Bachelor Degree Project

NFtables och IPtables

En jämförelse av latens

NFtables and IPtables

A Comparison in Latency

Bachelors Degree Project in Computer Science
Network and Systems Administration, G2E, 22.5 hp
IT604G

Jonas Svensson Eidsheim
a14jonei@student.his.se

Examiner
Jonas Gamalielsson

Supervisor
Johan Zaxmy

Abstract

Firewalls are one of the essential tools to secure any network. IPtables has been the de facto firewall in all Linux systems, and the developers behind IPtables are also responsible for its intended replacement, NFtables. Both IPtables and NFtables are firewalls developed to filter packets. Some services are heavily dependent on low latency transport of packets, such as VoIP, cloud gaming, storage area networks and stock trading. This work is aiming to compare the latency between the selected firewalls while under generated network load. The network traffic is generated by iPerf and the latency is measured by using ping. The measurement of the latency is done on ping packets between two dedicated hosts, one on either side of the firewall. The measurement was done on two configurations one with regular forwarding and another with PAT (Port Address Translation). Both configurations are measured while under network load and while not under network load. Each test is repeated ten times to increase the statistical power behind the conclusion. The results gathered in the experiment resulted in NFtables being the firewall with overall lower latency both while under network load and not under network load.

Abstrakt

Brandväggen är ett av de viktigaste verktygen för att säkra upp nätverk. IPtables har varit den främst använda brandväggen i alla Linux-system och utvecklarna bakom IPtables är också ansvariga för den avsedda ersättaren, NFtables. Både IPtables och NFtables är brandväggar som utvecklats för att filtrera paket. Vissa tjänster är starkt beroende av att paket som skickas anländer med låg latens. Tjänster som VoIP, cloud gaming, lagringsnät och aktiehandel. Detta arbete syftar till att jämföra latensen mellan de valda brandväggarna under en genererad nätverkslast. Nätverkslasten genereras av iPerf och latensen mäts med hjälp av ping. Mätningen av latensen görs på pingpaketen mellan två dedikerade värdar, en på vardera sida av brandväggen. Mätningen gjordes på två olika konfigurationer, en med vidarebefordran och en annan med portadressöversättning (eng. PAT, Port Address Translation). Båda konfigurationerna mäts både under nätverksbelastning och utan nätverksbelastning. Varje test upprepas tio gånger för att öka den statistiska signifikansen bakom slutsatsen. Resultaten som samlats in i experimentet visade att NFtables var brandväggen med generell lägre latens både under last och inte under last.

Popular scientific summary

A firewalls primary purpose is to filter traffic to and from the internet. This traffic is made up of IP packets. The IP packet is made of a header, some contents and trailer. IP is the primary protocol used on the internet today to send packets from one source on the Internet to a destination somewhere else on the internet. The primary information found in an IP packet header is the destination address, the source address and its time-to-live. The time-to-live field is a counter that goes down during the packet's transport. When this reaches zero, then it is dropped by the first router that processes the packet next. Rules can be configured to match this IP header information and the IP packet contents to determine if the packet should be let through the firewall or if the packet should just be dropped. A firewall can also make decisions based on information found deeper in the packet but the required processing increases for every layer deeper the firewall goes.

Firewalls can also change information in packets before passing it on to the next device. This is often done on regular home routers in a process called PAT (Port Address Translation). This process changes the stated source address in packets that are going out to the internet and the destination address on the way back from the internet. This is done by mapping ports, an element found in the packet that is contained in the IP packet to one single IP address. This allows for multiple internet connected devices to communicate on the internet from a single address.

Both of these processes add to the time taken from the source device to the destination device. The time taken from the source to its destination is called latency. Latency is an issue for services that are sensitive to delays such as Voice over IP, used for talking with others over the internet. Other services that are affected are cloud gaming, where the game that is played is not rendered on the computer where the game is shown. The game itself is rendered on a server with specialised hardware to do that processing. The delay from when the player is pressing a key to the time when a reaction is had is noticeable if the delay on the way to and from the server is high.

Two firewalls were selected for this study, the current firewall that has been around since 1998, IPtables and its intended replacement, NFtables. NFtables has since 2014 slowly been rolling out in the most popular distributions of Linux. New implementations of Linux-based firewalls are recommended to use NFtables instead of its predecessor, IPtables.

This study aimed to determine how the latency compares between IPtables and NFtables in its current state while under a generated network load, each with two configurations. The first explained above using forwarding and the other configuration using PAT, also explained above. The generated traffic is originating from a source in one of the networks and is continuously sending traffic through the firewall to the network on the opposing side of the firewall. During the network load, another machine on each side of the firewall is sending pings to one another. The results show that there is not much difference between both firewalls with the given criteria. NFtables is marginally better than IPtables with only a few microseconds differentiating them.

Populärvetenskaplig sammanfattning

En brandväggs huvudsakliga syfte är att filtrera trafik till och från internet. Denna trafik består av IP-paket. Ett IP-paket består av ett huvud, innehåll och en svans. IP är det primära protokollet som används på internet idag för att skicka paket från en källa på internet till en destination någon annanstans på internet. Den primära informationen som finns i pakethuvudet i ett IP-paket är en destinationsadress, en källadress och antalet routingsteg som är kvar innan paketet måste ha nått en destination. Tiden för att nå sin destination är ett fält som räknas ner på vägen mot destinationen. När denna siffra når noll så kastas den av nästa router som den anländer vid. Regler kan konfigureras för att matcha information som hittas i IP-huvudet och innehållet i paketet för att avgöra om paketet ska släppas igenom brandväggen eller ifall paketet bara ska slängas. En brandvägg kan även fatta beslut utifrån information som finns djupare i paketet men ifall en brandvägg behöver hämta denna information så kräver detta mer bearbetning och tid.

Brandväggar kan också skriva om adresserna i paket innan de skickas vidare till nästa enhet. Detta görs ofta på vanliga hem-routrar i en process som heter portadressöversättning (eng. PAT, Port Address Translation). Denna process ändrar källadressen i paket som går ut mot internet och destinationsadressen på vägen tillbaka från internet. Detta görs genom att kartlägga portar som finns i det inkapslade paketet inom IP-paketet. Genom att kartlägga porten så kan brandväggen översätta en intern adress till en enda extern IP-adress. Detta gör det möjligt för flera internetanslutna enheter att kommunicera på internet från en enda adress.

Båda dessa processer kräver tid som läggs till på den totala tid som krävts från källenheten till destinationsenheten. Tiden som tas från källan till dess destination kallas latens. Latensen är ett problem för tjänster som är känsliga för förseningar som Röst över IP (eng. VoIP, Voice over IP), som används för att prata med andra via internet. Andra tjänster som påverkas är bland annat molnspel, där spelet som spelas inte bearbetas på samma dator som spelet visas på. Spelet i sig bearbetas på en server med specialiserad hårdvara för att göra detta. Förseningar från när spelaren trycker på en knapp till att spelet svarar på knapptrycket resulterat i något är märkbart om förseningen på vägen till och från servern är hög.

Två brandväggar valdes ut för denna studie. Den nuvarande brandväggen som har funnits sedan 1998, IPtables och dess avsedda ersättare, NFtables. NFtables har sedan 2014 långsamt rullat ut i de mest populära distributionerna av Linux. Nya implementeringar av Linux-baserade brandväggar rekommenderas att använda NFtables istället för dess föregångare, IPtables.

Denna studie har haft som målsättning att ta reda på hur hög latensen är i jämförelse mellan IPtables och NFtables under en genererad nätverksbelastning, var och en med två olika konfigurationer. Den genererade trafiken kommer från en källa i ett av nätverken och skickar kontinuerligt trafik genom brandväggen till nätverket på brandväggens motsatta sida. Under nätverksbelastningen skickar en annan maskin på varje sida av brandväggen pingpaket till varandra. Resultaten visar att det är väldigt liten skillnad mellan IPtables och NFtables både

under genererad belastning och utan. Resultaten visar att även ifall det är väldigt liten skillnad mellan brandväggarna så är ändå NFtables den brandvägg som har lägst latens.

Table of contents

Appendix F – Data gathering script

1 Introduction

Firewalls should be used in any system that connects to the internet in one way or another and is essential to stop the most immediate threats (Microsoft, n.d.). A firewall can on many different layers of the network stack filter based on the contents of the packet it is processing. The primary task of any firewall is to filter out unwanted traffic. Two firewalls and firewall frameworks are the older IPtables which utilizes the netfilter framework and the newer NFtables which utilizes the nftables framework.

