Henri Kalliosaari

Deploying NFV Services with NFX250

Helsinki Metropolia University of Applied Sciences

Master’s Degree

Information Technology

Master’s Thesis

17 May 2017

Author Title Number of Pages Date	Henri Kalliosaari Deploying NFV Services with NFX 49 pages + 0 appendices 17 May 2017
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Ville Jääskeläinen, Principal Lecturer

This Master's Thesis is about deploying virtual network functions with a NFV platform. The Network Function Virtualization (NFV) is a very interesting and relevant topic in the current era of virtualization. The idea behind NFV is to virtualize network functions that usually have their own physical appliances and run them from a single device or a pool of devices such as servers. This kind of virtualized deployment reduces CAPEX, power and space.

The goal of this thesis was to research and deploy a virtual network model to a branch network and see how well it could follow current the guidelines relevant to operational usage. In the thesis requirements and current network functions were mapped. The NFV model was compared in theory against the requirements. Network services were migrated to use the virtual network function architecture and a completely new service was deployed as a VNF to the branch network. Examples of this configuration are shown in the study.

The NFV model was proven to meet the requirements derived from operational guidelines. All requirements required from the previous physical network model were met with appropriate and acceptable accuracy. The delivery of new services took a major leap forward with the virtualized model and the delivery time was reduced to one day. It is expected to be reduced even more in the future.

The NFV model is the foundation and the first step towards a world of automated service delivery and orchestration. It blurs the borders between the two traditional working groups - servers and the networking side.

Keywords	NFV, VNF, NFX250, Virtualization

Contents

Acronyms

CAPEX	Capital Expenditure
EOL	End-Of-Life
ESP	Encapsulating Security Payload
ETSI	European Telecommunications Standards Institute
FTP	File Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IPsec	IP Security Architecture
KVM	Kernel-based Virtual Machine
MPLS	Multiprotocol Label Switching
MSP	Managed Service Provider
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NIC	Network Interface Card
PCI-SIG	Peripheral Component Interconnect Special Interest Group
SCP	Secure Copy

SDWAN	Software-Defined Wide Area Network
SDN	Software-Defined Networking
SFC	Service Function Chaining
SNMP	Simple Network Management Protocol
SR-IOV	Single Root Input/Output Virtualization
SSH	Secure Shell
TLS	Transport Layer Security
VNF	Virtual Network Function
VPN	Virtual Private Network
VM	Virtual Machine

1 Introduction

Service provider networks are currently populated with a large number of physical appliances. For customers who have their network services provided by MSP (Managed Service Provider), one medium-sized enterprise site can hold numerous devices used for different kind of services. To deploy a new network service it is often required to physically ship a completely new device. Finding room and power to hold these physical appliances is becoming more and more difficult. When this is added to the pool of increased energy costs, capital investment issues and rarity of network design skills, integration and operation skills, the need to reduce the number of physical appliances is well justified. Physical appliances also suffer heavily from the EOL (End of Life) effect as the current pace of IT innovation is accelerating more and more. [1]

Network functions virtualization (NFV) is a network architecture concept that uses several IT technologies to virtualize different physical network appliance functions. These virtualized functions can then be brought anywhere and everywhere in the network. By default, all network functions can be considered virtualizable. NFV still differs from standard server virtualization although it has similar attributes. Network functions that could be virtualized are for example firewalls, routers, load balancers, OSI-layer 4-7 security devices, WLAN controllers and WAN accelerators. [2] SDWAN (Software-Defined Wide Area Network) is a great example of a network function that could be deployed using VNFs. VNF (Virtual Network Function) is the virtual version of a traditional physical network function like a router or a firewall. VNF can accomplish the same network function as its physical counterpart but has the benefits of virtualized appliance.

While software-defined networking SDN and NFV are very close together, they are not the same thing and NFV is not dependant on SDN. They coexist and support each other. Figure 1 displays the relation that SDN and NFV have. Important thing about NFV is that it can be achieved using non-SDN mechanisms. This means that a company can for example follow their current server virtualisation path, research NFV and possibly adopt it without the need of having SDN deployed in their network making the threshold for studying NFV and migrating to it is more feasible. NFV could bring several benefits for daily operations and could be very helpful to many enterprises in delivering new services.

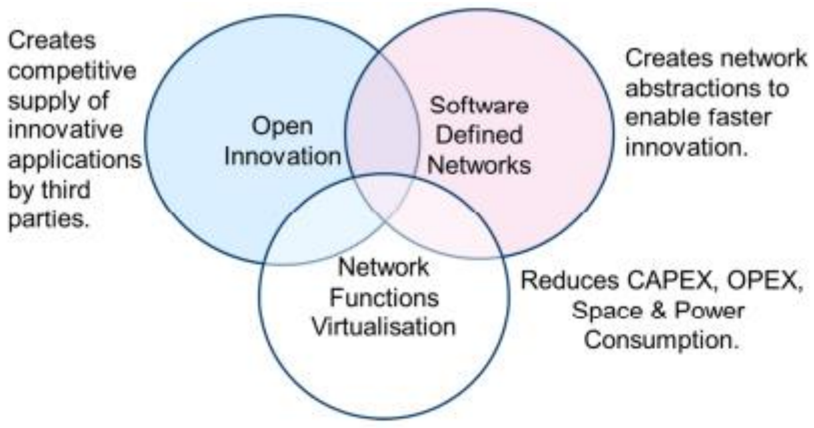

Creates competitive supply of innovative applications by third parties.
Open Innovation
Software Defined Networks
Creates network abstractions to enable faster innovation.
Network Functions Virtualisation
Reduces CAPEX, OPEX, Space & Power Consumption.

Figure 1. NFV relationship with SDN. [1]

The NFX Network Services Platform is a physical appliance from Juniper Networks that introduces virtualization to an enterprise site. It eliminates the need for multiple devices located on enterprise premises and enables robust service delivery opportunities. It acts as a physical host for different VNFs. Having the possibility to deliver certain services to a site almost immediately is a huge asset when compared to the old deployment model that needs new equipment and almost always a technician for installation.

This thesis studies and evaluates migrating existing physical network functions to virtual network functions and how a virtual approach complied with the common guidelines defined in enterprise network policy. It also examines the delivery of new virtual network function services to a branch network. Physical appliance chosen for the present study was the NFX network services platform, a product of Juniper Networks.

The current state of an example branch network was analyzed and the different network functions in use were mapped. Requirements for the NFX service platform and VNFs were imported from already in-use common guidelines of operational usage. These guideline subjects are management, visibility/monitoring and ease-of-deployment for new services. Current state analysis ended with conclusions made based on the operational requirements.

1.1 Target of Study

This thesis describes the reasoning of a new virtualized approach. It explains in detail the service chain that can be created using virtualization and also explains how the pre-defined requirements can be met in theory. Solutions on management, visibility and ease-of-deployment for new services are described in detail.

The research question of the study can be posed as follows:

“Can the Virtual Network Function model meet the common operational guidelines of branch network service deployment and management?”

1.2 Scope of Study

The study shows an example of virtualizing certain common network functions and evaluates briefly the virtualization capabilities of NFX network virtualization platform and common VNFs. The thesis relies heavily on practical approach for researching the topic.

The scope of the study is restricted to the following common branch site network functions:

1. Routing
2. Firewall
3. SDWAN service
4. Local services server

Virtualization of HQ network functions such as datacenter services are not included in the thesis. WAN acceleration is a branch network function that is also excluded from the thesis as it can be very similar to a SDWAN service.

Part of the thesis was implementing the new virtualized network functions approach to an example customer branch site. The VNF services were deployed one after another and the network model was compared against operational usage guidelines and how well it could comply with these guidelines. New virtual network functions were deployed to a branch network. This deployment and network model was reviewed on the last part of the thesis. NFV as an architecture and technology could be seen as successful if the predefined guidelines could be met and the overall service delivery could be made more agile.

The thesis is structured as follows:

Network Function Virtualization as per ETSI standard is covered in Chapter 2, the Service Chaining and different types of it are gone through in Chapter 3. The branch network, its operational guidelines and current network functions are explained in Chapter 4 and the VNF approach, reasoning and meeting the requirements is covered in Chapter 5. Chapter 6 introduces the documentation of the deployment of VNFs using NFV platform and Chapter 7 includes the discussions and conclusions.

2 Network Functions Virtualization

This section opens up the concept of Network Functions Virtualization as defined by ETSI (European Telecommunications Standards Institute). It also describes the benefits, challenges, current use-cases and future vision for NFV setups.

2.1 Definition of Network Functions Virtualization

Network Functions Virtualization aims to transform the current network architectures from using physical dedicated appliances more towards virtualized components. Virtualized components would then be run on industry standard high volume servers, switches and storages. Servers, switches and storages could be located in different places like data centers, network nodes or even end user premises. [1]

The idea is to virtualize the network functions in a manner that network function services are easy to deploy and available everywhere – without the need to install new equipment on premises. [1]

Figure 2 illustrates the VNF implementation idea of European Telecommunications Standards Institute (ETSI). Network functions are run over the Network Function Virtualization Infrastructure (NFVI). [3]

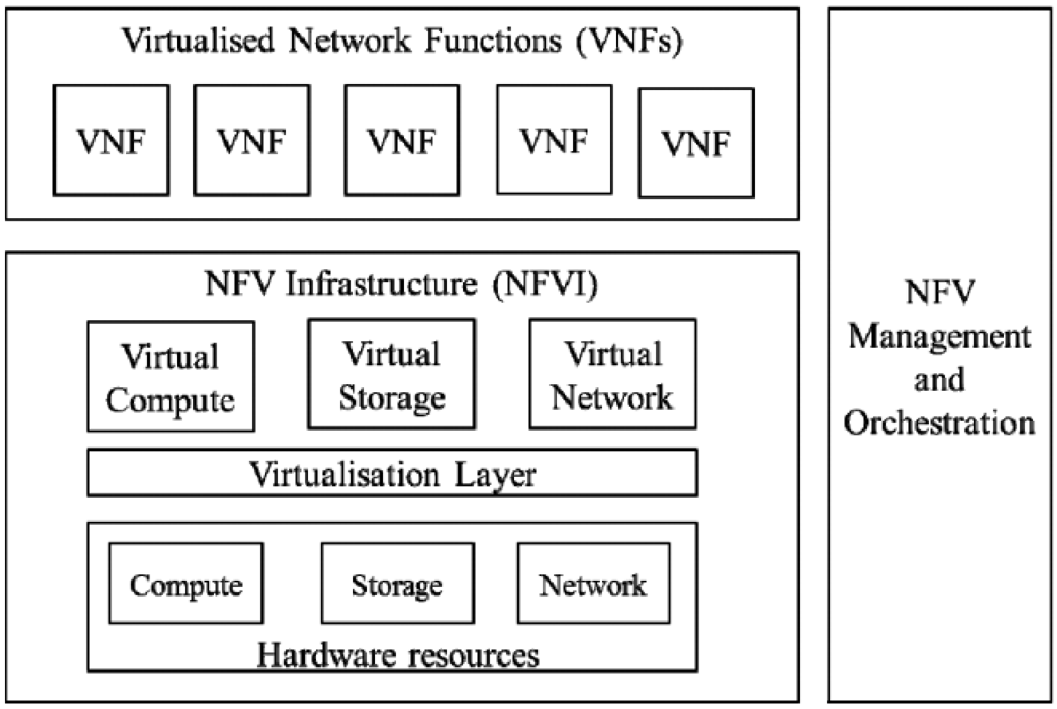


Figure 2. High-level diagram of NFV framework. [3]

The main working domains of NFVs are:

1. VNF, a software implementation of a network function. VNF is capable of running on the NFVI.

2. NFVI, which includes the physical and virtualized resources. NFVI is the platform where VNFs are deployed to.

3. NFV Management and Orchestration, which is used to govern the lifecycle of VNFs and control the physical and virtual resources.

2.2 Benefits of Network Functions Virtualization

ETSI Network Functions Virtualization white paper lists a number of benefits that using NFV will bring to the telecommunications industry. These benefits are the following:

- Reduced equipment costs and power consumption.
- Increased velocity of Time to Market
- Possibility to run production, test and reference facilities using the same equipment.
- Targeted service introduction
- Optimization of network configuration and topology

Because of deploying network functions more and more on the same physical devices, adding network functions to the network will no longer create such a big leap in power consumption and deployment of new network services will also be more affordable. A new service will not come with a new physical device.

Bringing a new product to the market will be much easier because investments in hardware-based functionalities are no longer needed – rather it is the software that is developed. This should reduce significantly the maturation cycle for network operators.

Virtualized components also mean that the physical hardware can be shared to run production, testing and DevOps facilities. This will also help with migrations since a virtualized test environment is run basically on same hardware as the production environment. This leads to less variables during deployment of new features. This method of deployment can reduce administrative and development costs.

Virtualization also helps with targeted service introduction based on geography or customer set – services can be scaled up or down when required and the service deployment velocity is improved. Optimisation of network configuration and service location is possible based on real-time traffic and mobility patterns. Services can be brought closer to the mass of customers in public events for example - automated.

2.3 Challenges of Network Functions Virtualisation

ETSI Network Function Virtualization white paper lists some of the challenges on implementing Network Functions Virtualisation. These challenges should be addressed by the community in hopes of accelerating the NFV progress and making it wide-spread.

The main challenges from the ETSI NFV white paper are the following:

- Portability/Interoperability between different vendors.
- Performance trade-off due to virtualization.
- Co-existence with legacy platforms - Compatibility and migration.
- Security & Resilience of VNFs and hypervisors.
- Network Stability
- Simplicity and integration

One of the challenges seen is defining a unified interface which decouples the software instances from hardware underneath. This can be thought of as virtual machines and hypervisor. The ability to run virtual appliances on industry-standard system makes them more portable and interoperable.

With the use of industry-standard systems, no proprietary hardware should be used. This causes performance degradation and it has to be taken into account as virtualization itself causes processing overhead already. The available output of the underlying hardware has to be visible, so that virtual machines know the maximum output they can get from the hardware. Authors of the ETSI NFV white paper also mention that they believe choosing the right technology will allow virtualization of the data/user plane also, instead of just the network control functions.

Transition from legacy architectures to newer ones are never immediate. This is true with network function virtualization even more than ever before. One of the biggest challenges is the co-existence with and migration from legacy setups and equipment. VNFs have to be manageable with existing management and orchestration systems that are used for legacy network functions. Network function virtualization architecture must also support a migration path from the legacy physical appliances to the more open standards based virtual machine solutions.

Virtualization creates more opportunities for cyber attacks. Virtual appliance is run on a hypervisor which can also have vulnerabilities and they can be abused. These vulnerabilities in some situations can for example grant access to the attacker even though the virtual appliance itself would be secure enough. Virtual appliance can be as secure as a physical device if the infrastructure, hypervisor and its configuration is secured properly.

Network stability should be taken into consideration and impact of virtual function relocation or re-configuration should be very minimal or even nonexistent. The management and orchestration of large number of virtual appliances should be vendor independent and should not affect the network stability. All mechanisms increasing network stability will be beneficial to the progress of network functions virtualization.

Simplicity is one challenge. One needs to make sure that the virtualized network platforms are simpler to operate when compared to the currently existing physical network platforms. Automation plays also a key role on simplicity as it can make the day-to-day operational usage easier. Integration should also be seamless and simple – physical platform should be compatible with different hypervisors and virtual appliances. [1]

3 Service Chaining

Service chaining or service function chaining (SFC) means the capability to connect network services in a chain and this way deploy different network services to the traffic flow. Examples of these network services can be firewalls, Internet breakout, Intrustion Preventation System (IPS) / Intrusion Detection System (IDS) etc. [4]

Service chaining in a virtualized environment is faster than using physical appliances and regular truck roll deployments.[2] One can deploy firewall service with internet breakout to the branch site and if that branch site needs MPLS (Multiprotocol Label Switching) capability for a new WAN connection, MPLS router can also be deployed to the virtualized environment. It can then be connected to the firewall in the virtualized network. No physical connections are needed for the interconnections between the firewall and MPLS router.

Figure 3 displays an example of a small branch site that has a firewall service deployed in a virtualized network environment.

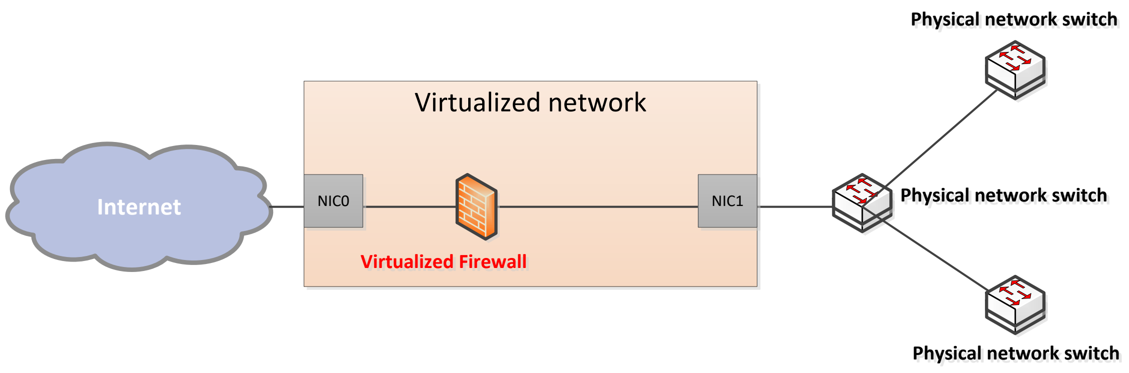


Figure 3: Example network with virtualized firewall service

NIC0 and NIC1 in Figure 3 represent physical ports provided by the network services platform. NIC1 is connected to the physical network switches that provide connection to the users.

Figure 4 shows the logical idea of adding a new component to the virtualized network.

