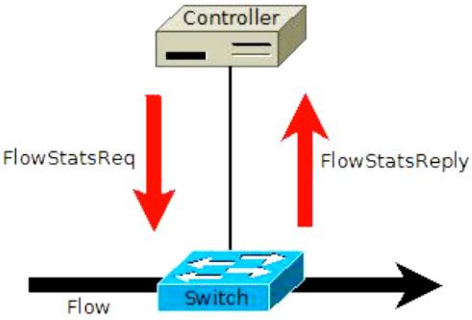


Figure 16a: When a flow exists, the controller retrieves flow statistics

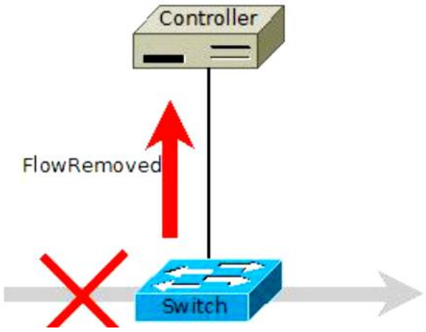


Figure 16b: When a flow ends, a FlowRemoved message notifies the controller

A proposed monitoring solution named OpenNetMon [25], an add-on for the OpenFlow POX controller, has shown that throughput, packet loss and round-trip delay statistics can be monitored in SDN networks by taking advantage of the OpenFlow features. OpenNetMon uses the method discussed in the previous paragraph in order to retrieve flow statistics. Regularly, the controller queries the switches and retrieves bytes sent and flow duration. Then, the OpenNetMon module calculates the throughput for queried flows. To improve efficiency and reduce complexity the polling involves only the last switch in each network path and the polling intervals adapt to flow’s behavior.

An important requirement for performance management, is the ability to maintain link utilization at high levels to reduce costs without degrading the performance of network services. Thus, throughput statistics should be accurate enough to show the fluctuating traffic needs of applications (e.g. high-definition stream) that may vary from 1Mbps up to 9Mbps. Measurements that take into account increased intervals for retrieving statistics, are much more prone to lose throughput traffic-peaks. OpenNetMon attempts to address this issue by adapting the sampling rate to traffic's profile. The sampling rate increases when new flows are defined or flow statistics vary, and backs-off when there is no fluctuation in measured bandwidth. As can be seen in figure 17, the proposed method takes into account the most, but not all, the transmitted traffic. TcpStat shows that transmitted traffic is not always captured by OpenNetMon. The authors of [25] claim that a synchronization issue between the two setups causes a bandwidth measurement inaccuracy.

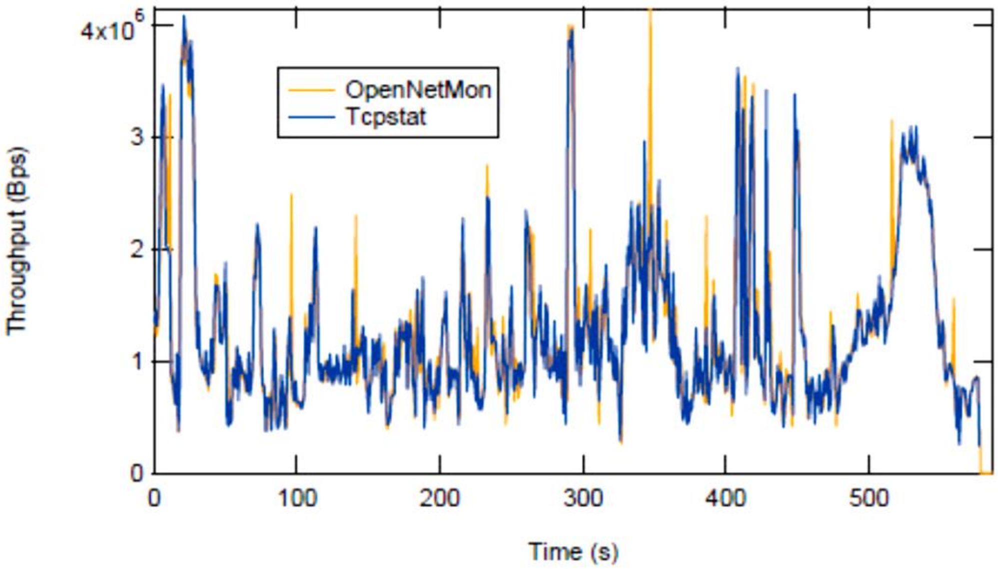


Figure 17: Comparison of OpenNetMon measurement and transmitted throughput [25]

Typically, packet loss in an OpenFlow switch can be estimated by polling port statistics while considering that the relation between throughput rate and packet loss is linear. This approach produces false estimations in case the traffic is affected by QoS configuration. An alternative estimation technique is the retrieval of flow statistics from the ingress and egress switches of each path. The abstraction of the first measurement (ingress) with the second one (egress), allows the estimation of packet loss.

Path delay can be calculated by using OpenFlow features. Although OpenFlow switches are not able to use tagged packets with timestamps to measure and compare inter-arrival times, OpenNetMon takes advantage of OpenFlow's capability to send packets into the network. In monitored paths, a packet is sent from the controller to the first switch of the path. The path delay can be expressed as the difference between the initial and the end switch's time-stamps while the subtraction of estimated delay between the controller and the first switch gives an accurate estimation of actual path delay (figure 18).

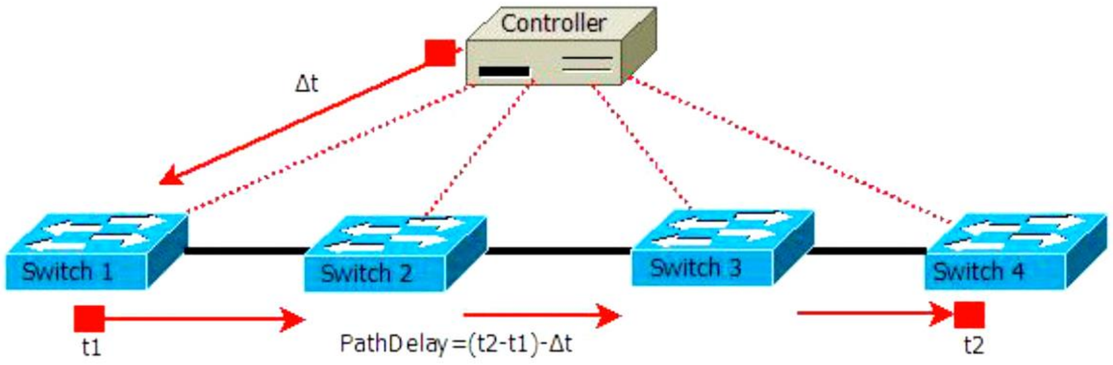


Figure 18: Path delay measurement

The delay between the controller and the delay (Δt) can be estimated by measuring the round-trip time (RTT) required for a packet sent by the controller to reach the first switch and then arrive back to the controller (Δt=RTT/2). Experiments have shown that software schedules running in the switches could affect delay measurements [25]. This issue can be addressed by using exclusively a separate VLAN for the transportation of probe packets, and also by adjusting the rate of injected packets according the size of the measured flow.

The work of [27] have also shown how OpenFlow’s per-flow statistics can be used in monitoring network traffic. The proposed technique (OpenTM) keeps track of flow statistics and provides a Traffic Matrix (TM). The controller application builds a TM by querying OpenFlow switches on fixed intervals and storing the results. The comparison of several polling algorithms shows that the most accurate statistics can be gathered by polling switches located in the end of path.

Beyond the measurement techniques being based on OpenFlow features, alternative solutions might also be implemented is SDN networks. OpenSketch [26] is a whole new monitoring architecture for SDN networks. This OpenSketch proposal addresses the measurement issue by introducing a new measurement API in order to support efficiency and a wide variety of traffic measurement tasks. The OpenSketch architecture is based on the separation of the measurement data plane from control plane and its implementation requires the replacement of existing network nodes. Another measurement technique that relies on both OpenFlow and sFlow protocols, provides the capability for real-time measurements for SDN networks being based on sFlow-RT (proprietary) measurement engine [41] [42]. SFlow-RT and OpenFlow based controllers, can contribute to SDN scalability and also to the development of performance aware SDN applications such as load balancing or queue priority marking, by providing real-time notifications and detection of flows [41] (Figure 19).

