Fredrik Andreassen

Experimental assessment on routing of network traffic in software using commodity hardware

Master's thesis in Communication Technology
Supervisor: Yuming Jiang and Ivar Arnesen
May 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

NTNU
Kunnskap for en bedre verden

Fredrik Andreassen

Experimental assessment on routing of network traffic in software using commodity hardware

Master's thesis in Communication Technology
Supervisor: Yuming Jiang and Ivar Arnesen
May 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

Title: Experimental assessment on routing of network traffic in software using commodity hardware

Student: Fredrik A. Andreassen

Problem description:

Traditionally, network routing is done with specialised hardware capable of routing line rate traffic in all directions. Network softwarization is the trend of doing networking in software, using commodity hardware instead of specialised hardware. Specialised equipment is expensive, so using commodity hardware instead could reduce the cost significantly. In the past years, a lot of research has been done in this field and made this possible. For instance, for software routing, both complete solutions and solutions that are still in the research phase exist.

The purpose of this project is to identify different kinds of software routers, both complete solutions and solutions in research phase, and do an assessment on these based on experimental testing. These routers will be assessed using test specific traffic and realistic traffic, and compared based on throughput achieved and the resource consumption used.

Responsible professor: Yuming Jiang, NTNU IIK

Supervisor: Ivar Arnesen, Ivar Arnesen Invest AS

Abstract

Network packet processing in software on commodity hardware is the main topic of this master thesis. The transition of moving the packet processing from specialized hardware into software has had a rapid improvement over the last years, primarily driven forth by the telecommunication industry. In the development of the new 5G mobile network, the ability to do packet processing in software has been pointed out as a critical factor for its success: moving the network functions into a virtualized environment on commodity hardware.

More specifically, this thesis will focus on doing software routing of network traffic in a virtual environment on commodity hardware. Different kinds of challenges by doing this has been identified and different types of software solutions have been evaluated that can accomplish these tasks. Four different routers have tested in an experimental testbed, with focus on the performance. The four tested routers have been the Cisco CSR 1000v, pfSense, VPP and the OVS router. The experimental testing includes some test traffic to push the router to their limits, near realistic traffic and some test to see how they perform against specialized physical hardware.

All of these routers can perform at speeds close to 10 Gbit/s, and some of them are able to perform way beyond this. Out of these four routers, the VPP is the one that has the best performance and shown that routing in software at high-speeds is possible today.

Sammendrag

Prosessering av nettverkstrafikk i programvare på standardisert maskinvare er hovedtemaet for denne masteroppgaven. Overgangen fra å flytte prosesseringen fra spesialisert maskinvare til programvare har hatt en drastisk utvikling de siste årene, spesielt drevet frem av mobiltelefoni industrien. Det å kunne gjøre pakke prosessering i programvare har blitt pekt ut som en nøkkelfaktor for utviklingen av det nye 5G mobilnettet. Det vil si å flytte nettverksfunksjoner til virtuelle miljøer på standardisert maskinvare.

Mer spesifikt så vil oppgaven ta for seg nettverks ruting av trafikk i et virtuelt miljø på standardisert maskinvare. Flere utfordringer knyttet til akkurat dette har blitt identifisert og forskjellige typer programvare som kan gjøre ruting har blitt evaluert. Totalt har 4 forskjellige programvarerutere blitt evaluert og ytelsestester har blitt gjennomført på disse i en testlab. De fire ruterne som har blitt testet har vært Ciscos CSR 1000v, pfSense, VPP og OVS. Den eksperimentelle testingen har inkludert test-trafikk som skal presse ruterne til sine grenser, trafikk som er så reel som mulig og en test for å se hvordan de gjør det i forhold til en spesialisert fysisk ruter.

Resultatet av disse testene har vært at alle ruterne klarer å prosessere trafikk opp til 10 Gbit/s, og noen langt over dette igjen. Av disse 4 ruterne er VPP ruteren den som har hatt den beste ytelsen og vist at den er kapabel til å gjøre ruting i programvare med dagens krav til ytelse.

Preface

This master thesis is a completion of the specialisation in Information Security and Master of Science degree at Norwegian University of Science and Technology (NTNU). The two years I have spent at this university have given me the knowledge and experience to complete this master thesis.

Before I started, I had very little knowledge of how packet processing is being done in software. I had some experience with hardware routers and the configuration of these, especially Cisco routers. This has been very helpful in order to understand and configure other kinds of routers. It has been fun to experiment with these different kinds of software routers and to see how they utilise various types of technology to get high performance.

I want to thank my responsible professor, Yuming Jiang, and my supervisor, Ivar Arnesen, for their support and guidance during this thesis.

Fredrik Andreassen
Trondheim, Norway
May, 2019

Contents

List of Figures

List of Tables

List of Acronyms

ACL Access Control List.

ASIC Application Specific Integrated Circuits.

AWS Amazon Web Services.

BGP Border Gateway Protocol.

CLI Command Line Interface.

COTS Commercial Off-The-Shelf.

CPS Connections Per Second.

CSR Cloud Service Router.

DHCP Dynamic Host Configuration Protocol.

DMA Direct Memory Access.

DPDK Data Plane Development Kit.

DPI Deep Packet Inspection.

DUT Device Under Test.

FIB Forwarding Information Base.

GCP Google Cloud Platform.

GUI Graphical User Interface.

HLR Home Location Register.

HSS Home Subscriber Server.

IDS Intrusion Detection System.

IEEE Institute of Electrical and Electronics Engineers.

IMS Ip Multimedia Subsystem.

IP-FW IP firewall.

ISP Internet service provider.

LB Load Balancer.

MPLS Multiprotocol Label Switching.

MTU Maximum Transmission Unit.

NAT Network Address Translation.

NFV Network Function Virtualization.

NIC Network Interface Card.

NTNU Norwegian University of Science and Technology.

NUMA Non-Uniform Memory Access.

OS Operating System.

OVA Open Virtual Appliance.

OVS Open vSwitch.

PPS Packets Per Second.

RSS Receive Side Scaling.

SCTP Stream Control Transmission Protocol.

SDN Software Defined Network.

SFP Small form-factor pluggable (transceiver).

SR-IOV Single Rooted I/O Virtualization.

SSH Secure Shell.

STP Spanning Tree Protocol.

TCP Transmission Control Protocol.

TP Twisted Pair.

UDP User Datagram Protocol.

VLAN Virtual LAN.

VM Virtual Machine.

VNF Virtual Network Function.

VPN Virtual Private Network.

VPP Vector Packet Processing.

Chapter 1
Introduction

The topic of this report is about softwarization, which is about doing networking in software on commodity hardware instead of specialized hardware. The main focus will be to see what challenges there is by doing packet processing in software and finding different kind of solutions that is able to do this. Specifically, software routers that can do layer 3 forwarding will be investigated. The second main focus is to do a performance test on these software routers in a virtual environment.

The main reason for moving the networking into software is flexibility, better resource utilization and to make it hardware independent. The telecommunication industry has been the driving force for bringing this technology forth, and mainly because of the development of the new 5G mobile network. One of the main goals of the 5G network is to be able to offer differentiated services like low latency, high throughput and both low latency and high throughput combined. The use of Network Function Virtualization (NFV), and then specifically virtual routers in software, really help achieve these goals. Combined with Software Defined Network (SDN), it is possible to make different kind of network flows take different paths in the network, for instance pass by different types of virtual routers.

Regarding the 5G network, virtual routers are not the only functions needed. However, the most important part is to be able to process large network traffic loads in software. It is not only the telecommunication industry that can take advantage of this technology; all other places that use routers could potentially use virtual routers. The cloud providers have begun to use this technology heavily, and providers like Amazon Web Services (AWS), Google Cloud Platform (GCP) and Microsoft Azure is already started using virtual routers to some degree.

SDN is the idea of separating the data plane (the forwarding of packets) and the control plane (controlling how the data plane is forwarding packets). Much research has been done on in this field, but mainly the controller is the only part that has been taken out of physical hardware and put into software. The data plane is still,

in most cases, being executed by specialized equipment. In softwarization, the data plane is also moved into software, and this introduces many challenges. Dedicated hardware that does packet processing, often uses specialized Application Specific Integrated Circuits (ASIC) to perform most of the processing and does this very efficiently. ASIC is also much more expensive than CPUs, as it is produced in much smaller quantities and this gives a high production cost. Commodity hardware and CPUs is produced in millions of units, which helps to keep the production cost down. CPUs is built for multipurpose operations, so doing the processing will not be as efficient as in ASIC. However, commodity hardware has become quite powerful, and with multi-core CPUs, it has become possible to do networking in software quite efficient and achieve throughput needed for to days high-speed networks.

To be able to determine how far the development has come in softwarization and virtual routers, performance tests are needed on different kinds of software routers. This has to be tested using realistic and test-specific network traffic, to see the actual throughput that these routers can perform. To see how they perform in a NFV setting, the routers will be virtualized and tested in a virtual environment. The first objective of this thesis will be to research what kind of software routers that are available and the challenges with doing this in software compared to doing it in specialized hardware. The search for routers will include commercially available software, open-source routers and routers that are still in the research phase. The second objective will be to install and configure these routers in a virtual environment and do the same performance tests on them, so the results are comparable.

1.1 Motivation and Thesis Scope

Specialized hardware network equipment that uses ASIC is quite expensive compared to multi-purpose hardware. If the networking could be moved to commodity hardware, the cost gain perspective is potentially huge. However, it is not only the cost perspective that is a great advantage, but also the flexibility it offers. Softwarization and virtualization are considered as two disruptive paradigms and the basis in the design process of 5G networks [CM18]. In all the use cases of the 5G network and the different kinds of slices proposed, it would be extremely costly and hard to maintain if a lot of specialized physical equipment is used. Softwarization, SDN and NFV could potentially solve all of these problems.

For virtual routers, the manufactures and developers all states different kind of specifications for their product, and these data is not always directly comparable. The knowledge of how different types of routers compares to each other, specifically as a Virtual Network Function (VNF), does not exist today. This new knowledge could shed some light on how different kind of routers perform in different settings and using different types of functions. The performance of these routers is also

compared with some specialized hardware, to give some perspective on the difference between software and hardware ASIC packet processing.

The scope of this report will be to find some software routers that can be virtualized, and do some performance tests on these. Some of the routers offer the possibility for configuration and performance tuning, so this will be done to get the best performance possible. This will not involve changing the source code and writing own code, but changing the configuration parameters available. An RFC exists for doing performance tests on network equipment [IET], but it does not fully specify everything in the tests that should be done. The test parameters have to be defined on how the performance test should be done, and some software/equipment for the traffic generation/measurement has to be explored. The scope is limited to only testing layer 3 forwarding, purely layer 2 forwarding is not considered [1].

1.2 Research questions

The main research questions, based on the problem description, are as stated:

1. What are the challenges of doing fast packet processing in software, and how can these challenges be overcome?
2. What is the performance of different kinds of software solutions for routing that is available today when they are used as VNFs?

1.3 Contributions

This thesis provides the following contributions:

- An example of a virtualized experimental testbed setup for performance testing of software network routers.
- Script for automated performance testing
- Comparable test results of different kinds of software routers used as VNFs

1.4 Thesis Outline

The master thesis is structured as follow:

- **Chapter 1** is an introduction to, motivation for and a scope of the thesis work.

[1] The layer 2 performance for the VMware ESXi hypervisor is indirectly tested during the VNF chaining tests

- **Chapter 2** introduces some background for the work and some theory of technologies relevant for the thesis.
- **Chapter 3** describes the choice of methods.
- **Chapter 4** presents the experiment design selected and how this is implemented in the test lab.
- **Chapter 5** presents how the tests are to be conducted and the evaluation methodology chosen.
- **Chapter 6** presents the results from the experimental testing
- **Chapter 7** discuss the theory and the results from the thesis.
- **Chapter 8** gives a conclusion on the thesis work, and some suggestions for future work on this topic

Chapter 2
Background/Theory

2.1 NFV and SDN

NFV is a way to decouple the network functions that traditionally is being done by specialised and proprietary hardware, into software that can be run on infrastructure independent from the hardware underneath. The telecommunication industry has been a driving force for NFV, with especially focus for the new 5G mobile network. This means that the use of generic hardware Commercial Off-The-Shelf (COTS) is preferred, instead of much more expensive specialised hardware. These network function can be:

- IP firewall (IP-FW)
- Network Address Translation (NAT)
- Load Balancer (LB)
- Proxy
- Deep Packet Inspection (DPI)
- Intrusion Detection System (IDS)
- virtual switch/router
- mobile network specific services like Ip Multimedia Subsystem (IMS)/Home Location Register (HLR)/Home Subscriber Server (HSS)

They are run as applications (VNFs) on top of a virtualised layer, often as Virtual Machine (VM) on top of a hypervisor (Figure 2.1). This gives the VNF the same flexibility as other VMs, like elastic capacity, better utilisation of resources, portable, hardware independent and the possibility to make them more fault tolerant [Shi18].

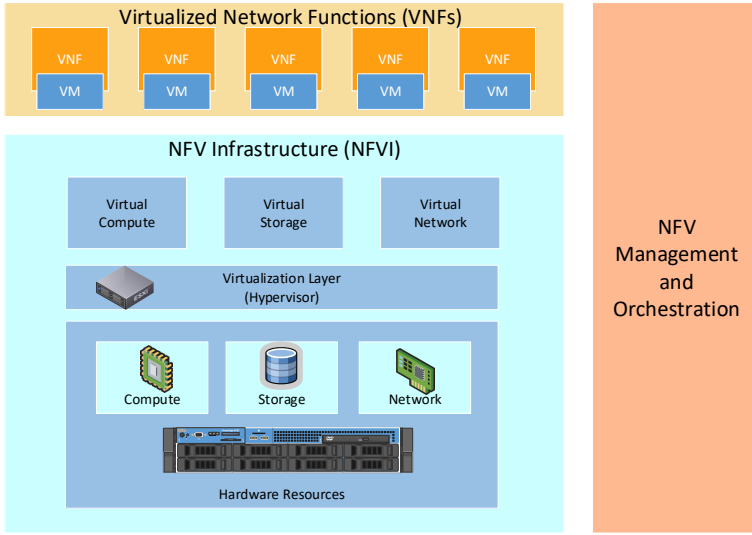


Figure 2.1: NFV Architecture

SDN is the decoupling the control plane from the data plane, which means that the device that is controlling how traffic is to be forwarded (controller) is separated from the devices that are doing the forwarding (Figure 2.2). One controller can manage multiple devices and can be centralised, which offers the possibility for more smart and sophisticated networks. Both SDN and NFV are promising driving technologies to dis-aggregate the traditional vertically integrated systems into components by using softwarization[Shi18]. These components consist of both software and hardware. Network softwarization is the trend of doing networking in software, using commodity COTS hardware instead of specialised hardware.

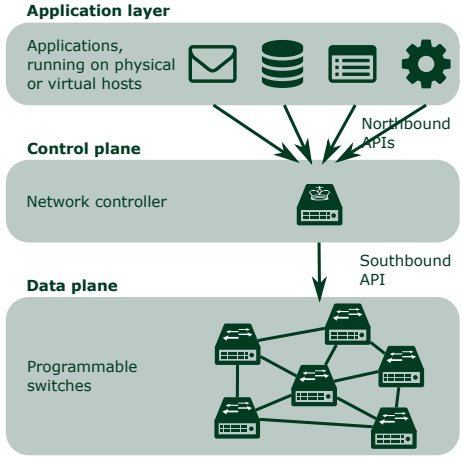


Figure 2.2: SDN overview [Mon]

A problem with NFV is that it could lead to large variations in latency and erratic throughput. An example of this was the findings Wang did find in the Amazon EC2 cloud[WN10]. They use virtualised software routing as a VNF, and it resulted in very unstable TCP/UDP throughput. The delay variations were about 100 times larger than most propagation delays, which is around 0.2 ms. Another problem with VNF in VM, is that it has slow instantiation time and relatively large memory footprint, because an entire Operating System (OS) has to be run for each VNF. The use of containers instead of VM has shown to overcome a lot of the problems with VMs, and is promising for further use with VNFs [NLH+15]

VNF chaining is an important concept within NFV. It offers the possibility of chaining different functions (VNFs) within the same physical machine (see Figure 2.3). The VNFs in this setting could be a vRouter, an IP-FW and an IDS. The internal boxes are virtual networks created by the hypervisor (see Section 2.4) and are just an internal network within the physical machine. Only VNF1 and VNF4 are directly connected to the physical world (physical Network Interface Card (NIC)s). This offers the flexibility to easily add/remove functions in a network path, and combined with SDN the controller could determine what kind of VNF different kinds of flows should go through.

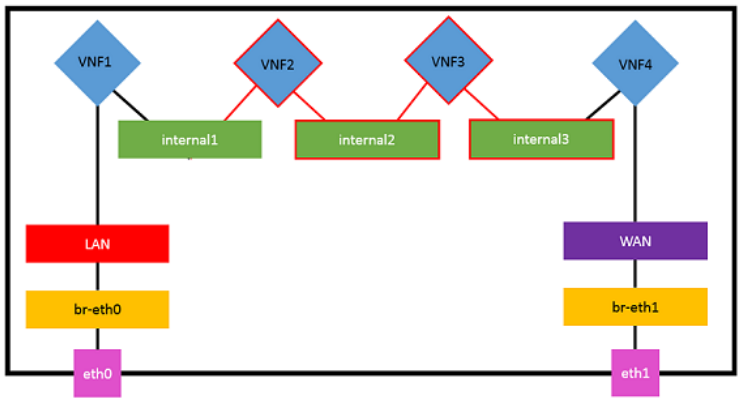


Figure 2.3: VNF chaining [25]

2.2 Challenges with routing in software

When doing routing in software, there is many challenges that have to be overcome. Usually routing is being done in specialized hardware designed to do just that, but now the data plane (Figure 2.2) is moved into software. When doing routing in software on commodity hardware, this becomes a problem since it is designed for doing more generic tasks. This means that it would not be nearly as efficient to do packet processings as specialized ASIC. The CPU is always in any cases the bottleneck when processing packets in software [BRR+16]. Other factors like Memory (RAM, CPU cache), NIC functionality, PCIe bus lanes and Non-Uniform Memory Access (NUMA) locality for CPU cores is also important but is rarely the bottleneck. For instance at 10 Gbit/s, with the smallest frame size of 64 bytes, the time between packets is 67,2 ns (at 14.88 Million Packets Per Second (PPS))[1]. On a CPU with 1,7 GHz clock speed, this means that it has approximately 114 [2] CPU cycles per packet. To compare, a single cache-miss takes 32 ns, userspace system call is between 40 and 75 ns and RAM access is around 70 ns[Bro15]. 67,2 ns is not much processing time for each packet, and you don't have time to do unnecessary operations.

When routing network traffic, there is often a lot more functionality than just routing a packet from one interface to another. Firstly, a look up in the routing table has to be done to determine what interface to send the packets. On the routers on the internet, the table can have around 0,5 - 1 million entries, which is many entries to look through. Other services like Access Control List (ACL), NAT, Virtual Private

[1] Total size of the frame on the wire is 84 bytes (64 bytes + 7-byte preamble, 1-byte start-of-frame delimiter and 12-byte inter-frame gap). 10 Gbit/s / (84 bytes * 8 bit) = 67,2 ns. (84 bytes * 8 bit) / 10 Gbit/s = 14.88 MPPS

[2] 67,2 ns * 1,7GHz = 114,24 Cycles per packet

Network (VPN) and IDS can be performed by a router, and requires additional resources to be processed. Again specialized hardware is designed to do these tasks, but commodity hardware is not. So to be able to do routing in software efficiently, optimisation in the way the packet processing path is needed. This can for instance be:

- Multi-threading (more CPU cores)
- optimise the use of CPU memory cache
- reduce cache line misses
- batch processing of packets
- pre-fetching instructions from memory ahead of time
- offloading some packet processing to the NIC

The most obvious choice is to increase the number of CPU cores, as a typical modern server CPU has around 20-30 cores pr. socket. This means that eventually the address bus between the NIC and the socket will be a bottleneck, but we are then talking about throughput far beyond 10 Gbit/s.