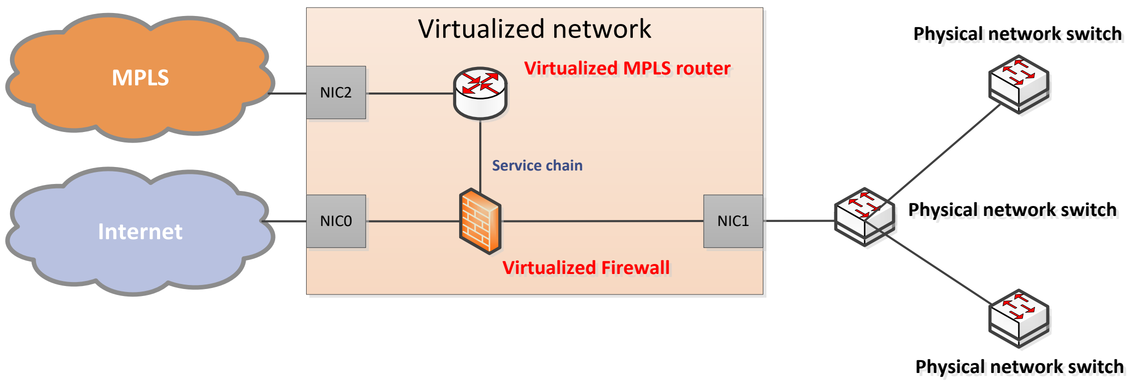


Figure 4: MPLS router service is added to the virtualized network

NIC2 in Figure 4 represents a physical port that has been allocated to the virtualized MPLS router on the network services platform. Connecting the MPLS router to the MPLS cloud is the only physical connection needed in this example. Service chaining provides an easy way of bringing a MPLS router to the network and existing services without having to deploy a physical appliance and connecting it via multiple physical cables. Local hands-and-eyes support is not needed for deployment as much as before.

Figure 5 displays the deployment of a virtualized IDS service.

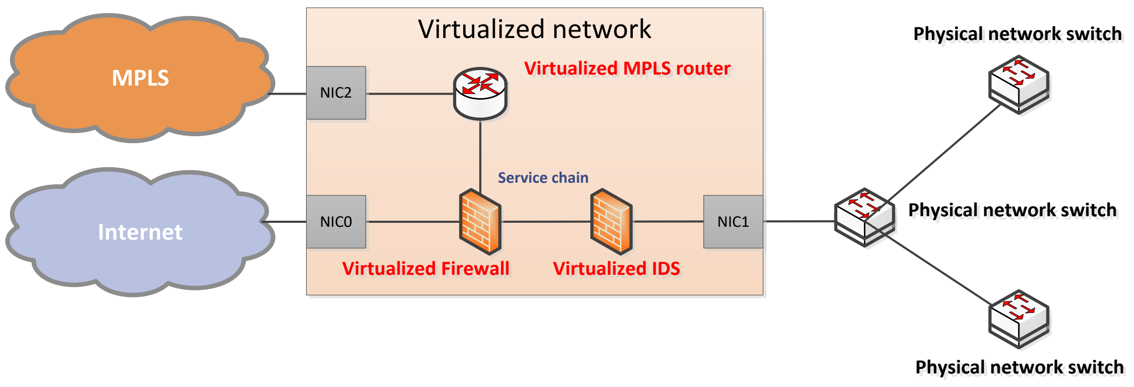


Figure 5. A new VNF is added to the branch site network

The IDS service can be deployed completely without any physical procedures. The deployment should not cause any service disruption either.

3.1 Service Chaining with SR-IOV

SR-IOV or Single Root Input/Output Virtualization is a standard developed by the Peripheral Component Interconnect Special Interest Group (PCI-SIG) for virtualized servers. Founded in 1992, PCI-SIG is an electronics industry group advancing the non-proprietary Peripheral Component Interconnect (PCI) technology. Tasks mentioned in PCI-SIG homepage:

- Defining PCI specifications to deliver required I/O functionality
- Adapting PCI technology to future applications
- Maintaining backward compatibility with previous specifications
- Supporting industry-wide product development by offering compliance and interoperability test support

SR-IOV introduces two function types which are:

- Physical Functions (PFs)
  - PCIe (Peripheral Component Interconnect Express) functions that support the SR-IOV Extended Capability. This is used for configuration and management of the SR-IOV functionality. [5]
- Virtual Functions (VFs)
  - PCIe (Peripheral Component Interconnect Express) functions that only contain the resources necessary for moving data, but have minimized set of configuration resources. [5]

SR-IOV defines a method to share PF (Physical function) of the I/O port without software emulation. The process creates a number of VFs (Virtual Functions) per physical port of the I/O device. A virtual function is then directly assigned to a virtual machine increasing the performance to almost that of a native physical device. [6] Figure 6 illustrates an example of SR-IOV.

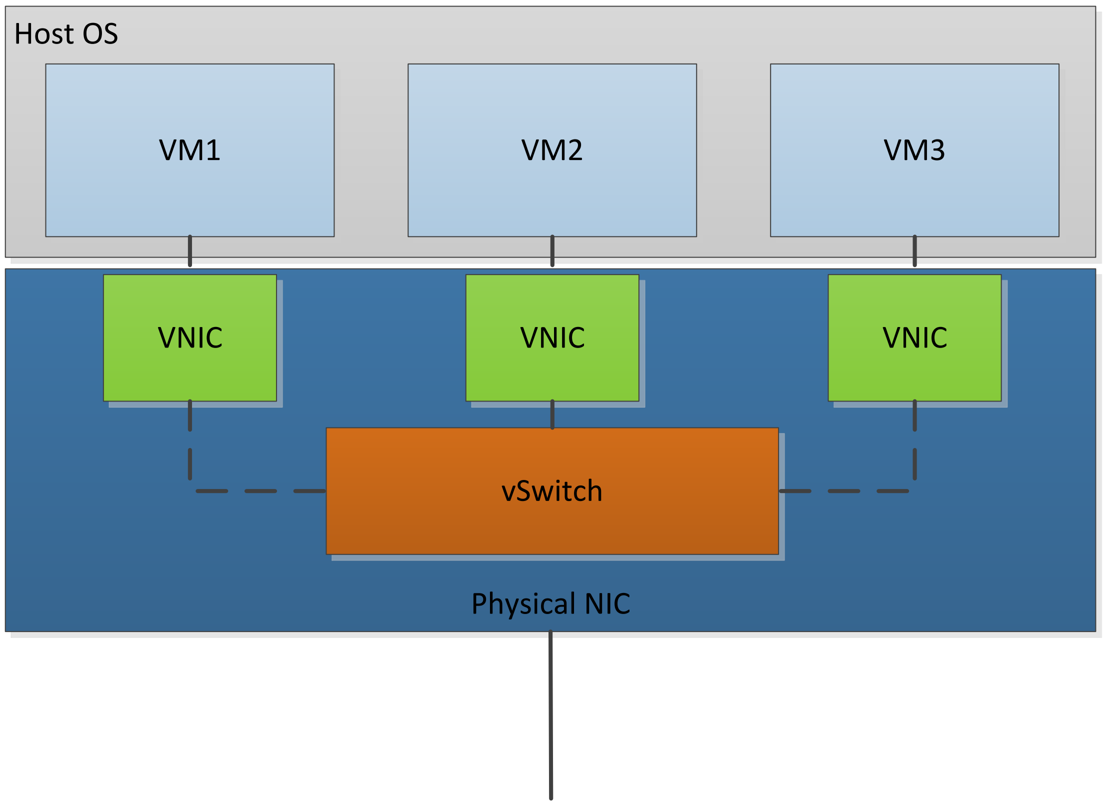


Figure 6. Example of SR-IOV

When using SR-IOV, a single physical card can be partitioned into 16 different sections per physical port. These sections match to the virtual functions at higher layers. Figure 6 illustrates three VMs each having their own VFs. Communication between these virtual functions is handled by using a bridge. SR-IOV also includes a set of methods for different operations for managing and instructing the SR-IOV NIC switch. [7]

3.2 Service Chaining Using Virtio

Virtio belongs to the standard Linux libvirt library that holds useful virtualization functions and is very common in most versions of Linux – also any Linux-run device can practically use virtio. Virtio is the software-only approach for VNF communication and bridging. [7]

Virtio enables the virtual machines to connect to simple internal bridges or virtual switches. (see Figure 7). Internal bridges can be of several different types and can link many virtualized internal NICs together by using a virtualized switch function in the host OS itself. These internal bridges can then be connected to an external bridge for outside physical connections. [7]

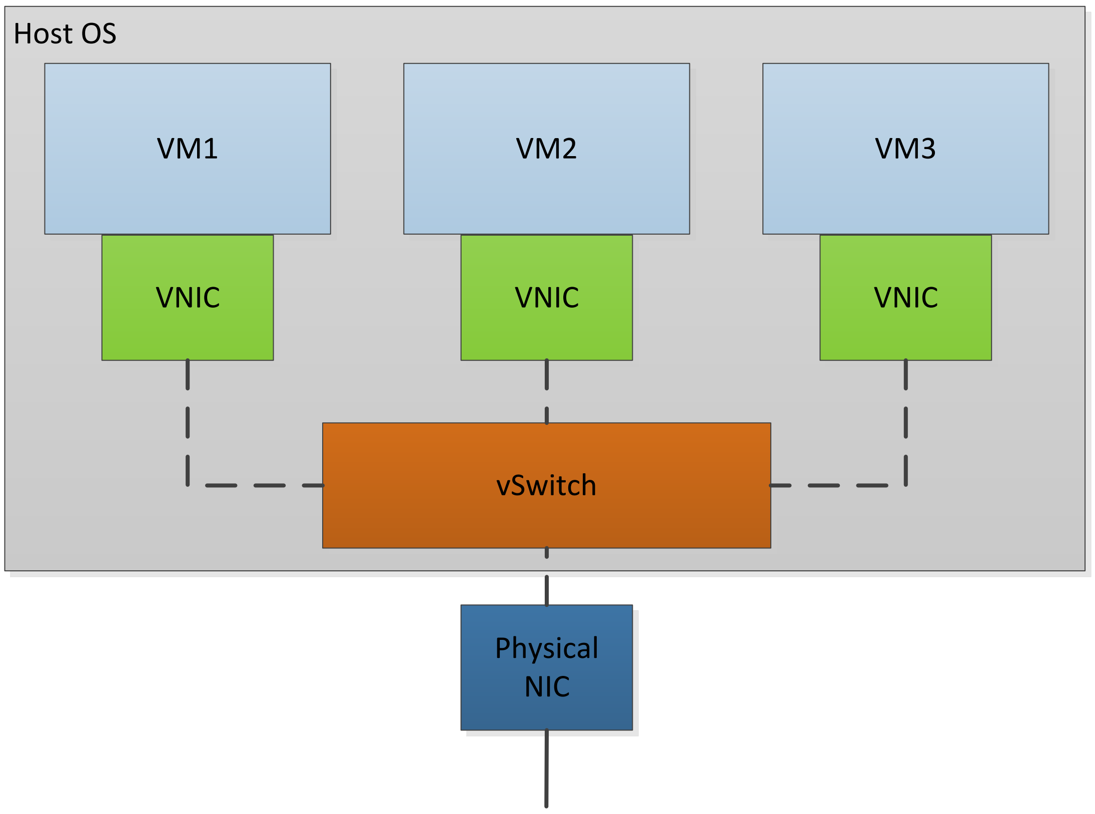


Figure 7: VM connectivity with an internal switch

Main differences between SR-IOV and virtio are the deployment model and support for them. Generally, virtio is supported in all Linux versions and is fairly simple and quick to use. While SR-IOV does provide lower latency and CPU usage, it is only supported by certain NIC hardware and platforms. [7]

4 Current Status of Branch Network

This chapter explains the topology of the current branch network case and the guidelines followed in daily operational usage. Local network functions are also mapped and explained. Furthermore, the chapter defines the requirements for the NFV model – successfulness can be defined as the ability to meet operational guidelines.

Branch network case is derived from a typical customer network topology. Customer has requested researching the subject of VNFs and the possibility to deploy a VNF architecture model to their branch network.

Branch network consists of two routers, firewall, switch and a local services server, which are presented in the logical topology below (Figure 8):

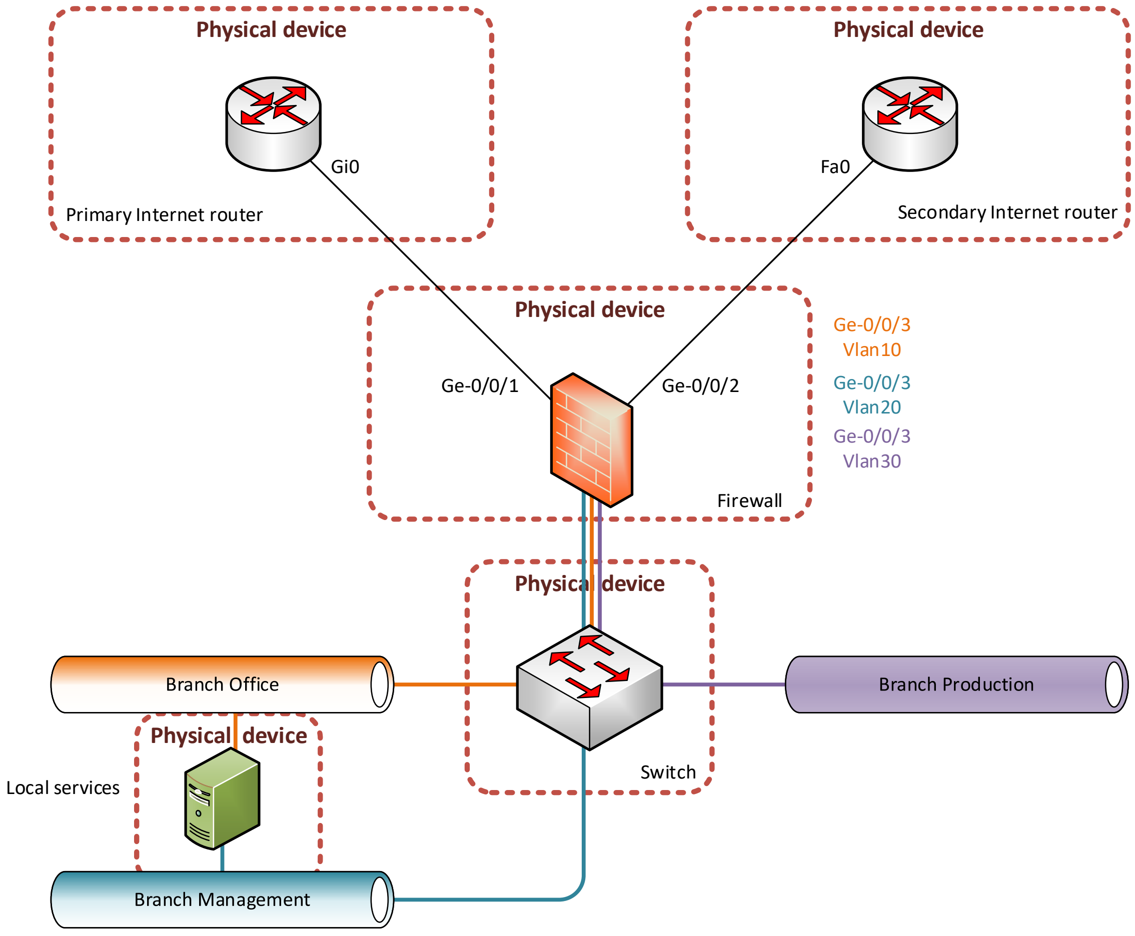


Figure 8: Current network topology with five different physical devices

Each physical appliance seen in Figure 8 has a network function of its own. One can list these network functions separately and see what they are:

I. Primary Internet router is used for primary Internet line and connectivity provided is by the firewall to create the primary VPN connection.

II. Secondary Internet router provides backup Internet connectivity and the backup VPN tunnel is created by using this connection.

III. Firewall has multiple network functions. It is the default gateway for the three different networks (Office, Production and Management). It also functions as a VPN gateway – it creates the primary and backup VPN connection to the Headquarters firewall. It also defines the network segments and enforces security policy between these segments.

IV. Network function of the switch is to provide layer 2 connectivity. This is for the end-users and also for the wireless access points. End-users and certain network devices have two ways to connect to the network – wired and wireless.

V. Local services server provides DHCP and backup for certain DC services like NTP and DNS.

The state of the branch network is viable but not optimal. A branch network has 30-50 end-users depending on the weekday and how many work from home. Network services are redundant enough for a small office like this one. The absence of physical redundancy is a known risk as the capital expenditure is a concern.

Branch network is geographically 60km away from the company headquarters. The headquarters is the main location of internal network services. Only some small services such as DHCP and backup NTP/DNS are run locally. The internal connectivity can be seen in Figure 9.

