DEGREE PROJECT IN COMMUNICATION SYSTEMS, FIRST LEVEL
STOCKHOLM, SWEDEN 2014

Security challenges within Software Defined Networks

GABRIEL SUND and HAROON AHMED

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Security challenges within Software Defined Networks

Gabriel Sund and Haroon Ahmed

2014-11-13

Bachelor's Thesis

Examiner and academic adviser
Professor Gerald Q. Maguire Jr.

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

A large amount of today's communication occurs *within* data centers where a large number of virtual servers (running one or more virtual machines) provide service providers with the infrastructure needed for their applications and services. In this thesis, we will look at the next step in the virtualization revolution, the virtualized network. Software-defined networking (SDN) is a relatively new concept that is moving the field towards a more software-based solution to networking. Today when a packet is forwarded through a network of routers, decisions are made at *each* router as to which router is the next hop destination for the packet. With SDN these decisions are made by a centralized SDN controller that decides upon the best path and instructs the devices along this path as to what action each should perform. Taking SDN to its extreme minimizes the physical network components and increases the number of virtualized components. The reasons behind this trend are several, although the most prominent are simplified processing and network administration, a greater degree of automation, increased flexibility, and shorter provisioning times. This in turn leads to a reduction in operating expenditures and capital expenditures for data center owners, which both drive the further development of this technology.

Virtualization has been gaining ground in the last decade. However, the initial introduction of virtualization began in the 1970s with server virtualization offering the ability to create several virtual server instances on one physical server. Today we already have taken small steps towards a virtualized network by virtualization of network equipment such as switches, routers, and firewalls. Common to virtualization is that it is in early stages all of the technologies have encountered trust issues and general concerns related to whether software-based solutions are as rugged and reliable as hardware-based solutions. SDN has also encountered these issues, and discussion of these issues continues among both believers and skeptics. Concerns about trust remain a problem for the growing number of cloud-based services where multitenant deployments may lead to loss of personal integrity and other security risks. As a relatively new technology, SDN is still immature and has a number of vulnerabilities. As with most software-based solutions, the potential for security risks increases. This thesis investigates how denial-of-service (DoS) attacks affect an SDN environment and a single-threaded controller, described by text and via simulations.

The results of our investigations concerning trust in a multi-tenancy environment in SDN suggest that standardization and clear service level agreements are necessary to consolidate customers' confidence. Attracting small groups of customers to participate in user cases in the initial stages of implementation can generate valuable support for a broader implementation of SDN in the underlying infrastructure. With regard to denial-of-service attacks, our conclusion is that hackers can by target the centralized SDN controller, thus negatively affect most of the network infrastructure (because the entire infrastructure directly depends upon a functioning SDN controller). SDN introduces new vulnerabilities, which is natural as SDN is a relatively new technology. Therefore, SDN needs to be thoroughly tested and examined before making a widespread deployment.

Keywords

Software Defined Networks (SDN), network security, denial of service, distributed denial of service, multi-tenancy

Sammanfattning

Dagens kommunikation sker till stor del via serverhallar där till stor grad virtualiserade servermiljöer förser serviceleverantörer med infrastukturen som krävs för att driva dess applikationer och tjänster. I vårt arbete kommer vi titta på nästa steg i denna virtualiseringsrevolution, den om virtualiserade nätverk. mjukvarudefinierat nätverk (eng. Software-defined network, eller SDN) kallas detta förhållandevis nya begrepp som syftar till mjukvarubaserade nätverk. När ett paket idag transporteras genom ett nätverk tas beslut lokalt vid varje router vilken router som är nästa destination för paketet, skillnaden i ett SDN nätverk är att besluten istället tas utifrån ett fågelperspektiv där den bästa vägen beslutas i en centraliserad mjukvaruprocess med överblick över hela nätverket och inte bara tom nästa router, denna process är även kallad SDN kontroll.

Drar man uttrycket SDN till sin spets handlar det om att ersätta befintlig nätverksutrustning med virtualiserade dito. Anledningen till stegen mot denna utveckling är flera, de mest framträdande torde vara; förenklade processer samt nätverksadministration, större grad av automation, ökad flexibilitet och kortare provisionstider. Detta i sin tur leder till en sänkning av löpande kostnader samt anläggningskostnader för serverhallsinnehavare, något som driver på utvecklingen.

Virtualisering har sedan början på 2000-talet varit på stark frammarsch, det började med servervirtualisering och förmågan att skapa flertalet virtualiserade servrar på en fysisk server. Idag har vi virtualisering av nätverksutrustning, såsom switchar, routrar och brandväggar. Gemensamt för all denna utveckling är att den har i tidigt stadie stött på förtroendefrågor och överlag problem kopplade till huruvida mjukvarubaserade lösningar är likvärdigt robusta och pålitliga som traditionella hårdvarubaserade lösningar. Detta problem är även något som SDN stött på och det diskuteras idag flitigt bland förespråkare och skeptiker. Dessa förtroendefrågor går på tvären mot det ökande antalet molnbaserade tjänster, typiska tjänster där säkerheten och den personliga integriten är vital. Vidare räknar man med att SDN, liksom annan ny teknik medför vissa barnsjukdomar såsom kryphål i säkerheten. Vi kommer i detta arbete att undersöka hur överbelastningsattacker (eng. Denial-of-Service, eller DoS-attacker) påverkar en SDN miljö och en singel-trådig kontroller, i text och genom simulering.

Resultatet av våra undersökningar i ämnet SDN i en multitenans miljö är att standardisering och tydliga servicenivåavtal behövs för att befästa förtroendet bland kunder. Att attrahera kunder för att delta i mindre användningsfall (eng. user cases) i ett inledningsskede är också värdefullt i argumenteringen för en bredare implementering av SDN i underliggande infrastruktur. Vad gäller DoS-attacker kom vi fram till att det som hackare går att manipulera en SDN infrastruktur på ett sätt som inte är möjligt med dagens lösningar. Till exempel riktade attacker mot den centraliserade SDN kontrollen, slår man denna kontroll ur funktion påverkas stora delar av infrastrukturen eftersom de är i ett direkt beroende av en fungerande SDN kontroll. I och med att SDN är en ny teknik så öppnas också upp nya möjligheter för angrepp, med det i åtanke är det viktigt att SDN genomgår rigorösa tester innan större implementation.

Nyckelord

mjukvarudefinierat nätverk, nätverkssäkerhet, överbelastningsattack, distribuerad överbelastningsattack, multitenans

Acknowledgements

First and foremost we would like to thank our examiner, Professor Gerald Q. Maguire Jr. for his fantastic support during the process of writing this thesis. We are truly blessed to have had such a knowledgeable examiner in the subject.

We would also like to thank our tutor at TeliaSonera, Nina Roed, for her patience, and guidance. She made it easy for us to get in touch with TeliaSonera personnel and also other companies within the field of networking, for discussion and insight on the matter we looked into. Her patience and support helped us overcome the obstacles we faced throughout the project.

Last, but not least, we would like to thank our families for their continuous support during our studies at the KTH Royal University of Technology, School of ICT.

Also a special thanks also goes to postgraduates Guofei Gu and Seungwon Shin at the Texas A&M University for sharing their work on DoS-attacks in an SDN environment.

Stockholm, 12 November 2014

Haroon Ahmed

Gabriel Sund

Table of contents

List of Figures

List of acronyms and abbreviations

Abbreviation	Description
API	Application Programmable Interface
ASIC	Application Specific Integrated Circuit
BGP	Border Gateway Protocol
DC	Data Center
DDC	Dual Data Center
DDoS	Distributed Denial of Service
DEI	Drop Eligible Indicator
DoS	Denial of Service
FTag	Forwarding Tag
HaaS	Hardware as a Service
IaaS	Infrastructure as a Service
IGP	Interior Gateway Protocol
IS-IS	Intermediate System to Intermediate System
LAN	Local Area Network
MAC	Media Access and Control
MPLS	Multi layer Protocol Label Switching
NFV	Network Function Virtualization
ONF	Open Networking Foundation
ONE	Open Network Environment
OS	Operating System
OSPF	Open Shortest Path First
PaaS	Platform as a Service
PCP	Priority Code Point
pps	packets per second
QoS	Quality of Service
RHEL	Red Hat Enterprise Linux
RIP	Routing Information Protocol
SaaS	Service as a Service

SDDC	Software Defined Data Center
SDN	Sofware Defined Networking
SLA	Service Level Agreement
STP	Spanning Tree Protocol
TCI	Tag Control Information
TPID	Tag Protocol Identifier
TSIN	TeliaSonera Internal Network
TSS	TeliaSonera Service
TTL	Time To Live
VID	(IEEE 802.1Q) VLAN Identifier
VLAN	Virtual Local Area Network
VLAN-id	VLAN identifier
VM	Virtual Machine
VNI	VXLAN Network Identifier
VTEP	VXLAN Tunnel End Point
VXLAN	Virtual Extensible Local Area Network

1 Introduction

Software-defined networking (SDN) is an emerging computer networking paradigm. The term itself has only been around for a couple of years. SDN moves the focus to software from the hardware, by separating the control plane of today's routers and routing switches and moving this control plane to (centralized) software. SDN enables network administrators to manage and operate the network via abstraction of the lower level functionality within the Open Systems Interconnection model (OSI-model).

Our initial task was to find the most significant security risks when implementing SDN within one of TeliaSonera's data centers and to suggest how to manage and/or mitigate these risks. After our literature study and meetings with our advisers at TeliaSonera the most critical security risks in an SDN environment that were identified were Distributed Denial of Serivce (DDoS) attacks and overall multi-tenancy issues. However, because SDN is not (yet) widely implemented it is hard to come to definite conclusions.

The rest of this chapter describes the specific problem that this thesis addresses, the context of the problem, the goals of this thesis project, and outlines the structure of the thesis.

1.1 General introduction to SDN

SDN offers a dynamic approach to networking by separating (decoupling) the control plane and data forwarding plane of network devices. The data plane is responsible for the actual forwarding of the data packets through the network using the paths chosen by the control plane. The control plane realizes the intelligence of a network. When the control plane is implemented in the hardware of a device (e.g. a router) forwarding decisions are based upon matching entries in a routing table stored in the router's memory. The entries in this routing table are based upon information about the network's topology. Routing can utilize static routes, where network administrators explicitly program routers to use a certain path to reach a certain destination within or outside of the network that they administer. Alternatively, dynamic routing uses dynamic routing protocols* to generate entries in the routing table. In the case of dynamic routing protocols, the routers within the network help one another by spreading network topology information, thus making it possible to for each router to decide how to forward packets along suitable paths through the network. In addition, there are multipath routing technologies, such as FabricPath - described in Section 2.3.4.

By decoupling the data and control planes routing decisions can be centralized and made by software, rather than decentralized decisions at every router within the network. In SDN, the network is controlled through an application-programming interface (API). This API enables innovation and offers new possibilities for configuring, managing, and optimizing the network for specific flows of traffic. This in turn offers great opportunities for controlling and adapting the network to meet specific needs during everyday usage.

The separation of the control- and data plane introduces a need for some protocol to support the communication between these two planes. One such protocol is called "OpenFlow". This protocol which will be described in more detail in Section 2.1. Dan Pitt, Executive Director of the Open Networking Foundation describes OpenFlow as:

> *"OpenFlow, as a standard, lays the foundation for a new network software discipline, working towards a high-level language that will make networks as readily programmable as a PC" [1]*

There are numerous benefits of SDN for both users and managers of the network. For example, the network could operate more effectively as the network manager can prioritize certain data packets in real-time via the SDN controller, thus optimizing data flows and exploiting the flexibility of SDN to use alternative paths for other traffic. Using these alternative paths reduces the latency of some of the traffic at the cost in increased latency for other traffic. Additionally, using these alternative paths distributes the load over a larger number of paths, thus potentially allowing the network operator to delay scaling up their physical network, reducing their capital expenditures.

SDN provides an API making the network programmable. This ultimately enables applications to be aware of the network, and enables the network to be aware of the needs of applications. Both of these enable improved automation and controlling of the network and traffic flows. This API improves

* Such as Open Shortest Path First (OSPF) or Routing Information Protocol (RIP).

the use of existing resources and allows greater innovation in the future, bringing the rate of evolution of networks (and network protocols) closer to the rate of software development.

Today's data centers are built using a large number of different network devices for routing, load balancing, switching, etc. Because many companies employ a multiple vendor strategy for their purchases, the network consists of a heterogeneous collection of devices, i.e., with different devices produced by different manufacturers. The diversity of devices increases the complexity of configuration and management because there are generally vendor specific APIs to take into consideration. This requires either a network management solution that can deal with each of these different APIs (such as Tail-f Systems Network Control System [2]) or use of SDN where the configuration is done through a centralized API and standardized for the whole network.

1.1.1 The security issues

Systems are becoming increasingly complex. This complexity in turn has led to increased risk and severity of bugs and errors in implementations. This complexity increases with virtualization within networks, as increasing numbers of traditional hardware functions are realized by software. This puts a large amount of pressure upon programmers to deliver flawless software solutions. Additionally, this pressure is increasing due to the business trend toward cloud computing.