2.3 Kernel bypassing

One of the most important thing when doing fast packet processing is to bypass the kernel network stack. This means that the packets are delivered directly from the NIC to userspace, with minimal kernel interference. There are many reasons for this, but the main reason is that the kernel is not able to process that kind of packet that is needed when moving beyond 1 Gbit/s. Studies have shown that using kernel bypassing framework like Data Plane Development Kit (DPDK) is 12 times faster than using the Linux kernel for packet processing [ERWC15]. For instance, the Linux kernel can process between 1-2 MPPS, which is a far behind the line rate of 10 Gbit/s (14.88 MPPS). Other advantages of doing the processing in userspace are that it is much easier to write some new code and gives the possibility for a fast upgrade. The kernel has to be stable, which mean that changing the code is not something that is done overnight since it requires revisions thoroughly. Changing the code in userspace is much easier as it is just a process, and can also be updated/restarted easily without the need for a full system restart. Figure 2.4 show how the Linux kernel is bypassed with DPDK and the userspace applications are interacting directly to the NIC

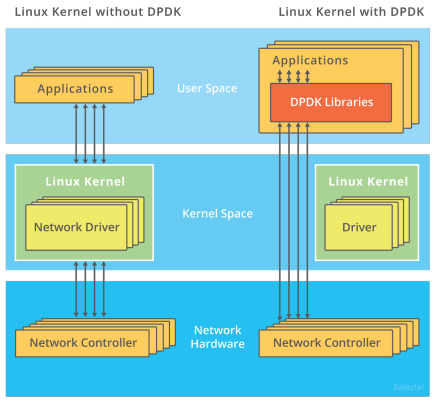


Figure 2.4: DPDK Kernel Bypassing [AY]

Intel DPDK is considered the de-facto standard for kernel bypassing[BBGJ18], to be able to do fast-packet processing in commodity hardware. It is an open-source project managed by Intel and supports a variation of NICs (not only the one from Intel). There exist other options as well like PCAP, PF_RING DNA, netmap and HPCAP. These were tested and compared, and PF_RING DNA and Intel DPDK were the only ones that could achieve line rate for 10 Gbit/s when fewer than 4 receiving queues (4 CPU core) where used[MRdR+15]. One large disadvantage with DPDK, is that pulling is used to receive packets from the NIC. This means that 100% of the core is utilised, even if no packets are incoming. The Linux kernel uses interrupt mode, where the CPU is only involved when the packets are incoming.

DPDK uses huge memory pages in RAM, which is much larger than default memory allocation. When increasing the chunk of memory, the time to find where a particular memory is mapped is decreased significantly, since the Page Table Entry is much smaller. A userspace application can write and read data directly into these pages using Direct Memory Access (DMA). DPDK also uses a NUMA aware design, which is the memory design for each CPU socket. This ensures that each packet that is processed stays within the same local memory bank. Each processor may in a NUMA design access its own chunk of memory in parallel, which significantly improves the performance and reducing the CPU data starvation problem [MRdR+15]. If CPU has multiple sockets (therefore multiple NUMA nodes), it is important that the NIC is attached to a PCIe slot assigned to the socket that is processing the packet. This to avoid having to transmit data between sockets, degrading system performance (illustrated in Figure 2.5).

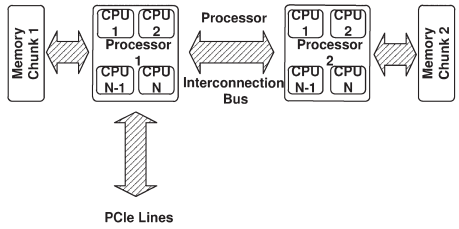


Figure 2.5: NUMA architecture with 2 sockets [MRdR+15]

The NIC has some important features that can be utilised by DPDK and applications, and the most important is the Receive Side Scaling (RSS) feature. The main feature for RSS is to distribute incoming packets to different queues (CPU cores). The distribution is based on a hash value calculated from IPv4/6 source and destination address, protocol fields and TCP/UDP source and destination ports. This should ensure that sessions are processed by the same CPU core, to minimise jitter. Other features from the NIC that can be utilised is the calculation of the checksum for IPv4 and TCP/UDP packets. This is done much faster in hardware, and this can be offloaded to most NIC.

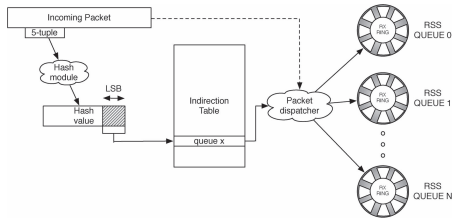


Figure 2.6: RSS architecture [MRdR+15]

2.4 Virtualization

Virtualization is a key component to be able to virtualize the layer 3 forwarding (routing). There is a lot of virtualization software to choose from like Microsoft's Hyper-V, VMware's ESXi, Citrix XenServer or Oracle VM. These are bare-metal hypervisors, which means that they are installed directly on the server without any other underlying OS. Each of the solutions offers different kinds of features and licensing options. VMware ESXi is the one offering the most advanced features, but also the one that has the highest licensing costs[Sie]. The hypervisors are used in combination with VMs and are not used when using containers.

When doing routing in software, it requires a lot of network I/O. A problem when virtualizing is that some overhead between the hypervisor and the guest OS/VNF can occur when the network I/O are copied. When using DPDK, it requires to be able to read and write directly to the memory using DMA. This access can have lower performance when doing virtualization compared to a native network I/O[Shi18].

VMware ESXi and DPDK supports two kinds of connections to the physical NICs, which both bypasses the hypervisors layer for network I/O (vSwitch)[VMw]. They are DirectPath I/O (PCI passthrough) and Single Rooted I/O Virtualization (SR-IOV) (Figure 2.7). DirectPath I/O gives a VM direct and exclusive access to a NIC, which generate the least overhead of I/O between NIC and VM. SR-IOV does not give exclusive access to a NIC, so multiple VM can access a single NIC. DirectPath I/O can be considered as a hypervisor bypassing, and SR-IOV is almost the same as bypassing. A small switch in the hypervisor is added when using SR-IOV that uses MAC addresses to identify the different VMs. The NIC still appear as a physical PCIe device to the VM and all the functions from the physical NIC is available through DMA. This small switch requires some computational power from the hypervisor, as all of the MAC addresses for the traffic has to be read, to deliver the frames to the correct VM. However, the computational power and generated overhead are much smaller than the network stack integrated into the hypervisor (vSwitch for VMware).

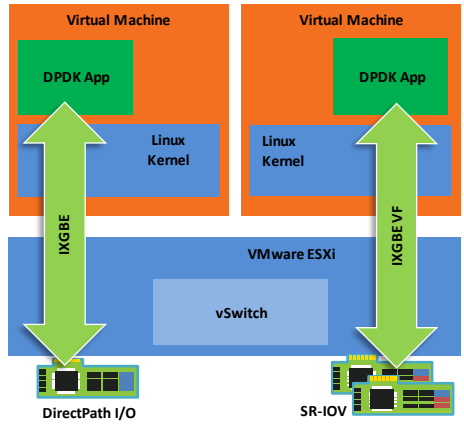


Figure 2.7: Intel DPDK and VMware NIC direct assignment[VMw]

2.5 Why use software routers

Routers are in use all over the world and are essential in today‘s internet architecture. Traditionally this has been done in specialised hardware routers (because of the challenges stated in section 2.2). However, with to days use of the cloud infrastructure and the coming 5G mobile network, the use of virtual routers in software as a VNF

is essential. There are many advantages by virtualizing the routers, and the biggest is that it utilises the hardware better. Other advantages with vRouters as VNF is that it is possible to have multiple routers on a single hardware, making it possible for instance to have one router per segment (slices) in a 5G network, depending on the requirement for that segment (delay, throughput).

Many functions in a router are used, depending on how/where the router is used. A core router for an Internet service provider (ISP) uses different functionalities than branch and home routers do. These functionalities can be dynamic routing protocols and Multiprotocol Label Switching (MPLS), different VPN technologies, IP-FW, and Dynamic Host Configuration Protocol (DHCP). A core router often uses Border Gateway Protocol (BGP) as a dynamic routing protocol with large routing tables with around 500-800k entries [BGP]) and with very high throughput. A branch router often uses VPN to connect to different sites and uses some security functions, but the throughput and routing table is not near that of a core router. Also, a typical home router often has quite a few functions in use, and the throughput is quite low (1-1000 MBit/s). So the requirements for different use cases of the routers vary quite a bit. A physical router often has all the functions built in, often in ASIC. This means that if an ACL is used in the router, there is almost no performance degrading because of the use of ASIC. In a software router, only the functions needed are used. The software router can then be tailored for a specific purpose. However, for each function that is added, more computational power from the CPU is required. That is one of the reasons why a software router is not always preferable, it will depend on the use case.

2.6 Software routers

There exists a lot of software and OSes that are capable of doing routing. For instance, both Linux and Windows can do layer 3 routing. Other OSes like Cisco IOS, which originally is used on physical hardware, has been designed to be run in a virtual software environment. However, most of the available high-speed software routers are designed to be run in Linux, with kernel bypassing frameworks. Most software routers can handle traffic up to 10 Gbps, but what packet sizes that this depends on is often not stated by the developers.

2.6.1 VPP

VPP is a platform that is designed by Cisco and is released as an open-source project as part of FD.IO. FD.IO is a collection of several projects with the goal of optimising networking in commodity hardware [FD.a]. It utilises the DPDK framework to bypass the kernel, and the main advantage with VPP is that it is batch processing all the packets (in vectors). This makes it possible to save a lot of CPU cycles since

the same processing is done with similar packets. The first packet in a vector then "warms up" the instruction set for that vector, and when the vector is full/time limit reached, the entire vector is processed as one. This make is possible to achieve 14+MPPS on a single core with layer 2 forwarding, which is close to the wire rate for 10 Gbps (14.88 MPPS). For IPv4 layer 3 forwarding it has a capacity of processing 9 MPPS [FD.e].

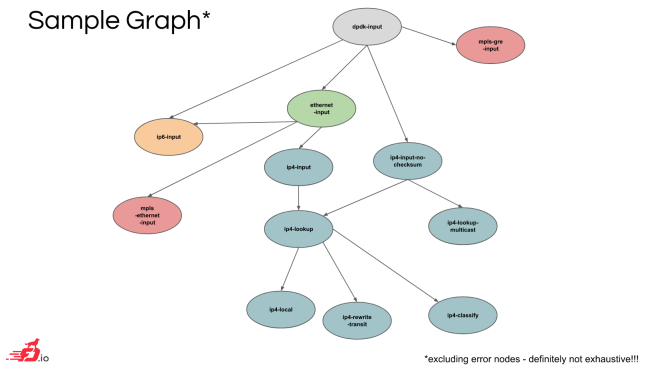


Figure 2.8: VPP Sample graph of plugins[FD.c]

The VPP framework runs on Linux and supports the same physical and virtual NICs that DPDK supports. It has for example support for L2/L3 forwarding, MPLS, IPv4/IPv6, VLAN, ACLs and multi-million entry for the Forwarding Information Base (FIB). All these functions are realised with plugins and a graph tree which the packets are processed (see Figure 2.8). So each vector of packets is processed by one node before it is passed to the next one. This makes this architecture very modular and flexible, to add a new feature or modify an existing one, just modify the plugin(s). One such feature could be a firewall or IDS. Researchers have shown that VPP is a fully functional router, well suited for NFV applications on commodity hardware [LRP+17]. It is also capable to handle on a single server over 100 Gbps, only limited by the PCIx express bus. The key factor for VPP to be able to achieve these throughputs is the low-level vectorized processing primitive. This increases both the data cache hit rate and the instruction cache hit rate.

VPP uses polling mode when using the CPU, which means that 100% of the core is utilised when in use. This offers more efficient use of the CPU on high loads since it does not have to do any other processing. However, if the load is low, the efficiency is not very high since the CPU utilisation is still 100%. The alternative to pulling mode is interrupted mode, where only the CPU resource necessary is used. However, this mode is not as efficient with high loads as pulling mode. It is possible to configure

VPP to use interrupt mode instead of pulling mode.

2.6.2 OVS

Open vSwitch (OVS) is virtual network switch, primary designed to be used in a virtual environment and provides a fast data-plane in userspace. It offers the possibility to interconnect VMs with high throughput and rich with features like Spanning Tree Protocol (STP), OpenFlow, Virtual LAN (VLAN), NIC bonding, multiple tunnelling protocols and more [Pro]. It can (like VPP) take advantage of DPDK to be able to move the packets directly from the physical NIC to userspace and the vSwitch. Tests have shown that OVS can process packets at wire-speed for 10 Gbps (14.88 MPPS) [Cha]. Other similar project like Vosyswitch (based on OVS can perform even better than OVS when doing L2 forwarding with small frames [PFNR16].

Since OVS supports protocols like OpenFlow, it can be controlled by a SDN controller and therefore do L3 forwarding. This makes OVS capable of being a fast vRouter since it takes advantage of the fast data-plane provided by the switch. The functions of the vRouter are then decided by the SDN controller used. It can then be used as a VNF in a NFV setting. Like VPP, OVS uses polling mode for the CPU.

2.6.3 Cisco 1000v CSR

Cisco 1000v Cloud Service Router (CSR) is a router specially designed to be used in the cloud in a virtual environment. It has almost the same functions as Cisco's physical routers, and it runs the same IOS. The Command Line Interface (CLI) is the same, so for a network administrator, it is easy to switch from configuring physical hardware to the virtualized software. Based on the licensing and features/functions needed, the Cisco 1000v supports up to 10 Gbit/s network speeds with the configuration of 2 CPUs and 4 GB RAM [Cisa]. It is not stated what packet size that is used when supporting 10 Gbps. At that speed, only basic IP features are supported (the IP Base license) like dynamic routing, tunnelling and basic networking. The router is supported by the public cloud providers AWS, GCP and Microsoft Azure. The most common hypervisors for virtualization is also supported, including VMware ESXi. The Cisco CSR is not a free product, but they offer a fully functional 60 days trial, and this is used in this thesis. VPP is stated to be the packet processing in the software [Cisc].

2.6.4 pfSense

pfSense is a free network firewall distribution and based on the FreeBSD operating system. It is like the Cisco CSR a complete solution with OS and software, and is capable of being run in a virtual environment. It is officially supported by the cloud

providers AWS and Microsoft Azure and supports the most common hypervisors [Netb]. pfSense also comes with physical hardware where the software is pre-installed. The current version (2.4.4) used a modified Linux kernel for the network stack, so the data plane has almost the same limitations as the Linux kernel. In the road map for pfSense, it is stated that support for DPDK and VPP is planned, so the limitation in the Linux kernel will be eliminated in the future [Neta]. Netgate also has a software TNSR, which is based on DPDK and VPP for the dataplane. This is not an open and free solution.

2.7 Packet generation and testing

To be able to test and compare different kinds of vRouters, a key component is a traffic generator capable of generating enough traffic to test the performance. Then this traffic has to be measured with respect to throughput and latency. Ideally, this measurement is done by the traffic generator and not by the vRouter itself, to get comparable results. Moreover, ideally, the generator is a hardware ASIC, since it has improved performance over software generators, and is less affected by other factors like hypervisor overhead (when virtualized).

When measuring throughput on routers, the most common measurements units are the number of bits/bytes that is pushed through and the number of packets processed (PPS). Since the packets sizes vary based on the traffic, PPS is a good measurements unit when comparing routers. The CPU is in almost any case when processing packets in software the bottleneck [BRR+16]. Since memory is often in abundance on commodity servers nowadays, the actual packet size does not affect the overall performance much, but the number of packets processed does. When testing router performance, the smallest packet size of 64B (84B on the wire) is used to get the actual PPS performance. The stateless protocol User Datagram Protocol (UDP) is used, since it does not set up a connection (like Transmission Control Protocol (TCP)) and it is easier to have control on whats actual transmitted (no retransmissions and congestion). By using UDP the routers maximum performance will be around where packets start to get lost (packet loss). However, it is still possible to get a higher throughput even when packets are starting to get dropped by the router.

Many router manufactures reports that their products support 10 Gbps or 40 Gbps, but does not supply the information on what packet sizes this applies for. Most normal traffic does not include just 64B packets, but often quite larger than this up to 1518B packets [3].The average packets size for internet connections was measured on Campus traffic in 2018 to 870B [JRB18]. 870B packet sizes is a realistic size to

[3]Maximum Ethernet Maximum Transmission Unit (MTU) (1500B + 18B header), from the IEEE802.3 specification

use to test the router for a realistic throughput, in the number of bytes. To get an even more realistic measurement, the use of real traffic like http/mail/dns is ideal in a server/client setting. This does not just use UDP for transport protocol for transmission, but also distributes the traffic on the ports more like a server/client setting (more traffic from the server to the client, than the other way around).

The overall performance is also affected by what kind of features/functions that are being used on the router. This can be the size of the FIB (routing table) or the number of ACLs that is in use for instance. These factors can greatly impact overall performance, especially in software routers. In hardware routers, these tasks are often done in ASIC, so if you have 0, 1 or 1000 ACLs, it does not affect the overall performance. In software, on the other hand, the performance with 0 or 1 ACL could affect the performance greatly. This is because when there is no ACL, no packets is being checked against any ACL, and that actual ACL process/plugin/component is not run. However, as soon as you have an ACL, each packet has to be checked, and this requires extra computation. Look-up in the FIB has to be done anyway when doing routing, but the size of the table could have an effect on the performance. To compare, the BGP internet IPv4 routing table had in Januar 2018 700 000 entries [Hus].

2.7.1 TRex

TRex is an open-source software traffic generator designed and developed by Cisco. It runs on Linux, uses DPDK for kernel bypassing and supports up to 20 MPPS packet generation [Tt]. It can generate realistic traffic up to 200-400 Gbps (depending on hardware) and has built-in measurements for latency and jitter. It comes pre-shipped with traffic templates to use for testing, like 64B/1518B UDP packets and HTTP/HTTPS traffic. It also has the option to add own templates and pcap files[4], so the traffic generated can be custom made.

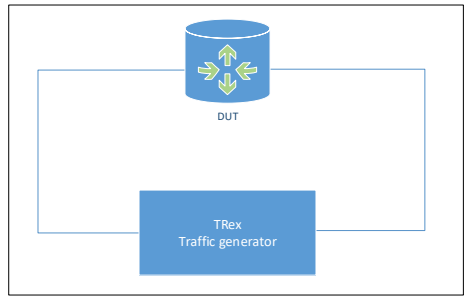


Figure 2.9: Test topology, defined by RFC 2544[IET]

[4]Network traffic dump

TRex simulates any number of servers/clients needed and rewrites the IP source/destination from the pcap files. This makes it ideal to test large routing tables where all the entries are used. This means that if the routing table contains 1k entries, TRex can simulate 1k servers/clients. For latency measurements, the Stream Control Transmission Protocol (SCTP) is used and sent in addition to the other traffic with a defined interval. This allows for an accurate measurement using the time stamps on the packets, and since the sender and the receiver are within the same machine.

Chapter 3
Method

The following chapter presents the separate phases of the work and choice of method. There are two main methods within the scientific method of research, qualitative and quantitative. These will be explained in the following sections.

3.1 Qualitative method, literature study

The qualitative method literature study was chosen to gather information about the topic area. The literature study aims to get a better understanding about the reason packet processing in software is needed, the challenges related to doing it and how this kind of research could benefit for further development into the field. There is a lot of research being done in the field of software packet processing and NFV, both by academics and commercial parties. The focus is often very different, ranging from a detailed description of one of the building blocks needed, to different combination of technologies and techniques to complete solutions. To get an overview and a real understanding of how far the development has come, the literature study has to be quite extensive to get an overview of the topic. The Institute of Electrical and Electronics Engineers (IEEE) has published a lot of research regarding this topic and will be the primary source of information. However, other scientific publication sites will also be used. Public websites for the software routers, guides and forums will be used to learn about the software, but also how to install and configure them.

The result of the qualitative method will be a good understanding of how packet processing in software is achieved, challenges with doing this and where the technology is today. It will result in 3-5 different solutions/products that can do routing in software. They will be routers that are fully working, able to be virtualised and to be configured according to RFC2544 [IET]. Each of these routers will have a specification on performance, but they are most likely not comparable due to different testing criteria. A quantitative method for testing these routers with equal criteria is therefore required.

