

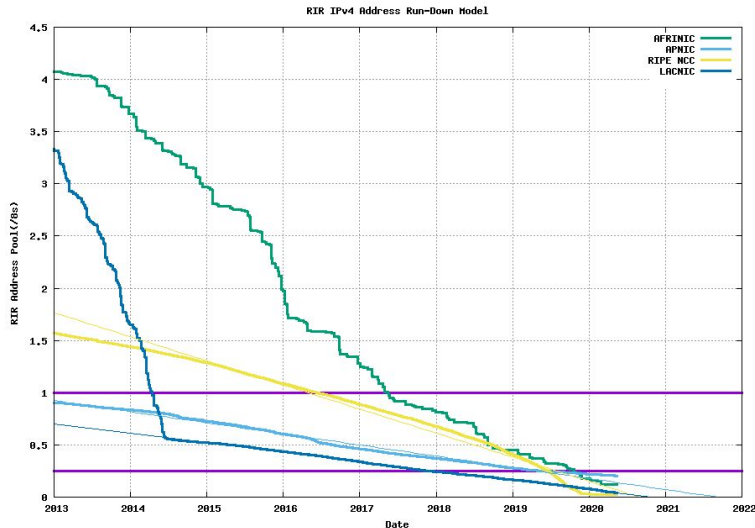
# HIGH SPEED NAT64 WITH P4



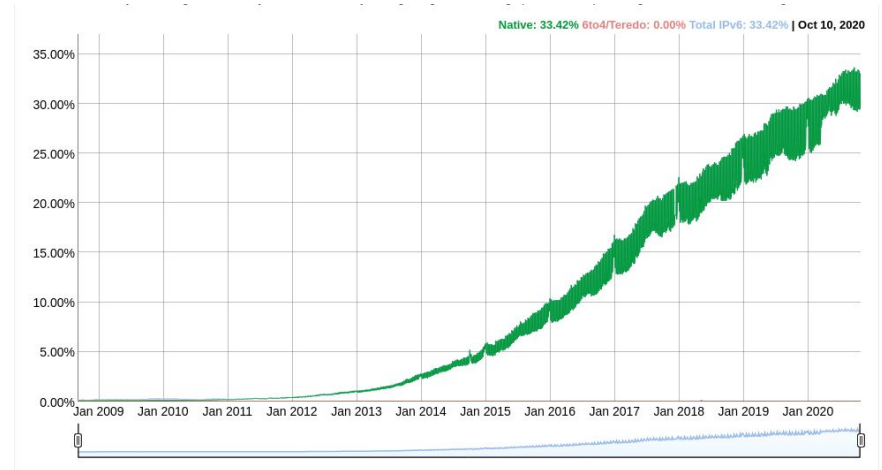
# Motivation

# Motivation: IPv4 depletion & IPv6 rise

- Only 0.39 /8s (or ca. 6.5 million IPv4 addresses) available **world wide**
- More than  $\frac{1}{3}$  IPv6 traffic at Google
- Need to bridge the gap



From <https://ipv4.potaroo.net/>, 2020-10-27



From <https://www.google.com/intl/en/ipv6/statistics.html>, 2020-10-27

# **Key Technologies**

# IPv6 and IPv4

- IPv6 and IPv4 are incompatible
  - Ethernet type: 0x86dd vs. 0x0800
  - Address sizes: 128 Bit vs. 32 Bit
  - Header format
  - Checksum
- Translation methods
  - Higher level, protocol dependent (“proxying”)
  - NAT64

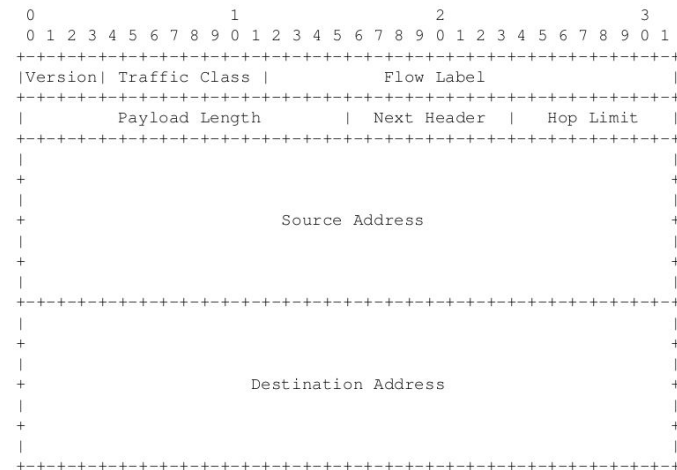


Figure 2.4: IPv6 Header [17]

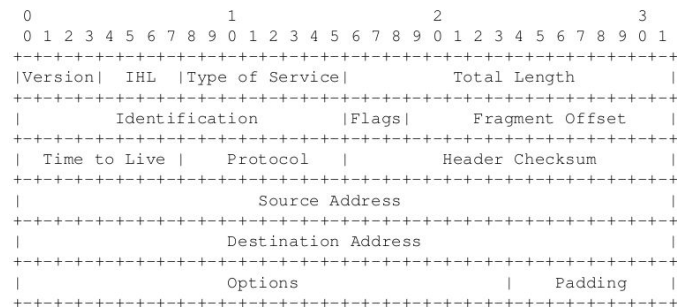


Figure 2.5: IPv4 Header [43]

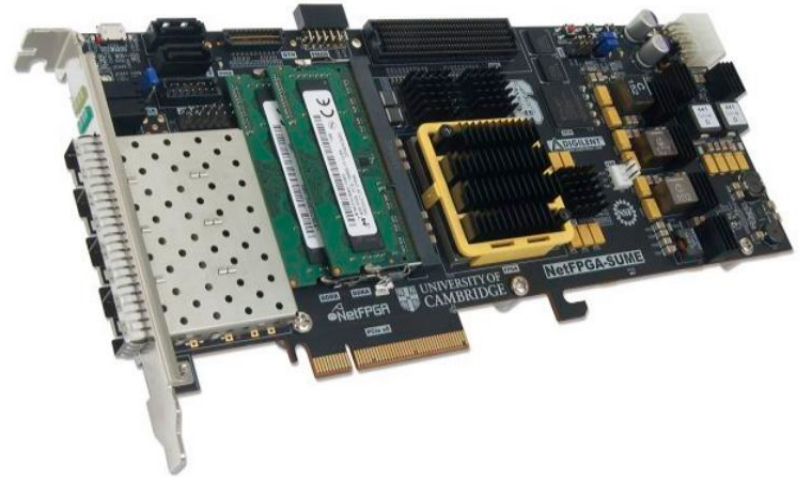
# NAT64: Overview

- Translation on IP level
- Steps
  - Adjust lower level (Ethernet) protocol
  - Change IPv4 <-> IPv6 headers
  - Adjust higher level (TCP/UDP/ICMP/ICMP6) protocol checksum

**P4**

# P4 Targets

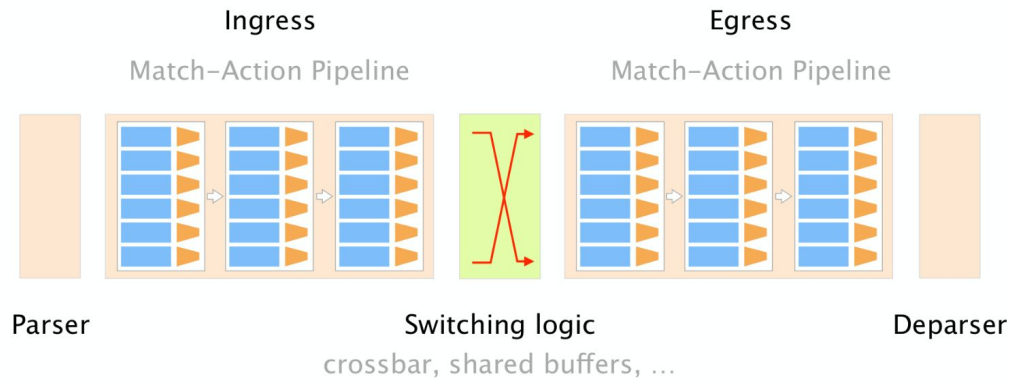
- BMV2
  - Software emulation
  - Fast prototyping
  - Checksum over payload support
- NetFPGA
  - P4->PX->HDL->Bitstream
  - Near line speed processing
  - No payload checksum support





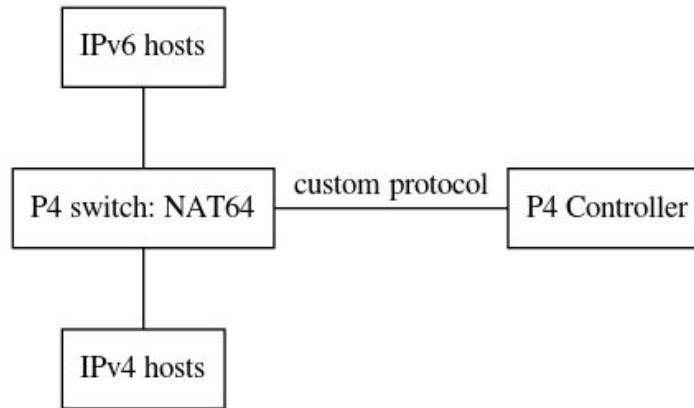
# P4 Language

- Protocol independent
- Target independent: same code, different line speed
  - BMW2 and NetFPGA
- Parsing of well defined fields

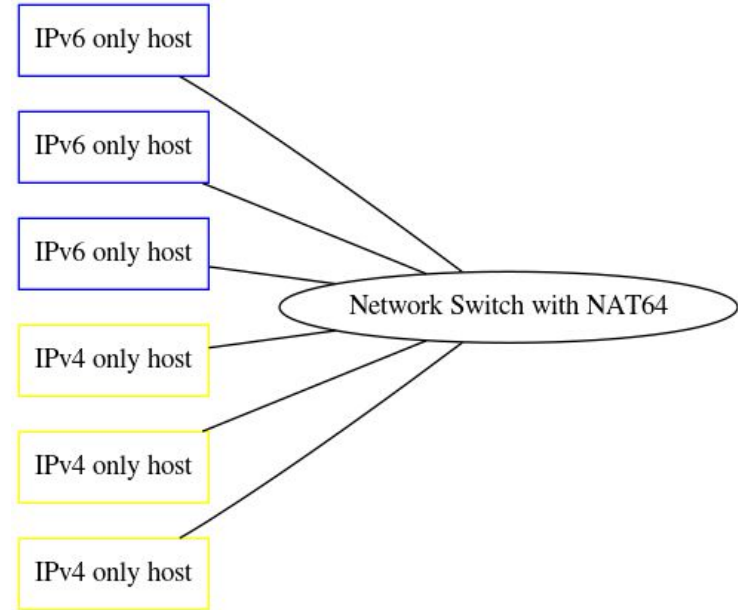
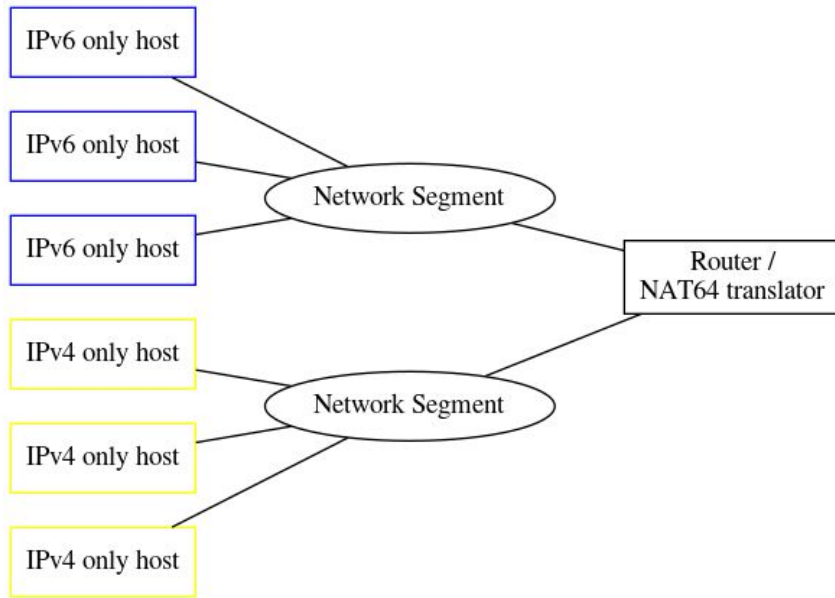