A major advantage of software realizations of functionality is that this software can be rapidly deployed to a large number of computers, thus enabling the functionality to scale up or down to follow demand. Unfortunately, a corollary to this is that a single exploit in the software could affect a large number of users and their personal & data integrity. An example of such an error in software, is the programming mistake resulting in the *Heartbleed-bug* discovered in OpenSSL. This error allows anyone to access and steal information which should be protected by encryption technologies (such as SSL/TLS)[3].

One of the most common types of security problems today is denial-of-service attacks (DoS). A DoS attack denies requests from legitimate users being served by the target of the attack. The main goal of a DoS attack is to make the victim’s service unavailable, hence the victim will be unable to provide the expected service to its customers. For example, a DoS attack on a web server would make the web service unavailable to legitimate user’s browsers. This is achieved by flooding the web server with more requests that it can handle, thus interrupting and/or suspending the service provided by this web server. DoS attacks have been carried out for political, economic, and malicious purposes. For some examples of such attacks see [4] and [5].

Another security risk that is important to consider is the growing trend of employees bringing their own devices, such as smartphones, tablets, computers, etc. to their workplace. This policy is often referred to as Bring Your Own Device (BYOD). Allowing all these personal devices to connect to the network increases the chances of a device internally infecting the network with malicious software. Such an infection could lead to illegitimate access to sensitive information since these devices frequently have access to sensitive company information.

1.1.2 Multi-tenancy

Today multi-tenancy is used more and more together with virtualization. Multi-tenancy means multiple customers share a single hardware instance, often sharing the same network interface and storage infrastructure. However, multi-tenancy can cause challenges for service providers as the different customers are usually isolated by having their processing done in different virtual machines (VMs) running on a hypervisor. In single-tenancy, customers are each assigned their own server and storage within a data center. Today many customers who demand high availability and high security for their operations require a single-tenancy solution. Moreover, there is always a risk of human error and in a multi-tenant architecture; such an error could affect multiple users. For example, if an attacker is able to circumvent encryption for a shared database in a multi-tenant service, then the attacker would be able to access data of all the particular database instances of these users. Thus a number of different customers could all be negatively affected at once[6].

An important characteristic of multi-tenancy is dynamic scaling. In addition to this scaling, the service provider needs to be able to fulfill each customer’s specific service level agreement (SLA) and guarantee isolation of each customer’s data from that of other customers. Today isolation is realized on the networking layer by placing each customer into a specific virtual local area network (VLAN) and on the server layer by running each customer’s computation in a VM.

1.2 Problem definition

This thesis project began with a literature study to provide relevant background information about the field and to give us a firm base to stand on as we moved ahead with this thesis project.

Defining our problem began with a case study agreed upon with our industrial supervisors at TeliaSonera. The problem that was identified via this first case study is: How does a SDN controller react to a DDoS attack and how to provide security in an SDN environment? The focus was to be on how to manage and mitigate distributed denial of service (DDoS) attacks on TeliaSonera's data center web-applications.

A second case study concerned how to establish trust amongst customers for multi-tenancy services in an SDN environment, where everything is virtualized and the underlying hardware is shared. This second case study also examined the major differences in how multi-tenancy is implemented today in TeliaSonera's cloud environment and how it could be implemented in a future SDN environment.

1.3 Purpose

The purpose of this thesis project was to help TeliaSonera understand how they could exploit the adoption of SDNs in their data centers. The advantages for them as a company are: (1) to decrease capital expenditures by making better use of their existing infrastructure; (2) increase their ability to manage the networking resources within their data centers; (3) reduce their energy and cooling costs, as a smaller pool of servers can be shared in a multi-tenancy configuration; and (4) make the advantages of their adopting SDN available to their customers (in terms of improved security, reducing time to market, lower costs via scalability, and simplifying configuration and management).

As noted above, TeliaSonera's customers will also gain from the adoption of SDN within the datacenter. However, it is important that TeliaSonera properly address the security and multi-tenancy issues so that their customers' personal and data integrity & privacy can be ensured. If they are not successful in addressing these issues, then the gains from adopting SDN will be reduced and they would risk damaging their reputation if customers' information were to be leaked or made accessible to unauthorized parties, thus running a risk of losing current and potential customers.

1.4 Goals

The goals set up are organized into seven stages – listed in Table 1-1. Stages 1 through 4 led to Chapters 1 and 2 and are based on the initial literature study. The remaining five weeks of the project focused upon the two case studies and the problems identified in Section 1.2.

Table 1-1: The project's seven stages

Stage 1	**The Start**: Identify a problem in cooperation with TeliaSonera. The output was a draft project plan.
Stage 2	**Project Planning**: The output was a project plan with scheme, timetable, milestones, objectives, etc.
Stage 3	**Primary Data Collection**: Literature studies and gathering of articles. Output of this stage were the literature study and a final project plan.
Stage 4	**Secondary Data Collection**: Meetings with TeliaSonera to discuss the literature studies and a guided tour of data center site.
Stage 5	**Case Studies**: Continuous discussions and interviews with TeliaSonera personnel and insight on chosen case studies.
Stage 6	Draft version of the thesis.
Stage 7	Oral thesis presentation and submission of the final version of the thesis.

1.5 Research Methodology

Our *modus operandi* (method of working) mainly consisted of gathering data via literature studies and via consultation with TeliaSonera. Communication with TeliaSonera were expected to be the most important source of information, while the literature studies would act as a complement to continued discussions with TeliaSonera.

Interviews with professionals who deal with different layers of the OSI-model were conducted to gain a broader view of the issues regarding multi-tenancy in the target environment. Their input provided the backbone of our analysis. Simulation and testing of a SDN network, with a single-threaded controller was benchmarked over the course of this project. This benchmarking was done to gain hands-on experience with SDN.

The research methodology chosen for this project was a qualitative research methodology (paradigm) because of its inductive and postmodernist nature. Consideration of the limited duration and the overall size of the project, as well as the fact that SDN has *not* yet been implemented in TeliaSonera's data centers, and our current knowledge about the field of interest were also taken into account when selecting this research methodology. Based upon our initial literature study and our meetings with TeliaSonera our understanding of the issues evolved, while our objectivity may have been colored by the research and our own work in the field. We rejected the use of a quantitative methodology, as it would have been a deductive and a non-value-loaded approach. Instead, we chose a qualitative research methodology approach. This approach was expected to give qualitative insights into the security issues of introducing a SDN into TeliaSonera's data centers for use with their web services. However, our study should be followed up in a future project via a quantitative evaluation *after* the implementation of SDN for these services has been completed.

1.6 Delimitations

One of the important limitations of this project was the limited duration of this project. This meant that this project could only study the *potential* impact of a future implementation of SDN in TeliaSonera's data centers - as this implementation has not yet been carried out and would not be carried out during the period of our thesis project. An additional limitation was our lack of knowledge concerning SDN, virtualization, and data centers when starting this project.

Out of scope for this thesis project is whatever happens outside of the data center’s site. In this project, the focus of our attention is solely on security related issues *within* a SDN environment, specifically DoS-attacks and how multi-tenancy is solved within a Software Defined Data Center (SDDC). Together with TeliaSonera we chose these two security challenges, because according to them these are the most likely to pose a threat to the company’s security. Therefore, other potential security related problems were not considered in this thesis project.

1.7 Structure of this thesis

Chapter 1 contains the introduction, definition, and purpose of this thesis project, it also provides a general and brief introduction to SDN and related security problems. Chapter 2 presents relevant background information about SDN related technologies, related work, and what has been done previously in the field of interest. Chapter 3 presents the methodology and process followed in this project in more depth. Chapter 4 presents the analysis and results of our work. Chapter 5 summarizes lessons learned and states our conclusions based upon the analysis in the previous chapter. This final chapter also concludes with some suggestions for future work and comments on some of the social, ethical, and sustainability aspects of this thesis project.

2 Background

The overview of SDN given in Section 1.1 is a very generic and general view. Major networking companies, such as Cisco, Alcatel-Lucent, Juniper, etc., support this approach to SDN. However, there are some differences in how these companies attempt to meet the challenges and exploit the opportunities provided by SDN.

Although VMware does not manufacture network devices, VMware is actively developing software and services for cloud management and virtualization. VMware's approach is based upon their NSX™ software that provides a virtual network and security platform. This software is distributed in each of the hypervisors running on the computers in the data center. A hypervisor isolates the VM and applications from the physical server. Applications running in a VM do not see any difference (other than potentially more limited throughput) between the virtual network and the underlying physical network. As a result, applications do not require any special configuration to run on the virtualized network.

The NSX network hypervisor placed between the physical and application layer does not affect the hardware in any way, facilitating hardware upgrades and exchanges. Whenever a data center owner starts to run out of computing resources or storage capacity, NSX enables more hardware to be added to the underlying physical network to provide increased scalability.

During a VMware seminar at TeliaSonera's headquarters in Farsta on 13th February 2014, VMware's representative Andy Kennedy (and Amir Khan) spoke about VMware's take on SDN and SDDCs. Mr. Kennedy talked about how the functions that hardware provides today will be provided by software in the future, thus the hardware simply provides computing-, network-, and storage capacity – as the intelligence currently in many specialized devices is decoupled from the underlying hardware. He also spoke about the provisioning times to set up services, such as VMs, as this provisioning time is a critical issue for service providers. He presented how it was easy to perform configuration and troubleshooting through a centralized API. He also discussed decreasing provisioning time and the addition of new costs (these costs are associated with VMware's plans for revenue generation).

The main driver for SDN is the emergence of cloud services. From a Juniper Networks' point of view they say, "Software defined networking is designed to merge the network into the age of the cloud". As a network equipment vendor Juniper Networks' approach to SDN is obviously quite different from that of VMware. Along with VMware and Cisco, Juniper Networks has also identified the data center as an environment ripe for SDN implementation. Juniper is adapting to this by shifting towards selling network equipment, but licensing their software separately, rather than selling the network equipment with software as of today [7].

Cisco's SDN solution is the Cisco Open Network Environment (ONE) architecture. This architecture is expected to help networks to become more open and programmable. ONE builds on a protocol called *OpenFlow*. Section 2.1 describes the OpenFlow protocol in detail.

2.1 OpenFlow: The first SDN standard

OpenFlow is often referred to as the first open standard for SDN. The Open Networking Foundation (ONF) is a user-driven organization that strives for standardization and promotion of SDN. OpenFlow enables interaction between the data forwarding plane and the SDN controller. For this environment to work all of the devices, communicating with the SDN controller must be OpenFlow-enabled. Several Cisco switching and routing devices support SDN and talk OpenFlow. Multiple vendors make OpenFlow compatible devices, thus it is often unnecessary to invest in new hardware or feel restricted to a specific vendor when implementing SDN.[14]

OpenFlow operates on layer 2 (the data link layer) in the OSI-model. An OpenFlow-switch consists of flow table(s) and a group table. This information is used when forwarding frames (see Figure 2-1). An OpenFlow device utilizes a secure channel to communicate with the SDN controller. Through this secure channel, packets (i.e. containing an Ethernet frame - including frame header and payload) and commands are sent using the OpenFlow protocol. The flow table consists of a set of flow entries. Each flow entity consists of *match fields*, *counters*, and *instructions*. Frames arriving at the switch are compared with the flow entries in the flow tables and if there is a match, then the set of instructions in that flow entry will be executed. The frame might be directed to another of the switch's flow tables for further processing. This procedure is called *pipeline processing*. When the instructions do not (re-)direct the frame anymore the pipeline processing has finished and the action set associated with the frame is executed. This execution usually forwards the frame on some interface.

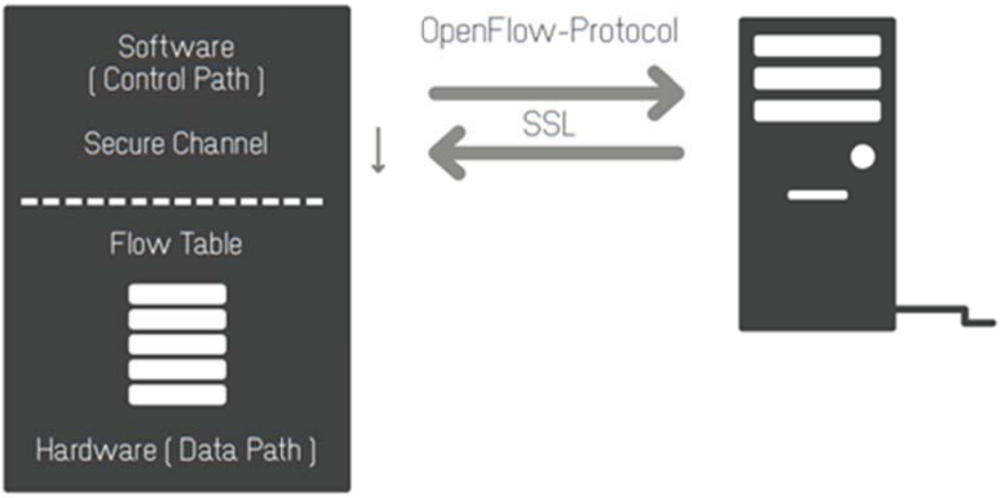

Figure 2-1: **An OpenFlow-switch communicating with an SDN controller over a secure connection (SSL) via the OpenFlow protocol (a recreation from Figure 1 of [15])**

There are 15 fields (shown in Figure 2-2) used when matching packets against flow entries. The flow entry could be more or less specified to control the network - depending on how the fields are set. If all fields are set, then the flow entry would be very specific and cover only a narrower range of possible frames. Creating flow rules covering a wider range of incoming frames is also possible by setting the "don't-care-bits"; this is beneficial when there is a need to limit the number of flow rules being created.

