Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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. We explain the current state of IPv4 exhaustion and IPv6 adoption and describe how it motivates our work to support ease transition to IPv6 networks.

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 [48], [4], [29], [1], [5]. Figure 1.2 contains summarised data from all RIRs and projects complete IPv4 addresses depletion by 2021. The LACNIC project even predicts complete exhaustion for 2020 as shown in figure 1.1. On the other

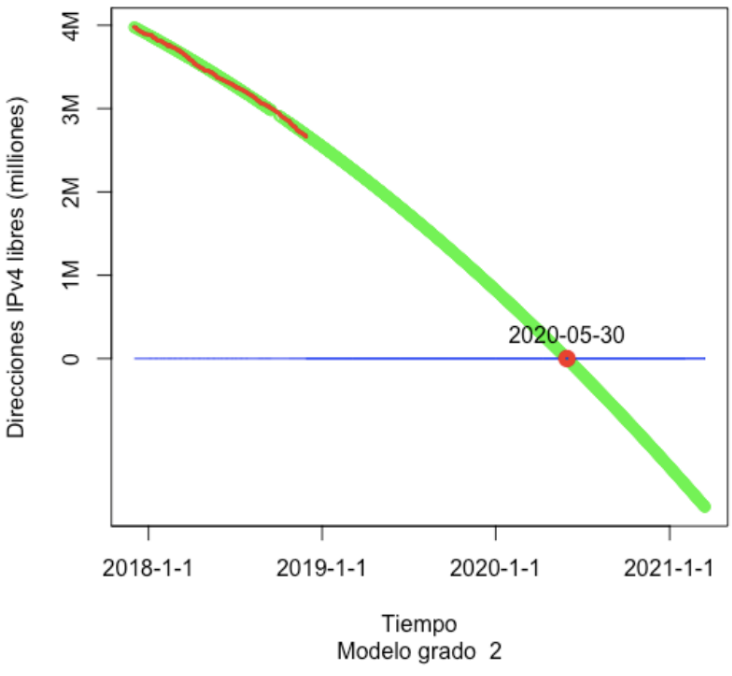

Figure 1.1: LACNIC Exhaustion projection, [29]

hand IPv6 adoption grows significantly, with at least three countries (India, US, Belgium) surpassing 50% adoption [2], [60], [14]. Traffic from Google users reaches almost 30% as of 2019-08-08 [22], see figure 1.3. 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.

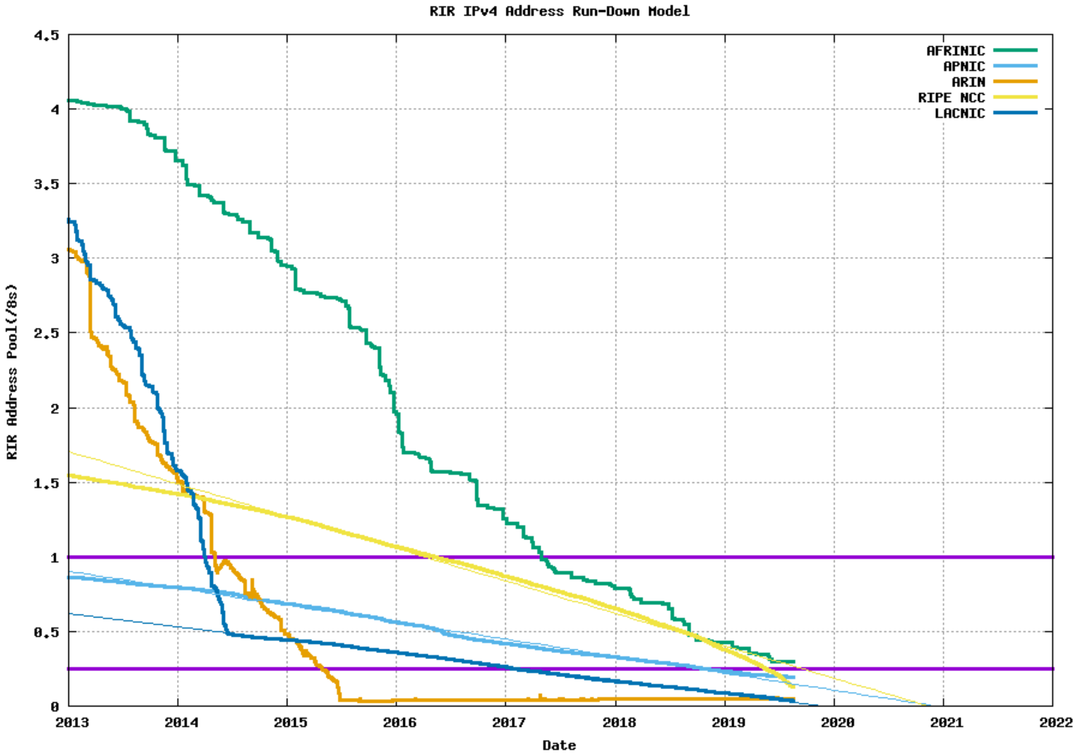

Figure 1.2: RIR IPv4 rundown projection from [26]

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 [62], [39]. 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 hosts to explicitly use an extra device in the network,[1] our proposed method is transparent to the hosts. This way the routing and network configuration does not need to be changed to support NAT64 within a network. Currently 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 1.4 shows how the complexity quickly grows even with a small number of hosts. The proposed 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 [37], FPGAs [36] or even in software [10]. Figure 3.3 shows how the design differs for an in-network solution. Even on fast CPUs, software solutions like tayga [31] can be CPU bound (see section 4.2) and are incapabale of translating protocols at line speed.

[1] Usually the default router will take this role.

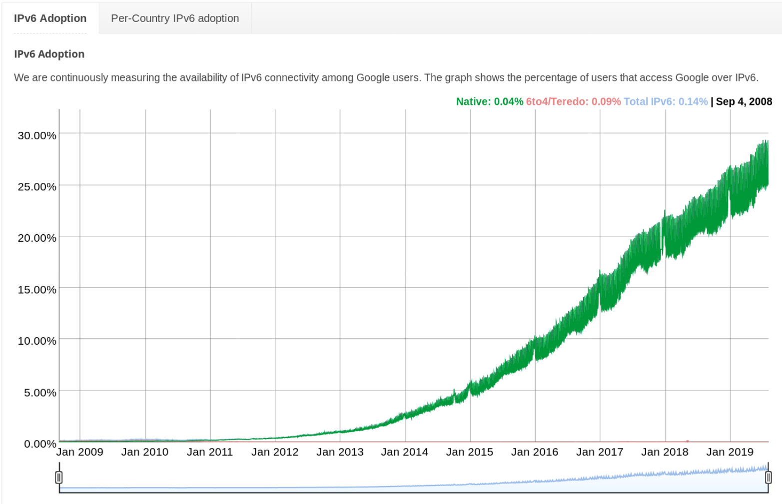

Figure 1.3: Google IPv6 Statistics from [22]

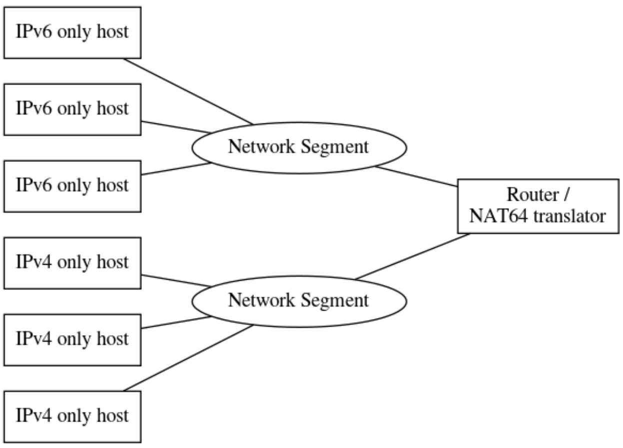

Figure 1.4: Separated IPv6 and IPv4 network segments

Chapter 2

Background

In this chapter we describe the key technologies involved and their relation to our work.

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 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 are 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 code to

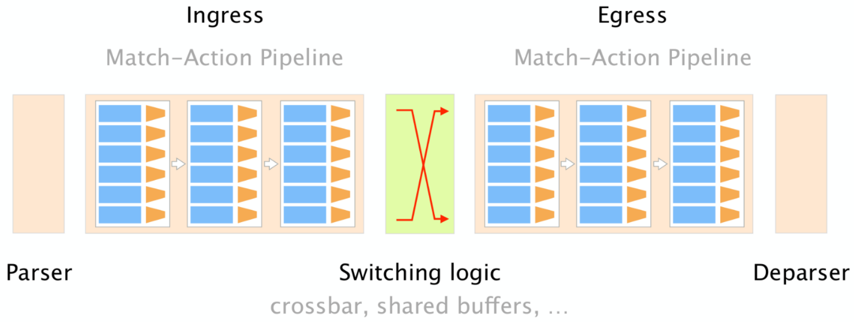

Figure 2.1: P4 protocol independence [58]

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 4.1.
As opposed to general purpose programming languages, P4 lacks some features. Most notably loops, floating point operations and modulo operations. 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 4.2 for details).

2.2 IPv6, IPv4 and Ethernet

The first IPv6 RFC was published in 1998 [18]. 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.[1]

[1] 0x0800 for IPv4 [25] and 0x86DD for IPv6 [16].

This is important, because Ethernet can only reference one protocol, which makes IPv4 and IPv6 mutually exclusive. The figures 2.5 and 2.4 show the packet headers of IPv4 and IPv6. The most notable differences between the two protocols for this thesis are:

- Different address lengths
  - IPv4: 32 bit
  - IPv6: 128 bit
- Lack of a checksum in IPv6
- Format of Pseudo headers (see section 2.5)

2.3 ARP and NDP, ICMP and ICMP6

While IPv6 and IPv4 are primarily used as a “shell” to support addressing for protocols that have no or limited addressing support (like TCP or UDP), protocols like ARP [40] and NDP [35] provide support for resolving IPv6 and IPv4 addresses to hardware (MAC) addresses. While both ARP and NDP are only used prior to establishing a connection on and their results are cached, their availability is crucial for operating a switch. Figure 2.2 illustrates a typical address resolution process. The major difference between ARP and NDP in relation to P4 are

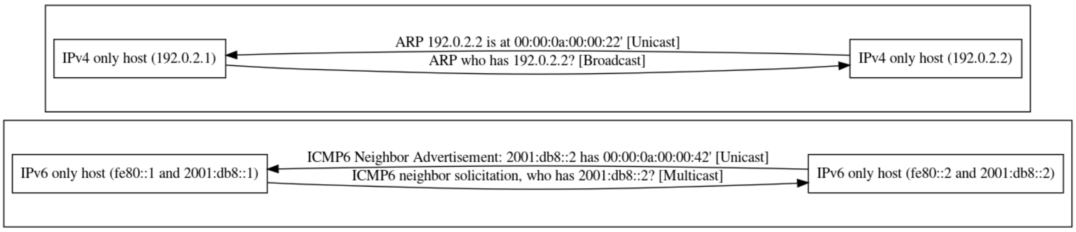

Figure 2.2: ARP and NDP

- ARP is a separate protocol on the same layer as IPv6 and IPv4,
- NDP operates below ICMP6 which operates below IPv6,
- NDP contains checksums over payload,
- and NDP in ICMP6 contains optional, non referenced option fields (specifically: ICMP6 link layer address option).

