University of Tartu
Faculty of Science and Techonology
Institute of Technology

Anders Martoja

FAULT TOLERANT NETWORKING USING LINUX BASED SYSTEMS AND OBSOLETE HARDWARE

Bachelor's Thesis (12 ECTS)
Computer Engineering

Supervisor: Artjom Lind,
Researcher,
Distributed Systems Group,
Institute of Computer Science

Tartu 2016

Tõrkeid taluv võrk, mis baseerub Linux süsteemil ja kasutab vana riistvara

Lühikokkuvõte: Arvuti võrk on ala, mida on kõige lihtsam mõista, kui panna see onlain kaubamaja konteksti. Arvuti võrk opereerib normaalselt siis, kui klient saab poele internetist ligi. See tähendab, et müügi sait saab raha teenida oma kleintidelt. Juhul, kui nüüd võrk peaks katki minema, siis ei saa enam klient poele ligi ega ka see onlain kaubamaja kasumit teenida.

Tõrkeid taluv võrk on võrgunduse üks alamosa, mis üritab vältida eelpool mainitus stsenaariumi ehk online kaubamaja oleks alati kättesaadav kleindile. Tõrkeid taluv võrk sisaldab endas tagavara kommutaatoreid ja ruutereid, mida kasutatakse koos spetsiaalse tarkvaraga, et luua võrk, mis kannatab eelpool mainitud seadmete tõrkeid.

Käesoleva bakalaureuse töö põhieesmärk on pakkuda dokumenteeritud lahenuds, mis oleks odav, lihtne üles seada ja pakuks head käitset võrgu tõrget vastu.

Üldjuhul on võimalik luua võrk, mis kannatab tõrkeid oma töös, aga samal ajal võib selle võrgul olla probleeme suure läbilaske võimega. Suur läbilaske võimekus pole selle töö eesmärk ja käesolevas tööks oleks selle probleemi allikaks vanad ja vähem võimekad võrgu komponendid.

Suures kontekstis, idee luua tõrkeid taluv võrk, pole uus. Leidub küllaldaselt võrgu lahendusi, mis pakuvad tõrke kindlust, aga autori arvates on need lahendused kas liiga kallid või on viletsalt dokumenteeritud.

Bakalaureuse töö pakub võrgulahendust, mis on odav ja lihtne üles seada, seega igaüks, kellel on juurdepääs vanale, mahakantud riistvarale saaks luua, kasutades selles töös esitatud dokumentatsiooniga. Autori arvates on selle töö suurim panus just dokumentatsioon, mis üritab võimalikult detailselt lugejat juhtida võrgu ülesseadmisel koos viidetega allikatele, kust informatsioon pärit on.

Võtmesõnad: tõrkeid taluv võrk, võrk, ruuter, Linux, vana riistvara

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia

Fault tolerant networking using Linux based systems and obsolete hardware

Abstract: Networking at enterprise level is fast, reliable,fault-tolerant, expensive and usually includes a vendor lock-in. This thesis tries to take the good qualities of the enterprise level networking and remove the negative side effects. One possible way to solve the problem is to create a networking solution that uses obsolete hardware and free software to neglect the negative properties of enterprise networking. This kind of solution can be achieved without any need for more system administration knowledge or know-how.

Moreover, the thesis conducts an analysis of the previous work done by other people on fault tolerance in networking and offers the author's own solution implementation. The main result of this thesis is a low-cost solution to Internet service provider and router failover. The tests that were carried out show that it is possible to create a network that, even in the case of ISP and/or router failure, can continue functioning at full speed without significant delays and without disrupting the presently occurring data transmission. The thesis's work uses computer hardware that has been announced old by its previous owners and software that is open source and readily deployable with a little skill.

From a theoretical point of view it should be acknowledged that the author understands that software routing is not as good as hardware routing, however the overall idea was to offer a solution which could theoretically perform as similar as possible to the hardware routing.

Keywords: fault tolerant networking, network, router, Linux, obsolete hardware

CERCS: T120 Systems engineering, computer technology

Contents

List of Figures

List of Listings

Acknowledgements

I would like to thank Artjom Lind for giving me an interesting topic for writing my thesis on while keeping in mind wishes for doing the practical work. In my mind my thesis would not have been possible if I had not received help from Kersti Taurus from the Faculty of Mathematics and Computer Science and Urmas Lett from EENet, both of whom generously gave me old computers which they did not need anymore. I would also like to thank Michael Brown and Christian Nilsson for helping me with my issues regarding iPXE and all of its wonders - both were readily available at iPXE IRC chat room.

1 Introduction

The idea is to create a computer network which can handle errors. Routers are integral parts of a modern networks along with switches. When a router and/or switch and/or Internet Service Provider (ISP) fails the user usually loses internet connection. The goal of this thesis was to create a computer network which could handle router and ISP fail overs so that the end user would not have connectivity issues.

The author of this thesis created a solution that utilizes old computer hardware along with dated networking gear. When the failure happens the network must retain connectivity and the ongoing processes can not be disturbed by the failure of network components. The solution offers protection against ISP and/or router failure. While fault tolerance is an prerequisite, high throughput is not, because the networking gear that is used is not capable of handling high bandwidth connections.

The thesis offers a solution to situations where fault tolerance is required but a lot of resources are not available. Also the solution offers a fine grained control over the network, since all of the parts making up the network can be chosen and replace if necessary by the administrator. For example if one part of the network is not performing as well as needed it can be changed easily, whether it is the software or the hardware — there is no vendor lock-in.

The literature review presented in this thesis can be categorized into two main groups. The first one offers solutions in theory that can tolerate failures to various degrees. Meanwhile, the second group offers solutions that increase fault tolerance in networks and uses more hardware. Moreover, there is a lack of documentation on how those networks were setup.

Author of the thesis learned to use all of the components making up the offered fault tolerant network and configured them in a way that produced said solution. The author also extensively documented how to get the work done, so that it can be replicated by other parties , with references to the source of the information.

Chapter 3 contains information about fault tolerance in general, different kinds of protocols used in fault tolerance, solutions that exist for the problem at hand and related works. In addition the the documentation for the created solution, which is written in great details. The 4th chapter talks about tests that network was put through and the outputs that were received. Chapter 5 is the conclusion of the thesis with a paragraph about future work.

2 Overview

The following chapter talks about fault tolerance, what kind of protocols allow for fault tolerant setup, what kind of network devices are used in redundancy. Also what are the requirements for fault tolerance and what kind of enterprise solutions are used by companies. Related works are included as well.

2.1 Introduction to fault-tolerant networking

Fault tolerance is a subject when put to todays business perspective is the easiest to understand. Considering online businesses, the service availability is the top priority. These businesses heavily rely on their availability. In case there is no network redundancy — there is a single copy of every computer component on what the server runs. In case of a hardware failure, at any level, the online business will not be accessible to the clients, which means a financial loss for the business. The spatial redundancy is missing in this case. The worst case scenario would not happen if there was hardware redundancies set up. Simple redundant router could easily take over the traffic if the main router fails. The redundant router does not have to be as good as the main router, it just has to keep the site available while the main one is fixed. Redundancy creates a fallback option which in online businesses is vital. [1]

2.2 Fault-tolerant network composition

The following subsection describes the protocols that are vital for achieving fault tolerance.

2.2.1 Protocols

TCP

Transmission Control Protocol (TCP) is a protocol which complements the Internet Protocol (IP). TCP is extensively utilized by many applications on the internet. Major applications like World Wide Web, mail, ftp, media streaming, ssh, heavily rely on it. TCP is focused on reliable, ordered and errorless delivery. Since TCP is focused on precise delivery of data, it may produce delays in data trasnmission — not focused on in-time delivery. TCP detects problems that might occur on lower levels of the Internet model and might request retransmission of data that is lost in transmission, rearrange data that is out of order, help minimize congestion problems. TCP hides all the aspects of lower layers from the upper layers. TCP focuses on reliable data transmission and it uses positive acknowledgement with retransmission to assure the quality of reliable packet delivery. The sending

party also keeps a record of packets that have been sent. In addition there is also a timer that checks if acknowledgement has been received before the timer runs out — latter is needed if the data should get lost, automatic retransmission is activated. [2]

IP

Internet protocol is in charge of addressing hosts and making sure that the datagram (encapsulated packet) reaches from the source host to the destination host. Datagram consists of an IP header and a payload. The header contains the source and destination IP addresses with some other necessary information like: header checksum, Time To Live, Protocol, Total Length, Flags, etc. The header is used for routing the packet on its journey. IP addressing includes giving IP addresses to hosts and handing out the necessary information to the hosts.[3, 4]

Virtual Router Redundancy Protocol

Virtual Router Redundancy Protocol (VRRP) is an open standard networking protocol. VRRP offers available routers to hosts that are connected to the subnetwork. VRRP automatically offers default gateways in subnetworks to increase fault tolerance and reliability. VRRP creates abstract representations of virtual routers, which act in master slave configuration. The default gateway is assigned to a virtual router, which can "move" from one physical router to another. This is achieved through voting of master and slave routers. When the master router (physical router) can not service the data trasnmission anymore (fails), the slave router (physical router) takes over the tasks of the master. VRRP can be used with IPv4 and IPv6 addresses. Routers in VRRP configuration have set priorites ranging from 1-255. When the routers started, the router with a higher priority wins the election and becomes the master router. Physical routers that are connected via VRRP, communicate with each other using a multicast IP (224.0.0.18) — heartbeat mechanism. VRRP is based on Hot Standby Router Protocol (HSRP), which is Cisco's proprietary protocol. [5]

Common Address Redundancy Protocol

Common Address Redundancy Protocol (CARP) is an free standard networking protocol. CARP allows routers in the same subnetwork to use a set of same predefined IP addresses — router redundancy. CARP is similart to VRRP and HSRP in its functionality. The redundancy the CARP offers is called "group redundancy" — group is divided to Master and Slaves. If the Master router fails the Slave router takes over the data transmission. To make sure that the Slave takes over at the right moment, the Master fails synchronization of state must

be supported. From a machine's perspective being behind the default gateway nothing changes if the Master should fail. CARP and VRRP can not exist in the same network simultaneously due to their MAC address conflicts if the VRRP group ID and CARP host ID's match. [6]

Hot Standby Router Protocol

Hot Standy Router Protocol (HSRP) is Cisco's proprietary networking protocol. It offers default gateway redundancy. When the routers are activated they notify each other of their priorities and their status. Again, the router with the highest priority wins, when deciding which router should be the "Master", that handles all the traffic. In case the "Master" fail the router with the next highest priority takes over. Besides, the same rule applies to the machines outside of the network nothing changes if one of the router fails and another takes over. Similar to VRRP and CARP. [7]

Extreme Standby Router Protocol

Extreme Standby Router Protocol (ESRP) is Extreme Networks's proprietary networking protocol, which offers quick failover and layer 2 protection. [8]

Gateway Load Balancing Protocol

Gateway Load Balancing Protocol (GLBP) is a Cisco proprietary networking protocol. GLBP provides automatic router backup, much like VRRP,CARP and HSRP, but with a added packet load sharing between a group of redundant routers. All routers which form the redundant configuration share one IP while maintaining their individual MAC addresses. When the Active Virtual Gateway (AVG) fails, the redundant routers become active and take over the tasks that the AVG was handling. GLBP's additional feature is providing load balancing over multiple routers. Therefore, sharing the load more equally among available resources. [9, 10]

Routed Split multi-link trunking

Router Split multi-link turnking (R-SMLT) is a proprietary developed by Nortel. It provides high-speed failover for networks designed with SMLT and Distributed-SMLT topologies. RSMLT forwards packets in routers and in case of a failure switches over to another router.RSMLT works with Avaya's Ethernet routing switches (few models). [11]

2.2.2 Switches

Switches are devices that connect different computers on the same network. Switches use packet switching to analyze, receive and forward packets. Switches, unlike hubs, can send packets to the specified targets, there may be one or more recipients. Switches have multiple ports, which are all internally connected, but the processor makes sure that only the right port gets the information that was meant for it. Ethernet switches use MAC addresses from packet headers to direct traffic in the network. Switches can operate on several OSI layers. Physical layer switches, hubs, are simple devices that do not manage anything that passes through them, the incoming packet is sent out to every connected port, except for the port from where it originates. Ethernet switches operate on Data link layer and acts like a bridge. The latter uses Spanning Tree Protocols to avoid loops in a Local Area Network (LAN). Microsegmentation and full duplex mode are used to prevent collisions between devices connected to the switch. The most common ability of layer-3 switch is IP multicast awareness - device has to signal if it wants to listen on the multicast, more efficient way to multicast. Layer 4 switches is defined differently by its manufacturers. The most common addition is NAT, but switches may have load balancing, firewall, VPN - depends on the company. Application layer switch may distribute load based on URLs or by using some fancy technique, which may include web cache and participate in content delivery network. [12, 13, 14]

2.2.3 Routers

Routers are devices that connect different networks together. Routers represent the backbone of the whole internet. One of their roles is to read addresses on the data packets sent by a client and figure out where to redirect it in order to reach the final destination. Moreover, routers are capable of forwarding the traffic that flows through them to assure the packets delivery. While processing data packets the router consults its routing table which tells the router what to do with the packet. Routers possess a central processing unit and some form of memory to keep the routing table accessible. Most routers are dedicated hardware units. This makes routing faster than it would be when software routing, although the latter is not that uncommon these days. This thesis also implements software routers, since the whole idea is free and open source networking. [15]

2.3 Requirements for fault-tolerance

Fault-tolerance in networks can be broken down to two different forms of redundancy. Firstly, there is spatial redundancy which means there is more than one copy of the computer hardware, software and/or data used in the network. Sec-

ondly, there is the temporal redundancy which means that if data packet is sent and there is no aknowledgement packet received during the timeout interval, about successful reception, the packet is resent. This kind of fault tolerance is provided by Automatic Repeat reQuest (ARQ protocols). [16, 17]

In a good fault tolerant network setup both spatial and temporal redundancy are taken into consideration when designing the system. Spatial redundancy offers good protection against hardware failures, where the hardware device can not reset - permanent failures. Temporal failures can pass if for example a network interface which is at one point under huge transmission load, gets the load serviced or shared by another interface. Therefore temporal redundancy requires less resources [17]. In case there is extra ethernet interfaces free, temporal redundancy could take a form ethernet bonding. The latter would allow for ethernet port fail over, also load balancing is possible, if configured correctly.

2.4 Common solutions

Cisco offers a whole range of networking devices for networking along with their own operating system for the devices. Using Cisco's networking solution means that the users accept vendor lock-in. Cisco's solutions offer reliable and fast redundancy for routers and switches. Cisco's networking gear offers high performance, scalability, automation and redundancy. Cisco also offers a wide range of routers and switches for different types of working conditions. [18, 19]

HP also offers their own routers and switches which bring consistent and open-standard networking to the enterprise level networks. HP also offers full support to their products and their networking solution is rich with features necessary for reliable networks. HP also claims that their networking solutions features dramatically reduces the network complexity and lowers the cost of ownership. [20, 21]

Juniper Networks also offers their own networking hardware with complete kit: switches, routers, etc. Besides, they offer a wide range of products for their networking solutions which deliver high-availability, fault tolerance, high throughput and scalability. [22, 23]

2.5 Related work

The following subsections cover several scientific research papers that offer different solutions to the fault tolerant networking problem.