Ingress Port	Metadata	Ether src	Ether dst	VLAN ID	VLAN Priority	MPLS label	MPLS traffic class	IPv4 src	IPv4 dst	IPv4 proto / ARP opcode	IPv4 YoS bits	TCP/UDP/SCTP src port ICMP Type	TCP/UDP/SCTP dstport ICMP Code

Figure 2-2: **The fields are used when matching packets with flow entriesaccording to the OpenFlow Switch Specification version 1.1.0 Implemented [16].**

If the frame does not match any flow entry, i.e. a table miss occurs, then the frame is sent to the SDN controller over a secure channel by including the *packet-in* message. Depending on the configuration of the switch, another alternative would be to drop the packet. However, by default the frame is usually sent to the SDN controller via a *packet-in* message to ask the SDN controller to make a decision about how to handle this specific frame. A *packet-in* contains either a part of the frame header which by default is set to the first 128 bytes or the entire frame depending on whether the switch supports internal buffering or not. If the frame can be buffered, i.e. temporarily stored within the switch, then a buffer ID will be included with the *packet-in* message. The SDN controller should send a *packet-out* and/or a *flow-modification* message. The *packet-out* message sends the frame out a specified port on the original switch. This *packet-out* message must contain a list of actions; otherwise the frame will be dropped. If the entire frame was not sent from the switch to the controller, then the buffer ID will be referenced by the controller in the *packet-out* message. Using this buffer ID, the original frame that triggered the corresponding *packet-in* will be forwarded by the switch via the specified port.

Alternatively, the controller might send a *flow-modification* message. The main purpose of this message is to add, delete, or modify the tables in the switch. In the case of an incoming frame that lead to a table-miss, the controller can send a *flow-modification* message to instruct the switch to add a new flow entry in its flow tables so that this switch will know what to do with similar frame in the future.

Each flow entry contains a timeout value. The idle timeout is a lower limit on the amount of time an entry will remain in the table, if there is no activity within a certain amount of time then the flow entry will be removed. The hard timeout is an upper limit, which indicates when the flow entry must be removed regardless of activity.

Priorities for the matches of entries are also important, thus if there are multiple flow entries that match an incoming packet, then the flow entry with the highest priority is used.[16] [O13]

Further details of this protocol can be found in:
http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

2.2 Denial-of-service-attacks

Computer networking has come a long way, but even with today's advanced network architecture, there are vulnerabilities. DoS attacks are one of the most common security-related problems of servers today. A DoS attack can be accomplished by several methods, but most of these attacks can be categorized into one of three different methods: vulnerability attacks, connection flooding, and bandwidth flooding.

Vulnerability attacks take advantage of bugs or exploits in the service at the server. In this way, the service stops functioning and in the worst case, the server hosting the service could crash.

Connection flooding, also called TCP SYN flood attacks, occur when a large number of TCP connection attempts arrive at the targeted server. The attacker causes these TCP SYN packets to be sent, either by one source or by many sources[*]. When a TCP connection is being created, the client and server exchange messages to establish a TCP connection before they send any data. The first packet sent by the client has the SYN (synchronization) flag set and an initial sequence number. The server allocates a TCP control block and sends a SYN-ACK (synchronization-acknowledgement) back to the client along with the server's SYN flag sent to indicate that it is sending its own initial sequence number. The client would normally send an ACK (acknowledgement) back to server thus establishing the TCP connection. If the last step of the procedure does not occur, there is a half-open TCP connection. At some point the server will not be able to establish anymore connections until the half open TCP connections are closed (thus releasing the storage associated with their TCP control blocks), therefore all new legitimate connection establishment attempts will be denied.

Bandwidth flooding occurs when a large number of packets are sent (nearly) simultaneously by the attacker (or by hosts controlled by the attacker) to the targeted host. The target's incoming link will be choked (i.e., all of the available bandwidth will be used up) and legitimate usage of the server becomes constrained. In some cases, one attack machine is insufficient to cause sufficient damage. For example, such a bandwidth flooding DoS attack would fail when the targeted server has an access bandwidth much greater than the amount of traffic coming from the attacker. In this case, a DDoS attack would be used by the attacker. In a DDoS attack the attacker creates a network, often referred to as a botnet, by infecting multiple computers with viruses or Trojans. These infected computers are often called zombie computers. The attacker can now have a much larger impact on the targeted server because it can coordinate multiple zombies to generate traffic at a much higher aggregate rate. Figure 2-3 shows an attacker and a botnet of zombie computers performing a DDoS attack on a data center. Moreover, there is a problem detecting DDoS attacks, as it is not obvious that these multiple sources are in fact intent upon attacking the victim. This is unlike a regular DoS attack where all of the traffic is coming from a single source. When such as DoS attack occurs, one could simply block packets from this source. Unfortunately, DDoS attacks are very common today, although mounting such an attack is considered a crime in many countries. Note that it is very hard to defend against a DDoS attack, as one cannot easily know which sources to block. To date there have been 2-3 major DDoS attacks aimed at TeliaSonera's data centers.

[*] The later case in an example of a distributed DoS (DDoS) attack.

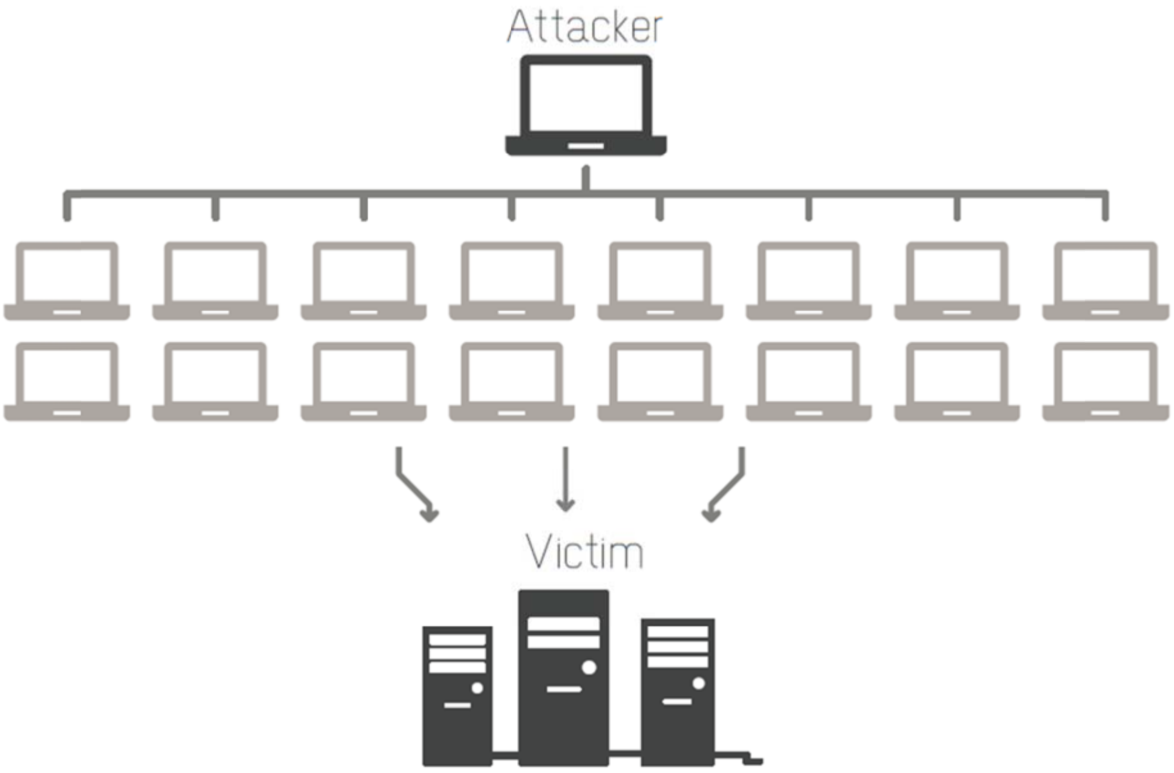

Figure 2-3: **An attacker and its botnet of zombie computers performing a DDoS attack on a data center (inspired by Figure 1.25 on page 85 of [17]).**

2.3 Layer two interconnections

Several different technologies are currently being used to realize layer two interconnections. Some of these are described in the following subsections.

2.3.1 Network bridges

Network bridging can be used to interconnect two or more LANs or VLANs. This bridging functionality can utilize a physical network bridge[*] or a virtual network bridge. As shown in Figure 2-4, bridges operate at layer 2. The bridged LANs are logically a single network segment, even though they are separate physical networks. Broadcast and multicast traffic in a local area network (LAN) place a burden on the network bridge and the network as a whole, thus it makes sense to divide a large network into smaller segments, be they virtual or physical segments. An example of a bridge that many users have at home is a Wi-Fi-enabled router. This device typically acts a bridge between its Ethernet interfaces and a wireless network interface, allowing both wired and wireless users to communicate despite being on different physical network segments [8]. Note that because the device realizes a bridged LAN an IP subnet's addresses can be used by any combination of devices on any of these segments. Despite the fact that we referred to the device as a Wi-Fi-enabled router, with respect to the local network segments it is typically configured to act as a bridge. From a security point of view, it would be preferable to configure this device as a router – so that the segments are isolated at the network layer, but this would require more complex configuration of the device by users[†].

[*] Such as a network switch.

[†] As usual security has been sacrificed for user ease.

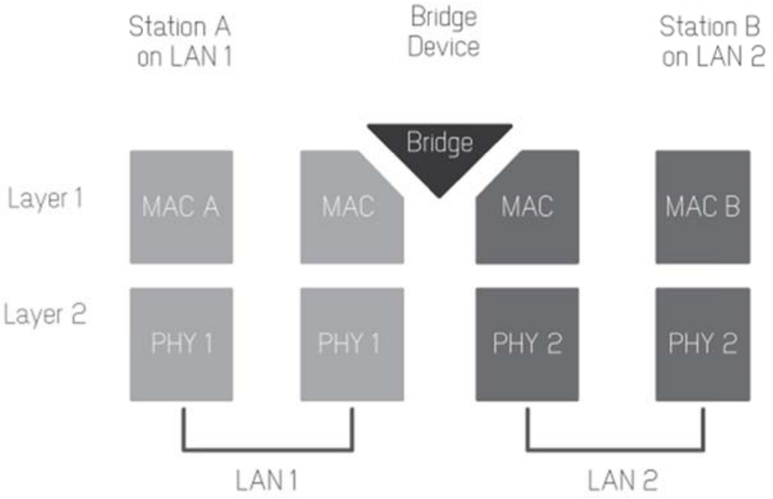

Figure 2-4: **Bridging two LANs (or VLANs) via a shared bridge entity on layer 2**

2.3.2 Multi-layer Protocol Label Switching

Multi-layer Protocol Label Switching (MPLS) is a multilayer protocol, often referred to as the "2.5 layer protocol" in the OSI-model, i.e. it is in between the traditional layer 3 routing and layer 2 switching layers. MPLS enables a variety of quality of service (QoS) features and traffic flows can be allocated to specific label switched paths to best utilize the current network capacity.

2.3.3 Spanning Tree Protocol

The Spanning Tree Protocol (STP) is a network protocol based on an algorithm developed by Radia Perlman. The protocol ensures loop-free forwarding in bridged networks by ensuring that there is only a single path active between two nodes in a network, thus preventing loops from occurring.

2.3.4 FabricPath

Cisco's FabricPath provides communication between two endpoint VMs. The process starts with a FabricPath network switch encapsulating an incoming Ethernet frame inside a 16-byte FabricPath header (as shown in Figure 2-5). When the Ethernet frame is about to exit the FabricPath network this FabricPath header is removed and the Ethernet frame is forwarded.

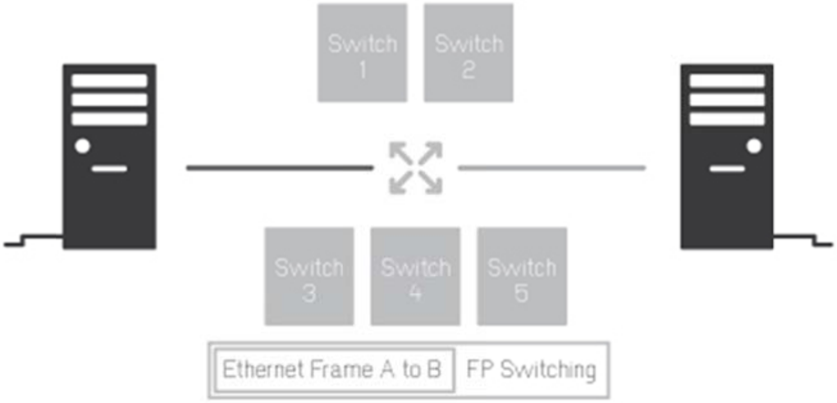

Figure 2-5: **Cisco's FabricPath header procedure.**

As soon as the Ethernet frame enters the FabricPath network (node 1) it is encapsulated and forwarded through the FabricPath network based upon MAC-addresses within the FabricPath network. The FabricPath header is referred to as the outer MAC header shown in Figure 2-6 (listed as

FP header). Whilst the encapsulated layer 2 Ethernet frame forwarding information is called the internal MAC (listed as iMAC below). What is unique about FabricPath is that the FabricPath header includes some layer 3 QoS functionality, such as Time-To-Live (TTL), at layer 2. The header also includes source and destination MAC addresses within the FabricPath network and a Forwarding Tag (FTag). The source address has fields for the source switch's ID, while the destination address field has a destination switch ID, and next a hop switch ID, called subswitch ID – as shown in Figure 2-6. The next hop switch ID identifies the next switch the frame will traverse on its way through the FabricPath network.