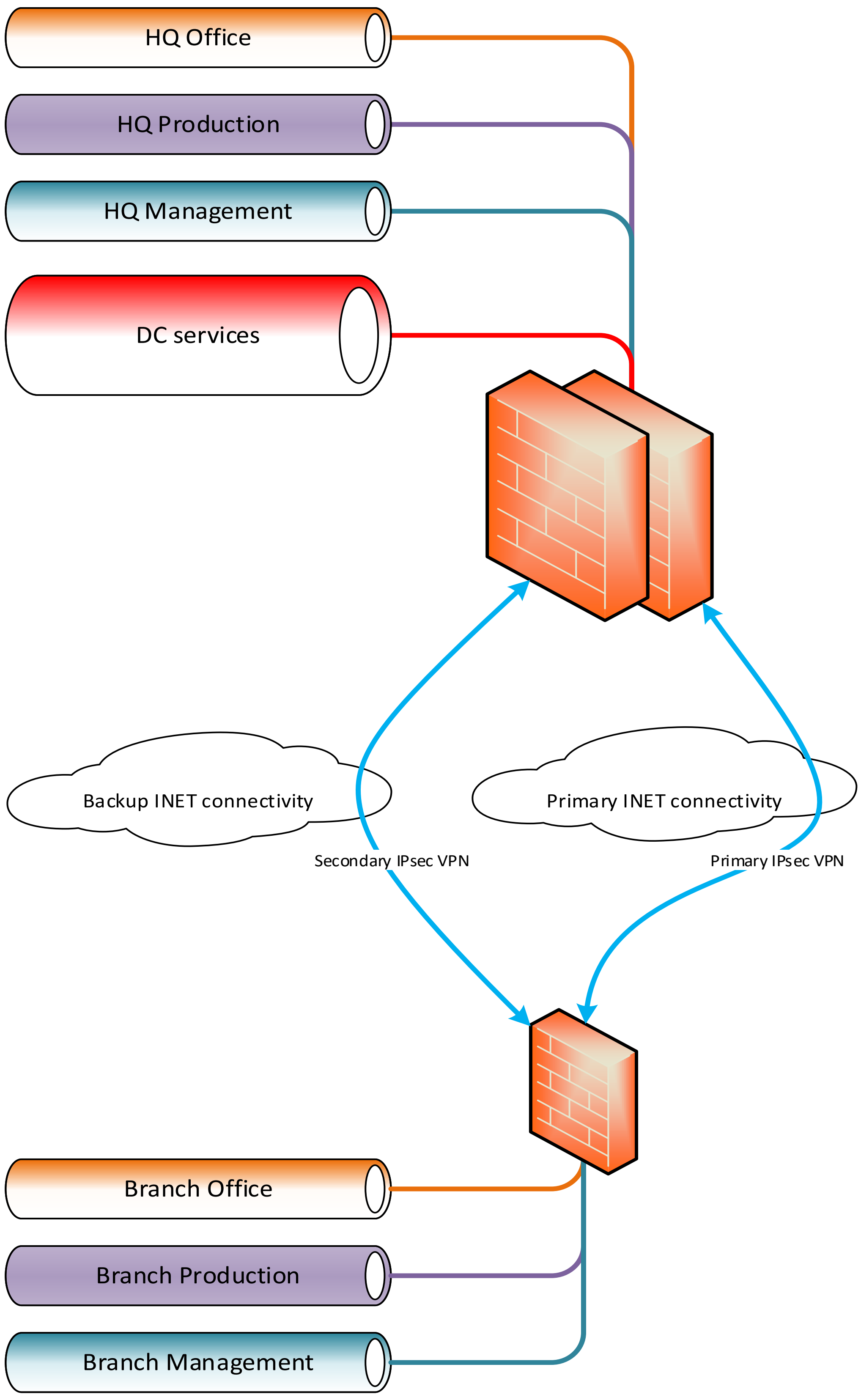


Figure 9: Internal connectivity over IPsec VPNs

While clearly it would be possible to group up many of the network functions to one of the devices (Firewall for example), it is important to understand that just changing the network topology is not the goal of the case study in this thesis. There might be situations where that is not acceptable due to security regulations or for example the firewall device itself cannot produce a certain network service/application that is needed.

Requirements in minimum can be brought from the already in-use guidelines for operations regarding customer network.

4.1 Management Guidelines for Branch Network

Operational guidelines regarding management are simple and are covered in customer-related operational documentation. Keypoints from the documentation can be used to devise the following list:

I. Devices should be managed using secure communication methods. This means SSH and HTTPS/TLS. All other management traffic that might be needed (such as FTP) will be transported inside ESP packets (IPsec)

II. Backups from the devices should be taken daily

III. Managing devices happens inside the Management-network.

4.2 Visibility Guidelines for Network

Idea behind visibility of the branch network events is to notice device errors and interruptions ideally before a network outage or disruption happens. Example can be a malfunction fan in a router. If this is noticed in time, there is a high possibility of avoiding an outage.

I. Device health is monitored using ICMP and SNMP. Note that SNMP doesn’t have to be version 3, monitoring is done over IPsec transport.

II. Network traffic is monitored using SNMP

4.3 Deployment Guidelines for New Services

Currently the deployment guidelines are not well-defined as the branch network environments are very static and don’t really have new deployable services.

If new services are deployed, physical appliances are required for that. They are shipped to the location after being configured in the operator premises. There the local contact for IT installations will connect the devices as instructed and power on the physical appliance. Device for the new service is normally installed to a small rack located in an office closet or a similar room. In a small office, device noise is an important contributing factor to the physical deployment location.

If the deployment of a new service can cause network outages to the branch network, it is normally done before or after office hours. After the deployment, checks have to be made to make sure that the old network services still work.

5 Virtual Network Function Approach

In the previous chapter the requirements for the current physical branch network architecture were listed and explained. Chapter 5 will show how well the virtual network architecture can meet the given requirements and open up the reasoning behind VNF approach.

5.1 Reasons for NFV

The example branch network topology (Figure 8) has been used in many different locations by the customer. Physical devices have been found to be a solid choice for the networks. The networks have very rarely been impacted with an outage caused by a physical device breakdown. It is much more often that a software bug or a software error happens, that causes the network function to fall apart and cause a disruption in service.

The ability to choose what vendor software technology one wants to use for routing or firewall service is one of the core reason to look into virtual network function deployment. The same goes for hardware though currently it doesn’t hold as high value as the software side on decision making.

The idea of deploying NFV services using standalone devices is a beginning of a larger technology renewal plan. One can portray a flow chart that represents this plan at a high-level, see Figure 10.

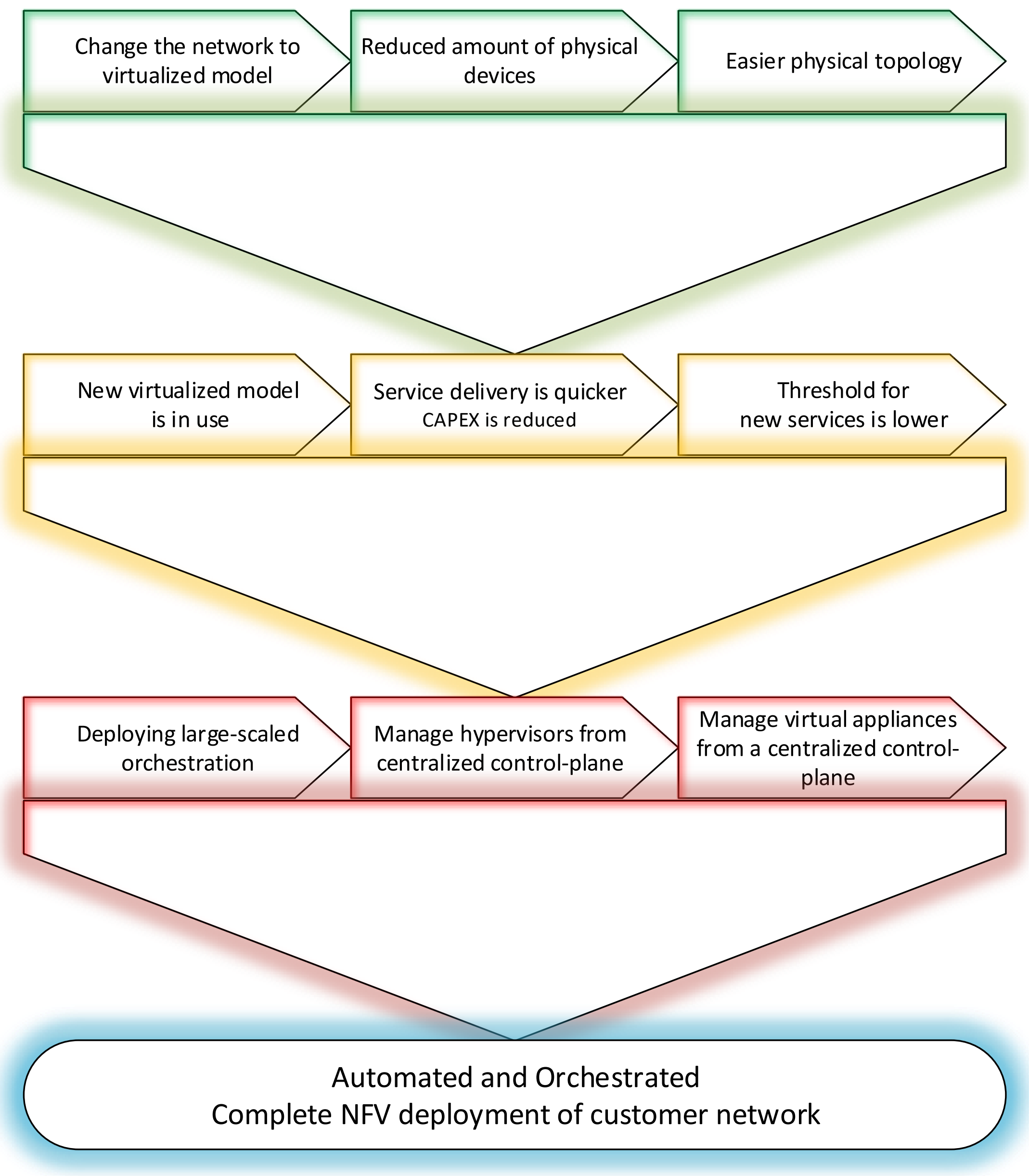

Change the network to virtualized model
Reduced amount of physical devices
Easier physical topology
New virtualized model is in use
Service delivery is quicker
CAPEX is reduced
Threshold for new services is lower
Deploying large-scaled orchestration
Manage hypervisors from centralized control-plane
Manage virtual appliances from a centralized control-plane
Automated and Orchestrated
Complete NFV deployment of customer network

Figure 10: Flowchart of the technology renewal plan

As mentioned previously, this case study is focused on the first green step – migrating the current network topology to a virtual architecture. Every step will have its benefits. These waypoints help in the long run, because getting to the end goal is easier – new results can be seen in all steps of the flowchart.

5.2 Meeting Requirements

The preliminary case study work is started by going through the previously created network function list and defining how the network diagram would look when virtualized.

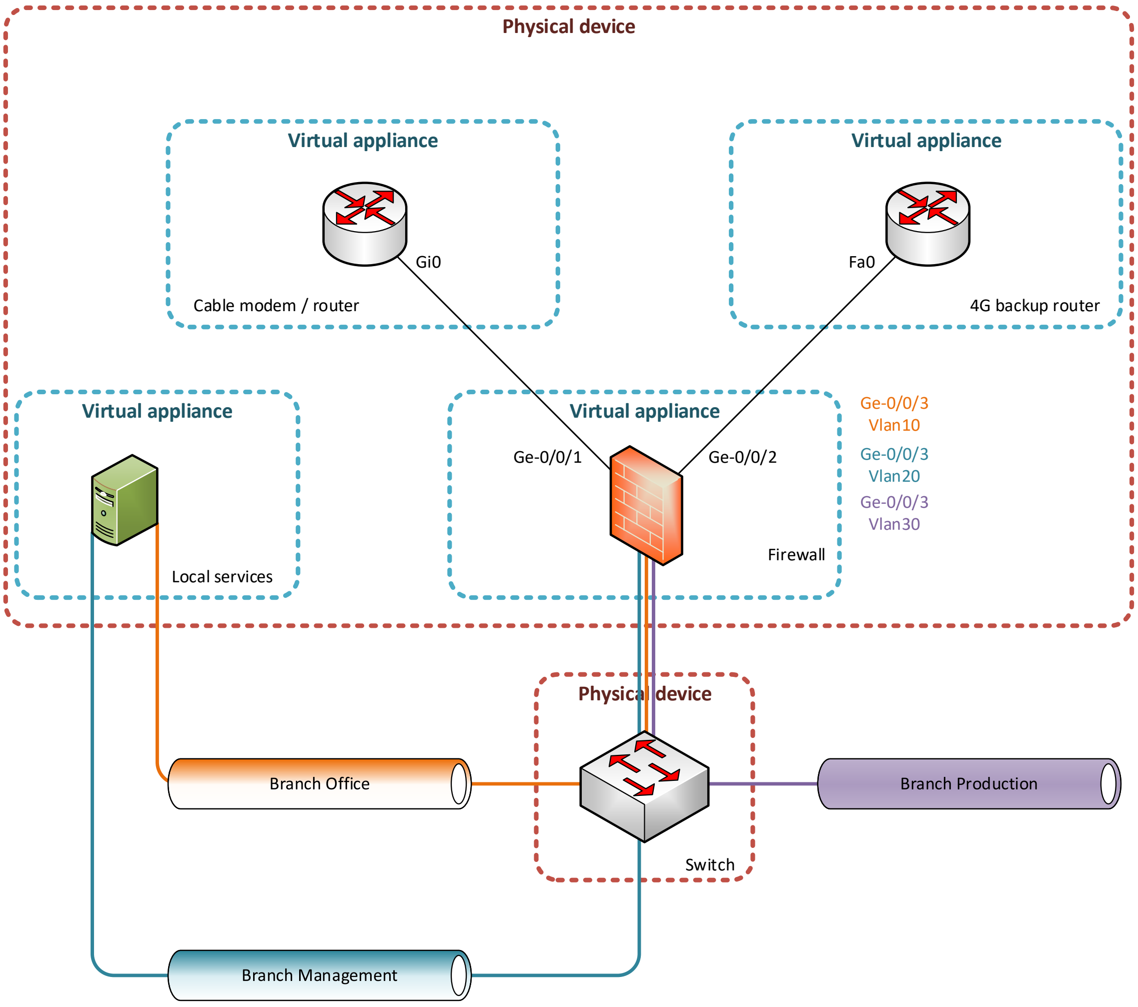


Figure 11: Current network topology virtualized

As presented in Figure 11, everything else is virtualized on paper except the physical switch. The branch network has 30-50 end-users and while most of them use wireless connectivity, there must be a possibility for wired connection also.

Like physical devices, the virtual appliances also can be managed using the same secure methods like SSH and HTTPS/TLS. All other management traffic that is necessary has been forwarded inside a IPsec VPN tunnel. The NFV platform that has been selected for this study supports securing the management traffic using IPsec VPN. IPsec VPN tunnel is created with a firewall container that uses the management Ethernet port.

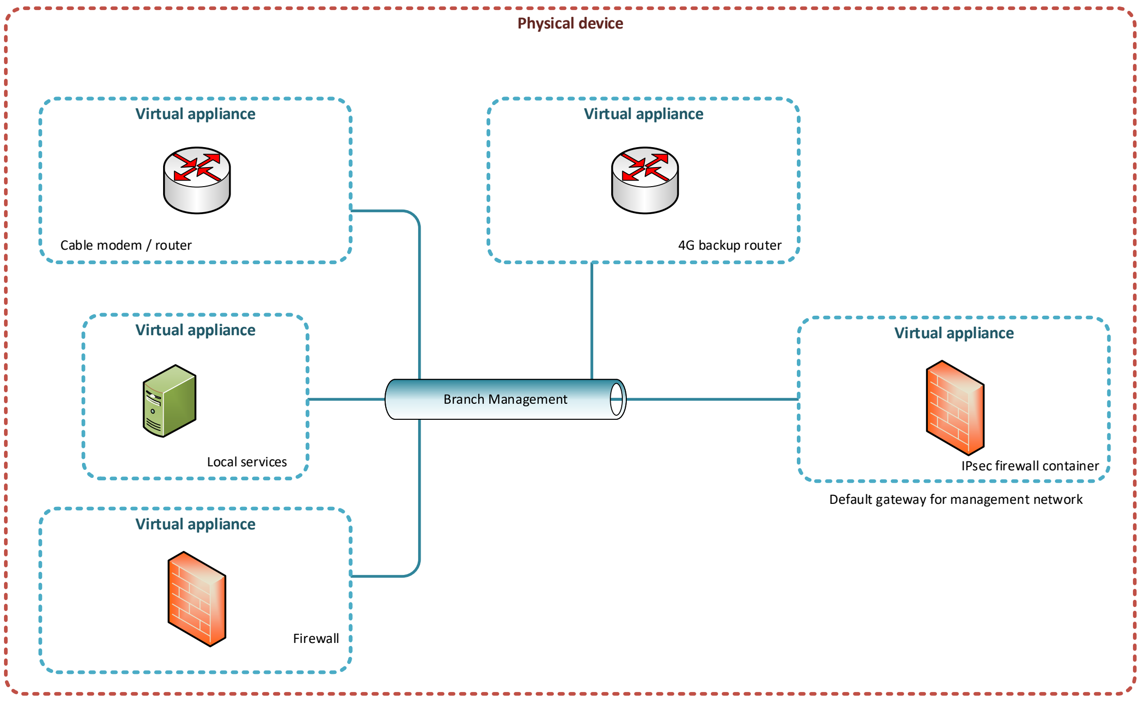


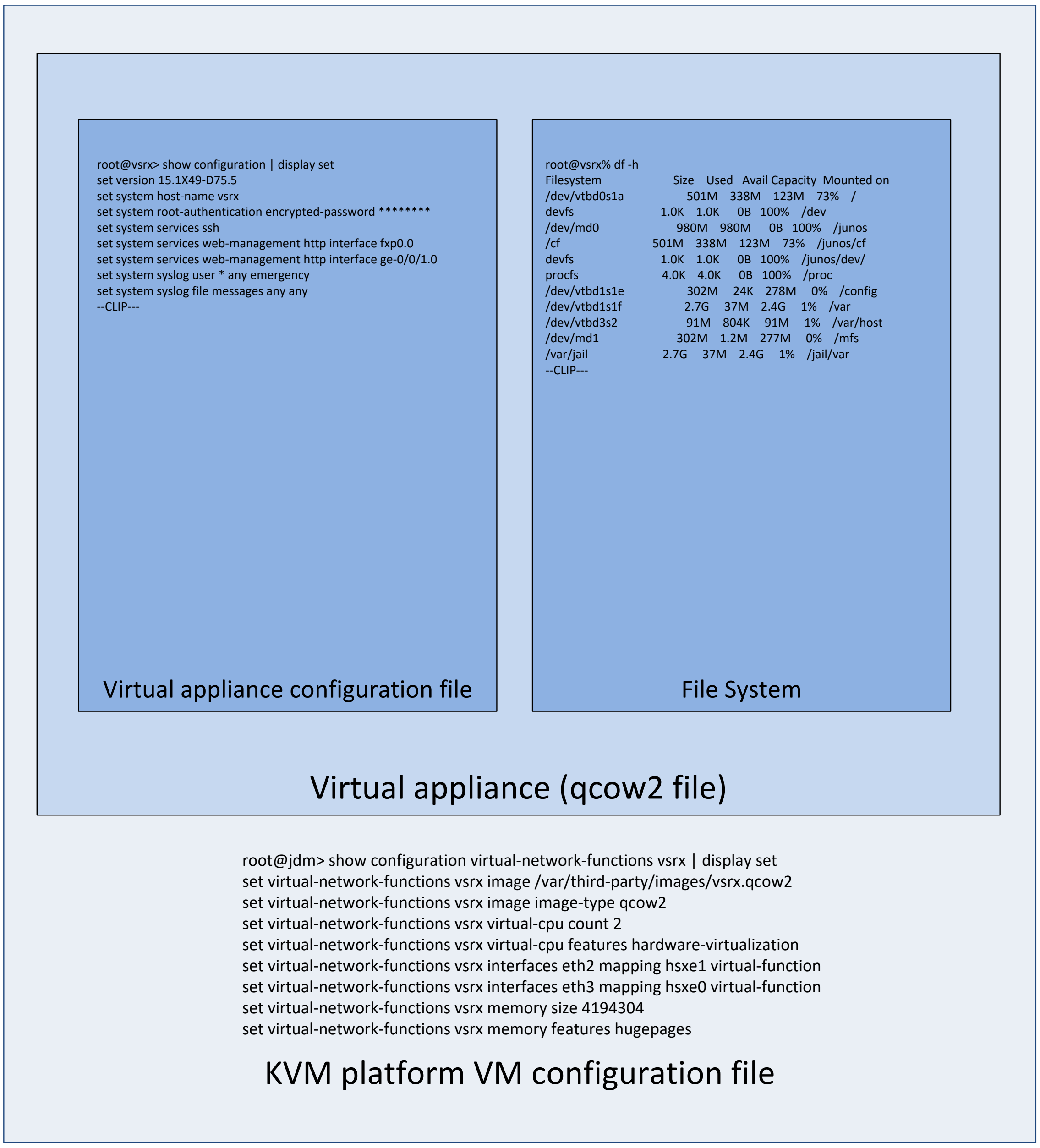
Figure 12: L3 network diagram of management network

Figure 12 displays the L3 management network. It is important to understand that the default gateway for this network is not the normal production firewall, but the IPsec firewall container natively deployed in the NFX platform. Management of all devices at the branch site can be done using the management network – Routers and firewalls do not need to be managed using possible public IP addresses.