2.5.1 A Scalable and Fault Tolerant Network Structure for Tree Networks of Mission Critical Systems

Hierarchical Scalable Fault-tolerant Ethernet (H-SAFE) offers fault-tolerance on two levels. Firstly, on the hardware level, the topology consists of subnets. Each subnet contains two switches and a fixed number of nodes. In the subnets each node has two NICs which are connected to the two switches of the subnet. The switches of the subnets are in turn connected to the core switches of the network and thus creating a hierarchical network structure. The core switches have dual connectivity to the subnet switches and the subnet switches have a dual connectivity to the nodes of the subnet. This creates multiple pathways for the information to travel along in case of a network error. Scaling this kind of topology is easy since additional subnets can be added easily and each subnet only takes up two ports of one core switch. [24]

Secondly, on the software level fault tolerance is implemented with the heartbeat mechanism to detect and recover from network faults. Each node has Fault Tolerant Ethernet (FTE) software installed and it operates within a subnet. Since there are two NICs per node, the two NICs broadcast their heartbeats in two different groups, group A and group B. Heartbeats are sent to everyone in the designated group (different multicast addresses) and if more than two heartbeats have not been received by other interfaces the software knows there is a fault in the path. The fault is usually detected within 750ms. [24]

Each subnet has a master node which has the job to maintain information about its own subnet and communicate that information to other master nodes within other subnets. When there is a path information in the subnet, the master node broadcasts it to other master nodes. Communication between subnets happens through master nodes. [24]

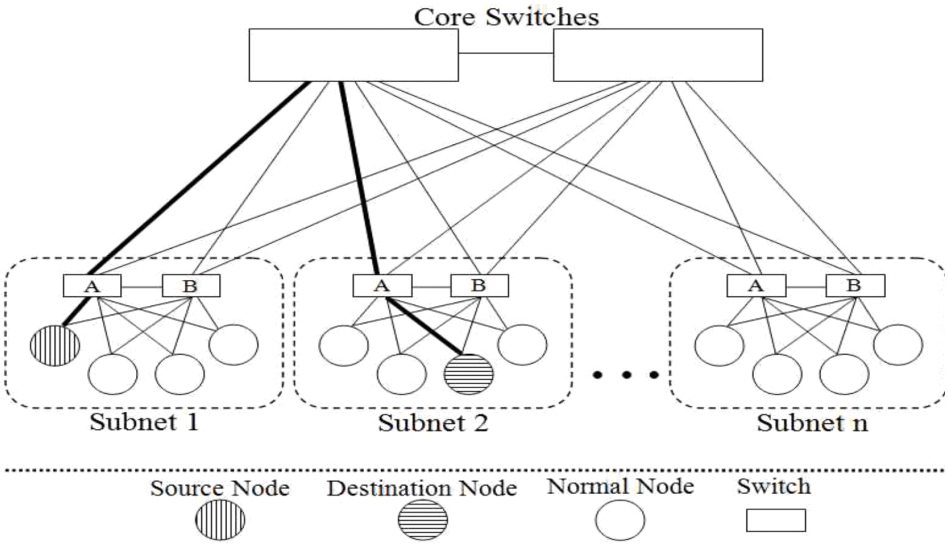

Figure 1: Overall Architecture [24]

2.5.2 A New Fault Tolerant Multistage Interconnection Network

A Four Tree network is an $N \times N$ network with multiple stages. Usually, the first and last stages of the network have 2×2 interconnects and the rest have 3×3 interconnects. Four Tree (FT) network is irregular network since it contains an unequal number of switches at its different stages. It also supports several paths from input to output and the path lengths may vary. A FT is constructed of two identical groups (MDOT networks [25]) on top of each other. Aforementioned groups are made with MSB in mind of the source-destination terminals. Every switching element of 3×3 in a stage is looped together with another switching element (SE) from another group in the corresponding stage. Inputs and outputs of the whole network are connected to both subgroups (MDOT networks [25]) via multiplexers and demultiplexers. This approach assures that if the primary path is busy or out of order, the data can be routed through a secondary path of the other sub-network. FT network can only survive one switch error. If more than one switch dies in a certain stage of the FT network, there are going to be some inputs and outputs disconnected. [26]

Modified Four Tree network (MFT) is an irregular network that provides several paths from source to destination. MFT also has a property of regular networks since it has the same number of switches in all the stages except for the first and last stage. MFT applies the same methodology to 3×3 SEs as does FT. It loops the SEs in the same stage together to boost its fault tolerance. Therefore, the SEs in the same stage are looped together with the SEs in the sub-groups but also in

the same stage. Every input and output is connected to both of the sub-groups via multiplexers and demultiplexers. MFT approach ensures that if two switches in the loop (talking about the same stage SEs) are faulty even then some form of connectivity is preserved in the network - some inputs are connected to the outputs. While the FT is more cost effective, the MFT is more fault tolerant. [26]

The paper also shows and analyses how the Permutation passable (one to one communication of the input and output) is better in the case of MFT when compared to a FT. The communications always along the most suitable (shortest) path, if the path is busy or faulty a secondary path is used. If no alternative path is found then the request is simply dropped or reported as having a clash. Permutation parameter also varies with path length. [26]

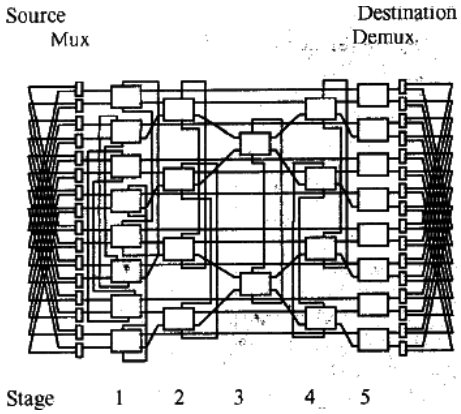

Figure 2: FT MIN of size N=16 [26]

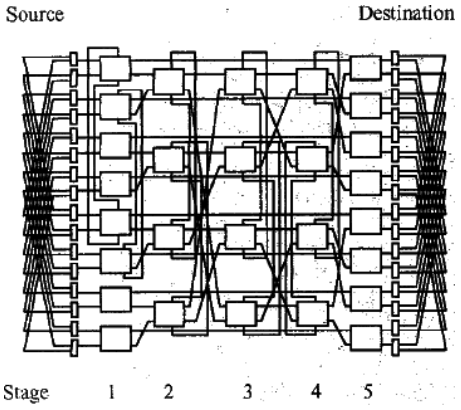

Figure 3: MFT MIN of size N=16 [26]

2.5.3 The Augmented 3D-Tree Fault-Tolerant Network

The Augmented 3D-Tree (A3DT) is an augmented version of 3D-Tree, it was designed because of fault tolerance issues the 3D-Tree suffered from. 3DT is based on 2×4 SEs and it has issues with certain failure patterns which cause the network to fail under a dynamic rerouting scheme even when fault free paths exist in the network. The A3DT is based on 4×6 SEs connected together in such a way that protects against the 3DT failures, but still keeping the underlying network properties. [27]

In 3DT consists of 2×4 SEs, two inputs and four outputs per switching element. Each consecutive stage contains double of the amount of SEs creating a 3D network topology - one layer of the 3DT looks like a tree graph and the 3 dimensions are achieved when these trees are stacked on top of each other. When a two-SE-fault pattern occurs (two switches fail in the same branch and same stage) a portion of the destinations would become unreachable. The dynamic rerouting property

of the 3DT would not be able to preserve connectivity even through a previous stage of the network. A3DT solves this two-SE-fault pattern by connecting the tree points which are in the same branch and same stage - although this assumes that the SEs are of crossbar-type and 4×6 SEs. [27]

The routing algorithm of the 3DT is similar to the A3DT when the network is not experiencing a critical two-SE-fault as mentioned above. If the latter does occur the A3DT routing scheme would offer an alternative path through previous stage to avoid counteract such occasion. [27]

The research paper analyses the effectiveness of the A3DT vs 3DT and points out that the A3DT topology is granted a better multiple-fault tolerance capability when compared to a 3DT. The fault tolerance could be even greater if the SEs in the same level are connected in a form of a chain, but this would result in additional complexity of the network which is not desired. The paper also analyses the mean time to failure (MTTF) and displays results which state that the MTTF of a 3DT is higher because there needs to be less fault-free SEs when compared to A3DT. The proposed A3DT is designed to address the two-SE-fault issue and it achieves it. "The advantage of the proposed augmentation scheme is the multiple-fault tolerance improvement along with the preservation of the basic properties of the original 3D-Tree network such as dynamic rerouting, nonblocking (under the fault-free behavior) and cell sequencing." [27]

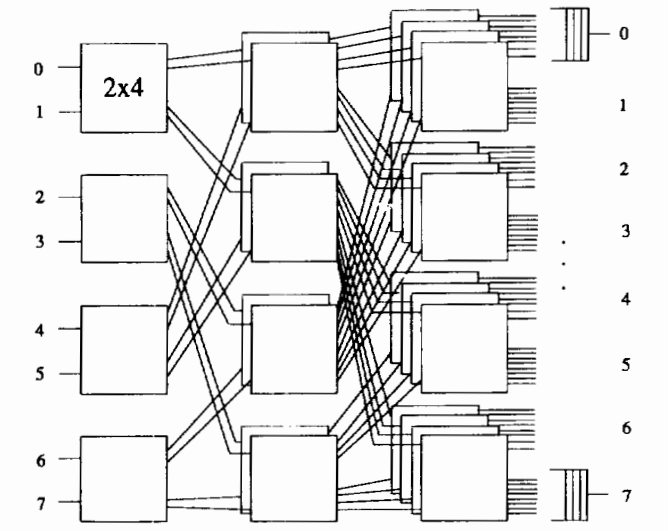

Figure 4: An 8×8 3D-Tree network [27]

2.5.4 Conclusion

The second chapter has covered prerequisites for fault tolerant networking. Different solutions were offered. Considering the available hardware for this thesis, the offered networks can not be implemented. Therefore, the following chapter, drawing knowledge from the related works and overall composition of the fault tolerant networks, focuses on a solution that is achievable with the possessed hardware. Chapter 3.1 firstly provides explanation on what software is necessary for the proposed solution. In 3.2-3.3 the author explains in detail how the main server and additional nodes were setup.