ARP is required to be a separate protocol, because IPv4 hosts don’t know how to communicate with each other yet, because they don’t have a way to communicate to the target IPv4 address (“The chicken and the egg problem”). NDP on the other hand already works within IPv6, as every IPv6 host is required to have a self-assigned link local IPv6 address from the range `fe80::/10` (compare RFC4291 [24]). NDP also does not require broadcast communication, because hosts automatically join multicast groups that embed parts of their IPv6 addresses [17], [63]. This way the collision domain is significantly reduced in IPv6, compared to IPv4.
As seen later in this document (compare section 4.1.2), the requirement to generate checksums over payload poses difficult problems for some hardware targets. Even more difficult is the use of options within ICMP6. Figure shows a typical layout of a neighbor advertisement messages. The problem arises from the layout of the options, as seen in the following quote and in figure

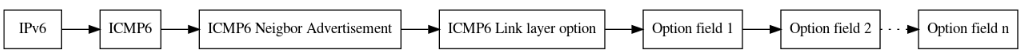

Figure 2.3: ICMP6 option fields

??:

> “Neighbor Discovery messages include zero or more options, some of which may appear multiple times in the same message. Options should be padded when necessary to ensure that they end on their natural 64-bit boundaries”.[2]

ICMP6 and ICMP are primarily used to signal errors in communication. Specifically signalling that a packet is too big to pass a certain link and needs fragmentation is a common functionality of both protocols. For a host (or switch) to be able to emit ICMP6 and ICMP messages, the host requires a valid IPv6 / IPv4 address. Without ICMP6 / ICMP support path MTU discovery [34], [32] does not work and the sender needs to determine different ways of finding out the maximum MTU on the path.

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 Stateless NAT64

Stateless 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

Figures 2.4 and 2.5 show the headers of IPv4 and IPv6. As can be seen in the diagrams not only are the addresses of different size, but fields have also been changed or removed when the version changed. Depending on the NAT64 translation direction, a translator will need to re-arrange fields to a different position, remove fields and add fields.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.4: IPv6 Header [18]

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 [54]: 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

[2] Quote from [35].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.5: IPv4 Header [43]

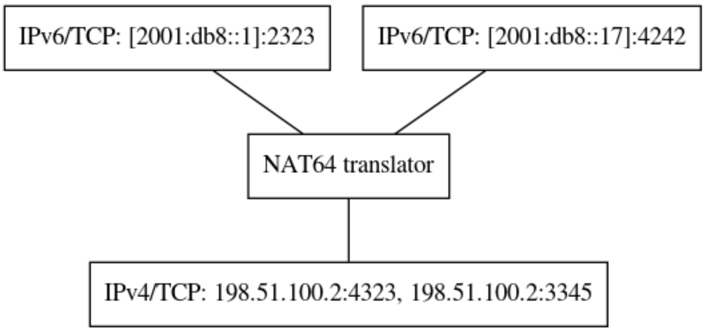

Figure 2.6: Stateful NAT64

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 NAT64 usually uses information found in protocols at layer 4 like TCP [44] or UDP [41]. However it can also support ICMP [42] and ICMP6 [15].

2.4.3 Higher Layer Protocol Dependent Translation

Further translation can be achieved by using information in higher level protocols like HTTP [20] or TLS [9]. Application proxies like nginx [38] 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 [46], 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 Mapping IPv4 Addresses in IPv6

As described in section 2.2, one of the major differences between IPv6 and IPv4 is the address length. As the whole IPv4 Internet can be represented in only 32 bits, it is a common practice to assign an IPv6 prefix for IPv6 hosts that represents a mapping to the IPv4 Internet. In RFC6052 [13] the well known prefix *64:ff9b::/96* is defined. One possibility to map an IPv4 address into the prefix is by adding its integer value to the prefix, treating it like an offset. In figure 2.7 we show an example python code of how this can be done. Network adminis-

```
»> import ipaddress
»> prefix=ipaddress.IPv6Network("64:ff9b::/96")
»> ipv4address=ipaddress.IPv4Address("192.0.2.0")
»> int(ipv4address)
3221225984
»> hex(3221225984)
'0xc0000200'
»> prefix[int(ipv4address)]
IPv6Address('64:ff9b::c000:200')
```

Figure 2.7: Representing an IPv4 address in an IPv6 prefix

trators can choose to use either the well known prefix or to use a network block of their own to map the Internet.[3] While a /96 prefix seems a natural selection (it provides exactly 32 bit), other prefix lengths are defined in RFC6052 (see figure 2.8) that allow flexible embedding of the IPv4 address. RFC6146, which describes stateful NAT64, states that "IPv4 addresses of IPv4

```
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|PL| 0---------32-40-48-56-64-72-80-88-96-104------|
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|32|     prefix    |v4(32)         | u | suffix                |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|40|     prefix        |v4(24)     | u |(8)| suffix            |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|48|     prefix            |v4(16) | u | (16)  | suffix        |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|56|     prefix                |(8)| u |  v4(24)   | suffix    |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|64|     prefix                    | u |   v4(32)      | suffix |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|96|     prefix                                    |    v4(32)   |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.8: IPv4 embedding depending on the prefix length

hosts are algorithmically translated to and from IPv6 addresses by using the algorithm defined in [RFC6052]" [6] While this sentencen does not use the typical RFC keywords like SHALL, REQUIRED, etc. [12], we interpret this sentence in the meaning of "a stateful NAT64 translator SHALL implement IPv4 address embedding as described in the algorithm of RFC6052".

2.4.5 DNS64

Tightly related to NAT64 is a technology known as DNS64 [28]. DNS64 tries to solve the problem of addressing IPv4 only hosts from IPv6 only hosts by adding a "fake" IPv6 (AAAA) DNS resource record, as shown in figure 2.9. The DNS64 DNS server will query the authora-

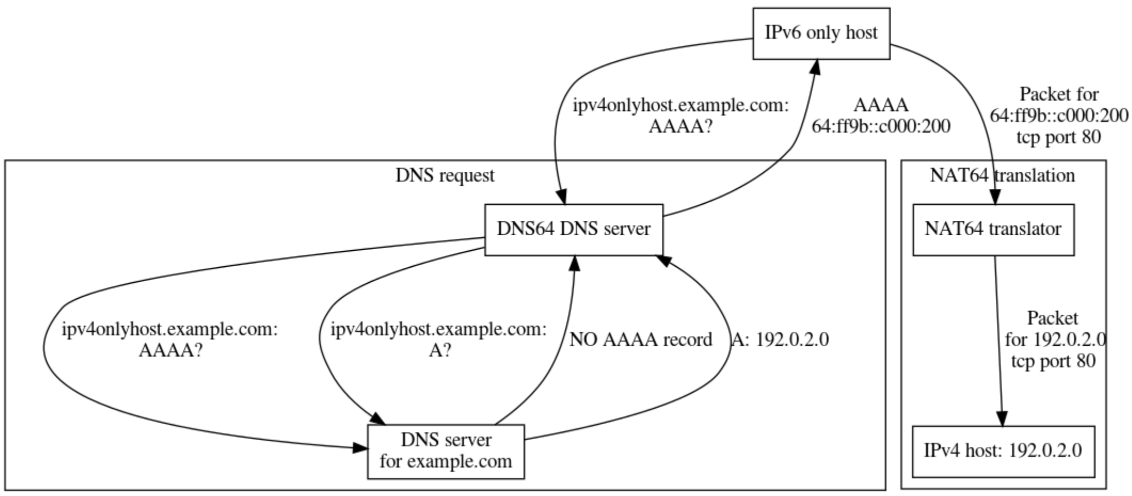

Figure 2.9: Illustration of DNS64

tive DNS server for an AAAA record. However as the host *ipv4onlyhost.example.com* is only reachable by IPv4, it also only has an A entry. After receiving the answer that there is no AAAA record, the DNS64 server will ask for an A record and gets an answer that the name *ipv4onlyhost.example.com* resolves to the IPv4 address *192.0.2.0*. The DNS64 server then embeds the IPv4 address in the configured IPv6 prefix (*64:ff9b::/96* in this case) and returns a fake AAAA record to the IPv6 only host. The IPv6 only host then will use address to connect to. The NAT64 translator recognises either that the address is part of a configured prefix or that it has a dedicated table entry for mapping this IPv6 address to an IPv4 address and translates it accordingly.

[3] For instance 2a0a:e5c0:0:1::/96 [57].

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 [18] and shown in figure 2.10, the IPv4 pseudo header for TCP and UDP are defined in RFC768 and RFC793 and are shown in 2.11. When

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Upper-Layer Packet Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      zero                     |  Next Header  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.10: IPv6 Pseudo Header

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 (see section 3.7). The checksums for IPv4, TCP, UDP and ICMP6 are all based on