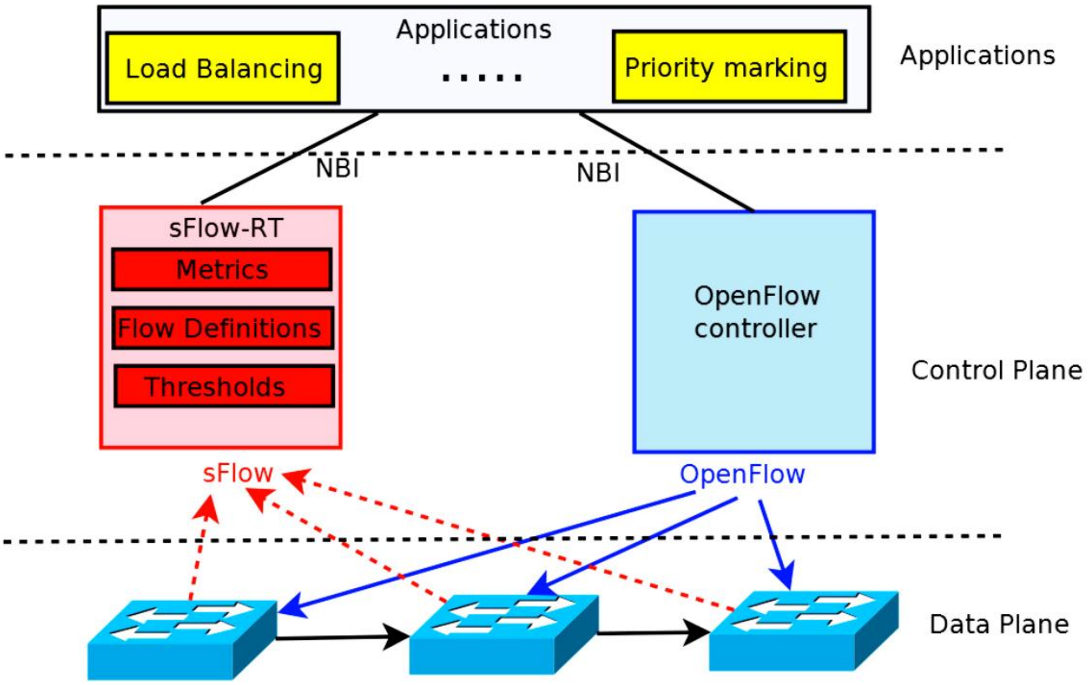


Figure 19: Using sFlow for measuring in SDN networks

The performance of an SDN network should be maintained within acceptable or predefined levels of operation not only by monitoring performance indexes but also by proceeding timely in proactive changes in network's configuration. For example, a proactive or a reactive load balancing configuration in network topology can improve performance indexes such as bandwidth utilization, packet loss and delay.

A typical load balancing implementation in a SDN network has been discussed in the work of [27] (figure 20). Web clients attempt to connect to a web site via internet. The web site is served from two virtual or physical server with different IPs. Whenever a client attempts to connect to the web site, the client sends a DNS request and receives the IP of one server (e.g. server 1). The OpenFlow switch receives from the client a packet with the destination IP. A software module is executed in the controller and adds flow-rules that forward the packet to server1 or modify the IP and the MAC address of the packet and forwards it to server2. The decision of which server will be selected by the controller depends on the load balancing policy.

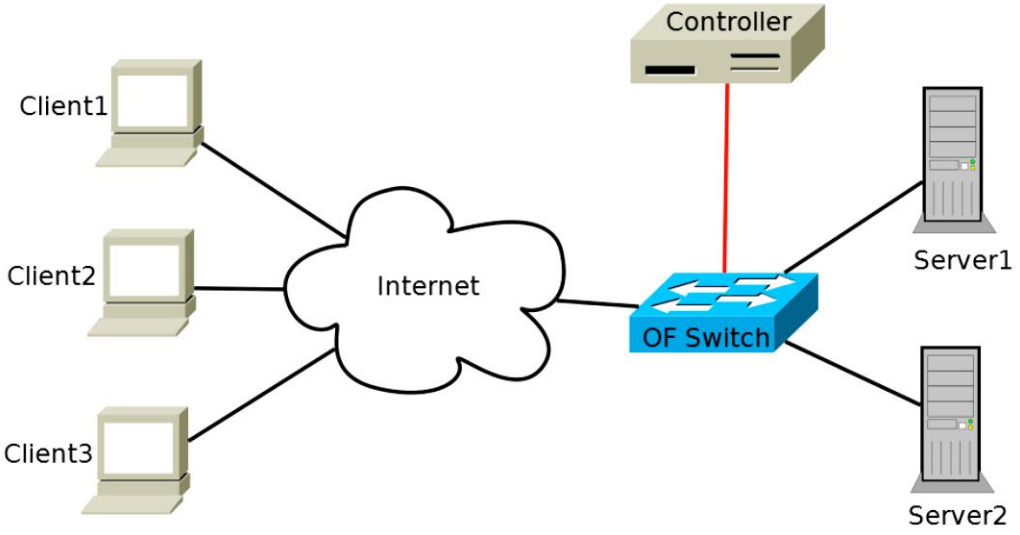


Figure 20. Load Balancing in SDN

Different balancing policies such as Random, Round Robin or Load Based can be applied [27]. If the server2 has been selected, the switch is instructed by the controller to modify the packets being send back to the client; Source IP and MAC address are replaced by the IP and the MAC address of server1.

High quality of experience in network applications requires proper QoS (Quality of Service) configuration on the underlying SDN network. The Unified Communications Operability Forum (UCIF) has published a use case that deals with the dynamic marking of video and voice traffic with QoS markings. In this use case, API events are communicated between a UC&C infrastructure and a “middleware” automated QoS application [45]. The middleware communicates with the SDN controller, as shown in figure 21, and allows the UC&C infrastructure to express QoS requirements using DiffServ (Differentiated Services) or IntServ (Integrated Services). It also translates the required class of service to particular DSCP (DiffServ Control Point) values and allocates queuing resources. According to the authors of [45], the automated QoS solution allows the implementation of automated QoS policies, makes simpler the deployment of QoS within SDN network and lowers deployment cost. Another important value delivered is the mitigation of QoS configuration errors due to misconfiguration.

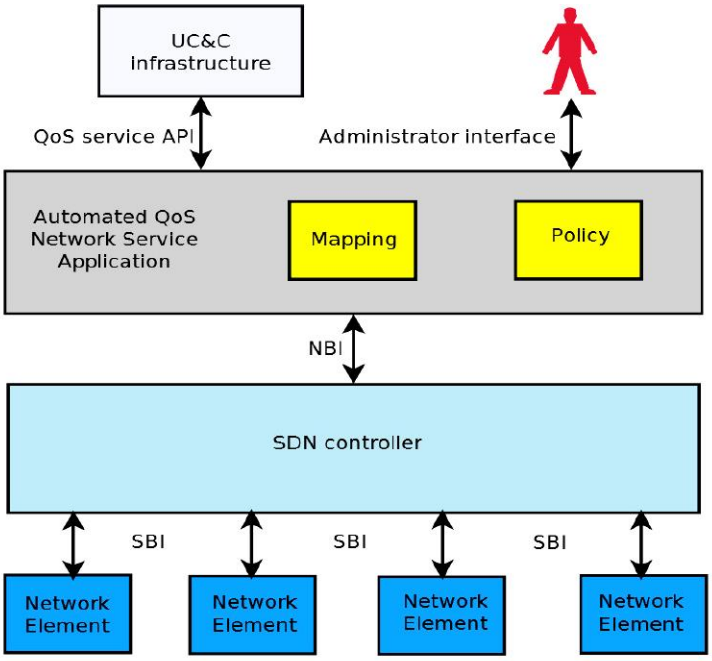


Figure 21: Automated QoS Network Service App

Latest OpenFlow switch specifications (1.3 and newer) support per-flow meters that reside in meter tables. Per-flow meters allow OpenFlow to implement basic QoS functions such as rate-limiting and can be used with per-port queues to support DiffServ

[12]. QoS schemes for queues, schedule schemes and queue management can be configured in OpenFlow switches through OF-config protocol by using the OF-config point [44].