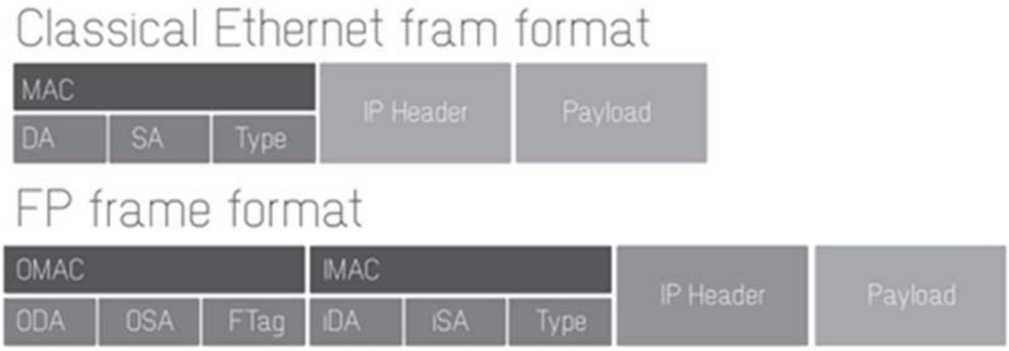

Figure 2-6: **FabricPath header and its components**

The FTag is used to determine the path the frame should take through the FabricPath network. The FTag has a 10-bit FTag field, hence it has a range from 0 to 1023. This enables the use of up to 1023 different paths through the network. A given FTag value is unique within a FabricPath network, thus this value does not changing during transmission through the network (nor do the source and destination field values change). This means that each network device along the path does not alter the path chosen by the initial edge FabricPath device that encapsulated the Ethernet frame. The TTL-field prevents loops from occurring and this field is decremented by 1 at every FabricPath device as the frame is forwarded within the FabricPath network. The subswitch ID is based upon the FTag field value and this determines the path that a frame travels through the FabricPath network. Notably the subswitch ID and FTag are the only fields that are altered in the outer MAC. FabricPath uses the IS-IS link-state routing protocol. [6, 8, 9] For further details see: http://www.packetmischief.ca/2012/04/17/five-functional-facts-about-fabricpath/.

2.4 Solving multi-tenancy in SDN

VLANs are currently widely used by service providers to isolate traffic within their data centers. The VLAN creates a logical cluster within a physical network, turning the physical network into multiple virtualized networks. Isolation is provided between these separate VLANs, since the traffic in one VLAN does not see the traffic within other VLANs due to the unique VLAN-id for each VLAN. This allows service providers, such as TeliaSonera, to cluster services and applications within their data centers. VLANs are also commonly used to logically isolate traffic within large networks with many different traffic flows, be the traffic from in-house departments or external customers.

The IEEE 802.1Q VLAN-tag includes a VLAN identifier (VLAN-id) that is 12 bits in size, hence there are 4096 (i.e. 2^{12}) unique VLAN-tags for each network. This is a problem that TeliaSonera has encountered and looks to solve via implementation of Virtual Extensible Local Area Networks (VXLAN). Details of VXLAN are described in Section 2.4.2.

2.4.1 VLAN

A VLAN is a layer 2 technology that enables isolation of broadcast traffic from VMs. These VMs can be located on a single host or on different physical hosts. Using VLANs also allows VMs to be moved to another physical host within a datacenter or Dual Data Center (DDC) *without* interfering with the VLAN configuration, i.e. there does not need to be a change to the VLAN-id.

TeliaSonera uses VLANs to isolate services and applications internally as well as externally to customers within their data centers. For example, in TeliaSonera Service (TSS) e-mail, there are various VLANs used within specific areas, such as network area, front end (web), and back end (application). Additionally, there are specific VLANs for backup and server management.

The IEEE 802.1Q standard defines VLANs on an Ethernet network. This standard describes how Ethernet frames are tagged with a VLAN tag, as shown in Figure 2-7. This standard also defines how switches and other network devices process and handle VLAN tagged frames along a path.

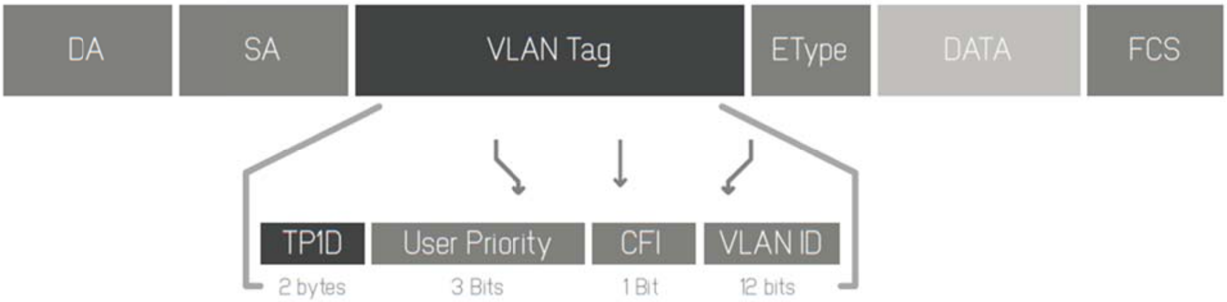

Figure 2-7: An Ethernet frame with added VLAN tag

The 4 byte VLAN-tag consists of a 16-bit Tag Protocol Identifier (TPID). A 3-bit User Priority Code Point (PCP), 1 bit Drop Eligible Indicator (DEI) - this bit is also known as the CFI-bit shown in the figure. These two together with the 12 bit VLAN ID (VID) make up the 16-bit Tag Control Information-field (TCI). The TPID and TCI are a total of 32-bits long.

The TPID field is by default set to 0x8100 when an Ethernet frame is VLAN tagged. PCP indicates the priority of the frame, with a system of different priority levels ranging from 0 to 7 (as there are 3-bits). Traffic is categorized into voice, video, audio, data, etc. and an appropriate PCI value set. DEI is a one bit field set to 0 or 1, where 1 indicates that this frame is drop eligible, i.e., that this frame could be dropped t in the event of traffic congestion. For example, a VoIP frame containing audio or video data might be marked as drop-eligible, indicating that it could be dropped if there is congestion.

IEEE 802.1ad extends the standard VLAN tagging standardized via stacked VLANs. The idea is to stack multiple VLAN headers on top of each, and that way expand the number of available unique VLANs. No extra functionality is added, other than the obvious increase in the number of available VLANs.

2.4.2 VXLAN

VXLAN is a layer 2 overlay on a layer 3 network. This allows VMs on different IP networks to operate as if they were connected to the same layer 2 network. In the VXLAN-tag, the VXLAN Network Identifier (VNI) is 24 bits allowing for a maximum of 16 million (i.e. 2^{24}) unique tags within a network domain. Each individual layer 2 overlay is called a VXLAN segment. Only VMs within the same VXLAN segment can communicate with one another. The communicating VMs must use same VNI *and* the same VLAN-id.

Whenever a VM generates and sends a frame, VXLAN encapsulation takes place at the physical host machine's hypervisor or possibly at a switch. As a result, the VM does *not* need any specific VXLAN configuration *and* it is unaware of the underlying VXLAN segment, hence the VM simply inserts a destination MAC addresses as usual (using either IPv4 ARP or IPv6 neighbor discover mechanisms). The encapsulation and de-encapsulation process occurs at a so-called VXLAN Tunnel End Point (VTEP). A VTEP is an endpoint and can realized through hardware or software. It is called a tunnel endpoint because the encapsulation is in effect from the sender VTEP to the receiver VTEP – thus realizing a "VXLAN tunnel" between the two end points.

Figure 2-8 illustrates the encapsulation and the added components inserted by the sender VTEP. Since a VXLAN is an overlay running on top of a layer 3 network it requires both IP source and destination addresses and MAC source and destination addresses for the physical interface of the destination interface. Note that the inner destination MAC address specifies which VM on the physical host that the packet is intended for. The VTEP does VNI lookups to see if the communication is *within* the same segment or not. The VTEP also performs encapsulate or de-encapsulate as necessary, hence the VM is unaware of the underlying transportation means. The (receiving) VTEPs[*] also stores the corresponding IP and MAC address mapping information in a table, so that a response packet does not need to be flooded throughout the network.[10]

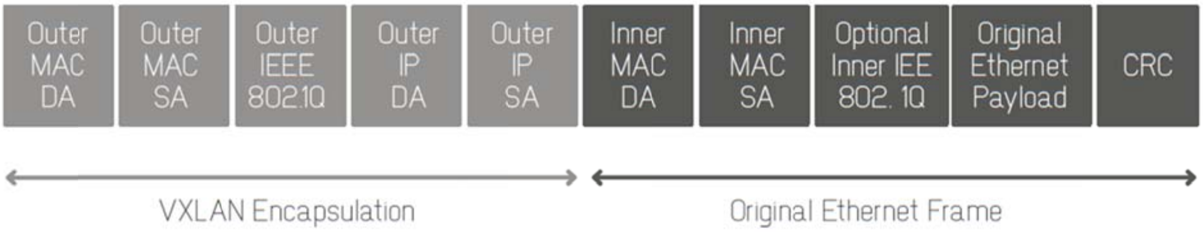

Figure 2-8: VXLAN encapsulation

2.4.3 Virtualization

Virtualization is an old concept within computer science, dating back to the mainframes of the 1960s, while the idea of virtual memory dates from 1956. In these early virtual machines, engineers partitioned computing and memory resources for use by different applications and allowed multiple operating systems to run on the same mainframe (for example, supporting both an interactive time-sharing system and one or more batch processing systems). In the early part of the 2000s virtual PCs and server virtualization took off in earnest. Virtualization has come a long way since then, and SDN is the next step in the virtualization revolution. The purpose of this virtualization is quite simple: to enable a physical resource, for example a physical server, to run multiple OSs and applications while providing isolation between the separate VMs. Virtualization makes the architecture easier to manage and potentially increases the utilization of physical servers[†].

There are two types of different server virtualization models (shown in Figure 2-9): bare-metal (also called "native") and hosted. TeliaSonera servers runs an OS, most commonly Red Hat Enterprise Linux (RHEL) or Microsoft's Windows Server. On top of this OS a hypervisor software is running, through this hypervisor VMs are created. The hypervisor also manages and operates the VMs running on top of it, ensure they get the computing and memory resources they need to operate. These resources can be specified through different SLAs with customers. Amazon's Elastic Compute Cloud is an example of another company that provides VMs to external customers with specific SLAs via their own data centers. The SLAs specify that certain levels of computing and memory resources are to be allocated for the customer. These resources are specified to and enforced via the hypervisor. The customer pays for the resources that are dynamically allocated for them at a price level related to their SLA.

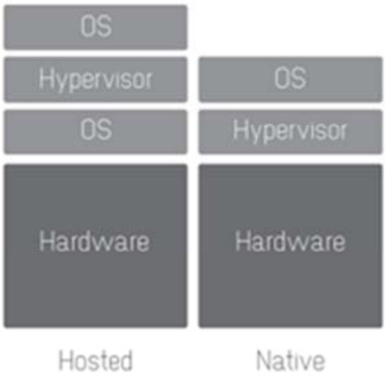

Figure 2-9: Alternative server virtualization stacks

[*] It is important to note that multiple VMs may receive a given frame, since VXLAN needs to support the broadcast and multicast semantics that both Ethernet and VLANs both support.

[†] This increased utilization of physical servers is thought to be an important means of decreasing energy consumption, as running idle processors also consume energy, while truly idle servers could be powered down to save a lot of energy.

Each of these two alternative virtualization stacks has an OS that actually realizes the VM. On top of this OS the customer specific applications are run. The difference between these two models is whether the hypervisor runs on another OS (referred to as the hosted server virtualization model) or whether the hypervisor runs directly on the hardware (referred to as a native-hypervisor stack). The native hypervisor mode is generally thought to be faster and more scalable.

The left-hand figure in Figure 2-10 illustrates a non-virtualized model (i.e., a traditional model) with three physical servers that are 5% to 30% utilized. Figure 2-10 shows two server models, virtualized (multi-tenant) and a traditional single-tenant approach. The three physical servers have been replaced by one physical server running three VMs in the virtualized model. The total utilization of the server capacity therefore is 55% (the grand total for the three physical servers in the non-virtualized model). As a result of using this virtualized model, the other two physical servers can now be used to support other applications or if there are no other applications they can be powered down to save energy.