A backup of a virtual appliance can be taken with two different methods. The first method is to create a backup file of the VM's configuration file. This is similar to a backup of a physical appliance. If a physical network function appliance breaks down, the configuration file is normally copied to another similar physical appliance. This works also for a virtual appliance.

Another way of taking a backup in this KVM (Kernel-based Virtual Machine) environment is to copy the VM file itself. This file type is usually qcow2. As the virtual appliance gets edited and the configuration file inside changes, all the changes happen to the qcow2 file. This qcow2 is the executed VM appliance with the edited configuration. A backup configuration file of the VM that was mentioned as the first method is not needed.

These are the methods of taking a backup for a VM, but it is not enough itself for a complete deployment. The KVM hypervisor side has to have a backup taken also. If this hasn't been taken into consideration, the internal service chainining and virtual machine system configuration might not be valid. Hypervisor connects the Virtual NICs to physical interfaces. Figure 13 shows the configuration backup levels and their idea.



```
root@vsrx> show configuration | display set
set version 15.1X49-D75.5
set system host-name vsrx
set system root-authentication encrypted-password ********
set system services ssh
set system services web-management http interface fxp0.0
set system services web-management http interface ge-0/0/1.0
set system syslog user * any emergency
set system syslog file messages any any
--CLIP---
```

```
root@vsrx% df -h
Filesystem          Size   Used   Avail Capacity  Mounted on
/dev/vtbd0s1a        501M   338M   123M   73%   /
devfs               1.0K   1.0K     0B  100%   /dev
/dev/md0             980M   980M     0B  100%   /junos
/cf                 501M   338M   123M   73%   /junos/cf
devfs               1.0K   1.0K     0B  100%   /junos/dev/
procfs              4.0K   4.0K     0B  100%   /proc
/dev/vtbd1s1e        302M    24K   278M    0%   /config
/dev/vtbd1s1f        2.7G    37M   2.4G    1%   /var
/dev/vtbd3s2          91M   804K    91M    1%   /var/host
/dev/md1             302M   1.2M   277M    0%   /mfs
/var/jail           2.7G    37M   2.4G    1%   /jail/var
--CLIP---
```

```
root@jdm> show configuration virtual-network-functions vsrx | display set
set virtual-network-functions vsrx image /var/third-party/images/vsrx.qcow2
set virtual-network-functions vsrx image image-type qcow2
set virtual-network-functions vsrx virtual-cpu count 2
set virtual-network-functions vsrx virtual-cpu features hardware-virtualization
set virtual-network-functions vsrx interfaces eth2 mapping hsxe1 virtual-function
set virtual-network-functions vsrx interfaces eth3 mapping hsxe0 virtual-function
set virtual-network-functions vsrx memory size 4194304
set virtual-network-functions vsrx memory features hugepages
```

Figure 13: Configuration file examples and their levels.

Take note of the syntax (Figure 13), which is the same for JDM (Juniper CLI abstract for virsh) and vSRX (Juniper virtual firewall appliance). Both use the Junos CLI model.

Physical appliances have been monitored using ICMP and SNMP management tools. The exactly same tools can be used to manage virtual appliances. In addition to this, one also needs to monitor the NFV platform health. It is critical to know what is happening in the hypervisor itself and whether there are any necessary informational messages or critical error notifications. Monitoring the NFV platform health is supported.

As previously mentioned in chapter 3, NFV can shorten the deployment schedule greatly. This is due to the fact that the local hand-and-eyes work is minimal – cabling is usually not needed and no new devices need to be installed to racks. In theory, the steps necessary to deploy a new service to a branch network are:

1. Transfer the VNF image to the site NFV platform. Prefer Secure Copy as the transfer protocol for this.

2. Deploy the image with the necessary service chain and VM specifications

3. Add the deployed VM to management and monitoring systems

6 Deploying Services with NFV Platform

In this part the virtual components of the NFV platform and their functionality were explained briefly. The migration from physical network functions to virtual ones was done and the virtual component configuration steps were explained. A completely new service for the network was deployed virtually to the migrated network. The outcome shows how the NFV architecture could meet the operational guidelines and enhance service delivery experience with reduced deployment time.

6.1 Juniper NFX250 NFV Platform

For this study, the chosen device to be tested was Juniper NFX250 as the NFV platform for the concept of virtual network function. The exact model is NFX250-S2. Specifications for the model are:

- CPU: Intel 6 Core Xeon D
- Memory: 32GB DDR4 RAM
- Storage: 400GB SSD
- Maximum number of VNFs: 8

There are two mandatory VMs/Containers and one optional container in the NFX250 platform. These components are the Juniper Device Manager (JDM), Juniper Control Plane (JCP) and the optional IPsec virtual network function (IPsec-NM). The mandatory VMs were selected as the core components for this study.

Juniper Device Manager (JDM) is the root container in the NFX250 platform (See Figure 14).

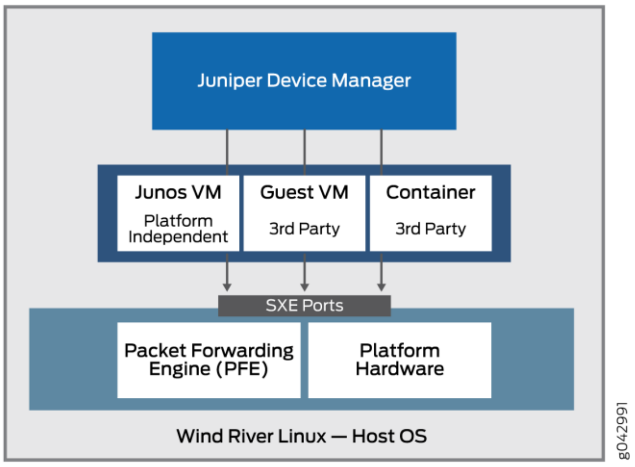


Figure 14: Position of the Juniper Device Manager. [7]

JDM is the Junos OS CLI abstract – one of JDMs tasks is to make the hardware look like a normal Junos OS-based physical system. It also prevents modifications and activities on the device from impacting the host OS, which in NFX250 is the Wind River Linux.

With JDM one can:

1. Deploy and manage VNFs, native or 3rd party.

2. Create service chains for VNFs.

3. Control the system inventory and resources.

The console and management port are linked to the JDM. Next Chapter includes more specific instructions of JDM.

Juniper Control Plane (JCP) is the core component in configuring the physical data plane of the NFX250. With JCP one can configure the NFX250 physical ports to correct VLANs and allow the traffic to move through the physical layer (Figure 15):

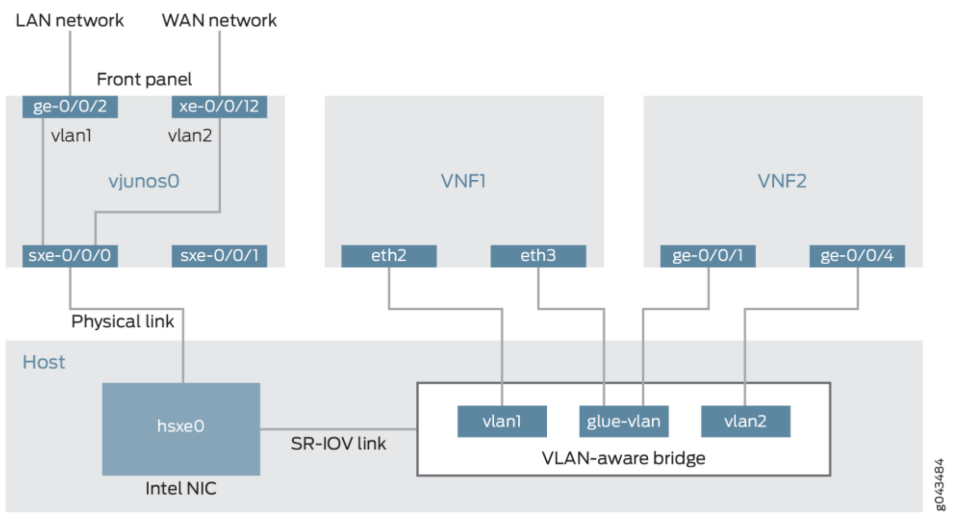


Figure 15: Service Chaining and Juniper Control Plane. [7]

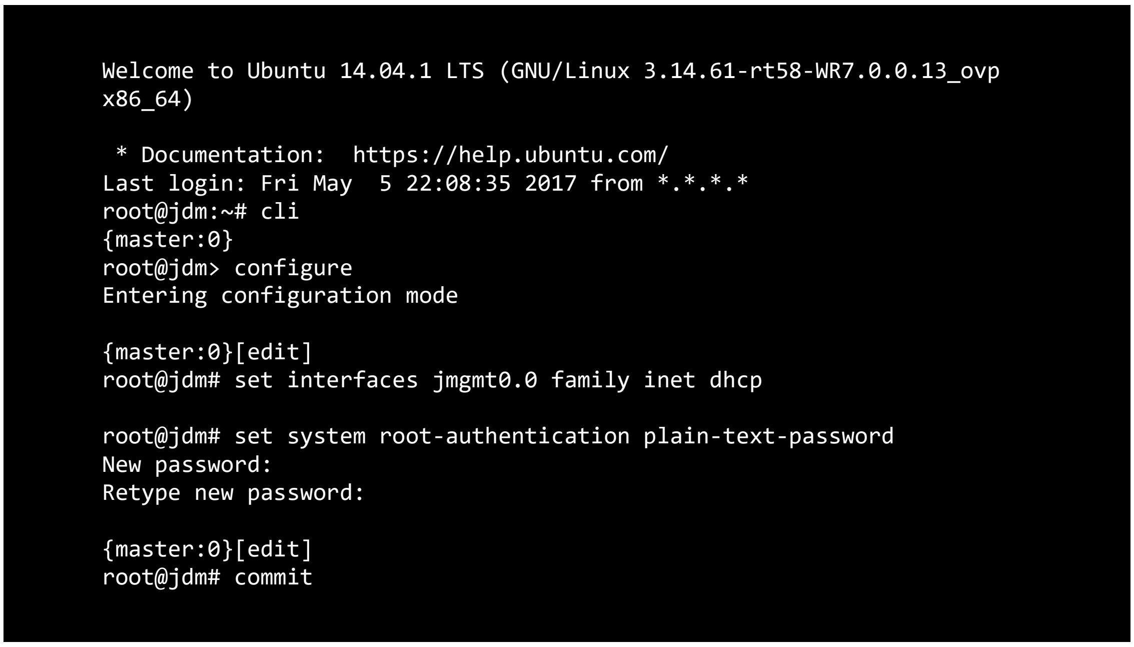
Figure 15 shows service chaining inside the NFX250 device. It is important to notice that the JCP is a VM called vjunos0 in the NFX250. The vjunos0 section of the figure displays the JCPs configuration outcome. The JCP can configure the PFEs (Packet Forwarding Engine) front panel ports and internal-facing ports. Front panel ports are the physical Ethernet media ports like ge-0/0/2 and xe-0/0/12 in the figure. The internal-facing ports are sxe-0/0/0 and sxe-0/0/1.

The IPsec-NM is an IPsec virtual network function docker container that is used for creating an IPsec tunnel for the management network. It is enabled by default, but not configured. When started, a manual definition of the IPsec VPN with the IPsec peer has to done. Container has no interface mapping by default in the JDM. Configuration of the IPsec-nm is not presented in this thesis.

6.2 Deployment Steps

Before going to the VNF specific configuration, the basic management for the platform and transferring the images had to be done.

By using appropriate channels, it is possible to receive the needed VMs for this migration. For configuring the device first time, one needs to establish connectivity using a console cable. When console connection is open, one logins to the JDM CLI and the config mode. Figure 16 shows the configuration of the management interface and the root-authentication password.



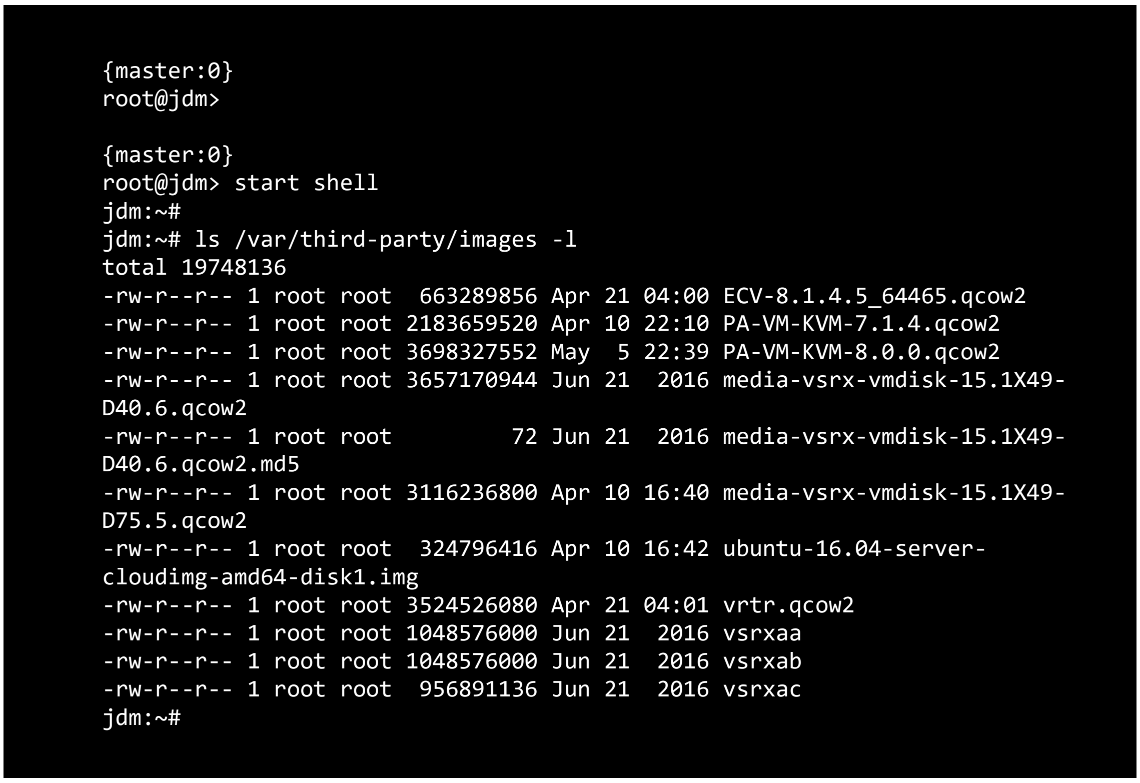
```
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.14.61-rt58-WR7.0.0.13_ovp
x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri May  5 22:08:35 2017 from *.*.*.*
root@jdm:~# cli
{master:0}
root@jdm> configure
Entering configuration mode

{master:0}[edit]
root@jdm# set interfaces jmgmt0.0 family inet dhcp

root@jdm# set system root-authentication plain-text-password
New password:
Retype new password:

{master:0}[edit]
root@jdm# commit
```

Figure 16. Configuration of management interface and root-authentication password.

In the config mode, The jmgmt0.0 interface has to be configured, which is the JDMs logical management interface. SSH and enhanced-orchestration are enabled by default, which means that it is only needed to configure the root authentication for the device and the jmgmt0 interface address (see Figure 16). For start, this will be an address assigned from DHCP. The enhanced-orchestration option toggles on more advanced configuration options that can be used for VNF management and configuration.

One can deploy qcow2 and raw image formats from the JDM. Usually any virtual appliance images meant for KVM have file type qcow or qcow2. The VM file must be transferred to the NFX250 device if it is desired to be deployed in the appliance. A suggested directory path is "/var/third-party/images".