```
0      7 8     15 16    23 24    31
+-----+-----+-----+-----+
|          source address          |
+-----+-----+-----+-----+
|        destination address       |
+-----+-----+-----+-----+
|  zero  |protocol| TCP/UDP length  |
+-----+-----+-----+-----+
```

Figure 2.11: IPv4 Pseudo Header

the “Internet Checksum” [43], [11]. 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.[4].

2.6 Network Designs

In relation to IPv6 and IPv4, there are in general three different network designs possible: The oldest form are IPv4 only networks. 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”. 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. 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 [56]. 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.

[4] Quote from Wikipedia [61].

2.6.1 IPv4 only network limitations

As shown in figures 2.5 and 2.4 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[5] 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[6] 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.

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.

[5] See section 2.2.

[6] 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.

Chapter 3

Design

In this chapter we describe the architecture of our solution and our design choices.

3.1 IPv6 and IPv4 configuration

The following sections refer to host and network configurations. In this section we describe the IPv6 and IPv4 configurations as a basis for the discussion.
All IPv6 addresses are from the documentation block *2001:DB8::/32* [27]. In particular the following sub networks and IPv6 addresses are used:

Address	Description
2001:db8:42::/64	IPv6 host network
2001:db8:23::/96	IPv6 mapping to the IPv4 Internet
2001:db8:42::42	IPv6 host address
2001:db8:42::77	IPv6 router address
2001:db8:42::a00:2a	In-network IPv6 address mapped to 10.0.0.42 (p4)
2001:db8:23::a00:2a	IPv6 address mapped to 10.0.0.42 (tayga)
2001:db8:23::2a	IPv6 address mapped to 10.0.0.42 (jool)

Table 3.1: IPv6 address and network overview

We use private IPv4 addresses as specified by RFC1918 [45] from the 10.0.0.0/8 range as follows:

Address	Description
10.0.0.0/24	IPv4 host network
10.0.1.0/24	IPv4 network mapping to IPv6
10.0.0.77	IPv4 router address
10.0.0.66	In-network IPv4 address mapped to 2001:db8:42::42 (p4)
10.0.1.42	IPv4 address mapped to 2001:db8:42::42 (tayga)
10.0.1.66	IPv4 address mapped to 2001:db8:42::42 (jool)

Table 3.2: IPv4 address and network overview

3.2 NAT64 Verification

We use socat [47] to verify basic operation of the NAT64 gateway and iperf [19] to test stability of the implementation and measure bandwidth. In particular we use the commands listed in table 3.3. The socat commands allow interactive testing on TCP and UDP connections, while the iperf commands fully utilise the available bandwidth with test data. The socat and iperf commands are used to verify all three NAT64 implementations (p4, tayga, jool).

Command	Example	Description
`socat - TCP6:HOST:PORT`	socat - TCP6:[2001:db8:42::a00:2a]:2345	Connect via IPv6/TCP to IPv4 host
`socat - UDP6:HOST:PORT`	socat - UDP6:[2001:db8:42::a00:2a]:2345	Connect via IPv6/UDP to IPv4 host
`socat - TCP:HOST:PORT`	socat - TCP:10.0.1.42:2345	Connect via IPv4/TCP to IPv6 host
`socat - UDP:HOST:PORT`	socat - UDP:10.0.1.42:2345	Connect via IPv4/UDP to IPv6 host
`socat - UDP6-LISTEN:PORT`	socat - UDP6-LISTEN:2345	Listen on IPv6/UDP
`socat - TCP6-LISTEN:PORT`	socat - TCP6-LISTEN:2345	Listen on IPv6/TCP
`socat - UDP-LISTEN:PORT`	socat - UDP-LISTEN:2345	Listen on IPv4/UDP
`socat - TCP-LISTEN:PORT`	socat - TCP-LISTEN:2345	Listen on IPv4/TCP
`iperf3 -PROTO -p PORT -B IP -s`	iperf3 -4 -p 2345 -B 10.0.0.42 -s	IPv4 iperf server
	iperf3 -6 -p 2345 -B 2001:db8:42::42 -s	IPv6 iperf server
`iperf3 -PROTO -p PORT -O IGNORETIME -t RUNTIME -P PARALLEL -c IP`	iperf3 -6 -p 2345 -O 10 -t 190 -P20 -c 2001:db8:23::2a	Connect to iperf server Run for 190 seconds, skip first 10 seconds with 20 sessions connecting to 2001:db8:23::2a
`iperf3 -PROTO -p PORT -O IGNORETIME -t RUNTIME -P PARALLEL -c IP -u -b0`		Same as above, but connect via UDP

Table 3.3: NAT64 verification commands

3.3 NAT64 with P4

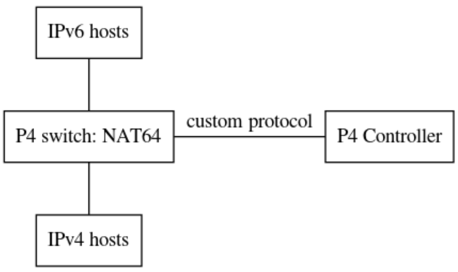

Figure 3.1: P4 Switch Architecture

In section 2.4 we discussed different translation mechanisms for IPv6 and IPv4. In this thesis we focus on the translation mechansims stateless and stateful NAT64. While higher layer protocol dependent translations are more flexible, this topic has already been addressed in [53] and the focus in this thesis is on the practicability of high speed NAT64 with P4. The high level design can be seen in figure 3.1: a P4 capable switch is running our code to provide NAT64 functionality. A P4 switch cannot manage its tables on it own and needs support for this from a controller. The controller also has the role to handle unknown packets and can modify the runtime configuration of the switch. This is especially useful in the case of stateful NAT64. If only static table entries are required, they can usually be added at the start of a P4 switch and the controller can also be omitted. However stateful NAT64 requires the use of a controller to create session entries in the switch tables. The P4 switch can use any protocol to communicate with the controller, as the connection to the controller is implemented as a separate ethernet port.

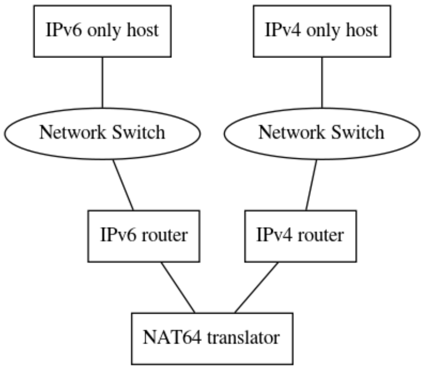

Figure 3.2: Standard NAT64 translation

Software NAT64 solutions typically require routing to be applied to transport the packet to the NAT64 translator as shown in figure 3.2.
Our design differs here: while routing could be used like described above, NAT64 with P4 does not require any routing to be setup. Figure 3.3 shows the network design that we realise using P4. This design has multiple advantages: first it reduces the number of devices to pass and thus directly reduces the RTT, secondly it allows translation of IP addresses within the same logic network segment.

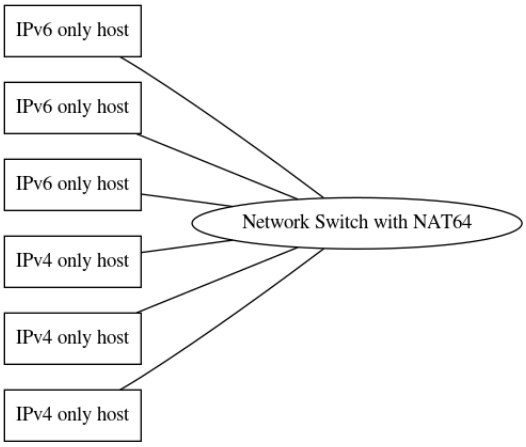

Figure 3.3: In-network NAT64 translation

3.4 Stateless NAT64

As seen in section 2.4.1, stateless NAT64 can be implemented using various factors. Our design for the stateless depends on the capabilities of the environment and is summarised in table 3.4. When using LPM for translating from IPv6 to IPv4, a /96 IPv6 network is configured for covering the whole IPv4 Internet and the individual IPv4 address is appended to the prefix (compare section 3.1). We also use LPM to match on an IPv4 sub network that translates to an IPv6 sub network. Individual entries are configured differently depending on the implementation: Limitations in the P4/NetFPGA environment require to use table entries. Jool supports individual entries as a special case of LPM, with a network mask matching only one IP address. Tayga support LPM for translation from IPv6 to IPv4, but requires invidiual entries for translating from IPv4 to IPv6. Our P4/BMV2 offers the highest degree of flexibility, as it provides support for invidual entries based on table entries and LPM table entries.

Implementation	NAT64 match
P4/BMV2	LPM (both directions) and individual entries (both directions)
P4/NetPFGA	Individual entries
Tayga	LPM (IPv6 to IPv4) and individual entries (IPv4 to IPv6)
Jool	LPM (both directions)

Table 3.4: NAT64 match factors

3.5 Stateful NAT64

Similar to stateless NAT64, the design of stateful NAT64 depends on the features of the invidual implementation. As pointed out in section 2.4.2, stateful NAT64 is very similar to stateless NAT64, with the main difference being an additional stateful table that helps to create 1:n mappings. We use different approaches within the implementations to solve this problem:

- For P4/BMV2 and P4/NetPFGA a python controller handles packets that don't have a table entry, sets the table entry in the P4 switch and inserts the original packet afterwards back into the switch.
- With tayga we rely on the Linux kernel NAT44 capabilities
- Jool implements its own stateful mechanism based on a port ranges

All methods though operate in a very similar fashion: A "controller" inspects the IPv6 packet and depending on the source address, destination address, protocol (TCP, UDP, ICMP, ICMP6, etc.) and the protocol ID (source / destination TCP/UDP port, ICMP identifier) it selects an outgoing IPv4 address, and source port or ICMP identifier. In case of Jool and Tayga this decision is based on a session table inside the Linux kernel, in case of P4 this decision is based on a session table inside the python controller. While the Jool and Tayga both support cleaning up old session entries, our P4 based solution does not support this feature at the moment.

3.6 P4/BMV2

```
/* 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 3.4: P4/BMV2 checksumming

The software emulated switch that is implemented using Open vSwitch [21] and the behavioral model [10] offers the fastest and easiest way of P4 development. All NAT64 features are tested first on P4/BMV2 and in a second step ported to P4/NetFPGA and modified, where necessary. The development follows closely the general design shown in section 3.3. As outlined in section 2.5, checksums inside higher level protocols need to be adjusted after translation. 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 sophisticated and provides direct support for calculating the checksum over the payload. This allows the BMV2 model to operate as a full featured host, including advanced features like responding to ICMP6 Neighbor discovery requests [35] that include payload checksums. Sample code that calculates the required checksum for answering NDP queries is shown in figure 3.4. The code shows how the field `hdr.icmp6.checksum` is updated with the `csum16` method depending on the IPv6 and ICMP6 headers as well as the payload. The second option of using the differences is described in section 3.7.

3.7 P4/NetFPGA

```
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 3.5: Calculating checksum based on header differences

While the P4-NetFPGA project [36] allows compiling P4 to the NetPFGA, the design slightly varies due to limitations in the available toolchain. In particular, the NetFPGA P4 compiler does not support reading the payload.[1] 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 is calculated using differences between the IPv6 and IPv4 headers.

As the checksum calculation only depends on the 1-complement sums of headers and the payload (compare section 2.5) and only headers are modified during NAT64 translations, the higher level protocol checksums can be corrected based on the sum of differences of both headers. Thus our P4/NetFPGA implementation first calculates the sum of the relevant IPv4 headers (`v4sum()`), the sum of the relevant IPv6 headers (`v6sum()`) and then calculates the difference including a possible carry bit and adjusts the higher level protocol by this difference (`delta_tcp_from_v6_to_v4()`). Figure 3.5 shows an excerpt of the code used for adjust the checksum when translating TCP from IPv6 to IPv4. It is notable that not the full headers are used, but only a “pseudo header” (compare figures 2.10 and 2.11).

[1] This feature could be implemented in theory, but isn’t available at the moment, see [51].

Chapter 4

Results

This section describes the achieved results and compares the P4 based implementation with real world software solutions.
We distinguish the software implementation of P4 (BMV2) and the hardware implementation (NetFPGA) due to significant differences in deployment and development. We present benchmarks for the existing software solutions as well as for our hardware implementation. As the objective of this thesis was to demonstrate the high speed capabilities of NAT64 in hardware, no benchmarks were performed on the P4 software implementation.

4.1 P4 based implementations

All planned features could be realised with P4 and a controller. For this thesis the parsing capabilities of P4 were adequate. However P4, at the time of writing, cannot parse ICMP6 options in general, as the upper level protocol does not specify the number of options that follow and parsing of an undefined number of 64 bit blocks is required.
The language has some limitations on where the placement of conditional statements (`if/switch`).[1] Furthermore P4/BMV2 does not support for multiple LPM keys in a table, however it supports multiple keys with ternary matching, which is a superset of LPM matching.
When developing P4 programs, the reason for incorrect behaviour we have seen were checksum problems. This is in retrospective expected, as the main task our implementation does is modify headers on which the checksums depend. In all cases we have seen Ethernet frame checksum errors, the effective length of the packet was incorrect.
The tooling around P4 is somewhat fragile. We encountered small language bugs during the development [50], **??** or found missing features [49], [55]: it is at the moment impossible to retrieve the matching key from table or the name of the action called. Thus if different table entries call the same action, it is impossible within the action, or if forwarded to the controller, within the controller to distinguish on which match the action was triggered. This problem is very consistent within P4, as not even the matching table name can be retrieved. While these information can be added manually as additional fields in the table entries, we would expect a language to support reading and forwarding this kind of meta information.
While in P4 the P4 code and the related controller are tightly coupled, their data definitions are not. Thus the packet format definition that is used between the P4 switch and the controller has to be duplicated. Our experiences in software development indicate that this duplication is a likely source of errors in bigger software projects.
The supporting scripts in the P4 toolchain are usually written in python2. However python2 “is legacy” [59]. During development errors with unicode string handling in python2 caused changes to IPv6 addresses. **??**
P4os - reusable code
****** TODO IPv6 udp -> IPv4 - Got 4-5 tuple ([proto], src ip, src port, dst ip, dst port) - Does not / never signal end - Needs timeout for cleaning up
P4/BMV2 thus allows us to closest resemble any other translation implementation.
Only supporting /96, not other embeddings as described in section 2.4.4.

[1] In general, if and switch statements in actions lead to errors, but not all constellations are forbidden.

4.1.1 BMV2

The software implementation of P4 has most features, which is mostly due to the capability of checksumming the payload: Acting as a “proper” participant in NDP, requires the host to calculate checksums over the payload.

List of features BMV2 [**?**]

Feature	Description	Status
Switch to controller	Switch forwards unhandeled packets to controller	fully implemented[a]
Controller to Switch	Controller can setup table entries	fully implemented[b]
NDP	Switch responds to ICMP6 neighbor solicitation request (without controller)	fully implemented[c]
ARP	Switch can answer ARP request (without controller)	fully implemented[d]
ICMP6	Switch responds to ICMP6 echo request (without controller)	fully implemented[e]
ICMP	Switch responds to ICMP echo request (without controller)	fully implemented[f]
NAT64: TCP	Switch translates TCP with checksumming from/to IPv6 to/from IPv4	fully implemented[g]
NAT64: UDP	Switch translates UDP with checksumming from/to IPv6 to/from IPv4	fully implemented[h]
NAT64: ICMP/ICMP6	Switch translates echo request/reply from/to ICMP6 to/from ICMP with checksumming	fully implemented[i]
NAT64: Sessions	Switch and controller create 1:n sessions/mappings	fully implemented[j]
Delta Checksum	Switch can calculate checksum without payload inspection	fully implemented[k]
Payload Checksum	Switch can calculate checksum with payload inspection	fully implemented[l]

[a] Source code: `actions_egress.p4`
[b] Source code: `controller.py`
[c] Source code: `actions_icmp6_ndp_icmp.p4`
[d] Source code: `actions_arp.p4`
[e] Source code: `actions_icmp6_ndp_icmp.p4`
[f] Source code: `actions_icmp6_ndp_icmp.p4`
[g] Source code: `actions_nat64_generic_icmp.p4`
[h] Source code: `actions_nat64_generic_icmp.p4`
[i] Source code: `actions_nat64_generic_icmp.p4`
[j] Source code: `actions_nat64_session.p4`, `controller.py`
[k] Source code: `actions_delta_checksum.p4`
[l] Source code: `checksum_bmv2.p4`

Table 4.1: P4 / BMV2 feature list

Responds to icmp, icmp6 ndp [35] arp

very easy to use

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 [30].

Stateful : no automatic removal

Session management not benchmarked, as it is only a matter of creating table entries.

Jool and tayga are supported by

4.1.2 NetFPGA - FIXME: writing

The reduced feature set of the NetPFGA implementation is due to two factors: compile time. Between 2 to 6 hours per compile run. No payload checksum

overview - general translation - not advanced features

Feature	Description	Status
Switch to controller	Switch forwards unhandeled packets to controller	portable[a]
Controller to Switch	Controller can setup table entries	portable[b]
NDP	Switch responds to ICMP6 neighbor solicitation request (without controller)	portable[c]
ARP	Switch can answer ARP request (without controller)	portable[d]
ICMP6	Switch responds to ICMP6 echo request (without controller)	portable[e]
ICMP	Switch responds to ICMP echo request (without controller)	portable[f]
NAT64: TCP	Switch translates TCP with checksumming from/to IPv6 to/from IPv4	fully implemented[g]
NAT64: UDP	Switch translates UDP with checksumming from/to IPv6 to/from IPv4	fully implemented[h]
NAT64: ICMP/ICMP6	Switch translates echo request/reply from/to ICMP6 to/from ICMP with checksumming	portable[i]
NAT64: Sessions	Switch and controller create 1:n sessions/mappings	portable[j]
Delta Checksum	Switch can calculate checksum without payload inspection	fully implemented[k]
Payload Checksum	Switch can calculate checksum with payload inspection	unsupported[l]

[a] While the NetFPGA P4 implementation does not have the clone3() extern that the BMV2 implementation offers, communication to the controller can easily be realised by using one of the additional ports of the NetFPGA and connect a physical network card to it.

[b] The p4utils suite offers an easy access to the switch tables. While the P4-NetFPGA support repository also offers python scripts to modify the switch tables, the code is less sophisticated and more fragile.

[c] NetFPGA/P4 does not offer calculating the checksume over the payload. However delta checksumming can be used to create the required checksum for replying.

[d] As ARP does not use checksums, integrating the source code `actions_arp.p4` into the netpfga code base is enough to enable ARP support in the NetPFGA.

[e] Same reasoning as NDP.

[f] Same reasoning as NDP.

[g] Source code: `actions_nat64_generic_icmp.p4`

[h] Source code: `actions_nat64_generic_icmp.p4`

[i] ICMP/ICMP6 translations only require enabling the icmp/icmp6 code in the netpfga code base.

[j] Same reasoning as “Controller to switch”.

[k] Source code: `actions_delta_checksum.p4`

[l] To support creating payload checksums, either an HDL module needs to be created or to modify the generated the PX program. [51]

Table 4.2: P4 / NetFPGA feature list

Features

Stability

Two different NetPFGA cards were used during the development of the thesis. The first card had consistent ioctl errors (compare section F.1) when writing table entries. The available hardware tests (compare figures 4.1 and 4.2) showed failures in both cards, however the first card reported an additional “10G_Loopback” failure. Due to the inability of setting table entries, no benchmarking was performed on the first NetFPGA card. During the development and bench-

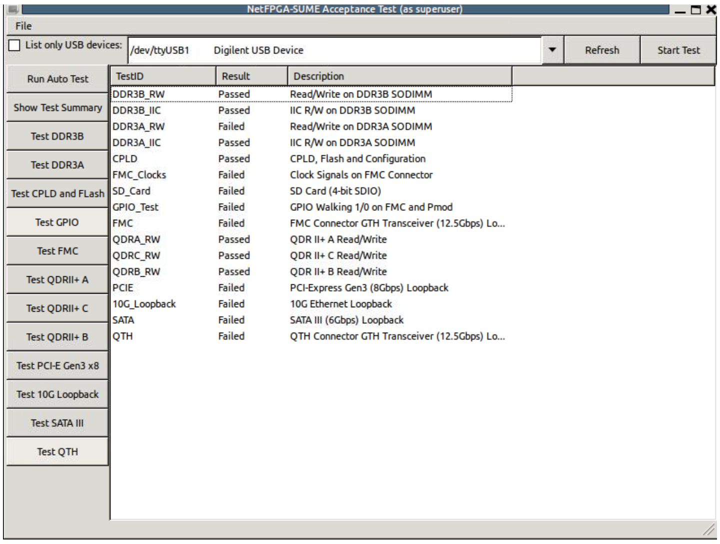

Figure 4.1: Hardware Test NetPFGA card 1

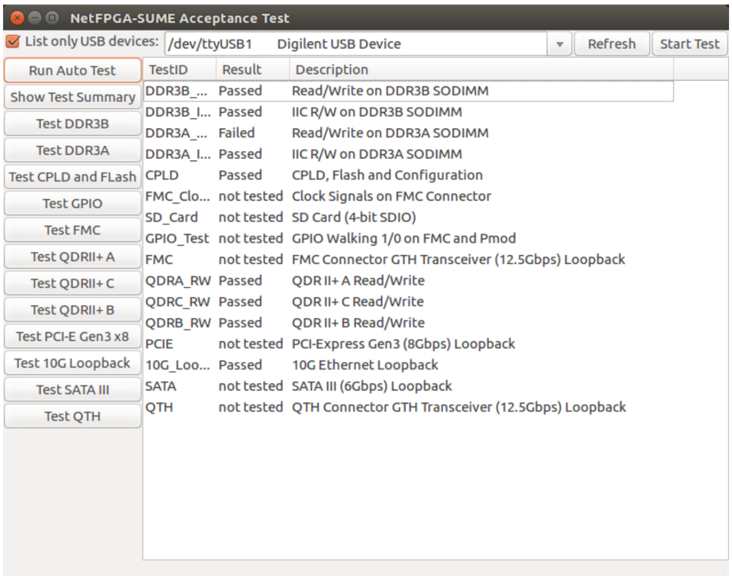

Figure 4.2: Hardware Test NetPFGA card 2, [23]

marking, the second NetFPGA card stopped to function properly multiple times. In both cases the card would not forward packets anymore. Multiple reboots (3 were usually enough) and multiple times reflashing the bitstream to the NetFPGA usually restored the intended behaviour. However due to this “crashes”, it was impossible to complete a full benchmark run that would last for more than one hour.
Sometimes it was also required to reboot the host containing the NetFPGA card 3 times to enable successful flashing.[2]

[2] Typical output of the flashing process would be: “fpga configuration failed. DONE PIN is not HIGH”

Performance

As expected, the NetFGPA card performed at near line speed and offers NAT64 translations at 9.28 Gbit/s. Single and multiple streams performed almost exactly identical and have been consistent through multiple iterations of the benchmarks.

Usability

To use the NetFGPA, Vivado and SDNET provided by Xilinx need to be installed. However a bug in the installer triggers an infinite loop, if a certain shared library[3] is missing on the target operating system. The installation program seems still to be progressing, however does never finish.
While the NetFPGA card supports P4, the toolchains and supporting scripts are in a immature state. The compilation process consists of at least 9 different steps, which are interdependent[4] Some of the steps generate shell scripts and python scripts that in turn generate JSON data.[5] However incorrect parsing generates syntactically incorrect scripts or scripts that generate incorrect output. The toolchain provided by the NetFGPA-P4 repository contains more than 80000 lines of code. The supporting scripts for setting table entries require setting the parameters for all possible actions, not only for the selected action. Supplying only the required parameters results in a crash of the supporting script.
The documentation for using the NetFPGA-P4 repository is very distributed and does not contain a reference on how to use the tools. Mapping of egress ports and their metadata field are found in a python script that is used for generating test data.
The compile process can take up to 6 hours and because the different steps are interdependent, errors in a previous stage were in our experiences detected hours after they happened. The resulting log files of the compilation process can be up to 5 MB in size. Within this log file various commands output references to other logfiles, however the referenced logfiles do not exist before or after the compile process.
During the compile process various informational, warning and error messages are printed. However some informational messages constitute critical errors, while on the other hand critical errors and syntax errors often do not constitue a critical error.[6] Also contradicting output is generated.[7]
Programs or scripts that are called during the compile process do not necessarily exit non zero if they encountered a critical error. Thus finding the source of an error can be difficult due to the compile process continuing after critical errors occured. Not only programs that have critical errors exit "successfully", but also python scripts that encounter critical paths don't abort with raise(), but print an error message to stdout and don't abort with an error.
The most often encountered critical compile error is "Run 'impl_1' has not been launched. Unable to open". This error indicates that something in the previous compile steps failed and can refer to incorrectly generated testdata to unsupported LPM tables.
The NetFPGA kernel module provides access to virtual Linux devices (nf0...nf3). However tcpdump does not see any packets that are emitted from the switch. The only possibility to capture packets that are emitted from the switch is by connecting a physical cable to the port and capturing on the other side.
Jumbo frames[8] are commonly used in 10 Gbit/s networks. According to **??**, even many gigabit network interface card support jumbo frames. However according to emails on the private NetPFGA mailing list, the NetFPGA only supports 1500 byte frames at the moment and additional work is required to implement support for bigger frames.

[3]The required shared library is libncurses5.

[4]See source code `bin/do-all-steps.sh`.

[5]One compilation step calls the script "config_writes.py". This script failed with a syntax error, as it contained incomplete python code. The scripts config_writes.py and config_writes.sh are generated by gen_config_writes.py. The output of the script gen_config_writes.py depends on the content of config_writes.txt. That file is generated by the simulation "xsim". The file "SimpleSumeSwitch_tb.sv" contains code that is responsible for writing config_writes.txt and uses a function named axi4_lite_master_write_request_control for generating the output. This in turn is dependent on the output of a script named gen_testdata.py.

[6]F.i. "CRITICAL WARNING: [BD 41-737] Cannot set the parameter TRANSLATION_MODE on /axi_interconnect_0. It is read-only." is a non critical warning.

[7]While using version 2018.2, the following message was printed: "WARNING: command 'get_user_parameter' will be removed in the 2015.3 release, use 'get_user_parameters' instead".

[8]Frames with an MTU greater than 1500 bytes.

Our P4 source code required contains Xilinx annotations[9] that define the maximum packet size in bits. We observed two different errors on the output packet, if the incoming packets exceeds the specified size:

- The output packet is longer then the original packet.
- The output packet is corrupted.

While most of the P4 language is supported on the netpfga, some key techniques are missing or not supported.

- Analysing / accessing payload is not supported
- Checksum computation over payload is not supported
- Using LPM tables can lead to compilation errors
- Depening on the match type, only certain table sizes are allowed

Renaming variables in the declaration of the parser or deparser lead to compilation errors. Function syntax is not supported. For this reason our implementation uses `#define` statements instead of functions.
FIXME:
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 Needed to debug internal parsing errors debugging generated tcl code to debug impl1 error

4.2 Software based NAT64

with Tayga and Jool Both cpu bound.
During the benchmark cpu bound, single thread tayga: Single threaded easy to use
Jool kernel module 100% cpu usage on 1 core for udp 0% visible cpu usage for tcp, might be tcp offloading Integration with iptables Requires routing

4.3 NAT64 Benchmarks - FIXME: explain numbers

4.3.1 Benchmark Design

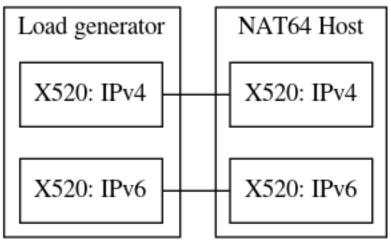

Figure 4.3: Benchmark design for NAT64 in software implementations

We use two hosts for performing benchmarks: a load generator and a NAT64 translator. Both hosts are equipped with a dual port Intel X520 10 Gbit/s network card. Both hosts are connected using DAC without any equipment in between. TCP offloading is enabled in the X520 cards. Figure 4.3 shows the network setup. When testing the NetPFGA/P4 performance, the X520 cards in the NAT64 translator are disconnected and instead the NetPFGA ports are connected, as show in figure 4.4. The load generator is equipped with a quad core CPU (Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz), enabled with hyperthreading and 16 GB RAM. The NAT64 translator

[9] F.i. “@Xilinx_MaxPacketRegion(1024)”

is also equipped with a quard core CPU (Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz) and 16 GB RAM. The first 10 seconds of the benchmark are excluded to avoid the TCP warm up phase.[10]

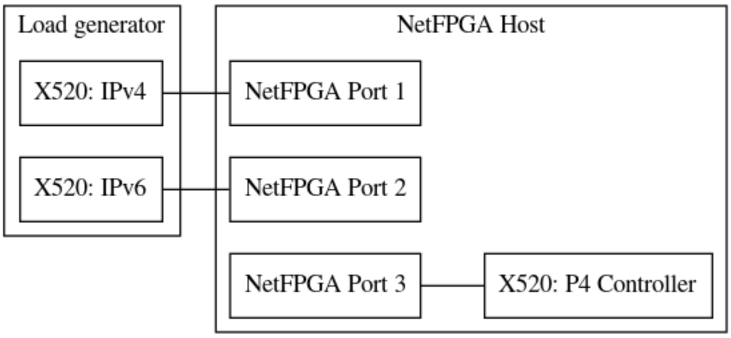

Figure 4.4: NAT64 with NetFPGA benchmark

We successfully implemented P4 code to realise NAT64 [52]. It contains parsers for all related protocols (ipv6, ipv4, udp, tcp, icmp, icmp6, ndp, arp), supports EAMT as defined by RFC7757 [3] and is feature equivalent to the two compared software solutions tayga [31] and jool [33]. Due to limitations in the P4 environment of the NetFPGA [**?**] environment, the BMV2 implementation is more feature rich. Table **??** summarises the achieved bandwidths of the NAT64 solutions.

Implementation	min/avg/max in Gbit/s			
Tayga	2.79 / 3.20 / 3.43	3.34 / 3.36 / 3.38	2.57 / 3.02 / 3.27	2.35 / 2.91 / 3.20
Jool	8.22 / 8.22 / 8.22	8.21 / 8.21 / 8.22	8.21 / 8.23 / 8.25	8.21 / 8.23 / 8.25
P4 / NetPFGA	9.28 / 9.28 / 9.29	9.28 / 9.28 / 9.29	9.28 / 9.28 / 9.29	9.28 / 9.28 / 9.29
Parallel connections	1	10	20	50

Table 4.3: IPv6 to IPv4 TCP NAT64 Benchmark

During the benchmarks the client – CPU usage

Implementation	min/avg/max in Gbit/s			
Tayga	2.90 / 3.15 / 3.34	2.87 / 3.01 / 3.22	2.68 / 2.85 / 3.09	2.60 / 2.78 / 2.88
Jool	7.18 / 7.56 / 8.24	7.97 / 8.05 / 8.09	8.05 / 8.08 / 8.10	8.10 / 8.12 / 8.13
P4 / NetPFGA	8.51 / 8.53 / 8.55	9.28 / 9.28 / 9.29	9.29 / 9.29 / 9.29	9.28 / 9.28 / 9.29
Parallel connections	1	10	20	50

Table 4.4: IPv4 to IPv6 TCP NAT64 Benchmark

UDP load generator hitting 100% cpu at P20. TCP confirmed. Over bandwidth results
Feature comparison speed - sessions - eamt can act as host lpm tables ping ping6 support ndp controller support
netpfga consistent

[10] iperf -O 10 parameter, see section 3.2.

Implementation	avg bandwidth in gbit/s / avg loss / adjusted bandwith			
Tayga	8.02 / 70% / 2.43	9.39 / 79% / 1.97	15.43 / 86% / 2.11	19.27 / 91% 1.73
Jool	6.44 / 0% / 6.41	6.37 / 2% / 6.25	16.13 / 64% / 5.75	20.83 / 71% / 6.04
P4 / NetPFGA	8.28 / 0% / 8.28	9.26 / 0% / 9.26	16.15 / 0% / 16.15	15.8 / 0% / 15.8
Parallel connections	1	10	20	50

Table 4.5: IPv6 to IPv4 UDP NAT64 Benchmark

Implementation	avg bandwidth in gbit/s / avg loss / adjusted bandwith			
Tayga	6.78 / 84% / 1.06	9.58 / 90% / 0.96	15.67 / 91% / 1.41	20.77 / 95% / 1.04
Jool	4.53 / 0% / 4.53	4.49 / 0% / 4.49	13.26 / 0% / 13.26	22.57 / 0% / 22.57
P4 / NetPFGA	7.04 / 0% / 7.04	9.58 / 0% / 9.58	9.78 / 0% / 9.78	14.37 / 0% / 14.37
Parallel connections	1	10	20	50

Table 4.6: IPv4 to IPv6 UDP NAT64 Benchmark

Chapter 5

Conclusion and Outlook

The objective of implementing high speed NAT64 in P4 has been achieved. The implementation at hand has been shown to be portable between 2 different P4 targets. It should be portable with minor target specific changes to faster hardware to support NAT64 at much higher line speeds, without any logic changes.
Our algorithm uses the IPv4-Compatible IPv6 Address [24] to embed IPv4 addresses. However RFC6052 [8] defines different embeddings depending on the prefix size. A future version should support these schemes to be compatible to other implementations.
PMTU handling error cases No fragmentation No address / mac learning
supporting migration to IPv6 only networks
P4 has been proven for us as a suitable programming language for network equipment with great potential. However in the current state the tooling and frameworks are still immature and need significant work to become usable for solving day-to-day challenges or supporting large scale projects. Even with the current state drawbacks, P4 is a very convincing language that has wide range of applications due to its protocol independence and easy to understand architecture.
The NetFGPA platform is a good showcase for the capabilities of P4, demonstrating almost line speed P4 programs. However the supporting code immaturity, logging ambiguity and enormous complexity of the development process.
Very time intensive development due to usability problems and uncertainty of functionality (compare sections 4.1.2 and 4.1.2).
While the port to NetPFGA was significantly more effort then expected, the learnings of the different layers were very much appreciated / liked
The availability of protocol independent programmable network equipment opens up many possibilities for in network programming. While this thesis focused on NAT64, the accompying technologie DNS64 [7] could also be implemented in P4, thus completing the translation mechanism.
Proxies / higher level protocols could be next level
Add helper in P4 to support checksum analysis a frequent problem and helper Allow ICMP6 option parsing: specify xtimes 64 bit blocks resulting in an array
Adding support for passing on meta information to controller: key or table
Support a meta language to define used types and/or export to popular languages. Long term supporting python3 would be helpful. P4OS.
- react on FIN/RST (?) – could be an addition

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 card)

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
git clone https://gitlab.ethz.ch/nsg/student-projects/ma-2019-19_high_speed_nat64_with_p4
```