3 Network hardware configuration

Previous chapter gave an overview of fault tolerant networking — what it embodies, consists of and uses to achieve redundancy technologies were used on it. The following chapters 3.1-3.3 will explain in detail how to setup the first stage of the network — the main emphasis is on the main server. The main server will start to provide a bootable images with iPXE and NFS. The picture below illustrates the first stage of the network.

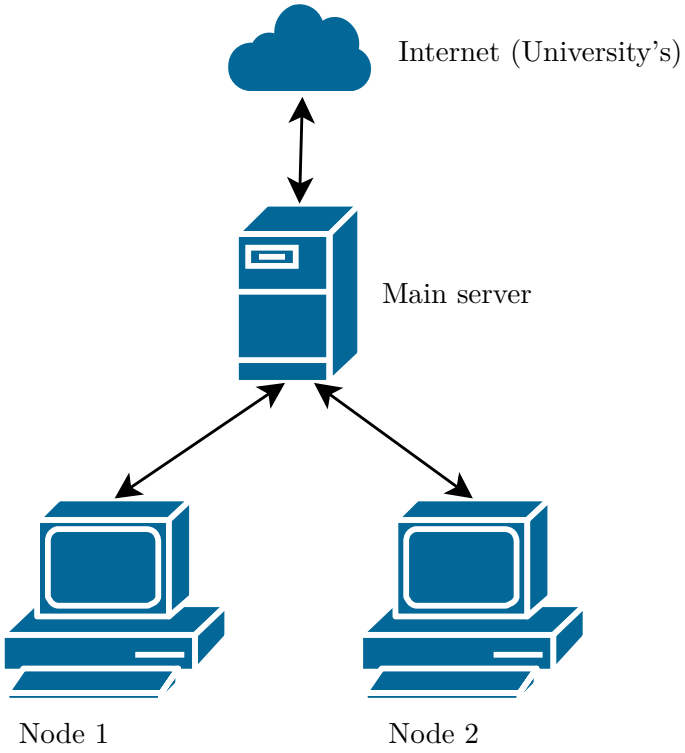

Figure 5: First stage of network topology

3.1 Used technologies

is subsection covers the technologies used on the main server in order to achieve the required features for the thesis. The over all guide for the 3.2.1-3.2.10 is based on Jacek Kowalski's blogpost "Debian Live webboot & DNS" [28].

3.1.1 DHCP

Dynamic Host Configuration Protocol (DHCP) is protocol that enables the computer it is installed on to hand out IP addresses to computers connected to it

in a given network. These IP addresses are predefined as a range in the DHCP configuration and whenever a new computer is connected to the DHCP server. The latter automatically hands out necessary information for the new computer to access the network. There might be a conflict with computers that request the IP address when two computers are configured to use the same static IP address, but when the computers are configured to use DHCP such conflicts are avoided. `/etc/default/isc-dhcp-server` and `set INTERFACES="eth0"` — this is the interface that serves the DHCP. [29]

3.1.2 NFS

Network file system (NFS) is a service that allows computers to mount file systems over the network. The mounted file system acts as a directory from where files can be read and written to, as if the directory is part of the computer it is mounted too. The latter option allows to save resources — no need for hard drives on local machines. NFSv4, which is also used in this thesis, works via firewalls and internet. NFSv4 requires TCP protocol communication in order to provide the service. NFSv4 uses TCP port 2049 for communication. [30]

3.1.3 iPXE

iPXE is an open source network boot firmware which is used in the thesis's setup. iPXE features full implementation of PXE but with added enhancements. It allows booting from a web server, wide area networks, Infiniband network, wireless network and via other technologies. Everything is controlled by a iPXE script. iPXE community offers a IRC chat where most problems which are not documented can be solved with the help of other users. [31]

3.1.4 Apache2

Apache HTTP server is a open-source web server software. The software is robust and able to handle commercial-grade sites with ease. It is currently the most used web server software. Apache software is highly configurable and is able to operate on many various operating systems. [32, 33]

3.1.5 BIND

Berkeley Internet Name Domain (BIND) is an open source software that implements Domain Name Server (DNS), protocols which specify how computers can find each other over the internet, based on their names, service for Unix-like systems. It offers a robust and reliable service for the users, at the same time offering a fine grained control for the administrators. [34, 35]

3.1.6 iptables

iptables is a command line program used to configure packet filtering ruleset on Linux systems — linux kernel based firewall. iptables is also configured to allow Network Address Translation (NAT) configuration (masquerading, port forwarding, transparent proxying). [36, 37]

3.1.7 TFTP

tftpd-hpa Trivial File Transfer Protocol (TFTP) is a file transfer protocol used for remote booting diskless devices. HPA's implementation of TFPT has many bugs fixed when compared to the original and is portable on almost all modern Unix variants. It is very simple to implement which can also be seen in Chapter 4 of this thesis. TFTP can only read/write files to/from the remote server. It lacks security features and therefore should be placed behind a firewall. [38, 39, 40]

3.1.8 Keepalived

Keepalived is project written in C to provide a Linux Virtual Router, with a robust and reliable fail over feature. Keepalived implements the VRR Protocol to handle router failures. Keepalived also includes load balancing through Linux Virtual Server and health checks of network servers. Keepalived allows administrators to build Virtual routers and load balancing with one single package without having to install and configure multiple packages like lvs, mon, fake, heartbeat. Sending notifications about the health of the network is also included in the package. [41, 42]

3.2 Configure the main server

The previous subsection described technologies that will be used in the main server setup. The following subsection will present the setup in detail with configuration file examples.

The main server has 5 ethernet interfaces, one is built in onto the motherboard and the other four are courtesy of a Mikrotik 4 port Gigabit ethernet PCIe card. These ports are configured as declared in the Listings 1 illustration. To briefly summarise, the first interface is devoted to joining the ISP network and the the Network Interface Card (NIC) — this offers a private network via DHCP which is also configured on the main server. The NIC's interfaces are joined under the software bridge which is given a static address and serves as a default gateway for the private network.

3.2.1 /etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet dhcp

iface eth1 inet manual
                up ifconfig eth1 up
        down ifconfig eth1 down

iface eth2 inet manual
        up ifconfig eth2 up
        down ifconfig eth2 up

iface eth3 inet manual
        up ifconfig eth3 up
        down ifconfig eth3 up

iface eth4 inet manual
        up ifconfig eth4 up
        down ifconfig eth4 down

auto br0
iface br0 inet static
        bridge_ports eth1 eth2 eth3 eth4
                address 192.168.66.40
                broadcast 192.168.66.63
                netmask 255.255.255.224
                network 192.168.66.32
                bridge_stp off
                bridge_waitport 0
                bridge_fd 0
                bridge_ports none
```

Listing 1: `/etc/networks/interfaces` configuration

3.2.2 /etc/resolv.conf

The address of the closest DNS resolver, may be the ISP one - assigned by DHCP or set manually to any public DNS resovler. In our setup the University's resolver was the closest. `/etc/resolv.conf` should contain the DNS address.