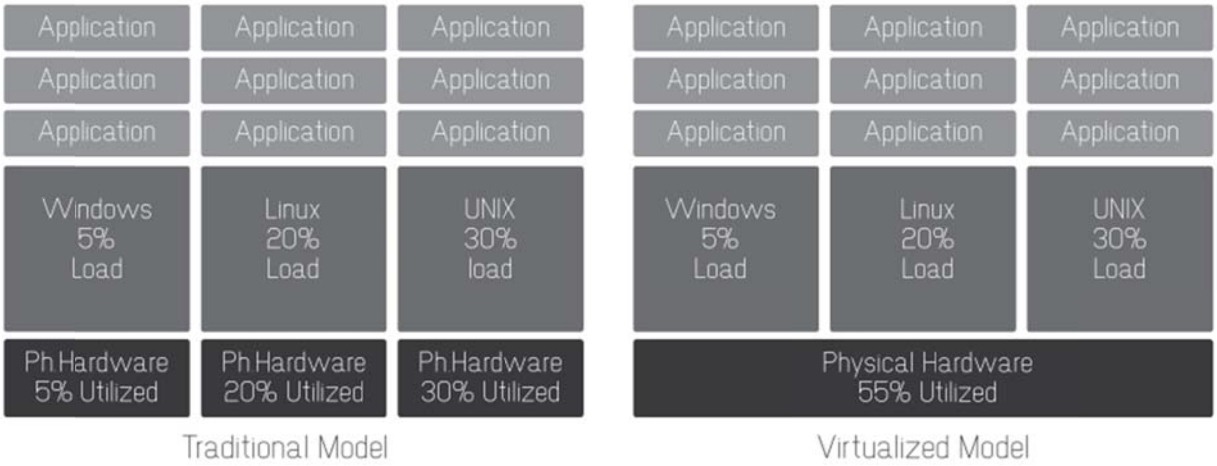

Figure 2-10: **A non-virtualized model (Traditional Model) and Virtualized Model, without regard to type of server virtualization**

In a data center, such as TeliaSonera's DDCs, VMs are running on identical racks of server hardware as well as on a homogenous physical network. Today isolation is mainly done through VLANs, with VLAN tags unique to a specific customer. This has become a problem due to the VLAN-tag field only being 12 bits, as this limits a data center to a maximum of 4096 unique VLANs. Amazon has solved this problem by means of VLAN stacking (as described earlier in Section 2.4.1).

2.5 Cloud computing service models

There are three different cloud computing service models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Service-as-a-Service (SaaS)]. They are dependent on one another since IaaS is the foundation on which PaaS and in turn, SaaS are built. These different models are described below based upon the description in [11].

IaaS, also referred to as Hardware-as-a-Service (HaaS), is the backbone for every type of cloud environment. It consists of networking equipment, storage devices, and computing resources. In this model the provider owns, maintains, and operates the infrastructure for their customers. The customer pays the cloud provider to run an operating system of their own choice along with their choice of applications on top of the underlying hardware provided by the provider. An IaaS service provider can provide different server virtualization stacks. TeliaSonera is an IaaS-provider.

PaaS refers to a model where the cloud service provider provides a platform complete with one or more tools and different programming languages for their customers to build their own applications. These applications are then delivered to users. Barium and TeliaSonera are both PaaS-providers.

In the SaaS service model the customer of the cloud service provider controls only application specific settings in the user interface. As a result, the cloud service provider must create and run a specific service that is made available to end customers. Facebook is a SaaS provider.

2.6 Cloud Environments

There are different approaches to cloud deployment. In this section, we will describe three of these: public cloud, private cloud, and hybrid cloud.

A private cloud is best suited for large companies with the competence to operate and maintain a data center. These companies often have legacy applications that are incompatible with cloud environments provided by IaaS providers. Additionally, many companies do not want to make certain of their data available to others. Storing data in your own private cloud means that your control over this data is guaranteed and that all the security processes are visible.

A public cloud is the dual of a private cloud. A public cloud deployment is best suited for small companies without the capital or competence to operate and maintain their own data center. By using a public cloud they rent computing and storage from a cloud provider.

A hybrid cloud combines a company operated and maintained private cloud (that internally offers some resources) with an IaaS (that provides some external resources). For example, storage could be spread over the different cloud environments with the location of specific data determined by the required level of secrecy and the availability of sufficient resources. Typical legacy applications are best suited for the private part of the cloud. Moreover, using cloud computing resources enables data and processing to effectively share and transfer utilization of different resources amongst the different cloud entities, thus allowing a company to avoid sharing critical business applications and sensitive data with a third-party – while gaining the advantages of cloud computing for other parts of their operations.

Figure 2-11 illustrates the idea of a hybrid cloud. The cloud orchestrator is not necessarily an application with a complete overview of the different cloud entities and resources, but rather a set of rules for data, computing, and application migration between the clouds. For example, if public cloud 1 goes down, certain data can be migrated to public cloud 2 and specific applications can be automatically started on public cloud 2 and the private cloud. Another interesting feature of a hybrid cloud is the heterogeneous nature of the cloud computing environment that it offers, thus allowing a company to have different cloud entities for specialized tasks. For example, public cloud 1 could be optimized for performance, while the private cloud is optimized for security and authentication for certain applications. A solution with an application running on the cloud public 1 could retrieve customer authentication service, database access tokens, etc. from the private cloud. Also cloud environments can extend the reach of a company by providing increased local bandwidth and reduced delay for users in a given geographical region and can provide geographical diversity (avoiding all of the company's records being lost or inaccessible due to a network partition, flood, fire, etc.). Although there are some disadvantages to heterogeneity, such as increased complexity of configuration when integrating multiple cloud environments, especially when combining different IaaS's and SaaSs.[12]

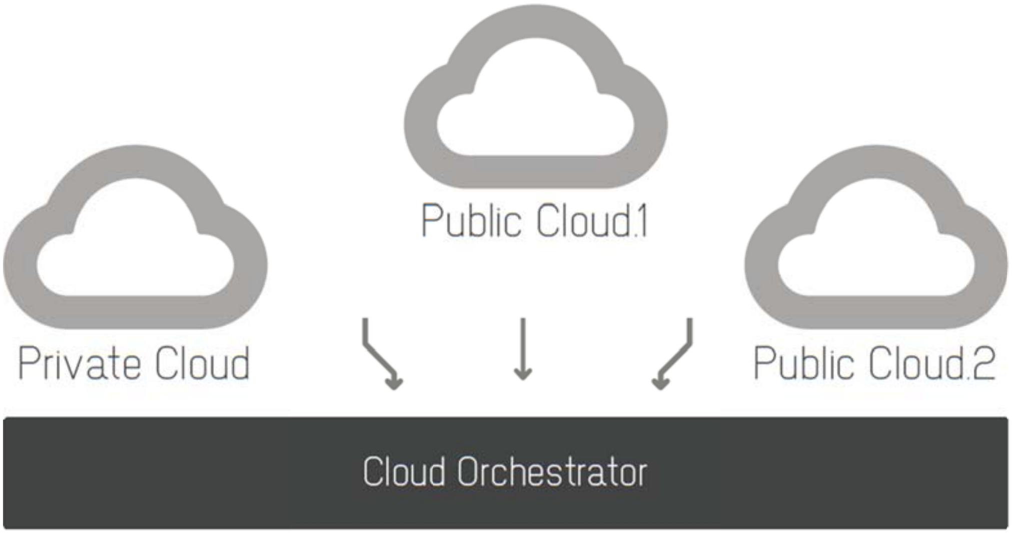

Figure 2-11: An overview of the hybrid cloud definition

A problem with hybrid cloud deployment is the inevitable diversity of the underlying hardware infrastructure and software. Getting all of these components to interact with one another *without* difficulties depends on a lot of initial manual configuration. The goal of hybrid cloud computing is to mesh the multiple cloud environments together, *despite* their differences – this requires cloud orchestration. It is important to point out that this heterogeneity is also the strength of a hybrid cloud deployment strategy.[13]

2.7 Related work

In this chapter we present work that has been conducted in the field of interest. Since the area is quite new there have not been any larger studies with regards to funding and overall depth available to the public, most work is being done behind closed doors at the large network companies, such as Cisco.

We have chosen to include the two following studies, the major related work done by Mr. Shin and Gu we found very enlightening and correlated to some bits with our own study, thus being relevant to include. Also the minor work where vulnerabilities within OpenFlow is being discussed.

2.7.1 Major related work

In 2013, Seungwon Shin and Guofei Gu showed that SDN introduces new security issues, mainly resource consumption attacks - specifically DoS-attacks aimed at the control plane (SDN controller) and the data plane. They set up a test environment consisting of one *OpenFlow* switch, an SDN controller, and two hosts who communicate with each other. The maximum number of flow rules was set to 1500 for the switch. The first test scenario sent packets at a rate of 50 packets per second (pps) and each packet was crafted to generate a new flow table entry. This is the minimum rate for which it was possible to flood an OpenFlow switch and SDN controller since the default timeout for a flow rule is 30 seconds, hence 50 pps for 30 seconds is 1500 packets. At 600 pps it takes only ~3 seconds(!) to flood the switch and overwhelm the SDN controller.

2.7.2 Minor related work

Varun Tiwari, Rushit Parekh, Vishal Patel in their “A Survey on Vulnerabilities of OpenFlow Network and its Impact on SDN/Openflow Controller” [19] investigated vulnerabilities of an OpenFlow network, specifically how an SDN controller could be interfered with. They briefly elaborate that the used of the Transport Layer Security (TLS) protocol, as used in OpenFlow networks, does not *per se* mean that the communication is secure. They also note that resource consumption attacks (caused by generating illegitimate traffic flows) directed at an SDN controller can cause problems (as described in Section 2.7.1). Their conclusions are sparse, but they stress that with SDN being a new field there certainly is room for improvement.

3 Methodology

The purpose of this chapter is to provide an overview of the research process used in this thesis, in addition to the methodology we have applied to the conducted work, as was described in Section 1.5.

3.1 Research Process

The research process started with us discussing possible topics of interest with the networks department of TeliaSonera. Looking ahead they saw great value in investigating SDN and the overall virtualization of their infrastructure, but were unclear about possible security issues associated with lower level virtualization. This resulted in TeliaSonera wanting us to provide them with a pilot study on SDN, specifically how security issues related to SDN could affect Teliasonera's data center infrastructure.

After agreeing upon SDN as the topic for this thesis, we started our literature study to gain an overview of the area. This literature study lasted for a couple of weeks.

The focus then shifted towards narrowing the subject of our research, this unfortunately took longer than we first anticipated. Finally with the subject and research question in place we started discussions with TeliaSonera based on our general view of SDN (obtained via our literature study).

We also conducted a series of simulations benchmarking an OpenFlow controller, POX – a single-threaded controller. We had two perspectives when benchmarking, the goal was to find out what is the biggest differentiator in terms of affecting overall system performance in terms of flows per second (throughput). The differentiators being the number of switches and unique MAC-addresses (hosts or VMs) within the network. We conducted our testing as two sets with three repetitions with the relevant input data in order to compare the impact of varying the number of switches and hosts respectively. First, we tested switches as a differentiator, with the number of switches being 10, 100, and 1000 during the three repetitions, we later added the case of 500 switches – in order to get more data. When testing hosts as a differentiator, we considered 10 000, 100 000 and 1 000 000 hosts per switch.

Because our research process was solely based on the qualitative research methodology approach, as we described in Section 1.5, the work process proceeded as follows:

1. **Identifying a question** - How well does SDN co-operate with TeliaSonera? And future security demands in a rapidly shifting industry?
2. **Review literature** - What has been done in the field of interest? Similar studies?
3. **Purpose of the study/question** - The main purpose of this study is to give TeliaSonera insight in lower level network virtualization in terms of security.
4. **Define the problem area** - What concepts and terms are today associated with SDN and related security issues?
5. **Target group** - TeliaSonera and employees.
6. **Road map** - Specifies who will participate in this thesis project, and also how, when, and where insight and content will be gathered and produced during this project.
7. **Analyze literature** - Analyzing the content we have gathered along the way and sum it up in text.

3.2 Simulation environment

For this experiment we the following software was used and run on an Ubuntu 14.04.1 operating system:

- VirtualBox
- Cbench
- Mininet
- POX

VirtualBox was used for the virtualization. We set up a VM, running Mininet on top of an Ubuntu server. Cbench is a program for benchmarking OpenFlow controllers. The program works by generating packet-in messages. Cbench also has the capability to emulate switches and hosts connected to a controller. The number of switches and hosts are specified as argument when running Cbench. The output from Cbench could be used to compare the performance of different controllers. The controller that was used in this experiment was POX, a single-threaded controller implemented in the Python programming language. Mininet was used for emulating a network consisting of virtual hosts, switches, and controllers.

We accessed the Mininet VM that we created through VirtualBox via SSH. In one terminal window we started the POX controller and in another window we ran Cbench altering the number of switches and hosts in order to benchmark different scenarios.

3.3 Evaluation of the work process

When evaluating how well the work process actually worked, we realized that overall it should have been conducted better. We did not clearly define points 3 and 4 (as described in Section 3.1) early enough in time. This led us to do a lot of unnecessary work during this thesis project. This was a result of having too wide a scope at the beginning of the project, with no clear purpose or problem area. The overall focus on SDN and the security issues it brings was clear from the start, but aside from that, the potential problem areas proved hard to define.

