SONiC Deployments
Powered by Programmable Dataplane

Arkadiy Shapiro
Product Line Manager
Barefoot Networks
Dataplane Programmability

“This is precisely how you must process packets”

Consequence:
Vendor-driven replaced by user-driven

```c
table int_table {
    reads {
        ip.protocol;
    }
    actions {
        export_queue_latency;
    }
}
```

```c
actionadd_header(int_header);
modify_field(int_header.kind, TCP_OPTION_INT);
modify_field(int_header.len, TCP_OPTION_INT_LEN);
modify_field(int_header.sw_id, sw_id);
modify_field(int_header.q_latency, intrinsic_metadata.deq_timedelta);
add_to_field(tcp.dataOffset, 2);
add_to_field(ipv4.totalLen, 8);
subtract_from_field(ingress_metadata.tcpLength, 12);
}
export_queue_latency (sw_id) {
```
SONiC Background

- Announced @ March 2016 OCP Summit
- Growing open-source network OS project with multiple contributors (MSDC, ASIC, ODM)
- Originally focused on fixed function switches, so how can we leverage programmable dataplane?
Barefoot Capilano SDE for SONiC

- switch.p4
  - Multiple profiles

- Barefoot Compiler & Dev Tools

SONiC

- SAI

SwitchAPI

Your Auto-generated API

Protocol-independent API

Chip Driver

Tofino: Best-in-class P4 Targets

ASIC Model

Packet Test Framework

Tofino:
Best-in-class P4 Targets
Sample Switch.p4 Features for SONiC

- **Ethernet switching**
  - VLAN Flooding
  - MAC Learning & Aging
  - STP state
  - VLAN Translation

- **IPv4 and IPv6 routing**
  - Unicast Routing
  - Routed Ports & SVI
  - VRF
  - Unicast RPF - Strict and Loose
  - Multicast - PIM-SM/DM & PIM-Bidir

- **QOS**
  - QoS Classification & marking
  - Drop profiles/WRED
  - RoCE v2 / PFC
  - CoPP (Control plane policing)
  - WRED-based ECN marking

- **MPLS**
- **ACL**
  - MAC ACL, IPv4/v6 ACL, RACL
  - QoS ACL, System ACL, PBR
  - Port Range lookups in ACLs

- **Security Features**
  - Storm Control,
  - IP Source Guard

- **sFlow**
- **PTP**
- **Counters**
  - Route Table Entry Counters
  - VLAN/Bridge Domain Counters
  - Port/Interface Counters
  - ACL stats
- **Barefoot Dataplane Telemetry**
SONiC with Programmable Silicon

- Delivery with different switch.p4 profiles
  - Compile different SONiC images
  - Update SONiC image with new compiled SDE

- Profile defines
  - Features enabled / disabled
  - Scale for each table
    - Feature may define several tables in P4
## SONiC Delivery

### How to get started?

<table>
<thead>
<tr>
<th>Option</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary file on SONiC community page</td>
<td>Quick start with SONiC supported features</td>
</tr>
<tr>
<td>Binary files provided by Barefoot support</td>
<td>Quick start with features not upstreamed</td>
</tr>
<tr>
<td>Compile from SDE</td>
<td>SDE modifications (platform support, P4 program / profile change)</td>
</tr>
</tbody>
</table>
Barefoot Announces SONiC Support

- Support for Edgecore and WNC Tofino-based platforms
- Community test validation for available features
- Broad availability to jump start evaluations
- Hardened by months of intense customer qualification testing

Supported Devices and Platforms

Lihua Yuan edited this page 2 days ago · 48 revisions

Following is the list of platforms that support SONiC. Last updated Mar 2018.

<table>
<thead>
<tr>
<th>Switch Vendor</th>
<th>Switch SKU</th>
<th>ASIC Vendor</th>
<th>Switch ASIC</th>
<th>Port Configuration</th>
<th>SONIC Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>WNC</td>
<td>OSW1800</td>
<td>Barefoot</td>
<td>Tofino-T10-018D</td>
<td>48x25G+6x100G</td>
<td>SONIC-ONIE-Barefoot⁶</td>
</tr>
<tr>
<td>Edgecore</td>
<td>Wedge 100BF-32X</td>
<td>Barefoot</td>
<td>Tofino-T10-032D</td>
<td>32x100G</td>
<td>SONIC-ONIE-Barefoot⁶</td>
</tr>
<tr>
<td>Edgecore</td>
<td>Wedge 100BF-65X</td>
<td>Barefoot</td>
<td>Tofino-T10-064Q</td>
<td>65x100G</td>
<td>SONIC-ONIE-Barefoot⁶</td>
</tr>
</tbody>
</table>

Copyright 2018 - Barefoot Networks
Use-case 1: SONiC and Table Scale

- Different table sizes for leaf and spine
- Different table sizes for different deployments
- Example:
  - IPv4 vs IPv6 heavy fabric
  - Local vs remote host route heavy design

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 Host Local</td>
<td>8K</td>
<td>16K</td>
</tr>
<tr>
<td>IPv4 Host Remote</td>
<td>16K</td>
<td>8K</td>
</tr>
<tr>
<td>IPv4 LPM</td>
<td>32K</td>
<td>40K</td>
</tr>
<tr>
<td>IPv6 Host</td>
<td>8K</td>
<td>4K</td>
</tr>
<tr>
<td>IPv6 LPM</td>
<td>8K</td>
<td>4K</td>
</tr>
</tbody>
</table>

Note: This is a sample multi-dimensional scale scenario, not reflective of maximum ASIC capabilities.
Use-case 2: SONiC & Data-Plane Telemetry

Barefoot Data-Plane Telemetry

- In-Band Network Telemetry (INT)
- Intelligent Deduplication and Triggers
- Path & Latency Change Detection
- Microburst detection
- Report on Drop

3rd Party Network Management Solutions

Deep Insight
Open Northbound APIs

Open Telemetry Report Format defined by the P4.org Applications Working Group

Copyright 2018 - Barefoot Networks
SAI Dataplane Telemetry APIs

- Upstreamed to SAI master as experimental
- Will be part of SAI 1.3
- Defines entire spectrum of dataplane telemetry configuration
  - Flow watchlists
  - Switch ID
  - Report destination
SONiC and Telemetry

- SONiC Telemetry Feature
  - New tables in several SONiC databases
  - Configuration script
- Feature community review pending

```python
# Instantiate a switch:
my_switch = sonic_switch.SONiCSwitch(dtel_switch_id='123',
  management_ip='10.10.10.10',
  dtel_monitoring_type='int_endpoint')

# Create a report session:
rs = my_switch.create_dtel_report_session('192.168.0.1')

# Create a watchlist:
wl = my_switch.create_dtel_watchlist('flow')

# Add entries to the watchlist:
wl.create_entry(priority=10,
  src_ip='10.131.0.0',
  src_ip_mask=11,
  dst_ip='10.131.0.0',
  dst_ip_mask=11,
  dtel_sample_percent=100,
  dtel_report_all=True)
```