ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Nico Schottelius

# High speed NAT64 with P4



Master Thesis MA-2019-19
February 2019 to August 2019

Tutors: Alexander Dietmüller, Edgar Costa Molero, Tobias Bühler
Supervisor: Prof. Dr. Laurent Vanbever

**Abstract**

Due to the lack of IPv4 addresses, IPv6 deployements have recently gained in importance in the Internet. Several transition mechanism exist that allow translating IPv6 packets into IPv4 packets, thus enabling the coexistence and interoperability of both protocols.
This thesis describes an implementation of the transition mechanism NAT64 implemented in P4. Using the P4 programming language a software emulated switch was created that translates IPv4 to IPv6 and vice versa. Due to the target independence of P4 the same code can be compiled for and deployed to on the FPGA hardware platform “NetFPGA”.
Within the NetFPGA the NAT64 implementation achieves a stable throughput of 9.29 Gigabit/s and allows in network translations without a router. Due to the nature of P4, the implementation runs at line speed and thus with different hardware the same code can run potentially at much higher speeds, for instance on 100 Gbit/s switches.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In this chapter we give an introduction about the topic of the master thesis, the motivation and problemes that we address.

## 1.1 IPv4 exhaustion and IPv6 adoption

The Internet has almost completely run out of public IPv4 space. The 5 Regional Internet Registries (RIRs) report IPv4 exhaustion world wide ([33], [4], [20], [1], [5]) and LACNIC project complete exhaustion for 2020 (see figure 1.1).

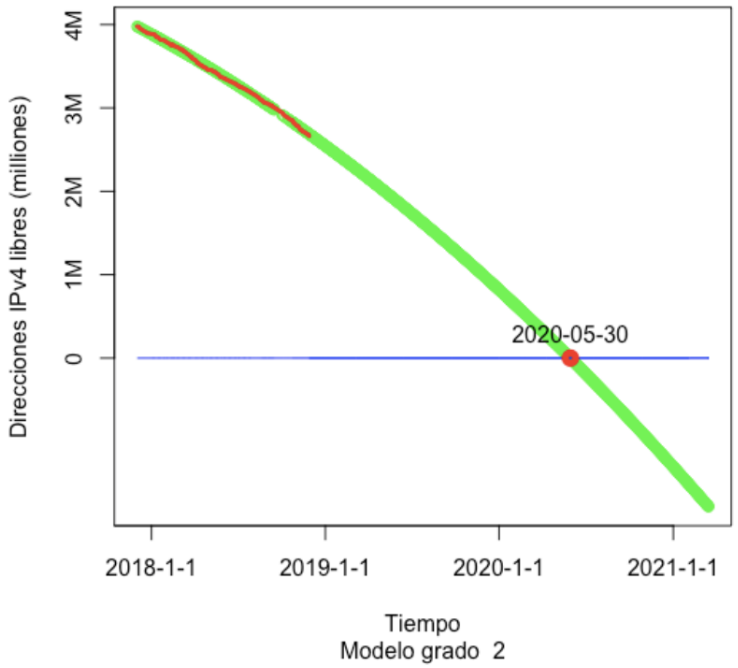


Figure 1.1: LACNIC Exhaustion projection, [20]

On the other hand IPv6 adoption grows significantly, with at least three countries (India, US, Belgium) surpassing 50% adoption ([2], [39]).
[11]). Traffic from Google users reaches almost 30% as of 2019-08-08 ([17], see figure 1.2). We conclude that IPv6 is a technology strongly gaining importance with the IPv4 depletion that is estimated to be world wide happening in the next years. Thus more devices will be using IPv6, while communication to legacy IPv4 devices still needs to be provided.

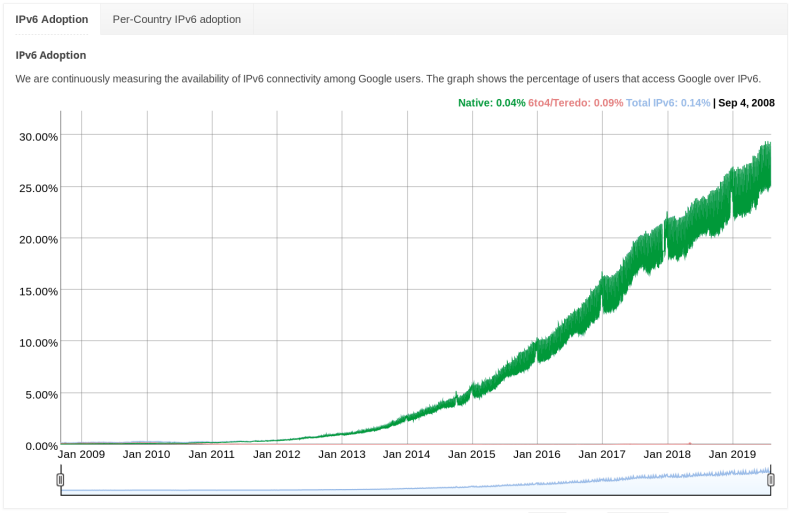


Figure 1.2: Google IPv6 Statistics, [17]

## 1.2 Motivation

IPv6 nodes and IPv4 nodes cannot directly connect to each other, because the protocols are incompatible to each other. To allow communication between different protocol nodes, several transition mechanism have been proposed [41], [27].
However installation and configuration of the transition mechanism usually require in depth knowledge about both protocols and require additional hardware to be added in the network.
In this thesis we show an in-network transition method based on NAT64 [6]. Compared to traditional NAT64 methods which require an extra device in the network, our proposed method is transparent to the user. This way neither the operator nor the end user has to configure extra devices. Figures 1.4 shows the standard NAT64 approach and 1.3 shows our solution. Cur-

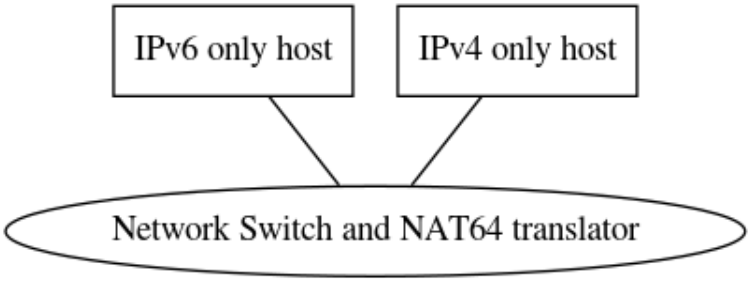


Figure 1.3: In Network NAT64 translation

rently network operators have to focus on two network stacks when designing networks: IPv6 and IPv4. While in a small scale setup this might not introduce significant complexity, figure 1.5 shows how the complexity quickly grows with the number of hosts. The in network solution does not only ease the installation and deployment of IPv6, but it also allows line speed translation, because it is compiled into target dependent low level code that can run in ASICs[25], FPGAs[24] or even in software [9]. Even on fast CPUs, software solutions like tayga [22] can be CPU bound and are not capabale of translating protocols at line speed.

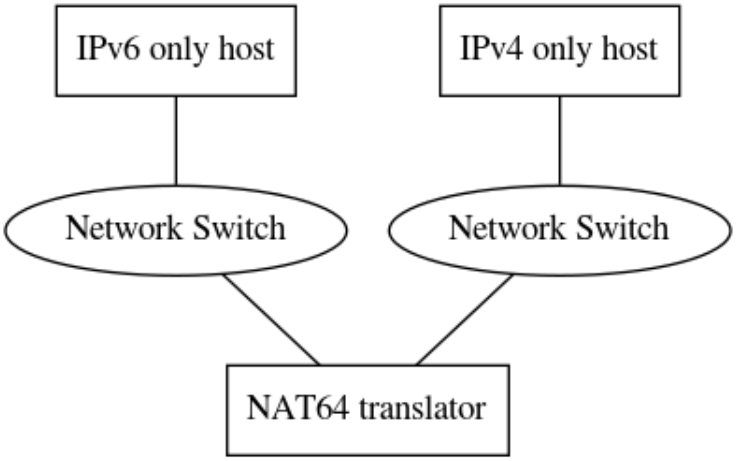


Figure 1.4: Standard NAT64 translation

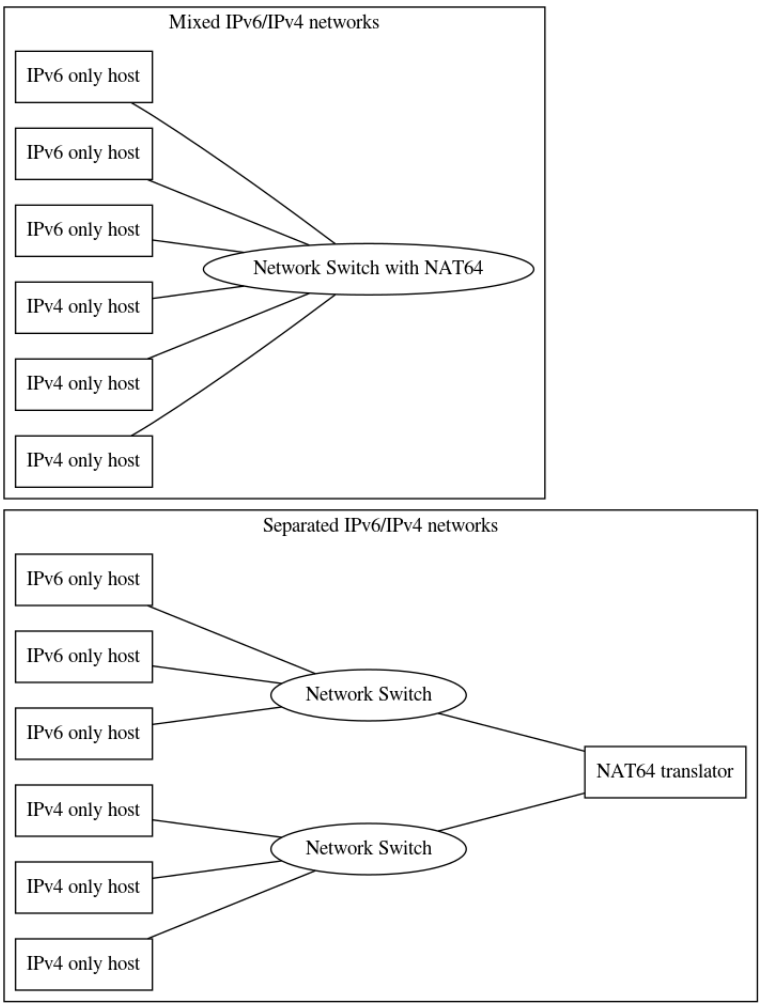


Figure 1.5: Differenent network design with in network NAT64 translation

# Chapter 2

# Background

In this chapter we describe the key technologies involved.

## 2.1 P4

P4 is a programming language designed to program inside network equipment. It's main features are protocol and target independence. The *protocol independence* refers to the separation of concerns in terms of language and protocols: P4 generally speaking operates on bits that are parsed and then accessible in the (self) defined structures, also called headers. The general flow can be seen in figure 2.1: a parser parses the incoming packet and prepares it for processing in the switching logic. Afterwards the packets is output and deparsing of the parsed data might follow. In the context of NAT64 this is a very important feature: while the parser will read and parse in the ingress pipeline one protocol (f.i. IPv6), the deparser will output a different protocol (f.i. IPv4). The *target independence* is the second very powerful feature of P4: it allows

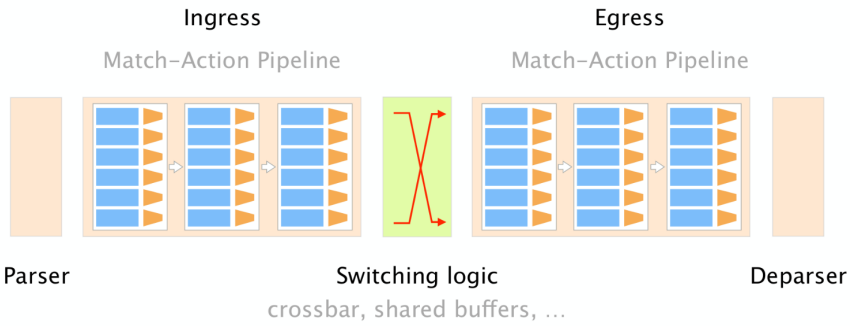


Figure 2.1: P4 protocol independence, [38]

code to be compiled to different targets. While in theory the P4 code should be completely target independent, in reality there are some modifications needed on a per-target basis and each target faces different restrictions. The challenges arising from this are discussed in section 5.4. As opposed to general purpose programming languages, P4 lacks some features, most notably loops. However within its constraints, P4 can guarantee operation at line speed, which general purpose programming languages cannot guarantee and also fail to achieve in reality (see section 4.6 for details).