WinSCP is used for moving the VM qcow2 files to the NFX250 as it is fairly lightweight software that can be used in Windows. When using Linux, a good option is to use SCP. After transferring the files to the device, they can be seen in the directory (see Figure 17).



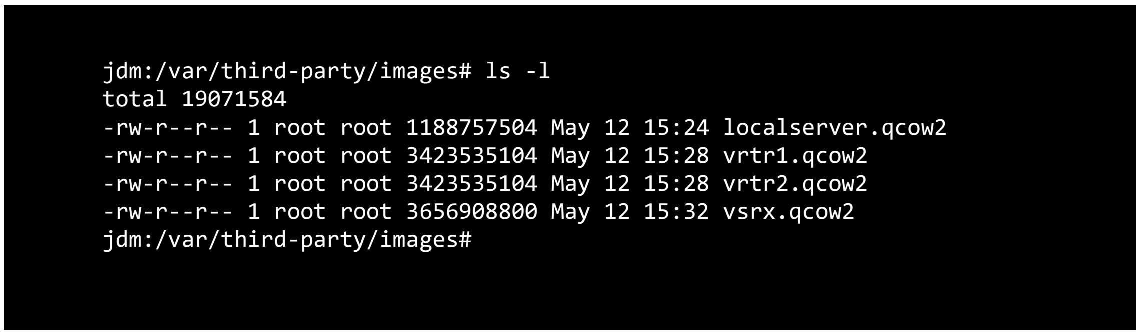
```
{master:0}
root@jdm>

{master:0}
root@jdm> start shell
jdm:~#
jdm:~# ls /var/third-party/images -l
total 19748136
-rw-r--r-- 1 root root  663289856 Apr 21 04:00 ECV-8.1.4.5_64465.qcow2
-rw-r--r-- 1 root root 2183659520 Apr 10 22:10 PA-VM-KVM-7.1.4.qcow2
-rw-r--r-- 1 root root 3698327552 May  5 22:39 PA-VM-KVM-8.0.0.qcow2
-rw-r--r-- 1 root root 3657170944 Jun 21  2016 media-vsrx-vmdisk-15.1X49-
D40.6.qcow2
-rw-r--r-- 1 root root         72 Jun 21  2016 media-vsrx-vmdisk-15.1X49-
D40.6.qcow2.md5
-rw-r--r-- 1 root root 3116236800 Apr 10 16:40 media-vsrx-vmdisk-15.1X49-
D75.5.qcow2
-rw-r--r-- 1 root root  324796416 Apr 10 16:42 ubuntu-16.04-server-
cloudimg-amd64-disk1.img
-rw-r--r-- 1 root root 3524526080 Apr 21 04:01 vrtr.qcow2
-rw-r--r-- 1 root root 1048576000 Jun 21  2016 vsrxaa
-rw-r--r-- 1 root root 1048576000 Jun 21  2016 vsrxab
-rw-r--r-- 1 root root  956891136 Jun 21  2016 vsrxac
jdm:~#
```

Figure 17. Displaying the /var/third-party/images directory and its contents.

Now that the necessary VM files for creating an exact copy of the logical network topology (that was introduced in Figure 8) are in the NFX appliance, the deployment of VNFs can be done.

VNF Deployment

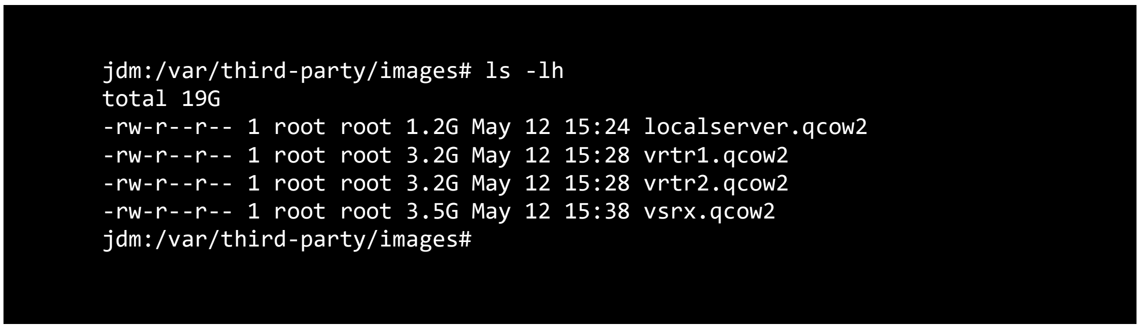
Deploying the VNFs was a simple task and didn’t require much configuration. For displaying the easiness of deployment, one of the VM images was replicated until three qcow2 files existed. These files are used to deploy routing and firewall services. Like mentioned earlier, if a qcow2 file gets copied, it is its own instance and can be executed as such. Figure 18 shows the images needed for the branch network migration.



```
jdm:/var/third-party/images# ls -l
total 19071584
-rw-r--r-- 1 root root 1188757504 May 12 15:24 localserver.qcow2
-rw-r--r-- 1 root root 3423535104 May 12 15:28 vrtr1.qcow2
-rw-r--r-- 1 root root 3423535104 May 12 15:28 vrtr2.qcow2
-rw-r--r-- 1 root root 3656908800 May 12 15:32 vsrx.qcow2
jdm:/var/third-party/images#
```

Figure 18. Necessary images for deploying the current network topology virtually.

The size of the vsrx.qcow2 file is different than the vrtr1 and vrtr2 in Figure 18, even though they origin from the same file. This is because vsrx.qcow2 has already been deployed and configured. The modified date will be changing as long as the VNF is running as shown in Figure 19.

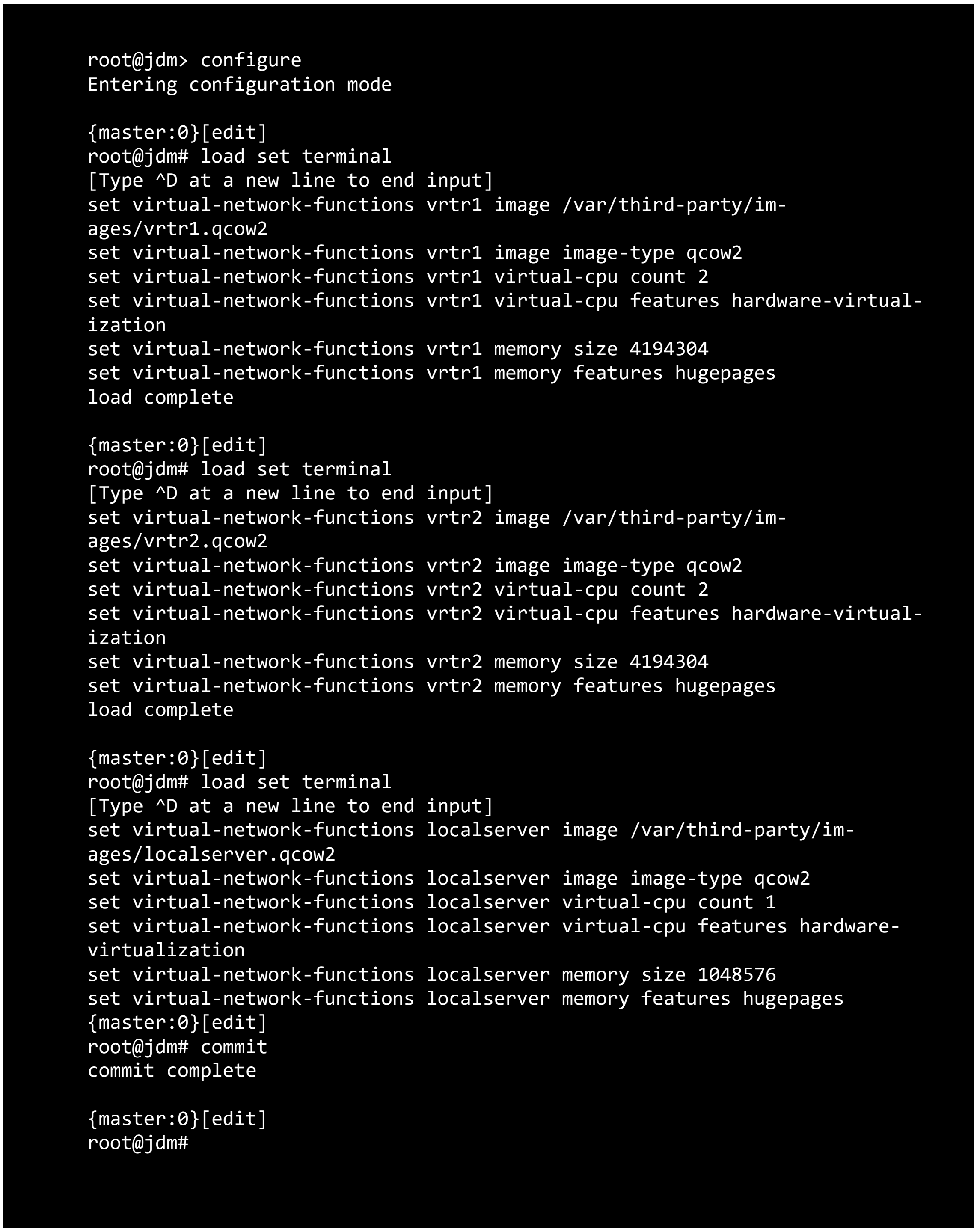


```
jdm:/var/third-party/images# ls -lh
total 19G
-rw-r--r-- 1 root root 1.2G May 12 15:24 localserver.qcow2
-rw-r--r-- 1 root root 3.2G May 12 15:28 vrtr1.qcow2
-rw-r--r-- 1 root root 3.2G May 12 15:28 vrtr2.qcow2
-rw-r--r-- 1 root root 3.5G May 12 15:38 vsrx.qcow2
jdm:/var/third-party/images#
```

Figure 19. Modified date is changing for the vsrx.qcow2 file. Some files are hidden, leading to inaccurate total size shown.

Figure 19 shows that there is a 6 minute difference in the modified date of vsrx.qcow2 file when compared to the modified date in Figure 18.

The first part of the configuration was to deploy all four VMs without data interfaces. The only configuration done was the VNF system requirements needed to run the VMs. Deployment is shown in Figure 20.



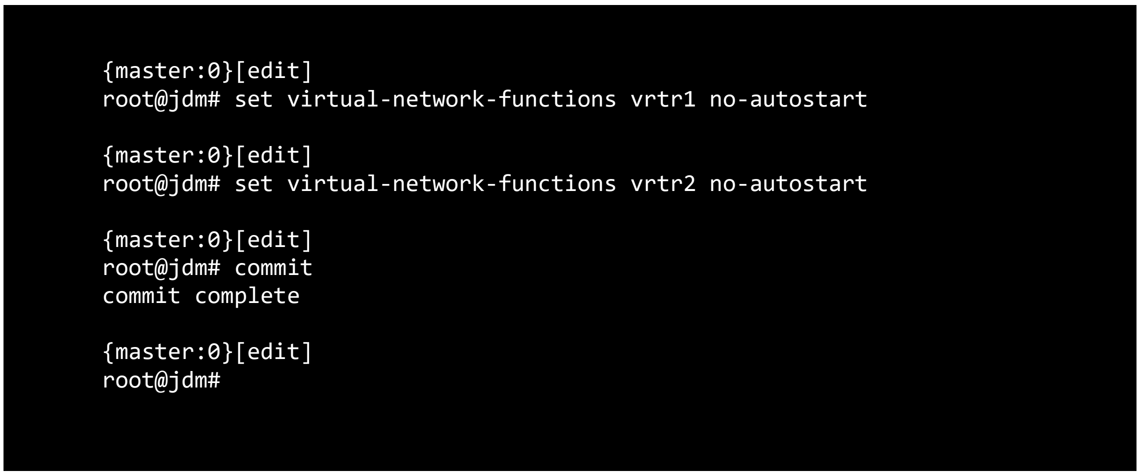
```
root@jdm> configure
Entering configuration mode

{master:0}[edit]
root@jdm# load set terminal
[Type ^D at a new line to end input]
set virtual-network-functions vrtr1 image /var/third-party/im-
ages/vrtr1.qcow2
set virtual-network-functions vrtr1 image image-type qcow2
set virtual-network-functions vrtr1 virtual-cpu count 2
set virtual-network-functions vrtr1 virtual-cpu features hardware-virtual-
ization
set virtual-network-functions vrtr1 memory size 4194304
set virtual-network-functions vrtr1 memory features hugepages
load complete

{master:0}[edit]
root@jdm# load set terminal
[Type ^D at a new line to end input]
set virtual-network-functions vrtr2 image /var/third-party/im-
ages/vrtr2.qcow2
set virtual-network-functions vrtr2 image image-type qcow2
set virtual-network-functions vrtr2 virtual-cpu count 2
set virtual-network-functions vrtr2 virtual-cpu features hardware-virtual-
ization
set virtual-network-functions vrtr2 memory size 4194304
set virtual-network-functions vrtr2 memory features hugepages
load complete

{master:0}[edit]
root@jdm# load set terminal
[Type ^D at a new line to end input]
set virtual-network-functions localserver image /var/third-party/im-
ages/localserver.qcow2
set virtual-network-functions localserver image image-type qcow2
set virtual-network-functions localserver virtual-cpu count 1
set virtual-network-functions localserver virtual-cpu features hardware-
virtualization
set virtual-network-functions localserver memory size 1048576
set virtual-network-functions localserver memory features hugepages
{master:0}[edit]
root@jdm# commit
commit complete

{master:0}[edit]
root@jdm#
```

Figure 20. Deployment of vrtr1, vrtr2 and localserver VNFs.

The system configuration is the same for both of the vrtr images in Figure 20. Deployment of vsrx.qcow2 is not shown in as it is already running.

The configuration in Figure 20 defines the path to the qcow2 image that is going to be used for a specific VNF. The image file type and the number of virtual CPUs allocated is defined. For these specific VNFs, hardware virtualization is toggled on. The amount of memory allocated comes from the system requirements of the specific VNF. Option hugepages is configured to support the pre-allocated memory pages defined in JDM system options.

The VNFs boot up and start the initial configuration after commiting the configuration. This behaviour can be deactivated with the command “no-autostart” shown in Figure 21.



```
{master:0}[edit]
root@jdm# set virtual-network-functions vrtr1 no-autostart

{master:0}[edit]
root@jdm# set virtual-network-functions vrtr2 no-autostart

{master:0}[edit]
root@jdm# commit
commit complete

{master:0}[edit]
root@jdm#
```

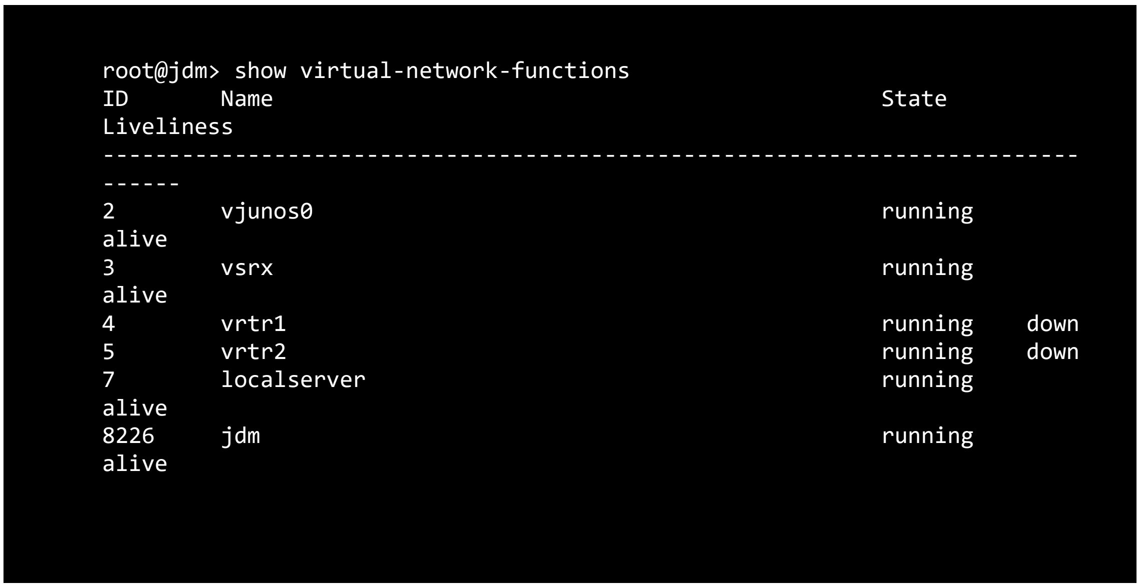
Figure 21. Configuration of "no-autostart" option.

If a VNF is not deployed before configuring no-autostart, it requires manual input to be started. Manual input is also needed always after a power outage or a complete system restart of the NFX250. In certain situations, this can cause issues as there is a possibility that critical VNFs will not restart automatically.

The NFX appliance allocates two virtual interfaces to a VM every time a VM is created. These are called eth0 and eth1. Eth0 is the first interface and is mapped to the out-of-band management interface of the VNF. This provides an IP connection from hypervisor to VNF. It is possible to use SSH to login from JDM to a VNF after this connectivity.

Eth1 is used for a connection to the dedicated management bridge. This management bridge is directly connected to the out-of-band Ethernet management interface of NFX250.