The results of this work are collected through a quantitative experiment where the latency of ICMP packets is collected. The tests are conducted with multiple configurations. The configurations are with normal forwarding and with network address translation. Data is collected for both firewalls while not under load and while under load from iPerf. The collected data is then compared to determine if the results are answering the research questions found in section 3.2 and if the hypotheses found in section 3.4. is correct. The results are also compared with previous research which is explained further in section 3.3.

The paper is structured as follows: the background is presented in section two, the problem definition is presented in section three, the method for the paper is presented in section four, validity is discussed in section five, and the results are presented in section six. The conclusion that can be drawn from the results is presented in section seven and finishing with suggestions of future work in section eight.

2 Background

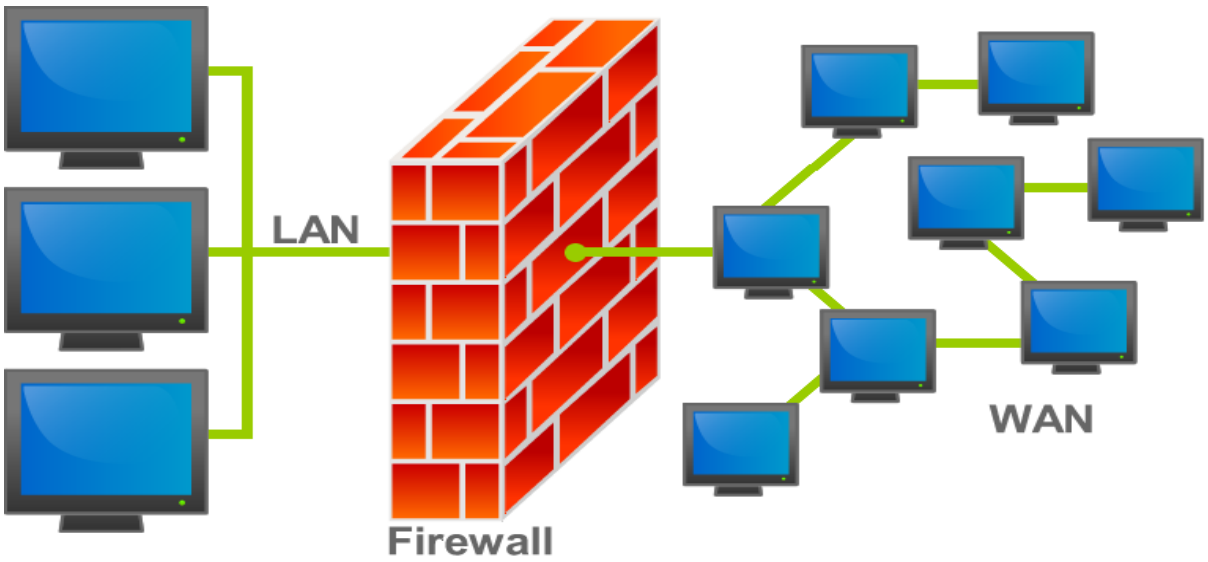

Figure 2.1 The purpose of a firewall
("An illustration of where a firewall would be located in a network" by Bruno Pedrozo is licensed under CC BY-SA 3.0)

According to the Oxford dictionary, a firewall is defined as “A part of a computer system or network which is designed to block unauthorised access while permitting outward communication.” (Oxford Dictionary, n.d.). Figure 2.1. shows a typical placement of a firewall between a LAN (Local Area Network) and a WAN (Wide Area Network) such as the internet.

A firewall is designed to perform several different tasks as it is most often placed on the border of the internet and a home network. Since the firewall is placed on the border it can be used to redirect traffic, filter incoming traffic to the firewall and forward traffic that is sent through the firewall according to a predefined policy (Ioannidis, S., Keromytis, A., Bellovin, S. & Smith, J., 2000).

A firewall could be either stateful or stateless (Kearns, 2017). A stateful firewall keeps track of currently active connections and allows returning answers based on the outgoing traffic. This is done by looking through the IP header of outgoing TCP packets and opening the port which is used to send the packet, allowing the traffic to flow back in. UDP and ICMP are harder for a stateful firewall to deal with so other data in the header are used in combination to keep track of sessions (Wilkins, 2013).

A stateless firewall only accepts incoming packages that have been explicitly allowed in the firewall rules (Suehring, 2014). If a packet is received that is not explicitly allowed then it will be dropped. A stateless firewall is not the best choice from a security nor an administration perspective since it requires all ports that could be needed to be explicitly allowed through the firewall. It could also leave open ports which could pose a threat to the security of the network.

The term “firewall” is used loosely throughout the article and depending on the context can refer to either the hardware that the firewall software is being run on or the actual software running on the hardware.

NAT (Network Address Translation) is the concept of translating the source IP address on packets from one host to another address which is routable from the Internet. The packets on the way back have the destination IP address swapped out for the associated address on the inside (Cisco, 2017).

PAT (Port Address Translation) is basically the same as NAT but instead of translating one IP address to another, PAT allows multiple IP addresses to be masked behind one IP address. Usually, this involves the use of private IP addresses that are not routed on the internet to be masked behind a public IP address that is routed on the internet. PAT has made it possible to keep using IPv4 addresses even though all have been used up (Cisco, 2017).

Some firewalls are can be configured to allow ports through the firewall based on the ports that the program itself is listening to. This makes the configuration of the firewall easier. Neither NFtables or IPtables have support to be configured on a program basis.

The most basic rules of a firewall are filtering the information found in the network layer of the OSI model (layer 3) such as destination and source IP. Information from the transport layer (layer 4) which contains destination port, source port, sequence numbers and protocol can be used for more detailed filtering.

Filtering can also be done on higher layers, all the way up to the application layer (layer 7) such as HTTP information. The further up in the OSI model the filtering is done on, the higher the load on the firewall itself.

Latency is the time it takes for a packet to get from one point to another and for the response to return to the sending computer. The latency is often measured in thousands of a second but can also be measured in millionths of a second.

Bandwidth is the unit to measure the rate of bits going through the physical cables that connect routers and computers. The bandwidth is usually measured in b/s (bits per second).

Both NFtables and IPtables have support for counting the number of times a certain rule has been matched during evaluation. This counter can count the number of matches and the summarised packet sizes for those packets. This option is mandatory in IPtables and optional in NFtables.

Logging is an optional keyword when configuring rules. When a rule with logging configured is matched, then the information about that packet is logged. This is mainly used for monitoring purposes (Hoffman, Prabhakar & Strooper, 2003).

There are two versions of the network layer protocol IP, IPv4 and IPv6. IPv4 is the protocol which all internet connected devices are using. IPv4 addresses are made up of 32 bits of data

which is represented in a series of four octets which are separated with a "." and each octet is a number between 0 and 255. A typical IPv4 address could look like this "193.11.164.51".

IPv6 is the newer replacement for IPv4 and was introduced to solve the problem that IPv4 is limited to 4 294 967 296 unique addresses. To solve this, IPv6 have 128-bit addresses which result in approximately 340 Undecillion ($340 * 10^{36}$) unique addresses. IPv6 addresses are made up by eight groups of four hexadecimal characters where each group is separated by a ":". A typical IPv6 address could look like this "2001:6b0:1234:24::33". Today, most servers are dual stacked. This means that they have both an IPv4 and IPv6 configured which they can serve on.

QoS (Quality of Service) refers to the ability to control the priority of packages based on the type of traffic to achieve a certain quality of different data flows (Sulaiman, Carrasco & Chester, 2007).

Firewalls are mostly dependent on the physical network interface which it is running on. In addition to the network interface, there is the CPU, RAM and data buses that connects the internal components, that determines the traffic load which it can handle. Filtering on the transport layer is not very demanding, but layers above that require more CPU cycles.

There are several services that are sensitive to high latency. Some but not all are:

- Cloud gaming. A relatively new concept where the rendition of games is not computed on the local machine. The game itself is rendered on the cloud gaming providers hardware, the only thing that is required of the local machine is to handle input, display the rendered frames from the streaming service and play the sounds. This puts the network interface under high network load and latency affects the playing experience very much.
- Stock trading. Many financial customers are seeking zero latency for their trading systems to increase the profitability of trades (Verge, 2013).
- Voice over IP. Traditional IP traffic is serviced on a "best effort" to its destination. This is a problem in IP-telephony and VoIP as they are classified as real-time applications. An acceptable delay in VoIP is below 300 milliseconds and below 150 milliseconds for a good connection (Sulaiman et al., 2007).
- Helland (2001) explains that latency quickly degrades the throughput of data in Storage area networks.