```
nameserver 193.40.5.39
```

Listing 2: `/etc/resolv.conf` configuration

3.2.3 /etc/hosts

Hosts configuration file provides the hostname aliases for IP. Which makes it possible to use the service by hostname as opposed to by IP. Extremely useful in absence of Name Resolver by DNS server. `/etc/hosts` should contain

```
127.0.0.1       localhost
127.0.1.1       cockshot
```

Listing 3: `/etc/hosts` configuration

3.2.4 iptables rules

```
-A INPUT -p tcp -m state --state NEW -m tcp --dport 2049 -j ACCEPT
-A INPUT -j DROP
-A FORWARD -i br0 -o eth0 -j ACCEPT
-A FORWARD -i eth0 -o br0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -m limit --limit 50/min -j LOG --log-prefix "Firewall denied
 ↪  50/m: "
-A FORWARD -j DROP
-T NAT -A POSTROUTING -o eth0 -j MASQUERADE
```

Listing 4: iptables configuration

Make sure your port 2049 is open, it is necessary for TFTP server communication (rule 1). Afterwards, if the communcation was not meant to pass port 2049, deny other port interactions. Rules 3-6 denote classic gateway and router serving between ISP and private LAN. Here we allow all the packets from the private LAN

(br0) to be sent to ISP network (eth0) - rule 3. Backwards we allow only in case connection was previously established or if it is related to previously established one - so cutting of the undesired incoming communication (rule 4). The last rules are for logging before cutting off (rule 5) and the actual cut off (rule 6). Rule 7 hides the private network connections behind a NAT with the command. Run the following commands as `root`:

```
# sysctl net.ipv4.conf.all.forwarding=1
# sysctl net.ipv4.conf.default.forwarding=1
```

Iptables' FORWARD chain of the INPUT table will not be accepted by the kernel, if the `/etc/sysctl.conf` is not configured correctly — will not allow kernel to actually forward packages between interfaces. Therefore, these lines have to be appended to the `/etc/sysctl.conf` file in order for the system to fetch them automatically after a reboot.

`/etc/hosts.allow` and `/etc/hosts.deny` are configuration files which make sure, that the services that are accessed by other computers on the main server are legitimate — not some unaccounted for computers on the network. `hosts.allow` and `hosts.deny` configuration files are currently empty since this way all connections are allowed. But the `hosts.deny` should contain `ALL:ALL` and `hosts.allow` should contain services and IP addresses which should be allowed - this is a more secure and proper way to use these files. Since the setup network was a third private subnet from the outside world, the security threats were small.

These commands download the files which are needed for iPXE boot. The first four are needed to boot Debian Live. The last two are necessary for iPXE boot.

```
 $ wget http://cdimage.debian.org/debian-cd/current-live/amd64/webboot/
↪  debian-live-8.4.0-amd64-standard.squashfs
 $ wget http://cdimage.debian.org/debian-cd/current-live/amd64/webboot/
↪  debian-live-8.4.0-amd64-standard.initrd.img
 $ wget http://cdimage.debian.org/debian-cd/current-live/amd64/webboot/
↪  debian-live-8.4.0-amd64-standard.squashfs.packages
 $ wget http://cdimage.debian.org/debian-cd/current-live/amd64/webboot/
↪  debian-live-8.4.0-amd64-standard.vmlinuz
 $ wget http://ftp.de.debian.org/debian/pool/main/g/glibc/
↪  libc6-udeb_2.19-18+deb8u4_amd64.udeb
 $ wget http://ftp.de.debian.org/debian/pool/main/g/glibc/
 $ wget http://boot.ipxe.org/ipxe.pxe
```

3.2.5 BIND

Copy the default `/etc/bind/db.127` as a template for your bind setup. In this case `/etc/bind/db.192`, which should contain the following. By default iPXE will try to fetch the following web address: `http://pxe.local/menu.cfg`. Therefore, the local network's DNS has to resolve the adderss `pxe.local` into an actual IP of the local network — DNS server is necessary. The `BIND9` was selected and configured as follows.

```
1  ...
2  @       IN      SOA     pxe.local. root.pxe.local. (
3  ...
4  @       IN      NS      ns.pxe.local.
5  1       IN      PTR     pxe.local.
6  2       IN      PTR     pc.pxe.local.
```

Listing 5: `/etc/bind/db.127` configuration

The `db.local` also has to be modified. "A" kind of record is needed for resolving iPXE's `http://pxe.local/menu.cfg` into a preferrable IP address.

```
1  ...
2  @       IN      SOA     localhost. root.localhost. (
3  ...
4  @       IN      NS      localhost.
5  @       IN      A       127.0.0.1
6  PXE     IN      A       192.168.66.40
```

Listing 6: `db.local` configuration

This file will contain a NS pointer for a `pxe.local`. Create a file `db.pxe.local` and fill it with.

```
...
@       IN      SOA     pxe.local. root.pxe.local. (
...
@       IN      NS      ns1.pxe.local.
@       IN      A       193.40.5.39
ns1     IN      A       193.40.5.39
```

Listing 7: `db.pxe.local` configuration

Enabling recursion allows iteration over queries. Setting bind9 to listen on the private LAN./`etc/bind/named.conf.options` has to have the following two lines added into the options' parentheses:

```
...
listen-on {127.0.0.1;192.168.66.40;};
recursion-yes;
...
```

Listing 8: `named.conf.options` configuration

Make sure that the `/etc/bind/named.conf.default-zones` contains the following:

```
...
zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};
...
```

Listing 9: `/etc/bind/named.conf.default-zones` configuration

3.2.6 DHCP server configuration

The hosts that join the private LAN have to be given an IP, subnet, default gateway and DNS information, in order for the hosts to be able to access the internet. Therefore, edit the `/etc/dhcp/dhcpd.conf`, so it contains the necessary information.

```
...
#the next line helps dchp to understand the network it is on.
subnet 192.168.66.0 netmask 255.255.255.0{}
...
subnet 192.168.66.32 netmask 255.255.255.224 {
  range 192.168.66.34 192.168.66.62;
  option domain-name-servers 192.168.66.40;
  option domain-name "local";
  option routers 192.168.66.40;
  option broadcast-address 192.168.66.63;
  default-least-time 600; # 10 minutes
  max-lease-time 3600; # 1 hour
  next-server 192.168.66.40;
  if exists user-class and option user-class = "iPXE" {
        filename "http://pxe.local/menu.cfg";
  } else {
        filename "ipxe.pxe"; #alternative "undionly.kpxe"
  }
}
...
```

Listing 10: `/etc/dhcp/dhcpd.conf` configuration

In our custom settings we additionally provide `next-server option` pointing to the address of the TFTP server. Additionally the `filename option` is used to point the TFTP client to the corresponding file which to download. This is crucial for achieving PXE HTTP boot functionality. In our setup we have a conditional file name option for the "two-step" iPXE firmware boot. First, the BIOS starts the PXE boot sequence issuing a DHCPDISCOVER and receives a "ipxe.pxe" firmware to execute. Afterwards, the running iPXE firware issues DHCPDISCOVER once again and is pointed at "http://pxe.local/menu.cfg" for iPXE configuration. Rest of the `dhcpd.conf` file can be left like it is by default.

3.2.7 TFTP

In the previou section we described the iPXE boot process. The DHCP options `next-server` and `filename` only make sense in a presence of a TFTP server. The BIOS PXE boot sequence needs to know from where to fetch the PXE firmware (and we provide it through DHCPOFFER using option `next-server`). Additionally, the BIOS PXE boot sequence is using TFTP protocol to actually fetch the firmware data, and following the TFTP protocol we need to provide the filename to fetch (we do this with DHCP option `filename`). In our setup the host carrying the DHCP server is also the one serving the TFTP. Therfore, the same IP address

is used for the `next-server` option. Finally, we point the context ROOT of the TFTP to `/srv/tftp`. Latter contains all the files we are going to provide over `filename` DHCP option.

When setting up `tftp-hpa` package, the installation destination is `/etc/srv/`.

In the following setup username `"tftp"` is required for the `/srv/tftp` director, in order to let TFTP share the files. By having permissions 755 on directory and 644 on files. The address assigned as such to listen by default to all IPs attached to the network interfaces of the host. The secure option is the most critical here. Before starting TFTP, it will actually `change-root` into `/srv/tftp` and switch to tftp user, so the server process will never run in root permission or in / directory.

Make sure the `/etc/default/tftpd-hpa` has the following content (should be the default):

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

Listing 11: `/etc/default/tftpd-hpa` configuration

3.2.8 Apache2

After iPXE firmware is loaded and menu configuration is applied, the actual boot data (OS, kernels, etc.) is obtained from web. The default location is `http://pxe.local/`. Therefore, we setup the Apache2 web server with VirtualHost reference `pxe.local` serving the contents of `/var/www/tftp`. Create `/etc/apache2/sites-available/pxe.local.conf` file and fill it with:

```
<VirtualHost *:80>
        ServerName pxe.local
        DocumentRoot "/var/www/tftp"
</VirtualHost>
```

Listing 12: `pxe.local.conf` configuration

In `apache2.conf` file there is an option that `/srv/` directory can be set to service the tftp and nfs, but for the sake of this thesis this option was left commented. Therefore, the directory that the Apache2 services is `/var/www/`. The `apache2.conf` should contain the following lines:

```
...
<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
...
```

Listing 13: `apache2.conf` configuration

Use the following command to make your newly created site available to the network:

```
# a2ensite pxe.local.conf
```

3.2.9 iPXE

This subsection contains the customised iPXE configuration. We added a start point for an actual disk-less installation of the Debian8 on the `mainserver` host. We provide its kernel with essential arguments to find its rootfs using Network File System (NFS). The rootfs is afterwards mounted with read and write flags and therefore it is used by host to store all the changes.

