Network Telemetry in SONiC
(2018 August OCP workshop)

Jipan Yang
(Alibaba)

Zhenggen Xu
zxu@linkedin.com
What is network telemetry

• Full visibility of network status down to individual node is needed.
  • To fix or isolate network problem in a timely manner.
  • To take necessary preventive measures for network segment under pressure.

• Network Telemetry provides tools and interfaces to collect the network status data.
Common network telemetry methods

• SNMP
• Syslog
• CLI
• Packet mirroring
• Data plane telemetry (INT)
• ...

...
Requirements for hype scale data center network telemetry

• Extensible and flexible data collection
  • Whatever, whenever and each node, to pinpoint any problem.

• Structured data
  • To easily consume and correlate.
  • Unstructured data like syslog is hard to analyze.

• Efficient data transportation and encoding
  • Well understood
  • Little overhead
Network telemetry in SONiC

- SONiC is a database centric network OS.
  - Most of the critical control plane data could be found in redis DB.
  - The data in redis DB may be subscribed or polled as needed.
- SONiC telemetry service provides interfaces to collect control plane states.
Network telemetry in SONiC

- gRPC, the right framework for network telemetry
  - High performance.
  - Data streaming.
  - Tooling and libraries.

- gNMI pushed it further in the network domain for config and telemetry.
  - SONiC utilized gRPC with reference to gNMI for telemetry implementation

- Support both DB data and nonDB data
gRPC vs others

• gRPC vs snmp
  • Faster (1% cpu of white-box to poll ~6K counter per second)
  • Subscribe and streaming/event mode
  • Easy to expand
  • Security

• gRPC vs syslog/CLI
  • More data
  • Formatted data
Database and Virtual Pathes

<table>
<thead>
<tr>
<th>DB name</th>
<th>DB No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPL_DB</td>
<td>0</td>
<td>Application running data</td>
</tr>
<tr>
<td>ASIC_DB</td>
<td>1</td>
<td>ASIC configuration and state data</td>
</tr>
<tr>
<td>COUNTERS_DB</td>
<td>2</td>
<td>Counter data for port, lag, queue</td>
</tr>
<tr>
<td>LOGLEVEL_DB</td>
<td>3</td>
<td>Log level control for SONIC modules</td>
</tr>
<tr>
<td>CONFIG_DB</td>
<td>4</td>
<td>Source of truth for SONIC configuration</td>
</tr>
<tr>
<td>FLEX_COUNTERS_DB</td>
<td>5</td>
<td>For PFC watch dog counters control and other plugin extensions</td>
</tr>
<tr>
<td>STATE_DB</td>
<td>6</td>
<td>Configuration state for object in CONFIG_DB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DB target</th>
<th>Virtual Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTERS_DB</td>
<td>&quot;COUNTERS/Ethernet***&quot;</td>
<td>All counters on all Ethernet ports</td>
</tr>
<tr>
<td>COUNTERS_DB</td>
<td>&quot;COUNTERS/Ethernet*/&lt;counter name &gt;&quot;</td>
<td>One counter on all Ethernet ports</td>
</tr>
<tr>
<td>COUNTERS_DB</td>
<td>&quot;COUNTERS/Ethernet &lt;port number &gt;/&lt;counter name &gt;&quot;</td>
<td>One counter on one Ethernet port</td>
</tr>
<tr>
<td>COUNTERS_DB</td>
<td>&quot;COUNTERS/Ethernet*/Queues&quot;</td>
<td>Queues stats on all Ethernet ports</td>
</tr>
<tr>
<td>COUNTERS_DB</td>
<td>&quot;COUNTERS/Ethernet &lt;port number &gt;/Queues&quot;</td>
<td>Queue stats on one Ethernet ports</td>
</tr>
</tbody>
</table>
Collector, Data-Store and More

• Collector cluster collects all telemetry data through gRPC
  • Polling
  • Streaming
  • Dial in
  • Dial out

• Data is saved to time-series DB.
  • Schema-less
  • Preserve history

• Data displayed on GUI

• Rules defined to send events to alert system.

• Data could be used for data-processing (ML etc) and apply actions back to the devices.
Demo setup and Demo cases

- gRPC streaming mode is used.
- Components are built on top of open-source
- Built it fast!

Demo scenarios:
• Traffic flowing from S1->S3, S2->S3

• Displaying Interface speed
  • Speed changes when changing the app data
  • Historic data

• Monitoring Link state changes
  • Flap the link once or multiple times
  • Interface speed changes
  • Link flap counters in 1m
  • Alert sent to email

• Monitoring route nexthop member counters
  • Add/delete nexthops in vtysh
  • Flap links so nexthops got added/deleted automatically
Demo snapshot

Rules:

- **InterfaceFlaps**: 1 active

  ```
  alert: InterfaceFlaps
  expr: changes(oper_status[1m]) / 2 >= 5
  for: 20s
  labels:
  severity: ticket
  annotations:
  summary: interface flaps.
  ```

Email alerts received: