PBT-on-Demand on Mellanox P4-Capable Hybrid Switch

Itzik Ashkenazi
Networking Lab Chief Engineer
CS Faculty, Technion, Israel
Agenda

• Introduction
• Enhanced Network Monitoring
• PBT-on-Demand
• Mellanox Spectrum-2
  • P4 Architecture
  • Extensibility Mechanisms
  • Telemetry Header
• PoC Implementation
Introduction

• Joint Project

![Mellanox Technologies Logo](image1.png)

• Students: Sami Zreik, Abdallah Yassin

• Instructors: Alan Lo (Mellanox), Matty Kadosh (Mellanox), Itzik Ashkenazi (LCCN)
Enhanced Network Monitoring

• Needed for:
  • Traffic Engineering
  • Security
  • Anomaly Detection
  • Online Troubleshooting

• OAM Protocols: IEEE 802.1ag, ITU-T Y.1731 are useful, but...

• Flow statistics (NetFlow) is not enough...

• Need the real packet level information

• Mirroring all packets is not a good option...

• So.. mirror sampled packets
Sampling-on-Demand

- Mirroring and Sampling is an expensive resource
- sampling-on-demand monitoring framework proposed by [1]
- Sampling Management Module (SMM)
  - An SDN controller application.
  - Determines the sampling rate of each flow at each Switch according to the monitoring goals of the network operator, while taking into account the monitoring capabilities of each switch.
- Sampling Module
  - Added to some or all network switches/routers.
  - Encapsulates each sampled packet and sends it to a collecting server.
- Collecting Server
  - One or more are located in the network in order to collect and process the sampled packets.

PBT-on-Demand

- Postcard-Based Telemetry (PBT) exports the packet with on-line meta-data like:
  - Switch ID
  - Time Stamp
  - Latency
  - Ingress/Egress queues occupancy

- PBT-on-Demand allows the network controller to tell the network switches how to use the expensive PBT resources in most optimal way
Mellanox Ethernet Switch

- Flexible form-factors with 16 to 128 physical ports.
- Supporting 1GbE through 400GbE.
- Based on Mellanox Spectrum-2 silicon.
- Hybrid packet forwarding concurrent capability: Legacy and Programmable.
• **Programmable block 1: parser**
  • Mellanox provides parsing graph base line - User will be able to add new headers to the packet-parsing graph.

• **Programmable block 2: ingress port**
  • Ability to define chain of multiple match action table. Supported actions: drop, forward to port, mirror, packet modification, tunnel encapsulation, tunnel decapsulation, set QoS, counters, meters, go to table.

• **Programmable block 3: ingress router**
  • Ability to define chain of multiple match action tables. Supported actions: drop, mirror, packet modification, routing (including ECMP), tunnel encapsulation, tunnel decapsulation, set QoS, counters, meters, go to table.

• **Programmable block 4: egress router**
  • Ability to define chain of multiple match action tables. Supported actions: drop, mirror, packet, forward to port, packet modification, set QoS, counters, meters, go to table.

• **Programmable block 5: egress port**
  • Ability to define chain of multiple match action tables. Supported actions: drop, egress mirror, packet modification, set QoS, counters, meters, go to table.
Hybrid Programmability

User programs

P4 Compiler

SDK

SONiC

NOS

Application Sandbox

tunnel app
BMToR app
Telemetry app

Auto generated API

Mellanox SDK

Spectrum pipeline

Parser
port
Flex1 Bridge
Flex2 router
flex3 tunnel
deparser

HW RO blocks
flex M&A
flex parser
flex de-parser

User programs

Tunnel.p4
BMToR.p4
telemetry.p4

Telemetry.p4

flex M&A

BMToR

Telemetry

SONiC

SDK

Auto generated API

Mellanox SDK

Spectrum pipeline

Parser
port
Flex1 Bridge
Flex2 router
flex3 tunnel
deparser

HW RO blocks
flex M&A
flex parser
flex de-parser

User programs

Tunnel.p4
BMToR.p4
telemetry.p4

Telemetry.p4

flex M&A

BMToR

Telemetry

SONiC

SDK

Auto generated API

Mellanox SDK

Spectrum pipeline

Parser
port
Flex1 Bridge
Flex2 router
flex3 tunnel
deparser

HW RO blocks
flex M&A
flex parser
flex de-parser
P4 Extensibility Mechanisms

• Spectrum architecture extends on P4 by providing access to hybrid mode actions as externs
  • Policy based switching
    • extern void set_pbs_port(in label_port_t pbs_port);
  • Trapping packet and send to CPU
    • extern void trap(in bit<8> trap_type, in bit<32> trap_id);
  • Mirroring packets to a remote controller using GRE
    • extern void mirror_to_remote_l3(in bit<8> session_id);
  • Setting QOS and shapers
    • extern void set_policer(in bit<64> policer_id);

• User P4 code can combine these primitives using a standard action

• A rich set of Spectrum pipeline metadata (via standard_metadata_t) is provided at various control points in the pipeline
**Spectrum-2 Telemetry Header Implementation**

<table>
<thead>
<tr>
<th>Offset</th>
<th>00h</th>
<th>04h</th>
<th>08h</th>
<th>0Ch</th>
<th>10h</th>
<th>14h</th>
</tr>
</thead>
<tbody>
<tr>
<td>mirror_ether_type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>raw_opcode</td>
<td>pad_count</td>
<td>flags</td>
<td>packet_type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>timestamp[80:48]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>timestamp[47:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ingress_label_port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>original_packet_size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>egress_label_port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>psn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mirror_header_tlv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Internal] For Spectrum-2 the TLVs are hard coded as follows:

- '0': ing_buff_occupancy
- '1': egr_buff_occupancy
- '2': latency
- '3': flags cxt
- '4': mirror_reason
- '5': telass
- '6': mirror_agent
- '135' [internal] 128+7
Our PoC
Postcard Based Telemetry on Mellanox Switch
Real-Time Graphs

- Egress Buffer Occupancy

![Egress Buffer Occupancy Graph]

- Latency

![Latency Graph]
Real-Time Heavy-Heater Distribution
Thank You

iashken@cs.technion.ac.il
Lccn.cs.technion.ac.il