```
#!ipxe
iseq ${mac} ac:9e:17:f0:61:fd && goto noblesse ||
iseq ${mac} 00:26:55:3c:65:09 && goto mainserver ||
iseq ${mac} 00:16:36:d2:ed:91 && goto minazuki ||
iseq ${mac} 00:16:36:68:51:b5 && goto yachiru

:minazuki
        kernel minazuki-root/vmlinuz noswap noprompt
 ↪  ip=192.168.66.39:::255.255.255.224:minazuki:eth3:off netboot=nfs
 ↪  nfsroot=192.168.66.40:/srv/nfs/minazuki rw root=/dev/nfs
        initrd minazuki-root/initrd.img
        boot

:noblesse
        kernel noblesse_boot/vmlinuz noswap noprompt
 ↪  ip=192.168.66.62:::255.255.255.224:noblesse:eth0:off netboot=nfs
 ↪  nfsroot=192.168.66.40:/srv/nfs/noblesse rw root=/dev/nfs
        initrd noblesse_boot/initrd.img
        boot

:mainserver
        kernel mainserver-root/vmlinuz noswap noprompt
 ↪  ip=192.168.66.45:::255.255.255.224:mainserver:eth1:off netboot=nfs
 ↪  nfsroot=192.168.66.40:/srv/nfs/main_server rw root=/dev/nfs
        initrd mainserver-root/initrd.img
        boot

:yachiru
        kernel yachiru-root/vmlinuz noswap noprompt
 ↪  ip=192.168.66.37:::255.255.255.224:yachiru:eth3:off netboot=nfs
 ↪  nfsroot=192.168.66.40:/srv/nfs/yachiru rw root=/dev/nfs
        initrd yachiru-root/initrd.img
        boot
```

Listing 14: `menu.cfg` configuration

3.2.10 NFS

In the previous section we described the disk-less host installation of Debian8. The NFS server however is crucial for the setup. The NFS server is storing the root file system for all the hosts we setup in disk-less mode. In the setup we host NFSserver on the same nodewith DHCP and TFTP. This one line exports the `main_server` directory to the static IP-address that the main_server node holds:

```
1 /srv/nfs/main_server 192.168.66.45(rw,sync,subtree_check,no_root_squash)
```

Listing 15: `/etc/exports` configuration

3.3 Adding nodes to the configuration

In order to start adding machines to the network a live boot must be performed. Simply installing Debian onto the PCs will not work since. None of the computers on the network, with the exception of the main server, have any storage devices installed in them. After the live environment is up and running, debootstrap tool must be used which allows to install the operating system into the exported directory. The latter is offered by the main server which has NFS setup. While no storage devices are present (not essential) an access to the Debian repository has to be available. A private network was used, that was specifically created for this thesis, in order to access the internet and the NFS server.
In the following section we describe how to create a bootable media device. The latter is necessary for firstly configuring the client to accept `PXE boot`. Afterwards, an momentary environment is needed, where the necessary steps can be performed, to make the client fully part of the disk-less boot setup.

3.3.1 Creating bootable media

While it is easy to configure iPXE menu.cfg to offer Debian Live image to every new client, that is connected to the network, the author opted for a bootable USB way. This way any computer connected to the network via ethernet cable can not be booted via iPXE automatically. This adds a layer of security to the network setup.

The USB bootable media was created using UNetbootin software, available at http://unetbootin.github.io (available for: Windows, Linux, OS X). Debian Live CD image is also needed, and it can be downloaded form Debian's homepage. Using the UNetbootin software is self explanatory and all what needs to be done is to follow the instructions presented by the software. Bootable media device can also be created by using alternative software (e.g. Universal USB installer (Windows), Rufus USB installer(Windows) , Startup Disk Creator (Ubuntu),Winusb (Ubuntu). After the bootable media is ready the BIOS settings of the target machine may have to be tweaked to allow USB boot.

3.3.2 Making changes to the main server

Go to your `nfs` directory which contains all the exported file systems - `/srv/nfs/`. Create a new directory to where the new node will be installed to.

```
# mkdir yachiru
```

Create another directory to keep the files essential for iPXE boot in `/var/www/tftp/` directory - vmlinuz, initrd, System.map binary and configuration binary for the debian image.

```
# mkdir yachiru_boot
```

Thirdly a symlink is needed to create a shortcut for the `/srv/nfs/yachiru` to be accessible in the `/var/www/tftp/` directory.

```
# ln -s /srv/nfs/yachiru yachiru-root
```

After the necessary directories are created a line should be appended to the `/etc/exports` file. Something like this:

```
1 ...
2 /srv/nfs/yachiru 192.168.66.37(rw,sync,subtree_check,no_root_squash)
3 ...
```

Listing 16: `/etc/exports` configuration

Entering the correct IP address does assume that the Debian Live image has successfully booted from the USB stick and the IP address given by the DHCP on the main server has been checked - commands like `ifconfig` and `ip addr l` accomplish the last task just fine.

3.3.3 Configuring the Live image

Now that the Live image has successfully booted the first thing to do would be to check what is the IP address given by the DHCP and whether or not the new

node is able to ping the main server.

```
# ifconfig
# ping 192.168.66.40
```

Both of these commands should have a positive output — DHCP has given the new node an IP and the node is able to ping the main server.

Now to check if the `/etc/exports` file is exporting the right directory to the new node the following command should be run:

```
# showmount -e 192.168.66.40
```

After the last command has returned a successful response, two directories have to be created to where to mount the exported NFS folder and finally to mount the exported directory in the new node.

```
# mkdir /mnt/nfs
# mkdir /mnt/nfs/yachiru
# mount -t nfs 192.168.66.40:/srv/nfs/yachiru /mnt/nfs/yachiru
```

To check if the NFS is really mounted in the read-write mode on all computers a test file should be created into the `/mnt/nfs/yachiru`. The latter should show up on the main server in the `/srv/nfs/yachiru` directory. The server will see the files stored by the client and allow making modifications to them.

Afterwards debootstrapping tool has to be installed to install the Debian Jessie operating system into the `/mnt/nfs/yachiru` directory.

```
# debootstrap jessie /mnt/nfs/yachiru http://ftp.ee.debian/org/debian
```

If everything installed where it was supposed to, the `etc/fstab` should be given a proper mounting configuration. In order to avoid writing everything by hand the following command can be executed.

```
# mount | tail -n1 >> etc/fstab
```

The appended line should also be modified since the current syntax does not work for the `fstab` file.

```
192.168.66.40:/srv/nfs/yachiru / nfs
 ↪  rw,relatime,vers=4.0,rsize=524288,wsize=524288,namelen=255,hard,
 ↪  proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=192.168.66.37,
 ↪  local_lock=none,addr=192.168.66.40 0 0
```

Listing 17: fstab configuration

Change the hostname to represent the new node on the network and add proper nameservers to `etc/resolv.conf`. The `etc/network/interfaces` needs to be modified according to the network requirements as well.

```
auto lo
iface lo inet loopback

auto eth3 -192.168.66.37
iface eth3 inet static
        address 192.168.66.37
        netmask 255.255.255.224
        network 192.168.66.32
        broadcast 192.168.66.63
        gateway 192.168.66.45
```

Listing 18: `etc/network/interfaces` configuration

The `/proc /dev /sys` directories can be linked to the freshly installed image with `--bind` option of the `mount` command.

While in `/mnt/nfs/yachiru` directory a `chroot .` should be executed to change the current root to a freshly installed Debian's root. Now the following packages should be installed - locales,tzdata,ntp,pciutils,aptitude,ssh. The `locales` configures the language settings. `tzdata` configures time zone information. `ntp` allows for automatic time adjustments. `pciutils` is necessary for hardware inspection. `aptitude` is used for package managemet. `ssh` is essential for remote access. The root password should be set and a non-root user should be created. `pciutils` package contains tools to figure out the node's current hardware configuration and allows to check which firmware packages should be installed. With Aptitude the following packages were installed on yachiru:

initramfs-tools, binfmt-support, binutils, nfs-common, alsa-firmware-loaders, alsa-utils, autoconf, bison, broadcom-sta-dkms, db5.3-util, debian-keyring, dh-make, ed, firmware-atheros, firmware-b43-installer, firmware-bnx2, firmware-bnx2x, firmware-brcm80211, firmware-linux, firmware-linux-free, firmware-ralink, firmware-realtek, firmware-samsung, flex, g++-4.9-multilib, g++-multilib, gcc-4.8-locales, gcc-4.8-multilib, gcc-4.9-locales, libtool, linux-wlan-nf-firmware, subversion-tools, extlinux, linux-image-3.16.0-4-amd64, linux-image-amd64

Finally `ls /boot/` should contain the initrd image, vmlinuz image, System.map binary and configuration binary for the Debian image if all of the packages installed in Aptitude were successfully set up.

3.4 Second stage of the network setup

The second stage of the hardware configuration means, that the nodes that were configured previously will now be made into routers with firewalls. Also the private networks DHCP server will be moved behind the routers. While allowing more machines to be added to the private network, by setting them up as shown in chapter 3.3 and connecting them to the switch that connects the main server and routers. Previously we had the DHCP, NFS, TFPT, iPXE and the default gateway on the same machine. Now we separate the DHCP, PXE, TFTP, NFS from the firewall, iptables and default gateway. The last three are moved onto the previously configured disk-less boot hosts. Which will now act as firewalled routers for the private network. The figure below illustrates the aforementioned second stage.

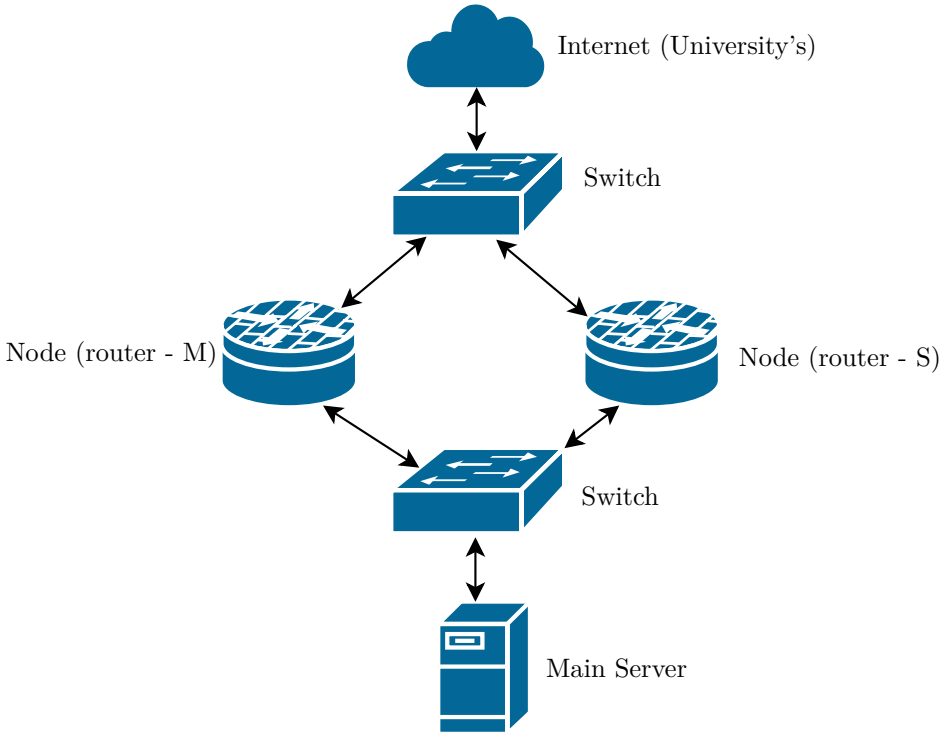

Figure 6: Second stage of network topology

To enable router fail over the `keepalived` package needs to be installed on the routers. The `/etc/keepalived/keepalived.conf` file is configured as follows.

```
vrrp_instance public_ips {
state MASTER
interface eth0
priority 101
virtual_router_id 12
advert_int 1

virtual_ipaddress {
  192.168.66.224 dev eth0 label eth0:VirtIP
  }
}
vrrp_instance private_ips{
state MASTER
interface eth1
virtual_router_id 13
priority 101
advert_int 1

virtual_ipaddress {
  192.168.66.45 dev eth1 eth1:VirtIP
  }
}
```

Listing 19: `etc/keepalived/keepalived.conf` configuration

When changes are made to `/etc/keepalived/keepalived.conf` the service needs to be restarted. This latter is accomplished by using `service` command with restart option. Now check your interfaces with `ip addr l` command. The output should contain two IPv4 addresses per interface with one having an ending that states `eth1:VirtIP`. The latter one is the virtual IP set by the `keepalived` package.

Now, since the machines that have `keepalived` installed on them are going to be the routers, then `iptables` has to be configured. The following commands should be entered:

```
  # iptables -A FORWARD -i br0 -o eth0 -j ACCEPT
  # iptables -A FORWARD -i eth0 -o br0 -m state --state
↪  RELATED,ESTABLISHED -j ACCEPT
  # iptables -A FORWARD -m limit --limit 50/min -j LOG --log-prefix
↪  "Firewall denied 50/m: "
  # iptables -A FORWARD -j DROP
  # iptables -A POSTROUTING -o eth1 -j MASQUERADE
```

Save the iptabels configuration and also make sure, that these rules are activated after every system reboot. [43]

Finally, make sure, that the `/etc/sysctl.conf` has `net.ipv4.ip_nonlocal_bind=1` set. Load the parameter with `sysctl -p` command. [44]

4 Testing

There are two different scenarios tested within this thesis. First, there is a file that is copied with `rsync` over SSH to another server. During the transmission, the ssh channel can not be closed or the data transfer will fail. To test if a router failure would cause an SSH channel shut down, the following command was executed.

```
$ rsync --bwlimit=100 /backup.tar ssh
↪  andersm9@newmath.ut.ee:/home/pohl01/andersm9/backup.tar
```

At first the previous command was executed. The console displayed an active data transmission. Then the MASTER router was disconnected from the network. Then the transmission was checked, if it continued to trasmit data, the first part of the test was considered a success. If the data transfer failed, the test was considered a failure. In case the open channel was ported onto the SLAVE router successfully without cancelling the transfer, the disconnected router was reconnected. The status of the MASTER router was also checked, if it entered the MASTER state. Then the SLAVE router was disconnected, and the author checked if the MASTER router was healthy and the data transfer was also ongoing. After verifying that the connection was still alive, the SLAVE was reconnected and the overall rsync shh test was considered a success.

The second command used for testing is `wget`, which is put to download a big `.iso` file. The `--limit-rate=200K` option limits the download speed.

