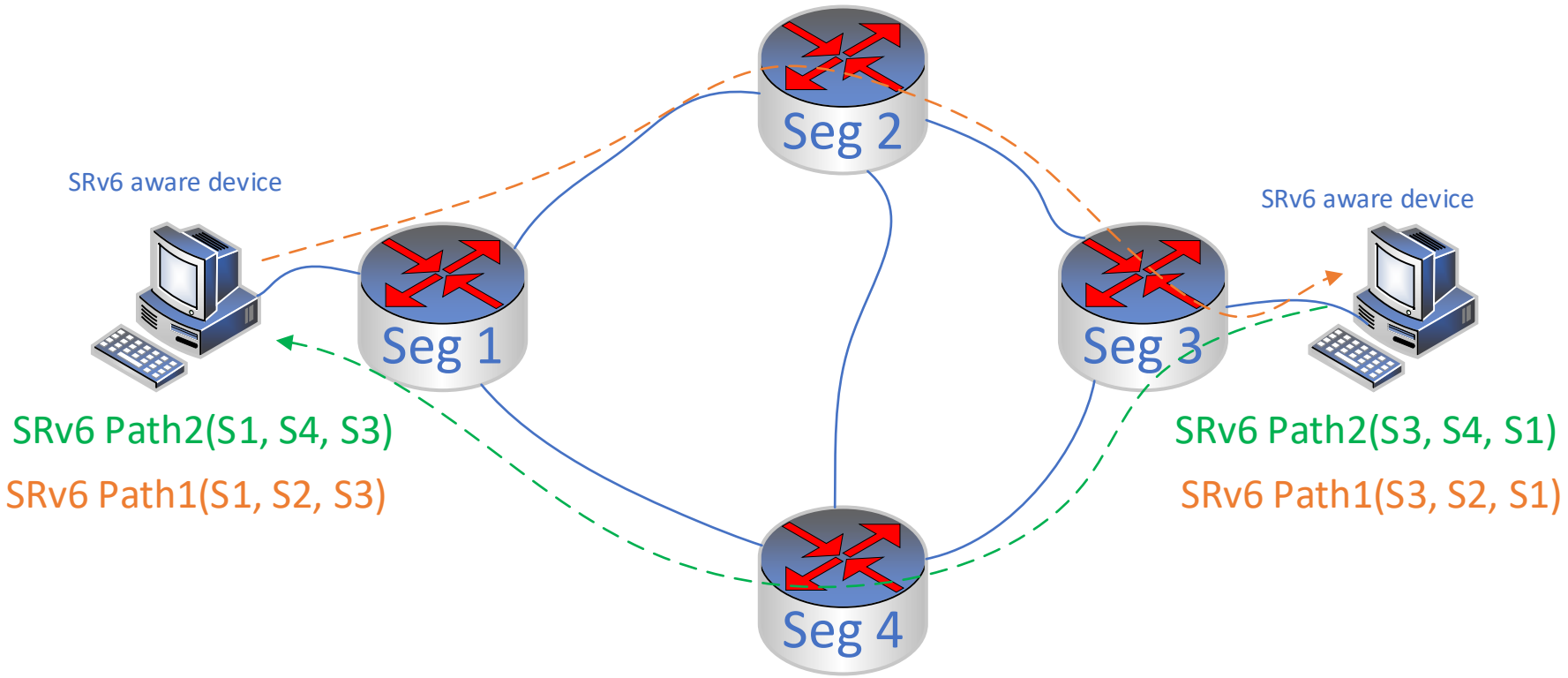
Segment Routing Proxy Device Implemented Using P4 on FPGA with Zero CPU Overhead

Mirek Walukiewicz
miroslaw.Walukiewicz@intel.com

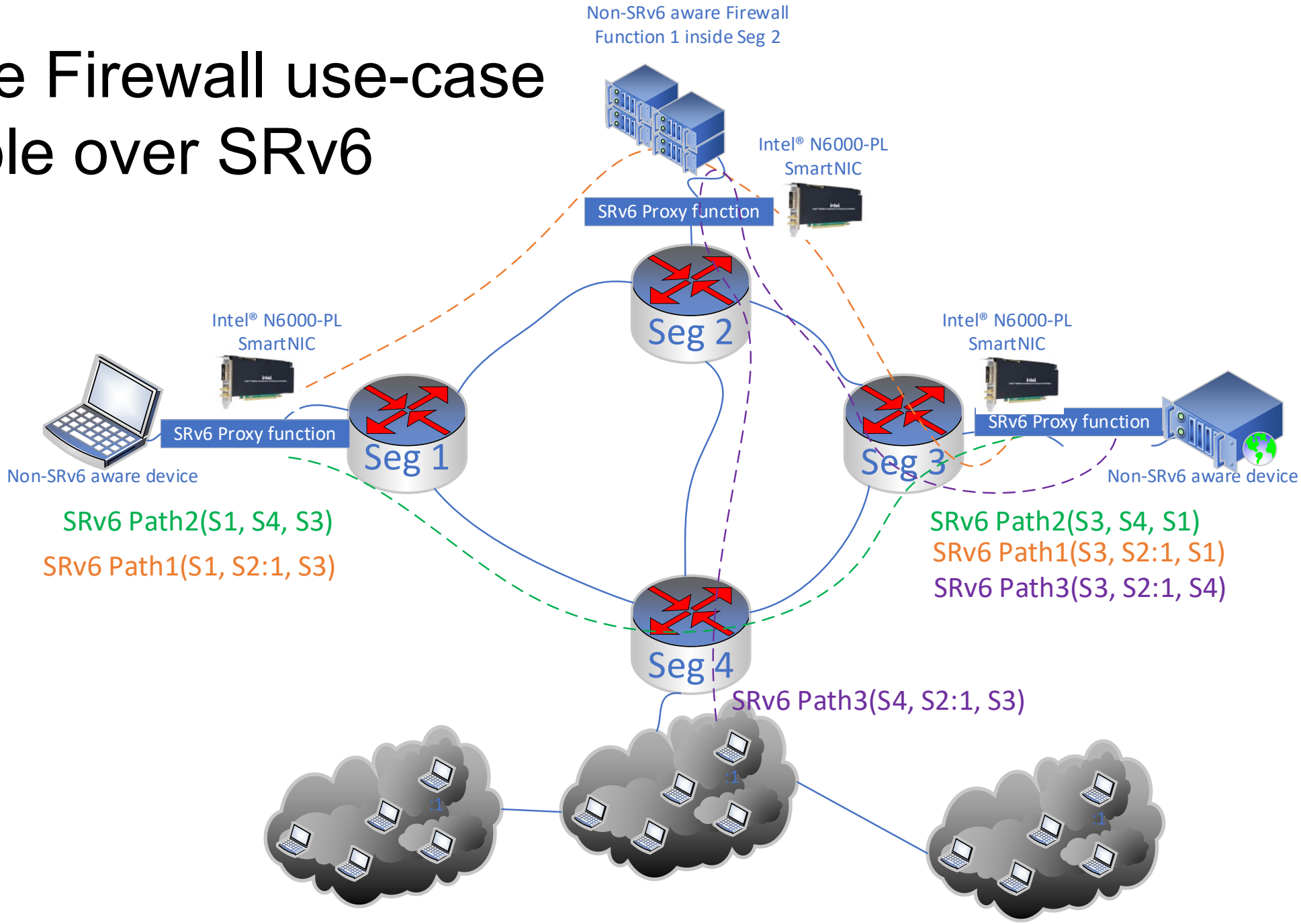
Big Thanks to Pavel Benacek, Jan Kubalek and Bert Klaps from Intel