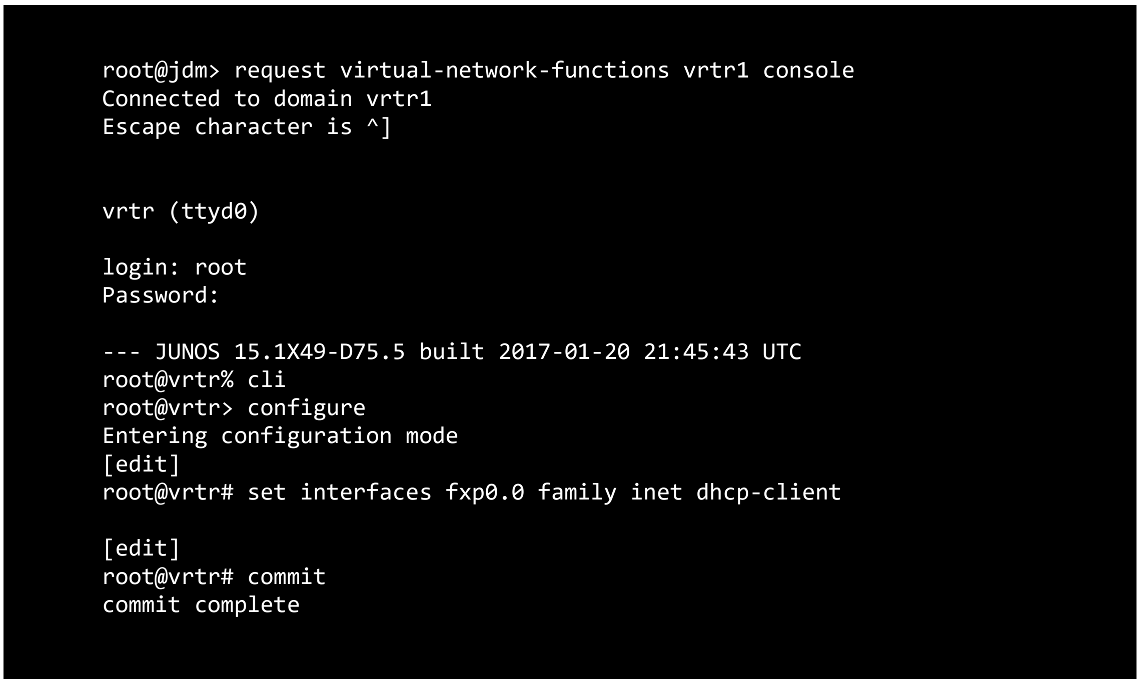
Some VM images do not have configuration ready to enable Hypervisor – VNF connectivity. The status of the VNFs can be checked with the command show in Figure 22.



```
root@jdm> show virtual-network-functions
ID       Name                                                  State
Liveliness
---------------------------------------------------------------------------
------
2        vjunos0                                               running
alive
3        vsrx                                                  running
alive
4        vrtr1                                                 running    down
5        vrtr2                                                 running    down
7        localserver                                           running
alive
8226     jdm                                                   running
alive
```

Figure 22. "show virtual-network-functions" displays the state and liveliness of the VNFs

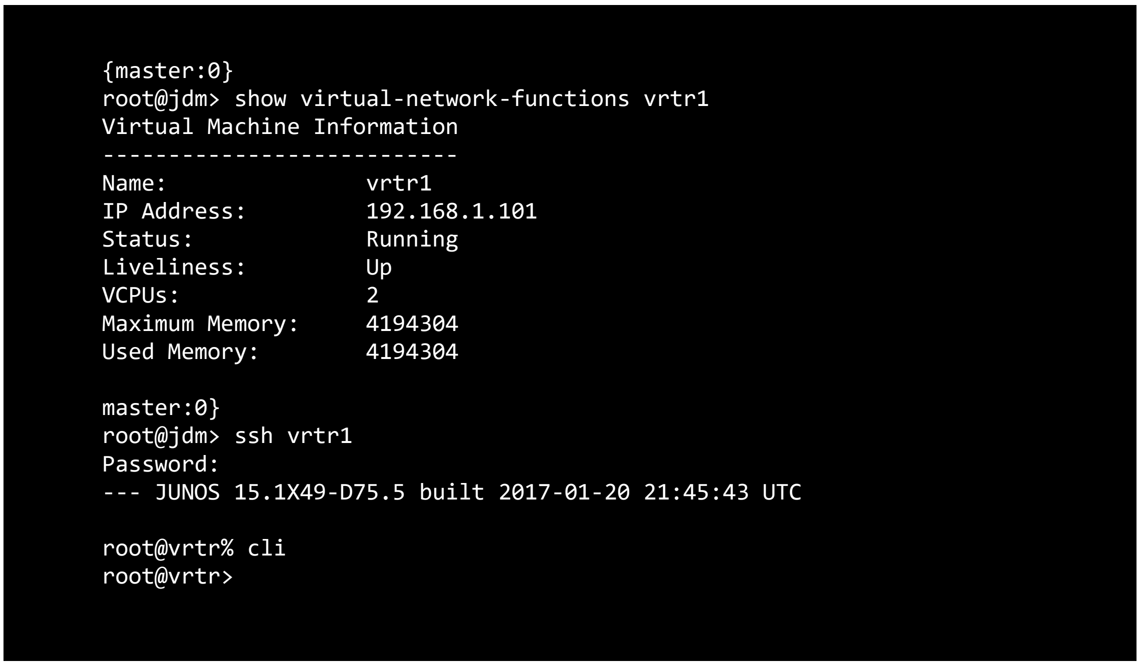
As shown in Figure 22, it can be noticed that the three newly deployed VNFs are currently running as intended. However, the vrtr1 and vrtr2 liveliness shows that they are down. This means that the JDM / Hypervisor does not have internal IP connectivity to the device. In this situation, a console connection can be taken from the JDM / Hypervisor to the VM and configure necessary parameters as shown in Figure 23.



```
root@jdm> request virtual-network-functions vrtr1 console
Connected to domain vrtr1
Escape character is ^]


vrtr (ttyd0)

login: root
Password:

--- JUNOS 15.1X49-D75.5 built 2017-01-20 21:45:43 UTC
root@vrtr% cli
root@vrtr> configure
Entering configuration mode
[edit]
root@vrtr# set interfaces fxp0.0 family inet dhcp-client

[edit]
root@vrtr# commit
commit complete
```

Figure 23. Example of a console connection to the vrtr1 and configuring hypervisor connectivity

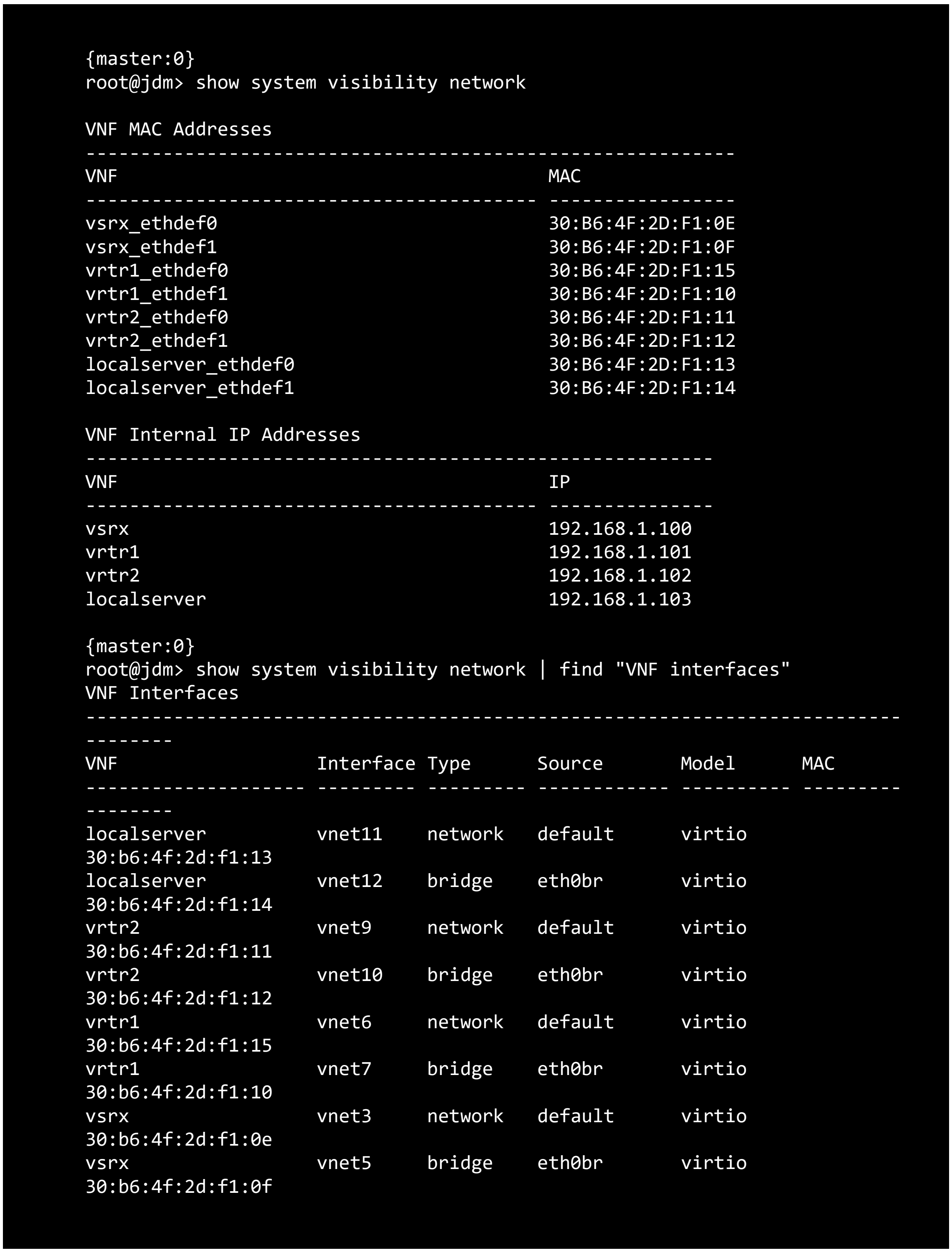
After configuring fxp0.0 (the out-of-band interface of the vrtr), the liveliness can be checked again. A command used for checking the state of a single VNF can be issued. (see Figure 24). VM became reachable via SSH.



```
{master:0}
root@jdm> show virtual-network-functions vrtr1
Virtual Machine Information
---------------------------
Name:                vrtr1
IP Address:          192.168.1.101
Status:              Running
Liveliness:          Up
VCPUs:               2
Maximum Memory:      4194304
Used Memory:         4194304

master:0}
root@jdm> ssh vrtr1
Password:
--- JUNOS 15.1X49-D75.5 built 2017-01-20 21:45:43 UTC

root@vrtr% cli
root@vrtr>
```

Figure 24. Displaying VM information and SSH connectivity to it.

When creating a new VM, the JDM automatically creates a name entry for the VMs IP address shown in the “show virtual-network-functions” command. To make sure that the default interfaces are actually mapped correctly, the system visibility can be viewed from the JDM side as shown in Figure 25.



```
{master:0}
root@jdm> show system visibility network

VNF MAC Addresses
-----------------------------------------------------------
VNF                                      MAC
---------------------------------------- -----------------
vsrx_ethdef0                             30:B6:4F:2D:F1:0E
vsrx_ethdef1                             30:B6:4F:2D:F1:0F
vrtr1_ethdef0                            30:B6:4F:2D:F1:15
vrtr1_ethdef1                            30:B6:4F:2D:F1:10
vrtr2_ethdef0                            30:B6:4F:2D:F1:11
vrtr2_ethdef1                            30:B6:4F:2D:F1:12
localserver_ethdef0                      30:B6:4F:2D:F1:13
localserver_ethdef1                      30:B6:4F:2D:F1:14

VNF Internal IP Addresses
---------------------------------------------------------
VNF                                      IP
---------------------------------------- ---------------
vsrx                                     192.168.1.100
vrtr1                                    192.168.1.101
vrtr2                                    192.168.1.102
localserver                              192.168.1.103

{master:0}
root@jdm> show system visibility network | find "VNF interfaces"
VNF Interfaces
--------------------------------------------------------------------------
--------
VNF                  Interface Type      Source       Model      MAC
-------------------- --------- --------- ------------ ---------- ---------
--------
localserver          vnet11    network   default      virtio
30:b6:4f:2d:f1:13
localserver          vnet12    bridge    eth0br       virtio
30:b6:4f:2d:f1:14
vrtr2                vnet9     network   default      virtio
30:b6:4f:2d:f1:11
vrtr2                vnet10    bridge    eth0br       virtio
30:b6:4f:2d:f1:12
vrtr1                vnet6     network   default      virtio
30:b6:4f:2d:f1:15
vrtr1                vnet7     bridge    eth0br       virtio
30:b6:4f:2d:f1:10
vsrx                 vnet3     network   default      virtio
30:b6:4f:2d:f1:0e
vsrx                 vnet5     bridge    eth0br       virtio
30:b6:4f:2d:f1:0f
```

Figure 25. Clipped output of "show system visibility network" command.

Data side deployment

To achieve the legacy networks service chaining inside the appliance, data interfaces must be configured. The management side configurations did not require any parameters from the JCP, however the data side does need. Two different methods were used for the VNFs in order to achieve proper service chain. These are service chaining using SR-IOV and Virtio. Figure 26 illustrates the interfaces mapped with virtio and SR-IOV.

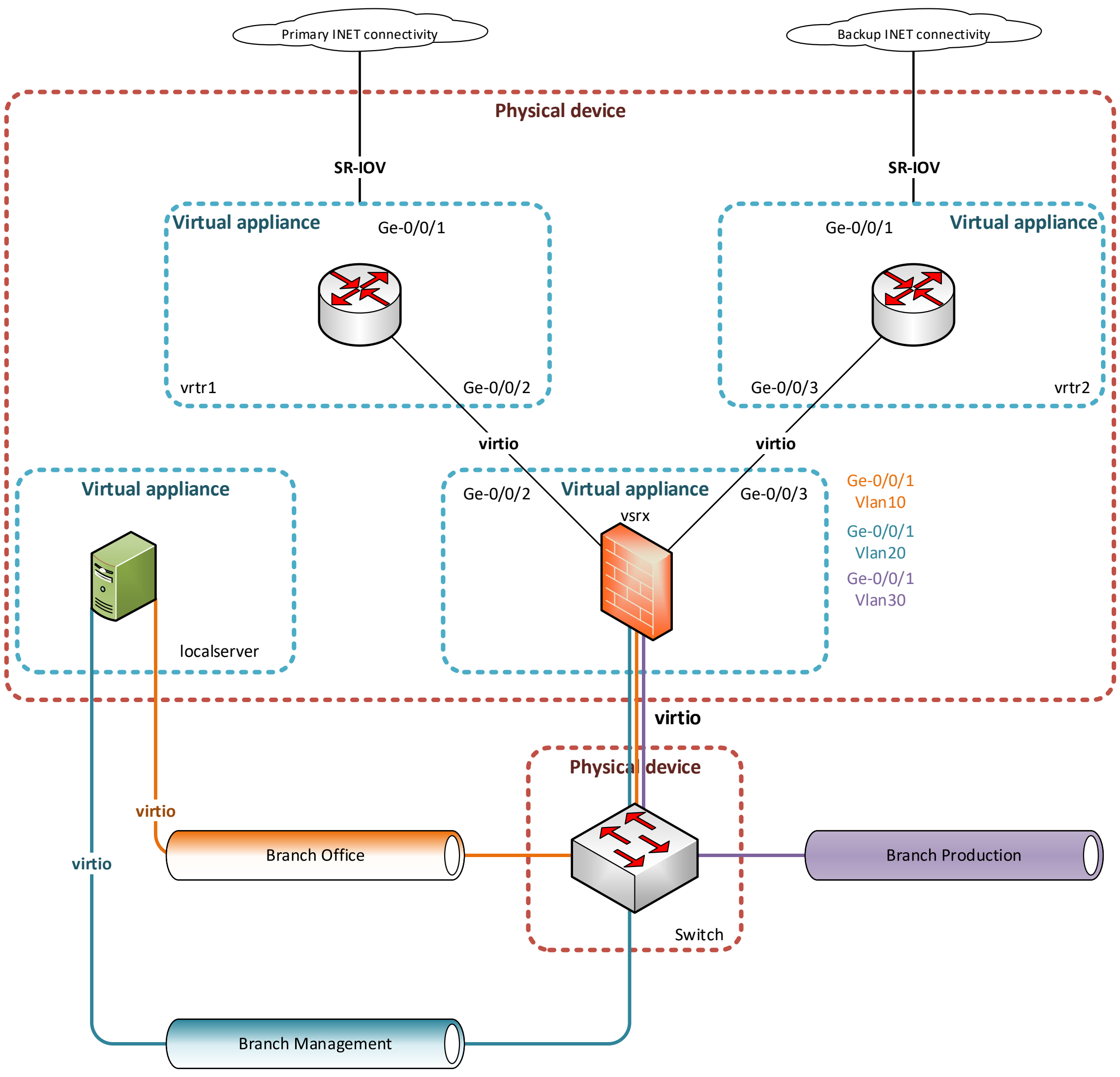


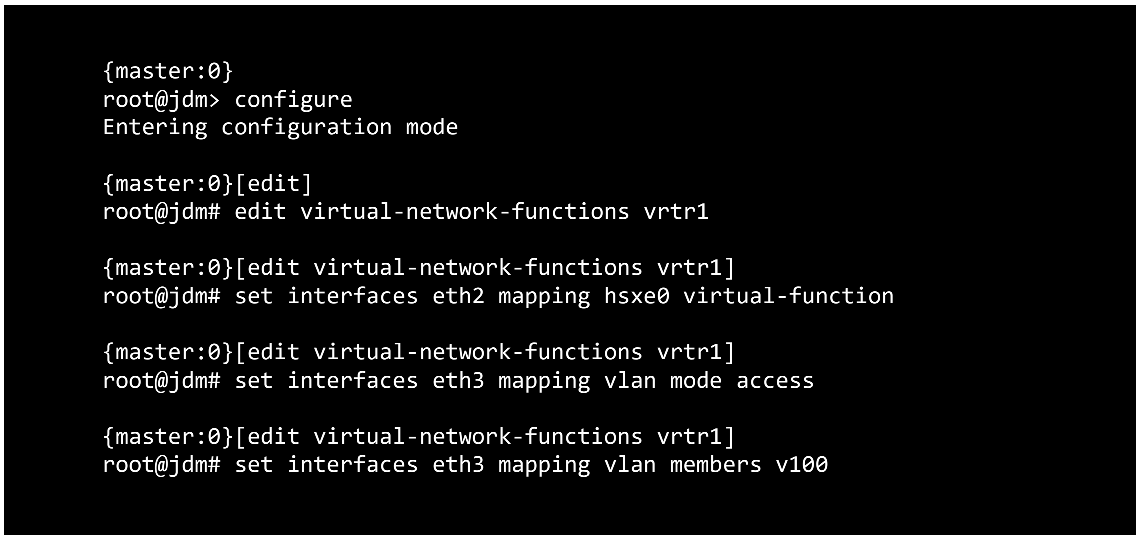
Figure 26. Interfaces mapped with virtio are internal and LAN connections. WAN connections use SR-IOV.