It can be browsed online on `https://gitlab.ethz.ch/nicosc/master-thesis` and on `https://gitlab.ethz.ch/nsg/student-projects/ma-2019-19_high_speed_nat64_with_p4`.

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. 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 Compile Flow

C.3 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.3.1 Test 1: IPv4 egress

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.3.2 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.3.3 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 425407664113623819
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 425407664113623819
```

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 excerpts of log files that might be relevant for reproducing the work.

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.

D.5 NetFPGA Kernel module

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%
```

```
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 00000000000000000000000000010000 >
$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%
```

```
[7:05] rainbow:netpfga% bash  build-load-drivers.sh
+ cd /home/nico/projects/P4-NetFPGA/lib/sw/std/driver/sume_riffa_v1_0_0
+ make all
make -C /lib/modules/5.0.0-16-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-5.0.0-16-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-5.0.0-16-generic'
+ sudo make install
make -C /lib/modules/5.0.0-16-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-5.0.0-16-generic'
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-5.0.0-16-generic'
install -o root -g root -m 0755 -d /lib/modules/5.0.0-16-generic/extra/sume_riffa/
install -o root -g root -m 0755 sume_riffa.ko /lib/modules/5.0.0-16-generic/extra/sume_riffa/
depmod -a 5.0.0-16-generic
+ sudo modprobe sume_riffa
modprobe: ERROR: could not insert 'sume_riffa': Exec format error
[7:06] rainbow:netpfga%
```

```
java traceback when trying to install sdnet
#+BEGIN_CENTER
Exception in thread "AWT-EventQueue-0" java.lang.IllegalArgumentException: Window must not be zero
at java.desktop/sun.awt.X11.XAtom.checkWindow(Unknown Source)
at java.desktop/sun.awt.X11.XAtom.getAtomData(Unknown Source)
at java.desktop/sun.awt.X11.XToolkit.getWorkArea(Unknown Source)
at java.desktop/sun.awt.X11.XToolkit.getInsets(Unknown Source)
at java.desktop/sun.awt.X11.XToolkit.getScreenInsets(Unknown Source)
at java.desktop/java.awt.Window.init(Unknown Source)
at java.desktop/java.awt.Window.<init>(Unknown Source)
at java.desktop/java.awt.Window.<init>(Unknown Source)
at java.desktop/java.awt.Dialog.<init>(Unknown Source)
at java.desktop/java.awt.Dialog.<init>(Unknown Source)
at java.desktop/javax.swing.JDialog.<init>(Unknown Source)
at java.desktop/javax.swing.JOptionPane.createDialog(Unknown Source)
at java.desktop/javax.swing.JOptionPane.createDialog(Unknown Source)
at j.a.c(Unknown Source)
at j.a.a(Unknown Source)
at j.a.a(Unknown Source)
at j.a.c(Unknown Source)
at com.xilinx.installer.gui.panel.destination.b.a(Unknown Source)
at com.xilinx.installer.gui.panel.destination.DestinationPanel.z(Unknown Source)
at com.xilinx.installer.gui.E.a(Unknown Source)
at com.xilinx.installer.gui.InstallerGUI.l(Unknown Source)
at com.xilinx.installer.gui.i.actionPerformed(Unknown Source)
at java.desktop/javax.swing.AbstractButton.fireActionPerformed(Unknown Source)
at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(Unknown Source)
at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(Unknown Source)
at java.desktop/javax.swing.DefaultButtonModel.setPressed(Unknown Source)
at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(Unknown Source)
at java.desktop/java.awt.Component.processMouseEvent(Unknown Source)
at java.desktop/javax.swing.JComponent.processMouseEvent(Unknown Source)
at java.desktop/java.awt.Component.processEvent(Unknown Source)
at java.desktop/java.awt.Container.processEvent(Unknown Source)
at java.desktop/java.awt.Component.dispatchEventImpl(Unknown Source)
at java.desktop/java.awt.Container.dispatchEventImpl(Unknown Source)
at java.desktop/java.awt.Component.dispatchEvent(Unknown Source)
at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Unknown Source)
at java.desktop/java.awt.LightweightDispatcher.processMouseEvent(Unknown Source)
at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Unknown Source)
at java.desktop/java.awt.Container.dispatchEventImpl(Unknown Source)
at java.desktop/java.awt.Window.dispatchEventImpl(Unknown Source)
at java.desktop/java.awt.Component.dispatchEvent(Unknown Source)
at java.desktop/java.awt.EventQueue.dispatchEventImpl(Unknown Source)
at java.desktop/java.awt.EventQueue.access$500(Unknown Source)
at java.desktop/java.awt.EventQueue$3.run(Unknown Source)
at java.desktop/java.awt.EventQueue$3.run(Unknown Source)
at java.base/java.security.AccessController.doPrivileged(Native Method)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(Unknown Source)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(Unknown Source)
at java.desktop/java.awt.EventQueue$4.run(Unknown Source)
at java.desktop/java.awt.EventQueue$4.run(Unknown Source)
at java.base/java.security.AccessController.doPrivileged(Native Method)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(Unknown Source)
at java.desktop/java.awt.EventQueue.dispatchEvent(Unknown Source)
at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown Source)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown Source)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.desktop/java.awt.EventDispatchThread.pumpEvents(Unknown Source)
at java.desktop/java.awt.EventDispatchThread.run(Unknown Source)

#+END_CENTER

Reason was a hidden window.
```

Testing the card

```
------------------------------
[ddr3B]: Running Auto Test
------------------------------
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/wx-3.0-gtk2/wx/_core.py", line 16765, in <lambda>
    lambda event: event.callable(*event.args, **event.kw) )
  File "sw/host/script/NfSumeTest.py", line 848, in UpdateProgress
    self.progressDlg.Update(self.curProgress, str(localLine))
  File "/usr/lib/python2.7/dist-packages/wx-3.0-gtk2/wx/_core.py", line 16710, in __getattr__
    raise PyDeadObjectError(self.attrStr % self._name)
wx._core.PyDeadObjectError: The C++ part of the NfSumeProgress object has been deleted, attribute access no longer allowed.
Exception in thread Thread-18:
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.py", line 801, in __bootstrap_inner
    self.run()
  File "sw/host/script/NfSumeTest.py", line 947, in run
    self.target(*self.data)
  File "sw/host/script/NfSumeTest.py", line 355, in StartAutoTest
    self.TestInterface(testName)
  File "sw/host/script/NfSumeTest.py", line 465, in TestInterface
    self.ProgramFpga('../../../bitfiles/' + self.nfSumeTestConfiguration[testName]['bitstream'])
  File "sw/host/script/NfSumeTest.py", line 586, in ProgramFpga
    self.getFpgaIndex()
  File "sw/host/script/NfSumeTest.py", line 574, in getFpgaIndex
    p = Popen(['djtgcfg', 'init', '-d', 'NetSUME'], stdout=PIPE, bufsize = 1)
  File "/usr/lib/python2.7/subprocess.py", line 711, in __init__
    errread, errwrite)
  File "/usr/lib/python2.7/subprocess.py", line 1343, in _execute_child
    raise child_exception
OSError: [Errno 2] No such file or directory


------------------------------
[pcie]: Running Auto Test
------------------------------
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/wx-3.0-gtk2/wx/_core.py", line 16765, in <lambda>
    lambda event: event.callable(*event.args, **event.kw) )
  File "sw/host/script/NfSumeTest.py", line 848, in UpdateProgress
    self.progressDlg.Update(self.curProgress, str(localLine))
  File "/usr/lib/python2.7/dist-packages/wx-3.0-gtk2/wx/_core.py", line 16710, in __getattr__
    raise PyDeadObjectError(self.attrStr % self._name)
wx._core.PyDeadObjectError: The C++ part of the NfSumeProgress object has been deleted, attribute access no longer allowed.
Exception in thread Thread-21:
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.py", line 801, in __bootstrap_inner
    self.run()
  File "sw/host/script/NfSumeTest.py", line 947, in run
    self.target(*self.data)
  File "sw/host/script/NfSumeTest.py", line 466, in TestInterface
    self.serialCon.readlines()
  File "/usr/lib/python2.7/dist-packages/serial/serialposix.py", line 495, in read
    raise SerialException('device reports readiness to read but returned no data (device disconnected or multiple access on port?)')
SerialException: device reports readiness to read but returned no data (device disconnected or multiple access on port?)
```

Another generated file problem:

```
nico@nsg-System:~/master-thesis/netpfga$ grep -i error $P4_PROJECT_DIR/nf_sume_sdnet_ip/SimpleSumeSwitch/LOG
ERROR: [VRFC 10-1491] unexpected EOF [/home/nico/master-thesis/netpfga/minip4/nf_sume_sdnet_ip/SimpleSumeSwitch/S_CONTROLLERs.HDL/S_CONTROLLER_SimpleSumeSwitch.vp:37]
INFO: [VRFC 10-311] analyzing module TopDeparser_t_EngineStage_0_ErrorCheck
INFO: [VRFC 10-311] analyzing module TopDeparser_t_EngineStage_1_ErrorCheck
INFO: [VRFC 10-311] analyzing module TopDeparser_t_EngineStage_2_ErrorCheck
INFO: [VRFC 10-311] analyzing module TopDeparser_t_EngineStage_3_ErrorCheck
INFO: [VRFC 10-311] analyzing module TopDeparser_t_EngineStage_4_ErrorCheck
INFO: [VRFC 10-311] analyzing module TopDeparser_t_EngineStage_5_ErrorCheck
INFO: [VRFC 10-311] analyzing module TopDeparser_t_EngineStage_6_ErrorCheck
```

function syntax not supported

```
make[1]: Entering directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/src'
p4c-sdnet -o minip4.sdnet -sdnet_info .sdnet_switch_info.dat minip4_solution.p4
headers.p4(246):syntax error, unexpected IDENTIFIER, expecting (
bit<16> ones_complement_sum
        ^^^^^^^^^^^^^^^^^^^
error: 1 errors encountered, aborting compilation
Makefile:34: recipe for target 'all' failed
make[1]: *** [all] Error 1
make[1]: Leaving directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/src'
Makefile:31: recipe for target 'frontend' failed
make: *** [frontend] Error 2
nico@nsg-System:~/master-thesis/netpfga$



nico@nsg-System:~/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/sim_switch_default$ cd $NF_DESIGN_DIR/test/sim_switch_
rm -f config_writes.py*
rm -f *.pyc
cp /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/testdata/config_writes.py ./
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/testdata/config_writes.py': No such file or directory
Makefile:36: recipe for target 'all' failed
make: *** [all] Error 1


Finished scanning sources
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1700] Loaded user IP repository '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw/ip_repo'.
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository '/opt/Xilinx/Vivado/2018.2/data/ip'.
WARNING: [IP_Flow 19-3664] IP 'bd_7ad4_xpcs_0' generated file not found '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_
WARNING: [IP_Flow 19-3664] IP 'bd_a1aa_xpcs_0' generated file not found '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_
open_project: Time (s): cpu = 00:00:05 ; elapsed = 00:00:05 . Memory (MB): peak = 1365.715 ; gain = 188.977 ; free physical = 9396 ; free virtual = 15104
# puts "\nOpening $design Implementation design\n"


WARNING: [Synth 8-689] width (12) of port connection 'control_S_AXI_ARADDR' does not match port width (8) of module 'SimpleSumeSwitch' [/home/nico/projects/P4-NetFPGA/
ERROR: [Synth 8-448] named port connection 'tuple_out_sume_metadata_VALID' does not exist for instance 'SimpleSumeSwitch_inst' of module 'SimpleSumeSwitch' [/home/nico
ERROR: [Synth 8-448] named port connection 'tuple_out_sume_metadata_DATA' does not exist for instance 'SimpleSumeSwitch_inst' of module 'SimpleSumeSwitch' [/home/nico/
ERROR: [Synth 8-6156] failed synthesizing module 'nf_sume_sdnet' [/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/
ERROR: [Synth 8-6156] failed synthesizing module 'nf_sume_sdnet_ip' [/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_swit
ERROR: [Synth 8-6156] failed synthesizing module 'nf_datapath' [/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw
ERROR: [Synth 8-6156] failed synthesizing module 'top' [/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw/hdl/top
```

Missing “souce” files:

```
cc -c -fPIC /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/sw/API/CAM.c -I/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-swit
cc -std=c99 -Wall -Werror -fPIC -c libcam.c -I/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/sw/sume -I/home/nico/projects/P4-NetFPGA/contrib-projects/su
cc -L/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/sw/sume -shared -o libcam.so libcam.o CAM.o -lsumereg
/usr/bin/ld: cannot find -lsumereg
collect2: error: ld returned 1 exit status
Makefile:52: recipe for target 'libcam' failed
make[1]: *** [libcam] Error 1
make[1]: Leaving directory '/home/nico/master-thesis/netpfga/minip4/sw/CLI'
ERROR: could not compile libcam souce files
```

Generated files not found:

```
make: Leaving directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test'
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/hw/Makefile': No such file or directory
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 directo
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 directo
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 or di
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 directo
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 directo
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 or di
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 directo
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 directo
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 or di
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 directo
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 directo
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 or di
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 directory
cp: cannot stat '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch/test/Makefile': 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.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
NetFPGA environment:
   Root dir:       /home/nico/projects/P4-NetFPGA
   Project name:   simple_sume_switch
   Project dir:    /tmp/nico/test/simple_sume_switch
   Work dir:       /tmp/nico
512
=== Work directory is /tmp/nico/test/simple_sume_switch
=== Setting up test in /tmp/nico/test/simple_sume_switch/sim_switch_default
=== 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/simp
+ date
Die Jul 23 13:34:54 CEST 2019
+ [  = no ]
+ cd /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/simple_sume_switch
+ make
make: *** No targets specified and no makefile found.  Stop.
```