# P4 NAT64 Design

- Same P4 design for both targets
  - Same checksum code
- No functions on NetFPGA
  - Using #defines

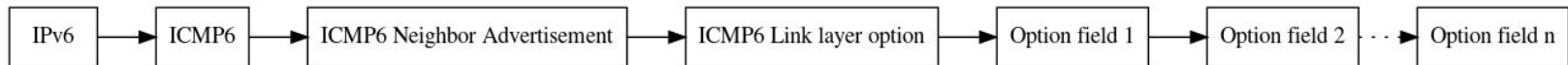
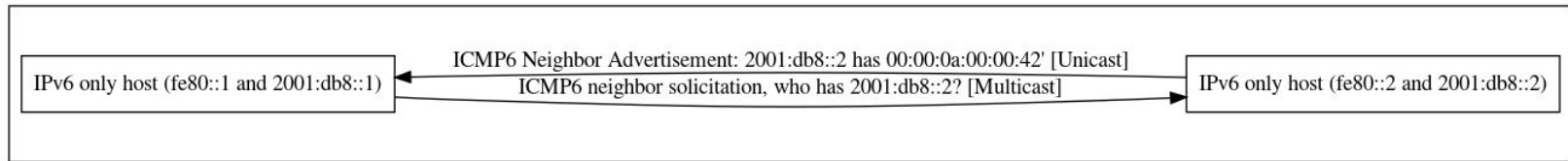
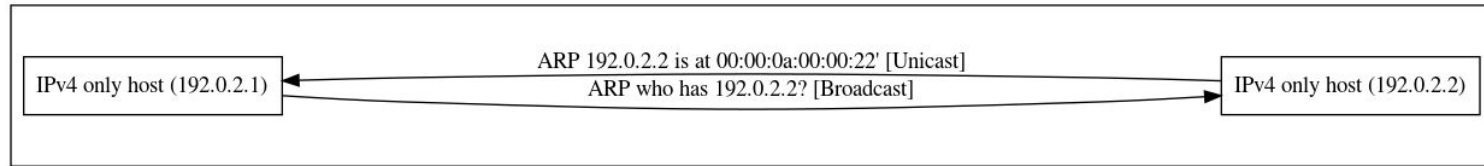


# P4 Network design: In-network translation



# Address resolution: ARP/NDP

- IPv4: ARP: separate protocol; no checksum; Broadcast
- IPv6: NDP: IPv6 only; checksum; Multicast
- ICMP6 option list of **64 bit blocks**