4 Analysis

In this chapter we will analyze resource consumption attacks, how they affect TeliaSonera's traditional network today, and how this would differ if an SDN environment was implemented.

We will also investigate how TeliaSonera could work towards increasing the level of confidence and trust in multi-tenancy solutions with a future implementation of SDN within their data centers. In this particular case study, we have discussed the topic with our tutors and TeliaSonera employees (mainly network architects). The analysis of this issue is based on our own academic thoughts gathered from the literature studies and feedback & support from our discussions with TeliaSonera.

4.1 Resource consumption issues

We started with a case study of DoS-attacks on TeliaSonera's data centers. During meetings with the professionals in charge of load-balancing within TeliaSonera we realized that the case study regarding DoS-attacks under normal conditions versus when there is a specific targeted service would **not** affect the network traffic in a way that it would be visible *within* the data center. This occurs because in the case of a DoS-attack, most of the malicious traffic would be stopped *outside* of the data center by the firewalls at the edge of the data center once a DoS-attack has been detected. Even after mitigation of the DoS-attack outside of the data center, there is still a risk that some of the malicious traffic might leak into the data center. However, in the current architecture there is no protection against DoS-attacks at the cloud service provider's level. Therefore, it would not matter whether the network architecture is virtualized using SDN or not, as the amount of damage done to the servers would be the same when using the same underlying infrastructure. Note that there is some increased risk when deploying SDN that the individual servers will be more heavily utilized (as illustrated in Figure 4-1), thus additional load due to an attack on one of these services could exceed the remaining resources of the underlying server. Additionally, there is a bottleneck effect to take into consideration when the hardware and software networking devices fail due to enormous amounts of traffic, thus bringing down one or more of these devices.

This realization was confirmed when discussing the issue with sales representatives from the company F5 Networks. They provided us with a non-TeliaSonera view of the issue that helped us to look at it from another point of view. This also made us more aware of the hollowness of the original problem statement. We contacted our tutor, Daniel Alfredsson an IT- and networks architect, for his input regarding the issue. He confirmed what we suspected; there would be no major difference in terms of security regarding DoS-attacks on TeliaSonera's data centers. Due to this, we felt we had reached a dead end for further analysis of this particular issue as the focus of our thesis project was on TeliaSonera's data centers. A further analysis of this issue would be out of our scope, since DoS-attacks are primarily *mitigated* outside of the data center's network. However, one should note that analysis of the traffic *within* the data center could be used to improve the *detection* of DoS attacks.

Figure 4-1 illustrates how illegitimate traffic from DDoS-attacks from the Internet are mitigated and dropped in TeliaSonera's networks today. *TeliaSonera Internal Network (*TSIN) is the network within the data center (DC) site. TeliaNet is TeliaSonera's access point to the internet for customers as well as internal traffic. A DoS-protection system provided by Arbor Networks is deployed within TeliaNet. This DoS-protection system works by collecting statistics and checking whether the traffic matches certain patterns, given this information the system determines if certain traffic flows are legitimate or illegitimate (i.e., coming from a DoS-attack). When an illegitimate traffic flow is detected, the DoS-protection system reroutes these illegitimate packets and drops them off elsewhere. Deploying the DoS-protection *outside* of the data center minimizes the probability of links to the data center becoming clogged with illegitimate traffic. This placement of this solution outside of the data center means that it is outside the scope of our thesis project, since our focus was on security issues of the traffic *within* the data center space.

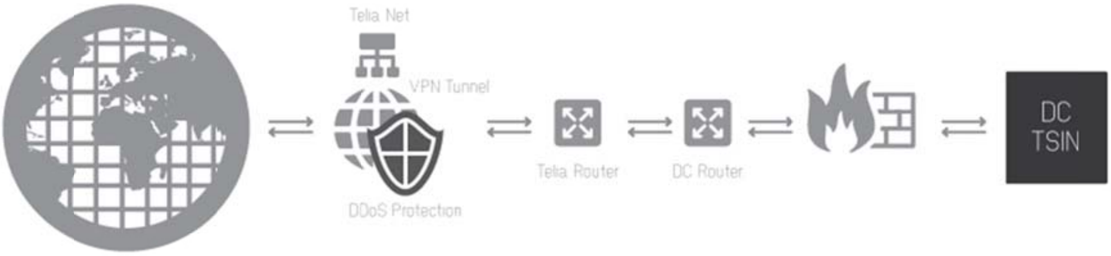

Figure 4-1: How a DDoS attack from the Internet is mitigated by the network outside of the data center (DC)

However, consider DoS-attacks from another perspective – specifically how could one potentially circumvent the security of a SDN network within the data center. As described in Section 2.2 a DoS-attack aims to consume all resources provided by the targeted service in order to lead to a breakdown of this service and/or deny legitimate users access to this service. Our academic supervisor pointed out that in an SDN environment based on *OpenFlow*, new types of attacking are possible because the control plane is separated from the data plane and moved to a centralized point, i.e., the SDN controller, thus there must be communication between these two planes (via the *OpenFlow* protocol). For example, this communication occurs when the data plane of a switch receives a frame that is not recognized, then the switch has to contact the SDN controller. The controller then makes a decision about what to do with this frame, either drop the frame or add a new flow entry to the flow table in the switch so that the next time a similar frame arrives the switch would know how to forward it. By exploiting this function of *OpenFlow* there is potential for DoS-attacks that target the SDN controller (much as the attack described in Section 2.7.1 did) and to target the data plane's bandwidth to and from this controller. The procedure of making flow requests (also packet-ins) and modifications consumes resources of the control plane. Thus after a certain number of requests within a short period of time the SDN controller would be flooded and hence would have difficulties to handle all of these requests. The data plane would also be affected as all previous flow requests sent to the controller and processed have produced new flow rules and actions (flow modifications and packet-outs) leading to an increased consumption of the resources of the data plane in both the network and in the switch itself. Placing all of these bogus flow entries into the flow table could also hinder normal traffic flows, especially if the attacker can generate enough new flow table entries to force entries for valid ongoing traffic to be flushed out of the flow table.

Figure 4-2 illustrates an example of the process of consuming all resources (hence mounting a DoS-attack) of an OpenFlow enabled switch by sending appropriately formed frames. In this case, we assume that the flow table can store up to 10 000 different flow rules. Each packet sent from the host (10.0.0.1) to the destination IP address (20.0.0.1) had a different destination port in order to make the switch ask the controller for a new flow rule for each frame. This could lead to an excessive number of packet-in messages being sent by the switch, and an excessive number of flow-modification messages being sent back by the controller, causing flooding of both the control- and data planes. All of this depends upon whether new flow entries will have to be created with the only difference being the destination port. However, it is often more complicated than this. A combination of altered match fields might be needed for a packet-in message to be generated. In our simulation we will show the effects on the controller of frames coming from multiple unique MAC-addresses.

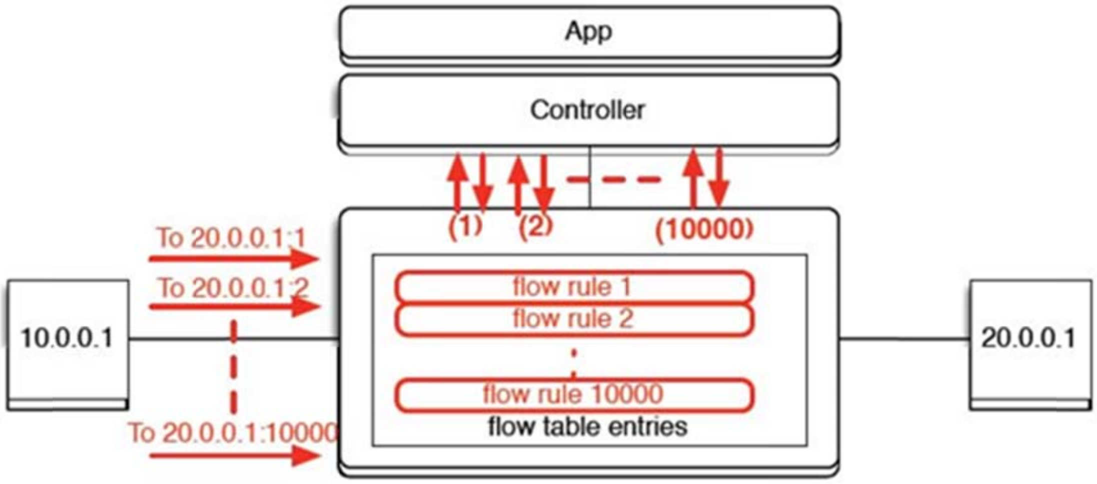

Figure 4-2: Performing a DoS-attack by taking advantage of vulnerabilities in OpenFlow (adapted from slide 11 of [20] and slide 16 of [21]).

4.2 Multi-tenancy issues

For TeliaSonera an SDN implementation within a data center would be done by initially creating a small deployment for selected customers to participate in. This would serve as a use case for further and broader implementation of SDN within TeliaSonera's data centers worldwide. It is important to stress the fact that the main issue is not the technological aspects of multi-tenancy. Today multi-tenancy is achieved by logical isolation through VLANs, VLAN stacking (or QinQ), or VXLANs[22]. These approaches divide the physical networks, applications, and database instances into logical segments for each customer. The problem we investigate is not a technological issue, but rather a psychological issue - how to increase the customers' level of trust and help them gain confidence in these solutions. Furthermore, these solutions are becoming more and more virtualized, thus multiple customers share the underlying hardware. History has shown that new technologies often are a target for skepticism, as was server virtualization ~10-15 years ago. Today network virtualization is going through the same cycle. Together with our tutors at TeliaSonera, we felt it would be of interest to consider this psychological problem because of the subject we had chosen, as it affects the deployment and adoption of SDN and could potentially impede the overall drive towards network virtualization and adoption of cloud environments.

The most important points we learned from our discussions with TeliaSonera concerned:

- SLAs
- Automatization, and
- Redundancy both path- and data wise

Of those three, we think the **SLA** is the most important point, because an SLA defines exactly what the customer expects from the service provider. An SLA codifies the customer's requirements for computing and storage capacity as well as the required security and overall the experience the customer is paying for. The SLA could for example include a specification of the minimum uptime, scalability, availability, etc. Finding appropriate metrics to rate and compare the statistical measurements of a service is important. Unfortunately, unlike metrics for computing and storage capacity, today there are no comparable metrics for availability and uptime. Currently the cloud providers themselves collect their own data - which is not necessarily incorrect, but is definitely biased. This leaves customers wondering if they are getting the services that they are paying for according to their SLA.

Automatization concerns the constant striving by humanity to automate manual processes (regardless of the industrial sector) in order to ease humans' work and to eliminate (or reduce) the possibilities and impact of human error. Today a single mistake in a configuration file could end up destroying a whole process within a data center, thus interrupting availability and possibly opening the data center up for a variety of security breaches. Automatization (hence less "hands-on" control) is desirable because TeliaSonera and most other IaaS-providers wish to gain the benefits of this automatization. TeliaSonera's current cloud service is built on an underlying traditional infrastructure, with virtualization mainly within the server space combined with isolation via VLANs. Because of this underlying infrastructure, certain steps require manual configuration when setting up a service for a customer. Additionally, manual maintenance and updates of hardware and software could end up causing trouble; therefore, the entire industry seeks to minimize the amount of "hands-on" operations required.

This bringing us to the need for **redundancy** of all of the different types of resources, as a customer of a cloud service wants constant accessibility to their data and services. However, this is counter to the effort by cloud providers to adopt a lean production approach to preserve value while minimizing work (and costs). Ensuring availability of cloud services requires both automatization and scheduled maintenance & updates of software during times when the data center and these services are lightly loaded. Logging and sniffing traffic flows is important due to the fact that cloud environments have allowed providers to over-subscribe their services (i.e., they have accepted more subscriptions than they have simultaneous resources for), which would not have been possible with a single-tenant deployment.

Another interesting topic arose during discussions with TeliaSonera, but was left out of the previous list of points. This topic concerns what to do when a customer wants to move from one cloud provider to another. With cloud computing being a relatively new concept there are currently no standardized way of doing a migration of functionality, e.g., an application. Migration of storage on the other hand is easier, although there are no cloud provided tools for such data migration today. This issue was investigated by Vytautas Zapolskas in his thesis on "Securing Cloud Storage Service"[23], where he discusses possibilities for data migration between different cloud providers. Full storage migration is best done today by using personal storage or a FTP-server to act as an intermediary during transmission between different cloud environments. Unfortunately, because of different underlying software and supporting libraries, certain applications are not suitable for every cloud environment. This also raises the question of whether after a successful migration; can a customer ensure that their data is properly deleted from the earlier cloud provider's infrastructure? This typically would be solved via an agreement where questions and certain grey zones are specified. However, a more secure approach is to keep the data only in encrypted format – then change the keys when the data is migrated.

Regarding deletion of data, another highly psychological issue is that of who is responsible for the data placement in a public cloud. On one hand, this issue could be addressed by a well-defined SLA, but customers could also encrypt their data when storing it in the cloud. Although storing documents of high secrecy is probably unsuitable for a public cloud service today, it is however clearly a goal of many storage providers to do so.