3.2 Quantitative method, experimental testing

The quantitative method will include some lab testing of different kinds of software solutions that were found in the literature study. The lab testing environment is to be mostly virtual, but two physical servers are to be interconnected using physical NICs and physical Twisted Pair (TP) cable. The actual setup of the test lab was done during the specialisation project in Fall 2018. The first thing will be to get a working traffic generator and measurement unit, to be able to verify the software routers are working. Then the different routers can be installed, configured and verified that they are working. They are also, if possible, tuned the perform at maximum (by changing the default configuration, not the source code). When the routers are confirmed working, the performance tests can be performed.

For the actual performance tests, they are divided into three different categories:

1. Different number of CPU cores, using optimal router configuration [1]
2. With 2 CPU cores, testing with respectively a large FIB and ACLs activated
3. One of the routers is multiplied into many VNFs and tested in parallel and series (chaining)

The critical data from these performance test will be the actual throughput of data the software can process and forward from one physical interface to another, the delay of the forwarding and the number of packets forwarded. For the test categories 1 and 2, the hypervisors network stack is bypassed completely (using PCI passthrough), so those test should be hypervisor independent. Category 3 uses the hypervisor network stack to pass traffic between different VMs, so that test will mostly be a test of the hypervisors ability to process packets.

[1] Minimal number of functions activated and minimum number of IP routes, to get the most optimal performance

Chapter 4
Experiment Design and Implementation

This chapter documents how the experiment testing is designed and implemented, and the setup for a virtual lab accessible from the internet. Next, it shows how the different kind of software routers where installed, configured and tuned. Lastly, the chapter documents how the various test was performed.

4.1 Lab Design and Setup

The RFC 2544 "Benchmarking Methodology for Network Interconnect Devices"[IET] has been a base for designing the performance tests used for the software routers. It defines how to do benchmarking or performance tests for network devices, including how to set up the experiment, different kinds of metrics, frame sizes to use, filters (ACLs), how many times/how long the test has to be run etc. It uses the notation Device Under Test (DUT) for the router, and the generic setup is as shown in figure 2.9. The RFC is from 1999 and is a little bit outdated, but the main concepts are being used in this experimental design.

When designing the lab, the goal was that it should be virtualized, no need for re-wiring between tests and accessible from everywhere. This was achieved by using VMware ESXi hypervisor for the virtualization part, and an IPsec VPN connection to get access to the internal management network. The network was set up as shown in figure 4.1. The firewall and IPsec server is logically placed before the physical servers, but is physically located on ESXi-host 1 in a VM running pfSense. This is just the network for managing the different kinds of servers and VMs, the test traffic from the software routers are not passed over this network.

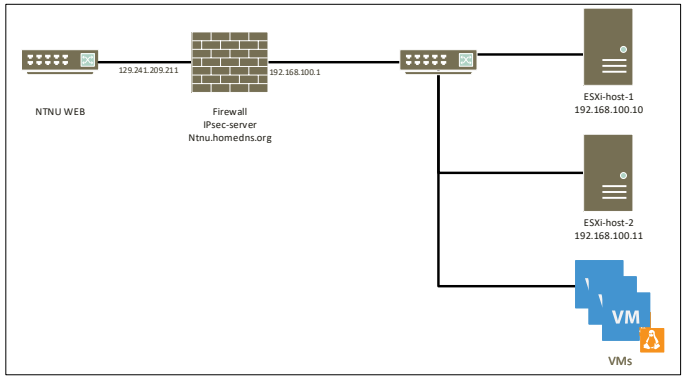


Figure 4.1: Overview Management Network in the virtual test lab

4.1.1 Physical setup

The lab was, as stated in section 3.2, set up during the specialization project period Fall 2018. Two Dell PowerEdge R730 Servers with the following specifications:

- Intel® Xeon® E5-2650LV v4 1,7 GHz, 14 cores/28 threads CPU
- 4 x 32GB RDDIM RAM
- 4 x 1TB 7.2K RPM SATA 2.5in Hot-plug Hard Drive
- Intel Ethernet x540 10Gb BT DP + i350 1Gb BT DP Network Card
- Intel Ethernet x710 Dual Port 10 Gigabit DA/SFP+ Network Card

In addition, a JH145 HPE 5510 24G 4SFP+ switch was used for the management connections and a reference test as a physical router. These pieces of equipment were connected as shown in figure 4.2. It show how the management network (yellow lines) is connected, the 10 Gbps TP test network (blue line) and single mode fibre (blue line) connections for testing the switch routing capability.

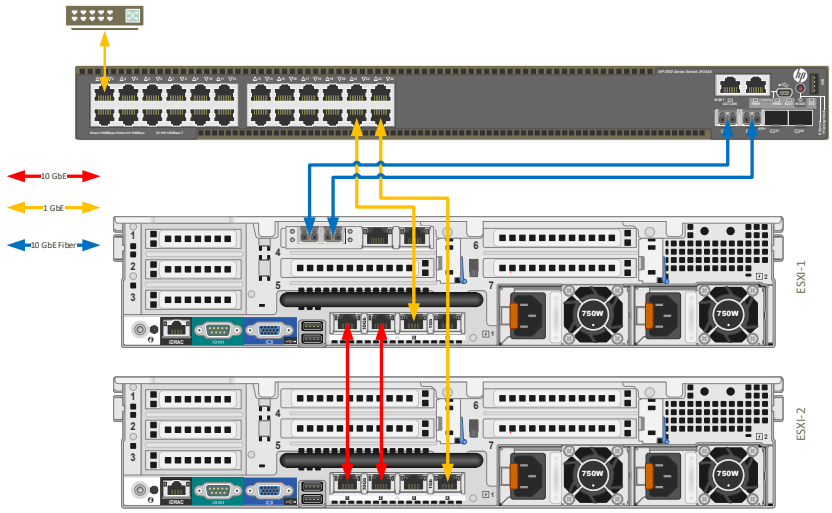


Figure 4.2: Connectivity of the physical equipment

4.1.2 Virtual environment

VMware ESXi v6.7 was used as the hypervisor for the virtual environment. This is because I have previous knowledge and training using this software, and because NTNU has student licenses available to unlock all the features of the hypervisor. VMware vCenter was installed (separate VM) to be able to administer both servers in one web interface (Figure 4.3). A virtual switch was added into both of the servers, to get inter-connectivity for the management network. The IP plan for the management network can be found in Appendix A. All the router VMs could be considered as individual VNFs.

The web interface for vCenter offers the possibilities to manage all the VMs with a lot of options. This includes basic options like the number of CPU cores, hard disk size, size of RAM, different kinds of NIC to more advanced features like specific core allocation to specific VM, RAM reservation, migration (moving between physical hosts) and duplication. This facilitates an excellent platform to perform the necessary tests on the VNFs, with the abilities to adjust the required options. The hypervisor also offers resource monitor, where it is possible to see how the resource consumption is on each VM through the Web interface. The resource monitor in vCenter is only used to see CPU consumption when the hypervisor vSwitch is in use, to see the CPU consumption for the virtual switch. For the VMs, the internal resource manager in the respectively OS is used.

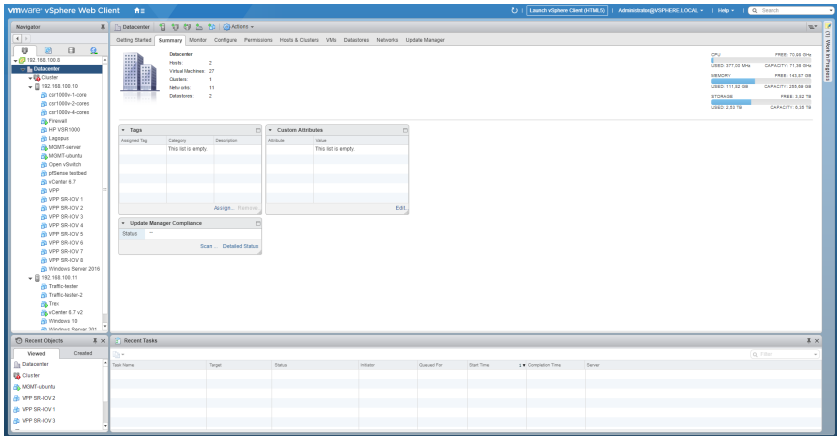


Figure 4.3: Screenshot of the web interface for vCenter

An essential part of this performance test is the connectivity to the physical ports on the NIC. There are multiple ways of doing this in VMware; one is as stated below to add a virtual switch inside the hypervisor, and connect both the physical ports and the VMs to this switch. With this method, the network stack in the hypervisor has to process all the packets in a virtual switch, which involve additional computational power and potential bottleneck when processing a large number of packets. It is also not possible for the VMs to directly access the NIC functions like RSS and IPv4/6 checksum offloading. The use of PCI passthrough and SR-IOV (described in section 2.3) gives the VM direct access to these functions and minimal/to none processing by the hypervisor.

To be able to access the management network and the test lab, a firewall has been set up with IPsec capabilities. pfSense has been used (the same software that is being performance tested, but two separated VMs). All of the software routers that are being installed and configured are equipped with three virtual NICs: two that either uses PCI passthrough or SR-IOV and one that is connected to the management network. This ensures connectivity to all the VMs, and that the management traffic is not affecting the network the tests are being conducted on. Each of the VMs is being set up with a Secure Shell (SSH) key pair, which enables the use for remote command executions. This is specifically used in the automation test script (described in section 5.3.1), to change different parameters during the tests.

4.2 Choice of software routers

During the research phase, many software routers where considered. MoonRoute was one of them, a promising software router based on DPDK capable of 14.6 MPPS

with one CPU core [GES+17]. I was not able to get the router to run in my test lab, so the router was not used in any further testing. Packet-journey is another capable router, which is also used DPDK. The performance for this one is 32.4 MPPS with 4 CPU cores, 50 ACLs and 500k routes, which is quite impressive [Git]. Like MoonRoute, I was not able to get this one to work. Lagopus is like the two mentioned above, a fast Linux router based on DPDK [IT16]. This one almost worked, but only managed to get half-duplex connections, so the throughput was not that good. For other commercially available software routers like HP VSR1000 and Vyatta (from Brocade), it was not so easy to get hold of a copy of their software for trial testing. However, they would have been good candidates for use in this test.

Cisco CSR 1000v offers a complete solution for deployment in a VMware virtual environment, in the form of a template which configures all the parameters needed. It also offers a 60 days trial for the 10 Gbps licensing, which make it ideal for this test purpose. pfSense is an open source and comes like Cisco with a complete solution. It is a popular software router (mostly because of its firewall qualities) and will offer a good comparison with other routers. It also mostly uses the Linux kernel for the network stack, so it would also indicate the performance for the Linux kernel. VPP and OVS do not come as a complete solution like Cisco and pfSense, but they are well established and a lot of guides and support is available on the internet. Both of them are mention in many research articles [KS18] [BA16] [BLM+18] [LRP+17], looks promising and have good performance results on paper.

Based on this, the 4 software routers Cisco CSR 1000v, pfSense, OVS and VPP was chosen to use in the performance tests and comparison. I addition, the HPE switch used to connect the two physical servers to the internet is also used for performance testing, to get an overview of the performance for a hardware ASIC router. The switch is not a high-performance router, but it can do layer three forwarding at decent speeds (214MPPS throughput and less than 3 μsec delay [HP]).

4.3 Configuration of the routers

All of the routers were installed and configured with the same IP settings. This simplifies the testing procedures, as the different VMs (VNFs) can just be turned on and off to test each different ones. All of the routers were configured as shown in Figure 4.4, but with unique management IP. Some of these parameters are changed when doing some of the tests described in section 5, but this is described in the details of that actual test.

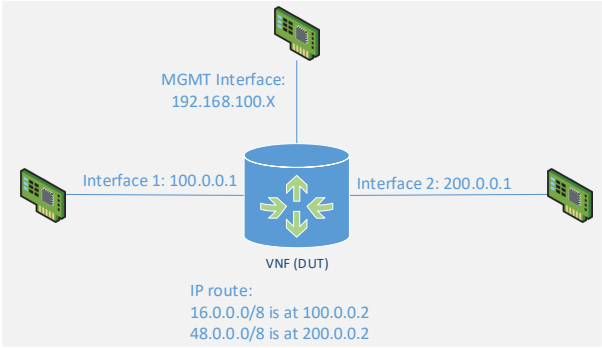


Figure 4.4: Generic router configuration

4.3.1 Cisco CSR 1000v

The Cisco router comes as stated above with a complete solution for deployment, which includes an Open Virtual Appliance (OVA) template. This is a pre-configured VM image that contains all that is necessary to get the VM up and running, including setting the parameters needed. These parameters can be IP address for the different interfaces, user name and passwords, enabling of remote access etc. During the installation, different choices can be made depending on how the router should perform. The choices are Small, Medium and Large, and all have 4 GB RAM and respectively 1, 2 and 4 CPU cores. Cisco recommends that Medium configuration must be used when processing 10 Gbps, which is the maximum license available. The large configuration is for using advanced security features (not available for the 10 Gbps licensing), and the small configuration is for bandwidths lower than 10 Gbps.

Three different VM were installed, each with different configuration options (respectively small, medium and large). The configuration of the router is the same as Cisco hardware routers, the CLI commands are the same. The routers were all configured according to Figure 4.4. No additional configuration was set/possible to set to improve the overall performance for the router. The version of the software used in the test was 16.09.02 and was downloaded directly from the Cisco website[Cisb].

4.3.2 pfSense

The pfSense comes shipped with an ISO image file, which only contains the software. Unlike the Cisco OVS file, the configuration of the VM has to be done manually. Only one VM was created, and the CPU configuration was just changed when doing different kinds of tests (described in chapter 5). The initial configuration of pfSense is done through the console in vCenter, like interface and IP assignment. After that, all the configuration is done through the web interface. The router was configured

according to figure 4.4. pfSense is mainly a firewall, but this feature is turned off, so it only acts as a basic layer three router. No other configuration or tuning of the performance was done (nor is it possible without changing the source code). The version used was 2.4.4 and downloaded directly from the pfSense website[Netb].

4.3.3 VPP

VPP does not come shipped as a complete solution; it has to be built from the source code manually. This first requires the installation of an OS, and the Linux distribution Ubuntu Server 18.04.1 was used for this purpose. This distribution comes shipped with minimal features and does not give any Graphical User Interface (GUI). The advantage compared to the desktop version (with GUI) is that only the minimum services and processes are installed, so the resource consumption is lower. The guide from FD.IO [FD.b] was followed to download, build and install the VPP software router. It was configured according to figure 4.4.

Unlike Cisco CSR and pfSense, VPP offers many possibilities when it comes to configuration for optimal performance, both with changing the source code and configuration options. The how-to guide from FD.IO was used to optimize the router [FD.d], and some of the tips were used like setting the power management, virtual extensions, CPU Affinity, NIC drivers etc. However, the tuning did not give any decisive improvements, as the default configuration is quite optimized. One important thing to make sure of is that the physical NIC is connected to the same NUMA as the cores that are processing the packets. Otherwise, the packets have to be transferred between different NUMA structures, which might introduce a performance bottleneck. This is not a problem with the CPU used in this test setup, as it only has one socket and therefore one NUMA architecture.

4.3.4 OVS

OVS, like VPP, has to be installed on top of a Linux distribution, and the source code built and installed. The guide from openvswitch.org was followed for this process [Ope]. OVS does not come with any routing capabilities; it is just a virtual switch which provides the dataplane. To make this into a working router, a SDN controller has to be used to control the switch. The open source SDN controller Faucet was chosen, since it was recommended by open vSwitch and guides to how to integrate OVS and the Faucet where provided [Dev]. The guide was followed to install and configure the SDN controller.

OVS also have many options, both in the source code and configuration files. In the installation guide, there is an own section for performance tuning. This guide was followed, but like the tuning of VPP, this did not have a significant impact on the

overall performance. The most important options are the number of CPU cores that is in use.

4.3.5 HP Switch

The HP Enterprise switch is both used as a switch to interconnect the two physical servers and to act as a hardware ASIC router to get some comparable results against the software routers. As stated before, this is primarily a switch designed to do layer 2 forwarding, but is quite capable of doing layer 3 forwarding as well. The 10 Gbit/s ports on this router is only accessible through Small form-factor pluggable (transceiver) (SFP) and fibre, so the 10 Gbit/s TP ports cannot be used in this setting. This means that another physical NIC is used for the testing of the HP switch than when the software routers are tested and could be a cause for some errors in the experimental testing.

The configuration of the HP switch is done through a web interface and is configured in the same way as the other software routers. One key difference with this switch is that it uses ASIC for the packet processing, so the core configuration done in the software router does not apply here. An Intel X710 NIC is used for the fibre traffic (see connectivity in figure 4.2). The X710 has almost the same specification as the X540 NIC, at least those relevant for these tests.

Chapter 5
Test Evaluation Methodology

This chapter will describe how the different test will be performed and what kind of results (metrics) that will be used. The main purpose of these experimental tests is to measure the performance of the different software routers in different scenarios. The tests are based on the RFC 2544 [IET] where the main ideas are used.

5.1 Network topology

The general network topology used for the tests is shown in Figure 5.1. This topology is used in all the tests, except those using VNF in parallel and chaining (described in section 5.4.6). The IP-addresses used were taken from a sample configuration for the TRex packet generation. TRex also simulates a chosen number of clients/servers to generate traffic from/to, to get a more realistic testing scenario. This is described in more detail in section 5.3. The actual physical setup of the TP cables is shown in Figure 5.2 (the full physical setup is shown in Figure 4.2).

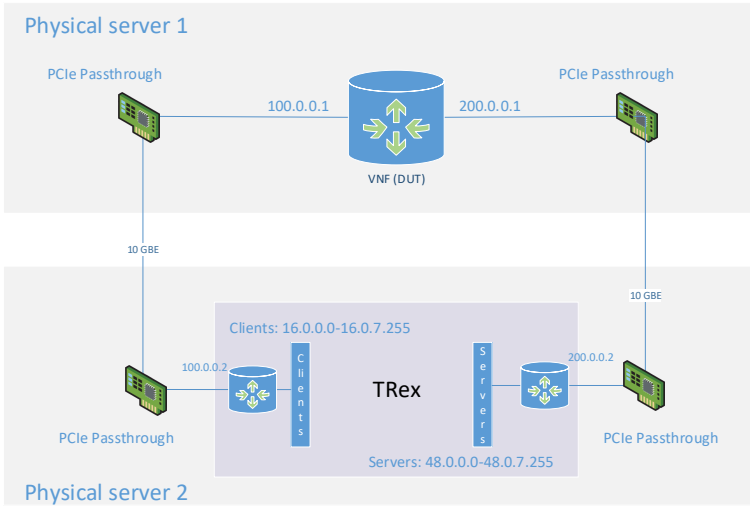


Figure 5.1: Generic Network topology for the tests

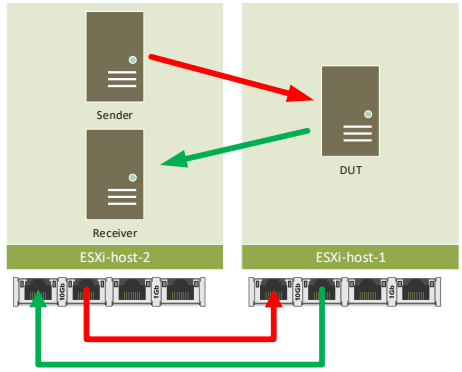


Figure 5.2: Physical setup

The physical NICs supports 10 Gb/s each, so the maximum of traffic to process for the DUT is theoretically 10 Gb/s when sending from end to another. To get more traffic for the DUT to process, 10 Gb/s traffic is sent from both ends simultaneously, so theoretically the DUT could be processing 20 Gb/s. Thus the term client/servers used is not entirely accurate, but TRex uses the terms and therefore used in the rest of the thesis.

5.2 Metrics

To be able to compare the different kinds of routers, it is necessary to establish some metrics to be used in the experimental testing. All of these metrics are outputs given by the traffic generator TRex. The metrics used are:

- Packet Loss - Given in percent and is the number of packets that are sent, but not received in the other end.
- Actual throughput - Given in PPS and is based on the number of packets that are sent and received in the other end, regardless of the packet loss. Calculated by taking the total number of packets received divided by the test duration time.
- Packet Delay - Given in seconds, the time from the packet is sent to it is received again. This is done by sending additional packets on top of the generated traffic, and the time difference measured. Since the sender and receiver are on the same physical CPU, the delay measurement should be quite accurate. The SCTP is used to measure the delay.