For internal and LAN connections, virtio and VLAN-aware bridge was used (see Figure 26). The WAN connectivity was handled using SR-IOV. This was to demonstrate both methods in connecting the VMs together.

Data interfaces (office and production traffic) in the JDM were mapped with the following logic:

1. Vrtr1
    a. Ge-0/0/1 – eth2 – SR-IOV
    b. Ge-0/0/2 – eth3 – virtio
2. Vrtr2
    a. Ge-0/0/1 – eth2 – SR-IOV
    b. Ge-0/0/3 – eth4 – virtio
3. Vsrx
    a. Ge-0/0/1 – eth2 – virtio
    b. Ge-0/0/2 – eth3 – virtio
    c. Ge-0/0/3 – eth4 – virtio
4. Localserver
    a. eth2 – eth2 – virtio

Data interfaces were mapped in the JDM according to the list previously displayed. The first interface variable “Ge-0/0/*” or in localserver “eth2” reflects the interface inside the VM. The second interface variable “eth*” reflects the JDM side interface abstract allocated for the VM. Virtio or SR-IOV is the type of NIC virtualization. These settings are configured for vrtr1 in Figure 27.

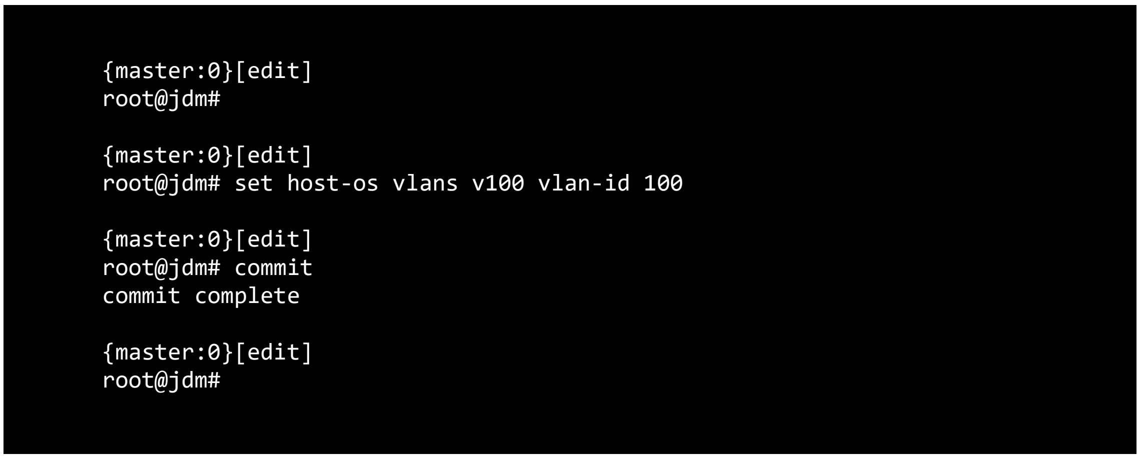


```
{master:0}
root@jdm> configure
Entering configuration mode

{master:0}[edit]
root@jdm# edit virtual-network-functions vrtr1

{master:0}[edit virtual-network-functions vrtr1]
root@jdm# set interfaces eth2 mapping hsxe0 virtual-function

{master:0}[edit virtual-network-functions vrtr1]
root@jdm# set interfaces eth3 mapping vlan mode access

{master:0}[edit virtual-network-functions vrtr1]
root@jdm# set interfaces eth3 mapping vlan members v100
```

Figure 27. Mapping of interfaces under the VM options.

The configuration in Figure 27 displays the parameters for data interfaces regarding the VM vrtr1 to display both virtio and SR-IOV model. Configuration is done in JDM under the “virtual-network-functions vrtr1” stanza.

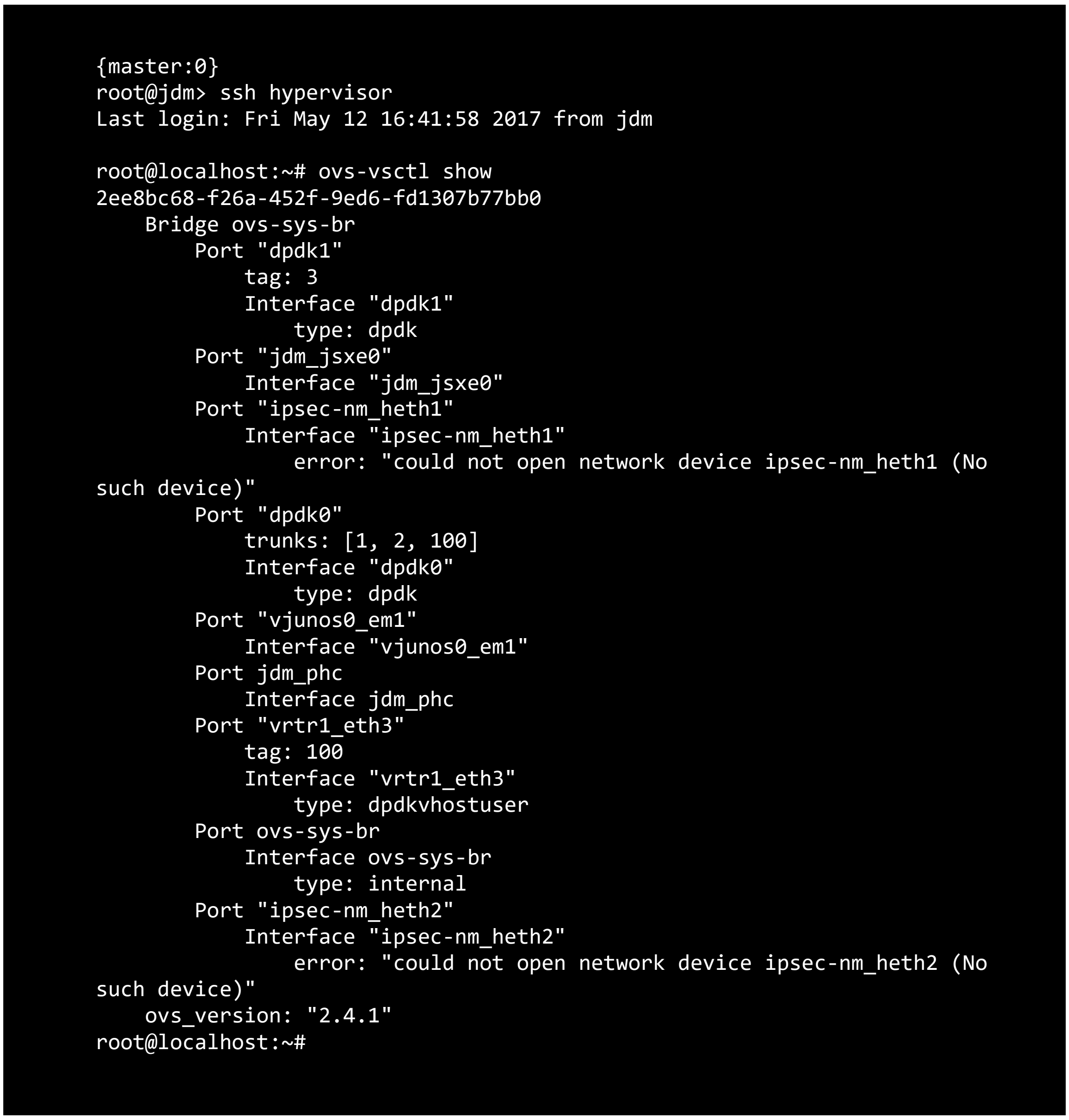
Before configuring the JCP for the physical connectivity to eth2, some other settings were still required in the JDM regarding eth3. These settings were about the VLAN-aware bridge, also called host-os switch. The host-os needed to have some configuration in place so that the vlan “v100” actually exists on the switch. This configuration can be seen in Figure 28.



```
{master:0}[edit]
root@jdm#

{master:0}[edit]
root@jdm# set host-os vlans v100 vlan-id 100

{master:0}[edit]
root@jdm# commit
commit complete

{master:0}[edit]
root@jdm#
```

Figure 28. Creating the vlan for host-os switch.

Configuration done in the JDM commands the host-os switch. This can be verified by checking the port information of the OVS (Open vSwitch) in hypervisor side as shown in Figure 29.



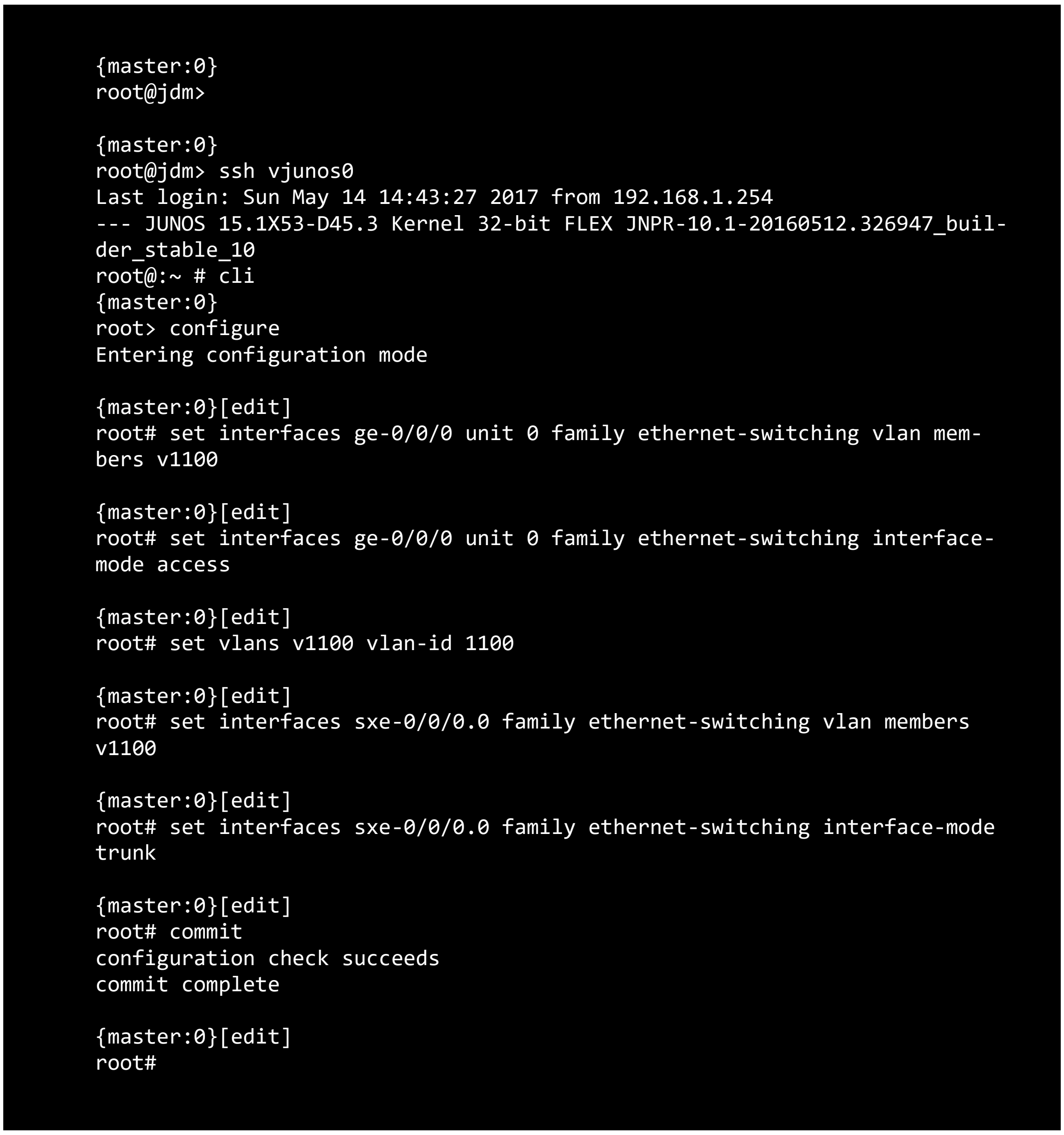
```
{master:0}
root@jdm> ssh hypervisor
Last login: Fri May 12 16:41:58 2017 from jdm

root@localhost:~# ovs-vsctl show
2ee8bc68-f26a-452f-9ed6-fd1307b77bb0
    Bridge ovs-sys-br
        Port "dpdk1"
            tag: 3
            Interface "dpdk1"
                type: dpdk
        Port "jdm_jsxe0"
            Interface "jdm_jsxe0"
        Port "ipsec-nm_heth1"
            Interface "ipsec-nm_heth1"
                error: "could not open network device ipsec-nm_heth1 (No
such device)"
        Port "dpdk0"
            trunks: [1, 2, 100]
            Interface "dpdk0"
                type: dpdk
        Port "vjunos0_em1"
            Interface "vjunos0_em1"
        Port jdm_phc
            Interface jdm_phc
        Port "vrtr1_eth3"
            tag: 100
            Interface "vrtr1_eth3"
                type: dpdkvhostuser
        Port ovs-sys-br
            Interface ovs-sys-br
                type: internal
        Port "ipsec-nm_heth2"
            Interface "ipsec-nm_heth2"
                error: "could not open network device ipsec-nm_heth2 (No
such device)"
    ovs_version: "2.4.1"
root@localhost:~#
```

Figure 29. Output of Open vSwitch port info.

As shown in Figure 29, the output reflects how the JDM works as an abstract for the underlying host-os components. The JDM configuration done regarding the vrtr1 has defined a port “vrtr1_eth3” to the OVS. No configuration should be done to the OVS side, rather let the JDM handle the configuration of the OVS.

Figure 29 displays that the Port “ipsec-nm_heth1” and “ipsec-nm_heth2” are giving an error as those network devices do not exist. This is because the ipsec-nm service is not currently enabled. When the service is disabled, the container doesn’t exists but the static configuration still exists on the OVS configuration.

The JCP handles the PFE connectivity. Internal connectivity between VMs can be achieved without any configuration to the JCP, but if traffic flow should be possible from the physical ports of the NFX appliance to a VM, then JCP needs to be configured. JCP is accessible from the JDM as shown in Figure 30.

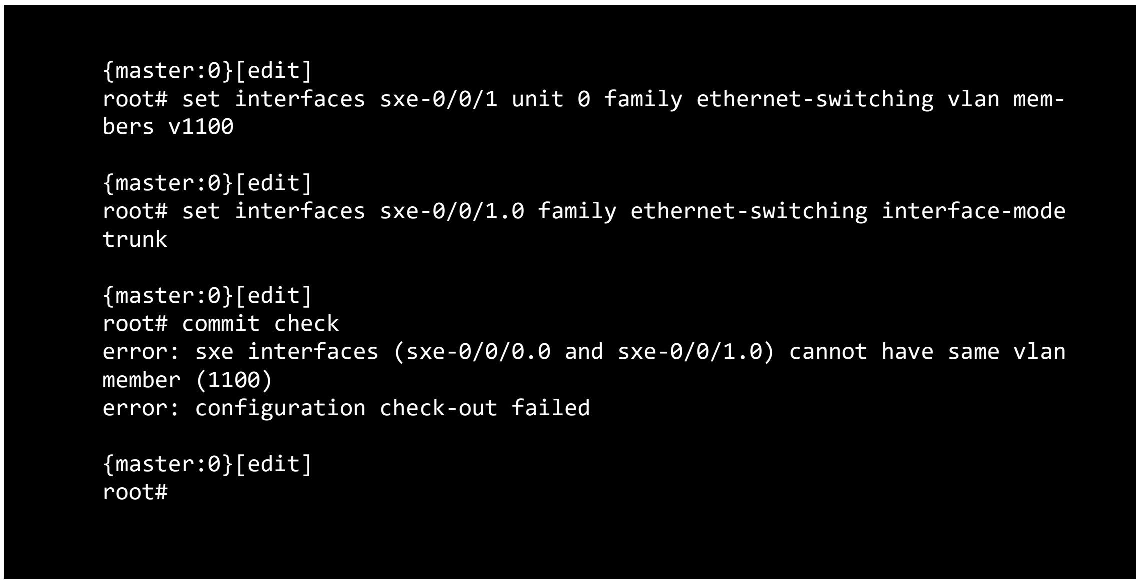


```
{master:0}
root@jdm>

{master:0}
root@jdm> ssh vjunos0
Last login: Sun May 14 14:43:27 2017 from 192.168.1.254
--- JUNOS 15.1X53-D45.3 Kernel 32-bit FLEX JNPR-10.1-20160512.326947_buil-
der_stable_10
root@:~ # cli
{master:0}
root> configure
Entering configuration mode

{master:0}[edit]
root# set interfaces ge-0/0/0 unit 0 family ethernet-switching vlan mem-
bers v1100

{master:0}[edit]
root# set interfaces ge-0/0/0 unit 0 family ethernet-switching interface-
mode access

{master:0}[edit]
root# set vlans v1100 vlan-id 1100

{master:0}[edit]
root# set interfaces sxe-0/0/0.0 family ethernet-switching vlan members
v1100

{master:0}[edit]
root# set interfaces sxe-0/0/0.0 family ethernet-switching interface-mode
trunk

{master:0}[edit]
root# commit
configuration check succeeds
commit complete

{master:0}[edit]
root#
```

Figure 30. JCP configuration for the vrtr1 VM.

Vrtr1 configuration in Figure 27 shows that the WAN interface uses SR-IOV and is mapped to hsxe0. The JCP equivalent for hsxe0 is the sxe-0/0/0 interface. Vlan1100 is used for the connectivity from physical port ge-0/0/0 to the vrtr1 port ge-0/0/1 in the configuration shown in Figure 30.