Renaming variables breaks the compile process

```
 @Xilinx_MaxPacketRegion(1024)
 control TopDeparser(
-    packet_out b,
-    in Parsed_packet p,
+    packet_out packet,
+    in Parsed_packet hdr,
     in user_metadata_t user_metadata,
     inout digest_data_t digest_data,
     inout sume_metadata_t sume_metadata) {

     apply {
-        b.emit(p.ethernet);
+        packet.emit(hdr.ethernet);
     }
+
+
 }
```

LPM size must be != 64

```
minip4_solution.p4(38): [-Wwarn=uninitialized_out_param] warning: out parameter meta may be uninitialized when RealParser terminates
    out metadata meta,
                 ^^^^
minip4_solution.p4(35)
parser RealParser(
       ^^^^^^^^^^
error: LPM table size should be 2^n - 1
actions_nat64_generic.p4(169): error: could not not map table size size
        size = 64;
        ^^^^
error: table match_types are not the same
actions_arp.p4(35): error: could not map table key(s) KeyElement
            hdr.arp.dst_ipv4_addr: lpm;
            ^^^^^^^^^^^^^^^^^^^^^
error: LPM table size should be 2^n - 1
actions_arp.p4(55): error: could not not map table size size
        size = 64;
        ^^^^
Makefile:34: recipe for target 'all' failed
make[1]: *** [all] Error 1
make[1]: Leaving directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/src'
Makefile:31: recipe for target 'frontend' failed
make: *** [frontend] Error 2
nico@nsg-System:~/master-thesis/netpfga/log$
```

LIMIT table match types are not the same error

```
make[1]: Entering directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/src'
p4c-sdnet -o minip4.sdnet -sdnet_info .sdnet_switch_info.dat minip4_solution.p4
actions_egress.p4(52): warning: Table v6_networks is not used; removing
table v6_networks {
      ^^^^^^^^^^^
actions_egress.p4(69): warning: Table v4_networks is not used; removing
```

```
table v4_networks {
      ^^^^^^^^^^^
actions_nat64_generic.p4(174): warning: Table nat46 is not used; removing
    table nat46 {
          ^^^^^
minip4_solution.p4(38): [-Wwarn=uninitialized_out_param] warning: out parameter meta may be uninitialized when RealParser terminates
    out metadata meta,
                 ^^^^
minip4_solution.p4(35)
parser RealParser(
       ^^^^^^^^^^
error: table match_types are not the same
actions_arp.p4(35): error: could not map table key(s) KeyElement
            hdr.arp.dst_ipv4_addr: lpm;
            ^^^^^^^^^^^^^^^^^^^^^
Makefile:34: recipe for target 'all' failed
make[1]: *** [all] Error 1

    table v4_arp {
        key = {
            hdr.ethernet.dst_addr: exact;
            hdr.arp.opcode: exact;
            hdr.arp.dst_ipv4_addr: lpm;
        }
        actions = {
            controller_debug_table_id;
            arp_reply;
            NoAction;
        }
        size = ICMP6_TABLE_SIZE;
        default_action = controller_debug_table_id(TABLE_ARP);
    }
```

Implicit error saying that LPM tables don't work:

```
s/sume-sdnet-switch/projects/minip4/nf_sume_sdnet_ip/SimpleSumeSwitch/realmain_lookup_table_0_t.HDL/xpm_memory.sv
[SW] LPM_Init() - start
[SW] LPM_Init() - done
[SW] LPM_LoadDataset() - start
[SW] LPM_LoadDataset() failed with error code = 12
FATAL_ERROR: Vivado Simulator kernel has encounted an exception from DPI C function: LPM_VerifyDataset(). Please correct.
Time: 2016466 ps  Iteration: 0  Process: /SimpleSumeSwitch_tb/LPM_VerifyDataset
  File: /home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/nf_sume_sdnet_ip/SimpleSumeSwitch/Testbench/SimpleSumeSwitch_tb.sv


minip4_solution.p4(35)
parser RealParser(
       ^^^^^^^^^^
actions_nat64_generic.p4(173): error: table size too small for match_type(EM): 63 < 64
        size = 63;
               ^^
actions_nat64_generic.p4(173): error: could not not map table size size
        size = 63;
        ^^^^
```

Unsupported default parameters

```
actions_egress.p4(89): error: data-plane arguments in default_actions are currently unsupported: realmain_controller_debug_table_id_0
    default_action = controller_debug_table_id(TABLE_V4_NETWORKS);
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
terminate called after throwing an instance of 'Util::CompilerBug'
  what():  In file: /wrk/hdscratch/staff/mohan/p4c_sdnet/build/p4c/extensions/sdnet/translate/core/lookupEngine.cpp:137
Compiler Bug: actions_egress.p4(89): unhandled expression realmain_controller_debug_table_id/realmain_controller_debug_table_id_0(5);
    default_action = controller_debug_table_id(TABLE_V4_NETWORKS);
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Compiler Bug / ifstatement

```
minip4_solution.p4(39)
parser RealParser(
       ^^^^^^^^^^
terminate called after throwing an instance of 'Util::CompilerBug'
  what():  In file: /wrk/hdscratch/staff/mohan/p4c_sdnet/build/p4c/extensions/sdnet/writers/pxWriter.h:20
Compiler Bug: unhandled node: <IfStatement>(471564)

Makefile:34: recipe for target 'all' failed
make[1]: *** [all] Error 134
make[1]: Leaving directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/src'
Makefile:31: recipe for target 'frontend' failed
```

Applying table "twice" in different branches is impossible (another compiler bug)

```
make -C src/
make[1]: Entering directory '/home/nico/projects/P4-NetFPGA/contrib-projects/sume-sdnet-switch/projects/minip4/src'
p4c-sdnet -o minip4.sdnet -sdnet_info .sdnet_switch_info.dat minip4_solution.p4
minip4_solution.p4(19): [-Wwarn=uninitialized_out_param] warning: out parameter meta may be uninitialized when RealParser terminates
    out metadata meta,
                 ^^^^
minip4_solution.p4(16)
parser RealParser(
       ^^^^^^^^^^
terminate called after throwing an instance of 'Util::CompilerBug'
  what():  In file: /wrk/hdscratch/staff/mohan/p4c_sdnet/build/p4c/extensions/sdnet/translate/core/tupleEngine.cpp:324
Compiler Bug: overwrite

Makefile:34: recipe for target 'all' failed
```

Adding entries requires setting all parameters

```
» table_cam_add_entry realmain_v6_networks_0 realmain.set_egress_port 42540766411362381960998550477184434178 => 1
ERROR: not enough fields provided to complete _hexify()
```

Broken code that cannot convret long to int:

```
» table_cam_delete_entry realmain_v6_networks_0 42540766411362381960998550477184434179
ERROR: failed to convert 42540766411362381960998550477184434179 of type <type 'long'> to an integer
nico@nsg-System:~/master-thesis/netpfga/minip4/sw/CLI$
```

F.2 P4 error messages

```
Warning: you requested the nanomsg event logger, but bmv2 was compiled without -DBMELOG, and the event logger cannot be activated
Calling target program-options parser
[14:01:44.334] [bmv2] [D] [thread 23356] Set default default entry for table 'MyIngress.icmp6': MyIngress.controller_debug_table_id - 2,
[14:01:44.341] [bmv2] [D] [thread 23356] Set default default entry for table 'MyIngress.nat64': MyIngress.controller_debug_table_id - 1,
[14:01:44.344] [bmv2] [D] [thread 23356] Set default default entry for table 'tbl_act': act -
[14:01:44.345] [bmv2] [D] [thread 23356] Set default default entry for table 'tbl_act_0': act_0 -
[14:01:44.345] [bmv2] [D] [thread 23356] Set default default entry for table 'tbl_nat64_icmp6_generic': MyIngress.nat64_icmp6_generic -
[14:01:44.345] [bmv2] [D] [thread 23356] Set default default entry for table 'tbl_act_1': act_1 -
[14:01:44.345] [bmv2] [D] [thread 23356] Set default default entry for table 'tbl_act_2': act_2 -
[14:01:44.345] [bmv2] [D] [thread 23356] Set default default entry for table 'MyIngress.v4_networks': MyIngress.controller_debug_table_id - 5,
[14:01:44.345] [bmv2] [D] [thread 23356] Set default default entry for table 'MyIngress.v6_networks': MyIngress.controller_debug_table_id - 3,
[14:01:44.346] [bmv2] [D] [thread 23356] Set default default entry for table 'tbl_act_3': act_3 -
Invalid entry type 'expression' in field list
bad json:
{
   "type" : "expression",
   "value" : {
      "type" : "expression",
      "value" : {
         "left" : null,
         "op" : "d2b",
         "right" : {
            "type" : "field",
            "value" : [ "scalars", "metadata.chk_icmp6_na_ns" ]
         }
      }
   }
}
```

```
../p4src/static-mapping.p4(121): error: MyIngress.nat64, Multiple LPM keys in table
    table nat64 {
          ^^^^^
Compilation Error

    table nat64 {
        key = {
            hdr.ipv6.src_addr: lpm;
            hdr.ipv6.dst_addr: lpm;
        }
        actions = {
            controller_debug;
            nat64_static;
            NoAction;
        }
        size = NAT64_TABLE_SIZE;
        default_action = controller_debug;
    }
```

```
../p4src/static-mapping.p4(60): error: SwitchStatement: switch statements not allowed in actions
        switch(hdr.icmp6.type) {
        ^^^^^^
```

No if in actions:

```
../p4src/static-mapping.p4(57): error: MethodCallStatement: Conditional execution in actions is not supported on this target
        hdr.icmp.setValid();
        ^^^^^^^^^^^^^^^^^^^
../p4src/static-mapping.p4(70): error: MethodCallStatement: Conditional execution in actions is not supported on this target
        hdr.icmp6.setInvalid();
        ^^^^^^^^^^^^^^^^^^^^^^
../p4src/static-mapping.p4(73): error: MethodCallStatement: Conditional execution in actions is not supported on this target
        hdr.icmp6_na_ns.setInvalid();
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
../p4src/static-mapping.p4(74): error: MethodCallStatement: Conditional execution in actions is not supported on this target
        hdr.icmp6_option_link_layer_addr.setInvalid();
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Compilation Error
p4@ubuntu:~/master-thesis/p4app$

        if(hdr.ipv6.next_header == PROTO_ICMP6) {
            nat64_icmp6();
        }
```

```
p4c -target bmv2 -arch v1model -std p4-16 "../p4src/checksum_diff.p4" -o "/home/p4/master-thesis/p4src"
In file: /home/p4/p4-tools/p4c/backends/bmv2/common/expression.cpp:168
Compiler Bug: ../p4src/actions_delta_checksum.p4(60): ones_complement_sum(hdr.udp.checksum, tmp);: unhandled case
    tmp = ones_complement_sum(hdr.udp.checksum, meta.v6sum);
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Compilation Error"`
Using the following code:
"`/* copied from
https://p4.org/p4-spec/docs/PSA-v1.1.0.html#appendix-internetchecksum-implementation
*/

bit<16> ones_complement_sum(in bit<16> x, in bit<16> y) {
    bit<17> ret = (bit<17>) x + (bit<17>) y;
    if (ret[16:16] == 1) {
        ret = ret + 1;
    }
    return ret[15:0];
}"`
And p4c version:
"`p4@ubuntu:~/master-thesis/p4app$ p4c -version
p4c 0.5 (SHA: 5ae30ee)"`
```

F.3 Traces

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

F.3.1 P4/BMV NAT64 Delta based traces