The number of packets the router can handle (PPS) is a more interesting number than the number of bits its able to handle because it is independent of the packet size. This number is not necessarily the metric Actual throughput since this could involve some packet loss. The maximum throughput in PPS for a software router is (in this thesis) determined by what the actual throughput is when the packet loss is less than 0.1%.

Other metrics, like CPU utilization and RAM usage, are not considered and evaluated in these tests. The primary focus on the tests is to find the maximum throughput, and then the CPU utilization will be around 100%. Some routers also use polling so the CPU utilization will always be 100%, regardless of the packet load. The CPU utilization is therefore not so interesting in this test setup. RAM is today considered quite cheap, and all the routers did from their specifications specify how much RAM they require to operate. All the routers were therefore given "unlimited" RAM (32 GB), which is way more than required. The size of the RAM will therefore not be a bottleneck and is not considered when measuring the routers. The number of cache misses is measured in a lot of research papers when testing software packet processing [ERWC15] [GES+17] [BLM+18]. This will also not be tested, mainly because of the difficulty of measuring this in the routers that do not run on Linux. Also, since maximum performance in the interesting part, cache misses is not so useful in this context.

5.3 TRex packet generation

TRex is configured as shown in figure 5.1 regarding the IP addresses. The platform was configured to use 1 CPU core for the general handling, 1 for latency measurement and 14 cores for traffic generation and measurement. This to ensure that TRex is not the bottleneck in the test setup. The general subnets used are 16.0.0.0-16.7.255.255 for the clients, and 48.0.0.0-48.7.255.255 for the servers. This means that 524k clients and 524k servers are simulated for the test. When testing with a large routing table and ACLs, the number of clients/servers are set to be equal to the size of the routing table and ACL entries.

TRex comes with some templates pre-installed. One template consists of one 64B UDP packet, that is sent from the client to the server. This template (specifically the pcap file) were modified using WireEdit [Omn] to consist of two 64B UDP packets. Wireshark [Wir] were used to verify that the packages created are valid (correct checksum). The two packets where modified so that one is sent from the client to the server, and the other is sent the other way. This template is used as a base for all the tests. When it is modified, or another template is used, it is specified in the test.

```
t-rex-64 -f udp_64B_2_ways.yaml -c 14 -l 100 -d 60 -m X
```

Figure 5.3: TRrex command

The command in figure 5.3 is what is used in most of the tests. It specifies what kind of template to be used (udp_64B_2_ways.yaml), the number of cores to use (14), number of packets per seconds to send for latency measurement (100), test duration (60) and how much traffic to send (the variable X). The number defining how much traffic to send is based on some numbers in the template file and for the "udp_64B_2_ways.yaml" the number 1 equals 0,181 MPPS [1]. It is this number that is the variable when doing the tests, to determine the maximum throughput.

5.3.1 Script for automating the tests

To be able to determine the performance of a router, it is necessary to perform multiple tests. The span chosen is from 0 to 35 MPPS. The theoretical maximum PPS is around 30M (2 * 14,88 MPPS), and the span was determined a little bit over this to see what happens when this limit is reached. An initial test is done

[1] This number is based on the Connections Per Second (CPS) in the template file, which is set to 90615. Sending on both interfaces, this equals 0.181 MCPS. Since one connection is just one packet, the CPS equals PPS in this case

prior to the final test, to see approximately where the maximum performance is. These results are then used to determine what kind of interval to use for m (in the TRex command, figure 5.3). The density of the tests is more intense around the approximately maximum performance acquired from the initial test, and not so intense elsewhere.

Each test has around 40 individual tests where the PPS is the variable, and since each of those tests takes 60 seconds, the total time for each router test is around 40 minutes. In order to be able to do all these performance tests, a script was developed to automate most of these tests. This can be found in Appendix B. This performs all the tests and stores the result in a csv file. For the routers VPP and OVS the script can also change the configuration between each tests using remote commands, like the CPU core configuration and routing table size. For the Cisco and pfSense router, these changes have to done manually between each test.

5.4 Tests scenarios

To be able to compare the different kinds of software routers when they are used as a VNF, different kinds of test scenarios has been formed. First is a test where different kind of core configuration is tested. Based on this test, one optimal core configuration is used for the rest of the tests, and other factors are changed like routing table, ACLs, realistic traffic etc... Then the software router that has the best performance based on the previous tests, is tested without virtualization, to see what impact the virtualization has on the performance. Lastly, the same software router is tested with multiple instances, in both parallel and when chained. Unless otherwise specified, the 64B UDP packet sending on both ports is used as test traffic.

5.4.1 Testing with different core configuration

This test purpose is to see how the number of CPU cores in a software router affects the overall performance. The network architecture from figure 5.1 is used, and the most optimal configuration on the router is used. This means that all unnecessary functions have been deactivated on the routers (that can be deactivated), so that they only perform as a layer 3 forwarder. 2 routes[2] is inserted into the router, to be able to forward the traffic correctly to/from the TRex traffic generator. The HP switch performance is also tested, but since ASIC is used when doing the packet processing, it is not possible to test with different core configurations. One test is therefore performed on the HP switch. 524k clients and 524k servers are simulated for sending and receiving the traffic.

[2]IP routes: 48.0.0.0/8 is at 200.0.0.2 and 16.0.0.0/8 is at 100.0.0.2

To generate the routing table, a script was created to do this. The generic script for this can be found in Appendix C. Since the configuration of the routing table is different for each of the routers, a script for each router has been created.

5.4.2 Testing with large routing table

This test purpose is to see how the performance is effected by using a large routing table. This involves IP route look-up for each incoming packets, which means more CPU computation. 262k of routing table entries was chosen to be tested since this was the maximum number of route entries the Cisco and pfSense allowed (found by trial and error). Each of the simulated client/servers has a corresponding route entry so that the router is forced to do a look-up for each of the client/servers. This means that only 262k client/servers are simulated in this test, opposite to the 524k in the previous test.

The HP Switch did only support 32k routes, so the number of routes tested on the switch is 8 times smaller than with the software routers. But it will still give an indication of how adding many route table entries will affect the overall performance.

5.4.3 Testing with ACL

This test purpose is to see how the performance is affected when turning the software router into a IP-FW using ACLs. To have the same number of ACLs on all the routers, 1024 entries are used. The number of simulated client/servers is reduced to 512, and each of them has a corresponding ACL. The packets are checked on the input on both of the ports. The ACLs where created from the scripts found in Appendix C.

5.4.4 Testing with realistic traffic

This test purpose is to test the performance of the routers with more realistic traffic, not just the smallest packet size possible. The use of 64B UDP packets give a good picture of how much packets the router is able to process, but in a real-life scenario, this is not the type of traffic that will be routed. As stated in section 2.7, the average packet on the internet is around 870 byte. To use this packet size will give a good indication on how the router will perform in a real-life scenario. A template was created based on the previous one ("udp_64B_2_ways.yaml") and changed to 870 bytes (888 bytes on the wire).

This test still only test UDP packets with a fixed size, which is still not realistic. In a real-life scenario packets have different kinds of sizes, may belong to the same sessions, uses other protocols than UDP etc... TRex comes pre-shipped with some other test templates, including an SFR template. This includes traffic like HTTP

and HTTPS browsing, exchange and mail operations, some streaming media over RTP, video call and DNS traffic. The purpose is to get more realistic traffic that could be found on the internet today, and this includes much of that traffic. This template does not send the same amount of traffic on each port, since it is a more realistic client/server distribution ratio.

5.4.5 Testing without virtualization

To be able to see how the virtualization is affecting the performance of the software routers, the best router (VPP) was installed directly on a server, without the VMware hypervisor and on the same type of server as used before. The Ubuntu server was therefore installed directly on the physical server, and VPP was installed on top of that. The configuration is exactly the same as used when the router was a VNF. The test traffic was with the 64B UDP template.

5.4.6 Testing with VNF in parallel and chaining

The purpose of this test is to see how the VPP router performs when used in chaining and parallel (offloading) mode. It also gives some indications on how well suited the router is to be used as a VNF, when it is cloned and multiplied into several instances. When testing VNF in parallel or chaining, this involves some processing for the hypervisor. Unlike the previous tests where the NIC was directly connected to the VNF (PCIe passthrough), some virtual network switch has to be used. In the parallel test, the SR-IOV is used to share access to the physical NIC. And in the chaining test, only the end VNF (figure 5.4) is connected to the physical NIC. The network between each VNF is a virtual one provided by the hypervisor, which requires CPU power to be processed.

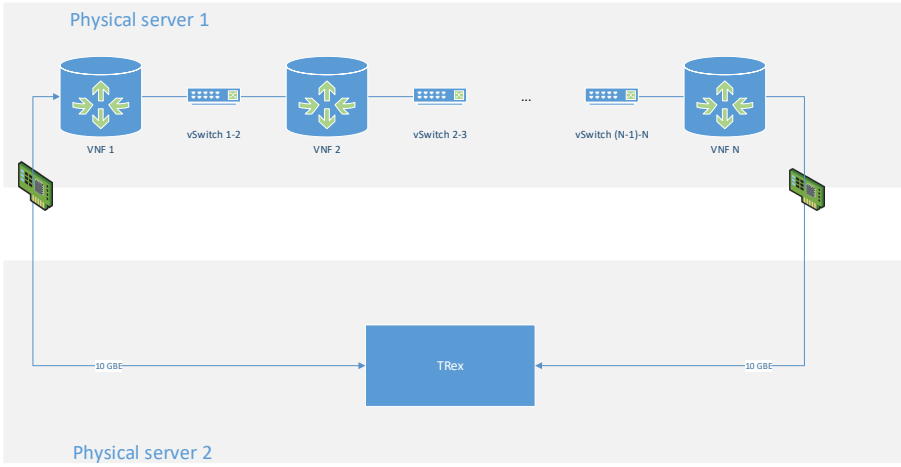


Figure 5.4: Topology for VNF chaining

The VNF chaining is shown in figure 5.4, and up to 8 VNF is tested (all the cores is then taken on the physical server). They are all cloned from the VPP VM used in previous tests, so the only changes that are done are reconfiguring the IP addresses and what virtual NIC that is connected. Each of the virtual network (vSwitch) between the VNF is using one core each for packet processing. All the routers are configured to send the traffic to the next VNF in the chain, so all the packets will have been processed N times after passing through the chain. A detail description on how the setup with 8 VNF was done, including IP addresses, can be found in Appendix D.

The VNF parallel testing is shown in figure 5.5. Basically what this setup is, is a kind of offloading of network functions, where each of the VNF is configured to do the same tasks. Normally there would be a load balancer before and after the VNFs, to distribute the load between them. In this setup, the TRex is doing the load balancing, by sending a different kind of traffic to the different IP addresses of the routers. SR-IOV is used to share the access of the physical NIC, and some computation is required from the hypervisor to distribute the traffic to the correct VNF. A detailed description of the test setup can be found in Appendix D.

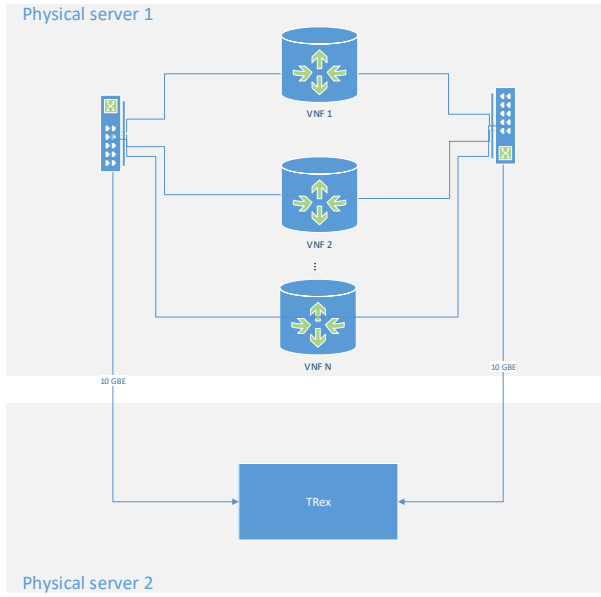


Figure 5.5: Topology for VNF in parallel

For all these tests; 2, 4 and 8 VNFs have been tested. The 64B UDP packet template is used for traffic generation.

Chapter 6
Results

This chapter describes the result found during the configuration part of the different kind of software routers and the performance results from the different scenarios in section 5.4. All of the proposed routers can process and forward IP packets when they are used as a VNF, but the rate they are able to do this varies. For the routers that have the possibility of performance tuning, this has been tried but did not give any significant improvements from the default configuration. The total number of individual tests run during this thesis is 1546; each consist of a 60 seconds test. All of these results are being presented in graphs, and the performance numbers have been collected from these graphs.

When the Cisco and pfSense router is tested with the number of cores specified, the VM is only allocated that exact number of cores. For the VPP and OVS router the VM is allocated 10 cores, but the configuration is restricting how many cores that can be used. The number of cores tested is the number of actual cores that are doing packet processing, but the software may use more cores than this. This can, for instance, be to coordinate the work between cores and compile statistics. For the Cisco and pfSense router, these tasks have to be done by the cores that are also doing the packet processing. That is the reason that during testing, the SSH session is disconnected from the Cisco router because the packet processing is prioritised. The same applies to the web interface for the pfSense during testing. This is not the case for VPP and OVS.

Not all of the software routers are able to be tested according to the test scenarios, as some have some limitation on how many IP routes that is possible to add or the number of ACL inserted. For instance the Cisco CSR, the packet loss is never under 2-3%. So for this router, the results are when the packet loss is 5%, to be able to compare it to the other routers. The final results for the Cisco router have been corrected and reduced by 5% and the other routers by 0,1%, to compensate for the packet loss.

6.1 Routers with different core configuration

In figure 6.1 to 6.4 are the test results from the scenario described in section 5.4.1. It shows how the different software routers performs regarding packet loss and latency when test traffic up to 35 MPPS is processed. Only the latency results for 1-core configuration is shown in this section, the rest of the results can be found in Appendix E

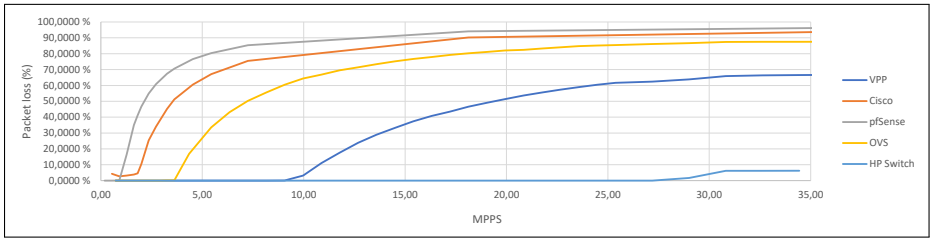


Figure 6.1: 1 core: Packet loss test result with optimal configuration

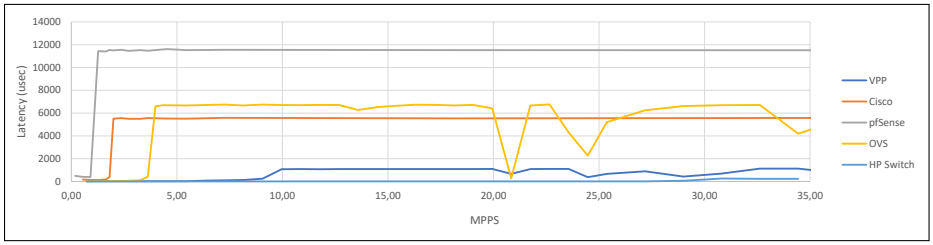


Figure 6.2: 1 core: Latency test result with optimal configuration

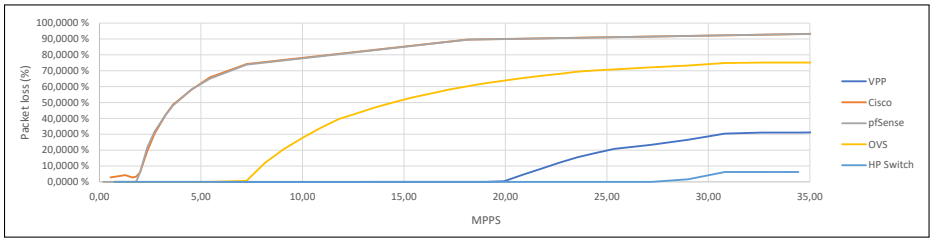


Figure 6.3: 2 core: Packet loss test result with optimal configuration

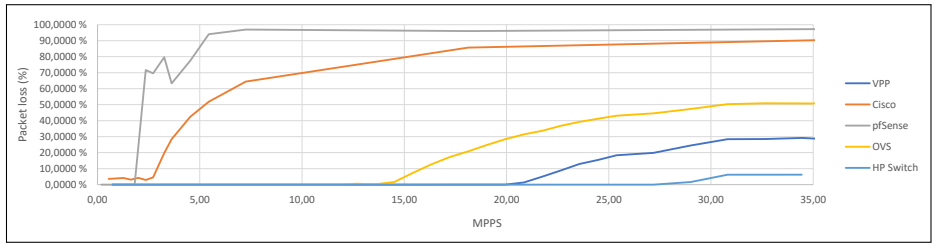


Figure 6.4: 4 core: Packet loss test result with optimal configuration

A summary of these results of what the maximum performance in PPS and what the latency is, can be found in table 6.1. The result is taken out of the graphs where the packet loss is 0,1%. The graph in figure 6.5 shows the same throughput numbers. As before, for the Cisco CSR router, the number is collected for a packet loss at 5%.

Cores:	1		2		4	
	MPPS	μsec	MPPS	μsec	MPPS	μsec
VPP	9,05	242	19,14	248	19,95	204
OVS	3,20	97	5,71	105	11,89	97
Cisco*	1,73	589	1,89	5062	2,60	4087
pfSense	0,91	508	1,77	3557	1,81	2574
HP Switch	27,27	12	N/A		N/A	

Table 6.1: Throughput and delay with 0,1% packet loss and different core configuration

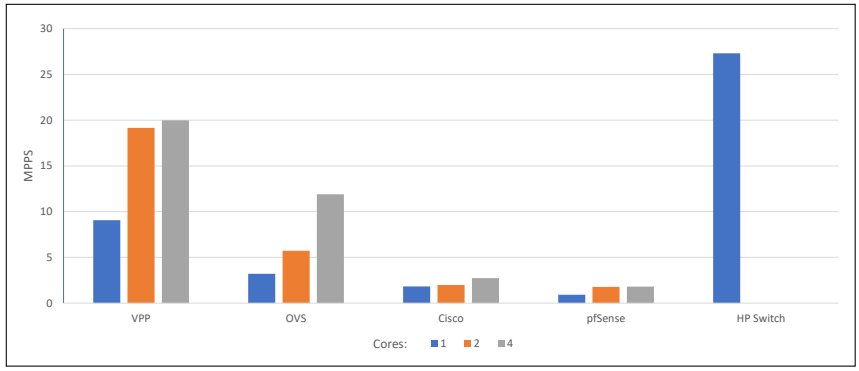


Figure 6.5: Throughput with 0.1% packet loss and different core configuration

6.2 Performance with additional features

Based on the test done in the previous section, all the rest of the tests are done with a 2-core CPU configuration. Only the PPS throughput performance test results are displayed in this section; the rest of the results can be found in Appendix E. The figure 6.7 to 6.10 show the results when a 262k large routing table was inserted and when 1024 ACLs was inserted (not at the same time). Some results also show results when just 1 or 2 ACLs are activated, to show the effect by just having a few rules.

The OVS is not included in these results, as it was not possible to add 262k IP routes or 1024 ACL and get the router to work. 1 ACL was added to see the performance effect, and it performed the same as without one. Figure 6.6 show how adding IP routes to OVS affects the performance compared to VPP, and shows that the performance reduces drastically when adding IP routes. This is only to give an indication of how the performance is degraded, as these tests are performed by the developers of the VPP router.