Although network statistics along with load balancing mechanisms can help in predicting congestion, in many cases congestion may be unavoidable. In Software Defined Networks, the centralized control may allow the network itself to tune existing congestion control mechanisms of TCP [RFC5681]. The study of [46] shows that an SDN - TCP adaptation framework (OpenTCP) allows network operators to tune TCP according to traffic and network conditions. According to OpenTCP approach (figure 22), the SDN controller (Oracle) is the core of OpenTCP and it collects statistics about traffic and network. When congestion occurs, the SDN controller application calculates changes in TCP parameters needed in order to improve network performance. Updated messages (CUEs) including TCP adjustment parameters are send to terminal hosts that run a Congestion Control Agent (CCA). The CCA modifies the TCP protocol parameters on each terminal host, resulting in an end to end congestion control.

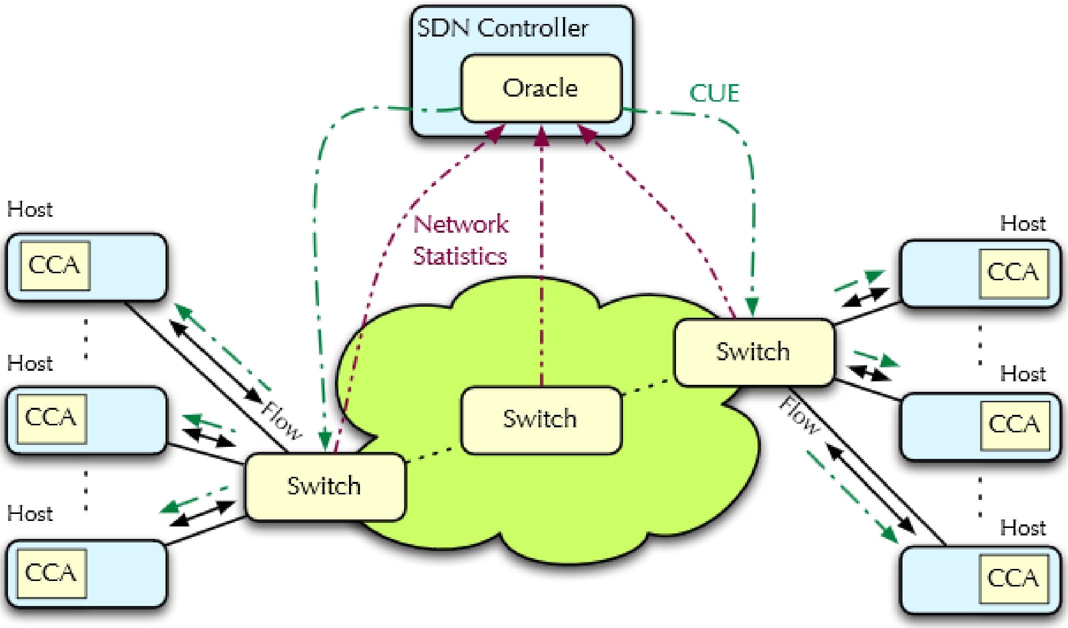


Figure 22: TCP congestion control in Software Defined Networks [46]

3.5 Security Management in SDN

The security management in SDN must be concerned with securing network resources and services form threats. OpenFlow offers to network management the ability to control data flows with specific characteristic. Some flows can then be regarded as hostile if they meet certain criteria defined by policies. OpenFlow security applications can halt a hostile flow or forwarded it to specific destination, or implement complex procedures [28]. For example, the hostile-flow producer may be isolated or a suspected flow may be further analyzed by a counter-acting intelligent application without affecting it.

As seen in 2.3.3, the OpenFlow switch can match the incoming packets against various fields such as source IP, destination IP, IP protocol type and TCP/UDP port. The switch can be instructed by the controller to drop traffic according to OpenFlow's matching headers [Appendix – Table A1]. Therefore, simple firewalling policies based on access control lists (ACL) can be implemented within SDN networks by taking advantage of OpenFlow features. More advanced firewalling techniques require the examination and action on fields that cannot be matched by OpenFlow (Figure 23). For example, URLs, viruses and hostnames included in packet payload (OSI layers 5-7) cannot be examined. OpenFlow must then forward the suspicious packets to a device or module that will implement a deep packet inspection [11]. Another limitation of SDN security-related applications that rely on OpenFlow is the lack of stateful awareness. The latest OpenFlow switch specification does not specify any stateful behavior for flows. Thus, specialized hardware or software could be used to address the lack of stateful awareness [11], [43].

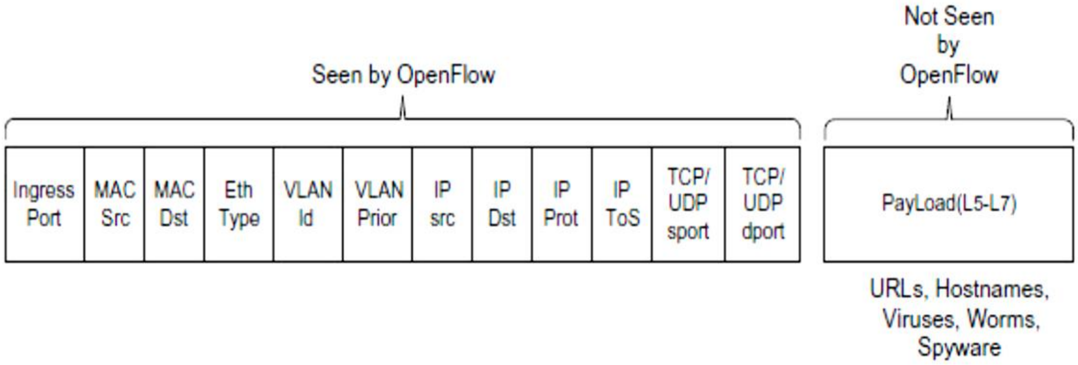


Figure 23: Packet inspection by OpenFlow [11]

The detection of traffic anomaly in SDN networks can be based in algorithms implemented in the control plane [29]. The study of [29] deals with the implementation of four existing threat detection algorithms in OpenFlow SDN networks with NOX controller. This study involves the following algorithms:

- *TRW-CB algorithm* [30]. It is based on the fact that a benign host establishes more successful TCP connections than a malicious one.
- *Rate Limiting algorithm* [31]. It considers the virus propagation behavior. When a virus propagates, the attacker attempts to establish connections to many different clients in a small time span.
- *Maximum Entropy Detector algorithm* [32]. It provides the normal traffic distribution by using a maximum entropy estimation. Although this algorithm requires the examination of every packet to build distributions, an indirect approach was used to avoid forwarding excessive amount of traffic to the controller: distributions were based only on TCP SYN/RST packets.
- *NETAD algorithm* [33]. It uses the assumption that traffic anomalies can be detected by examining the first packet of a connection. Since anomalies are detected, the rest of the traffic is rejected as “uninteresting”. For example, the NETAD implementation can result in the rejecting all incoming traffic to network where incoming traffic is not expected.

The work of [29] have shown that OpenFlow SDN networks allow the flexible and accurate detection of traffic anomalies within SOHO (Small Office – Home Office) networks. The programmability of SDN allows the development of applications that detect attempts for network exploitation and also mitigate the risk by immediate reaction to network threats.

A security challenge for networks is Distributed Denial of Service (DDoS) which consists of the most common attacks in Internet. DDoS is faced by the work of [34] within the context of OpenFlow SDN. The proposed method for detecting DDoS attacks assumes that a NOX SDN controller monitors OpenFlow switches during predefined intervals. During monitoring intervals, interesting features from flow entries are extracted and then passed to the classifier module. The classifier module uses the SOM algorithm [35] to evaluate gathered information and indicates whether the information corresponds to an attack or normal traffic. The proposed method (Figure 24) uses a detection loop consisted of three modules:

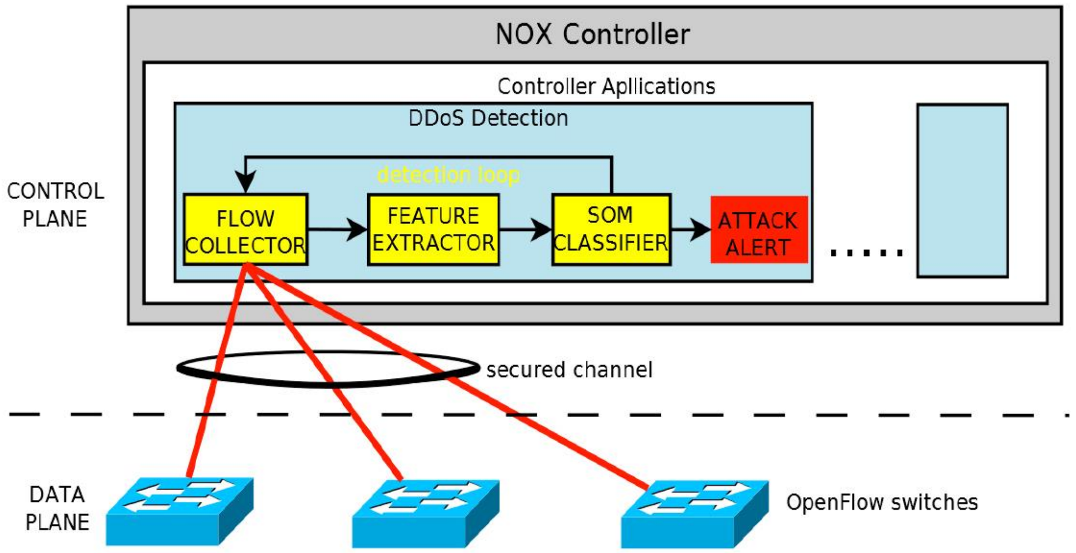


Figure 24: Detection of DDoS attack

1. *Flow collector.* This module periodically requests flow entries from all OpenFlow switches. The communication between the flow collector module and the switches remains isolated from any hosts attached to the switches as it is carried through a secure channel.
2. *Feature Extractor.* This module receives flows collected by Flow Collector, extracts important for DDoS detection features and forwards them to the Classifier module.
3. *Classifier.* This module analyzes the data provided by Feature Extractor and uses the SOM classification method to detect DDoS attacks.

The proposed solution for detecting DDoS attacks shows the flexibility of OpenFlow networks in implementing detector applications by obtaining flow-information from the data plane. The method is based on SDN’s centralized approach and OpenFlow’s efficiency in providing information required. According to the results of the work, detection rate and false alarm rate are similar with the results of other approaches. Due to the fact that the controller does not collect all packets sent to the attacked host but classifies patterns of flow-based information being obtained every 3 seconds, the overhead is remarkably lower compared to other approaches. As already seen in performance management section, flow statistics that support applications such as DDoS protection can be also be provided by sFlow implementation in OpenFlow SDN networks [41].

Security management can also be supported by automated malware quarantine (AMQ) [30] which takes advantage of OpenFlow features and existing security systems to reactively isolate infected hosts. An OpenFlow switch forwards traffic to a threat monitor system (e.g. an IDS) which monitors flows and detects infected hosts in real time (figure 25).

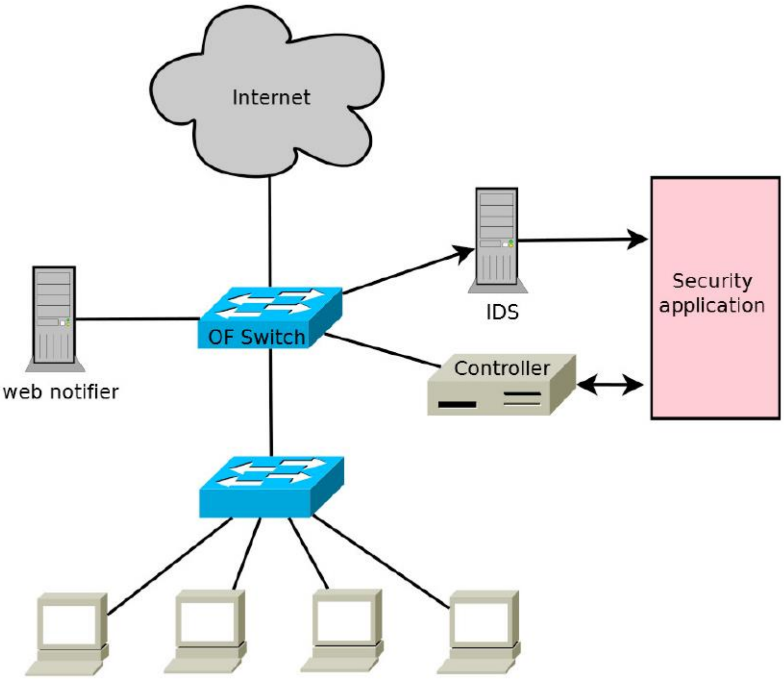


Figure 25: Automatic malware quarantine in SDN

In an AMQ case, a host is compromised by a rootkit which attempts to infect other hosts in the local network or to call home via internet. The threat monitoring system monitors network traffic and malicious patterns from the rootkit in order to create the infection profile. The source IP of the rootkit indicates the infected host enabling the security application to instruct the controller to isolate the infected host. The isolated host can then be connected to another VLAN where a web server is enabled to notify the user about the quarantine status and further actions that might be needed. Assuming that the threat has been fully analyzed and the necessary software patch is available, the host can be patched to remove the rootkit and then be connected back to the default VLAN.

Compared to traditional networks, the centralization of the control plane in SDN introduces a central control-point that needs protection against attacks [11]. Considering the case of a traditional campus network, there may be hundreds or thousands of control nodes that can be targeted by attackers. In an SDN campus network with similar size, the control nodes (SDN controllers) might be less than 30. In the first case, a compromised node most likely will bring down a part of the network or a subnet. In the case of SDN, the compromising of a controller could bring down the campus network. So, SDN's main benefits which are control centralization and programmability, introduce new attack and fault planes. The work of [36] identifies seven important threat vectors that could lead to exploit of OpenFlow SDN vulnerabilities and proposes a reliable and secure control platform for SDN. The seven potential security threats along with possible solutions are:

1. *Faked or forged data flows*. Such flows can be triggered by an attacker who uses network elements aiming to initiate a DoS attack against controller's or switch's resources. Authentication mechanisms can minimize the risk only if authentication service hasn't been compromised. Intrusion detection systems (IDS), coupled with switch behavior analysis and control, can help in identifying abnormalities within flows.
2. *Vulnerabilities in OpenFlow switches*. An affected switch might be used for slowing down packets, dropping packets, deviating traffic or injecting traffic for further exploits. The proposed mechanisms for counteracting this threat are: autonomic trust management [37], monitoring and detection of abnormal traffic.
3. *Compromising control plane communications*. Typically, the OpenFlow messages exchanged between the controller and the switches is implemented on top of TLS or SSL. Several studies and reports have shown that TLS/SSL cannot ensure the secure exchange of OpenFlow messages [38], [39]. Potential weakness can be any untrusted Certificate Authorities, self-signed certificates and vulnerabilities in applications. Moreover, trust between the controllers cannot be guaranteed with TLS/SSL model. Certification and trust issues between data plane and control plane devices can be addressed by using per controller or per subdomain certification authorities, improving encryption, and by using automated and dynamic device association.
4. *Vulnerabilities in controllers*. An attack that exploits a controller vulnerability could bring down the entire network. The detected malicious events may be a hard task as

such events must be identified and isolated. Several countermeasures can be implemented: replication, diversity and recovery,[36]. Security policies should also enforce the acceptable behavior of applications e.g., not allowing an application to use specific interfaces.

5. *Trust between management applications and controllers*. There are not any mechanisms to support the establishment of a trusted connection between a controller and the management application. Trust management mechanisms should be implemented to ensure the trust and eliminate this vulnerability.
6. *Vulnerabilities in administrative hosts.* Administrative hosts play a critical role in SDN network since they have the ability to program and change the behavior of the entire network. Obviously, an attacker who manages to exploit a vulnerability in an administrative host is able to reprogram the network. To minimize the risk of losing administrative control of the network, more strict policies can be applied to user-authentication protocols. For example, double authentication may be required to gain access to the management system. Furthermore, recovery mechanisms should be available in order to roll back the network to the last operational state in case an attacker has reprogrammed the network exploiting a vulnerability.
7. *Lack of forensics and remediation mechanisms.* As soon as a problem is detected, specific actions should be followed to ensure network recovery and ability to investigate the incident. Recovery requires historical snapshots of all network elements to guarantee a successful and fast remediation. The investigation of the incident requires information from all network elements. Logging must be enabled in all network devices from the data plane up to the management plane and logs should be kept in secure data storages.

The aforementioned seven threats of SDN and their association with of SDN’s planes can be seen in figure 26. These potential security threats can affect the entire network by exploiting vulnerabilities in data or control plane. Countermeasures based an existing techniques, or in OpenFlow features can be implemented to mitigate the risk.

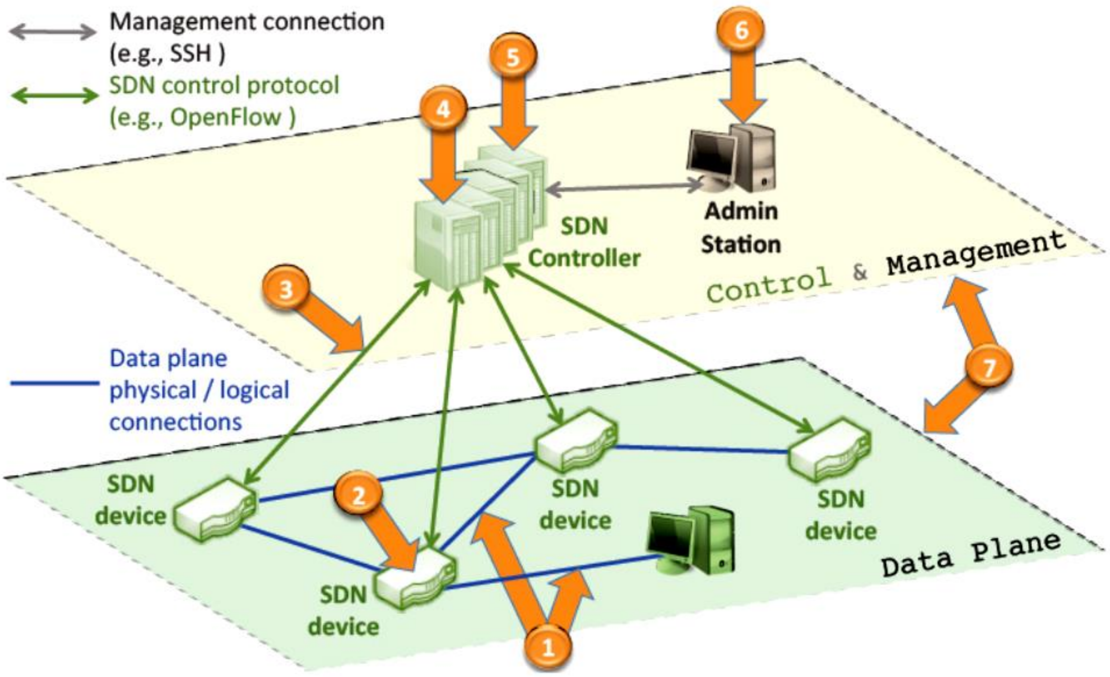

Management connection (e.g., SSH)
SDN control protocol (e.g., OpenFlow)
Data plane physical / logical connections
SDN Controller
Admin Station
Control & Management
SDN device
SDN device
SDN device
SDN device
Data Plane
1
2
3
4
5
6
7

Figure 26: SDN threats [36]

4 FCAPS evaluation in SDN

Several aspects of network management have been discussed and analyzed through studies, use cases and ONF specifications in chapter 3. All the material used has been categorized according to FCAPS management model relativity. Network management tasks can then be isolated and evaluated preserving the categorization according to the FCAPS model.

4.1 Methodology of evaluation

The methodology for evaluating network management in OpenFlow SDN networks is based on the following steps:

1. Identification and isolation of tasks implemented in network management.
2. Categorization of network management tasks according to the FCAPS model.
3. Network management task evaluation.
4. FCAPS category evaluation

The identification and isolation of functions is based on SDN studies, use cases and OpenFlow features that have been discussed in the previous chapter. For example, the case of detecting a link failure within an SDN network and the recovery techniques followed, allows the identification of network management tasks such as “Detection of link failure”, “Restoration of a failed link” and “Fast recovery (less than 50ms)”.

The coarse categorization used in chapter 3, helps in isolating and categorizing specific network management tasks according to the FCAPS model. However, particular functions may belong in more than one categories of the FCAPS model. For example, tasks related to user identification will belong to both accounting management and security management categories.

Since this dissertation aims to investigate whether ONF SDN can follow the FCAPS model, the evaluation of the functions will be based on questions that allow us to rate:

- ONF SDN's contribution in management functions
- need for additional protocols or techniques
- efficiency within the OpenFlow SDN context

The question that will be asked in order to rate management tasks are the following:

1. Is the function supported by OpenFlow, or an OpenFlow companion protocol such as OF-Config, or generally by the ONF SDN architecture?
2. Does the particular function require additional protocols or techniques to be implemented?
3. Can the implementation be considered as effective?

Since, in many cases the answers to the aforementioned questions cannot be clearly 'yes' or 'no', answers closer to 'yes' will be marked as 'High' and answers closer to 'no' will be marked as "Low".

In question 1, if there is unambiguous involvement and contribution of OpenFlow or OF-config protocols in the effective implementation of the management task, the answer will be 'High'. In all other cases, the answer will be 'Low'.

In question 2, if there are any additional protocols needed to effectively implement the management task, or programming effort is required in control/application planes, the answer will be 'High'. If there is no such need, the answer will be 'Low'

In question 3, if the implemented management task achieves its objective, the implementation will be considered as effective and the answer will be 'High'. The insufficiency to meet its goals will result in an answer marked as 'Low'. For example:

- *Fast Detection of path failure (Fault Management):* OpenFlow supports partially this operation (OpenFlow support is "Low"), there is a need for additional protocols such as BFD to be implemented (need for additional protocol is “High”) and the final implementation is acceptable (effectiveness is “High”)

- *Enabling or disabling switch ports (Configuration Management):* OF-Config is enough to perform such tasks (OpenFlow’s companion protocol support is “High”), there is no need to apply additional techniques or protocols (need is “Low”) and the implemented task is acceptable (effectiveness is “High”).

In each one of the five FCAPS categories, an overall assessment will be reported and discussed along with problems identified during the evaluation process.

4.2 Evaluation issues

Although the latest OpenFlow Switch specification (version 1.5) has been discussed in this dissertation, all studies and use cases that have been presented in chapter 3 are based on OpenFlow’s version 1.3 or earlier versions. Simulated networks and experiments in the studies [13], [15], [17], [19], [27] have been carried out using Mininet [47], an OpenFlow SDN network emulator that emulates SDN switches running versions up to OpenFlow 1.3. Although new features and advantages of latest OpenFlow version can obviously improve several aspects of network management, they have not been studied yet in SDN network simulations.

Different network implementations have different requirements from network management systems. Although the FCAPS model may be applied from small enterprise networks up to campus or carrier-grade networks, network functions required for fulfilling specific management needs may vary on importance. Service level agreements (SLAs) require excessive effort in performance management implementation in order to allow operators to meet demanding customer expectations while there no such need in SOHO networks. In this dissertation, the evaluation includes management functions from different network implementation scenarios. This evaluation presents a generalized view of FCAPS implementation in OpenFlow SDN networks. As this evaluation was mainly based on existing studies, the management functions required in particular network implementation may not have been reported.

It should also be mentioned that several management functions may fall not only in one category of FCAPS. For example, although "user identification" can be classified as a function that typically belongs in security management, it also plays an important role in accounting management as it might be associated with billing or per-user network utilization. Thereafter, some functions that have been identified in a section that deals with a particular FCAPS category, may be reported again in different categories.

4.3 Evaluation of Fault Management in SDN

The OpenFlow SDN fault management scenarios that have been discussed in chapter 3, include several fault management functions that can be seen in Table 3. The ONF SDN's contribution, the need for additional protocols or techniques and the efficiency within a general OpenFlow SDN context have been rated according to methodology already mentioned.

Some particular functions such as those related to recovery time can also be associated with network's performance and they will be mentioned again in performance management evaluation section. Also, the transfer of network changes to application layer and hardware, as part of recovery techniques, is considered as a function of configuration management category and it will be reported in the configuration management evaluation section.

As can be seen in Table 3, most of the examined functions are supported by ONF SDN architecture. It is clear that OpenFlow's features play key roles in detection and recovery mechanisms that follow network link failures. In many cases, the fault management functions require the contribution of traditional network management protocols in order to be implemented effectively.

Considering that:

1. The fast recovery speed required in carrier-grade networks (which is also related with network's performance) cannot be supported sufficiently nor by ONF SDN architecture neither by additional protocols.
2. The monitor of device-centric data have not been taken into consideration in SDN's architecture.

3. Three out of eleven fault management function heavily depend on traditional management protocols.

it can be argued that OND SDN needs further support by future SDN companion protocols. In current SDN management, the data plane equipment must be able to support protocols such as SNMP, BFD and LLDP. There is obvious need for new OpenFlow companion protocols to be introduced or existing ones (OpenFlow, OF-Config) to be upgraded accordingly.

Concerning the involved management mechanisms and the effectiveness rating, it can be argued that: OpenFlow SDN, programming effort in both control and application planes and legacy management protocols can result in a quite effective fault management.

It is also observed that although certain SNMP features such as port statistics can be accessed with OpenFlow based techniques, certain SNMP functionalities still play an important role in fault management.

Table 3
Evaluation of Fault Management Functions.

Fault management function	Contribution		Effectiveness	Key techniques and protocols
	Non-SDN based techniques	SDN based techniques		
Detection of link failure	○	●	🟢	OpenFlow port status Echo request
Restoration of a failed link	○	●	🟢	Flow tables update Flooding control traffic OpenFlow port status
Link protection	○	●	🟢	Path reservation OpenFlow port status Fast failover Group Type in Group Table Controller resiliency
Fast recovery (less than 50ms) in restored links (carrier-grade networks)	○	○	🔴	Reaction depends on controllers and delays in notifications Fast failover Group Type in Group Table BFD
Recovery within 50ms in protected links	●	○	🟢	Complex resource allocation BFD
Fast detection of path failure	●	○	🟢	BFD LLDP
Fault logging	○	●	🟢	NBI / Syslog handlers Requires programming effort in application plane
Monitoring topology based faults	○	●	🟢	Use of failed links and device status NBI /Syslog handlers Requires programming effort in application plane
Listing affected hosts	○	●	🟢	OpenFlow port status Use of flow entries NBI /Syslog handlers Requires programming effort in application plane
Root-cause analysis	○	●	🟢	NBI /Syslog handlers Requires programming effort in application plane Flow path analysis
Monitoring non-flow data	●	○	🟢	SNMP Software in Application Plane

● High	🟢 High
○ Low	🔴 Low

4.4 Evaluation of Configuration Management in SDN

Studies related to configuration management in OpenFlow SDN networks have been discussed in chapter 3. The identified configuration management functions can be seen in Table 4. The ONF SDN's contribution, the need for additional protocols or techniques and the effectiveness within a general OpenFlow SDN context, have been rated according to methodology already mentioned.

The transfer of network changes to application layer and hardware that has been discussed in 3.1 as part of recovery techniques, is a sub-function of configuration management that can be addressed by programming effort and NBI involvement.

As can be seen in table 4, OpenFlow and especially its companion protocol, OF-Config, contribute in an effective implementation of all examined configuration management functions. Although OF-Config supports effectively most functions, there is still the need for using traditional protocols such as BFD and LLDP. OF-Config is focused in data-plane configuration and as can be seen in table 4, most functions related to data plane configuration (queues, rates, ports) can be effectively implemented by using OF-Config.

It can be generally argued, that in all examined configuration management functions the implementation is effective and also it is highly supported by OND SDN architecture and protocols. SNMP's network configuration management features have been successfully replaced by OF-Config. However, limited additional support by legacy protocols is required in certain functions as seen.

Table 4
Evaluation of Configuration Management Functions.

Configuration management function	Contribution		Effectiveness	Key techniques and protocols
	Non-SDN based techniques	SDN based techniques		
Maintaining network topology	●	●	🟢	Flow tables update
				OpenFlow port status
				BFD / LLDP
				NBI
				Software in application plane
Awareness of network's state	●	●	🟢	Flow tables update
				OpenFlow port status
				BFD / LLDP
				NBI
				Software in application plane
Handling changes in network topology and network state	●	●	🟢	Flow tables update
				OpenFlow port status
				BFD / LLDP
				NBI
				OF-Config (NETCONF)
				Software in app plane
Transferring network changes to network applications and hardware	○	●	🟢	Recovery techniques (Fault management)
				NBI
				Software in application plane
IP address configuration of switch / controller / configuration point	●	●	🟢	OF-Config (NETCONF)
				DHCP
Configuration of transport layer used in communication (TCP/TLS)	●	●	🟢	OF-Config (NETCONF)
				Authentication and Authorization
Configuration of queues (rates) and ports	○	●	🟢	OF-Config (NETCONF)
Enable or disable ports	○	●	🟢	OF-Config (NETCONF)
Adjust speed on ports	○	●	🟢	OF-Config (NETCONF)
Configuration versioning	○	●	🟢	Flow installation
				Control plane programming effort
				Provided libraries

● High
○ Low

🟢 High
🔴 Low

4.5 Evaluation of Accounting Management in SDN

The evaluation of tasks involved in Accounting Management can be seen in Table 5. Certain tasks such as user identification, authorized access to network resources and user statistics can also be associated with Security Management Category. Network usage statistics required in accounting management can also be considered as performance management functions.

As can be seen in table 5, for an effective accounting management, OpenFlow features, existing user authentication mechanisms and programming effort in application plane are required. Traditional authentication services and mechanisms used in accounting and security management offer the information required for the correlation of user's identity (user name, IP) with flows. Thus, it is possible in SDN to personalize the usage of network resources. The collected data that associate users with network resource usage or resource allocation, can be further processed by billing software in order to allow network providers to charge their customers accordingly.

It can be argued that Northbound interface, software application that interract with databases in application plane and OpenFlow features allow accounting management functions to be implemented effectively.

Table 5
Evaluation of Accounting Management Functions.

Accounting management function	Contribution		Effectiveness	Key techniques and protocols
	Non-SDN based techniques	SDN based techniques		
User identification and access control	🟤	🟤	🟢	Process or drop non-matching packets Redirection and force authentication Authentication service Granting access by new flow installation in switch
Network usage statistics (OpenFlow based)	○	🟤	🟢	Per flow entry counters Per port entry counters Software defined counters in control plane NBI Application plane software & database
Network usage statistics (sFlow-RT based)	🟤	🟤	🟢	sFlow-RT engine in SDN controller sFlow & OpenFlow protocols
Correlation of user and network statistics	○	🟤	🟢	Ethernet source address in matching field of OpenFlow Per flow/port counters NBI Application plane software & database
Archiving user associated bandwidth consumption	🟤	○	🟢	Application plane software & database
Charging	🟤	🟤	🟢	Application plane software & database Correlation of user and network statistics function

🟤 High 🟢 High
○ Low 🔴 Low

4.6 Evaluation of Performance Management in SDN

The evaluation of functions involved in Performance Management can be seen in Table 6. Although fast recovery after a link failure typically falls under fault management, a slow recovery (more than 20ms) degrades the performance of real time network applications such as VoIP or video conference. For this reason, fast recovery function is reported in both fault management and performance management evaluation sections.

In most cases, OpenFlow features along with software modules running in SDN controllers support effectively the implementation of the performance management functions. It can also be observed that traditional network management techniques, play either an important or a complimentary role along with SDN protocols in the effective implementation of the related functions. According to fault management evaluation, network performance cannot be managed effectively in case link restoration is involved. Another performance management issue can be seen in the function of accurate estimation of fluctuating traffic. There is lack of intrinsic OpenFlow features to support performance monitor of real time application traffic. However, this problem can be addressed by using a proprietary solution such as the sFlow-RT controller engine. Throughput and packet loss measurement functions in passive mode are also ineffectively implemented although they are typically supported by OpenFlow and SDN control plane features.

Centralization and programmability can also improve network performance by adapting TCP’s congestion control parameters according to network’s condition. However, adaptation speed depends on network statistics and therefore a proprietary controller module such as sFlow-RT is required again to implement TCP congestion control with co-existing real time applications’ traffic.

Table 6
Evaluation of Performance Management Functions.