## 2.2 IPv6, IPv4 and Ethernet

The first IPv6 RFC was published in 1998[14]. Both IPv4 and IPv4 operate on layer 3 of the OSI model. In this thesis we only consider transmission via Ethernet, which operates at layer 2. Inside the Ethernet frame a field named "type" specifies the higher level protocol identifier (0x0800 for IPv4 [19] and 0x86DD for IPv6 [13]. This is important, because Ethernet can only

carry either of the two protocols. The figures 2.3 and 2.2 show the packet headers of IPv4 and IPv6. The most notable differences between the two protocols for this thesis are:

- Different address lengths (32 vs 128 bit)
- Lack of checksum in IPv6
- Format of Pseudo headers (see section 2.5)

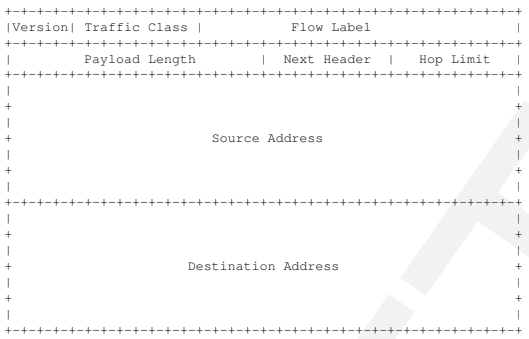


Figure 2.2: IPv6 Header, [14]

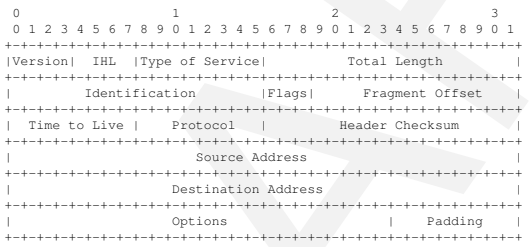


Figure 2.3: IPv4 Header, [30]

## 2.3 ARP and NDP, ICMP ICMP6- FIXME

Required for finding host. ARP who has NDP similar – add traces here
being able to answer to error messages MTU / pmtu

## 2.4 IPv6 Translation Mechanisms

While in this thesis the focus was in NAT64 as a translation mechanism, there are a variety of different approaches, some of which we would like to portray here.

### 2.4.1 Static NAT64

Static NAT64 describes static mappings between IPv6 and IPv4 addresses. This can be based on longest prefix matchings (LPM), ranges, bitmasks or individual entries.
NAT64 translations as described in this thesis modify multiple layers in the translation process:

- Ethernet (changing the type field)
- IPv4 / IPv6 (changing the protocol, changing the fields)
- TCP/UDP/ICMP/ICMP6 checksums

### 2.4.2 Stateful NAT64

Stateful NAT64 as defined in RFC6146[6] defines how to cretate 1:n mappings between IPv6 and IPv4 hosts. The motivation for stateful NAT64 is similar to stateful NAT44[**?**]: it allows translating many IPv6 addresses to one IPv4 address. While the opposite translation is also technically possible, the differences in address space don't justify its use in general.
Stateful NAT64 in particular uses information in higher level protocols to multiplex connections: Given one IPv4 address and the tcp protocol, outgoing connections from IPv6 hosts can dynamically mapped to the range of possible tcp ports. After a session is closed, the port can be reused again. The selection of mapped ports is usually based on the availability on the IPv4

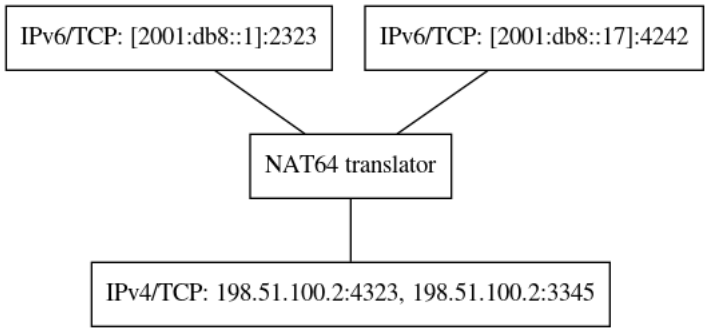


Figure 2.4: Stateful NAT64

side and not related to the original port. To support stateful NAT64, the translator needs to store the mapping in a table and purge entries regularly.
Stateful usually NAT64 uses information found in protocols at layer 4 like TCP [31] or UDP [28]. However it can also support ICMP [29] or ICMP6 [12].

### 2.4.3 Higher layer Protocol Dependent Translation

Further translation can be achieved by using information in higher level protocols like HTTP [15] or TLS [8]. Application proxies like nginx [26] use layer 7 protocol information to proxy towards backends. Within this proxying method, the underlying IP protocol can be changed from IPv6 to IPv4 and vice versa. However the requested hostname that is usually used for selecting the backend is encrypted in TLS 1.3 [32], which poses a challenge for implementations.
While protocol dependent translation has the highest amount of information to choose from for translation, complex parsers or even cryptographic methods are required for it. That reduces the opportunities of protocol dependent translation

### 2.4.4 Prefix based NAT - FIXME

Explain how it works in general **** RFC6052 - Defining well known prefix 64:ff9b::/96 - Defining embedding depending on prefix: /32../104 in 8 bit steps - Longer than /96: suffix support
- v4 to v6 / vice versa

## 2.5 Protocol Checksums

One challenge for translating IPv6-IPv4 are checksums of higher level protocols like TCP and UDP that incorporate information from the lower level protocols. The pseudo header for upper layer protocols for IPv6 is defined in RFC2460 [14] and shown in figure 2.5, the IPv4 pseudo header for TCP and UDP are defined in RFC768 and RFC793 and are shown in 2.6. When translating, the checksum fields in the higher protocols need to be adjusted. The checksums for TCP and UDP is calculated not only over the pseudo headers, but also contain the payload of the packet. This is important, because some targets (like the NetPFGA) do not allow to access the payload.


Figure 2.5: IPv6 Pseudo Header

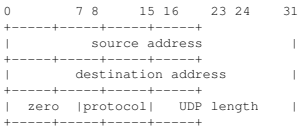


Figure 2.6: IPv4 Pseudo Header

## 2.6 Network Designs

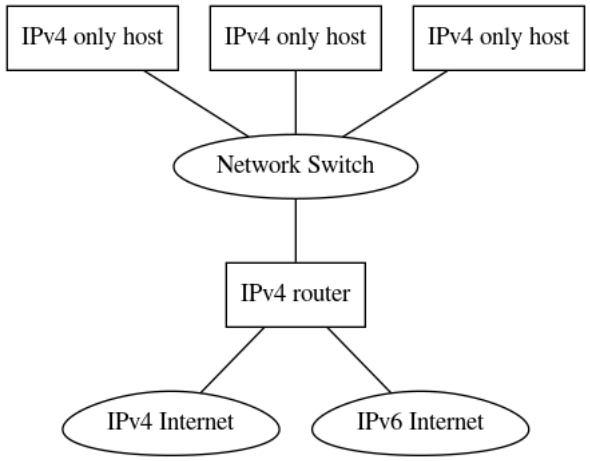


Figure 2.7: IPv4 only network

In relation to IPv6 and IPv4, there are in general three different network designs possible: The oldest form are IPv4 only networks (see figure 2.7. These networks consist of hosts that are either not configured for IPv6 or are even technically incapable of enabling the IPv6 protocol. These nodes are connected to an IPv4 router that is connected to the Internet. That router might be capable of translating IPv4 to IPv6 and vice versa. With the introduction of IPv6, hosts can have a separate IP stack active and in that configuration hosts are called “dualstack hosts” (see figure 2.8). Dualstack hosts are capabale of reaching both IPv6 and IPv4 hosts directly without the need of any translation mechanism.

The last possible network design is based on IPv6 only hosts, as shown in figure 2.9. While it is technically easy to disable IPv4, it seems that completely removing the IPv4 stack in current operating systems is not an easy task [37]. While the three network designs look similar, there are significant differences in operating them and limitations that are not easy to circumvent. In the following sections we describe the limitations and reason how a translation mechanism like our NAT64 implementation should be deployed.

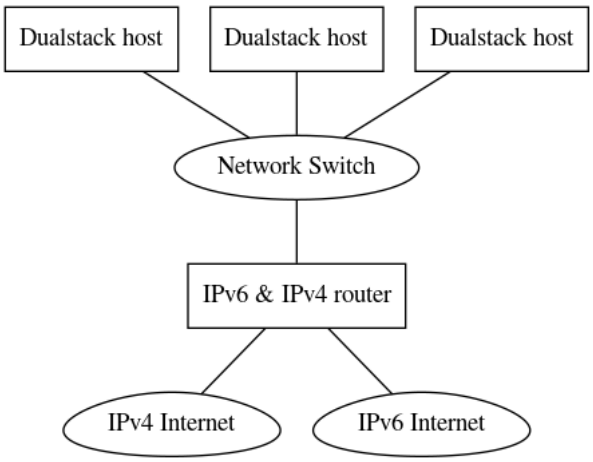


Figure 2.8: Dualstack network

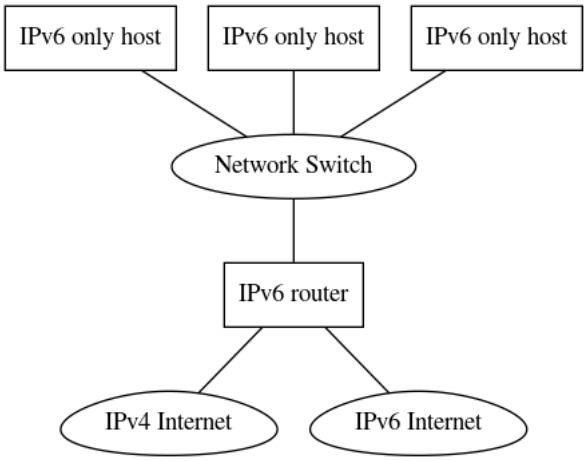


Figure 2.9: IPv6 only network

### 2.6.1 IPv4 only network limitations

As shown in figures 2.3 and 2.2 the IPv4 address size is 32 bit, while the IPv6 address size is 128 bit. Without an extension to the address space, there is no protocol independent mapping of IPv4 address to IPv6 (see section **??**) that can cover the whole IPv6 address space. Thus IPv4 only hosts can never address every host in the IPv6 Internet. While protocol dependent translations can try to minimise the impact, accessing all IPv6 addresses independent of the protcol is not possible.

### 2.6.2 Dualstack network maintenance

While dualstack hosts can address any host in either IPv6 or IPv4 networks, the deployment of dualstack hosts comes with a major disadvantage: all network configuration double. The required routing tables double, the firewall rules roughly double[1] and the number of network supporting systems (like DHCPv4, DHCPv6, router advertisement daemons, etc.) also roughly double. Additionally services that run on either IPv6 or IPv4 might need to be configured to run in dualstack mode as well and not every software might be capable of that. So while there is the instant benefit of not requiring any transition mechanism or translation method, we argue that the added complexity (and thus operational cost) of running dual stack networks can be significant.

[1] The rulesets even for identical policies in IPv6 and IPv4 networks are not identical, but similar. For this reason we state that roughly double the amount of firewall rules are required for the same policy to be applied.

### 2.6.3 IPv6 only networks

IPv6 only networks are in our opinion the best choice for long term deployments. The reasons for this are as follows: First of all hosts eventually will need to support IPv6 and secondly IPv6 hosts can address the whole 32 bit IPv4 Internet mapped in a single /96 IPv6 network. IPv6 only networks also allow the operators to focus on one IP stack.

# Chapter 3

# Design

Description of the theory/software/hardware that you designed. In this chapter we describe the architecture of our solution.

## 3.1 General

The high level design can be seen in figure 3.1: a P4 capable switch is running our code to provide NAT64 functionality. The P4 switch cannot manage its tables on it own and needs support for this from a controller. If only static table entries are required, the controller can also be omitted. However stateful NAT64 requires the use of a control to create session entries in the switch tables. The P4 switch can use any protocol to communicate with controller, as the connection to

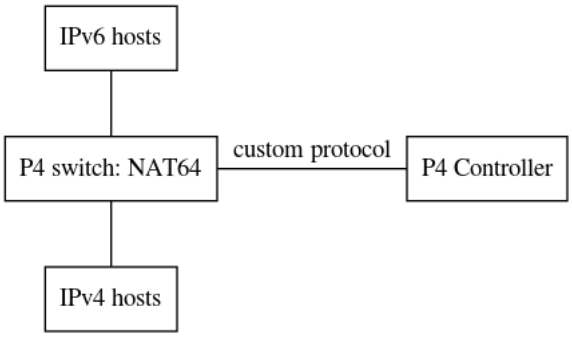


Figure 3.1: General Design

the controller is implemented as a separate ethernet port. The design allows our solution to be used as a standard NAT64 translation method or as an in network NAT64 translation (compare figures 1.3 and 1.4). The controller is implemented in python, the NAT64 solution is implemented in P4.