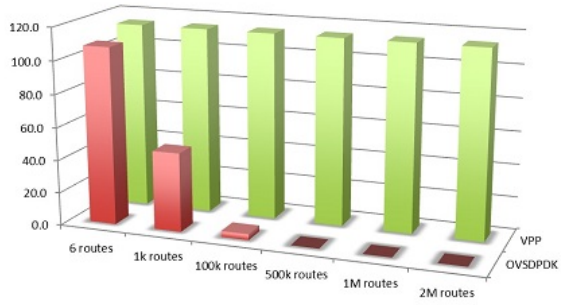


Figure 6.6: VPP and OVS comparison with various IP routes (tests performed by fd.io) [FD.e]

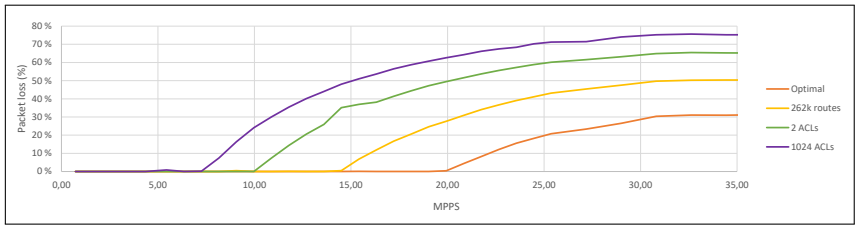


Figure 6.7: VPP: Performance with large routing table and ACLs activated

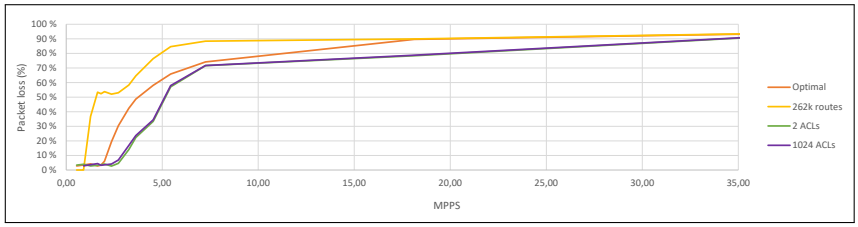


Figure 6.8: Cisco: Performance with large routing table and ACLs activated

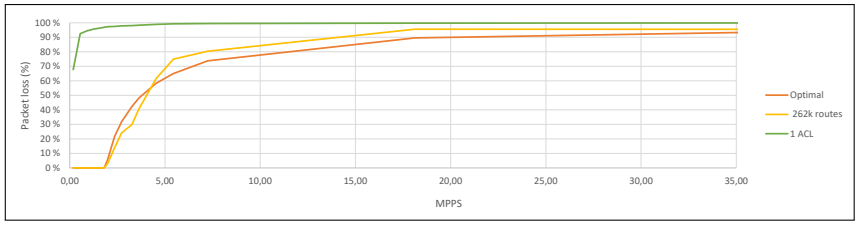


Figure 6.9: pfSense: Performance with large routing table and ACLs activated

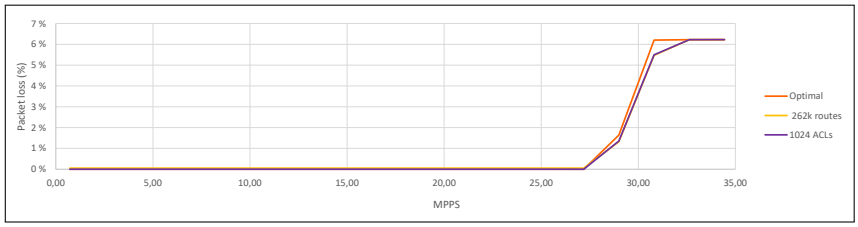


Figure 6.10: HP switch: Performance with large routing table and ACLs activated

For the summary after the tests with different core configuration, table 6.2 and figure 6.11 shows the summary of the maximum throughput with 0.1% packet loss. There are no new results for the OVS router, and pfSense has no results for the configuration with 1024 ACLs. For the Cisco router, the 5% packet loss limit is still used.

Configuration:	Optimal		262k Routes		1024 ACLs	
	MPPS	μsec	MPPS	μsec	MPPS	μsec
VPP	19,14	248	13,76	314	6,89	730
OVS	5,71	105	N/A		N/A	
Cisco*	1,89	5062	1,85	170	2,34	1984
pfSense	1,77	3557	1,82	4085	N/A	
HP Switch	27,27	12	27,23	20	27,29	20

Table 6.2: Throughput and delay with 0,1% packet loss and and different number of routes and ACLs

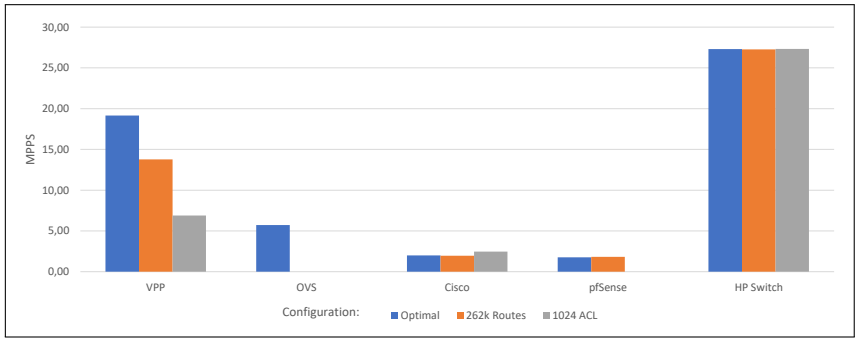


Figure 6.11: Throughput with 0.1% packet loss and different number of routes and ACLs

6.2.1 Realistic traffic

Table 6.3 and figure 6.12 shows the results from the test where (near) realistic traffic has been tested. The throughput is for the first time showed in bit/s, and not PPS. This is because when increasing the packet size above 64B, the number of bits that is pushed through the interface is more likely to be the limiting factor. In the table, the throughput of each of the two ports is shown individually, because of the different kind of traffic that is going through them in the SFR test.

	Traffic	Port 0 (Gbit/s)	Port 1 (Gbit/s)	Sum (Gbit/s)	MPPS
VPP	UDP 888B	9,79	9,79	19,58	2,76
	SFR	2,77	9,76	12,53	2,76
Cisco*	UDP 888B	5,03	5,03	10,06	1,41
	SFR	1,68	5,35	7,03	1,52
pfSense	UDP 888B	3,74	3,74	7,48	1,05
	SFR	1,08	3,47	4,55	0,98
OVS	UDP 888B	9,79	9,79	19,58	2,76
	SFR	3,02	9,77	12,79	2,75
HP Switch	UDP 888B	9,79	9,79	19,58	2,75
	SFR	3,07	9,77	12,84	2,75

Table 6.3: Throughput with realistic traffic

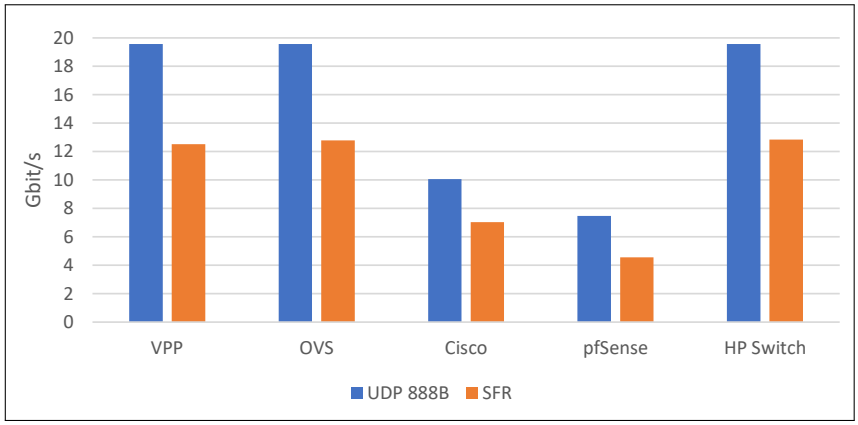


Figure 6.12: Throughput with with realistic traffic, the sum of traffic on both interfaces

6.2.2 Without virtualization

For the test to compare the difference in performance with and without virtualization, only the VPP software router was tested. This is because that VPP was the router that performed best in the previous tests. The same configuration files from the VPP VNF router was used. Figure 6.13 and table 6.4 shows the results from this tests, compared with the results from when VPP was used as a VNF. It was also tested with 4 and 6 core configuration, but these results are not shown here as they were identical to the 2-core configuration.

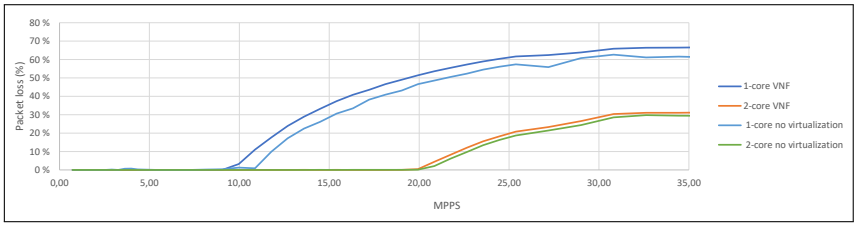


Figure 6.13: VPP performance throughput with and without virtualization

	MPPS	μsec
1-core VNF	9,05	242
1-core no virtualization	7,75	72
2-core VNF	19,14	248
2-core no virtualization	19,50	159

Table 6.4: VPP performance and latency with and without virtualization at 0.1% packet loss

The graph in figure 6.13 shows that the performance is slightly better without virtualization, but the table 6.4 shows that 1-core VNF performs better when virtualized. For unknown reasons the 1-core without virtualization has some packet loss around 1% from 8-11 MPPS. But at 10,88 MPPS, the packet loss for the VNF is 11% and the packet loss without virtualization is 1%, so the performance is still better without virtualization.

6.3 VNF in parallel and chaining

The test results for the parallel and chaining scenarios are shown in figure 6.14 and 6.16. In previous tests, the CPU usage has been easy to determine, since the pulling mode uses 100% of the CPU cores allocated. In the parallel and chaining scenario, additional computation has to be done by the hypervisor since the virtual network is introduced. And this is affecting the results when adding more VNFs and the total number of CPU cores for the physical server becomes a bottleneck. Everything is still working, but the hypervisor is then sharing the resources, which might mean that a VPP process is not actually getting 100% of a CPU core.

When testing with four VNFs in the chaining scenario, the total CPU consumption on the physical server was 83% (data collected from the vCenter). When testing with eight VNFs, the total consumption was almost 100%. In the parallel scenario, the physical CPU consumption never reached 100%, even with eight VNFs.

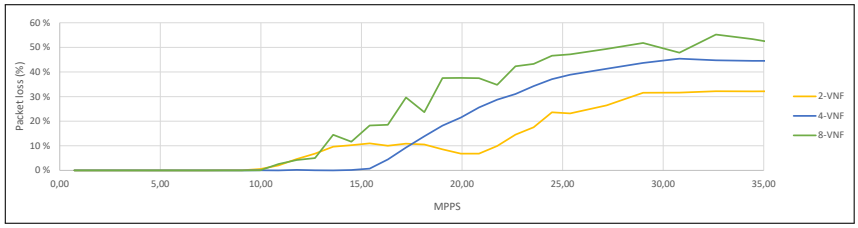


Figure 6.14: VPP performance throughput in a parallel scenario, packet loss

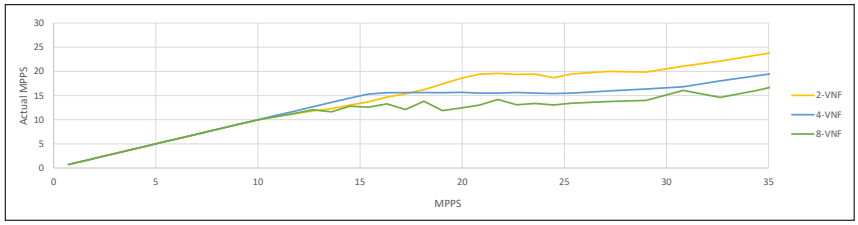


Figure 6.15: Actual throughput VPP, in a parallel scenario

At the time of the testing, VPP did not fully support the use of the VMXNET3 virtual NIC. One of the features that were not supported was the RSS feature, which is the NIC feature that can distribute packets between multiple CPU cores. VMXNET3 is the virtual network drivers that are used between the VNFs in the chaining scenario.

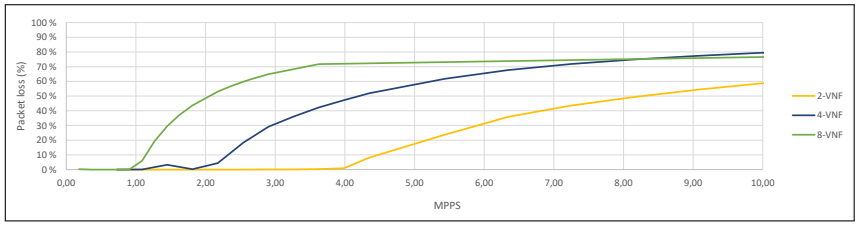


Figure 6.16: VPP performance throughput in a chaining scenario

Chapter 7
Discussion

This chapter is a discussion based on the theory in chapter 2 and the results from chapter 6 within the scope of the research question presented in section 1.2. It starts with a discussion on the experimental testing that has been performed, the possible errors in the test and the individual test results for the VNFs. The aim is to address the second research question. After that, a general discussion addressing the first research question on the different challenges of doing routing in software, and how the different routers tested in this thesis overcome these.

All of the proposed software routers were able to forward layer 3 IP traffic, and they did this while they were configured as VNFs. Most of the results were as expected, according to the research done and what the developers had claimed their product should be able to perform. The way the test was performed, and that the same experiment was able to be run on all the routers, gives a good comparison between the different solutions. This knowledge is directly linked to a part of the research question two.

The experimental testing has some room for errors in the results, but by doing many 60 seconds test for each specific configuration, these errors are hopefully reduced. Before each test, a short test is always performed, to see that the results are near what they are expected to be. Also, this short test has also helped to decide where the highest resolution of the actual test should be (around its maximum throughput performance). Doing this also helps reduce the possible errors that could occur. The measurement of the latency is done on top of the generated test traffic. It is in many ways a separate test, but it is performed by the same software that generates and checks the test traffic. This way it should give accurate latency results, but might affects the total performance some. 100 SCTP packets are sent per second to do the measurement, which means that at 1 MPPS test traffic, the latency measurement packets is 0,01% of this traffic. At 20 MPPS it is 0,0005%. This use of this latency measurement should not affect the overall performance throughput significantly.

During the test, some limitations have been discovered. Research has shown that it is possible to do almost 100 Gbit/s packet processing in commodity hardware [LWRL17]. In this test setup two 10 Gbit/s NIC is used, which at maximum allows for traffic up to 20 Gbit/s / 29,76 MPPS. As the tests have shown, this limits is only reached for the HP switch which used ASIC, but that still does not mean that the physical NIC is the only limitation for the tests. This is discussed more in 7.1. Other limitations in the test are the number of CPU and cores, but does not really kick in before the number of VNFs is increased above 4.

7.1 Individual test results

7.1.1 Cisco CRS 1000v

The Cisco CSR 1000v performs almost as expected and as promised by the developers. For the UDP 888B test, it manages to process and forward 10 Gbit/s. This is probably a limit set by the licensing for the router, as it is only for 10 Gbit/s. When testing with SFR data, it is only able to process 7 Gbit/s with 1,52 MPPS. With the UDP 64B test, it was able to process 1,89 MPPS, so the potential is there to manage to process 10 Gbit/s. However, the SFR traffic introduces some more complex traffic than simple UDP packets, and the average packet size is also smaller than 888B. One surprising thing about the router is that it has a constant packet loss around 3-5 percent, even when its maximum performance is not reached. The reason for this is unclear, and during the research phase, nothing suggested that this would be a problem. However, this means that the maximum performance is set to be when the packet loss is 5%, to get some numbers out of the test. This makes the numbers not completely comparable to the other routers in the test.

When reaching maximum throughput, and the CPU usage is around 100%, it shows clearly that all of the cores are used and prioritized in the packet processing. The SSH connection is disconnected when reaching 100% CPU usage, and it is not possible to connect to it again before the test traffic is stopped. This makes it impossible to monitor and make changes to the router during heavy load operation, which in some cases could be a problem. When increasing the IP routing table and adds ACLs, the router actually performs better than without. This is a little bit strange since more computation is required when these extra loads are introduced. Why the performance is better with is unclear, but it shows that the process of checking large IP routing tables and doing ACLs check does not affect the overall performance negatively.

7.1.2 pfSense

The pfSense router performs almost as good as the Cisco router does, with 1,77 MPPS performance with 2-core configuration. With the UDP 888B test, it performs 7,5 Gbit/s. With the SFR traffic, it performs even worse, only able to process 4,5 Gbit/s of data. The PPS is only around 1M, which is almost half the maximum PPS performance it has with 64B packets. This indicates that the assumption that only the PPS performance for a router is important, is not entirely correct. Other factors come into play when the packet size is increased from 64B, and the most likely factor that has an impact on this is the RAM (size and the access speed). pfSense does not take advantage of multi-core architecture, as only one core is used for packet processing. In the release of a new version of the router, it is announced that it will support DPDK and VPP for packet processing and the performance throughput is expected to be dramatically increased.

The latency is like the Cisco router quite high at maximum performance, around 3,5 msec. But up until that limit is reached, the latency is around 3-500 μsec, which is not that bad. It also does not start dropping packets before the maximum throughput is reached, which is what is expected for a router. When adding large IP routing table, the overall performance is not affected much by this. However, when activating the firewall function, which is needed to use the ACLs, the performance drops drastically. When the FW is activated, more than just checking ACLs is being done. One thing that is being done, is keeping track of sessions. Also, with the 64B UDP test traffic, each of the packets is treated as a session, and the maximum of sessions possible in pfSense is soon reached. And when this limit is reached, it stops processing further packets. The performance with the FW activated is much better than the results shows in this test, but in the 64B UDP test, it is not able to do that.

7.1.3 VPP

VPP is by far the best software router in this test when considering maximum throughput performance. The latency results are only beating by the OVS router, but expected since one of the main thing VPP does to boost performance, is batch processing. And this does naturally introduce some delay. With 9 MPPS performance with 1-core configuration, and 19 M with 2-core, means that it can process layer 3 packets at 10 Gbit/s line rate (14,88 MPPS) with just 2 CPU cores. At around 20 MPPS the performance is at a maximum, independent of the number of cores allocated (tested with up to 8 cores). For the HP switch ASIC the performance is measured to 27 MPPS, so the test method is able to measure throughput above 20 MPPS. The latency results after reaching 20 MPPS is also volatile. Up until 20 MPPS, it is quite stable, but after it varies both up and down. The reason for this is unclear, but one theory is that queuing in some part of the processing chain is kicking in because there is a performance bottleneck. Some sort of queuing would

explain the unstable latency results, but why and where the queuing is taking place is unclear.

When adding 262k IP routes into the FIB, the performance is reduced from 19 to 13 MPPS. That is far from what the developers claim, that the performance is hardly affected by the number of routes in the FIB (as shown in figure 6.6) They claim that the performance is the same with 6 routes, as it is with 2M routes. This is not true, based on the results produced for this thesis, but the developers do not in any way say how the test has been performed. A different test might give some other results.

When adding ACLs and by that activating the ACL-plugin for, the performance is reduced drastically. By just adding 2 ACLs, the performance is halved to around 10 MPPS, and when 1024 ACLs is added, the performance is reduced to 7-8 MPPS (from graph in figure 6.7). This clearly shows the plugin-tree used in VPP, where only the necessary plugins are used. When activation this plugin, every packet (or vector) has to be processed by that plugin, which introduced a considerable performance downgrading. Other routers are not heavily affected by adding ACLs, which indicates that a check is performed anyway, regardless of any ACL is entered or not. This shows that using plugins is a smart way to get high-performance since only the necessary packet processing is done.

When realistic traffic is tested on the router, it is only limited by the physical 10 Gbit/s NICs. It has no problem at processing regular traffic at 20 Gbit/s, with a 2-core configuration.

