Teemu Takasuo

Software-Defined Networking

Bachelor's Thesis
Information technology

December 2018

Kaakkois-Suomen
ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Teemu Takasuo	Insinööri (AMK)	Joulukuu 2018

Opinnäytetyön nimi

Ohjelmistopohjaiset verkot

45 sivua

Toimeksiantaja

Kymijoen ICT

Ohjaaja

Vesa Kankare

Tiivistelmä

Ohjelmistopohjainen verkottaminen on moderni lähestymistapa tietoverkkoihin tarjoten joustavan ja kustannustehokkaan tavan konfiguroida verkkoja ohjelmiston ja virtualisoinnin avulla. SDN-teknologia yleistyy kuukausittain ja on keskeisessä roolissa koskien tulevaisuuden verkkoja. Tämän opinnäytetyön tavoitteena oli selventää sitä, miten SDN-teknologia toimii ja mitä hyötyjä kyseisessä tekniikassa on verrattuna perinteisiin verkkoarkkitehtuureihin, kuin myös luoda esimerkki ohjelmistopohjaisesta verkkoratkaisusta VMware NSX:n avulla.

Tämä opinnäytetyö tehtiin kehittämistutkimuksena. Työn alkuvaiheet koostuivat SDN-aihealueen opiskelusta ja muistiinpanojen tekemisestä käytännön asennuksia varten, koska aiheesta ei ollut aiempaa tietoa ennen työn aloittamista.

Asennukset suoritettiin käyttäen fyysisiä ja virtuaalisia komponentteja. Fyysisille ESXi-isännille perustettiin vCenter-ympäristö, jonka jälkeen nämä ESXi-isännät liitettiin toisiinsa Ciscon 2960-kytkimen kautta. Myöhemmissä asennusvaiheissa fyysistä tasoa käytettiin harvoin, koska NSX-verkko oli kokonaan virtualisoitu ja reititys virtuaalikoneiden välillä toimi virtuaalisesti ilman, että fyysiset verkkolaitteet osallistuivat prosessiin.

SDN-ympäristö luotiin onnistuneesti erottaen fyysisen ja virtuaalisen tason toisistaan ja mahdollistaen kokonaan virtuaalisen reitityksen. Työlle asetetut tavoitteet saavutettiin onnistuneesti ja Kymijoen ICT sai hyödyllistä informaatiota SDN-teknologian hyödyistä ja käyttötarkoituksista. Tämä mahdollistaa paremman päätöksenteon mahdollisessa migraatiossa SDN-arkkitehtuuriin tulevaisuudessa.

Asiasanat

NSX, SDN, virtualisointi, ACI, ohjelmistot, tietoverkot

Author (authors)	Degree	Time
Teemu Takasuo	Bachelor of Engineering	December 2018

Thesis Title

Software-defined networking

45 pages

Commissioned by

Kymijoen ICT

Supervisor

Vesa Kankare

Abstract

Software-defined networking is a modern approach to networking, providing a flexible and cost-effective way of configuring networks with the help of software and virtualization. SDN technology is becoming more popular with each passing month and is in a defining role in the future of networking. The objective of this thesis was to clarify the way SDN technology works and what benefits can be gained from it compared to traditional networking architectures, and to create an example of software-defined network using VMware NSX.

This thesis was composed as a design-based study, and because there was zero-knowledge of the subject before the study was started, the initial phase consisted of studying the subject of SDN and making notes for installations.

The installations were done using both physical and virtual components. vCenter environment had to be established on physical ESXi hosts which were connected to each other through a Cisco 2960 switch. In the later phases of the installation, the physical layer was used minimally as the NSX network was fully virtualized and the routing between virtual machines occurred virtually without participation from physical network devices.

SDN environment was successfully created with separation between physical and virtual layers, enabling all-virtual routing. The objectives set for this thesis were successfully achieved. Kymijoen ICT gained useful information from the benefits and use cases of SDN technology, allowing them to make better decisions on the possible migration to SDN architecture in the future.

Keywords

NSX, SDN, virtualization, ACI, software, networks

CONTENTS

1 INTRODUCTION

Software-defined networking market grows worldwide each passing year. In 2013, SDN market volume worldwide was 0.41 billion U.S dollars. Now, in 2018, it has grown to 7.9 billion dollars and is excepted to grow up to 14 billion dollars in 2021. The increasing evolution and demand in network mobility drives businesses to switch to software-defined networks, and the need for this keeps growing each passing month. (Statista, 2018.)

This thesis study was commissioned by Kymijoen ICT, which provides diverse ICT-services for different businesses and municipalities in Finland. Kymijoen ICT has experience with multiple levels of networking and virtualization, and SDN is a potential networking model, which could be utilized in the future.

VMware, a global leader in cloud infrastructure and digital workspace technology, was also associated with the thesis, especially during the empirical part. The company provided the needed installation medias and licenses, enabling the creation and deployment of SDN environment with VMware NSX and always providing support when needed.

This thesis was composed as a design-based study. Kymijoen ICT required more information about SDN technology because of the potential advantages it could give for their company, thus creating the need for this study. The objective of the study is to create a better understanding of software-defined networking and its benefits, disadvantages and use cases. Lab environment, which demonstrates the use of SDN, is also created to see how the technology works in practice. (Hautamäki, 2015.)

2 SOFTWARE-DEFINED NETWORKS

2.1 What is Software-Defined Networking (SDN)

Traditional networks struggle to keep up with the networking requirements that are currently needed in the networking industry. Networks must get faster and more flexible every year, which in turn makes them more complicated and harder to manage. There is often need for on the fly configuration changes and data traffic management is becoming a problem due to increasing amounts of mobile data traffic. (Arora, 2017.)

Provisioning networks for new applications and users may take considerable amount of time and this creates increasing amounts of operating costs which businesses must handle. Cost optimizations like CAPEX and OPEX are barely present in modern networking at all and the time to market (TTM) is too long for today's needs. (Arora, 2017.)

Software-defined networking (SDN) is a technology, that aims to address many of the limitations that older network designs bring by virtualization, resource optimization, automation and simplification, which in turn reduce the costs of managing the network. SDN networks are highly flexible and can quickly adapt to the needs of different businesses on a case-by-case basis. (Pujolle 2015, 15.)

This is achieved by decoupling the physical and virtual layers from each other, enabling virtual devices to be loaded on hardware machines and making the network independent of the hardware. This way the system treats both physical and virtual devices the same way, making it possible to change the network without modifying the host machines at all. However, some manufacturers still require their own hardware for their SDN solutions to work. (Pujolle 2015, 16.)

2.2 Architecture

In a classic network architecture control plane and data planes are integrated and rely on each other to work. Data plane handles the traffic in the network

and the control plane defines the topology and how the traffic is handled. Every network device also has both planes on them at the same time, and because many protocols go through this one integrated layer, any changes made to the system depend on the network devices, protocols, and the software that is supported, which limits the changes that can be made. (Raza, 2018.)

In Software-defined networks, the architecture consists of three main layers, in which data plane and control planes are separated from each other. The layers are following:

2.2.1 Infrastructure Layer

This is where the data plane is, where data transportation happens and where the protocols and algorithms enable IP packets to find their destination in networks. (Pujolle 2015, 19.)

2.2.2 Control Layer

The second layer is the control layer, which contains the controllers and the control plane. The controllers provide the data plane with instructions on how to forward the data as effectively as possible. In SDN, controllers act as the “brains” of the network and are responsible of optimizing the performance of the network, deploying firewalls and overall managing the whole network and acting as a centralized control point. (Pujolle 2015, 19.)

2.2.3 Application Layer

The last layer is the application layer, which is responsible for the applications and their requirements for computing, storage, networking, security and management. It sends information to the controller about how the system must be customized for the applications to work properly. New services and applications can be easily implemented on networks, because of the communication between application layer and the controllers. See (Figure 1.) for reference and comparison of traditional and SDN architectures. (Pujolle 2015, 19.)

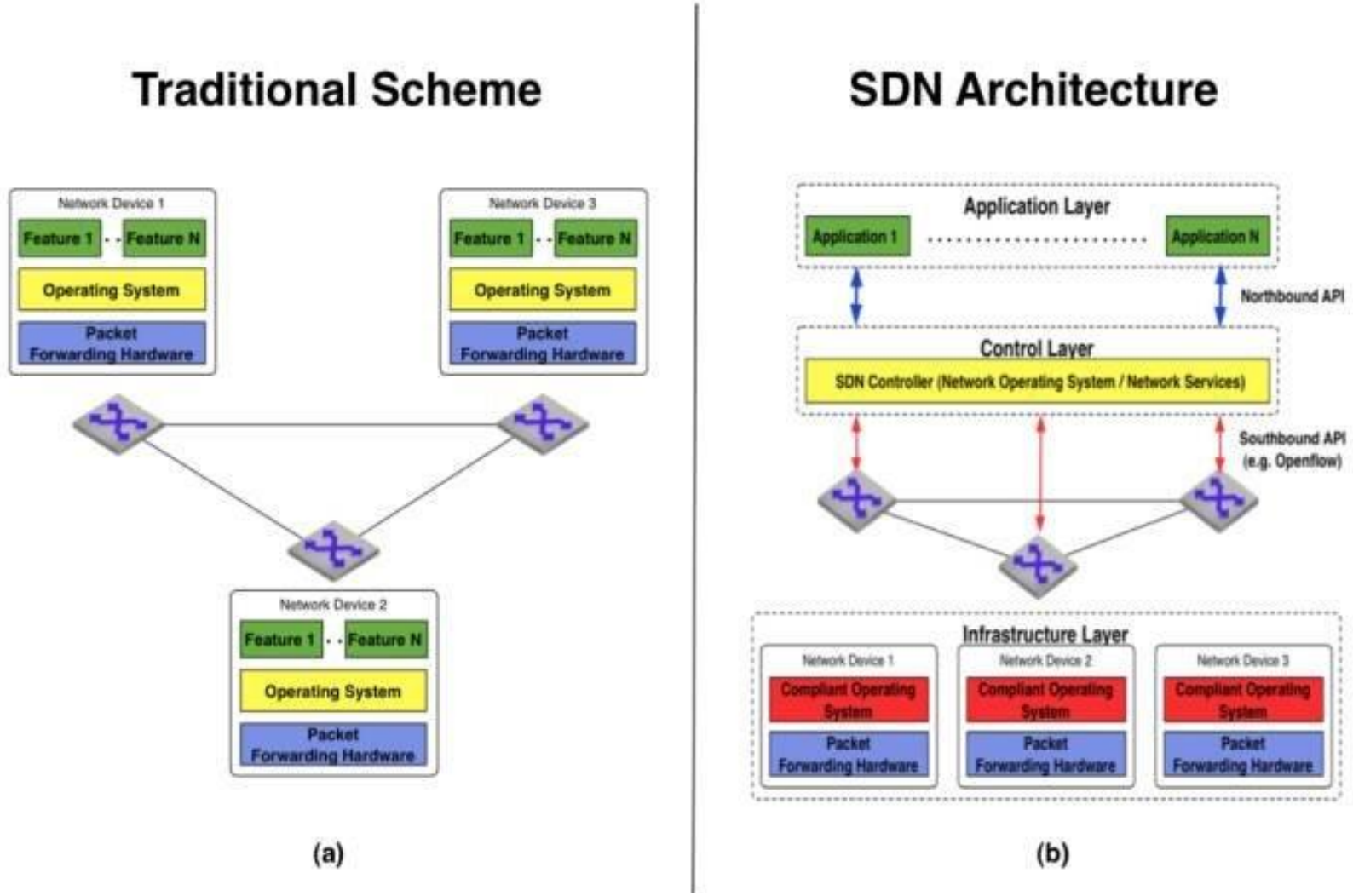

Figure 1. Comparison between traditional and SDN architecture (André, 2014)

2.3 Controllers

SDN controllers can perform numerous tasks and act as a centralized control point for the whole network. Because of their positioning in between of network devices and applications, they can manage flow control and enable intelligent networking. Controllers communicate through two different Application Programmable Interfaces, Northbound API and Southbound API. (Rouse, 2012.)

2.3.1 Northbound API

Controllers communicate with the services and applications through the Northbound APIs, so that they can analyze the applications needs such as security, quality of service and management. With the information gathered by analyzing, controllers try to build a network based on those needs. The proto-

col, which makes this communication possible, is called REST (Representative State Transfer). Currently there are not many different protocols for Northbound APIs to use, but many organizations are in the progress of designing different alternatives for REST. (Pujolle 2015, 31.)

2.3.2 Southbound API

In SDN architecture, Southbound APIs are used to communicate between the controller and network devices, such as virtual or physical switches and routers. This makes it possible for the controller to dynamically make changes on the network, based on the information received from the Northbound APIs. The most common protocol used in Southbound interfaces is OpenFlow, but many alternatives exist on the market. With OpenFlow, the flow-table of switches and routers can be changed based on the needs, thus making the network much more agile and responsive. (SDxCentral, s.a. a.)

3 SDN SOLUTIONS

Currently there are many different SDN solutions on the market from different vendors which could be talked about here. However, this thesis focuses on VMware NSX as the practical work was done based on it. However, because NSX is a fully software-based overlay network, which is built over a physical network, it is also good to talk about an underlay SDN solution, such as Cisco ACI (Application Centric Infrastructure), which is based on hardware. In some cases, both are used as an integrated SDN solution, NSX as the overlay network and ACI as the underlay network.