Segment Routing v6 overview

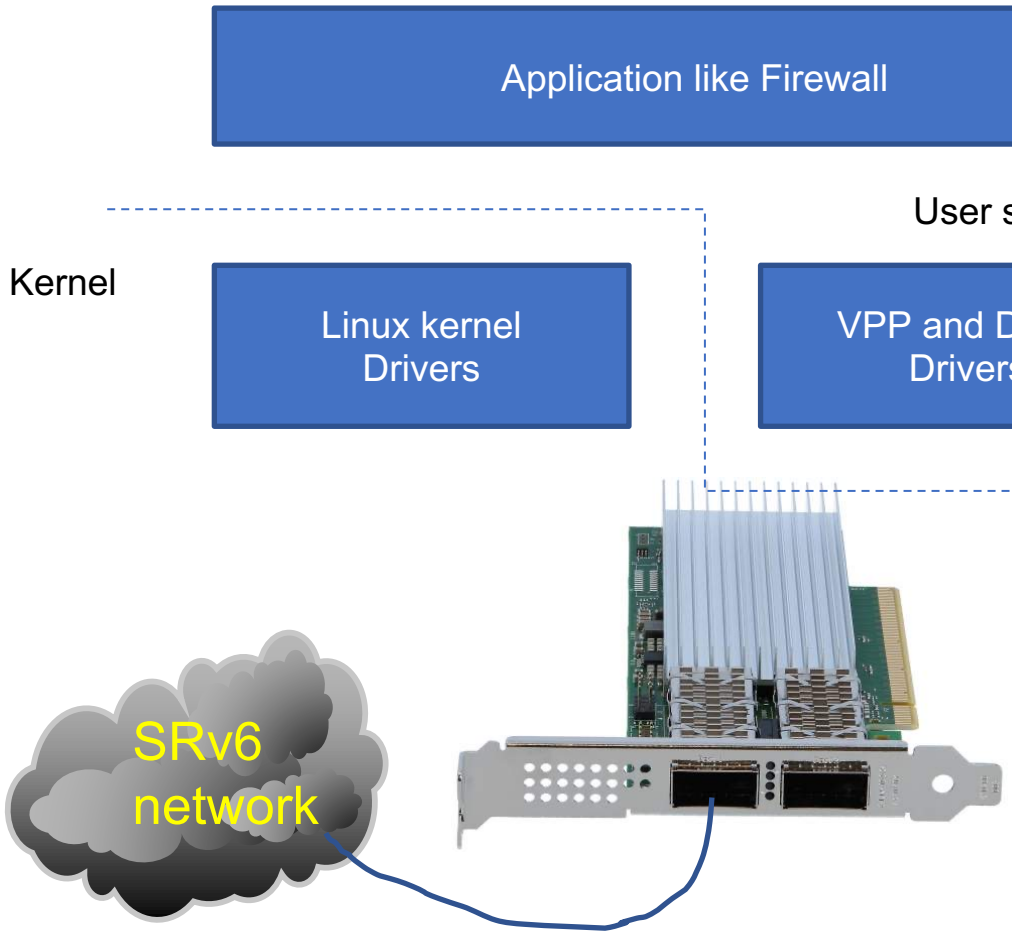


SRv6 is a new powerful protocol with many emerging use-case.
It is considered as successor of well-known GTP-U protocol.
It is very useful for implementing 5G network slicing.

Remote Firewall use-case available over SRv6



Software implementation of the SRv6-proxy



SRv6 in SW consumes CPU cycles
whatever method is used to processing packets

Foundational NIC
Like **Intel(R) - E810CQDA2 –**
Ethernet Network Adapter E810-CQDA2

Using P4 and FPGA based SmartNICs we can make SRv6-proxy system more efficient

Why P4 for SRv6-proxy implementation using FPGA?