The IPtables framework, Netfilter was released in 1998 and is still the de facto firewall framework to filter network packets in Linux systems, as Kenton (2015) formulates it, it has stood the test of time and is still used in the more recent distributions of Linux (Kenton, 2015).

NFtables is the newer framework that is intended to replace the old Netfilter framework and the user space utility to configure rules. NFtables is a pseudo-state machine that is running in the kernel (Ayuso, 2013). This results in that the rules are converted into a set of predefined instructions which allows for new functionality in NFtables simply by updating the user space

tool and not the entire kernel. Support for NFtables was first introduced in the Linux kernel version 3.13 which was released in January 2014. Hooks used by NFtables was introduced in this version of the kernel. These hooks allow NFtables to fetch the traffic from the Linux kernel in order to make forwarding or filtering decisions based on the pre-determined rules that are configured through the user space tool "nft" (Kearns, 2017). NFtables has slowly been rolling out through Linux repositories since its initial release 2014 (Kenton, 2015). The NFtables framework is less mature compared to the IPtables framework (Kenton, 2015). NFtables is currently lacking many features that IPtables has but it also has many features that IPtables is lacking (netfilter.org, n.d.). NFtables is developed by the same team that developed IPtables. New firewall implementations in Linux are recommended to use NFtables instead of using IPtables (NFtables, 2016).

Rules in both IPtables and NFtables are written in tables. Each table is a collection of chains. IPtables have five chains that are always present in all configurations. These tables cannot be removed as they are mandatory. The pre-defined tables are filter, nat, mangle, raw and security. NFtables have these tables available but they are not active unless explicitly configured.

Chains are a collection of user-defined rules that are evaluated in order. If no rule matches then the default policy is followed. If a packet matches a rule then it can be sent to another user-defined chain for more detailed evaluation. IPtables has several chains per table which cannot be removed while NFtables only have the explicitly configured chains active in each table.

The main reason stated for why NFtables started development is that there are several issues regarding scalability, performance and code maintenance (Nftables, 2016). Both wiki pages Nftables (2016) and Main difference with iptables (2016) lists several differences between IPtables and NFtables that are stated below.

- IPtables have many default chains that are included even though they may never be used. These include "INPUT", "OUTPUT" and "FORWARD" from the filter table, and "POSTROUTING" and "PREROUTING" in the NAT table. NFtables have no chains configured by default (Nftables, 2016).
- IPtables can only have one action configured per rule. This means that if a packet match should be logged and dropped then two rules needs to be added to the chain for the desired result. NFtables can have several actions configured per rule although only one terminating action. Terminating actions are "ACCEPT" and "DROP" (Nftables, 2016).
- IPtables has a separate tool to create datasets, called "ipset". A dataset can be used to create "lists" with addresses or ports that should be evaluated in the same rule. NFtables has this capability included (Nftables, 2016).
- IPtables have several different user space tools to manage rules; the default tool for IPv4 rules is "iptables" while the tool for IPv6 is "ip6tables". All types of traffic can be managed through the NFtables user space tool "nft" (Nftables, 2016).

- IPtables counts chain matches by default and cannot be disabled. NFtables does not count matches by default and if needed it can count matches on single rules instead of entire chains.
- Adding support for new protocols in NFtables does not require a kernel upgrade, a simple update of the user space tool should suffice.
- IPtables have counters enabled by default and cannot be disabled as opposed to NFtables which has it as an optional configuration.

There are multiple frameworks that can be used to act as a firewall but for this study, the focus will be on the IPtables framework and NFtables framework. Other choices may include PF, which is used in BSD (Berkley Software Distribution) based distributions.

iPerf is usually used to speed test end-to-end connections. iPerf is a tool to measure the maximum achievable bandwidth possible on IP networks (iPerf, n.d.). Since iPerf tries to achieve the highest possible bandwidth it was deemed sufficient to saturate the network interfaces of the firewall for this study.

Network load is the state when the firewall is filtering the traffic that is sent between the iPerf client and the iPerf server, both which are explained further later in the paper.

No network load is the state when no traffic is sent between the iPerf client and iPerf server.

3 Problem definition

This section is describing what the problem is, related work that has been done in this area, the research questions that are answered in the results section, the hypothesis, who could have use of this research, the objectives that have been chosen to answer the research question and limitations.

3.1 Aim

The aim is to compare NFtables and IPtables by gathering statistical data to make a comparison on how NFtables compares to IPtables in regard to latency, a comparison in both packet filtering and network address translation. Both NFtables and IPtables are similar enough to be compared since they both are used in GNU Linux but are different in the context that NFtables is a rewrite of IPtables and that NFtables contains several improvements compared to IPtables. The aim of this study is to analyse the latency of IPtables and NFtables handles packet filtering and address translation under load through benchmarks and compare the differences. By performing this research, it can be concluded if NFtables has come far enough in its development to replace IPtables.

3.2 Research questions

The following research questions have been selected to find an answer to the aim that is that this study aims to determine how well both firewalls handles network filtering while under network load.

1. Are there any significant differences in latency between IPtables and NFtables while under network load using forwarding and NAT?
2. Are there any significant differences in latency between IPtables and NFtables during no network load using forwarding and NAT?

The research questions are designed to answer not only how well either firewall works while under no load but also how well they work under heavier load. This is to show a high and a low load since neither firewall will be under a constant high or low load. This is also to show a more realistic scenario since neither firewall will be under constant heavy load in reality.

3.3 Previous work

Jakobson (2014) did a study where three fully featured firewalls, ClearOS, IPfire and Untangle NG Firewall was compared with each other. The firewalls were set up in between two clients on different networks. Throughput, latency and frame loss rate was measured using multiple frame sizes. The frame sizes used were 64, 128, 256, 512, 1024, 1280 and 1518 bytes. The results of their study showed that there was no major difference between the compared firewalls.

Hoffman et al. (2003) did a study where IPtables was tested with a focus on measuring the correctness of the configured rules. By using two computers with three network interfaces each. One of the computers were used to generate traffic and determine if the rule was correctly forwarded to the correct of both interfaces. Their tests were done on 10 Mbps, 100 Mbps and 1000 Mbps. On 10 Mbps, there were no issues. On 100 Mbps, there were no issues

up to 100 rules but after that, the number of correct rules degraded. On 1000 Mbps, it worked reasonably except for smaller frames where the delay was higher.

Sulaiman et al. (2007) measured the latency of VoIP over a 3G (Third Generation) mobile network. Their experiment was performed by testing roaming between several nodes while driving at 50 KM/h. Their results showed that the latency is only slightly increased at low traffic but increases drastically as the traffic volume increases.

3.4 Hypothesis

It is expected that NFtables will perform better than IPtables in all the tests, based on the fact that NFtables are developed by netfilter.org, the same group that developed IPtables.

The hypothesis is that NFtables outperforms IPtables while under load in the tests.

As a null hypothesis, it is expected that IPtables outperforms NFtables while under load in the tests.

3.5 Motivation

Latency was selected as the measuring unit in this research since that is what is first noticeable during normal usage of internet services as well as giving an idea of how resilient a network is against high loads of network traffic.

Zhang et al. (2013) explain that studies from both Google and Amazon show that web users are very sensitive to latency, even a 100ms increase in latency causes measurable revenue losses. All points where the latency can be reduced is important and in the case of Google and Amazon, there are plenty points. All the way from the initial request to access the backend storage to the actual delivery to the end user.

Services that send and receive continuous streams of time-sensitive data to their services. One such service is cloud gaming. An area where a difference of 20 milliseconds can mean the difference between playable and unplayable (Dickson, 2016).

Stock trading companies works is an area where latency is a hot topic and market brokers could pay a lot of money to reduce the latency. There is a goal in market trading which is “ultra-low latency trading”. This is the goal to reduce the latency between stock exchanges to make trades and gain a competitive edge against competitors (New Wave DV, n.d.).

No in-depth study that compares NFtables and IPtables have been found and since latency has so much impact on the user experience of a service it would be interesting to find out if the developers behind IPtables is headed in the right direction.

3.6 Objectives

In order to find a conclusion to the research question, the following objectives are set.

- A small-scale study to gather information needed to execute the experiment and determine a test duration which is suitable is done.

- Perform a quantitative experiment which is intended to answer the selected research question.
- Execute the experiment using the selected method with the firewalls and selected software to generate the traffic.
- Compile the results that are gathered in the experiment. Normalise the data and then draw tables or graphs.
- Analyse the data to answer the research questions and the hypotheses.