3.1 VMware NSX

VMware NSX is a platform for network security and virtualization from VMware. It allows virtual networks to be implemented on physical network and server infrastructure and to dynamically change, control and manage networks and their security through software. VMware NSX is also independent of the underlying hardware, as the virtual networks that it creates are programmatically managed and provisioned. This allows for very cost effective and adapt-

able network environments to be built without having to worry about any specific hardware. (Deuren, 2017.)

Even complex networks with multiple tiers can be created quickly with ease, because the entire network model is reproduced in software. Provisioning the network can take less than a minute to perform, which can take days without SDN solution like NSX. This is made possible, because NSX exposes RESTful API, which allows automated delivery of network services by cloud management platforms. This way there is no need of manual reconfiguration of hardware devices, as the network services are delivered to applications by the virtual network. (Deuren, 2017.)

The key functions VMware NSX can provide are its logical networking elements. These elements include:

- **Logical load balancing:** NSX provides support for layer 4 and 7 load distribution with SSL termination;

- **Logical firewall:** NSX uses a distributed, virtualization-oriented firewall which can monitor activity and provide identification at host kernel level;

- **Logical switching:** NSX can provide complete layer 2 or 3 switching virtually, decoupled from the underlying hardware;

- **Logical routing:** Dynamic routing between the logical switches and different virtual networks.

- **NSX gateway:** layer 2 gateways with transparent connection to the legacy VLANs and physical loads;

- **Logical VPN:** Software based site-to-site and Remote Access VPN (Virtual Private Network);

- **NSX API:** Any cloud management platform can be integrated with the RESTful API NSX uses.

These capabilities can be used to create diverse networks suited for the specific needs of different businesses. Next, NSX architecture and components will be described in detail. (Deuren, 2017.)

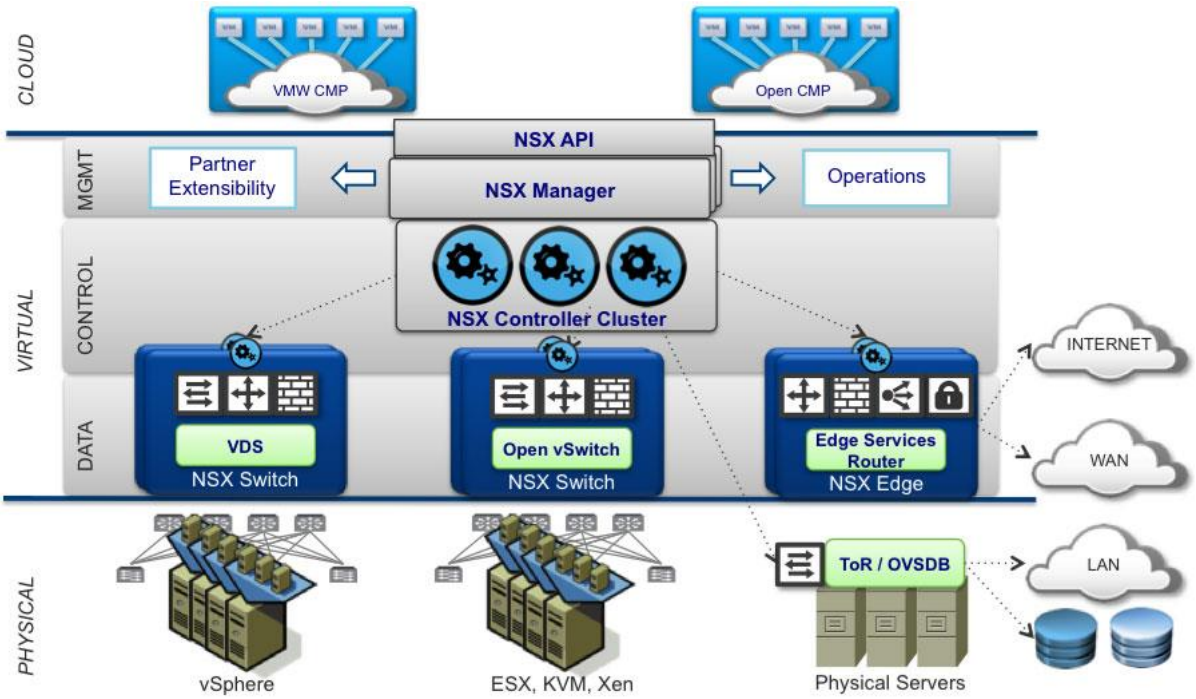

Figure 2. NSX architecture (Pujolle, 2015, 38)

3.1.1 NSX Manager

Starting from the management plane, referring to Figure 2, NSX manager allows the management of NSX environment through VMware vCenter. This is made possible by NSX APIs like REST, which vCenter uses to communicate with NSX and delegate tasks to correct parts of the environment. (Juniper Networks, 2014.)

3.1.2 NSX Controllers

NSX controllers are deployed in the control plane in a cluster of three controllers. They can be considered as the virtualized control plane of the SDN network. All the provisioning and the learning of MAC address table, ARP table and VTEP table is handled through the controllers, as is the controlling of vir-

tual firewalls and load balancing. VM and host information is collated together in the tables that each controller learns and then replicated through the NSX domain. This allows multi-cast free VXLAN to work on the underlay network, which reduces administrative overheard and alleviates a lot of complexity. (Juniper Networks, 2014.)

3.1.3 Logical switching and VXLAN

All the virtual routing and switching devices reside in the data plane, such as logical switches. Logical switches are software kernel based and they work as extensions to virtual Distributed Switches (vDS). Each switch is also part of a Transport Zone. Transport zones control which virtual machines and host clusters can use a particular network, and which hosts a logical switch can be part of. Unlike traditional virtual switches, logical switches offer a key feature of network virtualization; VXLAN (Virtual Extensible LAN). (Rajeev, 2017.)

VXLAN is a protocol used in overlay networks. It provides encapsulation of layer 2 ethernet frames in UDP (User Datagram Protocol) packets. VXLAN allows two devices in a virtualized network to communicate with each other by setting up an VXLAN tunnel between the devices. The encapsulation and de-encapsulation happens in the VXLAN Tunnel EndPoints (VTEPs), which in NSX are the Distributed Logical Routers (DLR) installed in the ESXi hypervisors. See Figure 3 for visual representation of the encapsulation process. (Rajeev, 2017.)

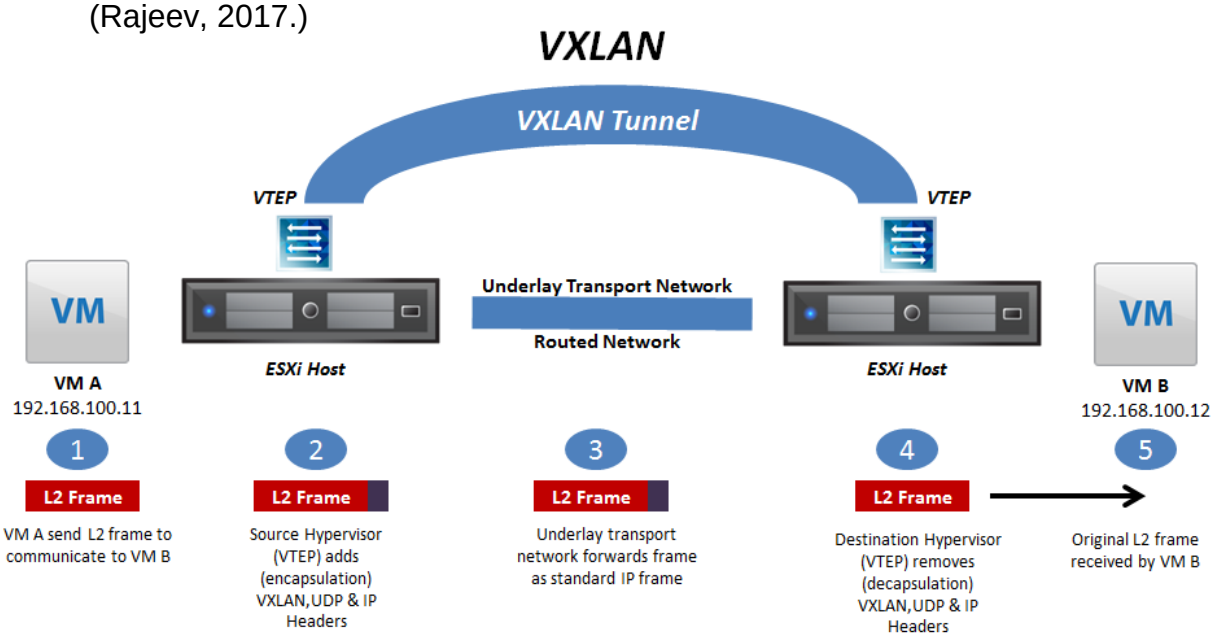

Figure 3. The process of VXLAN encapsulation. (Rajeev, 2017)

As with VLAN ID, that is used to identify different VLANs, VXLAN is provided with VXLAN Network Identifier (VNI). However, maximum number of traditional VLANs with the 802.1q encapsulation is 4096. Because VXLAN uses 24-bit VNIs, it is possible to deploy 16 million different VXLANs, making it much more scalable. (Gerard, 2017.)

The layer 2 ethernet frame segment is also identified by the VNI, making the communication between virtual machines or hosts that share the same VNI possible. However, MTU (Maximum Transmission Unit) of 1600 is required for the VXLAN encapsulation to work, as the process of encapsulating the layer 2 ethernet frames in UDP adds overhead. (Gerard, 2017.)

VXLAN uses different control plane modes (multicast, hybrid or unicast) for learning the destinations to forward traffic to, which depends on what is required on the network and what SDN solution is being used. With multicast, the process is basic “flood and learn”, which floods frames with multicast destination IP addresses that each VXLAN VNI is associated with and the physical network handles the replication and forwarding and does not require NSX controllers to work. (Kalitsev, 2015.)

In hybrid mode, NSX controllers maintain a table of VTEPs that have joined each VXLAN. Hosts directly communicate with controllers to check matching entries in their VTEP tables, and the physical network performs the forwarding process. (Kalitsev, 2015.)

Unicast mode does not require physical network replication and only relies on the controllers. Unicast does not use multicast IP addresses at all, instead both source and destination host compile a list of VTEPs with the same VNI as themselves. All replication is done on the hosts, resulting in unicast communications between the VTEPs. With the utilization of controllers, ARP suspension is also available. (Kalitsev, 2015.)

3.1.4 Logical distributed routing and edge gateway

Going back to Figure 2, there are still some important elements of NSX to discuss about, such as the DLR (Distributed Logical Router) and ESG (Edge Services Gateway).

DLR in NSX is a virtual appliance that contains the routing control plane. The data plane is distributed to each ESXi hypervisor in kernel modules, which encapsulate the traffic inside a VXLAN header, as talked about earlier. Each DLR can create logical switches with VNI, that act the same way as physical network VLANs. Multi-tier networks for different applications can be created with ease using logical switches and routers. (Juniper Networks, 2014.)

Distributed Logical Routers provide the virtual networks with many important functions, such as routing and bridging, enforcement and security policies, mac address learning and flooding and acting as a default gateway for all the VMs. They also provide many different routing functions, such as OSPF or BGP and they peer with ESG for egress traffic progressing, outside of the virtual networks. These qualities are why DLR is used for routing the east-west traffic, which means all the traffic that happens inside the network between the virtual machines and devices. (Juniper Networks, 2014.)

For all the traffic moving from the internal virtual network to the outside network such as physical devices and the internet (also called north-south traffic), Edge Services Gateway is used. ESG provides many gateway services, such as dynamic routing, load balancing, firewall, VPN, NAT (Network Address Translation) and DHCP (Dynamic Host Configuration Protocol). (VMware, s.a.)

In the top layer are different cloud management platforms, which are not the focus on this thesis study. Moving on to the bottom, all the physical networking and server equipment reside there, such as the ESXi hypervisors. The physical, underlay network is also where the Cisco ACI SDN solution is running on.

3.2 CISCO ACI

Cisco ACI is an underlay SDN solution for data center and cloud networks, which integrates hardware and software and is driven by policies. The ACI is based on the Cisco Nexus 9000 switch family and is required for the solution to work. However, it can be integrated to older Nexus 7000 fabric of switches for some savings when building the ACI fabric. The points where the software integration happens in ACI include components such as Data Center Policy Engine, additional Data Center Pod and virtual and physical leaf switches. The Application Virtual Switch (AVS) from Cisco is used to push policies to the virtual switches. (SDxCentral, s.a.)

3.2.1 ACI Leaf-Spine fabric

Cisco ACI uses a Leaf-Spine fabric, that offers a linear scale in performance and cost. If more device connectivity or servers are needed, a new Leaf switch can be added up to the capacity of the Spine. If more redundancy or bandwidth is needed, more Spines can be added. (Wang, 2016)

The Leaf-Spine ACI fabric uses native layer 3 IP provisioning, supporting ECMP (Equal-cost-multi-path) routing between the endpoints in the network as well as using VXLAN and other overlay protocols for network location independent workloads. Physical or virtual machines can reside in the same logical layer 2 domain network with routing on layer 3 ran down to the top of each rack. (SDxCentral, s.a. b.)