Performance management function	Contribution		Effectiveness	Key techniques and protocols
	Non-SDN based techniques	SDN based techniques		
Throughput measurement (passive measurement)	○	○	🔴	The controller periodically requests statistics from path's last switch Calculation of bytes sent & flow duration Polling interval adapted to flow behavior OpenNetMon (POX add-on)
Packet loss measurement (passive measurement)	○	○	🔴	The controller periodically requests statistics from path's first and last switches Subtraction of in – out measurements OpenNetMon (POX add-on)
Roundtrip delay measurement (active measurement)	○	●	🟢	The controller injects probe packets Exclusive VLAN for transportation of probe packets
Accurate estimation of fluctuating traffic / real time needs (sFlow-RT based)	●	●	🟢	sFlow-RT engine in SDN controller sFlow & OpenFlow protocols Real-time notifications & detection of flows
Server load balancing	○	●	🟢	Software module in controller Flow-rules to forward traffic according to balancing policy NAT
QoS	○	●	🟢	OpenFlow per-flow meters OF-Config Automated Network Service Application (Mapping, Policy) communicates via NBI with controller QoS Service API
Recovery within 50ms in protected links	●	○	🟢	Complex resource allocation BFD
Fast recovery (less than 50ms) in restored links (carrier-grade networks)	○	○	🔴	Reaction depends on control-lers and delays in notifications Fast failover Group Type in Group Table BFD
Network usage statistics	○	●	🟢	Per flow /port counters Software defined counters in control plane NBI /Application plane software & database
End to end congestion control	○	●	🟢	Controller application Traffic statistics Adjustment of TCP parameters on hosts

● High 🟢 High
○ Low 🔴 Low

4.7 Evaluation of Security Management in SDN

The evaluation of functions involved in Security Management can be seen in Table 7. Functions related to user identification and user access has also been mentioned in accounting management. As already discussed in 2.1.3 and seen in figure 1, security functions has to underpin all other functional areas (Fault, Configuration, Accounting and Performance) to allow security to be effective. The interaction between security and other functions can be clearly seen in key protocols and techniques involved in table 6. For example, protecting configuration process, recovery (configuration roll back), traffic monitor and user authentication, play important role in the implementation of security functions.

OpenFlow's header matching-fields allow data-plane equipment such as SDN switches to embed firewalling capabilities that in traditional network management would depend on the use of firewall-dedicated hardware. Additional mechanisms used for authentication purposes along with SDN's access control capabilities improve several aspects of security management. Malicious flows and hosts can be early detected and isolated resulting in minimized security risks. Access control is based on OpenFlow's ability to match header information queried in different OSI layers. However, there is not yet any SDN capability for implementing stateful packet inspection. The lack of statefulness in SDN firewalling capabilities shows that although OpenFlow switches can perform partial firewall operations and enhance security, they cannot replace traditional firewalls yet.

It can also be seen in table 7 that trust between management application and SDN controllers cannot be supported yet neither by SDN architecture nor by additional mechanisms.

Given the evaluation of security management functions as seen in table 7, it can also be argued that, OpenFlow and SDN's architecture are not able to support effectively security management functions without using additional mechanisms. Security management heavily depends on techniques and practices used in existing security management systems. However, OpenFlow's and OF-Config's evolution, along with SDN's architecture, allow control plane and application plane software to address in the future the issues that exist in current security management implementations.

Table 7
Evaluation of Security Management Functions.

Security management function	Contribution		Effectiveness	Key techniques and protocols
	Non-SDN based techniques	SDN based techniques		
ACL firewalling	○	●	🟢	OpenFlow's header matching fields
Stateful firewalling	○	○	🔴	Under investigation
Protection from DDoS attack	○	●	🟢	A module within the controller periodically requests flow entries from OpenFlow switches DDoS features are extracted from flows Classification of collected features and detection of DDoS attacks
Malware quarantine and infected host isolation / Payload examination	●	●	🟢	The OpenFlow switch forwards traffic to a threat moni-toring system By searching for malicious patterns within flows, infected hosts are detected Source IP of infected host is reported to application The controller instructs the switch to disconnect host from default VLAN and connect it another one
Protection from fake or forged data flows	●	●	🟢	As previous case
Addressing the risk of vulnerabilities in OpenFlow switches	●	●	🟢	Autonomic trust management [37] Monitoring traffic Detection of abnormal traffic
Addressing the risk of compromised control plane communications	●	○	🟢	Use certification authorities per subdomain or controller Improved encryption Automated and dynamic device association
Addressing controller's vulnerabilities	●	○	🟢	Replication Diversity Recovery Security policies (to enforce acceptable behavior of applications)
Building trust between management applications and controllers	○	○	🔴	(No mechanisms available to ensure trust between apps and controllers)
Mitigating effects of vulnerabilities in network's administrative hosts	●	○	🟢	Strict / double user authentication Recovery (roll back to last operational state)
Forensics and remediation	●	○	🟢	Logging in all network devices
User identification and access	●	○	🟢	Process or drop non-matching packets Redirection and force authentication Authentication service Granting access by new flow installation in switch

● High
○ Low

🟢 High
🔴 Low

5 Conclusions and future work

5.1 Summary of main points in SDN management

Through the perspective of the FCAPS ISO model, OpenFlow-based Software Defined Networks management can be effectively implemented. However, SDN Management needs to be supported by traditional management mechanisms and OpenFlow-companion third party solutions. The openness of ONF SDN architecture enables through programmability the effective implementation of most management functions although some features that could facilitate management functions haven't yet been adopted in Open-Flow and OF-Config protocols. The issues that still exist in a limited number of management functions mainly concern carrier-grade networks. For their solution, further investigation, programmability effort and continuous development of OpenFlow are required.

The generalized conclusion of this master thesis is that *SDN networks can be managed following the FCAPS model.*

5.2 Concluding statements

Based on the evaluation of management functions that have been categorized according to the FCAPS model, the following statements are concluded:

- OpenFlow based SDN network management can be effective in most management functions implemented within the FCAPS approach.
- Network management cannot be based exclusively on ONF SDN architecture, OpenFlow and OF-Config.
- Fault management, performance management and security management in SDN networks, encounter effectiveness issues due to:
    - Lack of ONF SDN intrinsic mechanism(s) to support fast detection of failed links
    - Lack of mechanisms that ensure trust between software applications and SDN controllers

- Since SDN network management techniques and protocols have not yet been standardized, hardware selection is constrained by network management supported features. Regarding the selection of SDN equipment, it is not enough for data plane hardware to be compatible with particular OpenFlow versions; it must also be compatible with traditional management protocols (e.g. SNMP)
- SDN's data plane equipment can implement several firewall operations that can improve security. However, SDN's data plane equipment cannot replace traditional firewall operation due to lack of statefulness.
- Programmability is involved in almost all management functions

5.3 Recommendations and future work

There are still unresolved issues in network management of SDNs that require further investigation. However, their impact is mitigated by traditional management methods, proprietary solutions and excessive programming effort. OpenFlow, OF-Config and SDN programmability, play key roles in network management. SDN management can become more effective by upgrading existing SDN protocols or by introducing new ones in order to support:

- fast detection and fast response to network changes
- accurate real-time statistics
- authentication and trust mechanisms
- the ability to exchange device-centric data between the SDN planes

Future investigation that will focus on the above areas will allow SDN administrators to optimize the allocation of network resources, minimize the costs of proprietary management techniques, reduce security risks and eliminate dependence on traditional mechanisms.

Bibliography

[1] KimH, Feamster N. Improving network management with software-defined networking. IEEE Comm Mag, 2013

[2] ONF White paper. Software-Defined Networking: The New Norm for Networks. Open Networking Foundation, 2012

[3] Sakir, S. Scott-Hayward, S. Chouhan, K, Fraser, B. Finnegan, J. VilJoen, N. Miller, M. Rao, N. “Are ready for SDN? Implementation Challenges for Software-Defined Networking”, IEEE Comm Mag, 2013

[4] Clemm, A. Network Management Fundamentals. Cisco Press, 2007

[5] Kurose, J. Ross, K. Computer Networking: A Top-Down Approach. Pearson, 2013

[6] Verma, D. Principle of Computer Systems and Network Management, Springer, 2009

[7] ONF White paper. OpenFlow-enabled SDN and Network Functions Virtualization. Open Networking Foundation, 2014

[8] Aboba, B. Arkko, J. Harrington, D. Introduction to Accounting Management, RFC 2975, IETF, 2000

[9] Cisco White paper. Network Reference Management Architecture, Cisco, 2008

[10] Boucadair, M. Jacquenet, C. SDN: A Perspective from within a Service Provider Environment. IETF, 2014

[11] Goranson, P. Blach, C. Software Defined Networks: A comprehensive approach. Morgan Kaufman, 2014

[12] ONF White paper. OpenFlow Switch Specification Version 1.5.0. Open Networking Foundation, 2014

[13] Sharma, S. Staessens, D. Colle, D., Pickavet M. Demeester, P. Fast failure recovery for in-band OpenFlow networks. IEEE Comm Mag, 2013

[14] Sharma, S. Staessens, D. Colle, D., Pickavet M. Demeester, P. OpenFlow: Meeting Carrier-Grade requirements. Computer Communications, 2013

[15] Sharma, S. Staessens, D. Colle, D., Pickavet M. Demeester, P. Enabling Fast failure recovery for in-band OpenFlow networks, DRCN, 2011

[16] Niven-Jenkins, B. Brungard, D. Betts, M. Sprecher, N. Ueno, S. MPLS-TP requirements, RFC5654, IETF 2009.