3.7 Limitations

No studies are made regarding other firewalls. The primary focus is on NFtables and IPtables. This to make the experiment manageable in the designated time span.

No rules are done on a per program basis. This is because there are not really any programs that require that type of interaction internally on the server and all outgoing traffic will pass through the firewall, which enables the firewall to keep track of all connections. This is because the firewalls are acting statefully. This also allows the firewall to keep track of all connections that originate from inside the network and thus allows the returning traffic to pass back in. All rules used in the firewalls are done on a per port basis.

As NFtables is feature-complete to two-thirds compared to IPtables. Although NFtables has several new features that do not exist in IPtables (Netfilter.org, n.d.). It has been decided that both firewalls will be configured with features that are available in both firewalls.

There are multiple different types of traffic that can be used for the selected experiment. One type if traffic is application layer traffic (layer seven in the OSI model) such as HTTP requests could be used to test how well the firewall handles both valid and invalid application layer traffic. This is not the interest in this article as the focus lies on finding out which firewall that handles the traffic load best on the network layer (layer three in the OSI model).

Another way to test firewalls is to test how well the firewall handles fragmented packages. This is not the aim of this work either so this will be left out.

4 Method

The chosen methods of this experiment are presented in the following sections.

4.1 Methodology choices

A pilot study is performed to gather information and initial data to base the experiment on. The time to run the tests is determined by the tests done in the pilot study.

4.1.1 Pilot study

To gather relevant literature regarding the terms and concepts of the open-source firewalls NFtables and IPtables, the bibliographical databases that have been chosen are the Google Scholar and semanticscholar.org databases. These two databases contain entries to Springer, IEEE, WorldCat, and ACM, all of which contain peer-reviewed articles, whitepapers and books, all of which are accessible through the school library proxy. Threats to validity regarding literature studies and experiments are also covered in this study – see Appendix A for validity threats and how they are handled. Section 6 also covers validity threats.

A smaller experiment is performed as a pilot study to determine the optimal time to run each test. The same test is done at different times to determine how long time each test would need to be run to provide comparable data between both firewalls.

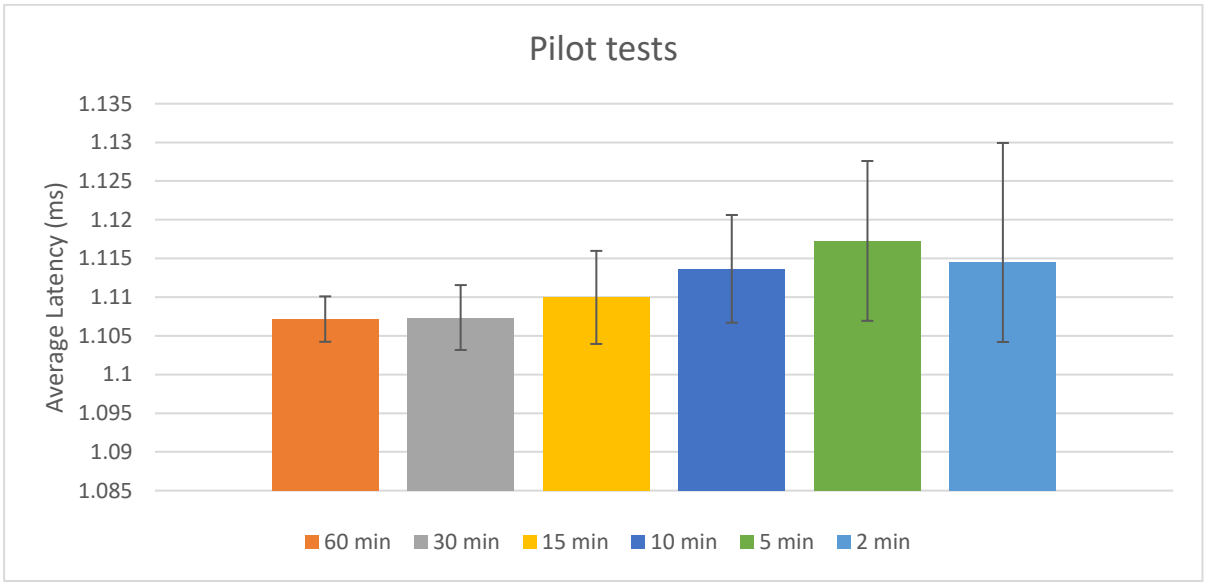

Figure 4.1 Comparison of test times

The results are not getting any more precise after 15 to 30 minutes as shown in figure 4.1. 15 minutes was close to the result that was collected in the 30 and 60-minute tests, and since time is an issue in this work, it is decided that 15 minutes is enough time to provide a good enough result. The error bars are calculated using the standard deviation for each test which is then used to calculate the spread with a 95 % confidence interval.

4.1.2 Experiment

The experiment, as defined by Berndtsson et al. (2008) is performed by creating an experimental environment where the chosen variables are altered and analysed to see if the stated hypothesis can be proven, and if not, disproven.

Each test is done ten times to collect enough data to mitigate potential validity threats. The duration of each test's set is fifteen minutes continuously. Both firewalls are tested in the same way and with the same test times to make the results comparable.

Each firewall is tested with two configurations. One configuration where the traffic generated is just forwarded through the firewall normally and the other configuration where the traffic generated is translated to the firewalls address towards the external network and the traffic from the outside is port forwarded. The selected configurations are tested since they are two of the most common configurations used in firewalls.

4.1.3 Experiment method

The experiment is conducted using the BMWG (Benchmarking Methodology Work Group) of the IETF (Internet Engineering Task Force). The BMWG describes how to benchmark a firewall in the four different areas forwarding, connection, latency and filtering. As explained in RFC3511 by Hickman et al. (2003), this work is focusing on one of the areas, latency.

The experiment is executed on physical computers instead of virtualized computers. Both IPtables and NFtables are run on the same hardware to eliminate the possible validity threat that one set of hardware is more powerful than the other and thus makes it easier to isolate the independent variables.

The firewall is placed between two network segments. Both networks contain two computers each. One of the computers continuously generates network traffic that is sent to its respective hosts on the other side of the firewall. The traffic generated contains very little information and all packets are sent using the same port. This configuration with two networks and a firewall in between is called "two homed" according to Hickman et al. (2003). The physical topology is displayed in figure 4.2. below.

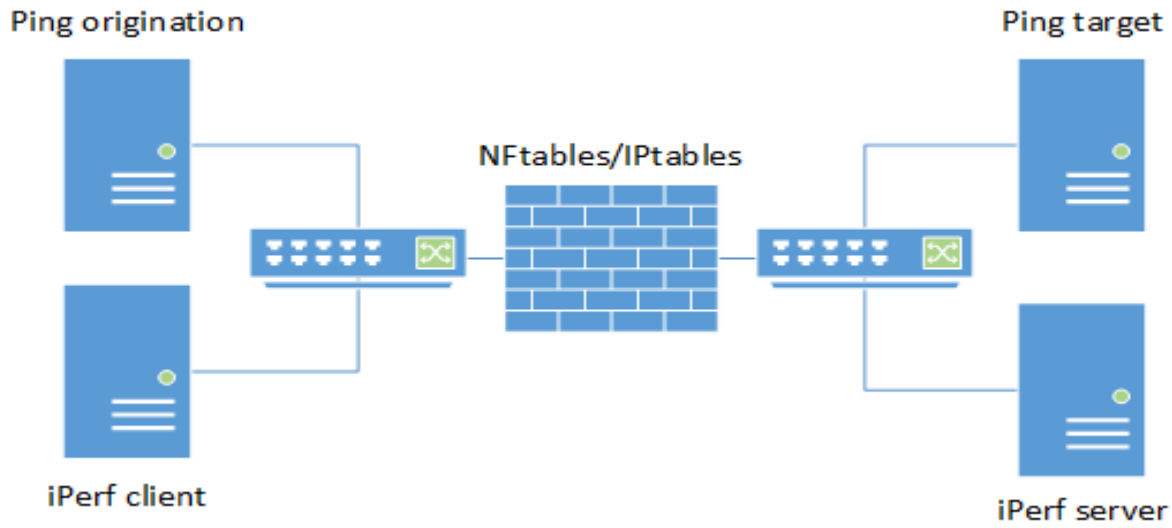

Figure 4.2 Physical topology

The selection of addresses used for the logical topology was chosen out of convenience since 192.168.x.x/16 is commonly used in private LANs (Local Area Network) and the results

should not be affected by the selection of address space. The logical topology is shown in figure 4.3. below.