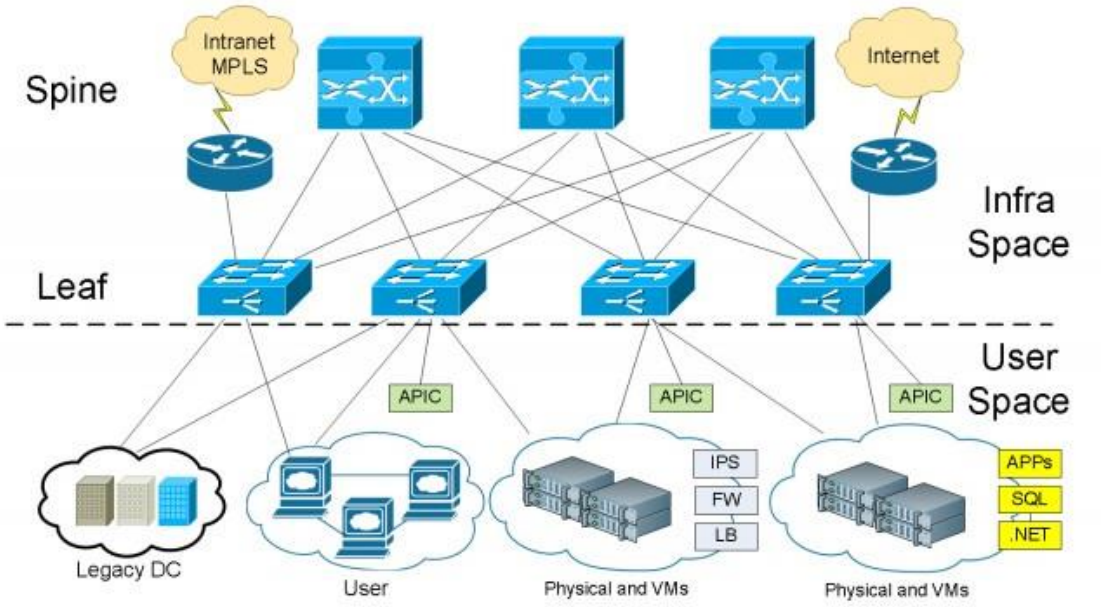

Figure 4. Cisco ACI fabric (Wang, 2016)

3.2.2 APIC

As with VMware NSX and the NSX controllers, Cisco Application Policy Infrastructure Controller (APIC) is in a central role of managing the network, as it is the single point of policy management in an ACI fabric. The APIC differs from the most of SDN controllers and designs in a way, that there is no de-coupling of the control and data planes from each other. It is only used for configuring the policies, which are then delivered and instantiated on every node in the network. This allows higher orders of logic to be better integrated with the consumers of the network, such as the system and application teams. (SDxCentral, s.a. b.)

For an example, when deploying a 3-tier application in traditional networks, administrators must at least know the IP ranges, VLANs, firewall and load balancing policies. These are network-centric terms that the consumers of the network may not know. (SDxCentral, s.a. b.)

Cisco ACI tries to resolve the communication gap between the consumers and the network adiminstrators by creating EPGs (Endpoint Groups), that focus on a specific area like web, applications, or database. Through APIC, contracts about the desired functions like quality of service, firewalls or load balancing

are created between the EPGs. When more hosts or virtual machines are required by businesses, all that is needed is to place the machine in the correct EPG, which dynamically changes all the required configurations according to the policies from the APIC. (SDxCentral, s.a. b.)

APIC also provides Layer 2 – 7 analytic visibility on per host, tenant or application level, providing tools to check information such as condition of the network, hardware utilization and latency. For more traditional testing pinging is also possible, as is checking bandwidth, packet loss and jitter, making management of the ACI fabric quick and efficient. See Figure 5 for reference of APIC in use. (Wang, 2016.)

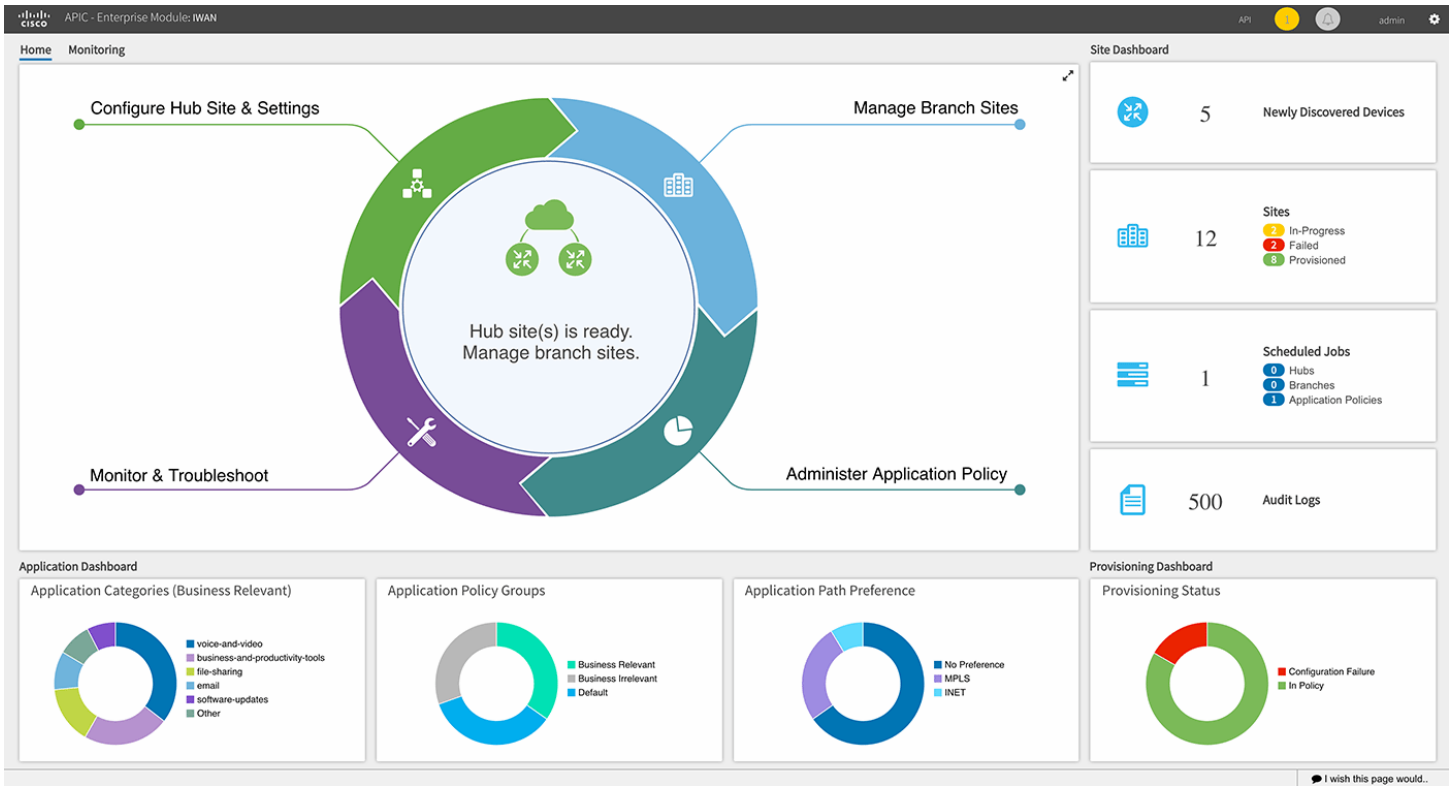

Figure 5. Cisco APIC GUI (Wang, 2016)

4 SDN USE CASES

Software-defined networking is a large concept. There are many different solutions and techniques being used and applied, and there is a clear benefit of transferring to SDN from traditional networks, at least on theory. One might wonder, however, what are the real use cases for different SDN solutions. This chapter aims to clarify, why different businesses have decided to change to SDN and why would it be beneficial to do so by giving examples and quotes from the actual businesses, that have done the change.

4.1 Use cases for VMware NSX

Starting from VMware NSX, many businesses first choose NSX because of the increased security that it provides with features such as microsegmentation. Hardware-based firewalls are not required with microsegmentation, as applications can receive security policies on the workload level, providing integrated network security to the virtualized workloads. However, some businesses find out about the other use cases of NSX than just security, such as automation, multi-data center pooling and disaster recovery, and expand to them. (Hardcastle, 2017.)

4.1.1 Security

Alliant Credit Union deployed NSX in June 2016 for its security features but found other benefits as well, as Julio Arevalo from Alliant Credit Union (Hardcastle, 2017) said: “I can’t say we didn’t expect it, but a number of issues were resolved with the migration. The most noteworthy was the data warehouse load.” The completion of the process would take up to four hours and occasionally cause timeouts. Timeouts haven’t happened with NSX and the process takes less time. (Hardcastle, 2017.)

Alliant Credit Union also had same positive effects affecting their SQL, which used to timeout too when transferring files from SQL to file server on off-site partners. The company also uses NSX load balancing and Guest Introspection properties, which removes the need of anti-virus agents within the guest OS (Operating System). (Hardcastle, 2017.)

4.1.2 Data restoration

As talked about in chapter 3.1, NSX networks are created in software, which can be saved, deleted and restored on demand without the need of addressing the physical network. For disaster recovery purpose, this means networks can be replicated automatically between protected and recovery sites and can be as simple as copying, pasting, keeping and syncing. (Hardcastle, 2017.)

A service provider company called Expedient use NSX to allow their IT infrastructure components to be replicated by the customers, reducing the recovery times considerably. Something that took hours to do before, could be done in minutes with NSX. (Hardcastle, 2017.)

John White from Expedient also tells: “We use NSX to basically span the customer’s network from their premises into one of our data centers to make that seamless to them. It allows them to move virtual machines from one place to the other without any reconfiguration, without any issue.” (Hardcastle, 2017.)

4.1.3 Pooling with NSX

John White also has use cases for NSX Pooling. In one situation an application of a customer had to be migrated to the cloud by Expedient. The way this was resolved, was by moving the application to a new private cloud created for this specific case in the customers datacenter. After that, they were able to replicate it in one of their datacenters and used NSX to create one big network for the customer between their site and Expedients data center. (Hardcastle, 2017.)

For the last use case, Expedient has a customer, who has data in two Expedient data centers. With NSX, the two datacenters are connected as one and visually made to seem as it would be located between the two sites to allow customers to transfer applications between them. The process is called multi-data center pooling, which gives flexibility and mobility and allows resources to be used on single operations on different sites, expanding applications to where the capacity is. This technology is used by companies to better allocate their resources. (Hardcastle, 2017.)

As can be seen, while this is only a small part of what VMware NSX can provide, there are many use cases where NSX has helped a company to overcome some of their problems and make their networking overall more fluid and reliable.

4.2 Use case for Cisco ACI

Moving on to the Cisco ACI there are some differences, on what businesses look for in underlay and overlay SDN solutions. Referencing to a Cisco interview with E-Trades Sr. Manager Network Engineer Jaz Rahul on (SDxCentral, 2015), there are some clear benefits and use cases with Cisco ACI.

E-Trade already had Nexus 9000- fabric running on their market data environment for high port density and bandwidth. However, they faced some problems that needed to be solved somehow, such as automation flow. Too much time was used on turning up ports and configuring the network to support the server. (SDxCentral, 2015.)

Using ACI, they were able to programmatically automate the process of putting the server on the network and the right work groups with the API calls, reducing the operational expenses considerably. Acquiring a server and becoming it running could take up to fifteen hours, and with ACI the process could be done in two hours. (SDxCentral, 2015.)

The process of becoming a server running required multiple teams working on it on a sequence of tasks such as OS installation, storage, network and application provisioning, security, and others. This process could take a long time, and because so many different persons were working on the same process, it was also prone to human errors. (SDxCentral, 2015.)

With ACI, every team can start off with the involvement and can talk about the required policies that must be distributed. When that policy is defined and working, it can be used and tweaked for the next workflow that is added to the network, without the need of going through the long sequence of tasks that was needed before, making the change for human error almost negligible. (SDxCentral, 2015.)

4.3 Conclusion

Overall, there are tens of different SDN solutions commercially available on the market right now. On this thesis, only the market leaders VMware NSX

and Cisco ACI are talked about, and there are many more with different properties and uses for the interested.

However, by comparing the two, both provide businesses with different properties of software-defined networking suited for the needs of the buyer. In some cases, both are used at the same time as an integrated solution, because of the different properties they offer, allowing the other to do work where the other one lacks, mitigating the potential weaknesses of one system.

Talking about the weaknesses, there are not many known currently, as SDN is still relatively new technology, and there has not been that much research done on the subject yet. One disadvantage, that often is talked about is the security. Even though SDN networks are advertised as secure solutions with properties like micro-segmentation and distributed firewall, all the way to zero-trust models, there is still one problem. With centralized control, comes centralized vulnerability.

There are concerns about how the potential hacker can access the controllers of SDN network, and gain access to everywhere through it. This is a real possibility, and businesses should do everything to keep the controllers as safe as possible from the hackers and other potential dangers.

The implementation cost is another disadvantage that comes with SDN, depending on the solution. In all cases, implementing SDN solution to an existing network requires some configuration changes in the network.