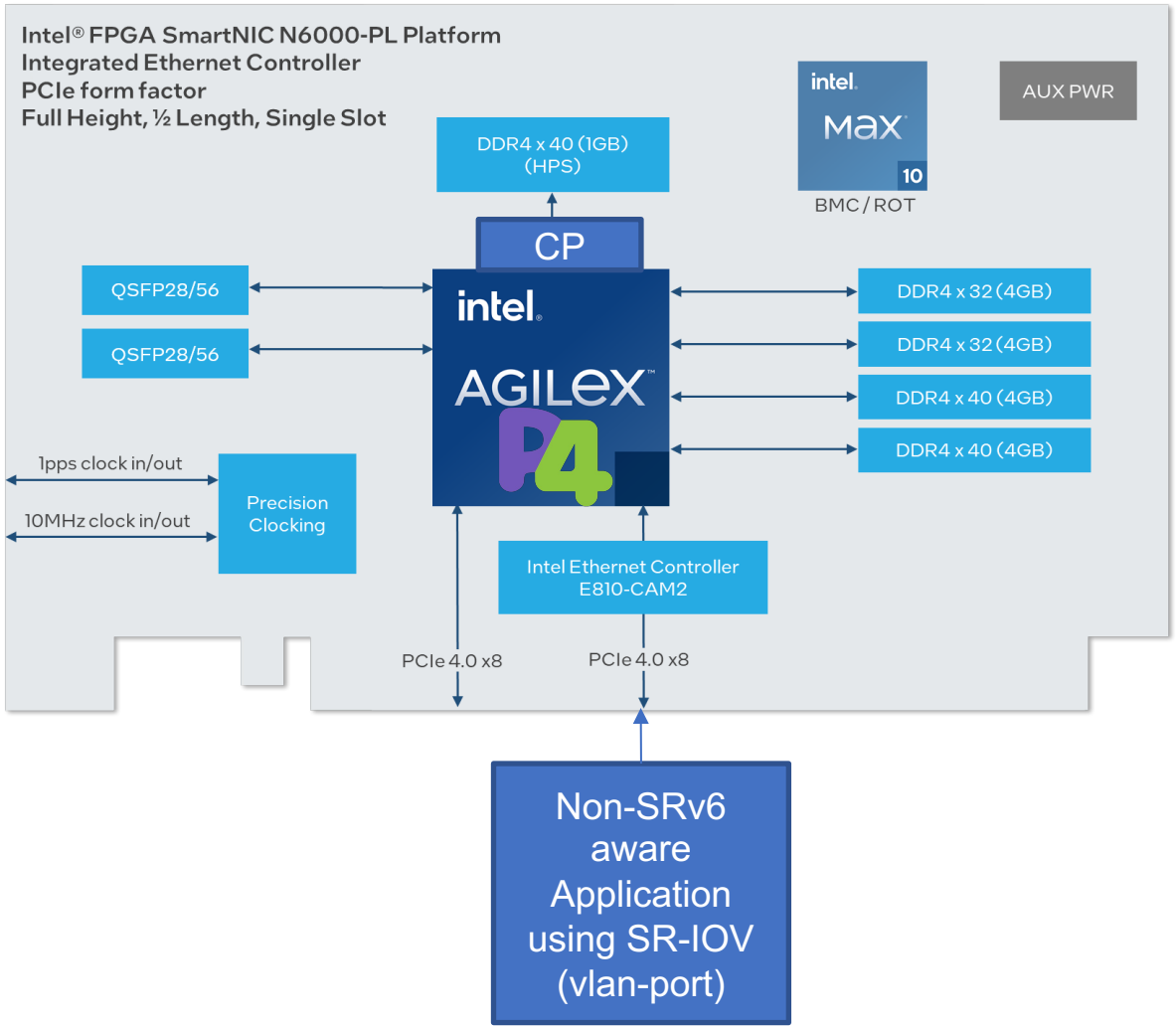
- There are SmartNIC that can implement any reasonable networking pipeline.
- P4 allows to define many classes of networking pipelines.
- Programming FPGAs is complex, in general.
- Very RTL skilled engineers are necessary to make FPGA projects.
- FPGA projects with complexity like SRv6 proxy can take months to complete.
- Using High Level Synthesis approach like P4 can reduce development time to weeks.
- No RTL skilled engineers are required to complete such work on SmartNIC.