TeliaSonera has the opinion that a certain number of their network services fit the cloud well, while some services are better suited for a more traditional, single-instance approach. Additionally, there are customers who demand very high security solutions, such as defense industries, government agencies, etc. From the customer's point of view there is the possibility to adopt a hybrid or private cloud deployment to ensure that certain data is not shared with a third-party – but this requires retaining appropriate resources and competence in-house (and both come at a high price).

4.3 Benchmarking OpenFlow Controller

During our testing we looked into how the number of switches as well as hosts is affecting system overall performance in regards to flows per second (fps). Testing using both switches as well as unique MACs (hosts) as the differentiators, the analysis are displayed as well described in text in the following subsections.

4.3.1 Switches being the differentiator

Looking into our first test, which was with switches being the differentiator. Figure 4-3 shows the responses per second occurring as we increased the number of switches as 10^n, for n=1 to 3. For each repetition, the number of hosts was a static 1 000 when benchmarking switches. The higher the rps value the better the performance of the controller. The results are displayed in Figure 4-3 in terms of thee *average responses per second (rps)*.

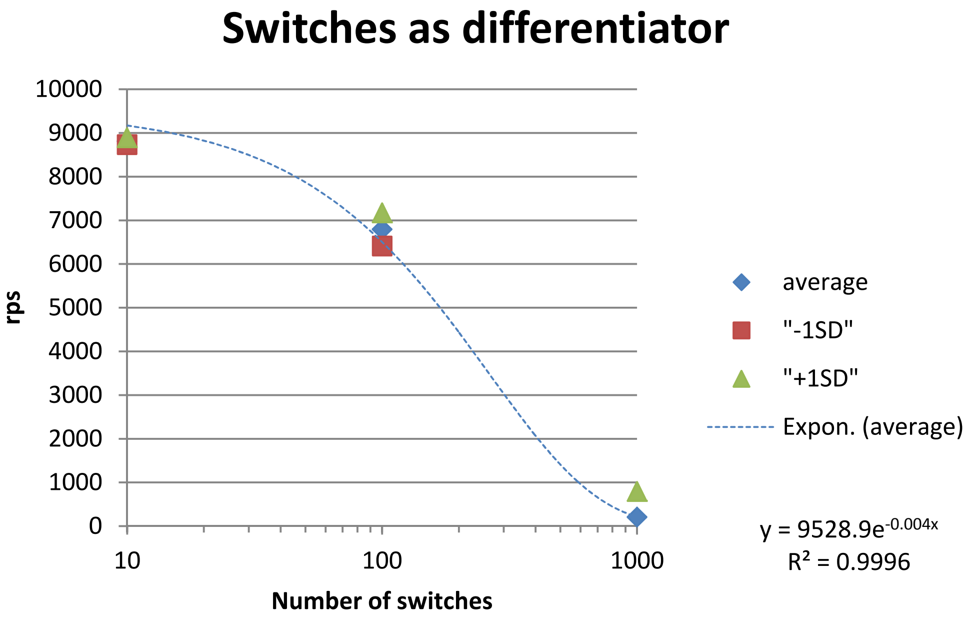

Figure 4-3: ***Average responses per second*** **for 10, 100, and 1 000 switches with a static 1 000 hosts**

We quickly noticed the significant drop in rps in the range from 100 to 1 000 switches. We suspected that in the range from 100 to 1 000 switches some switches were probably not able to transmit packet-ins to the controller, due to an overload of the controller. In the interval in which the drop was largest (i.e., between 100 and 1 000), we introduced a fourth repetition with 500 switches. As shown in Figure 4-4 when benchmarking using 500 switches, this configuration had a better average rps than when using 1 000 switches, the controller could also not handle the packet-ins even for 500 switches. As future work it would be interesting to introduce intermediate numbers of switches between 100 and 500 to see just where the behavior changes.

The benchmarking results, showing the exact numbers and min/max values for responses per second are shown below with terminal input and output for the four repetitions of the test when varying the number of switches:

```
cbench –c localhost –p 6633 –m 10000 –l 10 –s 10 –M 1000 –t
RESULT: 10 switches 10 tests min/max/avg/stdev =
8652.12/8882.30/8813.28/80.07 responses/s
```

```
cbench –c localhost –p 6633 –m 10000 –l 10 –s 100 –M 1000 –t
RESULT: 100 switches 10 tests min/max/avg/stdev =
5909.37/7118.32/6793.67/378.02 responses/s
```

```
cbench –c localhost –p 6633 –m 10000 –l 10 –s 500 –M 1000 –t
RESULT: 500 switches 10 tests min/max/avg/stdev =
0.00/3623.29/1229.18/1373.87 responses/s
```

```
cbench –c localhost –p 6633 –m 10000 –l 10 –s 1000 –M 1000 –t
RESULT: 1000 switches 10 tests min/max/avg/stdev =
0.00/1869.84/207.76/587.63 responses/s
```

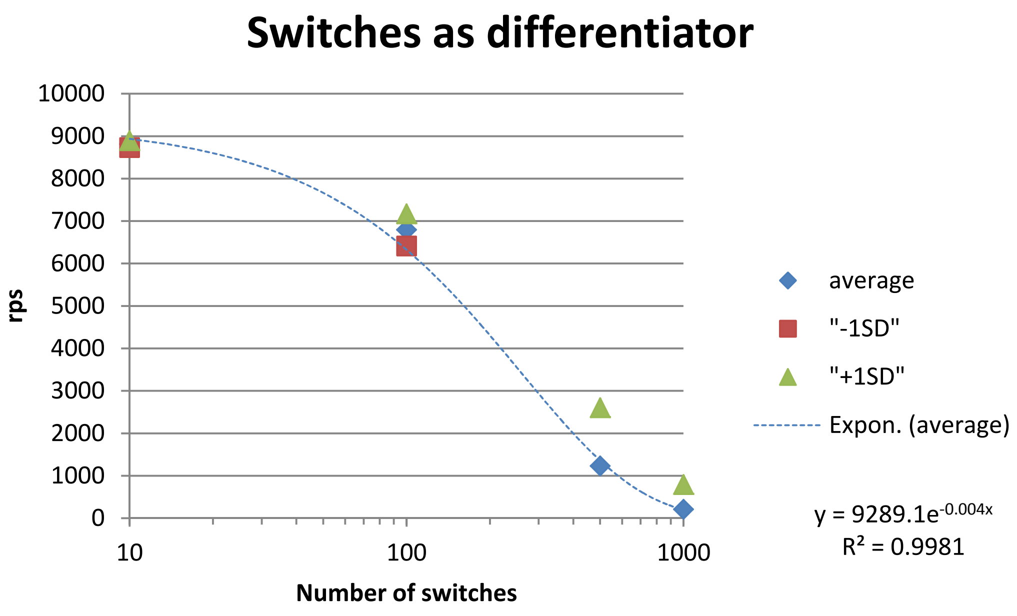

Figure 4-4: Switches being the differentiator showing the results for 10, 100, 500, and 1 000.

4.3.2 MAC-addresses (hosts) being the differentiator

Unlike the results of the first test with increasing numbers of switches, the results of the second test showed the decrease in performance varied linearly with $\log_{10}$ of the number of hosts, as measured with 10^n hosts, for n=1 to 6. The switches and controller were able to handle all the packet-ins sent at a better rate, although the overall system performance is decreasing it is not decreasing significantly. The test results are displayed in Figure 4-5 in terms of the *average responses per second*.

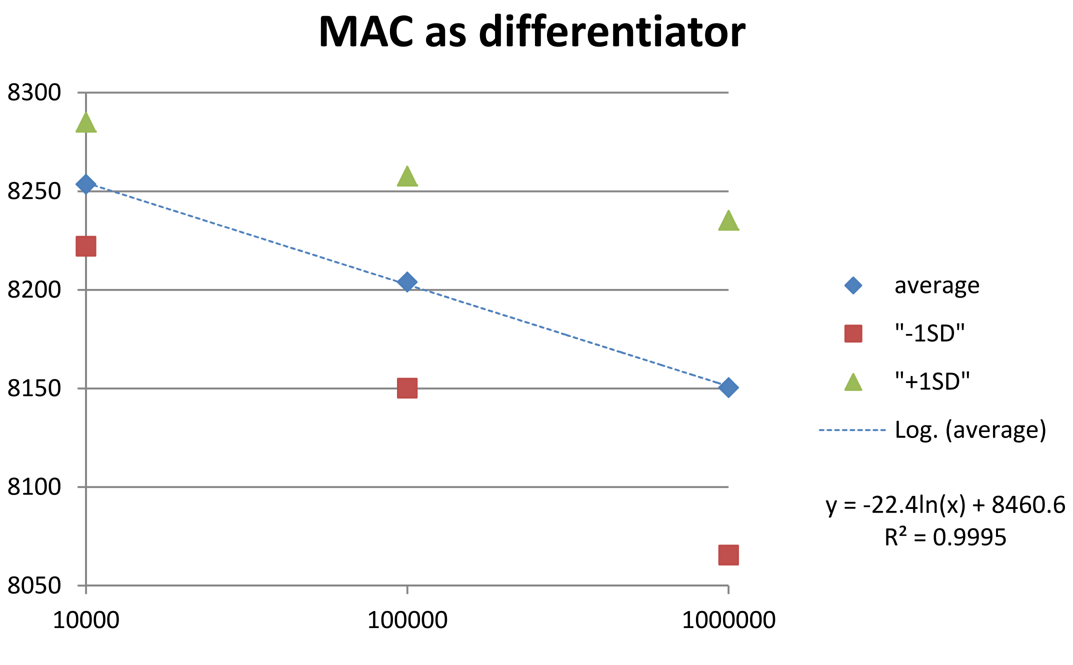

Figure 4-5: Hosts being the differentiator showing the results for the number of hosts being; 10 000, 100 000, and 1 000 000 on a set of 10 switches

Should we have done the same for 10 000 000 and more hosts the linear decrease is expected to continue if all parameters are the same. If more switches are added in combination with increasing number of hosts, performance is likely to drop even quicker as shown in the previous section regarding switches as the differentiator.

The terminal input and output was:

```
cbench –c localhost –p 6633 –m 10000 –l 10 –s 10 –M 10000 –t
RESULT: 10 switches 10 tests min/max/avg/stdev =
8176.24/8280.35/8253.60/31.46 responses/s
cbench –c localhost –p 6633 –m 10000 –l 10 –s 10 –M 100000 –t
RESULT: 100 switches 10 tests min/max/avg/stdev =
8109.62/8286.99/8203.98/53.77 responses/s
cbench –c localhost –p 6633 –m 10000 –l 10 –s 10 –M 1000000 –t
RESULT: 500 switches 10 tests min/max/avg/stdev =
7987.79/8251.91/8150.44/84.87 responses/s
```

5 Conclusions and Future work

In this chapter, we will introduce the conclusions we have come to. We also present what hindered us from reaching to definite conclusions in some cases and how this affected us. What we deemed out of our scoped we summarized in Section 5.3. Finally, we have reflected on our work and what it (SDN) could mean, socially, ethically, economically, and environmentally over time.

5.1 Conclusions

In this section, we present our conclusions on the work we have done over the course of this thesis project. The conclusions are gathered from Chapter 4, where we analyzed and presented data and ideas regarding the problem definition we described in Section 1.2.

5.1.1 New vulnerabilities in network security arise with SDN

With the introduction of a centralized SDN controller for the network, there are several possibilities to interfere with this controller, both internally and externally. An example of internal damage is the possibility of a configuration error that spreads throughout the network due to the centralized controller. Important to note is that this is also a problem with today's method of step-by-step (or device-by-device) configuration, but the extent of possible damage is wider with SDN because one single modification in the controller can affect multiple devices. In Section 5.2 and in Appendix A we further describe problems related to human error, specifically how the effects of such errors could affect overall trust of customers in SDN solutions.

The idea of attacking the SDN controller by flooding the control- and data plane with unique and new flow modification messages (i.e. introducing a large number of new flow rules) could be mitigated by adopting an SDN policy that overrides the decision to create new packet-in messages by replacing them with wider match fields. Widening the match fields will results in fewer packet-in messages being created by the data plane, while still forwarding traffic. However, these wider rules may lead to sub-optimal forwarding decisions.

5.1.2 Gaining trust in SDN solutions

As stated earlier, since trust is a rather psychological issue it is hard to say that we have achieved our goals (as stated in Section 1.2). However, it is possible to state some ideas of how to solve the problems faced, i.e. how to establish trust amongst TeliaSonera's broad (potential) customer base.

Standardization is of great importance for all the points regarding SLAs and automatization in the future, as mentioned in Section 2.6. Standards that are widely used throughout the industry and standards with metrics for providers to follow will enable both cloud providers and customers to compare their expected performance with actually delivered performance. These metrics will make it easier for customers to compare cloud providers and to make their decisions.

As of today, the metrics regarding computing and storage resources are easy to understand, i.e., the capacity of the CPU and the amount of available storage space. However, there are no standard metrics for comparison of accessibility and uptime for cloud environments, other than the service provider's own data, which is not always that informative or unbiased.

5.1.3 Benchmarking of POX controller

The testing we conducted lead us to the conclusion that it is possible to consume all resources of an OpenFlow controller which in turn denies or slows down access time for legitimate users. As our testing showed, a relatively small number of switches is needed to affect a single-threaded controller, it also showed for a linear decrease of performance when increasing the numbers of hosts each with unique MAC-addresses.