Underlying solutions such as Cisco ACI may cost more initially, as it requires exact hardware to work and it must be implemented in the physical network. VMware NSX, on the other hand, works with any underlay solution and is implemented virtually, in the overlay network. NSX is also sold on per CPU license basis and does not need new hardware to function, and as such is potentially cheaper to implement in smaller to mid-size businesses. Both offer potential savings in CAPEX and especially OPEX in the long run, but the initial cost and the risks of switching to completely different networking architecture may be too much for some businesses.

The next chapter is the start of the practical part of this thesis study, which is about demonstrating, how an SDN solution like VMware NSX is deployed to a small lab environment. In a small environment like that, NSX can't really be utilized properly, but the key elements in the deployment stay the same.

5 CREATING A VCENTER ENVIRONMENT WITH NSX NETWORKING

5.1 Preparing for the installations

The first installations were done without much knowledge about the vCenter environment. Trying to build the environment and learning while doing it seemed to be the best way to get more familiar with the environment and the software that is used with it. This approach, however, caused some errors in the configurations and a lot of troubleshooting had to be done while the environment was being built.

The errors and mistakes were fixed on later installations, when the environment was built with much more knowledge of the subject and better preparation before the installations. The whole process is described on this chapter, starting with the early installations and in the end proceeding to the final and working environment.

VMware NSX is a network virtualization platform software installed to vCenter Server, so a working vCenter environment must be created first to be able to install the NSX. It was decided to use physical server hardware on which to install ESXi and deploy the vCenter. A Sunfire server was chosen as the device to do the installations on, which should have had at least 32Gb RAM and a 4-core Intel Xeon processor. Later this turned to be false.

5.2 ESXi

ESXi hypervisor is the core of vCenter environment. Every virtual machine that is created runs on it, including the vCenter itself. The first step was to download an ESXI 6.5 ISO-file from VMware's website, which will be converted into an installation media using a software called "Rufus" and an USB-storage device

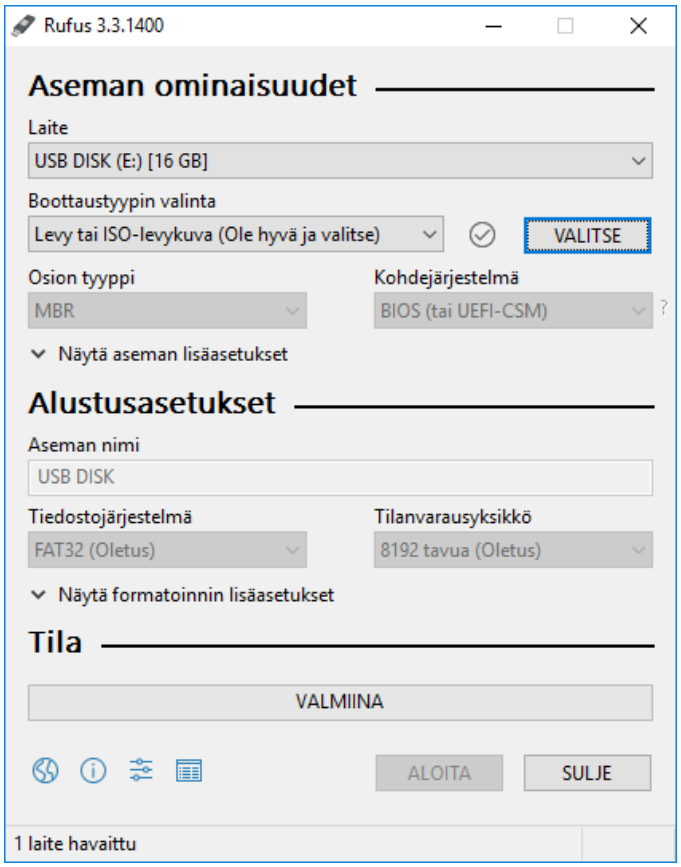

Figure 6. Rufus

.

The process formats the USB-storage device and converts its file system to FAT32. The ISO-file is also converted to a form, where devices can read the files and boot from the USB-storage.

Installation was started by inserting the USB-storage device to the servers USB-port and changing the BIOS settings to prioritize the storage device as the main booting option and then booting the server from the USB-storage. The installation itself is straightforward, passwords for the system and the location of the installation are the main settings to be considered. After the installation is completed, ESXi will try to get an IP address with DHCP, but a static IP is needed in this case, so the network settings were changed for the following:

IP address 10.69.10.120 255.255.255.0
Default gateway 10.69.10.1
DNS 8.8.8.8, 8.8.4.4 (Googles DNS servers)

Now the ESXi was properly configured and running. The network tests in ESXi passed and it was now possible to connect to the devices through the VPN in the ICTlab environment. The connection could be established by using Cisco Anyconnect- software and connecting to vpn.ictlab.fi, opening a browser and typing the ESXi hosts IP address. The browser will warn about the SSL certificates of the ESXis website, but the message can be ignored without hesitation. For this kind of lab environment, these warnings are normal.

The credentials that were defined during the ESXi installation could be used to log in to the system. At this point however, no changes were needed, and it was possible to move on to vCenter Server installations.

5.3 vCenter Server

vCenter Server has a few different versions with different properties available online. For this thesis study VCSA (vCenter Server Appliance) was chosen, so that there is not a need for Windows-servers or specific hardware. VCSA can be fully virtualized and ran on a ESXi host.

The installation was started by using an ISO-image of VCSA 6.5 downloaded from VMwares website and mounting it on a computers virtual CD drive and running it. It opens a clear installer that gives all the needed configuration options that for a working vCenter Server.

For the appliance installation type, Embedded Platform Services Controller was chosen. This means that the Platform Service Controller and all the services needed for the vCenter Server to run are installed on the same device as the server itself and everything runs on one vCenter Server instance. This works the best for small environments, in which simplicity and resource management are high priority, such as the environment in this thesis study.

The other option, External Platform Services Controller, would install all the services and the Platform Service Controller in a different system that is separated from the vCenter Server. This would work in a much larger scale network environment, where there is a need for many different vCenter Server

instances which can be controller through one Platform Service Controller, making larger scale management easier.

The vCenter Server will be installed on the ESXi host that was installed earlier and the deployment size was set to tiny to limit the use of resources as much as possible. At this point the DNS settings were not set up properly, so the deployment target was defined with an IP address instead of FQDN (Fully Qualified Domain Name).

SSO (Single Sign On) was configured with default values. Googles DNS and NTP servers were used and a static IP address from our school lab network was assigned to the server from the same network as the ESXi host. The settings could have been better and further thought out, but at the time the point was to get the environment running and to get more familiar with it. See Figure 7 for the configuration settings.

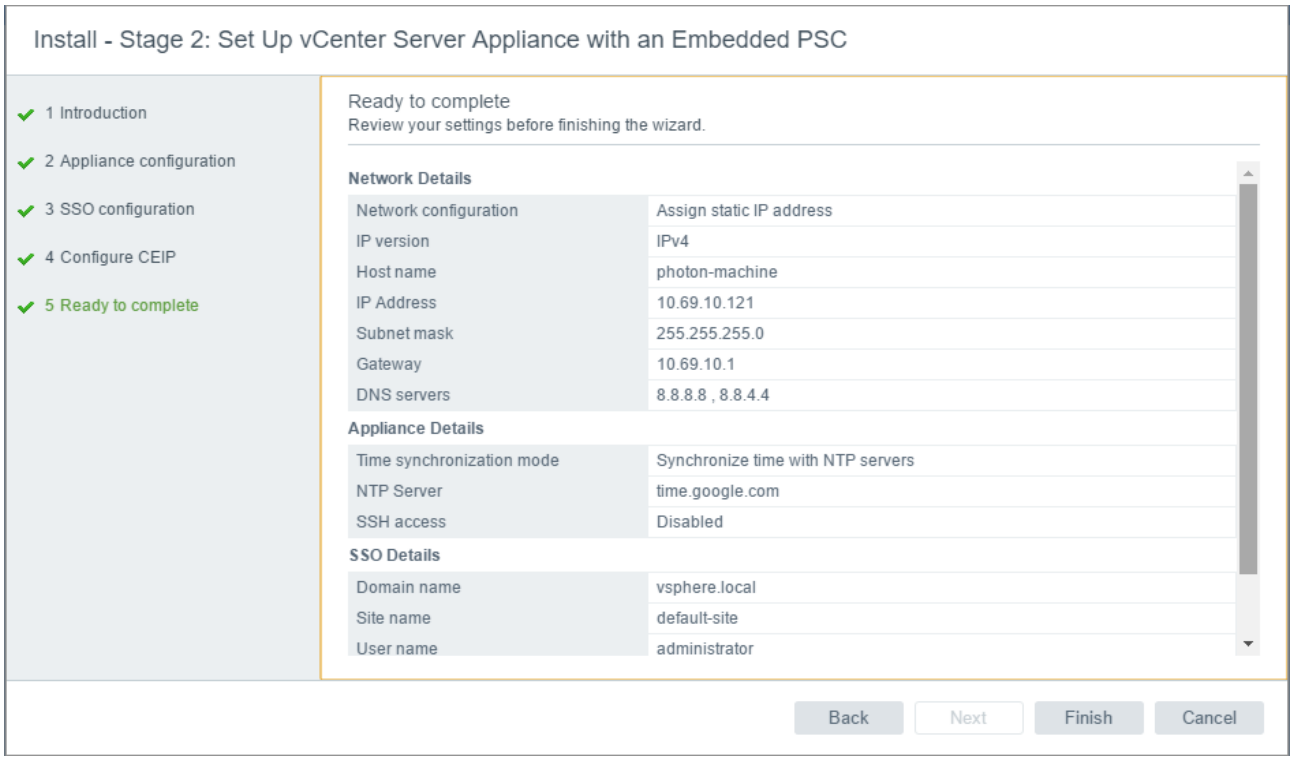

Figure 7. VCSA settings

After the installation was finished and everything was working, it was possible to connect to the vCenter Server through a web browser by typing in its IP address. This site is mainly for monitoring and general management and only the NTP settings were modified to use the correct time zones and to synchronize the time between vCenter and ESXi. (Figure 8.)

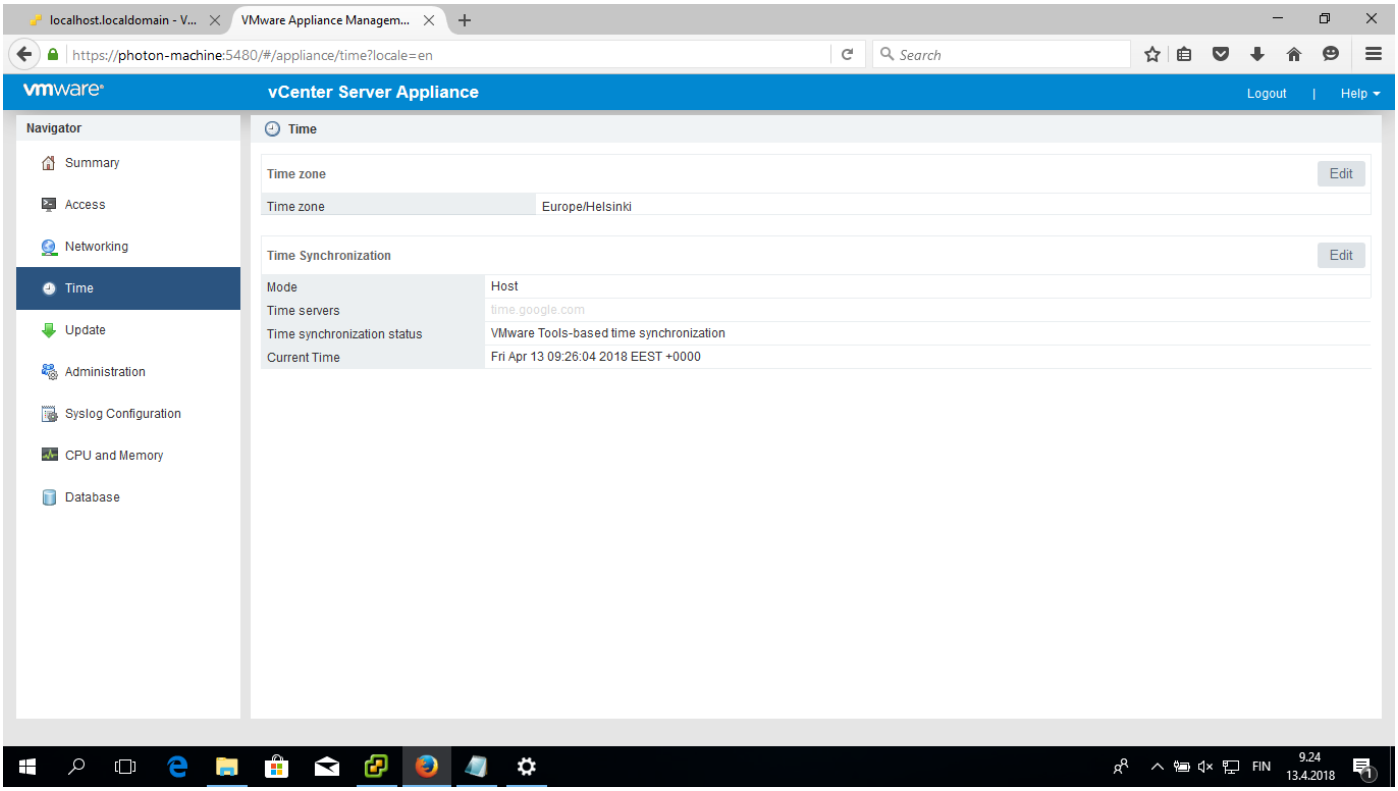

Figure 8. VCSA

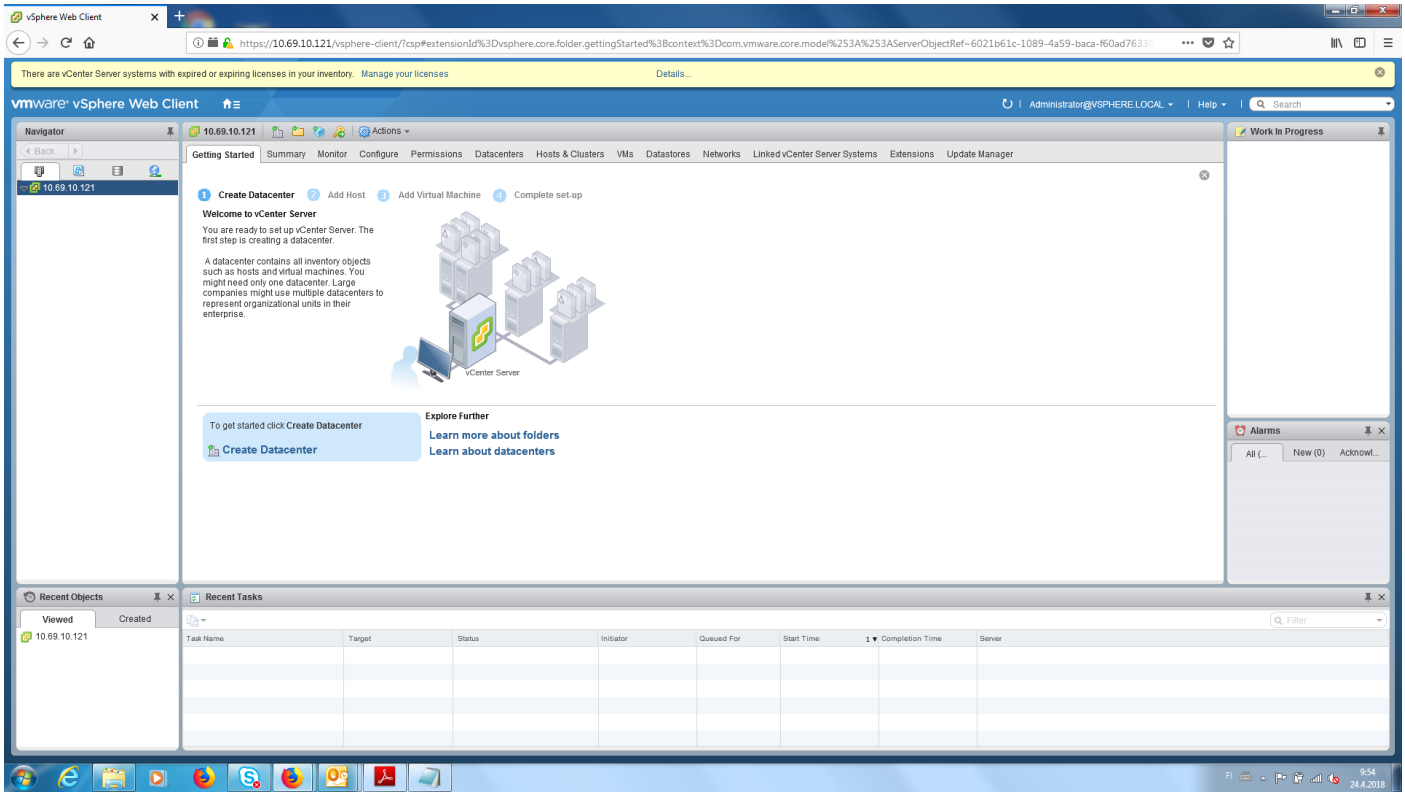

Figure 9. vSphere Web Client

For the ESXi and the vCenter to recognize each other, a virtual datacenter had to be created and the ESXi host added to the datacenter. This way they can communicate with each other and the hosts and virtual machines can be managed and changed from the vSphere Web Client.

After the datacenter was created and the host was added, it was possible to take a better look of the environment that was built, and the first major problem soon occurred. The physical server, on which everything was installed, had only 12Gb RAM instead of the 64gb that should have been there. At minimum, NSX environment needs 32Gb of memory to work correctly.

There was one another sunfire server in the school, which also had 12Gb ram installed. Everything on that server was removed and then installed to the one used in this thesis study. The RAM sticks also were older ddr- memory and in total the server still had only 24Gb to use, so there were two options left. The work could be started over with new devices or NSX could be deployed with current setup and 24Gb ram to use. It was decided to try and install NSX to see how it looks and to see if it is possible to run it on lower memory.

The installations were done quick just to see how the system handles NSX with such low memory and the actual installation process will be described in more detail later. Everything seemed to be working at first, but after deploying some new virtual machines and the controllers for NSX, the environment started to have some problems running properly and it was not possible to continue any further.

The work had to be started again, this time with different devices and better preparation. Now, as the software and the environment overall were both much more familiar, it was easier to find information and plan how to build this new environment.

5.4 Starting over

This time the ESXi installations were done on three different physical computers with 32Gb RAM and three ethernet ports available each, two on the external NIC and one integrated on the motherboard. The hosts were connected to

each other through a Cisco 2960 switch. Running the environment on three hosts provides redundancy and allows the NSX controllers to be placed correctly on their own hosts and datastores later when setting up NSX. The installations were done on a new network and the schools DNS was used to be able to use the hostnames of the machines when connecting to them.

IP address: 10.69.51.10, 10.69.51.20, 10.69.51.30
Default gateway: 10.69.51.1
Hostnames: nsx-esx1.ictlab.local, nsx-esx2.ictlab.local, nsx-esx3.ictlab.local
DNS: 10.69.10.11, 10.69.10.12
NTP: time.google.com

After the installations were finished it seemed like the NICs (Network Interface Card) were not working on any of the computers. This was caused by a compatibility error with the 6.5 version of ESXi and the NIC. It was possible, however, to enable community supported VIBs (vSphere Installation Bundle) by connecting to ESXi host with SSH using putty, logging in as root and typing the command:

```
# esxcli software acceptance set --level=CommunitySupported
```

Now, it was possible to download and install community made drivers and other packages from the internet and check the model of the NIC with the command:

```
# lspci -v | grep "Class 0200" -B 1
```

According to the output, the computer had a Realtek 8168 NIC installed in it and now it was possible to download and install the right drivers for the NIC with:

```
# esxcli software vib install -n net51-drivers -d https://vibsdepot.v-front.de
```

After the installation was complete the ESXi had to be booted for the changes to take effect. After this was done to all three hosts, they all had properly working NICs in them and the work could be continued with vCenter installations.

vCenter was installed on nsx-esx1 with some network configuration changes:

IP address: 10.69.51.100 255.255.255.0
Default gateway: 10.69.51.1
Hostname: nsx-vcsa.ictlab.local
DNS: 10.69.10.11, 10.69.10.12
NTP: time.google.com

The earlier DNS configuration was not working properly and none of the names of the devices and machines could be resolved. After adding all the ESXi hosts and vCenters hostnames and IP addresses to the DNS forwarding and reverse lookup zones, it was possible to connect to all the virtual machines and devices with their hostnames and use FQDN (Fully Qualified Domain Name) on the installations. All three ESXi hosts were added to one “Management and Edge” cluster after creating the datacenter.

5.5 vDS and migrating the network

Now, for the NSX to work properly, it needs the advanced features of vDS (vSphere Distributed Switch) instead of vSS (vSphere Standard Switch), which everything is connected to when vCenter environment is created. Distributed switches allow different hosts to use the switch if they exist within the same host cluster. It extends its ports and management across all the servers in a cluster and can support up to 500 hosts per switch. This means, that a VDS needs to be created and the current network must be migrated over to the VDS from the vSS. For the NSX VXLANs to work properly, MTU of 1600 must be configuired on the vDS.

After the vDS was created, uplink from the external NIC and two port groups were assigned to it, one for vCenter VM Network and one for ESXi hosts. However, migrating the hosts to the vDS caused a connection error between vCenter and the hosts. There shouldn’t have been any errors, because there was always a backup connection for the ESXi, as it had two uplinks, one to the VM Network and one to the vDS. After trying to migrate vCenter with the ESXi 1, the migration didn’t go through as the vCenter automatically performed a rollback on the settings with an error message stating, that the con-

nection to the ESXi host was lost and the configuration changes have been reset.

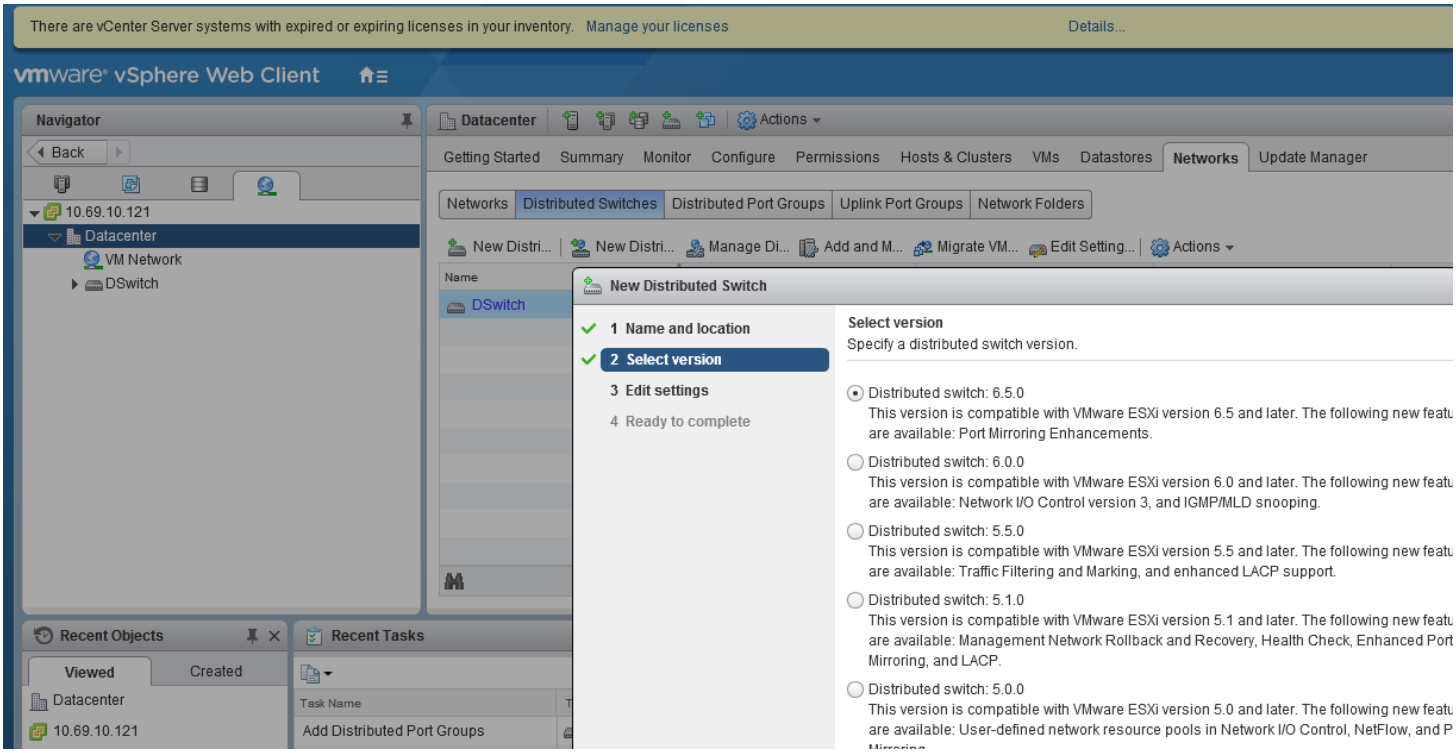

Figure 10. Creating a vDS

The fault appeared to be in the NIC of the computers, caused by the compatibility issue with ESXi, as discussed earlier. Even though vSphere recognized the NICs and they seemed to be usable, no network traffic was able to pass through their interfaces. This meant, that an alternative way to migrate the network from vSS to vDS had to be invented, as only one ethernet port was available for use and the network rollback feature prevented the migration.

A solution was found within vCenter Servers advanced settings. From there, it was possible to disable the network rollback feature. The only problem with disabling that option is, that it is possible to lose the connection between ESXi host and vCenter completely because there are not any safety measures to prevent it in a case where loss of connection happens. However, it was the only option available to migrate the network to the vDS.

After changing the setting, it was possible to try the migration again. ESXi hosts still lost the connection when migrated to vDS, as expected with just one uplink available and vCenter still connected to vSS. When vCenter vas migrated with ESXi 1, all the connections to vCenter vas lost. It was still possible

to connect to all the ESXi hosts, but vCenter just would not come back up, even after rebooting.

According to ESXi, vCenter did migrate with the host and a ping test between the two was successful using the ESXi hosts command line. However, it was not possible to connect to vCenter through browser or PuTTY (a SSH and telnet client for windows) through the management computer, on which all the setups and connections were done earlier.

Through ESXi, it was still possible to connect to the vCenter Servers console to see if all the settings were correct. First the shell had to be enabled and launched with the following commands:

shell.set --enabled true (enabling the shell)

shell (launching the shell)

cd /bin (changing to correct directory)

Then it was possible to list and try to restart the services with:

service-control --list

service-control --stop --*all*

service-control --start --all

Most of the services (in example vmware-vpdx) that are required for vCenter Server to run properly were stopped and trying to restart them returned an error message stating, that the service name is invalid. The service names were double checked, and the commands used were right, but the vCenter had just lost the services somehow and there was no way to restart them.

The real reason why vCenter could not find and start the services anymore was never found out. There is a change it was caused by an error during the migration to vDS, as the port group vCenter was migrated to was not set to

ephemeral binding, but as static binding. Also, the network rollback feature was disabled, so the false configuration would have gone through without vCenter stepping in and rollbacking the change.

Because the ESXi and vCenter trial licenses were running out and new 6.7 versions of them were just released, as well as NSX 6.4.1 that supported the new versions, it was decided to once again install everything with the new versions and new licenses.

5.6 The final installations

This time all the installations were done the same way, because everything was working as it should on the previous environment. Only the network migration from vSS to vDS was done differently, and successfully this time. When the port groups on the vDS were being created, the one used for vCenter migration was changed to ephemeral binding and the one used for the hosts was left to static binding. Figure 2 can be referred to when reading about the installations to see a better overall picture of what is being installed and where.

First all the hosts were migrated to the vDS including ESXi 1, which is where the vCenter was running on. This meant, that the connections were once again lost. Now, because the vDS port group was first set to ephemeral before the migration, it was possible to change the vCenters network through the ESXi 1 from the virtual machine settings. Without the ephemeral binding, it wouldn’t have been possible to switch the network from vSS to vDS because vCenter would not allow it.

After vCenter was successfully migrated and it was possible to connect to it through a browser, the port group binding was once again changed to static, which is recommended for general use. Ephemeral port binding is mostly used for recovery purposes, and if a virtual machine assigned to it powers off, the port is deleted. In this case it was required for the migration but has no other uses. Now with everything running through vDS, it was possible to start the NSX installations.

5.7 NSX

NSX is installed to vCenter Server by creating a new virtual machine and deploying it with an ova file. The settings include virtual machine name, installation destination, licenses, passwords for administrator usage, destination network and the actual network settings. Now, because the earlier network migration was successful, the NSX can be correctly deployed under the vDS network. The network settings were configured as following:

Hostname: NSX-manager
IP Address: 10.69.51.50 255.255.255.0
Default gateway: 10.69.51.1
DNS: 10.69.10.11, 10.69.51.12
NTP: time.google.com

After the installation was done, it was possible to connect to NSX IP address on a browser to access NSX Manager Virtual Appliance Management to see if NSX installed properly by checking if all the required services are running. The services to check are vPostgres, RabbitMQ and NSX Management Services, and all of them were running properly.

Before NSX can be used, adding it to the vCenter Server is required. This was done by adding the vCenters IP address and SSO user name to the settings. If the connection is successful, the status changes to “connected” and NSXs “Networking & Security” icon appears to vCenters inventory.

The Lookup Service was also configured, even though it is used mostly to authenticate users from other identify sources such as AD or LDAP. Both processes are done under the “Manage vCenter Registration” menu in the NSX Manager Virtual Appliance Management. Nothing else needed configuring here, so it was time to set up NSX for actual use. All the future NSX configurations are done under the “Networking & Security” menu in vCenter.

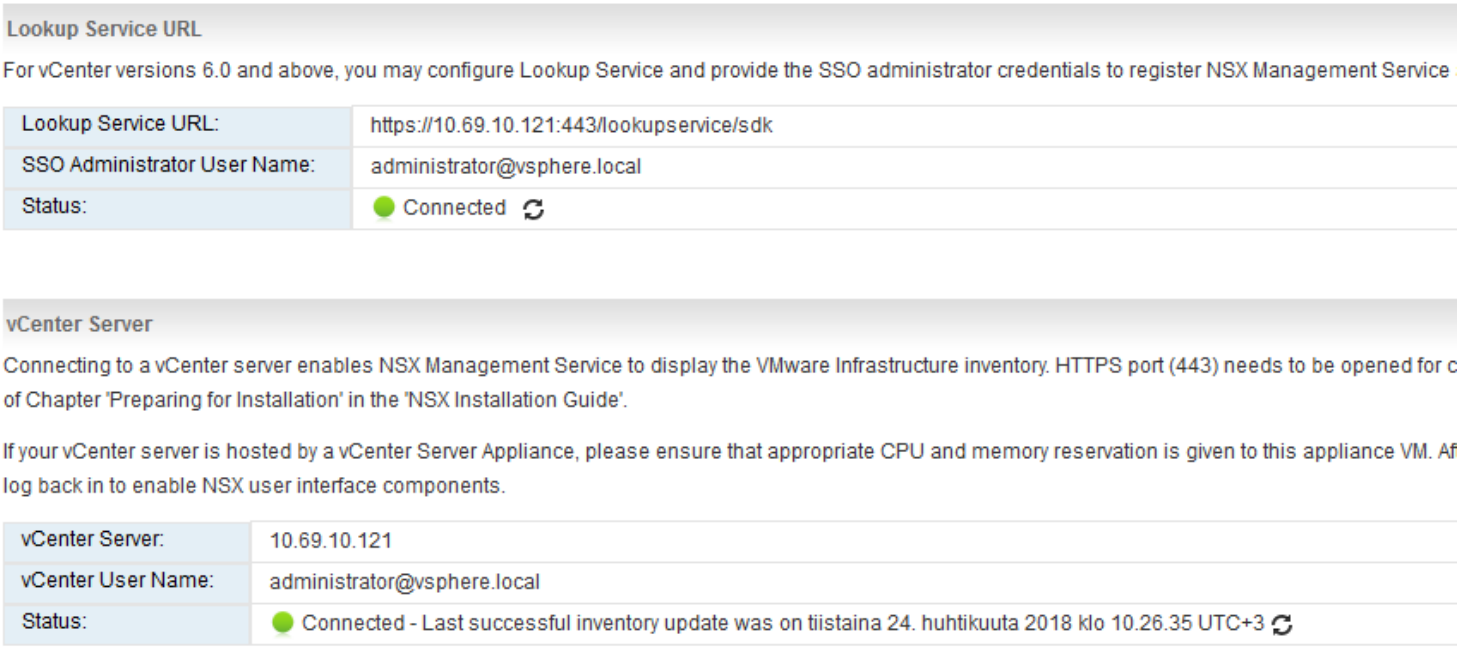

Figure 11. Adding NSX to vCenter

5.7.1 NSX Controllers

Three NSX controllers were deployed (Figure 12), each to different ESXi host. This is for redundancy and failover capacity. Even if one ESXi host fails the controllers can still operate without problems. When the controllers are deployed, they automatically create a cluster among themselves in which they will join one by one after the deployments.

All the controllers are placed to the vDS, as all the other components in the environment are. Also, an IP address pool of 10.69.51.40 - 10.69.51.42 is created for the controllers and an IP address from that pool is automatically assigned to every created controller one at a time. Now, with NSX manager and controllers deployed, management and control planes are established in the environment. (Figure 13.)

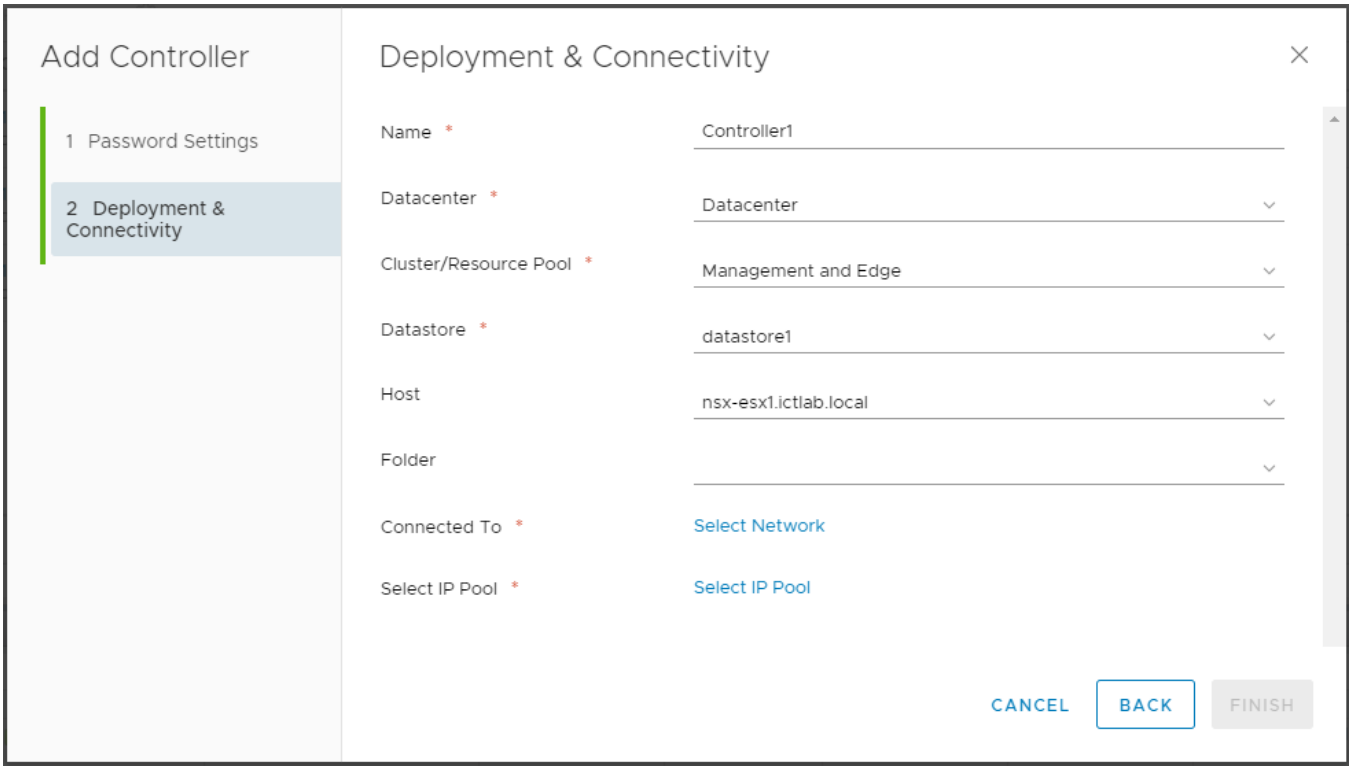

Figure 13. Adding the controllers

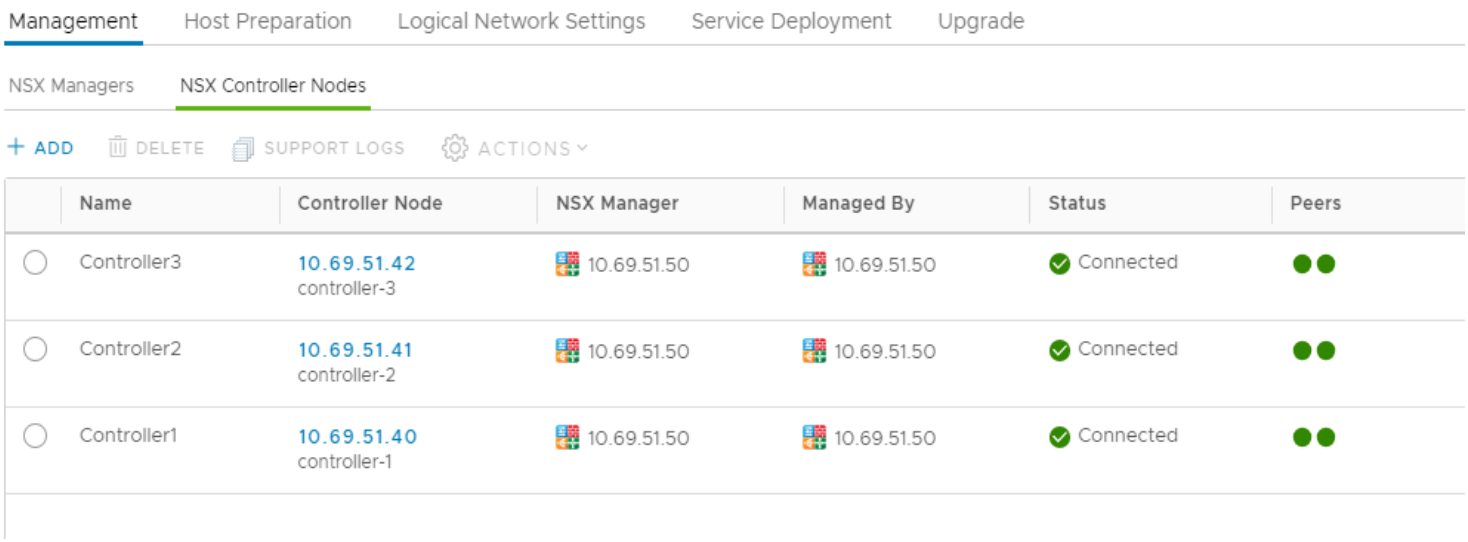

Figure 12. Controllers deployed

5.7.2 Host preparation

The hosts are prepared for NSX to install different features to ESXi using VIB, such as VXLAN, distributed firewall and distributed routing. The actual preparation process was as simple as clicking “Install” in host preparation panel. This process installs the necessary VIBs to all the hosts in the cluster.

5.7.3 VXLAN

As talked about in chapter 3.1.1.3, VXLAN allows for virtual machines to communicate with each other though the virtual networks. VXLAN is configured on all the hosts at the same time. Again, vDS was used for the VXLAN and MTU of 1600 was configured, considering the overheard which the encapsulation causes. Also, a new IP pool of 10.69.51.70-10.69.51.73 was created for the VXLAN. These are the VTEP addresses, that the NSX is going to assign to ESXi hosts after the configuration.

The configuration was successful, because the VXLAN status icon changed to “configured” and the VXLAN kernel interfaces appeared in the ESXi hosts. See (Figure 14.) for reference. However, for VXLAN to work properly, segment id, transport zones and a logical switch still must be configured.

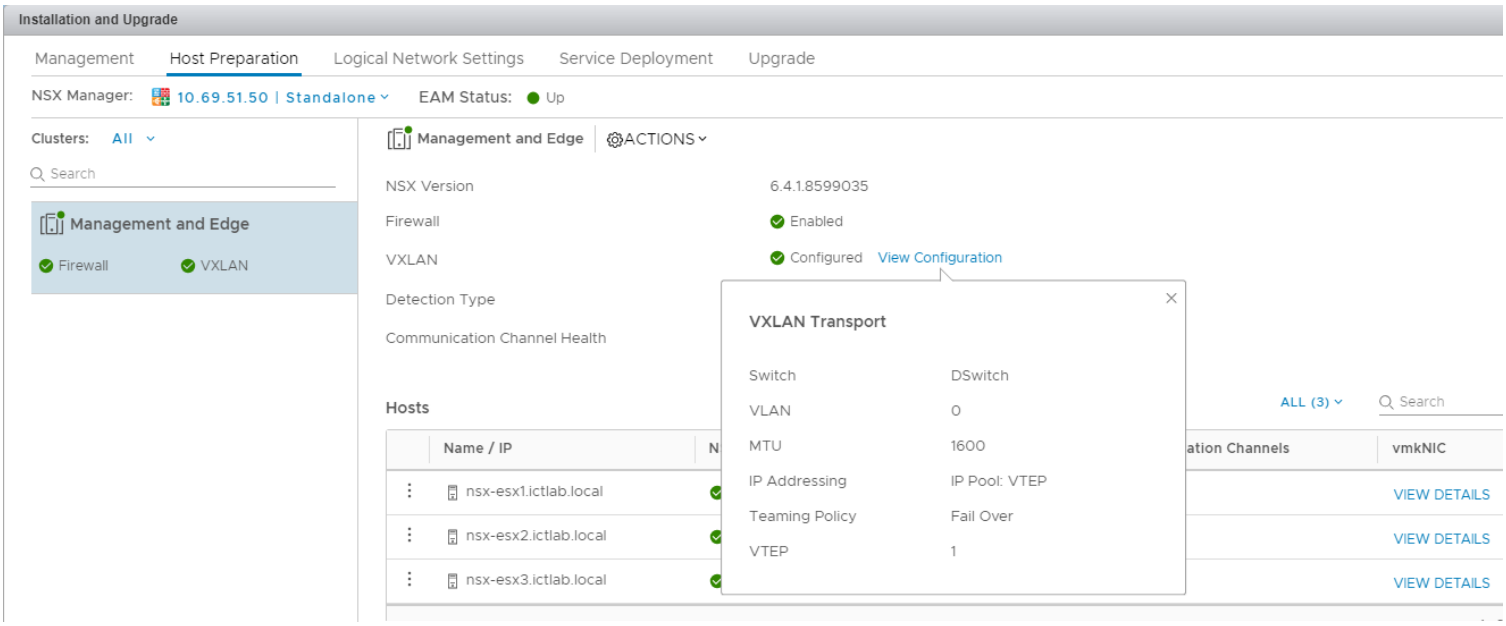

Figure 14. VXLAN configured

5.7.4 Segment ID and Transport Zones

Segment ID defines the maximum number of logical switches in the environment and to avoid confusion between VLANs, the range of Segment ID starts from 5000 to 16777216. A range of 5000-6000 was chosen for this small lab environment.

To control the width of VXLAN reach, a transport zone is created. All the logical switches created and assigned to the transport zone become available as distributed port groups on the vDS on every cluster in the transport zone.

The transport zone was named “Global” and the replication mode was set to “Unicast” to allow NSX controller to handle the VXLAN control plane. This was the best choice for this environment, as no special configuration is required. In example, the Hybrid mode offloads traffic to the physical network and requires specific underlay configuration.

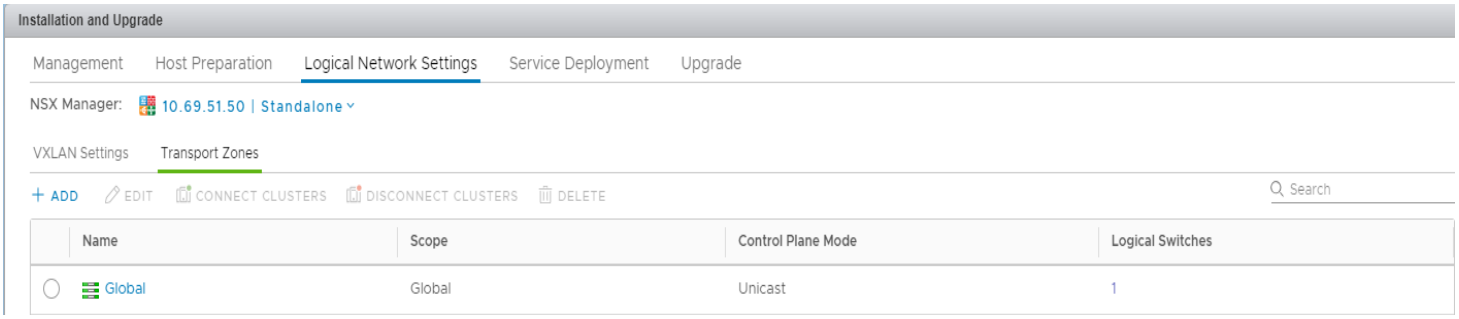

Figure 15. Transport Zone configured

5.7.5 Creating a logical switch

Now it was possible to create a logical switch to connect virtual machines together with the VXLAN encapsulation. The switch was named “Global”, unicast replication mode was chosen, and the switch was added to the transport zone which was created earlier. The port group vxw-dvs-22-virtualwire-1-sid-5000-Global was created in the vDS, meaning that the logical switch was created.

For the testing of the logical switch and VXLAN, two virtual machines were created, VM1 with IP of 172.16.51.10 and VM2 with 172.16.51.20. VM1 was running on nsx-esx1 and VM2 on nsx-esx2 and pinging between the two was not possible. However, after adding the virtual machines to the logical switch that was created, the ping went through, even when there was no subnet of 172.16.51.x created on the underlay. Next and the last step was to create a distributed logical router, that provides routing between two logical switches.

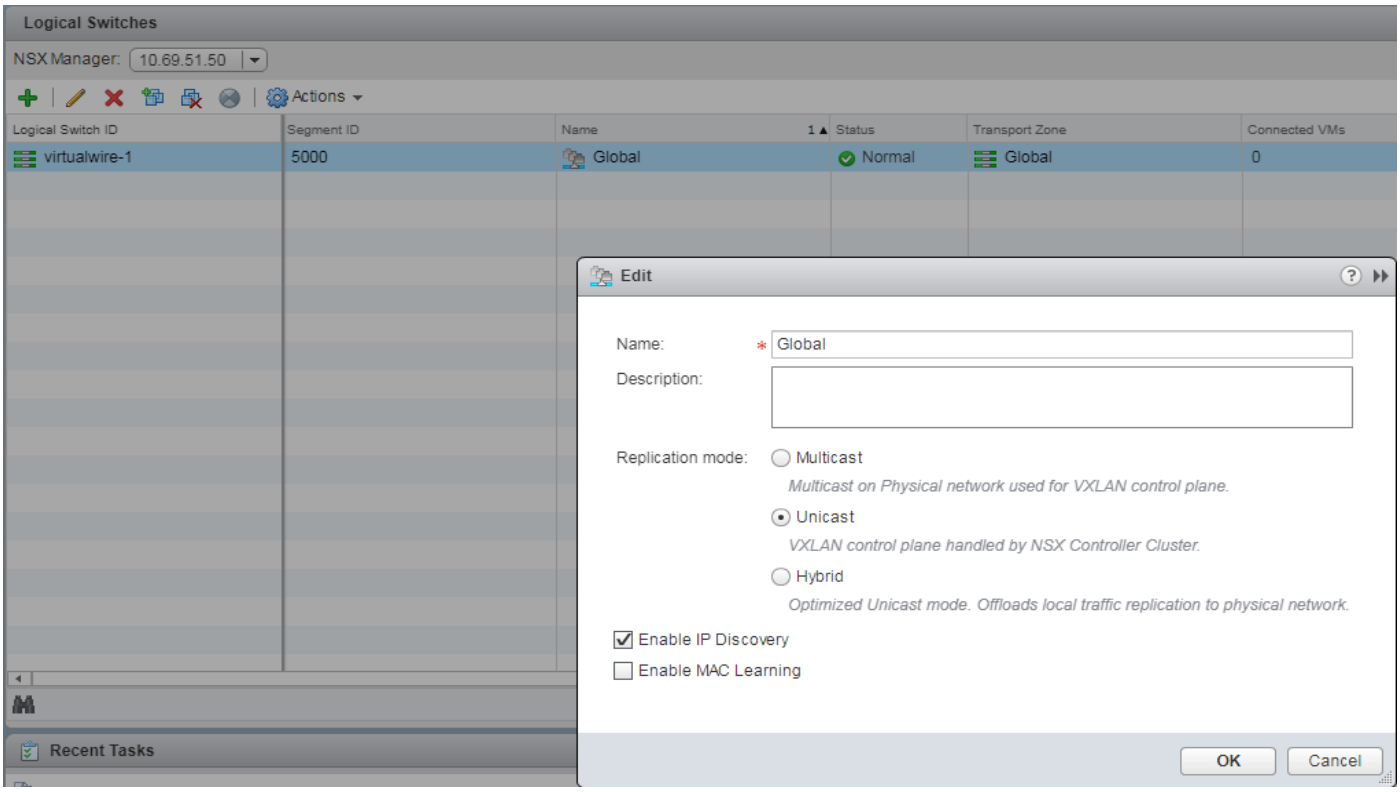

Figure 16. Logical switch created

5.7.6 Creating a distributed logical router

In a traditional vSphere network virtual machines communicating with other VM on different subnets must go through the physical adapter of the ESXi to a switch and to a physical router that provides the routing services before going back to the server and to the target VM. This kind of traffic flow is not optimal and is referred to as “hair pinning”. (Vinilth, 2016.)

DLR prevents the hair pinning with hypervisor level routing. Every ESXi hypervisor has a routing kernel module installed in it by NSX that performs routing between the logical interfaces defined on the DLR, which means the traffic never has to leave the virtual network. (Vinilth, 2016.)

First, one more logical switch named “Applikaatio” was created the same way as earlier and the “Global” switch was renamed to “Netti”. Also, one new virtual machine named “VM3” was deployed on nsx-esx3 with IP 172.16.52.30 and added to the switch “Applikaatio”. For a reference topology of the configuration, see Figure 18.

DLR is created from “NSX Edges” menu, with the possibility of creating ESG with the same installer. The DLR was named “DLR” with a hostname of DLR1. CLI credentials were defined, SSH access enabled and the Management and Edge cluster was chosen for the deployment target for the router.

Next the management interface for SSH connection was defined with an IP address of 10.69.51.120 /24 and the internal connections to the logical switches were configured as logical interfaces (LIF) in the DLR. The first interface was connected to “Netti” with an IP of 172.16.51.1 /24 and the second interface to “Applikaatio” with an IP of 172.16.52.1 /24. The configuration settings were then reviewed and the DLR deployed.

Now, ping tests between the virtual machines connected to two different switches were successful. This meant, that the routing was working as intended, still avoiding the physical underlay, and the environment was complete. NSX Edge was later added to see its features, but It was never implemented for real use. After the NSX licenses ran out, the lab environment was deleted and taken down.