7.1.4 OVS

OVS is the second best software router, based on the performance. With realistic traffic, it is able to perform at maximum with 20 Gbit/s. The throughput with the 64B UDP traffic is 5,7 MPPS with a 2-core configuration, and can achieve around 20 M with an 8-core configuration. This makes the OVS a perfectly capable packet processor but does not have so good per core as the VPP. The latency delay is the best results of all the software routers, with roughly 100 μsec at maximum throughput. So for a time delay sensitive use cases, this router could be the best choice. The OVS does not batch process packets, so each packet is processed when they are arriving, and therefore making the processing time shorter.

The same latency results as with VPP appear after 20 MPPS is reached, unstable and maybe indication some sort of queuing. A common denominator for the OVS and VPP is that they both uses the DPDK for kernel bypassing and network drivers. It might be something with DPDK and the hardware in use that is causing this, but that is unclear.

When adding a large routing table and 1024 ACLs to the SDN controller that is controlling the OVS, it fails because the loading time is too long. It might be that if another SDN controller was used than the Faucet, it might have been possible to test this. However, since the FIB and the ACLs were reloaded each time the test is started, it was not possible with the SDN controller Faucet.

7.1.5 HP Switch

No surprise, the HP switch has clearly the best performance, both in throughput and latency. This shows how hardware ASIC still has some huge advantages over packet processing in software. The traffic generator is not near the limitation for the switch of 214 MPPS. But here the limitation of the physical 10 Gbit/s NIC is the bottleneck, where it is able to process 27,27 MPPS (and 29,76 MPPS is the theoretical maximum with 2 x 10 Gbit/s NICs). With latencies around 5-7 μsec, and max latency of 12 μsec with high load, these results are in another league than the software routers tested in this thesis.

When adding a large IP table (just 32k, not 262k as for the other routers) and adding 1024 ACLs, the performance is not significantly changed. This shows the advantage of ASIC, where the packets have to go through all the circuits anyway, and therefore no performance downgrading is introduced when adding IP routes and ACLs. A different physical NIC (Intel X710) was used during testing, because of the fibre connection. This might be the reason why the throughput is able to pass the 20 MPPS limit that VPP and OVS could not pass. However, it should not be a problem, since the x540 NIC should sustain full line rate with 64B packets on both ports [ERWC15].

7.1.6 VNF Chaining and Parallel

When testing the VPP router in a VNF chain and parallel scenario, what is really being tested is the hypervisors ability to process packets. Since multiple VNF instances has to share a physical NIC, some processing has to be introduced to be able to distribute the packets correctly. Moreover, the hypervisors ability to process packets is considerably worse compared to the VPP processing. In the parallel scenario, the maximum throughput is around 15 MPPS, and since more CPU cores are actually in use to process the packets, you should expect that the performance would go up. The SR-IOV way of sharing the interface is most likely the cause of the performance downgrading and becomes the bottleneck. If we see on the actual throughput regardless of the packet loss, the performance for the 2-VNF setup is around 20 MPPS (from figure 6.15). That is the same performance as the optimal setup where the VNFs have exclusive access to the physical NIC (but with packet loss around 10%). This shows that the SR-IOV can perform at high bandwidth loads,

just that some packet loss is expected. The physical number of CPU cores available is not a problem before passing 8-VNFs, which can be seen as the unstable latency results for the 8-VNF configuration.

In the chaining scenario, the physical interfaces are not shared between multiple VNFs. But between each VNFs, a virtual network switch provided by the hypervisor is responsible for processing and transferring the packets. This is the bottleneck in this test scenario. The performance with 2-VNFs is around 4 MPPS, and just 1 MPPS with 8-VNFs. This shows that the virtual network provided by the hypervisor is not near the performance for SR-IOV, when a physical interface is shared with multiple VNFs. The overall performance is also going down when adding more VNFs to the chain. This suggests that the virtual networks are using the same CPU resources when processing the packets.

7.2 Challenges with routing in software

The software routers that had the best performance in this test was the one using DPDK for kernel bypassing. A key factor for success in processing packets in software is to have an efficient way of moving the packets from the physical NIC to the software that is to process the packets. The use of multi-cores is also essential, as it makes it possible to scale the performance based on the need. The Cisco and pfSense router has not the ability to utilize multiple CPU cores, and this is clearly shown on the results. A minimum number of CPU operations has to be done on each packet, so without any way of utilizing multiple cores, there are no possibilities to be able to process packets beyond 10 Gbit/s.

The RSS function in the NIC is a crucial part when using multiple CPU cores, as it can distribute the packets between the cores without using CPU processing power to do it. Offloading some of the packet processing to the NIC, like RSS and checksumming, is much faster than doing this in software. The advantage is that most modern NICs has these abilities. Batch processing of packets is the main difference between the VPP and OVS router. The ability to processes similar packets as one, and be able to pre-fetch instructions from memory ahead of time, saves many CPU cycles and have a significant impact on the overall performance. It does introduce a small delay, but in many use cases, this increase is insignificant. It is not without a reason that pfSense wants to use VPP for packet processing in future editions, and probably Cisco as well (as they are the creators behind it).

The experimental testing has shown that it is possible to do high-speed routing in software using commodity hardware. Compared to the physical ASIC router tested in this thesis, there is still a way to go to accomplish those kinds of throughput and latency results, even when it was compared with a switch and not a fully functional

router. However, depending on the use of virtual routers, the performance may be good enough in many scenarios. Moreover, the use of routing in commodity hardware has many advantages over specialized hardware: (often) cheaper, better utilization, the possibility to run multiple functions (VNFs) on one physical device etc.

Chapter 8
Conclusion

The concluding chapter of this thesis presents the conclusion and suggestions for future work with the topic. In the thesis, different kinds of software have been tested when used as a VNF, and these results have been compared to each other. It has also been a test to see if it is possible to do packet processing in software and what challenges it introduces, especially when doing routing in a virtualized environment.

8.1 Summary

The experimental testing that is done in the thesis shows that it is possible to do routing in software at high-speed (10 Gbit/s and above) with commodity hardware. It has also demonstrated that the software routers can be used as VNFs, which makes them possible to use in combination with other VNFs like an IDS. The testing done in this thesis was limited with the use of only 10 Gbit/s NICs, but other research has shown that it is possible to process packets at higher speeds than that using commodity hardware. With processing speeds of 20 MPPS, that was accomplished with the VPP software router in this thesis, and this is equivalent to 142 Gbit/s if using a packet size of 888B. It is not entirely correct to assume that the number of packets processed is the same with different packet sizes, but the test has shown that the PPS that is processed is the most critical factor and that 142 Gbit/s is probably not far off.

It is possible to overcome the challenges of doing packet processing in software, where kernel bypassing and the use of multi-threading is the most important way to overcome these. Since the CPU budget is so small, every aspect of the packet processing has to be optimized. So the optimizing of memory, batch processing and offloading including other factors are also very important, and only when every aspect is optimized, is it possible to do high-performance packet processing. This partly answers the research question one about the challenges in software packet processing. The most important challenges have been highlighted, but there are still

challenges that have not been addressed in this thesis.

The different routers tested has implemented one or many of the factors to do fast packet processing, but the results show that not all of the tested routers can perform 10 Gbit/s with near realistic traffic. The Cisco CSR 1000v and pfSense is nearly able to perform at 10 Gbit/s, while the VPP and OVS router has no problem at performing this throughput. The main difference between these is that VPP and OVS can scale up the performance by adding more CPU cores. The pfSense and Cisco routers are not able to scale up by just adding more CPU cores, as they have reached their maximum performance throughput. It is clear that the routers have not as efficient packet processing like VPP and OVS, like utilising multi-cores and kernel network stack bypassing.

The batch processing of packets is the main difference between the VPP router and the OVS router, and have shown to be a key part of getting high performance. It is clear that the technology used in the VPP software is one of the best available since other products like Cisco and pfSense is planning (or have already implemented) to use VPP for packet processing. The best software router tested in this thesis is the VPP router. It has the best performance, but does not necessarily have all the functions needed for every use case scenario. In this experimental test, only basic layer 3 forwarding was tested, but usually, more advanced functions are used when performing routing. Since VPP is based on plugins, it is easy to write and implement new features as needed, so this should not be a problem. The comparison based on the test results in this thesis answers the research question two, where some software solutions available today have been explored. There are still more solutions available today, where some mentioned in the theory part of this thesis. However, based on the research, these four software routers seemed the best to be tested out.

The ability to do fast packet processing in software is essential, especially now with the 5G telecommunication network that is now being designed and implemented. It is not only routing of packets that is needed here, but other services/features like LB, IDS and proxy could benefit from the ability to process packet at high speed. For instance, the VPP could be used to do many of these features, just by adding some more plugins.

8.2 Future Work

To take this work further, it would be interesting to test the VPP software with NICs that has higher speeds than 10 Gbit/s, for instance, 40 or 100 Gbit/s. This thesis work and what other researches have concluded with have shown that there is a potential for processing packets at these speeds in software. To test how the routers perform when changes and modifications are being done during heavy load

operation would also be interesting, as most of the routers today does not just have a static configuration. For instance, adding and modifying the routing table during a heavy load operation.

In this test, only the VMWare hypervisor was tested. An experimental test where other hypervisors like Citrix or Microsoft Hyper-V were used would be interesting, especially for testing where VNFs are used in a chaining scenario. The use of containers could be tested instead of VMs, as this could boost the performance.

References

[AY] Selectel Andrej Yemelianov. introduction to dpdk: Architecture and principles. https://blog.selectel.com/introduction-dpdk-architecture-principles/. Accessed: 2019-03-06.

[BA16] Dinh Thai Bui and Kahina Aberkane. A generic interface for open vswitch. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 53–57. IEEE, 2016.

[BBGJ18] Thomas Begin, Bruno Baynat, Guillaume Artero Gallardo, and Vincent Jardin. An accurate and efficient modeling framework for the performance evaluation of dpdk-based virtual switches. *IEEE Transactions on Network and Service Management*, 15(4):1407–1421, 2018.

[BGP] BGPhelp.com. bgp table size prediction and potential impact on stability of global internet infrastructure. http://bgphelp.com/2017/01/01/bgpsize/. Accessed: 2019-04-01.

[BLM+18] David Barach, Leonardo Linguaglossa, Damjan Marion, Pierre Pfister, Salvatore Pontarelli, Dario Rossi, and Jerome Tollet. Batched packet processing for high-speed software data plane functions. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2. IEEE, 2018.

[Bro15] Jesper Dangaard Brouer. Network stack challenges at increasing speeds. In *Proceedings of the Linux Conference, Auckland, New Zealand*, pages 12–16, 2015.

[BRR+16] Alexander Beifuß, Torsten M Runge, Daniel Raumer, Paul Emmerich, Bernd E Wolfinger, and Georg Carle. Building a low latency linux software router. In *ITC*, pages 35–43, 2016.

[Cha] Madhu Challa. openvswitch performance measurements & analysis. http://www.openvswitch.org/support/ovscon2014/18/1600-ovs_perf.pptx. Accessed: 2019-04-04.

[Cisa] Cisco. cisco 1000v csr datasheet. https://www.cisco.com/c/en/us/products/collateral/routers/cloud-services-router-1000v-series/data_sheet-c78-733443.pdf. Accessed: 2019-04-01.

[Cisb] Cisco. dcisco website. https://www.cisco.com. Accessed: 2018-10-25.

[Cisc] Jiri Chaloupka Cisco. esp - evolved services platform "mozart". https://www.cisco.com/c/dam/global/cs_cz/assets/ciscoconnect/2014/assets/tech_sdp5_sp_esp_jirichaloupka.pdf. Accessed: 2019-04-09.

[CM18] Massimo Condoluci and Toktam Mahmoodi. Softwarization and virtualization in 5g mobile networks: Benefits, trends and challenges. *Computer Networks*, 146:65–84, 2018.

[Dev] Faucet Developers. installing faucet for the first time. https://docs.faucet.nz/en/latest/tutorials/first_time.html#configure-faucet. Accessed: 2019-02-16.

[ERWC15] Paul Emmerich, Daniel Raumer, Florian Wohlfart, and Georg Carle. Assessing soft-and hardware bottlenecks in pc-based packet forwarding systems. *ICN 2015*, page 90, 2015.

[FD.a] FD.IO. fd.io - the fast data project. https://fd.io/. Accessed: 2019-04-02.

[FD.b] FD.IO. vpp installation guide. https://wiki.fd.io/view/VPP/Pulling,_Building,_Running,_Hacking_and_Pushing_VPP_Code. Accessed: 2018-09-24.

[FD.c] FD.IO. vpp sample graphs of plugins. https://www.asumu.xyz/blog/img/vpp-graph-from-slides.png. Accessed: 2019-04-03.

[FD.d] FD.IO. vpp system tuning. https://wiki.fd.io/view/VPP/How_To_Optimize_Performance_(System_Tuning). Accessed: 2019-02-10.

[FD.e] FD.IO. what is vpp. https://wiki.fd.io/view/VPP/What_is_VPP%3F. Accessed: 2019-04-03.

[GES+17] Sebastian Gallenmüller, Paul Emmerich, Rainer Schönberger, Daniel Raumer, and Georg Carle. Building fast but flexible software routers. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 101–102. IEEE, 2017.

[Git] Gandi Github. packet-journey, linux router based on dpdk. https://github.com/Gandi/packet-journey. Accessed: 2019-04-09.

[HP] HP. data sheet hpe flexnetwork 5510 hiswitch series. https://h50146.www5.hpe.com/products/networking/datasheet/HPE_FlexNetwork_5510_HI_Switch_Series.pdf. Accessed: 2019-04-09.

[Hus] Geoff Huston. bgp in 2018 — the bgp table. https://blog.apnic.net/2019/01/16/bgp-in-2018-the-bgp-table/. Accessed: 2019-04-04.

[IET] IETF. rfc 2544. https://tools.ietf.org/html/rfc2544. Accessed: 2019-04-04.

[25] Emma F. (Intel). enabling vcpe with openstack. https://software.intel.com/en-us/blogs/2016/06/16/enabling-vcpe-with-openstack-get-started. Accessed: 2019-04-04.

[IT16] Oki Ishiguro, Nakajima and Takahashi. Zebra 2.0 and lagopus: newly-designed routing stack on high-performance packet forwarder. In *Nippon Telegraph and Telephone Corporation, Tokyo, Japan*. Internet Initiative Japan Inc, 2016.

[JRB18] Piotr Jurkiewicz, Grzegorz Rzym, and Piotr Boryło. Flow length and size distributions in campus internet traffic. *arXiv preprint arXiv:1809.03486*, 2018.

[KS18] Pakapol Krongbaramee and Yuthapong Somchit. Implementation of sdn stateful firewall on data plane using open vswitch. In *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–5. IEEE, 2018.

[LRP+17] Leonardo Linguaglossa, Dario Rossi, Salvatore Pontarelli, Dave Barach, Damjan Marjon, and Pierre Pfister. High-speed software data plane via vectorized packet processing. Technical report, Technical report, Telecom ParisTech, CNIT and University of Rome Tor Vergata ..., 2017.

[LWRL17] Peilong Li, Xiaoban Wu, Yongyi Ran, and Yan Luo. Designing virtual network functions for 100 gbe network using multicore processors. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 49–59. IEEE, 2017.

[Mon] Quentin Monnet. an introduction to sdn. https://qmonnet.github.io/whirl-offload/2016/07/08/introduction-to-sdn/. Accessed: 2019-03-13.

[MRdR+15] Victor Moreno, Javier Ramos, Pedro M Santiago del Río, José Luis García-Dorado, Francisco J Gomez-Arribas, and Javier Aracil. Commodity packet capture engines: Tutorial, cookbook and applicability. *IEEE Communications Surveys & Tutorials*, 17(3):1364–1390, 2015.

[Neta] Netgate. pfsense roadmap. https://www.netgate.com/blog/application-detection-on-pfsense-software.html. Accessed: 2019-04-02.

[Netb] Netgate. pfsense website. https://www.pfsense.org/. Accessed: 2019-04-02.

[NLH+15] Yukihiro Nakagawa, Chunghan Lee, Kazuki Hyoudou, Shinji Kobayashi, Osamu Shiraki, Jun Tanaka, and Tomohiro Ishihara. Dynamic virtual network configuration between containers using physical switch functions for nfv infrastructure. In *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 156–162. IEEE, 2015.

[Omn] Omnipacket. wireedit - full stack wysiwyg pcap editor. https://wireedit.com/. Accessed: 2019-04-26.

[Ope] Openvswitch.org. open vswitch with dpdk installation. http://docs.openvswitch.org/en/latest/intro/install/dpdk/. Accessed: 2019-02-15.

[PFNR16] Michele Paolino, Jérémy Fanguède, Nikolay Nikolaev, and Daniel Raho. Turning an open source project into a carrier grade vswitch for nfv: Vosyswitch challenges & results. In *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pages 22–27. IEEE, 2016.

[Pro] Linux Foundation Collaborative Project. ovs features. http://www.openvswitch.org/features/. Accessed: 2019-04-04.

[Shi18] Kohei Shiomoto. Research challenges for network function virtualization-re-architecting middlebox for high performance and efficient, elastic and resilient platform to create new services. *IEICE Transactions on Communications*, 101(1):96–122, 2018.

[Sie] Eric Siebert. top 10 hypervisors: Choosing the best hypervisor technology. https://searchservervirtualization.techtarget.com/tip/Top-10-hypervisors-Choosing-the-best-hypervisor-technology. Accessed: 2019-03-13.

[Tt] TRex-tgn. about trex. https://trex-tgn.cisco.com/#book. Accessed: 2019-04-04.

[VMw] VMware. ntel® data plane development kit (intel® dpdk) with vmware vsphere®. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/intel-dpdk-vsphere-solution-brief-white-paper.pdf. Accessed: 2019-03-13.

[Wir] Wireshark.org. wireshark network protocol analyzer. https://www.wireshark.org/. Accessed: 2019-04-26.