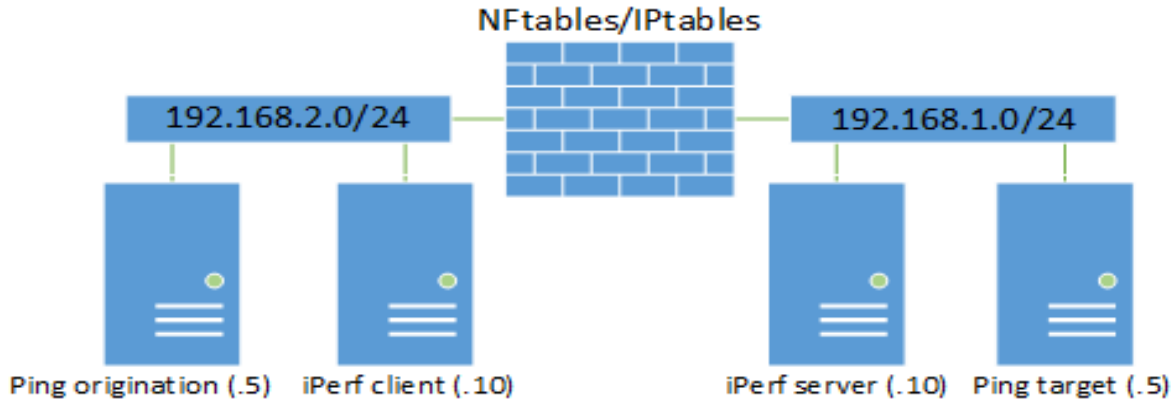

Figure 4.3 Logical Topology

The computers are running Ubuntu Server 16.04 since it is stable and comes installed with a reasonably new kernel. Ubuntu ships with Linux kernel 4.4. NFtables version 0.6.3 is installed from the Ubuntu Zesty repository in order to get the newest version of it. The version of IPtables used is 1.6.0 from the Ubuntu repository.

The installations of Ubuntu include only a minimal installation, skipping all unnecessary packages as the focus will only be on which of the firewalls that perform better. The following packages are installed after installation:

- Vim, to provide a usable text editor.
- NFtables or IPtables. These are the firewalls of interest. These will be mutually exclusively installed on their respective machines to avoid any potential sources of conflict.

NFtables has counters enabled on selected rules to make the comparison between NFtables and IPtables fairer.

The tools selected for the experiment is ping and iPerf. Ping is used because it is widely available and has sufficiently detailed statistics to precisely measure the latency all the way down to one microsecond.

iPerf is used to generate the traffic between both computers. iPerf is available in three versions, iPerf, iPerf2 and iPerf3. The selected version of iPerf is the first version. The choice of the first version is because it has the functionality to send bidirectional traffic, something that iPerf2 and iPerf3 have not yet implemented. Additionally, the goal of this experiment is not to test the performance of iPerf so the selection of an older version should not make any difference on the firewalls tested in this study.

The default values are used for iPerf throughout this experiment with the exception that iPerf is sending traffic in both directions simultaneously and the run time of iPerf. iPerf is used to

saturate the 1 Gigabit network card with TCP traffic using the command "`iperf -s`" on the server-side to prepare the server for incoming connections on port 5001.

The command "`iperf -c 192.168.1.10 -d -t 940`" is then run to start sending traffic from the client-side. The argument "-c" is used to tell the program to run in client mode, "-d" is used to tell the program to run bi-directionally and "-t" is used to tell the program to run for 15 minutes and 40 seconds. The extra 40 seconds are added to give some time to start the data gathering script on the machine used to ping.

4.1.4 Experiment implementation

The experiment execution is performed by setting up default rules that are representative and which tests relevant parts of the chosen firewalls. The selected parts are forwarding and NAT both while under load and while not under load.

All computers used in the tests are on their own physical hardware.

The hardware on the computers running iPerf does not matter much other than the fact that they each have a gigabit Ethernet interface and are sufficient to saturate traffic on the network interface. Both NFtables and IPtables will be running on the same physical machine to provide an even ground between both to be tested on.

Table 1 Hardware components for firewalls

Component	**Description**
CPU	Intel Xeon W3550, 3,07 GHz
RAM	24 GiB, 1333 MHz
Network interface 1	Broadcom BCM5764M Gigabit Ethernet
Network interface 2	Intel 82541PI Gigabit Ethernet Controller

The experiment is done with a total of five physical machines. One machine on the internal network acts as an iPerf server. One machine on the external network acts as an iPerf client. Nothing except for the name is differing between the client and the server since both are sending and receiving traffic at the same time, creating a bi-directional flow through the firewall.

Of the remaining three machines, one is acting as the firewall. The hardware that the firewalls are run on is shown in Table 1 above. The last two are on either side of the firewall pinging each other.

The procedure for each of the tests is that the iPerf machines are running for the selected duration, sending as much traffic as possible to put a network load on the firewall. The ping clients are pinging each other for a duration of fifteen minutes continuously, gathering

readings of the latency. The latency is measured in microseconds in order to be able to read out any difference.

The tests are repeated ten times to provide an average and greater statistical reliability. A reboot is done between each test to make sure that the system is in the same state at the start of each test.

Once the tests have been done ten times for each configuration, the experiment is concluded.

For all tests, the firewalls are configured as border routers with two network interfaces, one for each network.

For the first test, both firewalls are configured with forwarding only. Only the explicitly allowed ports are allowed in and the policy drops everything that is not matched. Appendix B lists the configuration for NFtables and Appendix D lists the equivalent configuration for IPtables.

The firewall is then configured to translate addresses between the networks and only port forward explicitly accepted traffic as well as connections that originate from the network labelled “internal” which is the trusted network. Appendix C lists the configuration for NFtables and Appendix E lists the configuration for IPtables.

Hickman et.al (2003) explains that the tests can be done either using directional or bidirectional unicast traffic. The chosen method in this test is to send bidirectional traffic in order to put additional load on the firewall as opposed to only send traffic in one direction.

Traffic is generated from one source inside the unprotected network to the protected and vice versa. The focus on the traffic is to generate as much as possible to put the firewalls under load.

4.1.5 Dependent variables

The dependent variables are the metrics that are measured and compared based on the changes of the independent variables (Wohlin et al., 2012).

The metrics that are measured are the latencies of both NFtables and IPtables where both firewalls are configured with equivalent rules and are under load.

4.1.6 Independent variables

The independent variables are the changes that are done to the firewall to see how the dependent variables are affected (Wohlin et al., 2012). The usage of NFtables or IPtables are the independent variables in this experiment.

5 Validity

Validity is required in order to arrive at results that are accurate and trustworthy according to Wohlin et al. (2012). There are four different areas of validity which needs to be covered – construct validity, internal validity, external validity and reliability. A list of identified validity threats to this experiment, how they could affect the experiment and what steps have been taken to avoid them are listed in Appendix A.

Construct validity means that all stated terms need to be defined to correctly represent the experiment. All terms used in the paper is defined to give an overall understanding of the experiment conducted.

Secondly, there is internal validity, where there are internal "hidden" factors that could affect the results. In this experiment one such could be if the experiment is conducted on a network shared by other users, then that could affect the collected results. This has been handled by running the tests on an isolated network where no other traffic passes through. More on how validity threats are handled this work is shown in Appendix A.

The scale of this experiment has been purposefully designed small to avoid these "hidden" factors that could cause issues.

Thirdly there is external validity if the used tools can be used in the same way to test other firewalls just as in this study. All the tools used to gather the data have been explained and additional arguments have been supplied.

Fourthly there is reliability; the experiment should not be dependent on one single person in order to replicate the results of the experiment. All configurations and topology have been included in this report in Appendix B-E.

5.1 Conclusion validity

According to Wohlin et al. (2012) conclusion validity is a threat against if a correct conclusion can be drawn from the results. Threats against the conclusion are low statistical power, violated assumptions of statistical tests, and fishing and error rate.

If insufficient data has been gathered during the testing phase, then this could result in low statistical power. Each test has been repeated ten times with the same conditions and reboots in between to counter this and to gather enough reliable data to draw an accurate conclusion.

Violated assumptions of statistical tests is a conclusion validity threat that could be the result of using a test with low robustness on the gathered data.

If the author of this work has predetermined thoughts of which firewalls that are best then this could result in fishing and error rate. By being aware of this threat, it can be avoided more easily. Additionally, this can be avoided by not cherry-picking data and by presenting all the data that has been gathered in the experiment.