```
$ wget --limit-rate=200K http://cdimage.debian.org/debian
↪  -cd/current-live/amd64/iso-hybrid/debian-live-8.4.0-amd64-standard.iso
↪  --progress=dot
```

The active download test was conducted exactly like the open channel rsync ssh test. Keeping the same prerequisites for a success and failure.

5 Conclusion

The goal of this thesis was to create a computer network which could handle router and ISP fail overs so that the end user would not have connectivity issues. Should the network suffer a router failure and/or a ISP failure, the end user's internet connection should not be diconnected nor should the ongoing data trasnmission be cancelled.

The overall requirements for the thesis was that the MASTER router failure and the automatic data trasmission switching over to the SLAVE router would not cause any issues in the data transfer. This was tested with an active SSH connection which by default will not reconnect once the channel is broken. Also the wget was used which does not re-establish the download connection by default. Both of the tests were successful, the ongoing channels and active data trasmissions were not broken during the MASTER or SLAVE failures.

The final network topology looks like in the figure 6.

Two router computers that were setup like firewalled routers, are the gateways for the private network. Whenever one of them fails, the other one takes over — the main server will not fail the ongoing internet connection.

The future work includes adding an ISP failover to the network and achieving the same results. Additionally, if some new hardware is aquired, it would be possible to try and implement high-availability into the network. Also, the author would like to research if it is possible to implement self healing into the network, to preserve connectivity at the expense of computing power.

References

[1] R. Shimonski, "The Importance of Network Redundancy." http://www.windowsnetworking.com/articles-tutorials/netgeneral/Importance-Network-Redundancy.html, 2010. [Accessed: 18.05.2016].

[2] Wikipedia, "Transmission Control Protocol — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Transmission_Control_Protocol&oldid=719428729, 2016. [Accessed: 18.05.2016].

[3] Wikipedia, "IP address — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=IP_address&oldid=714596935, 2016. [Accessed: 18.05.2016].

[4] Wikipedia, "Internet Protocol — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Internet_Protocol&oldid=720906813, 2016. [Accessed: 18.05.2016].

[5] Wikipedia, "Virtual Router Redundancy Protocol — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Virtual_Router_Redundancy_Protocol&oldid=717542673, 2016. [Accessed: 18.05.2016].

[6] Wikipedia, "Common Address Redundancy Protocol — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Common_Address_Redundancy_Protocol&oldid=706709601, 2016. [Accessed: 18.05.2016].

[7] Wikipedia, "Hot Standby Router Protocol — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Hot_Standby_Router_Protocol&oldid=720743530, 2016. [Accessed: 18.05.2016].

[8] Wikipedia, "First-hop redundancy protocols — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=First-hop_redundancy_protocols&oldid=709049366, 2016. [Accessed: 18.05.2016].

[9] Wikipedia, "Gateway Load Balancing Protocol — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Gateway_Load_Balancing_Protocol&oldid=683593613, 2015. [Accessed: 18.05.2016].

[10] Cisco, "GLBP - Gateway Load Balancing Protocol." http://www.cisco.com/en/US/docs/ios/12_2t/12_2t15/feature/guide/ft_glbp.html#wp1027177. [Accessed: 18.05.2016].

[11] Wikipedia, "R-SMLT — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=R-SMLT&oldid=659515512, 2015. [Accessed: 18.05.2016].

[12] Wikipedia, "Network switch — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Network_switch&oldid=714688611, 2016. [Accessed: 18.05.2016].

[13] J. Tyson, "How LAN Switches Work." http://computer.howstuffworks.com/lan-switch.htm, 2001. [Accessed: 18.05.2016].

[14] Wikipedia, "Data link layer — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Data_link_layer&oldid=713250654, 2016. [Accessed: 18.05.2016].

[15] Wikipedia, "Router (computing) — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Router_(computing)&oldid=720269160, 2016. [Accessed: 18.05.2016].

[16] P. L. Dordal, "6 Abstract Sliding Windows — An Introduction to Computer Networks, edition 1.8.20." http://intronetworks.cs.luc.edu/current/html/slidingwindows.html, 2015. [Accessed: 18.05.2016].

[17] M. Médard and S. S. Lumetta, "Network Reliability and Fault Tolerance," in *Wiley Encyclopedia of Telecommunications*, John Wiley & Sons, Inc., 2003.

[18] "All Routers Products — Cisco." http://www.cisco.com/c/en/us/products/routers/product-listing.html. [Accessed: 18.05.2016].

[19] Wikipedia, "Cisco Systems — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Cisco_Systems&oldid=720386486, 2016. [Accessed: 18.05.2016].

[20] "HP Networking Routers." http://www8.hp.com/us/en/networking/routers/index.html. [Accessed: 18.05.2016].

[21] "HP Networking Switches." http://www8.hp.com/us/en/networking/switches/index.html. [Accessed: 18.05.2016].

[22] "Routers - Secure Network Router Solutions - Juniper Networks." http://www.juniper.net/us/en/products-services/routing/. [Accessed: 18.05.2016].

[23] "Solutions – Juniper Networks." http://www.juniper.net/us/en/solutions/. [Accessed: 18.05.2016].

[24] R. A. Memon, Y. Ryu, M. Shin, and J. m. Rhee, "A scalable and fault tolerant network structure for tree networks of mission critical systems," in *ICTC 2011*, pp. 255–256, Sept 2011.

[25] P. Bansal, K. Singh, and R. Joshi, "Routing and path length algorithm for a cost-effective four-tree multistage interconnection network," *International Journal of Electronics*, vol. 73, no. 1, pp. 107–115, 1992.

[26] S. Sharma and P. K. Bansal, "A new fault tolerant multistage interconnection network," in *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 1, pp. 347–350 vol.1, Oct 2002.

[27] M. Belkadi and H. T. Mouftah, "The augmented 3D-tree fault-tolerant network," in *Circuits and Systems, 1993., Proceedings of the 36th Midwest Symposium on*, pp. 621–624 vol.1, Aug 1993.

[28] J. Kowalski, "Debian Live webboot & DNS." `https://blog.jacekk.info/2016/01/debian-live-webboot-dns/`. [Accessed: 20.05.2016].

[29] "What is DHCP? — Indiana University Knowledge Base." `https://kb.iu.edu/d/adov`, 2015. [Accessed: 18.05.2016].

[30] "Chapter 9. Network File System (NFS) — Red Hat Enterprise Linux 6 Storage Administration Guide." `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/ch-nfs.html`, 2015. [Accessed: 18.05.2016].

[31] "iPXE - open source boot firmware." `http://ipxe.org/`. [Accessed: 18.05.2016].

[32] "Apache HTTP Server Project." `https://httpd.apache.org/ABOUT_APACHE.html`. [Accessed: 18.05.2016].

[33] Wikipedia, "Apache HTTP Server — Wikipedia, The Free Encyclopedia." `https://en.wikipedia.org/w/index.php?title=Apache_HTTP_Server&oldid=719366249`, 2016. [Accessed: 18.05.2016].

[34] Wikipedia, "BIND — Wikipedia, The Free Encyclopedia." `https://en.wikipedia.org/w/index.php?title=BIND&oldid=712320757`, 2016. [Accessed: 18.05.2016].

[35] "BIND — The most widely used Name Server Software." `https://www.isc.org/downloads/bind/`, 2016. [Accessed: 18.05.2016].

[36] "The netfilter.org "iptables" project." http://www.netfilter.org/projects/iptables/index.html. [Accessed: 18.05.2016].

[37] "iptables — Freecode.com." http://freecode.com/projects/iptables/. [Accessed: 18.05.2016].

[38] "tftp-hpa — Freecode.com." http://www.freecode.com/projects/tftp-hpa/. [Accessed: 18.05.2016].

[39] "tftpd(8) - Linux man page." http://linux.die.net/man/8/tftpd. [Accessed: 18.05.2016].

[40] Wikipedia, "Trivial File Transfer Protocol — Wikipedia, The Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=Trivial_File_Transfer_Protocol&oldid=711713587, 2016. [Accessed: 18.05.2016].

[41] "Keepalived for Linux." http://www.keepalived.org/index.html. [Accessed: 18.05.2016].

[42] A. Fletcher, "LVS NAT + Keepalived HOWTO." http://www.keepalived.org/LVS-NAT-Keepalived-HOWTO.html, 2002. [Accessed: 18.05.2016].

[43] "Bringing VM online, initial network configuration." https://courses.cs.ut.ee/2016/sa/spring/Main/Week004. [Accessed: 19.05.2016].

[44] "VRRP (Virtual Router Redundancy Protocol) Keepalived Configuration." https://www.atlantic.net/community/howto/vrrp-keepalived-configuration/. [Accessed: 19.05.2016].

Non-exclusive licence to reproduce thesis and make thesis public

I, Anders Martoja (date of birth: 19th of February 1992),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Fault tolerant networking using Linux based systems and obsolete hardware

supervised by Artjom Lind

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 20.05.2016