The sxe-0/0/0 and the sxe-0/0/1 are normally configured as trunk ports. Same VLANs cannot pertain on both of these interfaces at the same time due to a possibility of a loop – the device will also give an error regarding this as shown in Figure 31.



```
{master:0}[edit]
root# set interfaces sxe-0/0/1 unit 0 family ethernet-switching vlan mem-
bers v1100

{master:0}[edit]
root# set interfaces sxe-0/0/1.0 family ethernet-switching interface-mode
trunk

{master:0}[edit]
root# commit check
error: sxe interfaces (sxe-0/0/0.0 and sxe-0/0/1.0) cannot have same vlan
member (1100)
error: configuration check-out failed

{master:0}[edit]
root#
```

Figure 31. Error in configuration check-out. This is caused by sxe-0/0/0 and sxe-0/0/1 having the same VLAN allowed.

Using the same configuration methods as in the examples of chapter 6.2, the network services for all VMs were configured to the NFX device. The end result of the topology for data traffic flow can be seen in Figure 32.

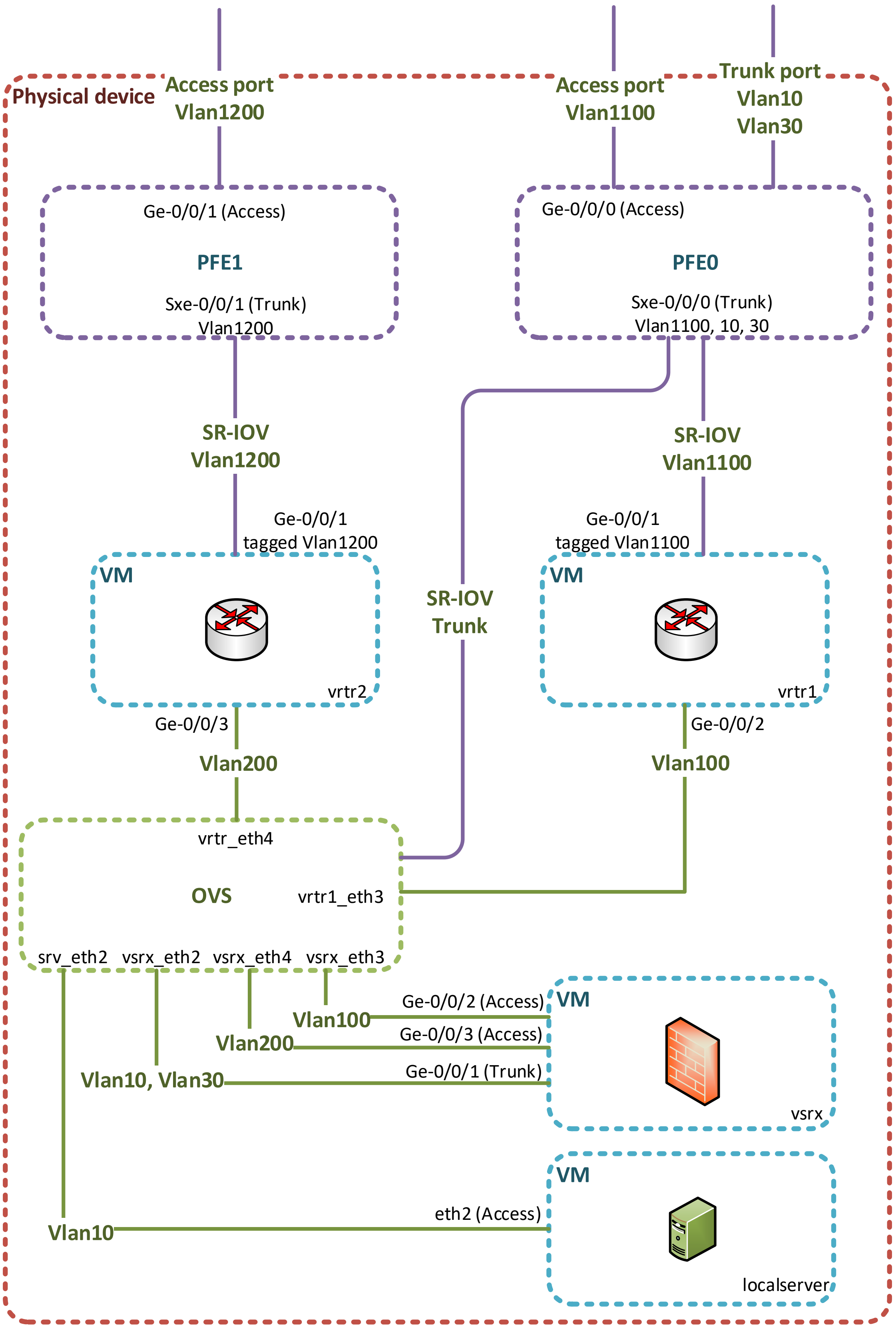


Figure 32. Logical topology of all components with network services deployed. This diagram shows only the data side as management side uses different components.

Deploying a completely new service

In the last part of the thesis, a completely new service was deployed to the virtualized network. The idea was to find out and demonstrate how swift the service deployment can be when new services are virtualized.

The steps for this deployment were mentioned earlier and are the following:

1. Transfer the VNF image to the site NFV platform. Prefer Secure Copy for transferring the image.

2. Deploy the image with the necessary service chain and VM specifications

3. Add the deployed VM to management and monitoring systems

The third step is not covered in the thesis.

The topology for the new VNF is shown in Figure 33 and it represents service chains only related for the new service - no service shown in Figure 32 was removed.

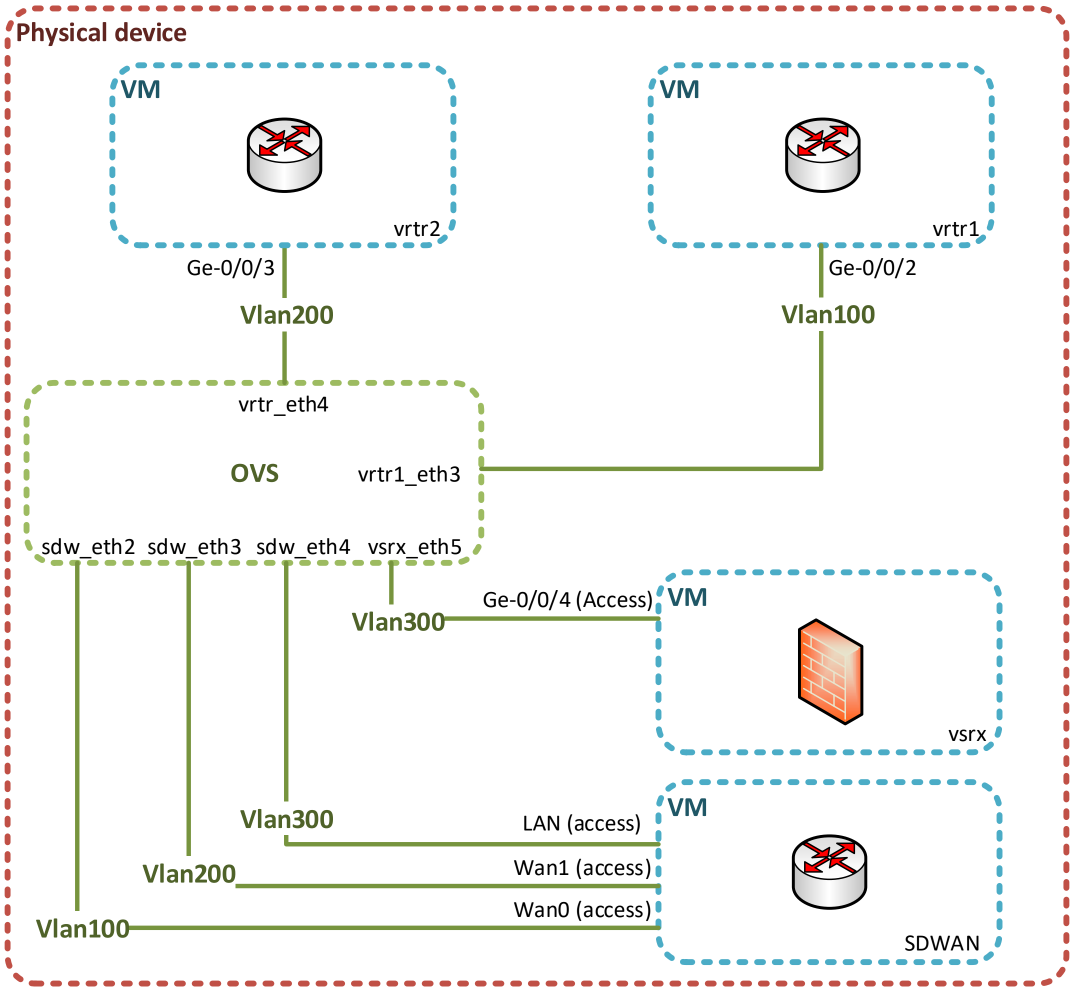


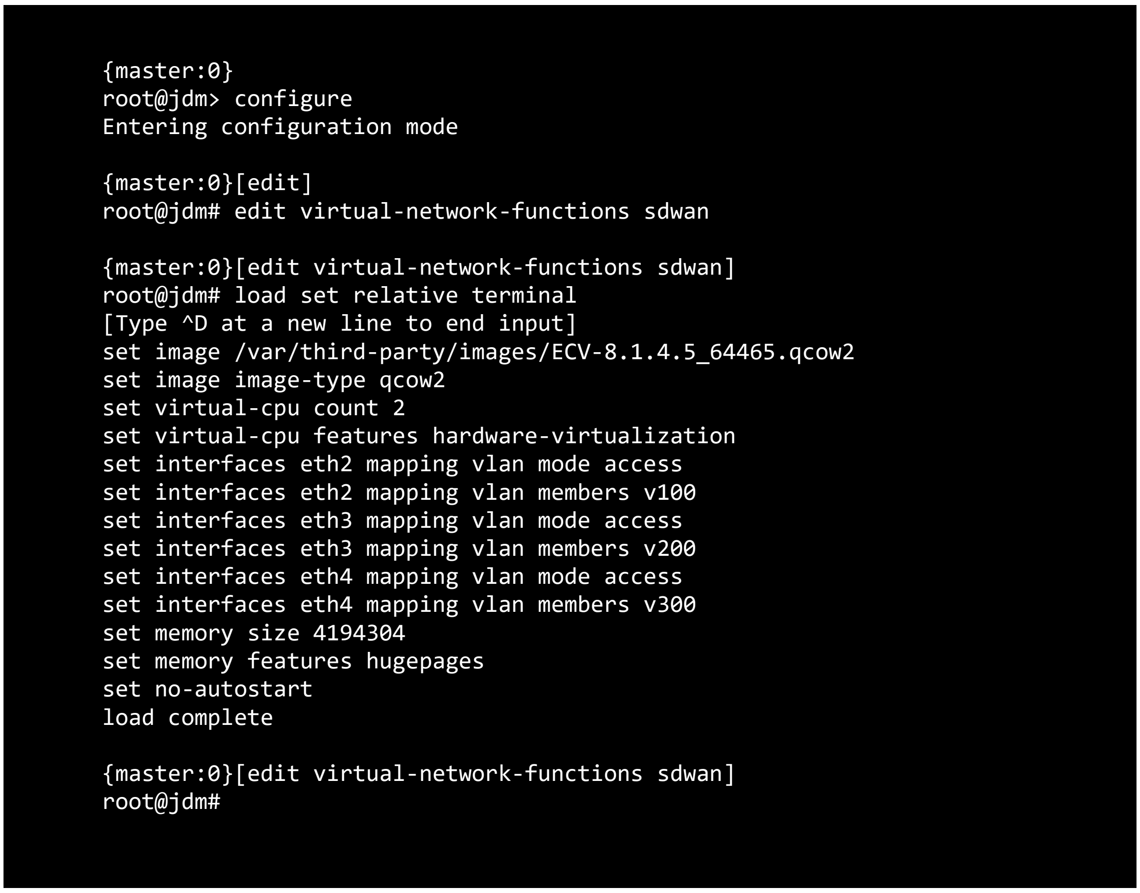
Figure 33. New service and its position in the service chain.

The SDWAN VM file was first moved to the branch network NFX appliance as shown in Figure 34.

```
root@jdm:/var/third-party/images# scp ECV-8.1.4.5_64465.qcow2
root@198.51.100.10:/var/third-party/images
The authenticity of host 198.51.100.10 (198.51.100.10)' can't be estab-
lished.
ECDSA key fingerprint is *******************.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '198.51.100.10' (ECDSA) to the list of known
hosts.
root@198.51.100.10's password:
ECV-8.1.4.5_64465.qcow2
100%  633MB   1.1MB/s   09:30
```

Figure 34. Moving VNF file to the branch network NFV platform.

The method used for transferring the image was SCP and it was copied from the HQ NFX device. After the file was transferred, service chain needed for this VNF was configured in the JDM as shown in Figure 35. This VNF did not require any configuration in the JCP side.



```
{master:0}
root@jdm> configure
Entering configuration mode

{master:0}[edit]
root@jdm# edit virtual-network-functions sdwan

{master:0}[edit virtual-network-functions sdwan]
root@jdm# load set relative terminal
[Type ^D at a new line to end input]
set image /var/third-party/images/ECV-8.1.4.5_64465.qcow2
set image image-type qcow2
set virtual-cpu count 2
set virtual-cpu features hardware-virtualization
set interfaces eth2 mapping vlan mode access
set interfaces eth2 mapping vlan members v100
set interfaces eth3 mapping vlan mode access
set interfaces eth3 mapping vlan members v200
set interfaces eth4 mapping vlan mode access
set interfaces eth4 mapping vlan members v300
set memory size 4194304
set memory features hugepages
set no-autostart
load complete

{master:0}[edit virtual-network-functions sdwan]
root@jdm#
```

Figure 35. SDWAN VNF configuration

Figure 35 shows the configuration for the new service VNF in the branch network. In addition to these parameters, the host-os vlan v300 was configured and the virtual firewall interface for it.

After committing the configuration, the VNF was started manually in the following evening and configuration of VNFs were changed so that the internal traffic started to use this new service VNF. Deployment of the VNF itself did not need any service breaks and was completed in one work day.

7 Discussion and Conclusions

The goal of this thesis was to research the Virtual Network Function model and find out whether it can meet the common operational guidelines of branch network service deployment and management. A virtualized approach on branch network needed to fulfil certain management and security guidelines required from already in-place physical appliances. Additionally some expectations were made for the virtualized model about the delivery of new services.

The first impressions did not change much during the study and the project went on quite smoothly. The architecture of the example branch network responded well to the virtual approach. As the deployment is still very recent, an intensive care period is on-going. This is normal with all larger migrations. No feedback has been received which is a positive sign indicating that everything is running as intended.

All requirements from the previous physical network model were met with appropriate and acceptable accuracy. The management of the VNFs and the hypervisor was done with secure methods, SSH and SCP. Visibility could be gained using SNMP after the migration for the VMs. Delivery of new services took a major leap forward with the virtualized model and the delivery time was reduced to one day instead of weeks that it used to be. It is expected to be reduced even more in the future.

Migrating to NFV model was not harder than migrating to any new physical network topology. Because the service chaining took care of the internal connectivity, it actually made the physical connections more simple. On the operational side some time was needed to learn technology as it is fairly new and necessary skills for production usage were missing. However, this is true with all new technology deployments.

The biggest challenge for this deployment model is still the VMs and their support in a virtual deployment scenario. Not all services have a virtual counterpart out yet. When deploying these VMs, it is extremely critical to test them in a lab environment first. Virtualizing a network will add one more layer for possible issues as the VMs run on a hypervisor. These are problems caused by the hypervisor or internal components of the platform. Technical assistance is usually needed (If the issue is not obvious) from the NFV platform vendor and the software vendor.

When deploying the VNF model to a production environment, it is important to understand the benefits it can have to the overall network and service delivery. Deploying NFV platforms to small sites with just one network function is not recommended as the capital expenditure for a NFV platform might be even higher than for example small physical routers.

Virtualization overall and especially network function virtualization brings the traditional server people and networking people much closer together. The borders between these two traditionally different groups is disappearing. This might cause issues and slow down the development as people usually oppose this kind of change.

As a summary one can state that the NFV model has successfully met all the requirements given to it and outperformed the old physical model in service delivery.

It looks like the virtualization of network functions will become more and more common in the future. As mentioned above, this stand-alone managed type of deployment is just a scratch to the surface of something larger that has been divided into smaller blocks to help the development of automated and orchestrated complete NFV deployment of customer network(Figure 10). The next steps regarding the VNF model will definitely be deploying it to production environments in a larger scale and the investigation of possible management and orchestration systems. The selection of platforms for NFV is still very small, however this will most definitely change in the coming years.

References

1 ETSI NFV ISG (2012) Network Functions Virtualisation – Introductory White Paper (October 22, 2012), https://portal.etsi.org/nfv/nfv_white_paper.pdf (Accessed April 3, 2017)

2 ETSI NFV ISG (2013) Use Cases, (October 2013), http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf (Accessed April 3, 2017).

3 ETSI NFV ISG (2013) Architectural Framework, (October 2013), http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf (Accessed April 3, 2017)

4 SDX Central (2017): What is Network Service Chaining, https://www.sdxcentral.com/sdn/network-virtualization/definitions/what-is-network-service-chaining/ (Accessed April 3, 2017)

5 PCI-SIG Single Root I/O Virtualization(SR-IOV) Support in Intel® Virtualization Technology for Connectivity (2008), http://www.intel.com/content/dam/doc/white-paper/pci-sig-single-root-io-virtualization-support-in-virtualization-technology-for-connectivity-paper.pdf (Accessed April 3, 2017)

6 Intel PCI-SIG SR-IOV Primer (2011), https://www.intel.sg/content/dam/doc/application-note/pci-sig-sr-iov-primer-sr-iov-technology-paper.pdf (Accessed April 3, 2017)

7 JDM User Guide for NFX250 Network Service Platform (2017), https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/nfx-series/nfx250-jdm-user-guide.pdf (Accessed April 3, 2017)