If someone is about to attack an SDN controller it is possible by altering specific match fields in this case the MAC-address when a switch encounters an incoming packet with an unknown source MAC-address for which there are no flow rules set in the flow table, then a look-up will take place, a packet-in message will be sent to the controller. By repeating this using a lot of different hosts with unique MAC-addresses it is possible to exhaust the controller. In a real-life scenario an attacker could mount a DDoS attack by infecting multiple computers and altogether directing a massive load of traffic towards the target.

5.2 Limitations

The psychological aspect of gaining trust is hard to anticipate in a customer segment without performing large scale surveys of these customers. We did not conduct a larger scale survey, instead we arranged meetings via TeliaSonera with representatives of several large networking companies (F5 Networks, Cisco Systems, and VMware) to gain qualitative insight. We did also participated in a smaller networking conference at Stockholm Water Front where we had the chance to speak to representatives from Arista Networks and Nuage Networks (Cisco, VMware, and F5 were also represented at this conference). While providing us with great class insight into the problems of lower level virtualization and the related trust issues, we did not gain a broad opinion and or deeper insight into the matter, but rather found a very industry specific vision. This is a limitation when trying to show results concerning the trust issues related to lower level virtualization and SDN combined with multi-tenancy.

Since SDN is a relatively new technology, the chances of getting to experiment with it were limited during the time of the project. In addition, the total duration for the thesis caused us consider right from the start whether to try to set up an environment for experimenting or not. We decided not to configure such an experimental environment in order to conduct an experiment due to the limited time frame and what we initially perceived as the lack of the necessary network equipment needed for such experiments.

5.3 Future work

In Section 5.1 we mentioned that in TeliaSonera the most commonly occurring DoS-attacks are mitigated outside of the data center. Hence, another student could possibly investigate this topic in the future. An issue that could be evaluated is the coupling of information from within the data center and the data available in the external DoS mitigation tool. Future work could include a practical demonstration of a DoS-attack on an SDN environment (such as might be deployed *within* a data center) and then examine methods for detecting and mitigating such an attack.

A broader survey to gain overall insight into the trust issues related to SDN in a multi-tenancy environment should be further explored.

5.4 Reflections

We have encountered different aspects and reflections during our thesis project. In the numerous discussions with TeliaSonera and overall the business side of the project we have discussed economics and possibilities of lowering costs, in accordance with the aim of this thesis project. With a future implementation of SDN in TeliaSonera's data centers, TeliaSonera's representatives have high hopes of lowering operating and capital expenditures. This would lead to standardizing network equipment to a higher degree than is possible today, easier configuration/management, and further exploitation of pooling (in order to make better use of available resources). What is important to remember is that these desires are the source of the overall drive for SDN within the industry – all in the interest of lowering capital and operating costs.

Ethical aspects of our thesis project are primarily linked to the trust issues regarding the multi-tenancy issues that we have investigated. Throughout this thesis, we have taken this aspect into consideration and placed a strong emphasis on it since we feel that this is important for net neutrality and the preservation of user integrity online.

Social and environmental aspects of our work are more difficult to point out. However, gains on the environmental side are quite obvious since SDN allows increased virtualization, leading to substantial gains since hardware will be more standardized and networking devices can be virtualized – both potentially increasing the useful lifetime of this equipment (further minimizing the amount of electronic waste created in a period of time). Additionally, as noted earlier in the thesis virtualization enables multiple VMs to be run on the same hardware, thus allowing other hardware to be powered off – thus saving electrical energy and reducing cooling requirements. The resulting higher utilization of the underlying hardware, while meeting the customer's needs, also minimizes the amount of hardware that is actually needed – providing both economic and environmental benefits.

Socially there is a possibility for downsizing the amount of personnel working hands-on with data centers, thus lowering the operating expenditures we mentioned earlier and enabling people to be deployed elsewhere in the organization – perhaps in more socially (and economically) relevant ways.

References

[1] Dan Pitt, ‘Trust in the cloud: the role of SDN’, *Netw. Secur.*, vol. 2013, no. 3, pp. 5–6, Mar. 2013. DOI: 10.1016/S1353-4858(13)70039-4

[2] Denys Knertser and Victor Tsarinenko, ‘Network Device Discovery’, Master’s thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, Kista, Stockholm, Sweden, 2013 [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-123509

[3] Codenomicon, ‘Heartbleed Bug’, 29-Apr-2014. [Online]. Available: http://heartbleed.com/. [Accessed: 01-Oct-2014]

[4] Linh Vu Hong, ‘DNS Traffic Analysis for Network-based Malware Detection’, Master’s thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, Kista, Stockholm, Sweden, 2012 [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-93842

[5] Emmanouil Karamanos, ‘Investigation of home router security’, Masters Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2010 [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-91107

[6] Eric Lai, ‘Multitenancy & Cloud Computing Platforms: Four Big Problems’, *Übertech*, 15-Feb-2012. [Online]. Available: http://www.zdnet.com/blog/sap/multitenancy-and-cloud-computing-platforms-four-big-problems/2559. [Accessed: 01-Oct-2014]

[7] Stacey Higginbotham, ‘Software-defined networking forces Juniper’s big shift — Tech News and Analysis’, *GIGACOM*, 15-Jan-2013. [Online]. Available: https://gigaom.com/2013/01/15/software-defined-networking-forces-junipers-big-shift/. [Accessed: 01-Oct-2014]

[8] H. Gredler and W. Goralski, *The complete IS-IS routing protocol*. London: Springer, 2005.

[9] Cisco Systems, Inc., ‘FabricPath Switching [Cisco Nexus 7000 Series Switches]’, *FabricPath Switching*, 25-Oct-2010. [Online]. Available: http://www.cisco.com/en/US/docs/switches/datacenter/sw/5_x/nx-os/fabricpath/configuration/guide/fp_switching.html. [Accessed: 01-Oct-2014]

[10] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, ‘Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks’, *Internet Req. Comments*, vol. RFC 7348 (Informational), Aug. 2014 [Online]. Available: http://www.rfc-editor.org/rfc/rfc7348.txt

[11] Cloud Computing Competence Center for Security, Fraunhofer Research Institution for Applied and Integrated Security (AISEC), ‘What are Service Models in Cloud Computing?’. [Online]. Available: http://www.cloud-competence-center.com/understanding/cloud-computing-service-models/. [Accessed: 01-Oct-2014]

[12] Eze Castle Integration, ‘Public and Private Clouds Explained’, *Public, Private & Hybrid Clouds*, 2013. [Online]. Available: http://www.eci.com/cloudforum/private-cloud-explained.html. [Accessed: 01-Oct-2014]

[13] Margaret Rouse, ‘What is hybrid cloud? - Definition from WhatIs.com’, 2013. [Online]. Available: http://searchcloudcomputing.techtarget.com/definition/hybrid-cloud. [Accessed: 01-Oct-2014]

[14] Cisco Systems, Inc., ‘Software-Defined Networking: Why We Like It and How We Are Building On It’, Cisco Systems, Inc., White Paper, 2013.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, ‘OpenFlow: Enabling Innovation in Campus Networks’, *SIGCOMM Comput Commun Rev*, vol. 38, no. 2, pp. 69–74, Mar. 2008. DOI: 10.1145/1355734.1355746

[16] Open Networking Foundation, *OpenFlow Switch Specification*, Version 1.4.0 (Wire Protocol 0x05). Open Networking Foundation, 2013 [Online]. Available:

https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

[17] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*, 5th ed. Boston: Addison-Wesley, 2010.

[18] S. Shin and G. Gu, 'Attacking Software-defined Networks: A First Feasibility Study', in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, New York, NY, USA, New York, NY, USA: ACM, 2013, pp. 165–166 [Online]. DOI: 10.1145/2491185.2491220

[19] Varun Tiwari, Rushit Parekh, and Vishal Patel, 'A Survey on Vulnerabilities of Openflow Network and its Impact on SDN/Openflow Controller', *World Acad. J. Eng. Sci.*, vol. 1, no. 1005, pp. 1005–1 to 1005–5, 2014.

[20] Seungwon Shin, 'Software Defined Networking Security: Security for SDN and Security with SDN', 17-Jun-2013 [Online]. Available: http://www.krnet.or.kr/board/include/download.php?no=1751&db=dprogram&fileno=2

[21] Seungwon Shin, 'Software Defined Networking Security: Security for SDN and Security with SDN', 11-Apr-2014 [Online]. Available: http://www.kics.or.kr/storage/mailing/20140411/140411_095144993.pdf

[22] Keith Barker, 'Difference between QinQ and VLAN stacking - The Cisco Learning Network', *The Cisco Learning Network*, 17-Apr-2010. [Online]. Available: https://learningnetwork.cisco.com/thread/12500#62026. [Accessed: 01-Oct-2014]

[23] Vytautas Zapolskas, 'Securing Cloud Storage Service', Master's thesis, KTH Royal Institute of Technology, School of Information and Communication Technology (ICT), Stockholm, Sweden, 2012 [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2:538638

[24] Sixto Ortiz Jr., 'Processor - Dual Data Centers', *Processor: Products, News & Information Data Center Can Trust*, vol. 26, no. 20, 14-May-2014 [Online]. Available: http://www.processor.com/article/6101/dual-data-centers. [Accessed: 01-Oct-2014]

[25] D. Oran, 'OSI IS-IS Intra-domain Routing Protocol', *Internet Req. Comments*, vol. RFC 1142 (Informational), Feb. 1990 [Online]. Available: http://www.rfc-editor.org/rfc/rfc1142.txt

Appendix A: The network infrastructure of TeliaSonera

In this thesis project, we look specifically at one of TeliaSonera's data centers in Sweden. The data center that we looked at is a Dual Data Center (DDC) located outside of Stockholm. A DDC consists of two data centers at different sites that are interlinked to enhance continuity, redundancy, and overall robustness of operations (e.g. computing and storage) [24]. The data centers feature similar hardware and overall network design. The connection between the data centers and their respective server clusters is usually through split high-speed optical fiber. The two data centers in a DDC are typically located within 60-80 kilometers of each other. This distance limitation is due to the attenuation of the signal within the optical fiber as after 60-80 kilometers repeaters are needed to regenerate the signal. Locating the data centers within 60-80 kilometers of each other avoids the need for a repeater; hence, there is one less network component that might fail.

TeliaSonera's data centers consist of a variety of different hardware managing routing and switching (also routing switches) for edge as well as core duties throughout the data center.

The main difference between a core and an edge switch is that the core switch is built to process and forward traffic as fast as possible. In traditional data center architecture the core switches are often topologically found in the midst of the data traffic, between the gateways out of the data center and gateways onto the server clusters. A core switch might also serve traffic coming to and from a nearby internal firewall. In contrast, edge switches (or routing switches) are placed on the edges of a network, for example as gateways to server clusters or as a gateway out of the network. An edge device forwards traffic to or from the internal network

A routing switch is a network device that combines the switch-functionality of forwarding packets based on layer 2 media access and control (MAC) addresses and the router-functionality of forwarding packets based on layer 3 IP addresses.

TeliaSonera's DDC's use Cisco's FabricPath (described in Section 2.3.4) for communication within the data center. FabricPath is a technology that combines a traditional spanning-tree with a link-state protocol to allow loop-free communication over several paths (hence it is provides a multipath forwarding scheme). Exploiting multiple paths is of great importance for traffic flow, reducing latency, and enhancing the overall reliability of communication within the data center. Using only a spanning-tree protocol would eliminate all but one path to avoid loops. As a result FabricPath enables much of the typical layer 3 functionality (e.g. multipathing routing) at layer 2.

The routing protocol used by TeliaSonera and many other data center owners is Intermediate System to Intermediate System (IS-IS). IS-IS is an Interior Gateway Protocol (IGP) using Dijkstra's algorithm. IS-IS is a link-state routing protocol (similar to Open Shortest Path First (OSPF)), thus every router constructs a graph of the topology of the complete network by sharing connectivity information with neighbors and then successively floods this information to the network. At each router, this information is stored in a link-state database (LSDB). Whenever a link fails, the neighbors will take note of this failure and spread this information to the rest of the network. As a result data is always forwarded in the best way possible, from a topological standpoint, within the network. As IS-IS an IGP protocol, thus it is used within a specific domain, while Border Gateway Protocol (BGP) can be used for routing between different domains.[25]

In contrast, the Routing Information Protocol (RIP) is a distance-vector based routing protocol. This means that each router shares network topology in a vector with numbers of hops to a destination. RIP sends these update either when a change occurs or (nearly) periodically (every ~30 seconds). In contrast, in link-state routing protocols network devices gather about their neighbors and inform the network of *changes* of the network's topology. A major disadvantage of RIP is the limit to 15 hops, which means that the protocol cannot be used in large networks, such as those in TeliaSonera's data centers.

TeliaSonera uses a wide variety of server operating systems (OSs) mainly because of older legacy network services that are still supported. The complete infrastructure is virtualized to some extent, although this virtualization is mainly of the servers. For the newer cloud platform, a majority of the servers are running Red Hat Enterprise (RHEL) and Microsoft's Windows Server OSs. These two OSs were chosen because they have been operating successfully with virtualization for a long time. This has instilled trust and increased the overall confidence in virtualized solutions.

TRITA-ICT-EX-2014:161

www.kth.se