5.2 Ethical aspects

This experiment does not involve any human subjects or personal information. This experiment is only done with computers and no identified ethical aspects have been taken into consideration because of that.

6 Results

The experiment has resulted in the following results. The significance of the results has been determined using the Mann-Whitney U-test with an alpha value of 0,05 (95 % confidence). First, a graph showing all tests next to each other. After that, the method used to determine the significance, namely the Mann-Whitney U-test is described. Then the results while the firewalls are not under load and lastly the results from when the firewalls are under load.

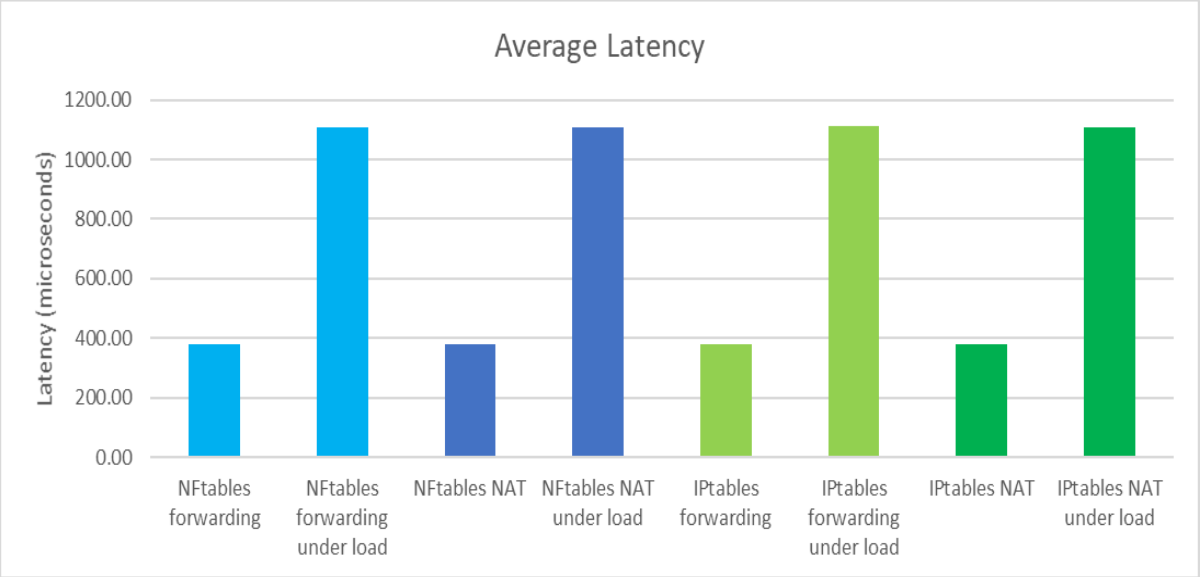

Figure 6.1 Results of the measurements

Figure 6.1 shows that there is a lot of difference between running iPerf during testing and not. The difference in latency between under load is that the latency is consistently almost three times higher in all the tested scenarios.

6.1 Significance

The Mann-Whitney U-test was used to determine if the values are significant compared to each other.

Mann-Whitney U-test is a nonparametric statistical test to determine if there is a significant difference between two datasets. To test two datasets, each value within that dataset needs to be ranked. The ranking determines where in a list the value would be if the value was in an ordered list with both datasets in it. The ranking is then summarised for each individual dataset to determine a rank sum for both sets.

A Z-value is calculated based on the rankings from the results. If the calculated Z-score is greater than the normal distribution then the difference between the data sets is statistically significant with 95 % certainty. How the U-test is calculated is further explained by Siegel and Castellan (1988).

6.2 No network load

All the scenarios tested in the experiment were first tested while not being under load to compare how the latency differs between the firewalls while not being under load. These results are also needed to be able to answer the second research question: “Are there any

significant differences in latency between IPtables and NFtables during no network load using forwarding and NAT?".

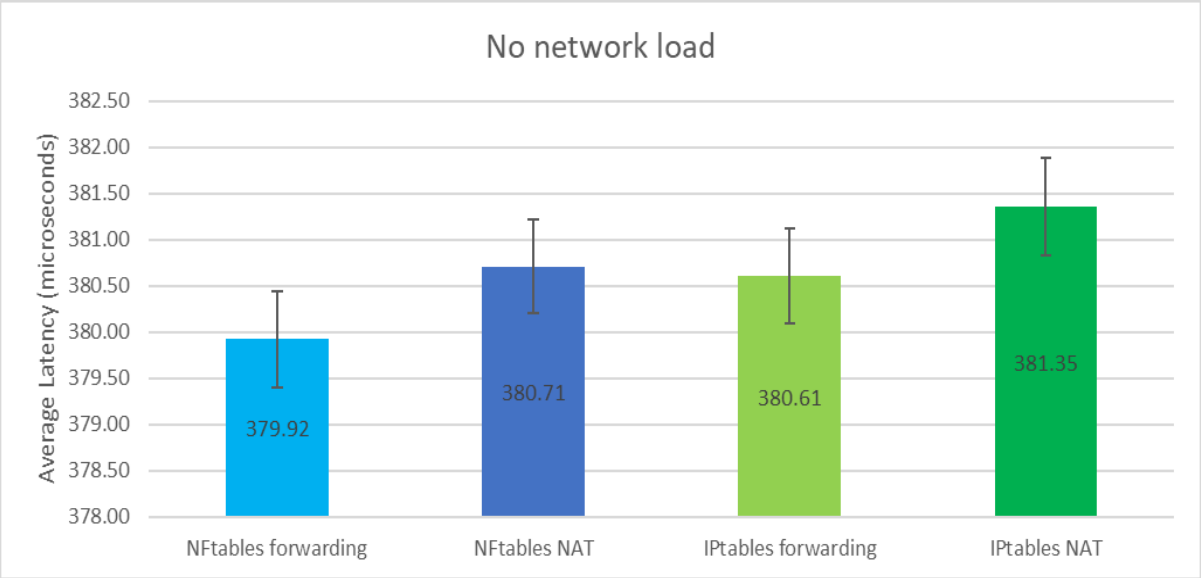

Figure 6.2 Average latency under no network load

Iptables forwarding under load vs. Nftables forwarding under load

From left to right, NFtables forwarding under no load vs. NFtables

Figure 6.2 above shows that the average latency for NFtables forwarding while not being under load is 379.92 microseconds. The median for NFtables forwarding with no load is 370.00 microseconds and the standard deviation is 25.19 microseconds.

Figure 6.2 above also shows that the average latency for NFtables NAT while not being under load is 380.71 microseconds. The median for NFtables forwarding with no load is 371.00 microseconds and the standard deviation is 24.55 microseconds.

The average latency for IPtables forwarding while not being under load is 380.61 microseconds. The median for NFtables forwarding with no load is 371.00 microseconds and the standard deviation is 24.90 microseconds.

The average latency for IPtables NAT while not being under load is 381.35 microseconds. The median for NFtables forwarding with no load is 372.00 microseconds and the standard deviation is 25.35 microseconds.

The error bars in figure 6.2 shows the interval where any given measurement will be within by 95% probability.

Table 2 Comparison of Mann-Whitney U-tests on results while under no load with 95% significance

Comparisons	**One-tailed**	**Two-tailed**
NFtables forwarding under no load vs. IPtables forwarding under no load	Yes	Yes

NFtables translating addresses under no load vs. IPtables translating addresses under no load	Yes	Yes

Table 2 above shows that the results of NFtables and IPtables are significant compared with one another both for a one-tailed and two-tailed U-test. This shows that significant conclusions can be drawn from these results.

