Accelerating DPDK via P4-programmable FPGA-based Smart NICs

Petr Kastovsky
kastovsky@netcope.com
Why FPGAs and P4?

- Goal: Accelerate packet processing
  - Reduce latency and jitter
  - Release cores for application processing rather than network processing

- FPGAs are great for data stream processing = Pipeline
  - Workload specific set of operations executed over a data stream

- FPGAs are hard to program
  - Have you ever seen Verilog or VHDL? It’s VERY verbose!

- P4 is target independent and abstract
  - open source, prevents vendor lock-in
Benefits of P4 + FPGAs

- FPGAs are just another software programmable HW
- Deterministic performance, low jitter and low latency
- Dynamic allocation of FPGA resources to P4 pipeline
  - Small number of large tables, large number of small tables
- Parser and deparser of arbitrary complexity
- Scalable FPGA chip offering
  - From 10G, 25G to Nx100G
- Flexible memory configurations
  - On-chip vs off-chip, static vs dynamic
- Extensible through P4 16 externs
  - E.g. payload processing (encryption, pattern matching)
Use case example - segment routing v6

- Segment Routing: Application steers packets through an ordered list of instructions and realizes end-to-end policy.
  - Promoted by Cisco, Microsoft, Bell Canada, Alibaba, SoftBank, ...
- The IPv6 flavor of Segment Routing (SRv6) allows user-defined functions to be associated with segments - network programming
- Needs dataplane support!

![IPv6 Segment Routing Diagram](image-url)
SRv6 - parser definition

```c
header_type ipv6_t {
    fields {
        ver : 4;
        trafClass : 8;
        flowLab : 20;
        payLen : 16;
        nextHead : 8;
        hopLim : 8;
        srcAddr : 128;
        dstAddr : 128;
    }
}

header_type ethernet_t {
    fields {
        dstAddr : 48;
        srcAddr : 48;
        etherType : 16;
    }
}

#define PROTOCOL_IPV6 0x86dd
#define PROTOCOL_V6EXT 0x2B

header ethernet_t ethernet_0;
header ipv6_t ipv6;

// Ethernet parsing
parser parse_ethernet {
    extract(ethernet_0);
    return select(latest.etherType) {
        PROTOCOL_IPV6 : parse_ipv6;
        default : ingress;
    }
}

// IPv6 parsing
parser parse_ipv6 {
    extract(ipv6);
    return select(latest.nextHead) {
        PROTOCOL_V6EXT : parse_ext;
        default : ingress;
    }
}
```
SRv6 - match-action tables

// Rewrites destination IPv6 address
action rewrite() {
    // Rewrite the destination IPv6
    // address with the last IPv6 segment
    modify_field(ipv6.dstAddr,lastSeg.segVal);
    add_to_field(ipv6_ext.next_seg,-1)
}

// Only default rule with action rewrite
table tab_rewrite {
    actions {
        rewrite;
    }
}

// Sets egress port to specific value
action set_egress_port(eport) {
    modify_field(ethernet_0.dstAddr,eport);
}

// Sets egress port to specific value
action forward_based_on_hash() {
    modify_field_with_hash_based_offset(
        ethernet_0.dstAddr, 0,ipv6_hash,65536);
}

// Set dst MAC based on destination IPv6
table table_set_egress_port {
    reads {
        ipv6.dstAddr : ternary;
    }
    actions {
        drop_p;
        permit;
        set_egress_port;
        forward_based_on_hash;
    }
    max_size: 15;
}

control ingress {
    // If any segment was valid
    if(valid(ipv6_seg0)) {
        // Apply the segment routing rewrite
        apply(tab_rewrite);
    }
    // Set destination MAC
    apply(table_set_egress_port);
}
Firmware architecture

Intel Arria 10 FPGA
NFV Wire Model
User Programmable Logic
Netcope P4 Atom
Parser → MAT → Deparser
Netcope P4 Atom
Parser → MAT → Deparser

Intel MAP + Netcope P4
Intel Xeon + Linux OS

QSFP
XL710
enp8s0f0

QSFP
XL710
enp8s0f1
Many more use cases

- Tunnel processing
- Traffic statistics
- Load balancing
- Traffic shaping
- Virtual switching
- Inband network telemetry
- ...

Diagram:

- OpenStack
- Neutron
- Host OS
- vRouter agent
- OvS agent
- VM1 Firewall
- More VMs
- VMn
- FPGA
- Ethernet
- Parse
- Match
- Action
- More M+As
- Ethernet
- L2 IP UDP VxLAN L2 IP Payload

Diagram shows the integration of OpenStack with Neutron, Host OS, and various components like vRouter and OvS agents, leading to VMs and FPGA with Ethernet and matched actions.
DPDK API

- DPDK RTE FLOW - API to expose P4 pipeline to a user

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>2</td>
</tr>
<tr>
<td>Priority</td>
<td>0</td>
</tr>
<tr>
<td>Traffic</td>
<td>Egress</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ethernet</td>
</tr>
<tr>
<td>1</td>
<td>IPv4</td>
</tr>
<tr>
<td>2</td>
<td>TCP</td>
</tr>
<tr>
<td>3</td>
<td>END</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OF_DEC_NW_TTL</td>
</tr>
<tr>
<td>1</td>
<td>OF_PUSH_VLAN</td>
</tr>
<tr>
<td>2</td>
<td>PHY_PORT</td>
</tr>
<tr>
<td>3</td>
<td>END</td>
</tr>
</tbody>
</table>
Attribute Group