[17] Capone, A. Cascone, C. Nguyen, A. Sanso, B. Detour Planning for Fast and Reliable Failure Recovery in SDN with OpenState. DRCN, 2015

[18] Kempf, J. Bellagamba, E. Kern, A.Jocha, D. Takacs, A. Scalable Fault Management for OpenFlow, IEEE ICC, 2012

[19] Niels, L. Benjamin, J. Kuipers A. Fast Recovery in Software-Defined Networks, EWSDN, 2014

[20] Devlic, A. John, W. Skoldstrom, P. A use-case based analysis of network management functions in the ONF SDN model. EWSDN, 2012

[21] ONF White paper. OpenFlow Management and Configuration Protocol. Open Networking Foundation, 2014

[22] Canini, M. Venzano. D, Kostic, D. Rexford, J., Peresini, P. A NICE way to test OpenFlow applications, NSDI, 2012

[23] Reitblat, M. Foster, N. Rexford, J. Schlesinger, C. Abstractions for Network Update, ACM SIGCOMM, 2012

[24] Yamashita, S. Tanaka, H. Hori, Y. Development of Network User Authentication System using OpenFlow, BWCCA, 2013

[25] Adrichem, N. Doerr, C. Kuipers, A. OpenNetMon, Network Monitoring in OpenFlow Software-Defined Networks, NOMS, 2014

[26] Yu, M. Jose, L. Miao, R. Software Defined Traffic Measurement with OpenSketch, NSDI, 2013

[27] Du. Q, Zhuang, H. OpenFlow-Based Dynamic Server Cluster Load Balancing with Measurement Support, Journal of Communications, 2015

[28] Porras, P. Cheung, S. Fong, M. Skinner, K. Yegneswaran, V. Securing the Software-Defined Network Control Layer, NDSS, 2015

[29] Mehdi, S. Khalid, J. Khayam, S. Revisiting Traffic Anomaly Detection Using Software Defined Networking, Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, Sommer, D. Balzarotti, and G. Maier, Eds. Springer, Berlin Heidelberg, 2011

[30] Schechter, S, Jung, J. Berger, A. Fast detection of scanning worm infections, RAID-2004-LNCS, Springer Heidelberg, 2004

[31] Williamson, M.: Throttling viruses: Restricting propagation to defeat malicious mobile code, ACSAC, 2002

[32] Gu, Y. McCallum, A. Towsley, D. Detecting anomalies in network traffic using maximum entropy estimation. ACM SIGCOMM, 2005

[33] Cai, Z. Cox, A., Eugene, N. Maestro: A System for Scalable OpenFlow Control, Rice University Technical Report, 2011

[34] Braga, R. Mota, E. Passito, A. Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow, IEEE LCN, 2010

[35] Haykin, S. Neural networks: a comprehensive foundation. Prentice Hall PTR Upper Saddle River, 1999

[36] Kreutz, D, Ramos, F., Verissimo, P. Towards Secure and Dependable Software-Defined Networks, ACM SIGCOMM, 2013

[37] Yan, Z. Prehofer, C. Autonomic Trust Management for a Component-Based Software System, IEEE Trans. Dep. and Sec. Computing, 2011

[38] Cryptanalysis.eu: SSL/TLS broken again, https://cryptanalysis.eu/blog/2013/03/15/ssltls-broken-again-a-weakness-in-the-rc4-stream-cipher/, 2013 (retrieved on 28/9/2015)

[39] Holz, R. X.509 Forensics: Detecting and Localizing the SSL/TLS Men-in-the-Middle, LNCS, 2012

[40] ONF, SDN Security Considerations in the Data Center, Open Networking Foundation, 2013

[41] Application Note, SDN Analytics for Elephant Flow Marking an Inherent, Scalable Solution for the Enterprise, Alcatel-Lucent, 2014

[42] Phaal, P. Panchen, S. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, RFC 3176, IETF, 2001

[43] Hu, H. Han, W. Ahn, G-J, Zhao, Z. FLOWGUARD: building robust firewalls for software-defined networks, HotSDN ’14, 2014

[44] Mukundha, C. Improving QoS in Cloud Based Networks with Software Defined Networks, International Journal of Science and Research, 2015

[45] Amdt, M. Kambli, R. Kittappa, T. Lauwers, C. Menezes, P. Rai, S. Tonogai, D. UC SDN Use Case Version 1.2: Automating QoS, Unified Communications Interoperability Forum, 2014

[46] Ghobadi, M. Yeganeh, S. Ganjali, Y. Rethinking End-to-End Congestion Control in Software-Defined Networks, Proceedings of the 11th ACM Workshop on Hot Topics in Networks, 2012

[47] Mininet.org: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what (retrieved on 1/11/2015)

[48] Feamster, N. Rexford, J. Zegura, H. The Road to SDN: An Intellectual History of Programmable Networks, ACM Queue, 2013

Appendix

*Table A*1: OpenFlow header matching fields.

	Field	Required	Description
1	OXM_OF_IN_PORT	YES	Ingress port (physical or logical)
2	OXM_OF_ACTSET_OUTPUT	YES	Egress port from action set
3	OXM_OF_ETH_DST	YES	Ethernet destination address
4	OXM_OF_ETH_SRC	YES	Ethernet source address
5	OXM_OF_ETH_TYPE	YES	Ethernet type of the OpenFlow packet payload (VLAN tag)
6	OXM_OF_VLAN_ID	NO	VLAN-ID from 802.1Q header
7	OXM_OF_VLAN_PCP	NO	VLAN-PCP from 802.1Q header
8	OXM_OF_IP_DSCP	NO	Diff Serv Code Point (DSCP) – Part of ToS (IPV4) or TC (IPv6)
9	OXM_OF_IP_ECN	NO	ECN bits in IP header – Part of ToS (IPV4) or TC (IPv6)
10	OXM_OF_IP_PROTO	YES	IPv4 or IPv6 protocol number
11	OXM_OF_IPV4_SRC	YES	IPv4 source address
12	OXM_OF_IPV4_DST	YES	IPv4 destination address
13	OXM_OF_TCP_SRC	YES	TCP source port
14	OXM_OF_TCP_DST	YES	TCP destination port
15	OXM_OF_TCP_FLAGS	NO	TCP flags
16	OXM_OF_UDP_SRC	YES	UDP source port
17	OXM_OF_UDP_DST_	YES	UDP destination port
18	OXM_OF_SCTP_SRC	NO	SCTP source port
19	OXM_OF_SCTP_DST	NO	SCTP destination port
20	OXM_OF_ICMPV4_TYPE	NO	ICMP type
21	OXM_OF_ICMPV4_CODE	NO	ICMP code
22	OXM_OF_ARP_OP	NO	ARP opcode
23	OXM_OF_ARP_SPA	NO	Source IPv4 address in ARP payload
24	OXM_OF_ARP_TPA	NO	Target IPv4 address in ARP payload
25	OXM_OF_ARP_SHA	NO	Source Ethernet address in ARP payload
26	OXM_OF_ARP_THA	NO	Target Ethernet address in ARP payload
27	OXM_OF_IPV6_SRC	YES	IPv6 source address
28	OXM_OF_IPV6_DST	YES	IPv6 destination address
29	OXM_OF_IPV6_FLABEL	NO	IPv6 flow label
30	OXM_OF_ICMPV6_TYPE	NO	ICMPv6 type
31	OXM_OF_ICMPV6_CODE	NO	ICMPv6 code
32	OXM_OF_IPV6_ND_TARGET	NO	The target address in an IPv6 Neighbor Discovery_message

	Field	Required	Description
33	OXM_OF_IPV6_ND_SLL	NO	
34	OXM_OF_IPV6_ND_TLL	NO	Link-layer target address option in IPv6 (Neighbor Discovery)
35	OXM_OF_MPLS_LABEL	NO	LABEL of the first MPLS shim header
36	OXM_OF_MPLS_TC	NO	TC of the first MPLS shim header.
37	OXM_OF_MPLS_BOS	NO	The BoS bit in the first MPLS shim header
38	OXM_OF_PBB_ISID	NO	I-SID in first PBB service instance tag.
49	OXM_OF_IPV6_EXTHDR	NO	IPv6 Extension Header pseudo-field.
40	OXM_OF_PBB_UCA	NO	The UCA field in the first PBB service instance tag.