6.3 Under network load

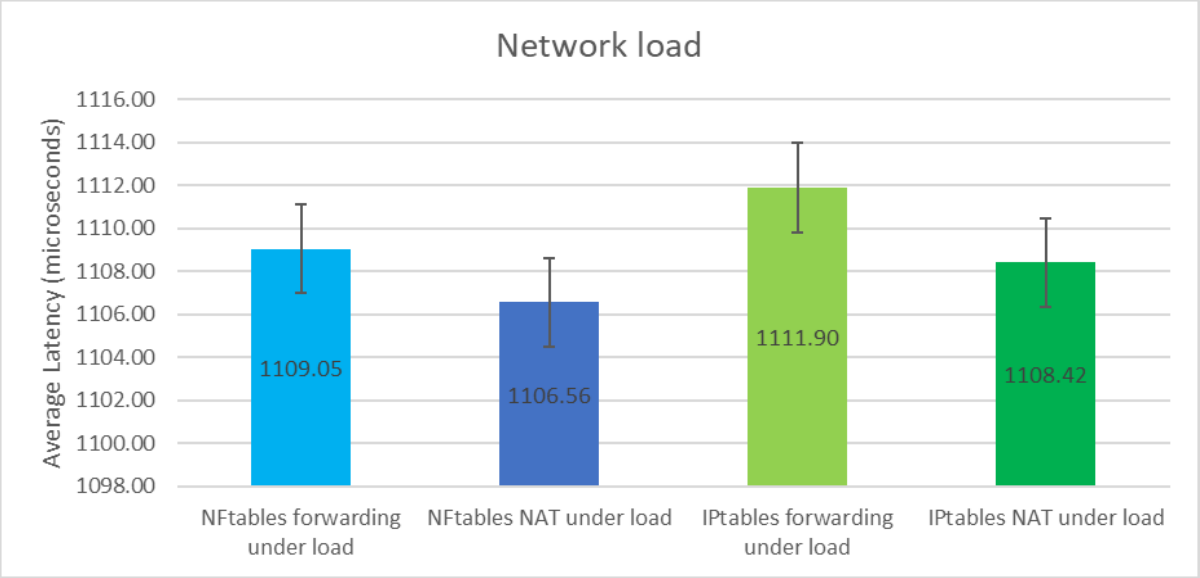

Figure 6.3 Average latency under network load

Figure 6.3 above shows that the average latency for NFtables forwarding while being under load is 1109.05 microseconds. The median for NFtables forwarding with no load is 1110.00 microseconds and the standard deviation is 99.89 microseconds.

Figure 6.3 above also shows that the average Latency for NFtables NAT while being under load is 1106.56 microseconds. The median for NFtables forwarding with no load is 1105.00 microseconds and the standard deviation is 100.72 microseconds.

The average Latency for IPtables forwarding while not being under load is 1111.90 microseconds. The median for NFtables forwarding with no load is 1107.00 microseconds and the standard deviation is 100.42 microseconds.

The average Latency for IPtables NAT while not being under load is 1108.42 microseconds. The median for NFtables forwarding with no load is 1105.00 microseconds and the standard deviation is 100.16 microseconds.

The error bars in figure 6.3 shows the interval where any given measurement will be within by 95% probability.

Table 3 Comparison of Mann-Whitney U-tests on results while under network load with 95% significance

Comparisons	**One-tailed significance**	**Two-tailed significance**

NFtables forwarding under load vs. IPtables forwarding under load	Yes	No
NFtables translating addresses under load vs. IPtables translating addresses under load	Yes	Yes

Table 3 above shows that the results of NFtables and IPtables are significant compared with one another in all cases but the two-tailed test where NFtables and IPtables forwarding are compared using the Mann-Whitney U-test. This shows that significant conclusions can be drawn in most cases.

7 Conclusion

Looking back to the research questions that are stated in section 3.2 which are the following:

1. Are there any significant differences in latency between IPtables and NFtables while under load using forwarding and NAT?

It can be concluded that in terms of the difference in latency between NFtables and IPtables, there is some difference between them while under load. NFtables has just slightly lower latency pinging while under load in both cases with regular forwarding and when performing address translation between the networks. It is possible to determine that the results are significant by comparing the results using the Mann-Whitney U-test that NFtables is the faster firewall while under load.

2. Are there any significant differences in latency between IPtables and NFtables during normal load using forwarding and NAT?

Based on the results, there is not much difference between any of the configurations. The only thing that can be concluded is that NFtables is the better firewall in regard to latency while not under load, also by comparing the results using the Mann-Whitney U-test. It is also worth mentioning that the difference between the firewalls while not under load is lower than 1 microsecond.

8 Discussion

Some things that would have been made differently from the currently done study is presented in this section.

It was a bit surprising that the latency of IPtables and NFtables was so close. It was expected that the difference would be greater between IPtables and NFtables.

If time would not be an issue then other data that could be gathered during this experiment would be collected and compared in addition to the latency, things that Jakobson (2014) collected in their experiment such as frame loss and throughput with multiple frame sizes.

If the hardware would not be a limiting factor, then this work would have been done using higher data rates, preferably up to 10 Gbps. This was not possible since 10 Gigabit network interfaces are quite expensive and that money was not available.

Netfilter which is used by IPtables can be configured slightly differently in regards to the connection state. The older “state” module could have been used instead of the newer conntrack module. This could have resulted in worse results for IPtables. How much it would have differed is a question for a different study. The configuration for both IPtables and NFtables are otherwise not very flexible to achieve the same result as the configuration used in this work.

The only information that could be used to identify a group of individuals is the firewalls themselves, this has been disregarded since the developers behind both firewalls are the same. No people or information that could be used to identify individuals were involved in this work. All the results have been gathered using unmodified open source software which is publicly available to everyone. All results are presented unmodified to show correct information and to avoid breaking any research ethics.

9 Future Research

This section is presenting examples of what could be done in the future to improve this research.

The experiment could be redone using 10 Gigabit network interfaces instead of the 1 Gigabit interfaces that were used in this research. A bigger impact should be noticed by running this test with a higher network load on the firewalls. The results of this study could be concluded with higher certainty by running this experiment with higher throughput.

This test only compares the difference between IPtables and NFtables. The experiment could be continued to compare either NFtables or IPtables with other firewalls such as BSDs Packet Filter. Since Packet Filter is running on a different operating system and uses a different framework to filter the packages there should be possible to notice a difference compared to IPtables and NFtables.

This test could be redone collecting frame loss and throughput at the same time instead of just collecting the latency.

A different approach could be done by using network interfaces made for higher speed to see is the traffic load affects the CPU load or the RAM usage on the hardware that the firewalls are running on.

References

Ayuso, P. N. (2013). *Nftables strikes back* [presentation]. Retrieved June 15, 2017, from https://workshop.netfilter.org/2013/wiki/images/e/ee/Nftables-osd-2013-developer.pdf

Berndtsson, M., Hansson, J., Olsson, B. & Lundell, B. (2008). Thesis Projects – A Guide for Students in Computer Science and Information Systems. Second Edition. Springer-Verlag. London.

Cisco (2017). *Configuring NAT for IP Address Conservation*. Retrieved June 17, 2017, from https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_nat/configuration/xe-16/nat-xe-16-book/iadnat-addr-consv.html

Dickson, C. (2016). *The technology behind a low latency cloud gaming service.* Retrieved August 7, 2017, from https://blog.parsec.tv/description-of-parsec-technology-b2738dcc3842

Distrowatch (2017). *distrowatch.com/.* Retrieved: May 21, 2017, from https://distrowatch.com/

Helland, A. (2001). *Bandwidth vs. latency in SAN extensions*. Retrieved: June 14, 2017, from http://www.infostor.com/index/articles/display/129149/articles/infostor/volume-5/issue-12/features/bandwidth-vs-latency-in-san-extensions.html

Hickman, B., Newman, D., Tadjudin, S. & Martin, T. (2003). *Benchmarking Methodology for Firewall Performance*. Retrieved May 7, 2017, from https://tools.ietf.org/html/rfc3511

Hoffman, D., Prabhakar, D. & Strooper, P. (2003). *Testing iptables*. CASCON '03 Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research, p. 80-91. Retrieved June 13, 2017, from https://dl.acm.org/citation.cfm?id=961337&dl=ACM&coll=DL&CFID=948080881&CFTOKEN=38669906

Ioannidis, S., Keromytis, A., Bellovin, S. & Smith. J. (2000). *Implementing a distributed firewall.* Proceedings of the 7th ACM conference on Computer and communications security, p. 190-199. doi:10.1145/352600.353052

Jakobson, F. (2014). *Open source routing software: A comparative study of open source software routers*. (Bachelors Degree Project in Computer Science, School of Informatics, University of Skövde, Skövde). Retrieved May 20, 2017, from http://his.diva-portal.org/smash/get/diva2:726337/FULLTEXT01.pdf

Kearns, P., & Marmorstein, R.M. (2005). *A Tool for Automated iptables Firewall Analysis*. Retrieved April 25, 2017, from

http://static.usenix.org/event/usenix05/tech/freenix/full_papers/marmorstein/marmorstein_html/

Kenton, A. (2015). *Nftables as a Second Language*. SANS Institute InfoSec Reading Room. Retrieved May 9, 2017, from https://www.sans.org/reading-room/whitepapers/firewalls/nftables-second-language-35937