[WN10] Guohui Wang and TS Eugene Ng. The impact of virtualization on network performance of amazon ec2 data center. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

Appendix A

IP Plan Management network

Hostname	IP MGMT	Description	Host
Firewall	192.168.100.1	pfSense 2.4.4	ESXi-1
vCenter	192.168.100.8	Vmware vCenter	ESXi-2
ESXi host 1	192.168.100.10	Vmware ESXi 6.7	ESXi-1
ESXi host 2	192.168.100.11	Vmware ESXi 6.7	ESXi-2
MGMT-server	192.168.100.15	Ubuntu Server 18.04 LTS	ESXi-1
HP 5510	192.168.100.2	JH145A HPE 5510	N/A
Lagopus testbed	192.168.100.20	Ubuntu Server 18.04 LTS	ESXi-1
Cisco V1000 testbed	192.168.100.21	Cisco IOS	ESXi-1
pfSense testbed	192.168.100.22	pfSense 2.4.4	ESXi-1
VPP testbed	192.168.100.23	Ubuntu Server 18.04 LTS	ESXi-1
Open vSwitch	192.168.100.24	Ubuntu Server 18.04 LTS	ESXi-1
MoonRoute	192.168.100.25	Ubuntu Server 18.04 LTS	ESXi-1
Traffic-tester-1	192.168.100.30	Ubuntu Server 18.04 LTS	ESXi-2
Traffic-tester-2	192.168.100.31	Ubuntu Server 18.04 LTS	ESXi-2
Trex	192.168.100.32	Ubuntu Server 14.04 LTS	ESXi-2
Windows 10 Client	192.168.100.33	Windows 10	ESXi-2
Ubuntu bare metal	192.168.100.26	Ubuntu Server 18.04 LTS	N/A
VPP SR-IOV 1	192.168.100.27	Ubuntu Server 18.04 LTS	ESXi-1
VPP SR-IOV 2	192.168.100.28	Ubuntu Server 18.04 LTS	ESXi-1
VPP SR-IOV 3	192.168.100.43	Ubuntu Server 18.04 LTS	ESXi-1
VPP SR-IOV 4	192.168.100.44	Ubuntu Server 18.04 LTS	ESXi-1
VPP SR-IOV 5	192.168.100.45	Ubuntu Server 18.04 LTS	ESXi-1
VPP SR-IOV 6	192.168.100.46	Ubuntu Server 18.04 LTS	ESXi-1
VPP SR-IOV 7	192.168.100.47	Ubuntu Server 18.04 LTS	ESXi-1
VPP SR-IOV 8	192.168.100.48	Ubuntu Server 14.04 LTS	ESXi-1

Appendix B
Automation test script

```
#!/bin/bash

time=`date +"%H%M"`
echo "$time: Starting TRex network througput test with router $name"


date=`date +"%Y-%m-%d-%H%M"`
outfile="/home/mgmt/tmp.txt"

result="/home/mgmt/Samba/Results/${date}-Result"
result="/home/mgmt/${date}-Result"
test_cores="1 2 4 6"
test_time="60"

test_2_way () {
test_m='4 6 8 10 12 14 16 18 20 22 24 30 35 40 45 50 55 60 65 70 75
↪  80 85 90 95 100 105 110 115 120 125 130 135 140 150 160 170 180
↪  190 200'

for i in $test_m; do
        time=`date +"%H%M"`
        sudo /opt/trex/v2.47/t-rex-64 -f cap2/udp_64B_2_ways.yaml -c
        ↪  14 -l 1000 -m $i -d $test_time >> $outfile
        #sudo /opt/trex/v2.47/t-rex-64 -f cap2/udp_888B_2_ways.yaml
        ↪  -c 14 -l 1000 -m $i -d $test_time >> $outfile
        Expected_PPS=`head -n 65 $outfile | grep "Expected-PPS" | tr
        ↪  -cd '[[:digit:]]'`
        total_tx_pkt=`head -n 1000000 $outfile | grep "Total-tx-pkt"
        ↪  | tr -cd '[[:digit:]]'`
```

```
24          total_rx_pkt=`head -n 1000000 $outfile | grep "Total-rx-pkt"
            ↪  | tr -cd '[[:digit:]]'`
25          maximum_latency=`head -n 1000000 $outfile | grep
            ↪  "maximum-latency" | tr -cd '[[:digit:]]'`
26          average_latency=`head -n 1000000 $outfile | grep
            ↪  "average-latency" | tr -cd '[[:digit:]]'`
27          Total_sw_err=`head -n 1000000 $outfile | grep "Total-sw-err"
            ↪  | tr -cd '[[:digit:]]'`
28          Total_sw_tx_pkt=`head -n 1000000 $outfile | grep
            ↪  "Total-sw-tx-pkt" | tr -cd '[[:digit:]]'`
29          Packets_lost=`expr $total_tx_pkt - $total_rx_pkt`
30
31          echo "$time: $2 Core(s), $3, M: $i, PPS: $Expected_PPS,
            ↪  Packets lost: $Packets_lost, Average latency:
            ↪  $average_latency, Max latency: $maximum_latency"
32
33          echo "$i, $total_tx_pkt, $total_rx_pkt, $maximum_latency,
            ↪  $average_latency, $Total_sw_err, $Total_sw_tx_pkt" >>
            ↪  $result$1
34          rm $outfile
35  done
36  }
37
38
39  test_VPP_no_virtualization_2_routes () {
40  for c in $test_cores; do #test all the cores with 2 routes
41          command="sudo cp /etc/vpp/vpp-$c-core.conf
            ↪  /etc/vpp/startup.conf; sudo cp /home/mgmt/2-routes.txt
            ↪  /home/mgmt/startup-vpp.txt; sudo service vpp restart"
42          sudo ssh mgmt@192.168.100.26 $command
43          sleep 5
44          result_output="VPP-bare-m-$c-cores-2-routes.csv"
45          echo "M, Total-tx-pkt, Total-rx-pkt, Maximum-latency,
            ↪  Average-latency, Total-sw-err, Total-sw-tx-pkt" >>
            ↪  $result$result_output
46          test_2_way "$result_output" "$c" "2-routes"
47  done
48  }
49
50  test_VPP_2_routes () {
51  for c in $test_cores; do #test all the cores with 2 routes
```

```
52          command="sudo cp /etc/vpp/vpp-$c-core.conf
            ↪ /etc/vpp/startup.conf; sudo cp /home/mgmt/2-routes.txt
            ↪ /home/mgmt/startup-vpp.txt; sudo service vpp restart"
53          sudo ssh mgmt@192.168.100.23 $command
54          sleep 5
55          result_output="-$c-cores-2-routes.csv"
56          echo "M, Total-tx-pkt, Total-rx-pkt, Maximum-latency,
            ↪ Average-latency, Total-sw-err, Total-sw-tx-pkt" >>
            ↪ $result$result_output
57          test_2_way "$result_output" "$c" "2-routes"
58  done
59  }
60
61
62  ovs_test_cores="1 6 1E 7E" #1, 2, 4, 6 cores
63  test_ovs_2_routes () {
64  for c in $ovs_test_cores; do #test all the cores with 2 routes
65          command="sudo ovs-vsctl set Open_vSwitch .
            ↪ other_config:pmd-cpu-mask=0x$c"
66                  sudo ssh mgmt@192.168.100.24 $command
67          sleep 15
68          result_output="OVS-0x$c-cores-2-routes.csv"
69          echo "M, Total-tx-pkt, Total-rx-pkt, Maximum-latency,
            ↪ Average-latency, Total-sw-err, Total-sw-tx-pkt" >>
            ↪ $result$result_output
70          test_2_way "$result_output" "$c" "2-routes"
71  done
72  }
73
74  #test_ovs_2_routes
75
76  test_2_way "-VPP-1-core-8-chain-2-routes-0-ACL.csv" "1"
    ↪ "8-chain-2-routes-0-ACL"
77
78  #test_VPP_no_virtualization_2_routes
```

Appendix C

Script for generating ACL and routing table

Powershell script for IP route generation:

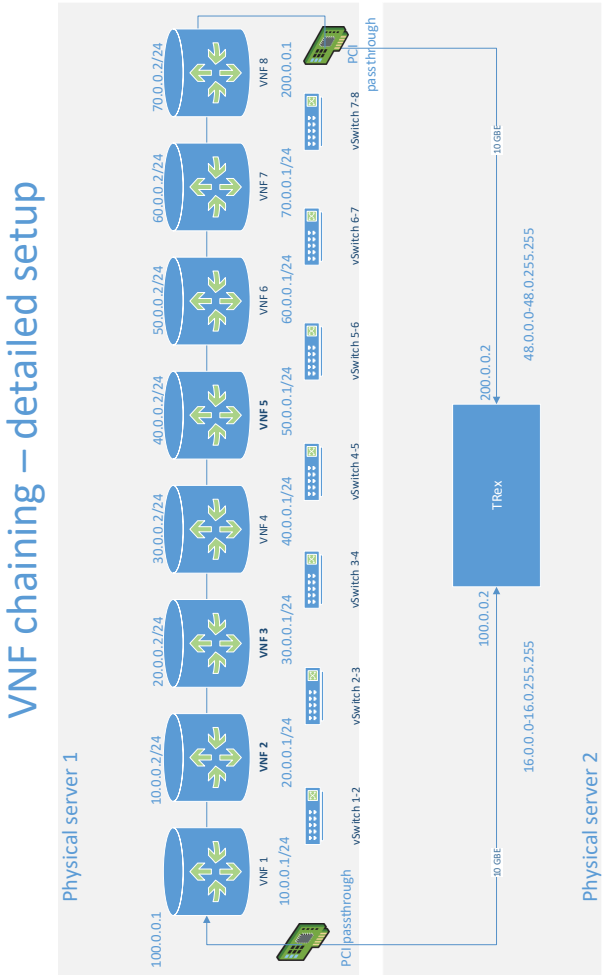
```
#Network 100: 16.0.0.0/8
#Network 200: 48.0.0.0/8

$location_file = "C:\Users\Fredrik\Dropbox\NTNU\Masteroppgave\Lab setup\Ruting tables\Ip_table_Cisco_262k.txt"

$b_num = [int]"3"
$c_num = [int]"255"
$d_num = [int]"255"


$a = [int]"16"
$b = [int]"0"
$c = [int]"0"
$d = [int]"0"


while ($b -le $b_num) {
    $c = 0
    while ($c -le $c_num) {
        $d = 0
        while ($d -le $d_num) {
            #Output for the Cisco router:
                        echo "ip route $a.$b.$c.$d 255.255.255.255 100.0.0.2" | Add-Content -Path $location_file

```

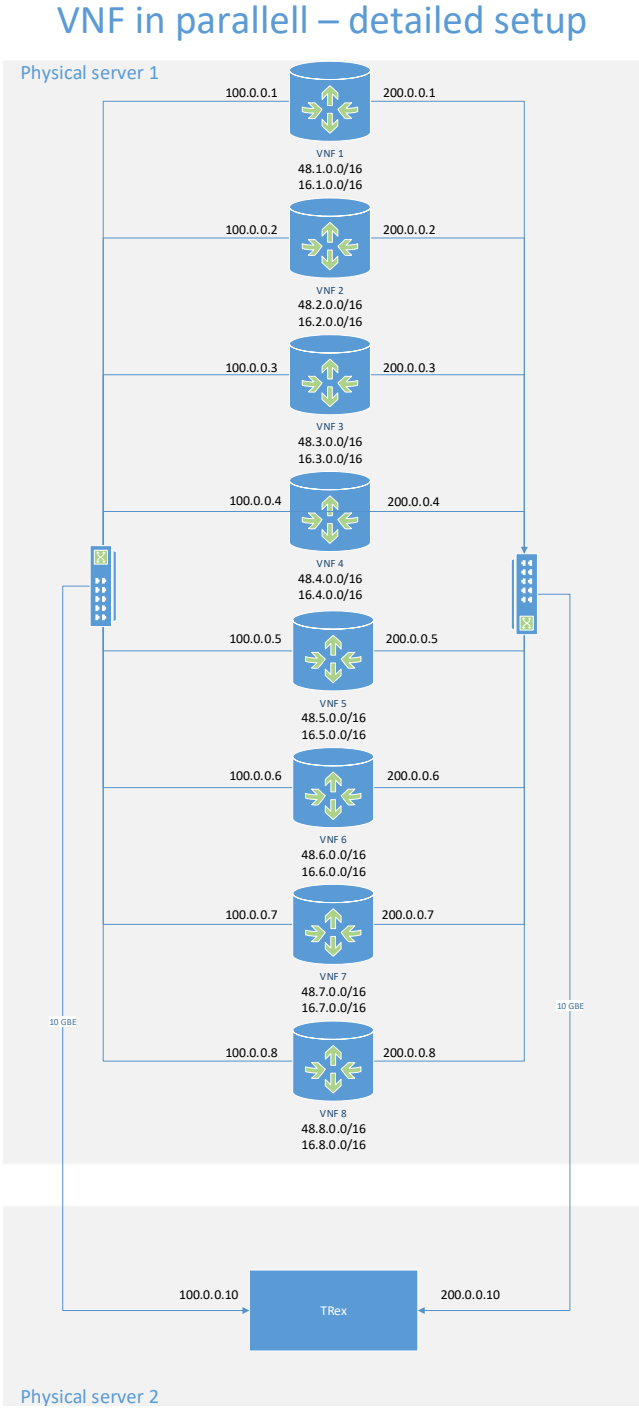
```
25                          #Output for the HP Switch:
26                                  #echo "ip route-static $a.$b.$c.$d
                                    ↪  32 Vlan-interface100 100.0.0.2"
                                    ↪  | Add-Content -Path
                                    ↪  $location_file
27
28                          #Output for the pfSense routing tables:
29                                  #echo "`t<route>" | Add-Content
                                    ↪  -Path $location_file
30                                  #echo "`t`t<network>$a.$b.$c.$d/32
                                    ↪  </network>" | Add-Content -Path
                                    ↪  $location_file
31                                  #echo
                                    ↪  "`t`t<gateway>GW_100</gateway>"
                                    ↪  | Add-Content -Path
                                    ↪  $location_file
32                                  #echo "`t`t<descr></descr>" |
                                    ↪  Add-Content -Path
                                    ↪  $location_file
33                                  #echo "`t</route>" | Add-Content
                                    ↪  -Path $location_file
34
35                          #Output for the OVS router:
36                                  #echo "            - route:" |
                                    ↪  Add-Content -Path
                                    ↪  $location_file
37                                  #echo "               ip_dst:
                                    ↪  `"$a.$b.$c.$d/32`"" |
                                    ↪  Add-Content -Path
                                    ↪  $location_file
38                                  #echo "               ip_gw:
                                    ↪  '100.0.0.2'" | Add-Content -Path
                                    ↪  $location_file
39              $d += 1
40          }
41          $c +=1
42      }
43      $b +=1
44  }
```

Powershell script for ACL generation:

```
#Network 100: 16.0.0.0/8
#Network 200: 48.0.0.0/8

$location_file = "C:\Users\Fredrik\Dropbox\NTNU\Masteroppgave\Lab setup\Ruting tables\Ip_ACL_Cisco_1024.txt"

$b_num = [int]"0"
$c_num = [int]"1"
$d_num = [int]"255"


$a = [int]"16"
$b = [int]"0"
$c = [int]"0"
$d = [int]"0"


while ($b -le $b_num) {
    $c = 0
    while ($c -le $c_num) {
        $d = 0
        while ($d -le $d_num) {
            #Cisco:
                        echo "access-list 103 permit ip $a.$b.$c.$d 0.0.0.0 48.0.0.0 0.255.255.255" | Add-Content -Path $location_file
                        #HP Switch:
                        #echo "rule permit source $a.$b.$c.$d 0.0.0.0" | Add-Content -Path $location_file
                        #VPP:
                        #Write-Host "src $a.$b.$c.$d/32," -nonewline
            $d += 1
        }
        $c +=1
    }
    $b +=1
}
```

Appendix D
VNF chaining setup




VNF in parallell – detailed setup
Physical server 1
100.0.0.1
200.0.0.1
VNF 1
48.1.0.0/16
16.1.0.0/16
100.0.0.2
200.0.0.2
VNF 2
48.2.0.0/16
16.2.0.0/16
100.0.0.3
200.0.0.3
VNF 3
48.3.0.0/16
16.3.0.0/16
100.0.0.4
200.0.0.4
VNF 4
48.4.0.0/16
16.4.0.0/16
100.0.0.5
200.0.0.5
VNF 5
48.5.0.0/16
16.5.0.0/16
100.0.0.6
200.0.0.6
VNF 6
48.6.0.0/16
16.6.0.0/16
100.0.0.7
200.0.0.7
VNF 7
48.7.0.0/16
16.7.0.0/16
100.0.0.8
200.0.0.8
VNF 8
48.8.0.0/16
16.8.0.0/16
10 GBE
10 GBE
100.0.0.10
200.0.0.10
TRex
Physical server 2