## 3.2 BMV2

Development of the thesis took place on a software emulated switch that is implemented using Open vSwitch [16] and the behavioral model [9]. The development followed closely the general design shown in section 3.1. Within the software emulation checksums can be computed with two different methods:

- Recalculating the checksum by inspecting headers and payload
- Calculating the difference between the translated headers

The BMV2 model is rather sophisticated and provides many standard features including checksumming over payload. This allows the BMV2 model to operate as a full featured host, including

advanced features like responding to ICMP6 Neighbor discovery requests [23] that include payload checksums. A typical code to create the checksum can be found in figure 3.2.

```
/* checksumming for icmp6_na_ns_option */
update_checksum_with_payload(meta.chk_icmp6_na_ns == 1,
{
        hdr.ipv6.src_addr,          /* 128 */
        hdr.ipv6.dst_addr,          /* 128 */
        meta.cast_length,           /* 32 */
        24w0,                       /* 24 0's */
        PROTO_ICMP6,                /* 8 */
        hdr.icmp6.type,             /* 8 */
        hdr.icmp6.code,             /* 8 */

        hdr.icmp6_na_ns.router,
        hdr.icmp6_na_ns.solicitated,
        hdr.icmp6_na_ns.override,
        hdr.icmp6_na_ns.reserved,
        hdr.icmp6_na_ns.target_addr,

        hdr.icmp6_option_link_layer_addr.type,
        hdr.icmp6_option_link_layer_addr.ll_length,
        hdr.icmp6_option_link_layer_addr.mac_addr
    },
    hdr.icmp6.checksum,
    HashAlgorithm.csum16
);
```

Figure 3.2: IPv4 Pseudo Header

## 3.3 NetFPGA

While the P4-NetFPGA project [24] allows compiling P4 to the NetPFGA, the design slightly varies. In particular, the NetFPGA P4 compiler does not support reading the payload. For this reason it also does not support creating the checksum based on the payload. To support checksum modifications in NAT64 on the NetFPGA, the checksum was calculated on the netpfga using differences between the IPv6 and IPv4 headers. Figure 3.3 shows an excerpt of the code used for calculating checksums in the netpfga. The checksums for IPv4, TCP, UDP and ICMP6

```
action v4sum() {
    bit<16> tmp = 0;

    tmp = tmp + (bit<16>) hdr.ipv4.src_addr[15:0];              // 16 bit
    tmp = tmp + (bit<16>) hdr.ipv4.src_addr[31:16];             // 16 bit
    tmp = tmp + (bit<16>) hdr.ipv4.dst_addr[15:0];              // 16 bit
    tmp = tmp + (bit<16>) hdr.ipv4.dst_addr[31:16];             // 16 bit

    tmp = tmp + (bit<16>) hdr.ipv4.totalLen -20;                // 16 bit
    tmp = tmp + (bit<16>) hdr.ipv4.protocol;                    // 8 bit

    meta.v4sum = ~tmp;
}

/* analogue code for v6sum skipped */

action delta_tcp_from_v6_to_v4()
{
    v6sum();
    v4sum();

    bit<17> tmp = (bit<17>) hdr.tcp.checksum + (bit<17>) meta.v4sum;
    if (tmp[16:16] == 1) {
        tmp = tmp + 1;
        tmp[16:16] = 0;
    }
    tmp = tmp + (bit<17>) (0xffff - meta.v6sum);
    if (tmp[16:16] == 1) {
        tmp = tmp + 1;
        tmp[16:16] = 0;
    }

    hdr.tcp.checksum = (bit<16>) tmp;
}
```

Figure 3.3: Calculating checksum based on header differences

are all based on the “Internet Checksum” ([30], [10]). Its calculation can be summarised as follows:

> The checksum field is the 16-bit one’s complement of the one’s complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.[1].

[1] Quote from Wikipedia[40].

As the calculation mainly depends on on (1-complement) sums, the checksums after translating the protocol can be corrected by subtracting the differences of the relevant fields. It is notable that not the full headers are used, but the pseudo headers (compare figures 2.5 and 2.6). To compensate the carry bit, our code uses 17 bit integers for correcting the carry.

# Chapter 4

# Results

## 4.1 General

Parser for all protocols (udp,tcp,icmp,icmp6)
BMV2: more feature rich, but software only solution NetFPGA: capabale of line speed Nat64, focused port on nat64
Both support EAMT as defined by RFC7757 [3].

## 4.2 BMV2

Responds to icmp, icmp6 ndp [23] arp
test framework openvswitch
Fully functional host Can compute checksums on its own.
focus on typical use cases of icmp, icmp6, the software implementation supports translating echo request and echo reply messages, but does not support all ICMP/ICMP6 translations that are defined in RFC6145[21].
Stateful : no automatic removal

## 4.3 Tayga

3gbit

## 4.4 Jool

## 4.5 NetFPGA

General result: limited NAT64 is working, however
No Payload checksumming - requires controller
Hash funktion in Arbeit
No NDP, no ARP - focused on key factors of NAT64 translation, other features can be supported by controller

## 4.6 NAT64 in Software

Tayga, Jool

## 4.7 Feature comparison

speed - sessions - eamt can act as host lpm tables ping ping6 support ndp controller support

## 4.8 todo - FIXME: remove

***** Dorth eher detailiertes Drawing
***** Längste Section!

# Chapter 5

# Conclusion

Sum up what you have done and recapitulate your key findings.

## 5.1 Software based NAT64

## 5.2 General

Many misleading

## 5.3 BMV2

## 5.4 P4

checksumming a frequent problem and helper
Many possibilities Protocol independent Easy architecture
Limitations in
if in action limitations
Limits if in actions
python2 only - unicode errors
IPv6: NDP: not easy to parse, as unknown number of following fields
No support for multiple LPM keys in a table, can be solved with ternary matching.
switch cannot be used in actions
if things don't work, often a checksum problem.
if frame checksum, then length of packet is broken

```
p4c -target bmv2 -arch v1model -std p4-16 "../p4src/static-mapping.p4" -o "/home/p4/master-thesis/p4src"
../p4src/static-mapping.p4(366): error: Program is not supported by this target, because table MyIngress.v6_networks has multiple successors
    table v6_networks {
          ^^^^^^^^^^^
```

```
ipaddress.ip_network("2001:db8:61::/64")
IPv6Network(u'3230:3031:3a64:6238:3a36:313a:3a2f:3634/128')

Fix:
from __future__ import unicode_literals
```

The tooling around P4 is still fragile, encountered many bugs in the development.[35]
or missing features ([34], [36])
Hitting expression bug

## 5.5 NetFGPA - all HERE

personal note here
MTU limitations: 1500 according to a private mail from Salvator Galea cambridge / uk

long compile process error prone compile process many dependencies lpm not supported! Netpfga live, Vivado SDNET xx k lines of supporting code
Vivado installation: silent errors, infinite loop, missing libncurses5
82k lines of code that are interdependent Many non critical error messages on the way Zero exit fatal errors
missing / spreaded documentation
tcpdump on local nfX doesn't work -> can only debug on other endpoint
First card: Writing tables fails hardware debug shows some errors but hardware debug on correct card also shows some error Debug ioctl errors when writing table entries
Output all ports -> port mapping documented only in a testdata script
hwtest: Execution fails due to missing djtgcfg
no payload accessq
Many workarounds
Table size 63, table size 64,
Table entries require arguments of all possible actions, not only used one.
Compile time hours
Silent errors
Unclear errors: broken board
Due to the very fragile nature of the build framework from the NetFPGA-Live repository,
Renaming VARIABLES in the definition of
Reproducibility:
hours for finding right output ports
packet size / annotation
Needed to debug internal parsing errors
3x rebooting to get card working with bitstream
Variable renaming breaks the compile process

## 5.6 Real world applications

Can be deployed using the netpfga. Or Barefoot or Arista.

## 5.7 Outlook

What are the consequences of your work for future work?
Different HW
Speed only limited to line speed. Could be running at 100 Gbit/s without modifications.
PMTU handling error cases
Our algorithm uses the IPv4-Compatible IPv6 Address[18] to embed IPv4 addresses. However RFC6052[7] defines different embeddings depending on the prefix size. A future version should support these schemes to be compatible to other implementations.
No fragmentation No address / mac learning

## 5.8 Closing words (NAME?)

While the port to NetPFGA was significantly more effort then expected, the learnings of the different layers were very much appreciated / liked
It was a

## 5.9 todo - FIXME: remove

```
***** Summary eher kurz
***** Outlook als subsection!
```

# Appendix A

# Resources and code repositories

The following sections describe how to acquire the resources to reproduce the test results. All compilations were made on Ubuntu 16.04 with kernels

- 4.15.0-54-generic (Supporting Desktop),
- 4.4.0-143-generic (BMV2 test VM)
- 4.15.0-55-generic (Desktop with NetFPGA

## A.1 Master Thesis

The master thesis including all self developed source code is available by git via

```
git clone git@gitlab.ethz.ch:nicosc/master-thesis.git
```

It can be browsed online on `https://gitlab.ethz.ch/nicosc/master-thesis`.

## A.2 Xilinx Toolchain

A prerequisite for building the NetFGPA source code is the installation of

- `Xilinx_SDNet_2018.2_1005_9`
- `Xilinx_Vivado_SDK_2018.2_0614_1954`

Both tools need to be installed to /opt/Xilinx/, as paths are hardcoded in various places.

## A.3 NetFGPA support scripts

To be able to compile P4 source code to the NetFPGA the collection of scripts, Makefiles and sample code of P4-NetFGPA is required.
The repository `git@github.com:NetFPGA/P4-NetFPGA-live.git` needs to be cloned to “projects” subdirectory as “P4-NetPFGA” of the user that wants to compile the source code. Access to the repository is granted after applying for access as described on `https://github.com/NetFPGA/P4-NetFPGA-public/wiki`. After that the variable `P4_PROJECT_NAME` in `/projects/P4-NetFPGA/tools/settings.sh` needs to be modified to read `export P4_PROJECT_NAME=minip4` instead of `export P4_PROJECT_NAME=switch_calc`.
Sample code for installation:

```
mkdir -p ~/projects
git clone git@github.com:NetFPGA/P4-NetFPGA-live.git P4-NetFPGA
sed -i 's/\(P4_PROJECT_NAME=\).*/\1minip4/' ~/projects/P4-NetFPGA/tools/settings.sh
```

Version **v1.3.1-46-g97d3aaa** of the P4-NetPFGA repository was used for creating the bitfiles of this project.

```
nico@nsg-System:~/projects/P4-NetFPGA$ git describe -always
v1.3.1-46-g97d3aaa
```

# Appendix B

# BMV2 environment and tests

All BMV2 based compilations were made with the following compiler:

```
p4@ubuntu:~$ p4c -version
p4c 0.5 (SHA: 5ae30ee)
```

The installation is based on the vagrant files that were provided in the “Advanced Topics in Communication Networks Fall 2018” course of ETHZ (`https://adv-net.ethz.ch/2018/`) and contains p4tools as well as all utilities that came with the vagrant installation.

## B.1 Diff based checksumming

For running the diff based checksum code, the following steps are necessary:
Compiling the p4 code and starting the switch:

```
cd ~/master-thesis/p4app
sudo p4run -config nat64-diff.json
```

Starting the controller which sets up the required table entries:

```
cd ~/master-thesis/p4app
sudo python ./controller.py -mode range_router
```

# Appendix C

# NetFPGA environment and tests

## C.1 NetFPGA Setup

Description of installation, commit of netpfga-live

## C.2 NetFPGA NAT64 Test cases

todo: add graphic of nsg <-> esprimo cabling

```
ip addr add 10.0.0.42/24 dev enp2s0f0

# Adding necessary ARP entries: for the virtual IPv4 address(es)
ip neigh add 10.0.0.6  lladdr f8:f2:1e:09:62:d1 dev enp2s0f0
ip neigh add 10.0.0.42 lladdr f8:f2:1e:09:62:d1 dev enp2s0f0
```

For all test cases the following network settings on esprimo:

```
12: enp2s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether f8:f2:1e:09:62:d0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.42/24 scope global enp2s0f0
       valid_lft forever preferred_lft forever
    inet6 fe80::faf2:1eff:fe09:62d0/64 scope link
       valid_lft forever preferred_lft forever
13: enp2s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether f8:f2:1e:09:62:d1 brd ff:ff:ff:ff:ff:ff
    inet6 2001:db8:42::42/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::faf2:1eff:fe09:62d1/64 scope link
       valid_lft forever preferred_lft forever
```

### C.2.1 Test 1: IPv4 egress settings work

Scenario: simple egress port setting for the IPv4 addresses
Step 1: getting correct values for table entries from python:

```
»> int(ipaddress.IPv4Address(u"10.0.0.42"))
167772202
»> int(ipaddress.IPv4Address(u"10.0.0.4"))
167772164
»>
```

Step 2: setting table netries

```
» table_cam_add_entry realmain_v4_networks_0 realmain.set_egress_port 167772202 => 16 0 0 0 0
fields = [(u'hit', 1), (u'action_run', 3), (u'out_port', 8), (u'out_port', 8), (u'mac_addr', 48), (u'task', 16), (u'table_id', 16)]
action_name = TopPipe.realmain.set_egress_port
field_vals = [1, '16', '0', '0', '0', '0']
CAM_Init_ValidateContext() - done
WROTE 0x44020250 = 0xa00002a
WROTE 0x44020280 = 0x0000
WROTE 0x44020284 = 0x0000
WROTE 0x44020288 = 0x10000000
WROTE 0x4402028c = 0x0001
READ  0x44020244 = 0x0001
WROTE 0x44020240 = 0x0001
READ  0x44020244 = 0x0001
READ  0x44020244 = 0x0001
success
» table_cam_add_entry realmain_v4_networks_0 realmain.set_egress_port 167772164 => 16 0 0 0 0
fields = [(u'hit', 1), (u'action_run', 3), (u'out_port', 8), (u'out_port', 8), (u'mac_addr', 48), (u'task', 16), (u'table_id', 16)]
action_name = TopPipe.realmain.set_egress_port
field_vals = [1, '16', '0', '0', '0', '0']
CAM_Init_ValidateContext() - done
WROTE 0x44020250 = 0xa000004
```

```
WROTE 0x44020280 = 0x0000
WROTE 0x44020284 = 0x0000
WROTE 0x44020288 = 0x10000000
WROTE 0x4402028c = 0x0001
READ  0x44020244 = 0x0001
WROTE 0x44020240 = 0x0001
READ  0x44020244 = 0x0001
READ  0x44020244 = 0x0001
success
»
```

Step 3: setting arp entries

```
root@ESPRIMO-P956:~# ip neigh add 10.0.0.6 lladdr f8:f2:1e:09:62:d1 dev enp2s0f0
root@ESPRIMO-P956:~# ip neigh add 10.0.0.4 lladdr f8:f2:1e:09:62:d1 dev enp2s0f0
```

Step 3: generating test packets, expecting 4 packets to show up on enp2s0f0:

```
nico@ESPRIMO-P956:~$ sudo tcpdump -ni enp2s0f0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp2s0f0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:49:28.200407 IP 10.0.0.42 > 10.0.0.4: ICMP echo request, id 4440, seq 1, length 64
10:49:28.200445 IP 10.0.0.42 > 10.0.0.4: ICMP echo request, id 4440, seq 1, length 64
10:49:29.222340 IP 10.0.0.42 > 10.0.0.4: ICMP echo request, id 4440, seq 2, length 64
10:49:29.222418 IP 10.0.0.42 > 10.0.0.4: ICMP echo request, id 4440, seq 2, length 64
```

Result: sucess

### C.2.2 Test 2: IPv6 egress

Similar to the IPv4 setting before, just for IPv6.
Step 1: getting IP address values

```
»> int(ipaddress.IPv6Address(u"2001:db8:42::4"))
42540766411362381960998550477184434180L
»> int(ipaddress.IPv6Address(u"2001:db8:42::6"))
42540766411362381960998550477184434182L
»> int(ipaddress.IPv6Address(u"2001:db8:42::42"))
42540766411362381960998550477184434242L
```

Step 2: setting table entries

```
» table_cam_add_entry realmain_v6_networks_0 realmain.set_egress_port 42540766411362381960998550477184434182 => 64  0 0 0 0
fields = [(u'hit', 1), (u'action_run', 3), (u'out_port', 8), (u'out_port', 8), (u'mac_addr', 48), (u'task', 16), (u'table_id', 16)]
action_name = TopPipe.realmain.set_egress_port
field_vals = [1, '64', '0', '0', '0', '0']
CAM_Init_ValidateContext() - done
WROTE 0x44020350 = 0x0006
WROTE 0x44020354 = 0x0000
WROTE 0x44020358 = 0x420000
WROTE 0x4402035c = 0x20010db8
WROTE 0x44020380 = 0x0000
WROTE 0x44020384 = 0x0000
WROTE 0x44020388 = 0x40000000
WROTE 0x4402038c = 0x0001
READ  0x44020344 = 0x0001
WROTE 0x44020340 = 0x0001
READ  0x44020344 = 0x0001
READ  0x44020344 = 0x0001
success
» table_cam_add_entry realmain_v6_networks_0 realmain.set_egress_port 42540766411362381960998550477184434242 => 64  0 0 0 0
fields = [(u'hit', 1), (u'action_run', 3), (u'out_port', 8), (u'out_port', 8), (u'mac_addr', 48), (u'task', 16), (u'table_id', 16)]
action_name = TopPipe.realmain.set_egress_port
field_vals = [1, '64', '0', '0', '0', '0']
CAM_Init_ValidateContext() - done
WROTE 0x44020350 = 0x0042
WROTE 0x44020354 = 0x0000
WROTE 0x44020358 = 0x420000
WROTE 0x4402035c = 0x20010db8
WROTE 0x44020380 = 0x0000
WROTE 0x44020384 = 0x0000
WROTE 0x44020388 = 0x40000000
WROTE 0x4402038c = 0x0001
READ  0x44020344 = 0x0001
WROTE 0x44020340 = 0x0001
READ  0x44020344 = 0x0001
READ  0x44020344 = 0x0001
success
»
```

Step 3: setting neighbor entries

```
nico@ESPRIMO-P956:~$ sudo ip -6 neigh add 2001:db8:42::6 lladdr f8:f2:1e:09:62:d0 dev enp2s0f1
nico@ESPRIMO-P956:~$ sudo ip -6 neigh add 2001:db8:42::4 lladdr f8:f2:1e:09:62:d0 dev enp2s0f1
```

Step 4: generating test packets

```
nico@ESPRIMO-P956:~$ ping6 -c2 2001:db8:42::6
PING 2001:db8:42::6(2001:db8:42::6) 56 data bytes


nico@ESPRIMO-P956:~$ sudo tcpdump -ni enp2s0f1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp2s0f1, link-type EN10MB (Ethernet), capture size 262144 bytes
11:30:17.287577 IP6 2001:db8:42::42 > 2001:db8:42::6: ICMP6, echo request, seq 1, length 64
11:30:17.287599 IP6 2001:db8:42::42 > 2001:db8:42::6: ICMP6, echo request, seq 1, length 64
11:30:18.310178 IP6 2001:db8:42::42 > 2001:db8:42::6: ICMP6, echo request, seq 2, length 64
11:30:18.310258 IP6 2001:db8:42::42 > 2001:db8:42::6: ICMP6, echo request, seq 2, length 64
```

Result: success, packet is seen twice.

### C.2.3 Test 3: NAT64

Additionally to the preparations done in test 1 and 2, the following steps were taken:
Step 1: getting IP address values via Python

```
>> int(ipaddress.IPv6Address(u"2001:db8:42::2a"))
42540766411362381960998550477184434218L

>> int(ipaddress.IPv6Address(u"2001:db8:42::"))
42540766411362381960998550477184434176L

>> int(ipaddress.IPv6Address(u"2001:db8:42::a00:2a"))
42540766411362381960998550477352206378

>> int(ipaddress.IPv4Address(u"10.0.0.0"))
167772160


>> int(ipaddress.IPv4Address(u"10.0.0.66"))
167772226
```

Add table entry for 2001:db8:42:2a to be translated to 10.0.0.42:

```
» table_cam_add_entry realmain_nat64_0  realmain.nat64_static 42540766411362381960998550477184434218 => 42540766411362381960998550477184434176 167772160 4254076641136238196
fields = [(u'hit', 1), (u'action_run', 3), (u'v6_src', 128), (u'v4_dst', 32), (u'nat64_prefix', 128), (u'table_id', 16)]
action_name = TopPipe.realmain.nat64_static
field_vals = [2, '42540766411362381960998550477184434176', '167772160', '42540766411362381960998550477184434176', '0']
CAM_Init_ValidateContext() - done
WROTE 0x44020050 = 0x002a
WROTE 0x44020054 = 0x0000
WROTE 0x44020058 = 0x420000
WROTE 0x4402005c = 0x20010db8
WROTE 0x44020080 = 0x0000
WROTE 0x44020084 = 0x0000
WROTE 0x44020088 = 0x0000
WROTE 0x4402008c = 0xdb80042
WROTE 0x44020090 = 0x2001
WROTE 0x44020094 = 0x0a00
WROTE 0x44020098 = 0x0000
WROTE 0x4402009c = 0x0000
WROTE 0x440200a0 = 0xdb80042
WROTE 0x440200a4 = 0x22001
READ  0x44020044 = 0x0001
WROTE 0x44020040 = 0x0001
READ  0x44020044 = 0x0001
READ  0x44020044 = 0x0001
success
»
```

Add table entry for 2001:db8:42::a00:2a to be translated to 10.0.0.66:

```
table_cam_add_entry realmain_nat64_0  realmain.nat64_static 42540766411362381960998550477352206378  => 42540766411362381960998550477184434176 167772160 4254076641136238196
```

Add table entry for 10.0.0.66 to be translated to 2001:db8:42:42:

```
» table_cam_add_entry realmain_nat46_0 realmain.nat46_static 167772226 => 42540766411362381960998550477184434176 167772160 42540766411362381960998550477184434176 0
fields = [(u'hit', 1), (u'action_run', 3), (u'v6_src', 128), (u'v4_dst', 32), (u'nat64_prefix', 128), (u'table_id', 16)]
action_name = TopPipe.realmain.nat46_static
field_vals = [2, '42540766411362381960998550477184434176', '167772160', '42540766411362381960998550477184434176', '0']
CAM_Init_ValidateContext() - done
WROTE 0x44020150 = 0xa000042
WROTE 0x44020180 = 0x0000
WROTE 0x44020184 = 0x0000
WROTE 0x44020188 = 0x0000
WROTE 0x4402018c = 0xdb80042
WROTE 0x44020190 = 0x2001
WROTE 0x44020194 = 0x0a00
WROTE 0x44020198 = 0x0000
WROTE 0x4402019c = 0x0000
WROTE 0x440201a0 = 0xdb80042
WROTE 0x440201a4 = 0x22001
READ  0x44020144 = 0x0001
WROTE 0x44020140 = 0x0001
READ  0x44020144 = 0x0001
READ  0x44020144 = 0x0001
success
»
```

Step 3: setting neighbor entries

```
sudo ip neigh add 10.0.0.66 lladdr f8:f2:1e:09:62:d1 dev enp2s0f0
sudo ip -6 neigh add 2001:db8:42::2a lladdr f8:f2:1e:09:62:d0 dev enp2s0f1
sudo ip -6 neighbor add 2001:db8:42::a00:2a lladdr f8:f2:1e:09:62:d0 dev enp2s0f1
```

Step 4: ping test should translate, but fail with wrong checksum:

# Appendix D

# NetFPGA Logs

Majority of the log files are stored inside the source code directory stored at “netpfga/logs”. It follows a selection of log files

## D.1 NetFPGA Flash Errors

Sometimes flashing bitfiles to the NetFPGA will fail. A random amount of reboots (1 to 3) and a random amount of reflashing will fix this problem.
Below can be found the log output from the flashing process.

```
nico@nsg-System:~/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/bitfiles$ sudo bash -c ". $HOME/master-thesis/netpfga/bashinit &
++ which vivado
+ xilinx_tool_path=/opt/Xilinx/Vivado/2018.2/bin/vivado
+ bitimage=minip4.bit
+ configWrites=config_writes.sh
+ '[' -z minip4.bit ']'
+ '[' -z config_writes.sh ']'
+ '[' /opt/Xilinx/Vivado/2018.2/bin/vivado == '' ']'
+ rmmod sume_riffa
+ xsct /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/tools/run_xsct.tcl -tclargs minip4.bit
rlwrap: warning: your $TERM is 'screen' but rlwrap couldn't find it in the terminfo database. Expect some problems.
RUN loading image file.
minip4.bit
100%   19MB   1.7MB/s  00:11
fpga configuration failed. DONE PIN is not HIGH
    invoked from within
"::tcf::eval -progress ::xsdb::print_progress {::tcf::cache_enter tcfchan#0 {tcf_cache_eval {process_tcf_actions_cache_client ::tcfclient#0::arg}}}"
    (procedure "::tcf::cache_eval_with_progress" line 2)
    invoked from within
"::tcf::cache_eval_with_progress [dict get $arg chan]  [list process_tcf_actions_cache_client $argvar] $progress"
    (procedure "process_tcf_actions" line 1)
    invoked from within
"process_tcf_actions $arg ::xsdb::print_progress"
    (procedure "fpga" line 430)
    invoked from within
"fpga -f $bitimage"
    (file "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/tools/run_xsct.tcl" line 33)

+ bash /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/tools/pci_rescan_run.sh
Check programming FPGA or Reboot machine !
+ rmmod sume_riffa
rmmod: ERROR: Module sume_riffa is not currently loaded
+ modprobe sume_riffa
+ ifconfig nf0 up
nf0: ERROR while getting interface flags: No such device
+ ifconfig nf1 up
nf1: ERROR while getting interface flags: No such device
+ ifconfig nf2 up
nf2: ERROR while getting interface flags: No such device
+ ifconfig nf3 up
nf3: ERROR while getting interface flags: No such device
+ bash config_writes.sh
```

## D.2 NetFPGA Flash Success

A successful flashing process also emits a couple of errors, however the message “fpga configuration failed. DONE PIN is not HIGH” and its succeeding lines are missing, as seen below.
After that in all cases a reboot is required; the PCI rescan in no tested case showed the nf devices.

```
nico@nsg-System:~$ cd $NF_DESIGN_DIR/bitfiles/
nico@nsg-System:~/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/bitfiles$ sudo bash -c ". $HOME/master-thesis/netpfga/bashinit &
++ which vivado
+ xilinx_tool_path=/opt/Xilinx/Vivado/2018.2/bin/vivado
+ bitimage=minip4.bit
+ configWrites=config_writes.sh
+ '[' -z minip4.bit ']'
```

```
+ '[' -z config_writes.sh ']'
+ '[' /opt/Xilinx/Vivado/2018.2/bin/vivado == '' ']'
+ rmmod sume_riffa
+ xsct /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/tools/run_xsct.tcl -tclargs minip4.bit
rlwrap: warning: your $TERM is 'xterm-256color' but rlwrap couldn't find it in the terminfo database. Expect some problems.
RUN loading image file.
minip4.bit
attempting to launch hw_server

****** Xilinx hw_server v2018.2
  **** Build date : Jun 14 2018-20:18:37
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: hw_server application started
INFO: Use Ctrl-C to exit hw_server application

INFO: To connect to this hw_server instance use url: TCP:127.0.0.1:3121

100%   19MB   1.7MB/s  00:11
+ bash /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/tools/pci_rescan_run.sh
Check programming FPGA or Reboot machine !
+ rmmod sume_riffa
rmmod: ERROR: Module sume_riffa is not currently loaded
+ modprobe sume_riffa
+ ifconfig nf0 up
nf0: ERROR while getting interface flags: No such device
+ ifconfig nf1 up
nf1: ERROR while getting interface flags: No such device
+ ifconfig nf2 up
nf2: ERROR while getting interface flags: No such device
+ ifconfig nf3 up
nf3: ERROR while getting interface flags: No such device
+ bash config_writes.sh
nico@nsg-System:~/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/bitfiles$
```

## D.3 NetFPGA Kernel module

After a successful flash, loading the kernel module will enable nf devices to appear in the operating system.

```
nico@nsg-System:~$ ip l
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 74:d0:2b:98:38:f6 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether f8:f2:1e:41:44:9c brd ff:ff:ff:ff:ff:ff
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether f8:f2:1e:41:44:9d brd ff:ff:ff:ff:ff:ff
5: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/none
nico@nsg-System:~$ ~/master-thesis/bin/build-load-drivers.sh
+ cd /home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0
+ sudo modprobe -r sume_riffa
+ make clean
make -C /lib/modules/4.15.0-55-generic/build M=/home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0 clean
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-55-generic'
  CLEAN   /home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0/.tmp_versions
  CLEAN   /home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-55-generic'
+ make all
make -C /lib/modules/4.15.0-55-generic/build M=/home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-55-generic'
  CC [M]  /home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0/sume_riffa.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0/sume_riffa.mod.o
  LD [M]  /home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0/sume_riffa.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-55-generic'
+ sudo make install
make -C /lib/modules/4.15.0-55-generic/build M=/home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-55-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-55-generic'
install -o root -g root -m 0755 -d /lib/modules/4.15.0-55-generic/extra/sume_riffa/
install -o root -g root -m 0755 sume_riffa.ko /lib/modules/4.15.0-55-generic/extra/sume_riffa/
depmod -a 4.15.0-55-generic
+ sudo modprobe sume_riffa
+ grep sume_riffa
+ lsmod
sume_riffa             28672  0
nico@nsg-System:~$
nico@nsg-System:~$ ip l
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 74:d0:2b:98:38:f6 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether f8:f2:1e:41:44:9c brd ff:ff:ff:ff:ff:ff
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether f8:f2:1e:41:44:9d brd ff:ff:ff:ff:ff:ff
5: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/none
6: nf0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:53:55:4d:45:00 brd ff:ff:ff:ff:ff:ff
7: nf1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:53:55:4d:45:01 brd ff:ff:ff:ff:ff:ff
8: nf2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:53:55:4d:45:02 brd ff:ff:ff:ff:ff:ff
9: nf3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 02:53:55:4d:45:03 brd ff:ff:ff:ff:ff:ff
nico@nsg-System:~$
```

## D.4 NetFPGA misses packets on nf*

While the nf devices appear in the operating system, packets emitted by the netpfga cannot be sniffed on the nf interfaces directly. Instead one has to sniff packets on a physical network card that is connected to the specific output port.

# Appendix E

# Benchmark Logs

## E.1 iperf

Omitting startup time

## E.2 General

MTU setting to 1500, as netpfga doesn't support jumbo frames
iperf3, iperf 3.0.11
50 parallel = 2x 10040 parallel = 10030 parallel = 70
Turning back on checksum offloading (see below)
30 parallel = 70

```
root@ESPRIMO-P956:~# ethtool -K enp2s0f0 tx on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: on
        tx-checksum-ip-generic: on
        tx-checksum-sctp: on
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp6-segmentation: on
root@ESPRIMO-P956:~#
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 tx on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: on
        tx-checksum-ip-generic: on
        tx-checksum-sctp: on
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp6-segmentation: on
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 rx on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~#
```

Results into

```
root@ESPRIMO-P956:~# ethtool -k enp2s0f0
Features for enp2s0f0:
Cannot get device udp-fragmentation-offload settings: Operation not supported
rx-checksumming: on
tx-checksumming: on
        tx-checksum-ipv4: off [fixed]
        tx-checksum-ip-generic: on
        tx-checksum-ipv6: off [fixed]
        tx-checksum-fcoe-crc: on [fixed]
        tx-checksum-sctp: on
scatter-gather: on
        tx-scatter-gather: on
        tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp-ecn-segmentation: off [fixed]
        tx-tcp-mangleid-segmentation: off
        tx-tcp6-segmentation: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
rx-vlan-offload: on
tx-vlan-offload: on
ntuple-filters: off
receive-hashing: on
highdma: on [fixed]
rx-vlan-filter: on
vlan-challenged: off [fixed]
```

```
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: on [fixed]
tx-gre-segmentation: on
tx-gre-csum-segmentation: on
tx-ipxip4-segmentation: on
tx-ipxip6-segmentation: on
tx-udp_tnl-segmentation: on
tx-udp_tnl-csum-segmentation: on
tx-gso-partial: on
tx-sctp-segmentation: off [fixed]
tx-esp-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off
loopback: off [fixed]
rx-fcs: off [fixed]
rx-all: off
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off
hw-tc-offload: off
esp-hw-offload: off [fixed]
esp-tx-csum-hw-offload: off [fixed]
rx-udp_tunnel-port-offload: off
root@ESPRIMO-P956:~# ethtool -k enp2s0f1
Features for enp2s0f1:
Cannot get device udp-fragmentation-offload settings: Operation not supported
rx-checksumming: on
tx-checksumming: on
        tx-checksum-ipv4: off [fixed]
        tx-checksum-ip-generic: on
        tx-checksum-ipv6: off [fixed]
        tx-checksum-fcoe-crc: on [fixed]
        tx-checksum-sctp: on
scatter-gather: on
        tx-scatter-gather: on
        tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp-ecn-segmentation: off [fixed]
        tx-tcp-mangleid-segmentation: off
        tx-tcp6-segmentation: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
rx-vlan-offload: on
tx-vlan-offload: on
ntuple-filters: off
receive-hashing: on
highdma: on [fixed]
rx-vlan-filter: on
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: on [fixed]
tx-gre-segmentation: on
tx-gre-csum-segmentation: on
tx-ipxip4-segmentation: on
tx-ipxip6-segmentation: on
tx-udp_tnl-segmentation: on
tx-udp_tnl-csum-segmentation: on
tx-gso-partial: on
tx-sctp-segmentation: off [fixed]
tx-esp-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off
loopback: off [fixed]
rx-fcs: off [fixed]
rx-all: off
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off
hw-tc-offload: off
esp-hw-offload: off [fixed]
esp-tx-csum-hw-offload: off [fixed]
rx-udp_tunnel-port-offload: off
root@ESPRIMO-P956:~#
```

## E.3 NetFPGA

### iperf3-tcp-listening-v4 connected by v6

```
nico@ESPRIMO-P956:~$ iperf3 -p 2345 -4 -B 10.0.0.42 -s
-----------------------------------------
Server listening on 2345
-----------------------------------------
Accepted connection from 10.0.0.66, port 50900
[  5] local 10.0.0.42 port 2345 connected to 10.0.0.66 port 50902
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec   693 MBytes  5.81 Gbits/sec
[  5]   1.00-2.00   sec   645 MBytes  5.41 Gbits/sec
[  5]   2.00-3.00   sec   644 MBytes  5.40 Gbits/sec
[  5]   3.00-4.00   sec   868 MBytes  7.28 Gbits/sec
[  5]   4.00-5.00   sec   853 MBytes  7.16 Gbits/sec
[  5]   5.00-6.00   sec   913 MBytes  7.66 Gbits/sec
[  5]   6.00-7.00   sec   774 MBytes  6.49 Gbits/sec
[  5]   7.00-8.00   sec   641 MBytes  5.38 Gbits/sec
[  5]   8.00-9.00   sec   911 MBytes  7.64 Gbits/sec
[  5]   9.00-10.00  sec   733 MBytes  6.15 Gbits/sec
[  5]  10.00-10.04  sec  25.8 MBytes  5.38 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.04  sec  7.52 GBytes  6.43 Gbits/sec   14             sender
[  5]   0.00-10.04  sec  7.52 GBytes  6.43 Gbits/sec                  receiver
```

```
-----------------------------------------
Server listening on 2345
-----------------------------------------

nico@ESPRIMO-P956:~$ iperf3 -6 -p 2345 -c 2001:db8:42::a00:2a
Connecting to host 2001:db8:42::a00:2a, port 2345
[  4] local 2001:db8:42::42 port 50902 connected to 2001:db8:42::a00:2a port 2345
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   719 MBytes  6.03 Gbits/sec   10    449 KBytes
[  4]   1.00-2.00   sec   645 MBytes  5.41 Gbits/sec    0    449 KBytes
[  4]   2.00-3.00   sec   644 MBytes  5.40 Gbits/sec    0    449 KBytes
[  4]   3.00-4.00   sec   878 MBytes  7.36 Gbits/sec    0    449 KBytes
[  4]   4.00-5.00   sec   859 MBytes  7.20 Gbits/sec    0    449 KBytes
[  4]   5.00-6.00   sec   910 MBytes  7.64 Gbits/sec    0    449 KBytes
[  4]   6.00-7.00   sec   758 MBytes  6.36 Gbits/sec    0    449 KBytes
[  4]   7.00-8.00   sec   658 MBytes  5.52 Gbits/sec    0    449 KBytes
[  4]   8.00-9.00   sec   906 MBytes  7.60 Gbits/sec    4    449 KBytes
[  4]   9.00-10.00  sec   724 MBytes  6.07 Gbits/sec    0    449 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  7.52 GBytes  6.46 Gbits/sec   14             sender
[  4]   0.00-10.00  sec  7.52 GBytes  6.46 Gbits/sec                  receiver

iperf Done.
nico@ESPRIMO-P956:~$
```

### listening on v6, connecting from v4:

```
nico@ESPRIMO-P956:~$ iperf3 -p 2345 -6 -B 2001:db8:42::42 -s
-----------------------------------------
Server listening on 2345
-----------------------------------------
Accepted connection from 2001:db8:42::a00:2a, port 47520
[  5] local 2001:db8:42::42 port 2345 connected to 2001:db8:42::a00:2a port 47522
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  1.02 GBytes  8.73 Gbits/sec
[  5]   1.00-2.00   sec   879 MBytes  7.38 Gbits/sec
[  5]   2.00-3.00   sec   859 MBytes  7.20 Gbits/sec
[  5]   3.00-4.00   sec  1.02 GBytes  8.78 Gbits/sec
[  5]   4.00-5.00   sec  1.04 GBytes  8.89 Gbits/sec
[  5]   5.00-6.00   sec  1.05 GBytes  9.00 Gbits/sec
[  5]   6.00-7.00   sec  1.03 GBytes  8.89 Gbits/sec
[  5]   7.00-8.00   sec  1.04 GBytes  8.91 Gbits/sec
[  5]   8.00-9.00   sec  1.03 GBytes  8.84 Gbits/sec
[  5]   9.00-10.00  sec   953 MBytes  7.99 Gbits/sec
[  5]  10.00-10.04  sec  38.6 MBytes  7.81 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.04  sec  9.89 GBytes  8.46 Gbits/sec  151             sender
[  5]   0.00-10.04  sec  9.89 GBytes  8.46 Gbits/sec                  receiver
-----------------------------------------
Server listening on 2345
-----------------------------------------

nico@ESPRIMO-P956:~$ iperf3 -4 -p 2345 -c 10.0.0.66
Connecting to host 10.0.0.66, port 2345
[  4] local 10.0.0.42 port 47522 connected to 10.0.0.66 port 2345
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  1.06 GBytes  9.10 Gbits/sec   53    208 KBytes
[  4]   1.00-2.00   sec   867 MBytes  7.27 Gbits/sec    6    379 KBytes
[  4]   2.00-3.00   sec   870 MBytes  7.29 Gbits/sec    0    423 KBytes
[  4]   3.00-4.00   sec  1.02 GBytes  8.77 Gbits/sec   37    364 KBytes
[  4]   4.00-5.00   sec  1.04 GBytes  8.91 Gbits/sec    1    450 KBytes
[  4]   5.00-6.00   sec  1.05 GBytes  8.98 Gbits/sec    0    462 KBytes
[  4]   6.00-7.00   sec  1.04 GBytes  8.92 Gbits/sec   30    324 KBytes
[  4]   7.00-8.00   sec  1.04 GBytes  8.88 Gbits/sec    0    471 KBytes
[  4]   8.00-9.00   sec  1.03 GBytes  8.86 Gbits/sec   10    452 KBytes
[  4]   9.00-10.00  sec   947 MBytes  7.94 Gbits/sec   14    409 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  9.89 GBytes  8.49 Gbits/sec  151             sender
[  4]   0.00-10.00  sec  9.89 GBytes  8.49 Gbits/sec                  receiver

iperf Done.
nico@ESPRIMO-P956:~$
```

## E.4 Tayga

```
ii  tayga                                  0.9.2-6                                amd64        userspace stateless NAT64
```

### Setting up IPv4 networking

```
[15:12] nsg-System:~# ip addr add 10.0.0.77/24 dev eth1
[15:12] nsg-System:~# ip l s eth1 up

nico@ESPRIMO-P956:~$ ~/master-thesis/bin/init_ipv4_esprimo.sh
nico@ESPRIMO-P956:~$ cat ~/master-thesis/bin/init_ipv4_esprimo.sh
#!/bin/sh

sudo ip addr add 10.0.0.42/24 dev enp2s0f0
sudo ip link set enp2s0f0 up

nico@ESPRIMO-P956:~$ sudo ip route add 10.0.1.0/24 via 10.0.0.77
```

### Verify networking works:

```
[15:12] nsg-System:~# ping 10.0.0.42
PING 10.0.0.42 (10.0.0.42) 56(84) bytes of data.
64 bytes from 10.0.0.42: icmp_seq=1 ttl=64 time=0.304 ms
64 bytes from 10.0.0.42: icmp_seq=2 ttl=64 time=0.097 ms
^C
-- 10.0.0.42 ping statistics --
```

```
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.097/0.200/0.304/0.104 ms
[15:12] nsg-System:~#
```

### Setting up IPv6 networking

```
nico@ESPRIMO-P956:~$ ip addr show dev enp2s0f1
13: enp2s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether f8:f2:1e:09:62:d1 brd ff:ff:ff:ff:ff:ff
    inet6 2001:db8:42::42/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::faf2:1eff:fe09:62d1/64 scope link
       valid_lft forever preferred_lft forever
nico@ESPRIMO-P956:~$ sudo ip route add 2001:db8:23::/96 via 2001:db8:42::77

[15:12] nsg-System:~# ip addr add 2001:db8:42::77/64 dev eth2
[15:15] nsg-System:~# ip link set eth2 up
```

### Verify IPv6 networking works:

```
nico@ESPRIMO-P956:~$ ping6 -c2 2001:db8:42::77
PING 2001:db8:42::77(2001:db8:42::77) 56 data bytes
64 bytes from 2001:db8:42::77: icmp_seq=1 ttl=64 time=0.169 ms
64 bytes from 2001:db8:42::77: icmp_seq=2 ttl=64 time=0.153 ms

-- 2001:db8:42::77 ping statistics --
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.153/0.161/0.169/0.008 ms
nico@ESPRIMO-P956:~$
```

### Enabling IPv6 and IPv4 forwarding:

```
[15:16] nsg-System:~# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1

[15:20] nsg-System:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

### Testing NAT64 in tayga

```
nico@ESPRIMO-P956:~$ ping -c2 10.0.1.42
PING 10.0.1.42 (10.0.1.42) 56(84) bytes of data.
64 bytes from 10.0.1.42: icmp_seq=1 ttl=61 time=0.356 ms
64 bytes from 10.0.1.42: icmp_seq=2 ttl=61 time=0.410 ms

-- 10.0.1.42 ping statistics --
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.356/0.383/0.410/0.027 ms
nico@ESPRIMO-P956:~$

nico@ESPRIMO-P956:~$ sudo tcpdump -ni enp2s0f1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp2s0f1, link-type EN10MB (Ethernet), capture size 262144 bytes
15:21:39.851057 IP6 2001:db8:23::a00:2a > 2001:db8:42::42: ICMP6, echo request, seq 1, length 64
15:21:39.851124 IP6 2001:db8:42::42 > 2001:db8:23::a00:2a: ICMP6, echo reply, seq 1, length 64
15:21:40.870448 IP6 2001:db8:23::a00:2a > 2001:db8:42::42: ICMP6, echo request, seq 2, length 64
15:21:40.870507 IP6 2001:db8:42::42 > 2001:db8:23::a00:2a: ICMP6, echo reply, seq 2, length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
nico@ESPRIMO-P956:~$
```

### Testing NAT64 (v6 to v4)

```
nico@ESPRIMO-P956:~$ ping6 -c2 2001:db8:23::a00:2a
PING 2001:db8:23::a00:2a(2001:db8:23::a00:2a) 56 data bytes
64 bytes from 2001:db8:23::a00:2a: icmp_seq=1 ttl=61 time=0.240 ms
64 bytes from 2001:db8:23::a00:2a: icmp_seq=2 ttl=61 time=0.400 ms

-- 2001:db8:23::a00:2a ping statistics --
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.240/0.320/0.400/0.080 ms
nico@ESPRIMO-P956:~$
```

## E.4.1 Tayga/TCP

Tayga running at 100
v4->v6 tcp delivering 3.36 gbit/s at P1 3.30 Gbit/s at P20 3.11 gbit/s at P50
v6->v4 tcp P1: 3.02 Gbit/s P20: 3.28 gbit/s P50: 2.85 gbit/s
Commands:

```
Server always: iperf3 -6 -p 2345 -B 2001:db8:42::42 -s  | tee iperf-tayga-v4tov6server-P50


nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P1 -c 10.0.1.42 -T taygav4tov6tcpP1 | tee iperf-tayga-v4tov6server-client

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P20 -c 10.0.1.42 -T taygav4tov6tcpP20 | tee iperf-tayga-v4tov6server-client-P20

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P50 -c 10.0.1.42 -T taygav4tov6tcpP50 | tee iperf-tayga-v4tov6server-client-P50


nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-tayga-v6tov4-server-P1
```

Testing v6->v4

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-tayga-v6tov4-server-P20

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P1 -c 2001:db8:23::a00:2a  -T taygav6tov4tcpP1 | tee iperf-tayga-v6tov4-client-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P20 -c 2001:db8:23::a00:2a  -T taygav6tov4tcpP20 | tee iperf-tayga-v6tov4-client-P20

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P50 -c 2001:db8:23::a00:2a  -T taygav6tov4tcpP50 | tee iperf-tayga-v6tov4-client-P50
```

UDP v6->v4, again 100
P1: 5.81 gbit/s P20: 9.40 gbit/s P50: 19.6 gbits/sec
On the line only ca. 3600 mbit/s seen

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-tayga-v6tov4-server-udp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P1 -u -b10000m -c 2001:db8:23::a00:2a  -T taygav6tov4tcpP50 | tee iperf-tayga-v6tov4-client-udp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P50 -u -b10000m -c 2001:db8:23::a00:2a  -T taygav6tov4tcpP50 | tee iperf-tayga-v6tov4-client-udp-P50
```

Messsages from server:

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-tayga-v6tov4-server-udp-P1
iperf3: OUT OF ORDER - incoming packet = 198902 and received packet = 198904 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 441615 and received packet = 441617 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 441616 and received packet = 441618 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 567495 and received packet = 567501 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 567496 and received packet = 567501 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 567497 and received packet = 567501 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 567499 and received packet = 567503 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 567500 and received packet = 567503 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 567502 and received packet = 567503 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 631160 and received packet = 631164 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 631161 and received packet = 631164 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 631162 and received packet = 631165 AND SP = 5
iperf3: OUT OF ORDER - incoming packet = 631163 and received packet = 631165 AND SP = 5
```

UDP v4->v6, again 100
P1: 8.26 gbit/s [atop:  2500 Mbit/s per direction] P20: 9.92 Gbits/sec [atop:  2500 Mbit/s per direction] P50: 19.3 gbit/s [atop:  2500 Mbit/s per direction]

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -B 2001:db8:42::42 -s  | tee iperf-tayga-v4tov6-server-udp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P1 -u -b0 -c 10.0.1.42 -T taygav4tov6udpP1 | tee iperf-tayga-v4tov6server-client-udp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P20 -u -b0 -c 10.0.1.42 -T taygav4tov6udpP20 | tee iperf-tayga-v4tov6server-client-udp-P20

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P50 -u -b0 -c 10.0.1.42 -T taygav4tov6udpP50 | tee iperf-tayga-v4tov6server-client-udp-P50
```

# E.5 Jool

## E.5.1 Jool Setup

Installation of 4.0.1 from `https://www.jool.mx/en/download.html`.

```
nico@nsg-System:~$ wget https://github.com/NICMx/Jool/releases/download/v4.0.1/jool_4.0.1.tar.gz
nico@nsg-System:~$ tar xvfz jool_4.0.1.tar.gz
nico@nsg-System:~$ cd jool-4.0.1/
nico@nsg-System:~/jool-4.0.1$ sudo  apt install linux-headers-$(uname -r)
nico@nsg-System:~/jool-4.0.1$ sudo  apt install libnl-genl-3-dev
```

xtables cannot be found:

```
nico@nsg-System:~/jool-4.0.1$ sudo apt install libxtables-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package libxtables-dev
nico@nsg-System:~/jool-4.0.1$
```

Does not compile without:

```
checking for library containing argp_parse... none required
checking for pkg-config... /usr/bin/pkg-config
checking pkg-config is at least version 0.9.0... yes
checking for LIBNLGENL3... yes
checking for XTABLES... no
configure: error: Package requirements (xtables) were not met:

No package 'xtables' found

Consider adjusting the PKG_CONFIG_PATH environment variable if you
installed software in a non-standard prefix.

Alternatively, you may set the environment variables XTABLES_CFLAGS
and XTABLES_LIBS to avoid the need to call pkg-config.
See the pkg-config man page for more details.
nico@nsg-System:~/jool-4.0.1$
```

Trying different package:

```
nico@nsg-System:~/jool-4.0.1$ sudo apt install iptables-dev
```

Compiles!

```
nico@nsg-System:~/jool-4.0.1$ sudo make install
```

### E.5.2 Jool Configuration

Loading module:

```
nico@nsg-System:~/jool-4.0.1$ sudo modprobe jool_siit
```

enabling forwarding:

```
sysctl -w net.ipv4.conf.all.forwarding=1
sysctl -w net.ipv6.conf.all.forwarding=1
```

Mapping configuration:

```
nico@nsg-System:~/jool-4.0.1$ sudo jool_siit instance add example -iptables -pool6 2001:db8:23::/96

nico@nsg-System:~/jool-4.0.1$ sudo ip6tables -t mangle -A PREROUTING \
                              -s 2001:db8:42::/64 -d 2001:db8:23::/96 -j JOOL_SIIT -instance example

nico@nsg-System:~/jool-4.0.1$ sudo iptables -t mangle -A PREROUTING \
                              -s 10.0.0.0/24 -j JOOL_SIIT -instance example
```

Debugging:

```
[16:39] nsg-System:~# lsmod| grep jool
jool_siit             147456  2
x_tables               40960  5 jool_siit,ip6_tables,ip_tables,ip6table_mangle,iptable_mangle
[16:39] nsg-System:~#

[16:41] nsg-System:~#  jool_siit -i  example stats display -explain
JSTAT64_DST: 276
Translations cancelled: IPv6 packet's destination address did not match pool6 nor any EAMT entries, or the resulting address was blacklist4ed.
```

Try 2 w/ eamt:

```
[16:53] nsg-System:~# modprobe jool_siit
[16:54] nsg-System:~# jool_siit instance add "example" -iptables
[16:54] nsg-System:~# jool_siit -i example eamt add 2001:db8:42::/120 10.0.1.0/24
[16:55] nsg-System:~# jool_siit -i example eamt add 2001:db8:23::/120 10.0.0.0/24
[16:57] nsg-System:~# ip6tables -t mangle -A PREROUTING -s 2001:db8:42::/120 -d 2001:db8:23::/120 -j JOOL_SIIT -instance example
[16:57] nsg-System:~# iptables -t mangle -A PREROUTING -s 10.0.0.0/24 -d 10.0.1.0/24 -j JOOL_SIIT -instance example
[16:57] nsg-System:~#
```

Testing NAT64:

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ ping6 2001:db8:23::2a
PING 2001:db8:23::2a(2001:db8:23::2a) 56 data bytes
64 bytes from 2001:db8:23::2a: icmp_seq=1 ttl=63 time=0.199 ms
64 bytes from 2001:db8:23::2a: icmp_seq=2 ttl=63 time=0.282 ms
64 bytes from 2001:db8:23::2a: icmp_seq=3 ttl=63 time=0.186 ms
^C
-- 2001:db8:23::2a ping statistics --
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.186/0.222/0.282/0.044 ms
nico@ESPRIMO-P956:~/master-thesis/iperf$ ping 10.0.1.66
PING 10.0.1.66 (10.0.1.66) 56(84) bytes of data.
64 bytes from 10.0.1.66: icmp_seq=1 ttl=63 time=0.218 ms
64 bytes from 10.0.1.66: icmp_seq=2 ttl=63 time=0.281 ms
64 bytes from 10.0.1.66: icmp_seq=3 ttl=63 time=0.280 ms
^C
-- 10.0.1.66 ping statistics --
3 packets transmitted, 3 received, 0% packet loss, time 2051ms
rtt min/avg/max/mdev = 0.218/0.259/0.281/0.034 ms
nico@ESPRIMO-P956:~/master-thesis/iperf$
```

### E.5.3 Jool Benchmarks

v4->v6 tcp
P1: 8.24 gbit/s no cpu load visible P20: 8.26 gbit/s iperf 42 + 10P50: 8.29 gbit/s
v6->v4 tcp
P1: 8.22 P20: 8.22 15/60P50: 8.23 iperf: 73/16
Commands:

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -B 2001:db8:42::42 -s  | tee iperf-jool-v4tov6-server-tcp-P50

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P1  -c 10.0.1.66 | tee iperf-jool-v4tov6-client-tcp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P20  -c 10.0.1.66 | tee iperf-jool-v4tov6-client-tcp-P20
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P50  -c 10.0.1.66 | tee iperf-jool-v4tov6-client-tcp-P50

Other way:

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-jool-v6tov4-server-tcp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P1  -c 2001:db8:23::2a | tee iperf-jool-v6tov4-client-tcp-P1

...

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P1 -b0 -u  -c 2001:db8:23::2a | tee iperf-jool-v6tov4-client-tcp-P1
```

v4->v6 udp
P1: 4.46 iperf 30P20: 18.8 iperf 100P50: 22.8 iperf 100

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -B 2001:db8:42::42 -s  | tee iperf-jool-v4tov6-server-udp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P1  -c 10.0.1.66 -u -b0 | tee iperf-jool-v4tov6-client-udp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P20  -c 10.0.1.66 -u -b0 | tee iperf-jool-v4tov6-client-udp-P20
```

v6->v4 udp
P1: 6.67 gbit/s iperf 50/50P20: 16.8 nat64: iperf: ? 100P50: 20.5 Gbits/sec nat64: 100

Turning off offloading, redoing tcp:

```
root@ESPRIMO-P956:~# ethtool -K enp2s0f0 gso off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~# ethtool -K enp2s0f0 rx off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~# ethtool -K enp2s0f0 tx off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: off
        tx-checksum-ip-generic: off
        tx-checksum-sctp: off
tcp-segmentation-offload: off
        tx-tcp-segmentation: off [requested on]
        tx-tcp6-segmentation: off [requested on]
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 tx off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: off
        tx-checksum-ip-generic: off
        tx-checksum-sctp: off
tcp-segmentation-offload: off
        tx-tcp-segmentation: off [requested on]
        tx-tcp6-segmentation: off [requested on]
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 rx off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 gso off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~#

[17:26] nsg-System:~# ethtool -K eth1 tx off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: off
        tx-checksum-ip-generic: off
        tx-checksum-sctp: off
tcp-segmentation-offload: off
        tx-tcp-segmentation: off [requested on]
        tx-tcp6-segmentation: off [requested on]
[17:26] nsg-System:~# ethtool -K eth1 gso off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
[17:26] nsg-System:~# ethtool -K eth2 gso off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
[17:26] nsg-System:~# ethtool -K eth2 rx off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
[17:26] nsg-System:~# ethtool -K eth2 tx off
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: off
        tx-checksum-ip-generic: off
        tx-checksum-sctp: off
tcp-segmentation-offload: off
        tx-tcp-segmentation: off [requested on]
        tx-tcp6-segmentation: off [requested on]
[17:26] nsg-System:~#
```

Retesting using -P50:
Still no cpu load with tcp, 100
result: 7.96 gbit/s

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-jool-v6tov4-server-tcp-P50-no-offload
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P50  -c 2001:db8:23::2a | tee iperf-jool-v6tov4-client-tcp-P50-no-offload

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P20 -u -b0  -c 10.0.0.66 | tee iperf-netpfga-v4tov6-client-udp-P20


nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-netfpga-v6tov4-server-tcp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P1  -c 2001:db8:42::a00:2a | tee iperf-netfpga-v6tov4-client-tcp-P1


nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -B 10.0.0.42 -s  | tee iperf-netfpga-v6tov4-server-udp-P1

nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -6 -p 2345 -t 70 -O 10 -P1 -b0 -u  -c 2001:db8:42::a00:2a | tee iperf-netfpga-v6tov4-client-udp-P1
```

### E.5.4 NetPFGA Benchmarks

Only 1 test did have offloading on esprimo off, was redone
v4->v6 tcp
P1: 7.41 gbit/s iperf 50P1-offload-on-esprimo: 8.43 gbit/s P20: 9.29 gbit/s iperf: 66/20P50: 9.29 gbit/s 84/42
v4->v6 udp
P1: 7.4gbit/s 100P20: 17.7gbit/s iperf 100P50: 21.5 gbit/s iperf 100
v6->v4 tcp
P1: 9.28 gbit/s atop  9800 mbit/s iperf 44P20: 9.29 gbit/s atop  9800 mbit/s iperf 70P50: 9.29 gbit/s atop  9800 mbit/s iperf 90
v6->v4 udp
P1: 7.96 gbit/s atop  8200mbit/s iperf 70P20: 13.4 gbit/s atop 9800 mbit/s iperf 100P50: 19.0 gbit/s atop 9800 mbit/s iperf 100
Commands:

```
nico@ESPRIMO-P956:~/master-thesis/iperf$ iperf3 -4 -p 2345 -t 70 -O 10 -P1 -u -b0  -c 10.0.0.66 | tee iperf-netpfga-v4tov6-client-udp-P1
```

After first netpfga, tcp v4->v6 p1 turned offloading on again

```
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 tx-checksum-ipv6 on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Could not change any device features
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 tx on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: on
        tx-checksum-ip-generic: on
        tx-checksum-sctp: on
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp6-segmentation: on
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 rx on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~# ethtool -K enp2s0f1 gso on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~# ethtool -K enp2s0f0 gso on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~# ethtool -K enp2s0f0 tx on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
Actual changes:
tx-checksumming: on
        tx-checksum-ip-generic: on
        tx-checksum-sctp: on
tcp-segmentation-offload: on
        tx-tcp-segmentation: on
        tx-tcp6-segmentation: on
root@ESPRIMO-P956:~# ethtool -K enp2s0f0 rx on
Cannot get device udp-fragmentation-offload settings: Operation not supported
Cannot get device udp-fragmentation-offload settings: Operation not supported
root@ESPRIMO-P956:~#
```

# Appendix F

# Buffer

## F.1 NetFPGA compile errors

- infinite loop in installer

```
# Fix introduced for SDNet 2017.4
sed -i 's/xsim\.dir\/xsc\/dpi\.so/dpi\.so/g' nf_sume_sdnet_ip/SimpleSumeSwitch/vivado_sim.bash
sed -i 's/xsim\.dir\/xsc\/dpi\.so/dpi\.so/g' nf_sume_sdnet_ip/SimpleSumeSwitch/vivado_sim_waveform.bash
# Fix introduced for SDNet 2018.2
sed -i 's/glbl_sim/glbl/g' nf_sume_sdnet_ip/SimpleSumeSwitch/vivado_sim_waveform.bash
sed -i 's/SimpleSumeSwitch_tb_sim#work.glbl/SimpleSumeSwitch_tb/g' nf_sume_sdnet_ip/SimpleSumeSwitch/vivado_sim_waveform.bash
cp src/*.tbl nf_sume_sdnet_ip/SimpleSumeSwitch/
cp: cannot stat 'src/*.tbl': No such file or directory
make: *** [Makefile:23: cpp_test] Error 1
[23:12] loch:minip4%


 ERROR: [XSIM 43-3409] Failed to compile generated C file xsim.dir/work.SimpleSumeSwitch_tb#work.glbl/obj/xsim_3.c.
 ERROR: [XSIM 43-3915] Encountered a fatal error. Cannot continue. Exiting...

/opt/Xilinx/Vivado/2018.2/data/../tps/llvm/3.1/lnx64.o/bin/clang  -fPIC -c -std=gnu89 -nobuiltininc -nostdinc++ -w  -Wl,-unres
olved-symbols=ignore-in-object-files -fbracket-depth=1048576 -I/opt/Xilinx/Vivado/2018.2/data/../tps/llvm/3.1/lnx64.o/bin/../li
b/clang/3.1/include -fPIC -m64  -I"/opt/Xilinx/Vivado/2018.2/data/xsim/include" "xsim.dir/work.SimpleSumeSwitch_tb#work.glbl/ob
j/xsim_3.c" -O0 -sim -o "xsim.dir/work.SimpleSumeSwitch_tb#work.glbl/obj/xsim_3.lnx64.o" -DXILINX_SIMULATOR
/opt/Xilinx/Vivado/2018.2/data/../tps/llvm/3.1/lnx64.o/bin/clang: error while loading shared libraries: libncurses.so.5: cannot
 open shared object file: No such file or directory
ERROR: [XSIM 43-3409] Failed to compile generated C file xsim.dir/work.SimpleSumeSwitch_tb#work.glbl/obj/xsim_3.c.
ERROR: [XSIM 43-3915] Encountered a fatal error. Cannot continue. Exiting...
[20:00] rainbow:SimpleSumeSwitch%


# Fix introduced for SDNet 2018.2
sed -i 's/glbl_sim/glbl/g' nf_sume_sdnet_ip/SimpleSumeSwitch/vivado_sim_waveform.bash
sed -i 's/SimpleSumeSwitch_tb_sim#work.glbl/SimpleSumeSwitch_tb/g' nf_sume_sdnet_ip/SimpleSumeSwitch/vivado_sim_waveform.bash
cp src/*.tbl nf_sume_sdnet_ip/SimpleSumeSwitch/
cp testdata/*.txt nf_sume_sdnet_ip/SimpleSumeSwitch/
cp: cannot stat 'testdata/*.txt': No such file or directory
make: *** [Makefile:17: all] Error 1
[15:46] rainbow:minip4%


make -C testdata/
make[1]: Entering directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/testdata'
./gen_testdata.py
/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/bin/pcap2axi -output Packet_in.axi -bus_width 256 src.pcap
Traceback (most recent call last):
  File "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/bin/pcap2axi", line 108, in <module>
    write_to_file(args.file_pcap, args.output)
  File "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/bin/pcap2axi", line 88, in write_to_file
    for pkt in rdpcap(file_in):
  File "/usr/lib/python2.7/dist-packages/scapy/utils.py", line 728, in rdpcap
    with PcapReader(filename) as fdesc:
  File "/usr/lib/python2.7/dist-packages/scapy/utils.py", line 751, in __call__
    filename, fdesc, magic = cls.open(filename)
  File "/usr/lib/python2.7/dist-packages/scapy/utils.py", line 778, in open
    fdesc = open(filename, "rb")
IOError: [Errno 2] No such file or directory: 'src.pcap'
make[1]: *** [Makefile:5: all] Error 1
make[1]: Leaving directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/testdata'
make: *** [Makefile:32: frontend] Error 2
[15:47] rainbow:minip4%


update_compile_order: Time (s): cpu = 00:00:17 ; elapsed = 00:00:09 . Memory (MB): peak = 1995.594 ; gain = 0.016 ; free physic
al = 21975 ; free virtual = 33161
loading libsume..
Traceback (most recent call last):
  File "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/sim_switch_de
fault/run.py", line 42, in <module>
    import config_writes
  File "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/sim_switch_de
fault/config_writes.py", line 7

                    ^
IndentationError: expected an indented block
    while executing
"exec python $::env(NF_DESIGN_DIR)/test/${test_name}/run.py"
    invoked from within
"set output [exec python $::env(NF_DESIGN_DIR)/test/${test_name}/run.py]"
    (file "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw/tcl/simple_s
ume_switch_sim.tcl" line 177)
INFO: [Common 17-206] Exiting Vivado at Sat May 18 15:21:21 2019...
```

```
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_0_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_0_stim.axi': No such file or di
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_0_expected.axi': No such file o
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_1_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_1_stim.axi': No such file or di
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_1_expected.axi': No such file o
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_2_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_2_stim.axi': No such file or di
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_2_expected.axi': No such file o
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_3_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_3_stim.axi': No such file or di
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_3_expected.axi': No such file o
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/dma_0_log.axi': No such file or directory
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/dma_0_expected.axi': No such file or directo
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/reg_stim.log': No such file or directory
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/reg_expect.axi': No such file or directory
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/reg_stim.axi': No such file or directory
=== Running test /tmp/nico/test/simple_sume_switch/sim_switch_default ... using cmd ['/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4