# NAT64 Translation: From IPv6 to IPv4

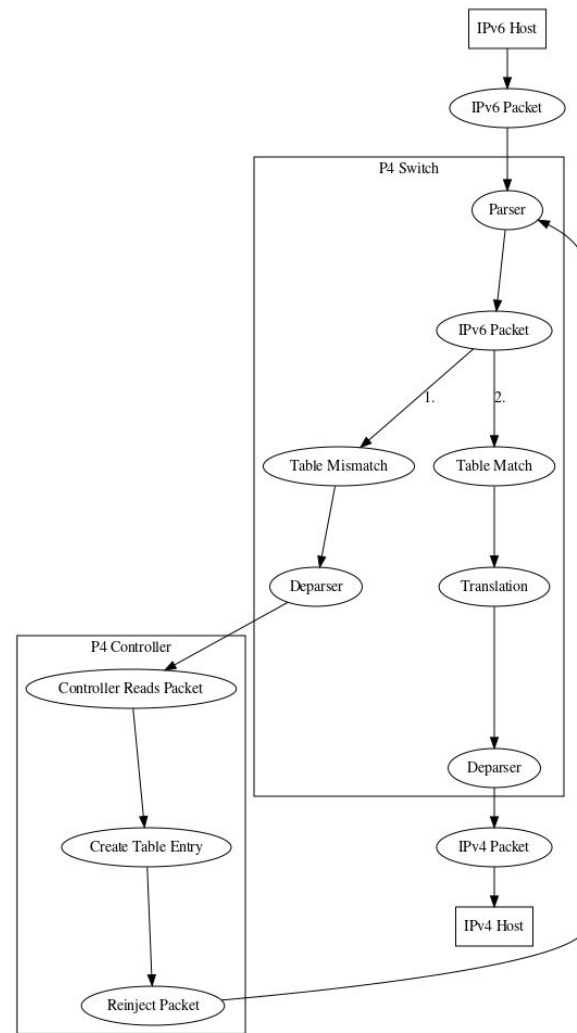
- IPv6 host sends packet to 2001:db8:cafe::192.0.2.2
- P4 switch table matches on 2001:db8:cafe::/96 (nat64 prefix)
- P4 switch calls nat64 action
  - nat64 action adds IPv4 header, maps IPv6 source and destination address
  - nat64 action removes IPv6 header
- NAT64 P4 switch deparsers/sets egress port

# NAT64 Translation: Directions matter



# Stateless vs. Stateful NAT64

- Stateless
  - Usually 1:1 mappings
  - Static mappings
- Stateful
  - Usually 1:n mappings
  - Session table
  - Active controller required



# NAT64: Checksum changes

- Used in TCP, UDP, ICMP, ICMP6
  - Includes payload
- P4/NetFPGA
  - No support for checksum over payload
- Internet checksum: “Sum of 1’s complements”
  - Solution: Calculate differences



# Delta Checksum in P4

- Example: UDP: IPv6 to IPv4
  - $v4sum = v4\_src\_addr + v4\_dst\_addr + (totalen-20) + protocol$
  - $v6sum = v6\_src\_addr + v6\_dst\_addr + payloadlen + next\_header$
  - $udpchecksum = udpchecksum + v4sum - v6sum$

```
action v6sum() {
  bit<16> tmp = 0;

  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[15:0];
  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[31:16];
  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[47:32];
  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[63:48];
  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[79:64];
  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[95:80];
  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[111:96];
  tmp = tmp + (bit<16>) hdr.ipv6.src_addr[127:112];

  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[15:0];
  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[31:16];
  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[47:32];
  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[63:48];
  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[79:64];
  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[95:80];
  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[111:96];
  tmp = tmp + (bit<16>) hdr.ipv6.dst_addr[127:112];

  tmp = tmp + (bit<16>) hdr.ipv6.payload_length;
  tmp = tmp + (bit<16>) hdr.ipv6.next_header;

  meta.v6sum = ~tmp;
}
```

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

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

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

  meta.v4sum = ~tmp;
}
```

```
action delta_udp_from_v6_to_v4()
{
  delta_prepare();

  bit<17> tmp = (bit<17>) hdr.udp.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.udp.checksum = (bit<16>) tmp;
}
```

# Results

# Results: NAT64 TCP Benchmark

- Measured and tested with iperf

Tayga	2.35-3.34 Gbit/s
Jool	7.18-8.25 Gbit/s
P4/NetFPGA	8.51-9.29 Gbit/s

Performance measurements with iperf, 190 seconds, 10 second warmup time, 1-50 parallel sessions, 3 repetitions; min / max values shown

# Conclusion and outlook

- NAT64 successfully implemented on 2 P4 targets
- Jool surprisingly fast
- P4/NetFPGA: research only target
- Many P4 improvements possible - even a P4OS?

Want to follow up? You find me on

- <https://IPv6.chat>
- nico -at- ungleich.ch

THIS PENGUIN  
NEEDS IPv6.