Group 0 (origin)

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>JUMP 1</td>
<td></td>
</tr>
<tr>
<td>JUMP 2</td>
<td></td>
</tr>
</tbody>
</table>

Group 1

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>JUMP 3</td>
<td></td>
</tr>
</tbody>
</table>

Group 2

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>JUMP 3</td>
<td></td>
</tr>
</tbody>
</table>

Group 3

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Matching patterns supported protocols

- **Packet headers**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH</td>
<td>UDP</td>
<td>NVGRE</td>
<td>GENEVE</td>
<td>ICMP6_ND_NS</td>
<td></td>
</tr>
<tr>
<td>VLAN</td>
<td>TCP</td>
<td>MPLS</td>
<td>VXLAN-GPE</td>
<td>ICMP6_ND_NA</td>
<td></td>
</tr>
<tr>
<td>IPV4</td>
<td>SCTP</td>
<td>GRE</td>
<td>ARP_ETH_IPV4</td>
<td>ICMP6_ND_OPT</td>
<td></td>
</tr>
<tr>
<td>IPV6</td>
<td>VXLAN</td>
<td>GTP, GTPC, GTPU</td>
<td>IPV6_EXT</td>
<td>ICMP6_ND_SLA_ETH</td>
<td></td>
</tr>
<tr>
<td>ICMP</td>
<td>E_TAG</td>
<td>ESP</td>
<td>ICMP6</td>
<td>ICMP6_ND_TLA_ETH</td>
<td></td>
</tr>
</tbody>
</table>

- **Special**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANY</td>
<td>Matches any protocol in place of the current layer</td>
</tr>
<tr>
<td>RAW</td>
<td>Matches a byte string of a given length at a given offset</td>
</tr>
<tr>
<td>FUZZY</td>
<td>Approximate Matching</td>
</tr>
</tbody>
</table>


**Support of actions depends on the available hardware**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>End marker for action lists</td>
</tr>
<tr>
<td>VOID</td>
<td>Used as a placeholder for convenience</td>
</tr>
<tr>
<td>PASSTHRU</td>
<td>Leaves traffic up for additional processing by subsequent flow rules</td>
</tr>
<tr>
<td>MARK</td>
<td>Attaches an integer value to packet</td>
</tr>
<tr>
<td>FLAG</td>
<td>Flags packets. Similar to Action: MARK without a specific value</td>
</tr>
<tr>
<td>DROP</td>
<td>Drop packet</td>
</tr>
<tr>
<td>SECURITY</td>
<td>Encrypt packet payload based on configuration (IPSec)</td>
</tr>
</tbody>
</table>
Actions II

● Packet redirection

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUMP</td>
<td>Redirects packets to a group</td>
</tr>
<tr>
<td>QUEUE</td>
<td>Assigns packets to a given queue index</td>
</tr>
<tr>
<td>RSS</td>
<td>Spread packets among several queues according to the parameters</td>
</tr>
<tr>
<td>PF</td>
<td>Directs matching traffic to the physical function (PF)</td>
</tr>
<tr>
<td>VF</td>
<td>Directs matching traffic to a given virtual function</td>
</tr>
<tr>
<td>PHY_PORT</td>
<td>Directs matching traffic to a given physical port index</td>
</tr>
<tr>
<td>PORT_ID</td>
<td>Directs matching traffic to a given DPDK port ID</td>
</tr>
</tbody>
</table>

● Packet statistics

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>Adds a counter action to a matched flow</td>
</tr>
<tr>
<td>METER</td>
<td>Applies a stage of metering and policing</td>
</tr>
</tbody>
</table>
Actions III

- “OpenFlow” Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Action</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF_SET_MPLS_TTL</td>
<td>OF_COPY_TTL_OUT</td>
<td>OF_SET_VLAN_VID</td>
</tr>
<tr>
<td>OF_DEC_MPLS_TTL</td>
<td>OF_COPY_TTL_IN</td>
<td>OF_SET_VLAN_PCP</td>
</tr>
<tr>
<td>OF_SET_NW_TTL</td>
<td>OF_POP_VLAN</td>
<td>OF_POP_MPLS</td>
</tr>
<tr>
<td>OF_DEC_NW_TTL</td>
<td>OF_PUSH_VLAN</td>
<td>OF_PUSH_MPLS</td>
</tr>
</tbody>
</table>

- Packet encap/decap

<table>
<thead>
<tr>
<th>Encap/Decap</th>
<th>Encap/Decap</th>
</tr>
</thead>
<tbody>
<tr>
<td>VXLAN_ENCAP</td>
<td>NVGRE_ENCAP</td>
</tr>
<tr>
<td>VXLAN_DECAP</td>
<td>NVGRE_DECAP</td>
</tr>
</tbody>
</table>
RTE_FLOW and P4

- RTE_FLOW
  - enables very dynamic scenarios
  - follows open flow

- P4
  - successor to open flow
  - protocols defined in the code
  - data plane operations defined in the code

- P4 for FPGA is a valid hardware backend for RTE_FLOW
  - Let’s work to propose API to fully exploit flexibility of P4 for FPGAs
Become **flexible** with Netcope P4 - paradigm shift in FPGA programming.

Petr Kastovsky
kastovsky@netcope.com