# add_wave $nf_sume_sdnet_ip/out_src_port
# add_wave $nf_sume_sdnet_ip/out_dst_port
# set const_reg_ip /top_tb/top_sim/nf_datapath_0/nf_sume_sdnet_wrapper_1/inst/SimpleSumeSwitch_inst/const_reg_rw_0/
# add_wave_divider {const reg extern signals}
# add_wave $const_reg_ip
ERROR: [Wavedata 42-471] Note: Nothing was found for the following items: /top_tb/top_sim/nf_datapath_0/nf_sume_sdnet_wrapper_1/inst/SimpleSumeSwitch_inst/const_reg_rw
ERROR: [Common 17-39] 'add_wave' failed due to earlier errors.

    while executing
"add_wave $const_reg_ip "
    (file "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw/tcl/simple_sume_switch_sim.tcl" line 328)
INFO: [Common 17-206] Exiting Vivado at Sat May 18 15:31:59 2019...
make: *** [Makefile:121: sim] Error 1
make: Leaving directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test'
512
=== Work directory is /tmp/nico/test/simple_sume_switch
=== Setting up test in /tmp/nico/test/simple_sume_switch/sim_switch_default
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_0_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_1_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_2_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/nf_interface_3_log.axi': No such file or dir
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/dma_0_log.axi': No such file or directory
=== Running test /tmp/nico/test/simple_sume_switch/sim_switch_default ... using cmd ['/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4
[15:31] rainbow:P4-NetFPGA%

 # open_run impl_1
 ERROR: [Common 17-69] Command failed: Run 'impl_1' has not been launched. Unable to open
 Vivado%

ERROR: [BD 41-171] The modes of the interface pins 'cfg_interrupt'(Slave) and 'pcie3_cfg_interrupt'(Slave) are incompatible. They cannot be connected.
ERROR: [BD 5-3] Error: running connect_bd_intf_net.
ERROR: [Common 17-39] 'connect_bd_intf_net' failed due to earlier errors.

    while executing
"connect_bd_intf_net -intf_net nf_riffa_dma_1_pcie3_cfg_interrupt [get_bd_intf_pins nf_riffa_dma_1/cfg_interrupt] [get_bd_intf_pins pcie3_7x_1/pcie3_cf..."
    (procedure "create_hier_cell_dma_sub" line 141)
    invoked from within
"create_hier_cell_dma_sub [current_bd_instance .] dma_sub"
    (procedure "create_root_design" line 68)
    invoked from within
"create_root_design """
    (file "./tcl/control_sub.tcl" line 729)

    while executing
"source ./tcl/control_sub.tcl"
    (file "tcl/simple_sume_switch.tcl" line 89)



### set NF_10G_INTERFACE3_BASEADDR $M07_BASEADDR
### set NF_10G_INTERFACE3_HIGHADDR $M07_HIGHADDR
### set NF_10G_INTERFACE3_SIZEADDR $M07_SIZEADDR
### set NF_RIFFA_DMA_BASEADDR $M08_BASEADDR
### set NF_RIFFA_DMA_HIGHADDR $M08_HIGHADDR
### set NF_RIFFA_DMA_SIZEADDR $M08_SIZEADDR
Wrote  : </home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw/project/simple_sume_switch.srcs/sources_1/bd/control_
# create_ip -name nf_sume_sdnet -vendor NetFPGA -library NetFPGA -module_name nf_sume_sdnet_ip
ERROR: [Coretcl 2-1134] No IP matching VLNV 'NetFPGA:NetFPGA:nf_sume_sdnet:*' was found. Please check your repository configuration.
INFO: [Common 17-206] Exiting Vivado at Sat May 25 11:52:01 2019...


# set_property compxlib.xsim_compiled_library_dir {} [current_project]                                                [0/1819]
# set_property top_lib xil_defaultlib [get_filesets sim_1]
# update_compile_order -fileset sim_1
update_compile_order: Time (s): cpu = 00:00:17 ; elapsed = 00:00:09 . Memory (MB): peak = 2003.578 ; gain = 8.004 ; free physical = 27661 ; free virtual = 33990
loading libsume..
Traceback (most recent call last):
  File "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/sim_switch_default/run.py", line 42, in <module>
    import config_writes
  File "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/sim_switch_default/config_writes.py", line 7

                    ^
IndentationError: expected an indented block
    while executing
"exec python $::env(NF_DESIGN_DIR)/test/${test_name}/run.py"
    invoked from within
"set output [exec python $::env(NF_DESIGN_DIR)/test/${test_name}/run.py]"
    (file "/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw/tcl/simple_sume_switch_sim.tcl" line 177)
INFO: [Common 17-206] Exiting Vivado at Sat May 25 13:45:13 2019...
make: *** [Makefile:121: sim] Error 1

[SW] CAM_EnableDevice() - done
[2420698]  INFO: finished packet stimulus file
[2735572] ERROR: tuple mismatch for packet 1
expected < tuple_out_digest_data, tuple_out_sume_metadata > = < 0000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000001010000 >
actual   < tuple_out_digest_data, tuple_out_sume_metadata > = < 0000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000010000 >
$finish called at time : 2735572 ps : File
"/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/nf_sume_sdnet_ip/SimpleSumeSwitch/Testbench/Check.v"
Line 120

[14:54] rainbow:bitfiles% cd $NF_DESIGN_DIR/bitfiles/ && sudo bash ./program_switch.sh
./program_switch.sh: line 34: /tools/program_switch.sh: No such file or directory
[14:56] rainbow:bitfiles% ls
config_writes.sh  minip4.bit  program_switch.sh  README
[14:56] rainbow:bitfiles%
```

```
root@rainbow:~/master-thesis/netpfga/minip4/sw/hw_test_tool# python switch_calc_tester.py
SIOCSIFADDR: No such device
eth1: ERROR while getting interface flags: No such device
SIOCSIFNETMASK: No such device
tcpdump: eth1: No such device exists
(SIOCGIFHWADDR: No such device)
The HW testing tool for the switch_calc design
 type help to see all commands
testing>
```

```
» table_cam_add_entry lookup_table send_to_port1 ff:ff:ff:ff:ff:ff =>
CAM_Init_ValidateContext() - done
WROTE 0x44020050 = 0xffffffff
WROTE 0x44020054 = 0xffff
WROTE 0x44020080 = 0x0003
python: ioctl: Unknown error 512
[20:27] rainbow:CLI%
```

## F.2 Traces

Proof of stuff working, reference for each stage / feature
Stuff that needs to be cleaned up

## F.3 Introduction

### F.3.1 The Task

- Milestone 1: Stateless NAT64/NAT46 translations in P4 - Milestone 2: Stateful (dynamic) NAT64/NAT46 translations - Milestone 3: Hardware adaption
This thesis is into 3 milestone P4 environment a lot of potential Programming language in the network Not only faster, but also more convienient.
**** High speed NAT64 with P4 Currently there are two main open source NAT64 solution available: tayga and jool. The former is a single threaded, cpu bound user space solution, the latter a custom Linux kernel module.
This thesis challenges this status quo by developing a P4 based solution supporting all features of jool/tayga and comparing the performance, security and adaptivity of the solutions.
Describe your task.

```
***** Motivation zeigen
***** IPv6, NetPFGA mehr Möglichketien
***** P4 erwähnen
***** Task gut zu zeigen,  alles erreicht
 use cases / sample applications
```

# List of Abbrevations

| | |
|---|---|
| ASIC | Application-specific integrated circuit |
| FGPA | Field-programmable gate array |
| LPM | Longes prefix matching |
| NAT | Network Address Translation |
| NAT64 | Network Address Translation from / to IPv6 to / from IPv4 |
| RIR | Regional Internet Registry |

# Bibliography


[1] AFRINIC. Afrinic ipv4 exhaustion. https://afrinic.net/exhaustion.

[2] Akamai. Ipv6 adoption visualization. https://www.akamai.com/us/en/resources/our-thinking/state-of-the-internet-report/state-of-the-internet-ipv6-adoption-visualization.jsp#countries.

[3] T. Anderson and A. L. Popper. Explicit Address Mappings for Stateless IP/ICMP Translation. RFC 7757 (Proposed Standard), Feb. 2016.

[4] APNIC. Apnic's ipv4 pool status. https://www.apnic.net/community/ipv4-exhaustion/graphical-information/.

[5] ARIN. Ipv4 addressing options. https://www.arin.net/resources/guide/ipv4/.

[6] M. Bagnulo, P. Matthews, and I. van Beijnum. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146 (Proposed Standard), Apr. 2011.

[7] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li. IPv6 Addressing of IPv4/IPv6 Translators. RFC 6052 (Proposed Standard), Oct. 2010.

[8] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 4366 (Proposed Standard), Apr. 2006. Obsoleted by RFCs 5246, 6066, updated by RFC 5746.

[9] BMV2. Implementing your switch target with bmv2. http://www.bmv2.org/.

[10] R. Braden, D. Borman, and C. Partridge. Computing the Internet checksum. RFC 1071 (Informational), Sept. 1988. Updated by RFC 1141.

[11] CISCO. 6lab - the place to monitor ipv6 adoption. https://6lab.cisco.com/stats/.

[12] A. Conta, S. Deering, and M. Gupta (Ed.). Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443 (Internet Standard), Mar. 2006. Updated by RFC 4884.

[13] M. Crawford. Transmission of IPv6 Packets over Ethernet Networks. RFC 2464 (Proposed Standard), Dec. 1998. Updated by RFCs 6085, 8064.

[14] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998. Obsoleted by RFC 8200, updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.

[15] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585.

[16] L. Foundation. Open vswitch. https://www.openvswitch.org/.

[17] Google. Ipv6 - google. https://www.google.com/intl/en/ipv6/statistics.html.

[18] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), Feb. 2006. Updated by RFCs 5952, 6052, 7136, 7346, 7371, 8064.

[19] C. Hornig. A Standard for the Transmission of IP Datagrams over Ethernet Networks. RFC 894 (Internet Standard), Apr. 1984.

[20] LACNIC. Ipv4 depletion phases. https://www.lacnic.net/1039/1/lacnic/ipv4-depletion-phases.

[21] X. Li, C. Bao, and F. Baker. IP/ICMP Translation Algorithm. RFC 6145 (Proposed Standard), Apr. 2011. Obsoleted by RFC 7915, updated by RFCs 6791, 7757.

[22] N. Lutchansky. Tayga - simple, no-fuss nat64 for linux. http://www.litech.org/tayga/.

[23] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), Sept. 2007. Updated by RFCs 5942, 6980, 7048, 7527, 7559, 8028, 8319, 8425.

[24] NetFPGA. P4-netpfga-public repository at github. https://github.com/NetFPGA/P4-NetFPGA-public.

[25] B. Networks. Tofino2. https://barefootnetworks.com/products/brief-tofino-2/.

[26] NGINX. Nginx | high performance load balancer, web server, & reverse proxy. https://www.nginx.com/.

[27] E. Nordmark and R. Gilligan. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213 (Proposed Standard), Oct. 2005.

[28] J. Postel. User Datagram Protocol. RFC 768 (Internet Standard), Aug. 1980.

[29] J. Postel. Internet Control Message Protocol. RFC 792 (Internet Standard), Sept. 1981. Updated by RFCs 950, 4884, 6633, 6918.

[30] J. Postel. Internet Protocol. RFC 791 (Internet Standard), Sept. 1981. Updated by RFCs 1349, 2474, 6864.

[31] J. Postel. Transmission Control Protocol. RFC 793 (Internet Standard), Sept. 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[32] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard), Aug. 2018.

[33] RIPE. Ipv4 exhaustion. https://www.ripe.net/publications/ipv6-info-centre/about-ipv6/ipv4-exhaustion.

[34] N. Schottelius. Add access to table keys. https://github.com/p4lang/p4-spec/issues/745.

[35] N. Schottelius. Casting bit<16> to bit<32> in checksum causes incorrect json to be generated. https://github.com/p4lang/p4c/issues/1765.

[36] theojepsen. Get size of header. https://github.com/p4lang/p4-spec/issues/660.

[37] ungleich. Die ipv4, die! https://ungleich.ch/en-us/cms/blog/2019/01/09/die-ipv4-die/.

[38] L. Vanbever. Programming network data planes. https://github.com/nsg-ethz/p4-learning/blob/master/slides/02_p4_env.pdf.

[39] E. Vyncke. Ipv6 deployment aggregated status. https://www.vyncke.org/ipv6status/.

[40] Wikipedia. Ipv4 header checksum. https://en.wikipedia.org/wiki/IPv4_header_checksum. Requested on 2019-08-12.

[41] Wikipedia. Ipv6 transition mechanism. https://en.wikipedia.org/wiki/IPv6_transition_mechanism. As requested on 2019-08-08.