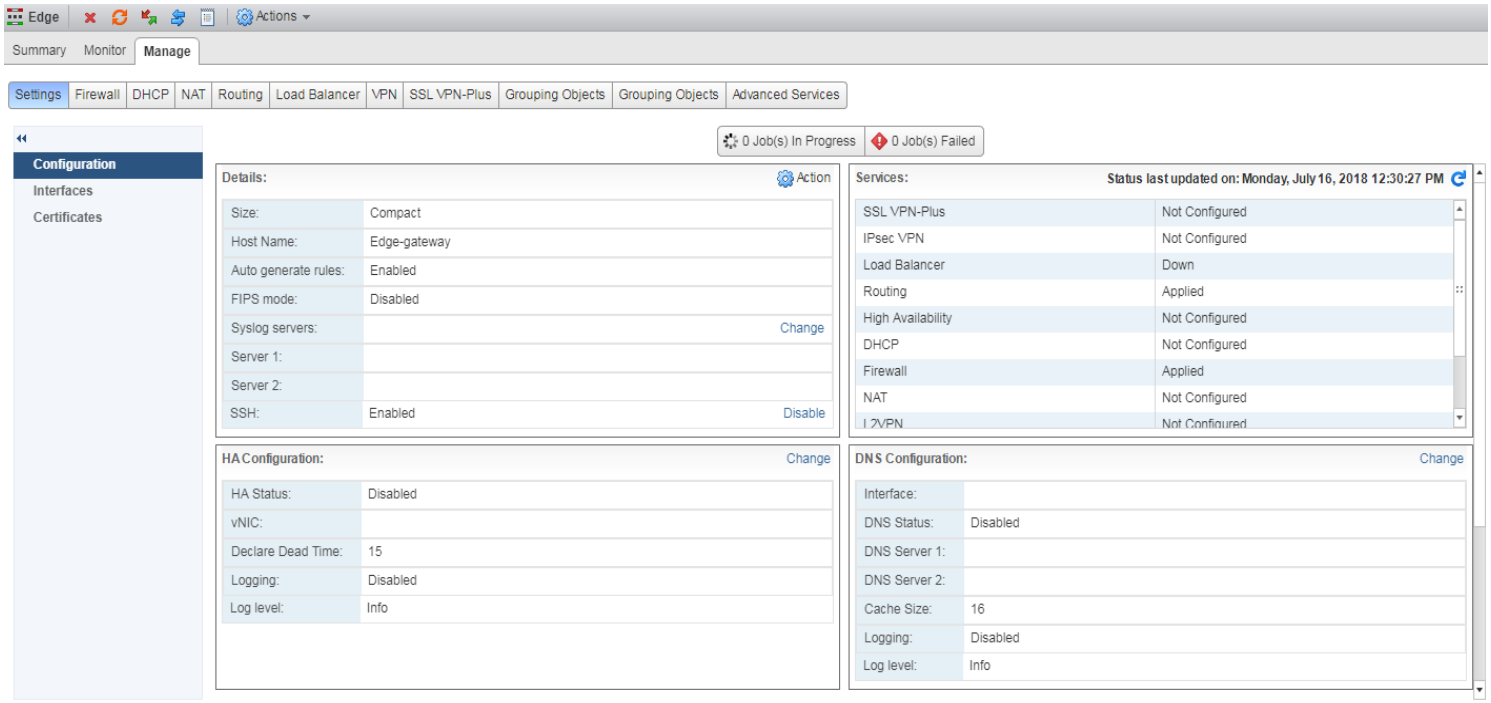

Figure 17. NSX ESG

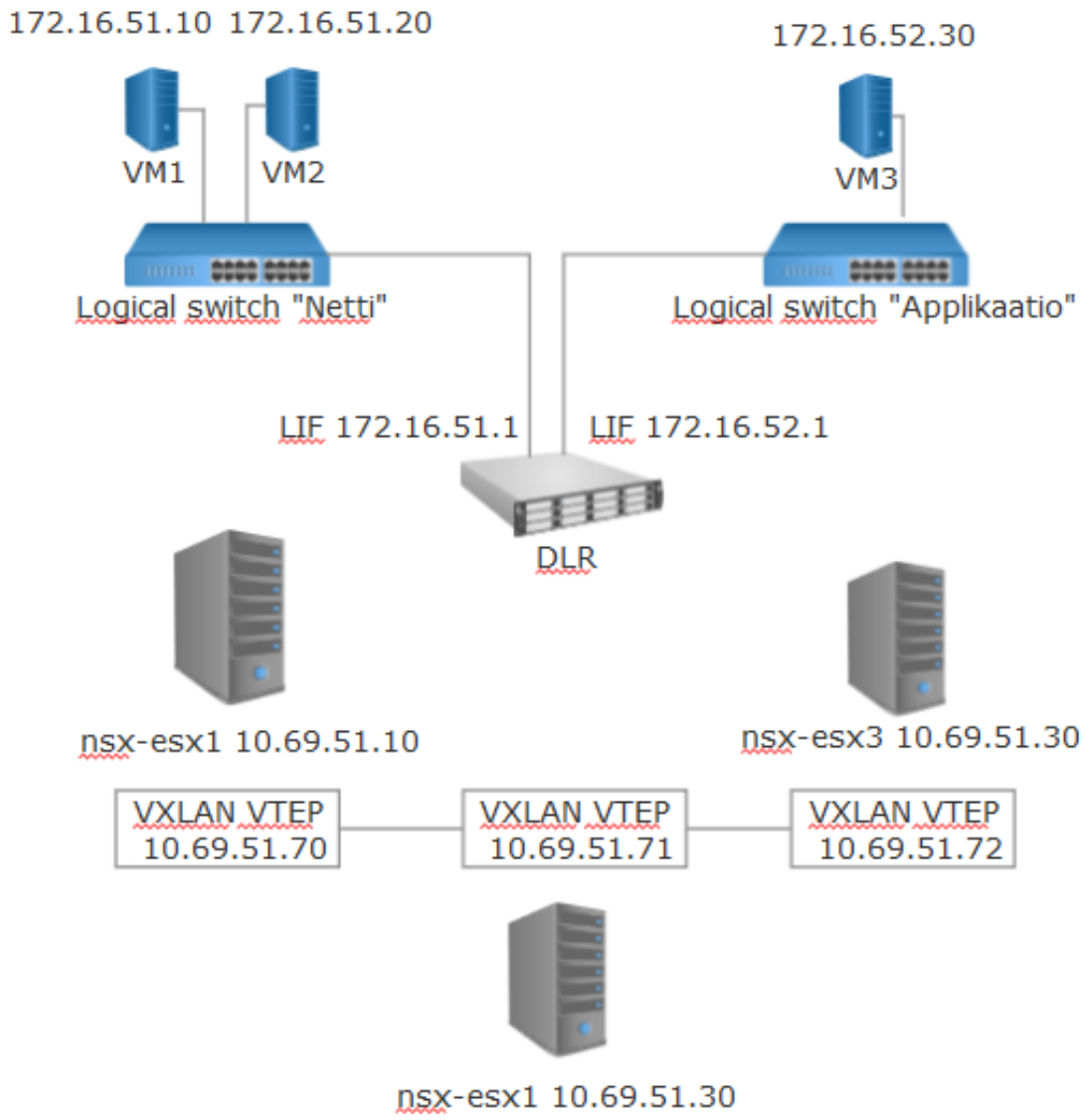

Figure 18. Topology

6 CONCLUSION AND FURTHER DEVELOPMENT

Software-defined networking is a technology that should be considered by every business in the IT field. Especially, service providers and other networking companies should pay attention to it. It is the future of networking, and more businesses adapt some form of SDN every passing year.

The objective of this thesis was to provide a better understanding of the SDN technology and how it can be used and what benefits it would give compared to traditional networks. Kymijoen ICT is currently not planning on transferring to SDN, but with this thesis they have much clearer understanding on how to do it and what they can achieve with it.

The objectives set for the thesis were successfully achieved and much was learned in the process of composing it. When Kymijoen ICT introduced the idea of a thesis about SDN, there was no information about the subject, and much research had to be done before the writing or the installations.

Even though the practical part of the thesis may seem like a simple step-by-step installation, many difficulties were faced during the installations, and every step required much examination before it was possible to continue further, especially on the last installations.

There are many ways, in which the study could be developed further, such as the security of SDN. There was only little reliable information about the security of software-defined networks, as most of the information was from forums and some video sources, and always concerned the same central vulnerability problem. It would be quite interesting to see which other vulnerabilities come with SDN, as it is a software, and software always has bugs which can possibly be exploited.

Another area of development would be continuing further with the NSX installations. The installations done in this thesis were a small part of what NSX can provide as the aim was to create an example of how the virtual routing and

switching is done and how the underlay is separated from the virtual network, since these are key elements in NSX.

VMware NSX is one of the most diverse SDN solutions on the market, and there are many ways to continue and expand the lab environment that was built, such as the many properties of NSX edge router. With it, larger more complex networks can be created with load balancing, firewall, routing protocols such as OSPF and BGP. Microsegmentation is also a feature of NSX that would be interesting to see when implemented to use.

SDN is such a vast concept, that there are countless of ways to study and develop it further. Software-defined networking was an interesting technology to study and the possibilities of what it can do are astonishing. It has clear benefits compared to traditional networks, and as the technology keeps evolving further, new possibilities are revealed each year. There may come a time, when the networks around the world are completely virtualized with no physical networking hardware used.

TABLE OF FIGURES

REFERENCES

André, R. 2014. Programmable networks: Separating the hype and the reality. Online.
Available at:
https://datablast.wordpress.com/tag/sdn/ André, R. 2014
(Accessed 16 October 2018)

Arora, H. 2017. Software Defined Networking (SDN) explained for beginners. Online.
Available at:
https://www.howtoforge.com/tutorial/software-defined-networking-sdn-explained-for-beginners/
(Accessed 12 October 2018)

Deuren, J. 2017. VMware NSX: Secure and Flexible Network Virtualization. Online.
Available at:
https://securelink.be/blog/nsx-vmware/
(Accessed 27 November 2018)

Gerard, S. 2017. VXLAN – Overlay protocol. Online.
Available at:
https://securelink.be/blog/vxlan-overlay-protocol/
(Accessed 30 November 2018)

Hardcastle, J. 2017. How Customers Use VMware's NSX (Hint: It's Not Just a Security Use Case). Online.
Available at:
https://www.SDxCentral.com/articles/news/customers-use-vmwares-nsx-hint-security-use-case/2017/07/
(Accessed 2 December 2018)

Juniper Networks. 2014. Understanding Network Virtualization with VMware NSX. Online.
Available at:
https://www.juniper.net/documentation/en_US/release-independent/nce/topics/concept/metafabric-2.0-vmware-nsx.html
(Accessed 1 December 2018)

Pujolle, G. 2015. Software Networks: Virtualization, SDN, 5G and Security. John Wiley & Sons, Incorporated. 262s.

Raza, M. 2018. What is Software Defined Networking? SDN Explained. Online.
Available at:
https://www.bmc.com/blogs/software-defined-networking/
(Accessed 15 October 2018)

Rouse, M. 2012. SDN controller (software-defined networking controller). Online.

Available at:
https://searchsdn.techtarget.com/definition/SDN-controller-software-defined-networking-controller
(Accessed 15 October 2018)

Rajeev, S. 2017. NSX Logical Switch. Online.
Available at:
http://www.rajeevsrikant.com/nsx-logical-switches/
(Accessed 30 November 2018)

SDxCentral. s.a. a. What are SDN Southbound APIs? Online.
Available at:
https://www.SDxCentral.com/sdn/definitions/southbound-interface-api/
(Accessed 17 October 2018)

SDxCentral. s.a. b. What Is Cisco Application Centric Infrastructure (or Cisco ACI or Cisco SDN)? Online.
Available at:
https://www.SDxCentral.com/cisco/datacenter/definitions/what-is-cisco-aci/
(Accessed 2 December 2018)

SDxCentral. 2015. Featured Videos: Customers Discuss Their Cisco ACI Use Cases. Online
Available at:
https://www.SDxCentral.com/articles/featured/cisco-aci-use-cases-videos/2015/03/
(Accessed 2 December 2018)

Statista. 2018. Software-defined networking (SDN) market size worldwide from 2013 to 2021 (in billion U.S. dollars). Online.
Available at:
https://www.statista.com/statistics/468636/global-sdn-market-size/
(Accessed 4 December 2018)

Vinilth. 2016. Hair-pinning Solved with VMware NSX DLR. Online.
Available at:
http://virtualize-automate.com/wp/index.php/2016/03/17/hair-pinning-solved-with-vmware-nsx-dlr/
(Accessed 3 December 2018)

VMware. s.a. NSX Edge. Online.
Available at:
https://pubs.vmware.com/NSX-6/index.jsp?topic=%2Fcom.vmware.nsx.admin.doc%2FGUID-3F96DECE-33FB-43EE-88D7-124A730830A4.html
(Accessed 1 December 2018)

Wang, J. 2016. What is Cisco ACI Fabric. Online.
Available at:
https://www.speaknetworks.com/what-is-cisco-aci-fabric/
(Accessed 2 December 2018)

Hautamäki, J. 2015. Kehittämistutkimusta ja ongelmanratkaisua YAMK-opinnäytetöissä. (Online)
Available at:
https://centriaamk.wordpress.com/2015/12/18/kehittamistutkimusta-ja-ongelmanratkaisua-yamk-opinnaytetoissa/
(Accessed 17 December 2018)

Kalintsev, D. 2015. NSX for vSphere: VXLAN Control Plane modes explained. Online.
Available at:
https://telecomoccasionally.wordpress.com/2015/01/11/nsx-for-vsphere-vxlan-control-plane-modes-explained/
(Accessed 17 December 2018)