So do it with FPGA and P4

HW platform for SRv6-proxy implementation in P4

Ethernet Interfaces 10G/25G/100G

Clock synchronization

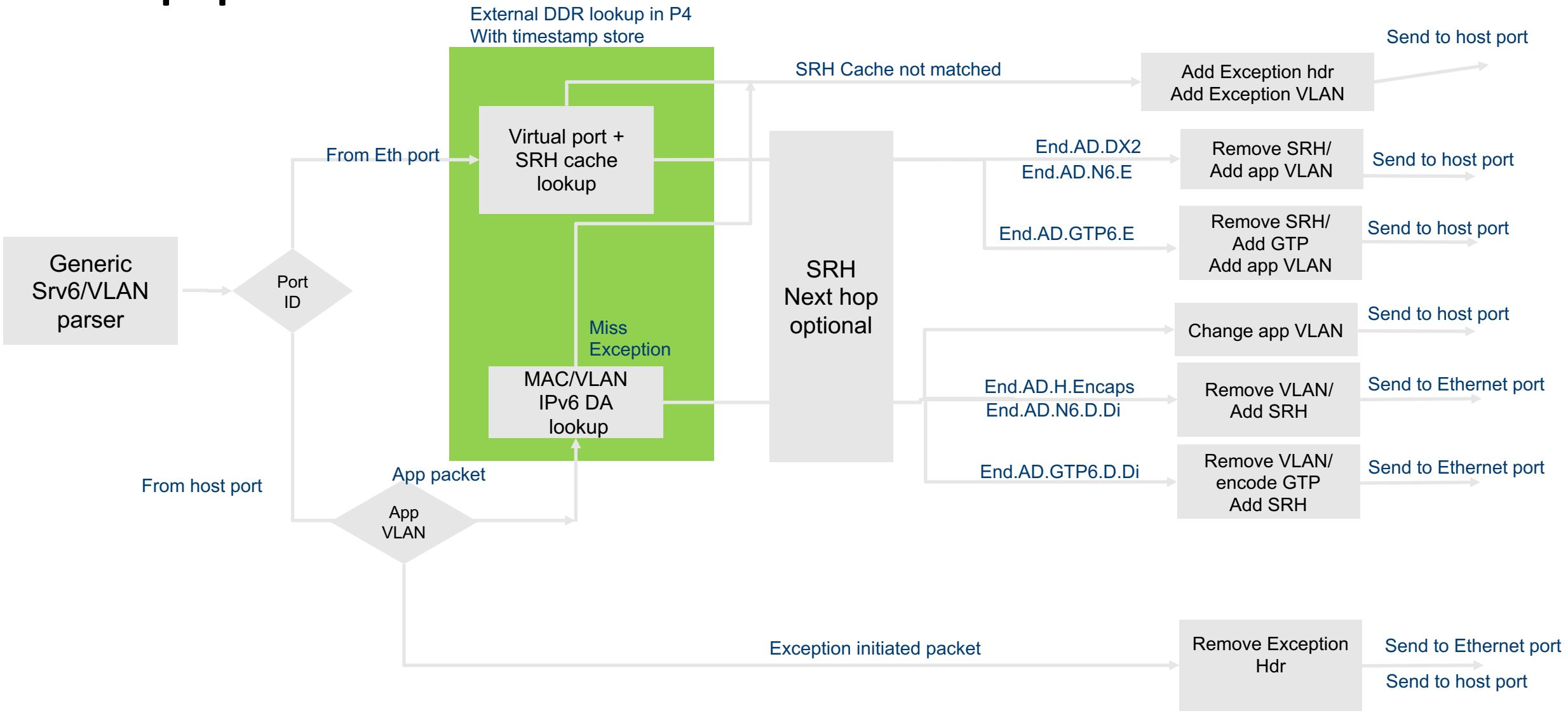


CP – control plane running on HPS A53 ARM cores

DDR4 is a store for big tables

No CPU cores spent on SRv6 proxy processing! Everything done in HW.

P4 pipeline



Exception handler/Control plane is on HPS or host.

Host->Network Transformation Table

```
@Intel_BindTo[arch_owned_iface="mbl_i1"]
@name(".h_ad_lookup")
table h_ad_lookup {
    key = {
        hdr.ethernet.dstAddr  : exact;
        meta.vlan_m.vlan_vid  : exact;
        hdr.vlan.isValid()    : exact;
        meta.ip.dstAddr       : exact;
        hdr.ipv4.isValid()    : exact;
        hdr.ipv6.isValid()    : exact;
        hdr.gtp_u.isValid()   : exact;
        hdr.pdu.isValid()     : exact;
    }
    actions = {
        h_ad_gtp6_d;
        h_ad_gtp4_d;
        h_ad_encaps;
        h_ad_n6_d;
        h_ad_n4_d;
        add_except_hdr;
    }
    size = 2097152;
    const default_action = add_except_hdr(EXP_HOST_LKUP_MISS);
}
```