Main differences with iptables (2016, 13 July). In wiki.nftables.org. Retrieved June 5, 2017, from https://wiki.nftables.org/wiki-nftables/index.php/Main_differences_with_iptables

Microsoft (n.d.). *What is a firewall?* Retrieved August 7, 2017, from https://www.microsoft.com/en-us/safety/pc-security/firewalls-whatis.aspx

Misherghi, G., Yuan, L., Su, Z., Chuah, C. & Chen, H. (2008). *A general framework for benchmarking firewall optimization techniques.* IEEE Transactions on Network and Service Management (volume 5, issue 4), p. 227-238. doi:10.1109/TNSM.2009.041104

Netfilter.org. (n.d.). *The netfilter.org "nftables" project*. Retrieved May 9, 2017, from https://netfilter.org/projects/nftables/index.html

New Wave DV (n.d.). *Ultra-low Latency Trading.* Retrieved August 7, 2017, from http://newwavedv.com/markets/financial/ultra-low-latency-trading/

Nftables (2016, December 24). In wiki.debian.org. Retrieved June 4, 2017, from https://wiki.debian.org/nftables

Oxford dictionary. (n.d.). firewall. Retrieved June 4, 2017, from https://en.oxforddictionaries.com/definition/firewall

Siegel, S., & Castellan, N. J. (1988). *Nonparametric statistics for the behavioral sciences (2nd ed.).* New York: McGraw-Hill.

Suehring, S. (2014). *Linux Firewalls Enhancing Security with nftables and Beyond (4th ed.)*. Retrieved April 19, 2017, from https://pdfs.semanticscholar.org/7bb5/963ecb1325ca0454d4c59dfd59ab09e19161.pdf

Sulaiman, N., Carrasco, R. & Chester, G. (2007). *Impact of Traffic on Multi Service IP Based Applications*. Second International Conference on Innovative Computing, Information and Control, 2007. ICICIC '07. doi: 10.1109/ICICIC.2007.344

iPerf. (n.d.). *What is iPerf / iPerf3 ?*. Retrieved June 13, 2017, from https://iperf.fr/

Verge, J. (2013). *Wall Street Going Wireless in Bid for Ultra-Low Latency*. Retrieved June 14, 2017, from http://www.datacenterknowledge.com/archives/2013/05/07/trend-watch-wireless-for-financials-from-325-hudson-street-hudson-fiber-networks/

What is nftables? (2017, April 8). In wiki.nftables.org. Retrieved May 9, 2017, from https://wiki.nftables.org/wiki-nftables/index.php/What_is_nftables%3F

Wilkins, S. (2013). *Stateful firewall fundamentals: A better, easier, more secure firewall.* Accessed: May 22, 2017, from https://www.pluralsight.com/blog/it-ops/stateful-firewall-fundamentals

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. & Wesslén, A. (2012). *Experimentation in Software Engineering.* Berlin. Springer Science and Business Media.

Zhang, Y., Power, R., Zhou, S., Sovran, Y., Aguilera, M. K. & Li, J. (2013). *Transaction chains: achieving serializability with low latency in geo-distributed storage systems.* Proceedings of the Twenty-Fourth ACM Symposium / Operating Systems Principles (SOSP '13). ACM, 2 Penn Plaza, Suite 701, New York, NY 10121-0701, USA. doi: 10.1145/2517349.2522729

Appendix A Validity Threats

Validity threat	Potential issue	Actions taken to avoid
The machines running the firewalls are not identical.	Can give skewed results that are not comparable.	Both firewalls will be run on the same physical machine.
The machines are not identical.	Could result in different traffic loads during the experiment.	All machines used in the experiment have sufficient hardware to saturate the 1 Gigabit network interface.
The systems are mixed steady-state and startup.	If any of the firewalls are caching results and uses this in filtering then the first test will always be worse than the others.	The machines will be rebooted after every test to measure the performance impact on a startup system instead of a steady state system.
The experiment was done on a shared network	If the amount of traffic that is sent on the network is not maintained then the values from the tests could fluctuate a lot.	All tests were done on two isolated networks, so all traffic that is sent is generated by the machines in the network.
Services running in the background which causes network load.	Could put the firewalls under unnecessary load.	None, all installations of Ubuntu in the experiment have a minimal number of packages installed to avoid this. If there is any unnecessary traffic then it is deemed negligible.
There are multiple ways to configure the firewalls	Could affect the performance of either firewall if one way is better than the other	Both firewalls has been configured in similar ways to minimise the risk of using unoptimized arguments

Appendix B NFtables forwarding rules

```
#!/usr/sbin/nft -f
flush ruleset
table ip firewall {
           chain input {
                      type filter hook input priority 0; policy drop;
                      ct state established,related accept
                      icmp type { echo-reply, echo-request} accept
                      iif "enp1s0" jump input-internal
                      iif "ens5" jump input-external
                      iif "lo" accept
           }
           chain input-internal {
                      accept
           }
           chain input-external {
                      tcp dport 22 accept
           }
           chain forward {
                      type filter hook forward priority 0; policy drop;
                      iif "enp1s0" jump forward-internal
                      iif "ens5" jump forward-external
           }
           chain forward-internal {
                      accept
           }
           chain forward-external {
                      counter
                      tcp dport 5001 counter accept
                      udp dport 5001 counter accept
                      icmp type { echo-request, echo-reply} accept
           }
}
```

Appendix C NFtables NAT rules

```
#!/usr/sbin/nft -f
flush ruleset
table ip firewall {
           chain input {
                      type filter hook input priority 0; policy drop;
                      ct state established,related accept
                      icmp type { echo-reply, echo-request} accept
                      iif "enp1s0" jump input-internal
                      iif "ens5" jump input-external
                      iif "lo" accept
           }
           chain input-internal {
                      accept
           }
           chain input-external {
                      tcp dport 22 accept
           }
           chain forward {
                      type filter hook forward priority 0; policy drop;
                      iif "enp1s0" jump forward-internal
                      iif "ens5" jump forward-external
           }
           chain forward-internal {
                      accept
           }
           chain forward-external {
                      counter
                      tcp dport 5001 counter accept
                      udp dport 5001 counter accept
                      icmp type { echo-request, echo-reply} accept
           }
           chain prerouting {
                      type nat hook prerouting priority 0; policy accept;
                      iif ens5 tcp dport { 5001} counter dnat 192.168.1.10
                      iif ens5 udp dport { 5001} counter dnat 192.168.1.10
           }
           chain postrouting {
                      type nat hook postrouting priority 100; policy \
accept;
                      ip saddr 192.168.1.10/32 oif ens5 snat 192.168.2.1
           }
}
```

Appendix D IPtables forwarding rules

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
:forward-external - [0:0]
:forward-internal - [0:0]
:input-external - [0:0]
:input-internal - [0:0]
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i enp1s0 -j input-internal
-A INPUT -i ens5 -j input-external
-A INPUT -i lo -j ACCEPT
-A FORWARD -i enp1s0 -j forward-internal
-A FORWARD -i ens5 -j forward-external
-A forward-external -p tcp -m tcp --dport 5001 -j ACCEPT
-A forward-external -p udp -m udp --dport 5001 -j ACCEPT
-A forward-external -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A forward-external -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A forward-internal -j ACCEPT
-A input-external -p tcp -m tcp --dport 22 -j ACCEPT
-A input-internal -j ACCEPT
COMMIT
```

Appendix E IPtables NAT rules

```
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -i ens5 -p tcp -m tcp --dport 5001 -j DNAT \
--to-destination 192.168.1.10
-A PREROUTING -i ens5 -p udp -m udp --dport 5001 -j DNAT \
--to-destination 192.168.1.10
-A POSTROUTING -s 192.168.1.10/32 -o ens5 -j SNAT --to-source 192.168.2.1
COMMIT
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
:forward-external - [0:0]
:forward-internal - [0:0]
:input-external - [0:0]
:input-internal - [0:0]
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i enp1s0 -j input-internal
-A INPUT -i ens5 -j input-external
-A INPUT -i lo -j ACCEPT
-A FORWARD -i enp1s0 -j forward-internal
-A FORWARD -i ens5 -j forward-external
-A forward-external -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A forward-external -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A forward-internal -j ACCEPT
-A input-external -p tcp -m tcp --dport 22 -j ACCEPT
-A input-internal -j ACCEPT
COMMIT
```

Appendix F Data gathering script

```
#!/bin/bash
ping 192.168.1.5 -w 900 | while read ping_output; do echo "$(date): \
$ping_output"; done
```