Appendix E
Detailed test results

1-Core configuration

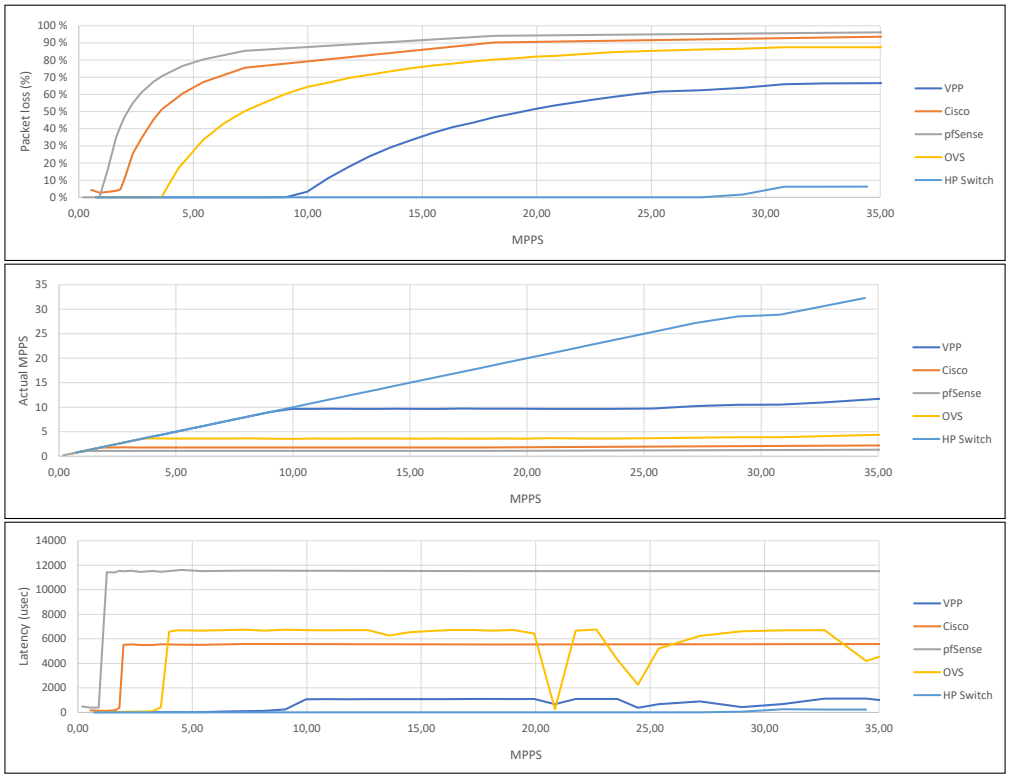


Figure E.1: Performance results 1-core configuration

2-Core configuration

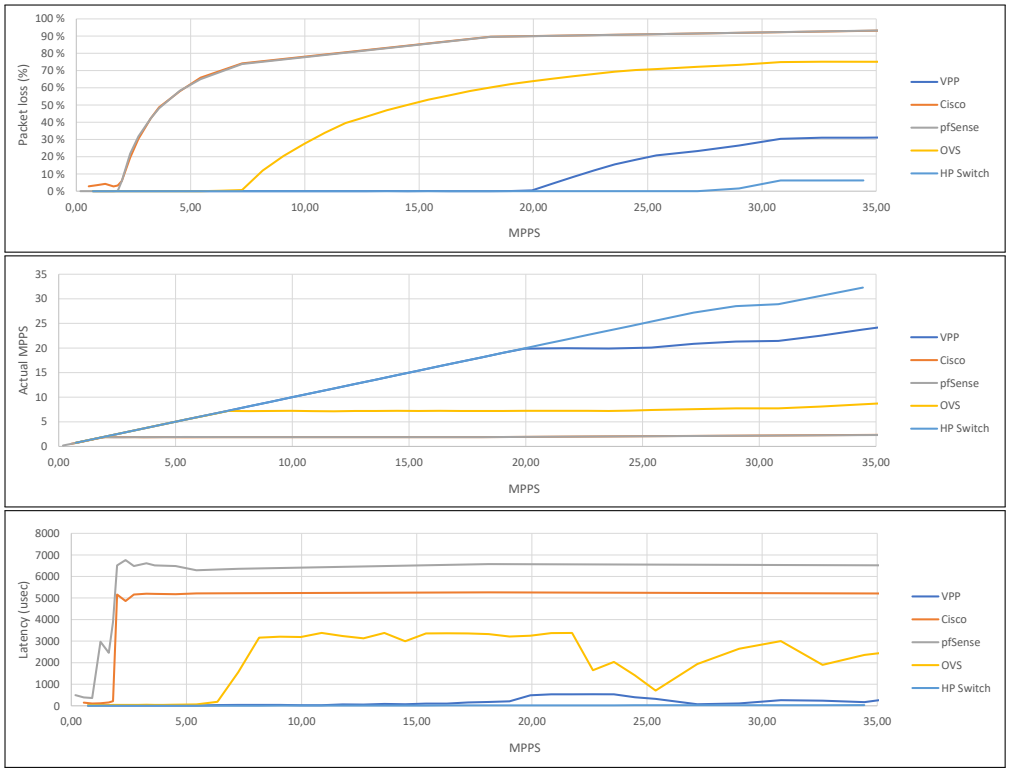


Figure E.2: Performance results 2-core configuration

4-Core configuration

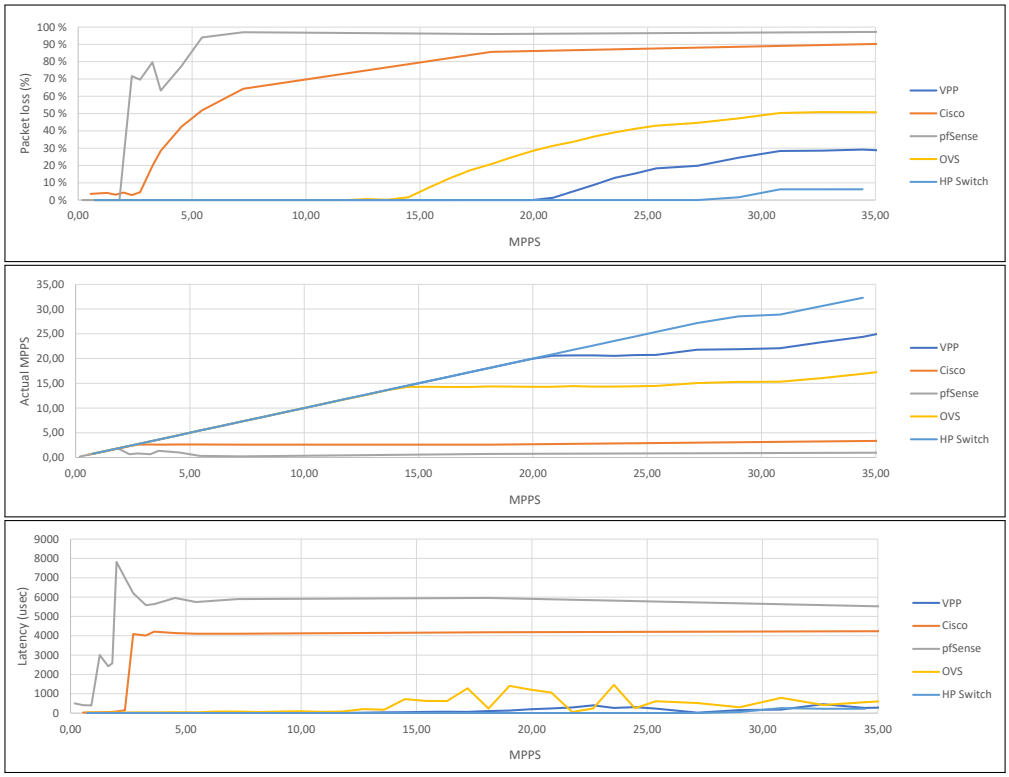


Figure E.3: Performance results 4-core configuration

VPP with different features

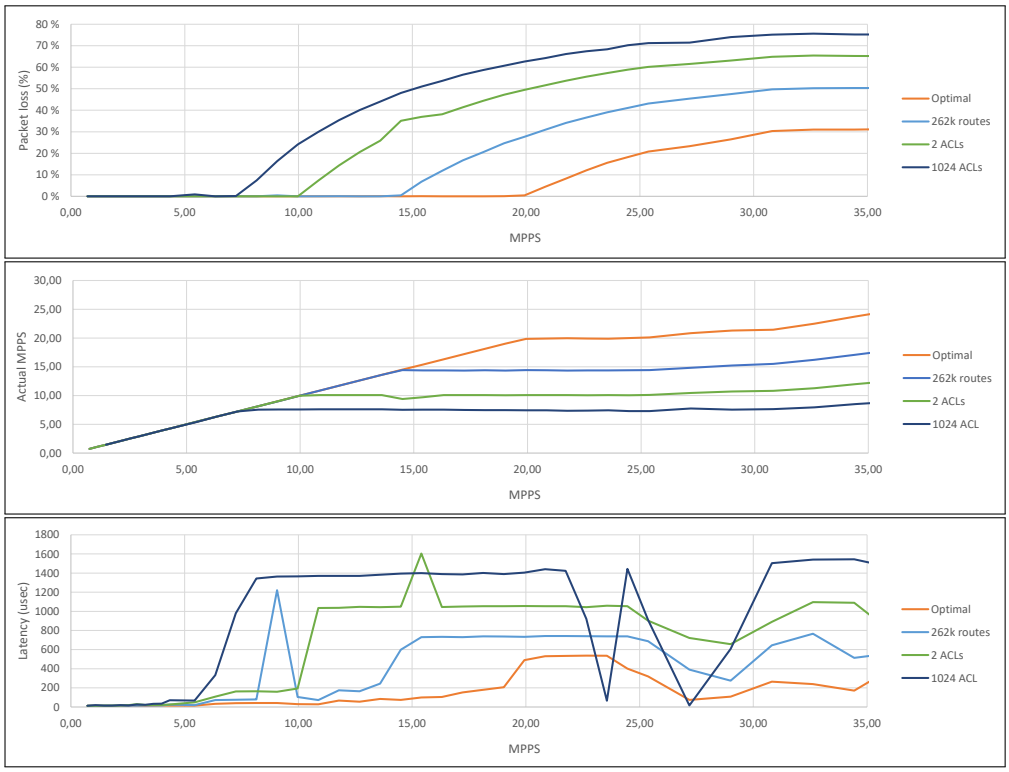


Figure E.4: Performance results VPP with different features

Cisco with different features

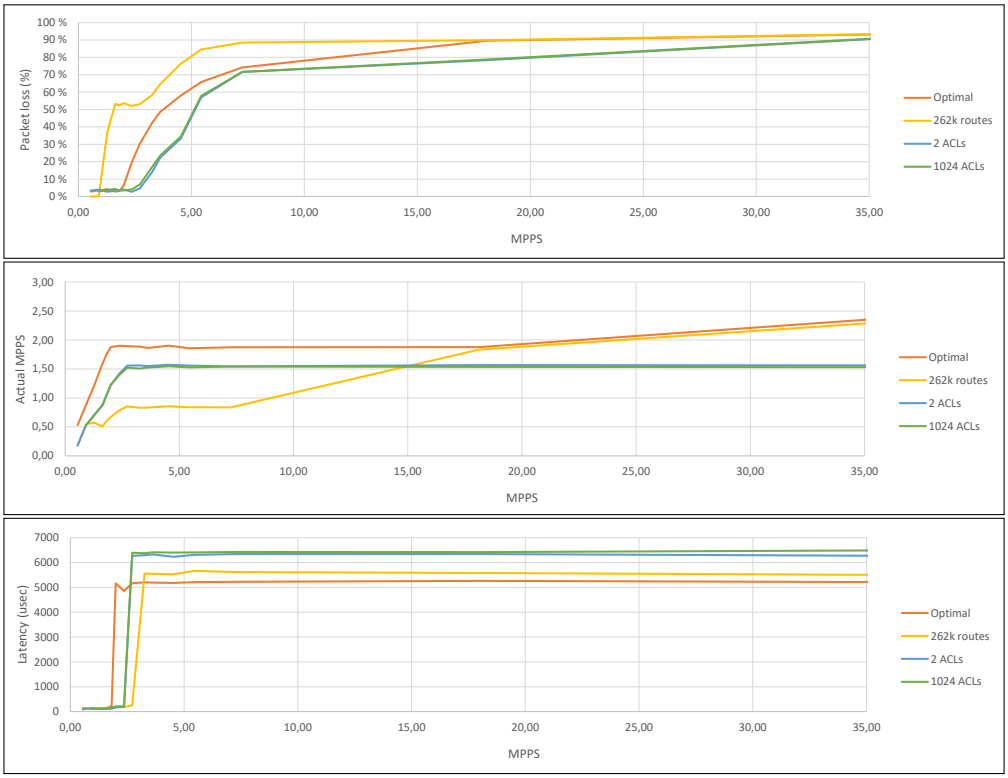


Figure E.5: Performance results Cisco with different features

pfSense with different features

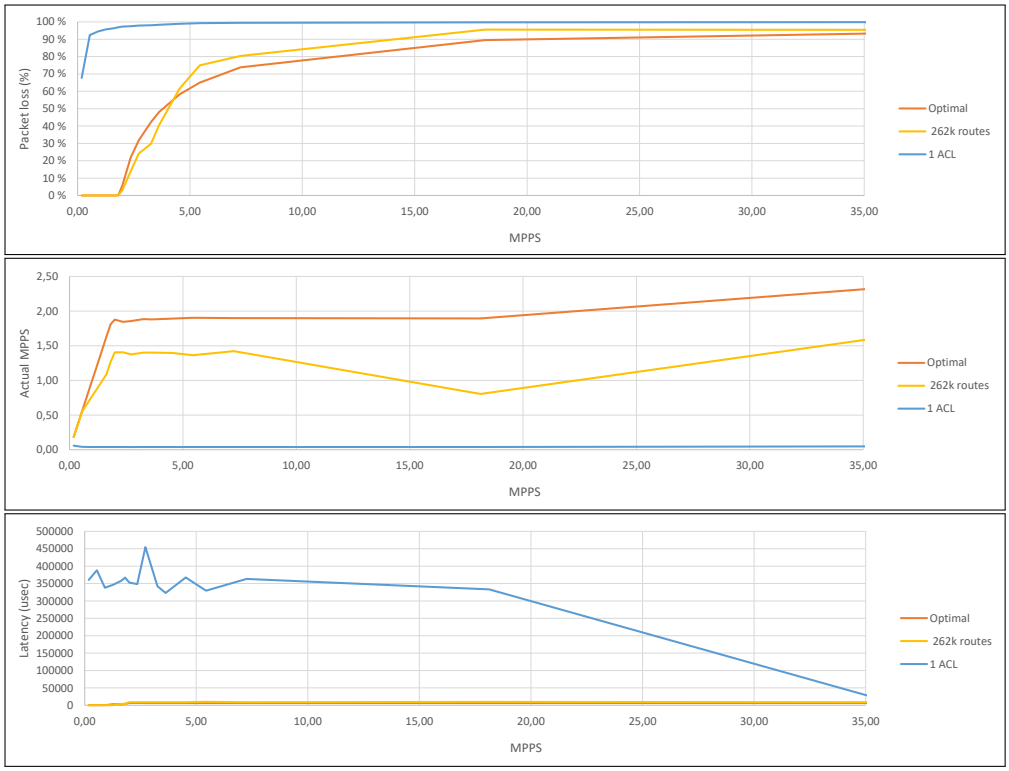


Figure E.6: Performance results pfSense with different features

HP Switch with different features

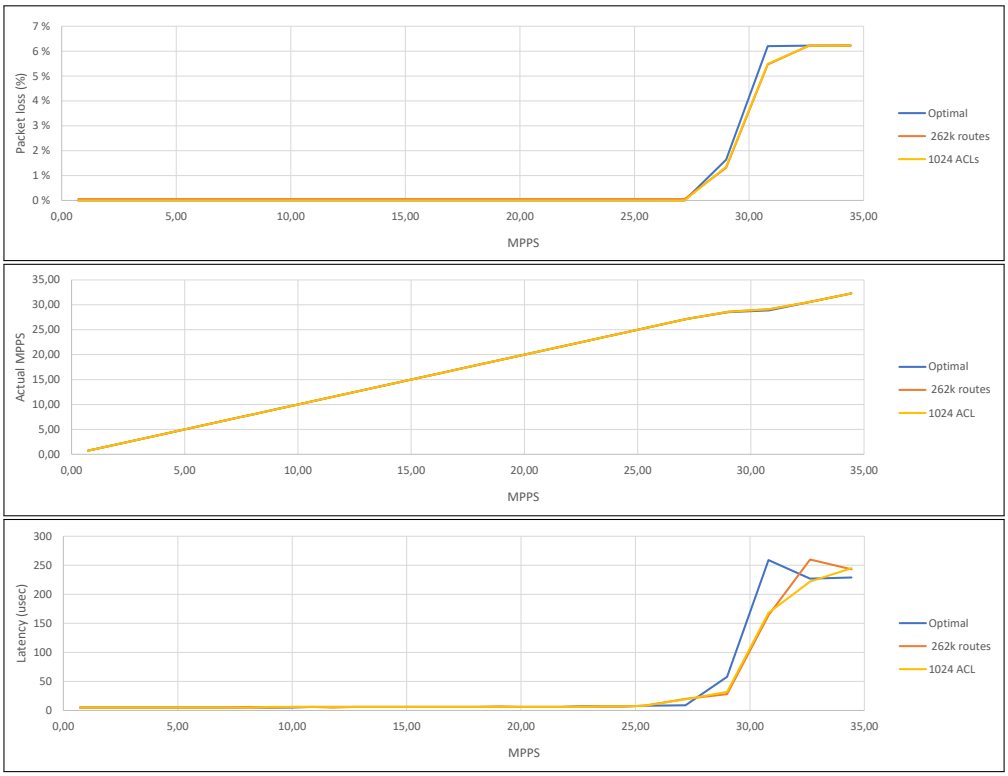


Figure E.7: Performance results HP Switch with different features

Results with and without virtualization

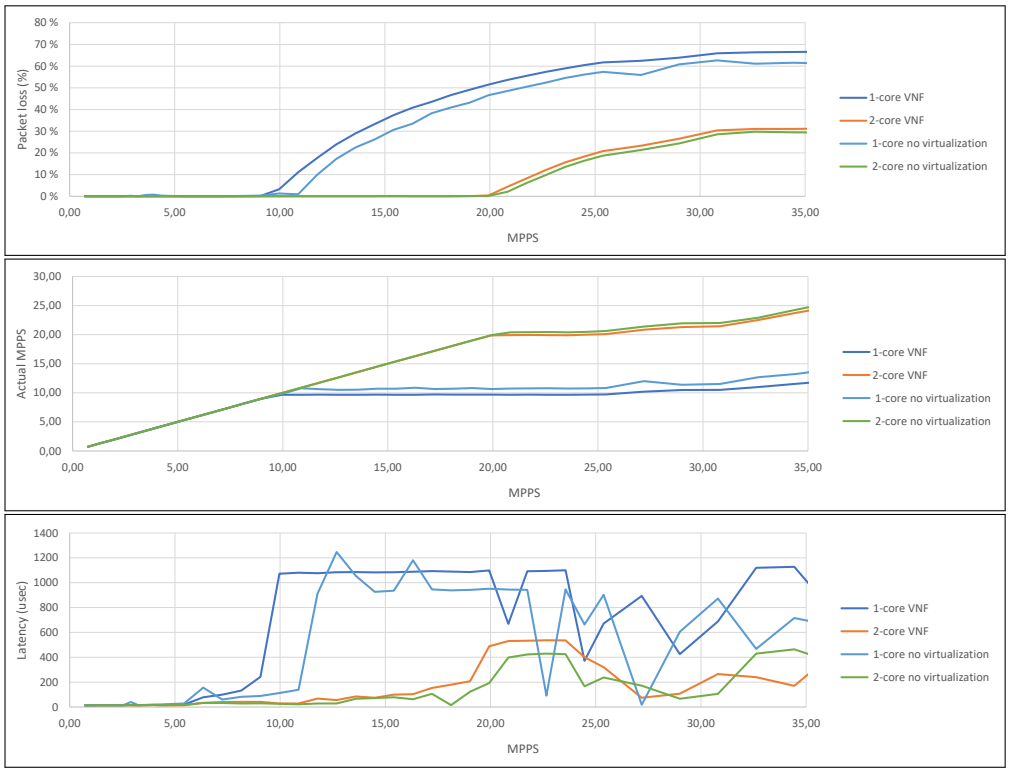


Figure E.8: Performance results with and without virtualization

Results VPP in parallel

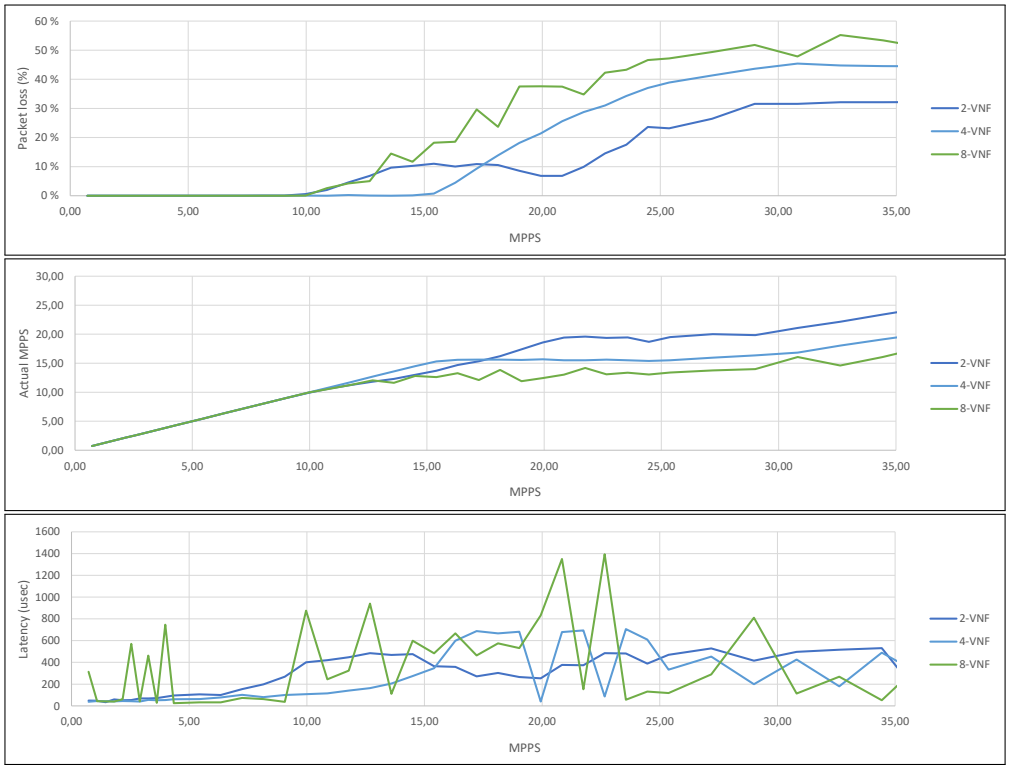


Figure E.9: Performance results for VPP in parallel

Results VPP in chaining

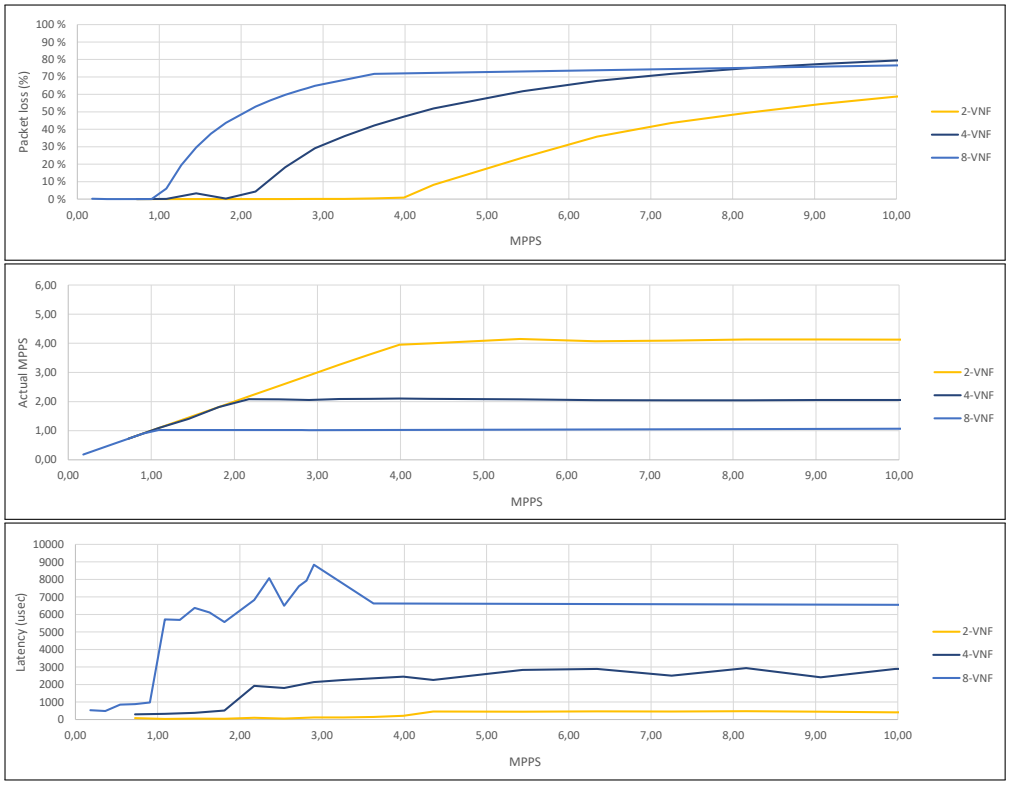


Figure E.10: Performance results for VPP in chaining mode

Fredrik Andreassen

Experimental assessment on routing of network traffic in software using commodity hardware

NTNU
Kunnskap for en bedre verden