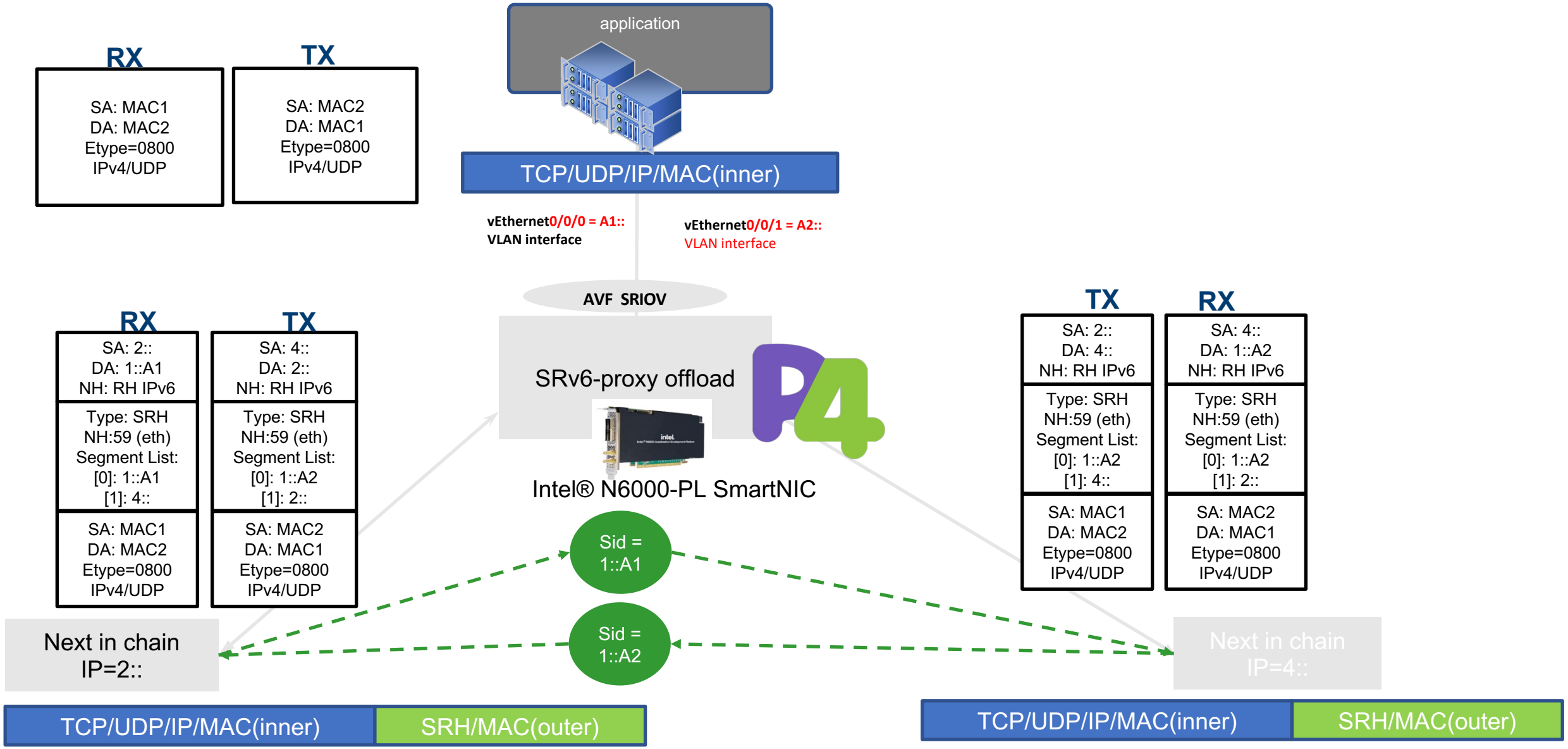
Biggest HW acceleration challenge –
Big exact match tables containing millions of entries
Table Key is simple, but actions parameters are very big due to SRv6 related definitions

Network->Host Transformation Table

```
@Intel_BindTo[arch_owned_iface="mbl_i0"]
@name(".end_ad_lookup")
table end_ad_lookup {
    key = {
        hdr.ipv6.srcAddr       : exact;
        meta.seg_meta.sr_loc   : exact;
        meta.seg_meta.proxy_f  : exact;
    }
    actions = {
        end_ad_gtp6_e;
        end_ad_gtp4_e;
        end_ad_dx2;
        end_ad_n6_e;
        end_ad_n4_e;
        add_except_hdr;
    }
    size = 2097152;
    const default_action = add_except_hdr(EXP_SRV6_MISS);
}
```

Big challenge for HW acceleration: very big exact match tables (2M) and large size keys due to full SRv6 header.

SRv6 test configurations - BIDIRECTIONAL TRAFFIC



Summary

Achievements to share:

- Full SRv6 proxy acceleration presented supporting millions of SRv6 paths using FPGA based SmartNIC
- No SW overhead for SRv6 processing by host application connects the embedded NIC SR-IOV interface.

Learnings from the project:

- FPGA acceleration made by engineers without Verilog/VHDL knowledge thanks to use P4 translation.
- Possibility to demonstrate use of very large exact match tables in DDR4 difficult to achieve by ASICs.
- FPGA-based SmartNIC development is much faster with P4.

Thank You!

Mirek Walukiewicz
miroslaw.Walukiewicz@intel.com