Rapid Prototyping of Avionic Applications Using P4

Dominik Scholz, Fabien Geyer, Sebastian Gallenmüller, Georg Carle
Motivation
What is this talk about?

You will see...

- ...P4 applied to a domain P4 was not specifically designed for ...
- ...what works ...
- ...what performance can be achieved ...
- ...and what is not explicitly supported by the P4 language
Motivation

The Challenge

Aircraft networks of the future

- Cheap (COTS) devices
- Implementing all required functions
- Not implementing anything else (safety!)
- … and short development times!

→ Ethernet-based networks ✓
→ P4?
Background: Reproducible Performance Measurements

TUM Testbed for Automated Network Experiments

Automated network experiments [1]

- Physical hosts
- Different wired network setups, also VM setups

Integrated P4 targets

- P4@ELTE/t4p4s
- PISCES
- p4c-behavioral/bmv2
- NetFPGA SUME
- Netronome Agilio Smart NIC


Outline

Avionics Full-Duplex Switched Ethernet (AFDX)

AFDX Implementation with P4

P4 Enhancements for Avionics Use Cases

Conclusion
Properties of AFDX

- Based on Ethernet
- Guaranteed bandwidth
- Guaranteed end-to-end latency
- Quality of Service guarantees
- Redundancy
Avionics Full-Duplex Switched Ethernet (AFDX)

Terminology

Virtual Link

- Single source, multiple destinations
- Rate-constrained network tunnel
- Dedicated bandwidth allocation
- Static configuration
Basic Components of AFDX Switch

1. Frame format
2. Integrity Checks
3. Static Forwarding per Virtual Link
4. Bandwidth allocation per Virtual Link
1. Frame Format

- Based on Ethernet frame
- Virtual Link encoded in destination MAC address

```c
header_type afdx_t {
  fields {
    dstConst : 32;
    dstVlinkID : 16;
    srcAddr : 48;
    etherType : 16;
  }
}
header afdx_t afdx;
```
2. Integrity Checks

- Header format
- Frame size per Virtual Link

Ingress Function

```c
control ingress {
  integrity_check();
  if (afdx.dstConst == DST_CONST) {
    apply(tbl_forward_virtual_link);
    traffic_policing();
  } else {
    apply(do_drop);
  }
}
```
3. Static Forwarding per Virtual Link

- Vendor-specific multicast

```p4
action forward(egress_ports) {
    modify_field(standard_metadata.egress_spec,
                 EGRESS_SPEC_MULTICAST);
    modify_field(intrinsic_metadata.egress_port_bitmap,
                 egress_ports);
}
table tbl_forward_virtual_link {
    reads {
        standard_metadata.ingress_port : exact;
        afdx.dstVlinkID : exact;
    }
    actions {
        drop;
        forward;
    }
    size : MAX_VIRTUAL_LINKS;
}
```
4. Bandwidth Allocation per Virtual Link

**AFDX terminology:** Filtering based on time between frames
- Minimum time between first bit of consecutive frames
- In AFDX called Bandwidth Allocation Gap

**Implementation 1:** Ingress timestamp & egress scheduler
- Requires egress time for egress scheduler
  → Additional support by P4 switch required

**Implementation 2:** Bandwidth limitation of Virtual Link
- Possible with P4 meter

```cpp
meter vlink_bandwidth { 
  type : bytes; 
  direct : tbl_forward_virtual_link; 
  result : scheduling_metadata.color_bytes; 
}

control traffic_policing { 
  if(scheduling_metadata.color_bytes != COLOR_GREEN) { 
    apply(do_drop); 
  } 
}
```
## AFDX Implementation with P4$_{14}$

**Performance**

### Prototypes
- **Software:** P4@ELTE
- **FPGA:** Xilinx Zynq
- **Network Flow Processor (NPU):** Netronome Agilio

### Switches

<table>
<thead>
<tr>
<th>Switch Estado</th>
<th>Latency (1500 B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rockwell Collins AFDX switch [1]</td>
<td>5 µs</td>
</tr>
<tr>
<td>HP E3800 without OpenFlow [1]</td>
<td>7,2 µs</td>
</tr>
<tr>
<td>HP E3800 with OpenFlow [1]</td>
<td>7,7 µs</td>
</tr>
<tr>
<td>HP E3800 with software switching [1]</td>
<td>613 µs (avg.)</td>
</tr>
<tr>
<td>P4 switch with P4@ELTE [2]</td>
<td>24 µs</td>
</tr>
<tr>
<td>P4 switch with NPU with CPU [2]</td>
<td>24 µs</td>
</tr>
<tr>
<td>P4 switch with NPU without CPU [2]</td>
<td>5,8 µs</td>
</tr>
<tr>
<td>P4 switch with FPGA [2]</td>
<td>1,2 µs</td>
</tr>
</tbody>
</table>

P4 switches competitive with existing hardware for Avionic Networks

---


P4 Enhancements for Avionics Use Cases

Scheduling

- Limited support (Strict Priority Queuing)
- Additional support desirable, for example: Weighted Fair Queuing [1] or Deficit Round Robin [2]
  → Required for QoS architectures
  → Programmable packet scheduling in parallel to P4 [3]

Time-based and Time-triggered Protocols

- No primitives to access clocking information
- Required for packet timestamping, egress scheduling based on timing information
- Cf. IEEE TSN: Time Sensitive Networking
  → Interface for retrieving time information
  → Vendor-specific and vendor-independent metadata

P4 Enhancements for Avionics Use Cases

Device Certification for Safety Requirements

Improvements from P4\textsubscript{14} to P4\textsubscript{16} for Safety of Avionics Hardware

- Defined behavior (casting, exceptions, initial values)
- Portable Switch Architecture
- General: no loops
- Formal analysis and functional validation improved \[1\][2]
  \rightarrow Verification of performance parameters

Goal: Certification of P4 hardware, tools and programs

Conclusion

- P4 suited for prototyping in other domains
- Implementation of (only) required features
- Reduced development time and cost
- Requires expertise in P4 and target switch
- Requires better support for scheduling and timing information
- Goal: Certification of P4 hardware, tools and programs

Relevant full paper:
In: 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), February, 2018
https://hal.archives-ouvertes.fr/hal-01711011/document