```
*** DONE 2019-07-21: Proof of v6->v4 working delta based
    CLOSED: [2019-07-21 Sun 12:30]
#+BEGIN_CENTER
 pcap/tcp-udp-delta-from-v6-2019-07-21-0853-h1.pcap | Bin 0 -> 4252 bytes
 pcap/tcp-udp-delta-from-v6-2019-07-21-0853-h3.pcap | Bin 0 -> 2544 bytes
#+END_CENTER


 create mode 100644 pcap/tcp-udp-delta-2019-07-17-1555-h1.pcap
 create mode 100644 pcap/tcp-udp-delta-2019-07-17-1555-h3.pcap
 create mode 100644 pcap/tcp-udp-delta-2019-07-17-1557-h1.pcap
 create mode 100644 pcap/tcp-udp-delta-2019-07-17-1558-h3.pcap
```

F.3.2 P4/NetFPGA NAT64 Delta based traces

```
**** DONE Testing v4->v6 tcp: ok (version 10.0)
     CLOSED: [2019-08-04 Sun 09:15]
#+BEGIN_CENTER
nico@ESPRIMO-P956:~/master-thesis/bin$ ./socat-connect-tcp-v4
+ echo from-v4-ok
+ socat - TCP:10.0.0.66:2345
TCPv6-ok
nico@ESPRIMO-P956:~/master-thesis/bin$ ./socat-listen-tcp-v6
from-v4-ok

#+END_CENTER

trace:
netfpga-nat64-2019-08-04-0907-enp2s0f0.pcap
netfpga-nat64-2019-08-04-0907-enp2s0f1.pcap

**** DONE Testing v4->v6 udp: ok (version 10.1)
trace:
 create mode 100644 pcap/netfpga-nat64-udp-2019-08-04-0913-enp2s0f0.pcap
 create mode 100644 pcap/netfpga-nat64-udp-2019-08-04-0913-enp2s0f1.pcap
```

Bigger packets

```
*** DONE 2019-08-04: version 10.1/10.2: new maxpacketregion: v4->v6 works
    CLOSED: [2019-08-04 Sun 19:42]
#+BEGIN_CENTER
nico@ESPRIMO-P956:~/master-thesis/bin$ ./init_ipv4_esprimo.sh
nico@ESPRIMO-P956:~/master-thesis/bin$ ./set_ipv4_neighbor.sh

#+END_CENTER

Test 20 first:

- Does't work -> missed to add table entries
- Does work after setting table entries
- 300 works
- 1450 works
- 1500 does not work

Proof:

 create mode 100644 pcap/netfpga-10.2-maxpacket-2019-08-04-1931-enp2s0f0.pcap
 create mode 100644 pcap/netfpga-10.2-maxpacket-2019-08-04-1931-enp2s0f1.pcap


*** DONE 2019-08-04: test v6 -> v4: works for 1420
    CLOSED: [2019-08-04 Sun 20:30]

Proof:
#+BEGIN_CENTER
 create mode 100644 pcap/netfpga-10.2-fromv6tov4-2019-08-04-1943-enp2s0f0.pcap
 create mode 100644 pcap/netfpga-10.2-fromv6tov4-2019-08-04-1943-enp2s0f1.pcap
```

F.4 Introduction

F.4.1 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
```

F.5 P4 notes

F.5.1 Key retrieval chat log

```
Key and mask for matching destination is in table. We need this
 information in the action. However this information is not exposed, so
 we need to specify another parameter with the same information as in
 the key(s).

 Log from slack: (2019-03-14)

 nico [1:55 PM]
 If I use LPM for matching, can I easily get the network address from P4 or do I have to use a bitmask myself? In the latter case it is not exactly clear how to get the ma

 Nate Foster [1:58 PM]
 You want to retrieve the address in the packet? In a table?
 And do you want to do the retrieving from the data plane or the control plane? (edited)

 nico [2:00 PM]
 If I have a match in a table that matches on LPM, it can be any IP address in a network
 For calculating the NAT64/NAT46 translation, I will need the base address, i.e. network address to do subtractions/additions
 So it is fully data plane, what I would like to do
 I'll commit sample code to show the use case more clearly
 https://gitlab.ethz.ch/nicosc/master-thesis/blob/master/p4src/static-mapping.p4#L73
 GitLab
 p4src/static-mapping.p4 · master · nicosc / master-thesis
 gitlab.ethz.ch
 So the action nat64_static() is used in the table v6_networks.
 In v6_networks I use a match on `hdr.ipv6.dst_addr: lpm;`
 What I would like to be able is to get the network address ; I can do that manually, if I have the mask
 I can also re-inject this parameter by another action argument, but I'd assume that I can somewhere read this out from the table / match

 Nate Foster [2:15 PM]
 To make sure I understand, in the data plane, you want to retrieve the address in the lpm pattern? (edited)

 nico [2:16 PM]
 I want to retrieve the key

 Nate Foster [2:16 PM]
 Wait. The value `hdr.ipv6.dst_addr` is the thing used in the match.
 So you have that.
 What you don't have is the IPv6 address and mask put into the table by the control plane.
 I assume you want the latter, right?

 nico [2:17 PM]
 For example, if my matching key is 2001:db8::/32 and the real address is 2001:db8::f00, then I would like to retrieve 2001:db8:: and 32 from the table
 exactly :slightly_smiling_face:
 I can "fix" this by adding another argument, but it feels somewhat wrong to do that
 Because the table already knows this information

 Nate Foster [2:26 PM]
 I can't think of a way other than the action parameter hack.

 nico [2:26 PM]
 Oh, ok
 Is it because the information is "lost in hardware"?

 Nate Foster [2:31 PM]
 No you're right that most implementations have the value in memory. And one can imagine a different table API that allowed one to retrieve it in the data plane.
 But unless I am missing something obvious, P4 hides it...
```

F.5.2 Table retrieval problem

```
Is there any meta information for "from which table was the action
called" available? My use case is having a debug action that sends
packets to the controller and I use it as a default_action in various
tables; however know I don't know anymore from which table the action
was called. Is there any kind of meta information which table called
me available?

I could work around this by using if(! .. .hit) { my_action(table_id)
}, but it would not work with using default_action = ...
```


```
Is there any meta information for "from which table was the action
called" available? My use case is having a debug action that sends
packets to the controller and I use it as a default_action in various
tables; however know I don't know anymore from which table the action
was called. Is there any kind of meta information which table called
me available?

I could work around this by using if(! .. .hit) { my_action(table_id)
}, but it would not work with using default_action = ...
```

F.5.3 Data definition redundancy

```
*** DONE Synchronisation with the controller
    - Double data type definition -> might differ
    - TYPE_CPU for ethernet
    - Port ingress offset (9 vs. 16 bit)
```

F.5.4 Python2 unicode issue

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

Fix:
from __future__ import unicode_literals
```

F.5.5 P4 OS

```
Not addressed so far: how to create re-usable code fragments that can
be plugged in easily. There could be a hypothetical "P4OS" that
manages code fragments. This might include, but not limited to
downloading (signed?) source code, managing dependencies similar to
Linux package management, handling updates, etc.
```

List of Abbrevations

ARP .... Address resolution protocol
ASIC .... Application-specific integrated circuit
DAC .... Direct attach cable
FGPA .... Field-programmable gate array
LPM .... Longes prefix matching
MTU .... Maximum transfer unit
NAT .... Network Address Translation
NAT64 .... Network Address Translation from / to IPv6 to / from IPv4
NDP .... Neighbor Discovery Protocol
RIR .... Regional Internet Registry
RTT .... Round Trip Time

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] M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147 (Proposed Standard), Apr. 2011.

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

[9] 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.

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

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

[12] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119 (Best Current Practice), Mar. 1997. Updated by RFC 8174.

[13] G. Camarillo, S. Srinivasan, R. Even, and J. Urpalainen. Conference Event Package Data Format Extension for Centralized Conferencing (XCON). RFC 6502 (Proposed Standard), Mar. 2012.

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

[15] 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.

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

[17] S. Deering, W. Fenner, and B. Haberman. Multicast Listener Discovery (MLD) for IPv6. RFC 2710 (Proposed Standard), Oct. 1999. Updated by RFCs 3590, 3810.

[18] 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.

[19] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu. iperf - the ultimate speed test tool for tcp, udp and sctp. https://iperf.fr/. Requested on 2019-08-19.

[20] 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.

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

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

[23] S. P. D. L. V. T. T. B. Hendrik Züllig. P4-programming on an fpga, semester thesis sa-2019-02. https://gitlab.ethz.ch/nsg/student-projects/sa-2019-02_p4_programming_sume_netfpga/blob/master/SA-2019-02.pdf.

[24] 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.

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

[26] G. Huston. Ipv4 address report. https://ipv4.potaroo.net/. Requested on 2019-08-18.

[27] G. Huston, A. Lord, and P. Smith. IPv6 Address Prefix Reserved for Documentation. RFC 3849 (Informational), July 2004.

[28] E. Juskevicius. Definition of IETF Working Group Document States. RFC 6174 (Informational), Mar. 2011.

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

[30] 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.

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

[32] J. McCann, S. Deering, J. Mogul, and R. Hinden (Ed.). Path MTU Discovery for IP version 6. RFC 8201 (Internet Standard), July 2017.

[33] N. Mexico. Jool an open source siit and nat64 for linux. https://www.jool.mx/en/index.html.

[34] J. Mogul and S. Deering. Path MTU discovery. RFC 1191 (Draft Standard), Nov. 1990.

[35] 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.

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

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

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

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

[40] D. Plummer. An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Internet Standard), Nov. 1982. Updated by RFCs 5227, 5494.

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

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

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

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

[45] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), Feb. 1996. Updated by RFC 6761.

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

[47] G. Rieger. socat - multipurpose relay. `http://www.dest-unreach.org/socat/`. Requested on 2019-08-19.

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

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

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

[51] N. Schottelius. Extern for checksum'ing payload (p4-netpfga-public). `https://github.com/NetFPGA/P4-NetFPGA-public/issues/13`.

[52] N. Schottelius. High speed nat64 in p4 (git repository). `https://gitlab.ethz.ch/nsg/student-projects/ma-2019-19_high_speed_nat64_with_p4`.

[53] N. Schottelius and S. Plocher. Implementation of a layer 7 ipv4 to ipv6 reverse proxy. Protected git repositry `https://gitlab.ethz.ch/nicosc/sdn-nat64/`, part of the Advanced topics in communication networks course fall 2019, `https://adv-net.ethz.ch/`, 2018.

[54] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), Jan. 2001.

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

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

[57] ungleich. The ungleich network infrastructure. `https://redmine.ungleich.ch/projects/open-infrastructure/wiki/The_ungleich_network_infrastructure`. Requested on 2019-08-18.

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

[59] Various. Should i use python 2 or python 3 for my development activity? `https://wiki.python.org/moin/Python2orPython3`. Requested on 2019-08-19.

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

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

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

[63] Wikipedia. Solicited-node multicast address. https://en.wikipedia.org/wiki/Solicited-node_multicast_address. Requested on 2019-08-13.