SONiC Platform Management Services
Joe LeVeque (Microsoft)
8/30/2018
Terminology: “Platform”

Platform
- SONiC shares definition with ONIE
- Entire collection of hardware comprising a physical device

Platform-specific peripheral hardware
- SFP transceivers
- Front-panel LEDs
- Fans
- etc.

ASIC model (sometimes referred to as ASIC platform)
- Chipset which implements dataplane
- Managed by SAI
- Not a part of this presentation
Platform Peripheral Devices

- Power supply units (PSUs)
- Fan modules
- SFP transceivers
- Front-panel LEDs
- Environment sensors
- System EEPROM
- System status registers
Platform Management Requirements

Read data to obtain device status

Write data to modify device state
Power Supply Units (PSUs)

• Read:
  • Number of PSU slots
  • PSU presence
  • PSU operational status
  • PSU fan direction
  • PSU fan speed
  • Temperature sensors

• Write:
  • PSU status LED
  • PSU fan speed
Fan Modules

• Read:
  • Number of fan module slots
  • Fan module presence
  • Fan direction
  • Fan speed
  • Expected fan speed and tolerance
  • Fan module EEPROM data (model #, serial #)
  • Fan interrupt events (remove/add fan module)

• Write:
  • Fan speed
  • Fan module status LED
SFP Transceivers

- Read:
  - Transceiver presence
  - Transceiver EEPROM data
    - Cable type, vendor, part #
    - Optical monitoring data
    - Temperature / voltage
  - Transceiver low-power mode status
  - Transceiver interrupt events (plug/unplug)

- Write:
  - Reset transceiver
  - Enable/disable low-power mode
Front-panel LEDs

• Write:
  • Set port LED states
    • SONiC-defined patterns/colors for link speed/breakout modes
  • Set available front-panel status LED states
    • Overall status
    • Fan status
    • PSU status
    • etc.
Environment Sensors

• Read:
  • All available temperature, voltage and fan speed sensors
  • Monitored by lm-sensors; alarms written to syslog
System EEPROM

• Read:
  • Model number
  • Serial number
  • Base MAC address
System Status Registers

- Read:
  - Determine if previous reboot was caused by hardware
    - Power loss
    - Thermal overload
    - etc.
SONiC Design Principles

1. Unified, standardized behavior
   • Consistent experience among all SONiC devices, regardless of underlying platform
   • Easy to understand, implement, test and debug

2. Kernel modules kept as simple as possible
   • Simply expose hardware registers
   • No control logic

3. Peripheral control logic implemented in user-space
   • Applications shared among all platforms
   • Standardized API defined by SONiC, implemented by vendors
**Implementation**

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel modules</strong></td>
<td>• Expose peripheral hardware registers&lt;br&gt;• Provided by platform vendor</td>
</tr>
<tr>
<td><strong>Platform-specific plugins</strong></td>
<td>• Standardized Python API&lt;br&gt;• Vendor implements functions to communicate with exposed hardware</td>
</tr>
<tr>
<td><strong>Client applications</strong></td>
<td>• Command line utilities, daemons&lt;br&gt;• Load platform-specific plugins</td>
</tr>
</tbody>
</table>
## Client Applications

<table>
<thead>
<tr>
<th>Command line utilities</th>
<th>Daemons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• sfputil, psuutil, decode-syseeprom, etc.</td>
<td>• ledd, xcvrd, psud, fand, etc.</td>
</tr>
<tr>
<td>• Query/modify platform peripheral state at user request</td>
<td>• Persistent applications which synchronize peripherals with SONiC state</td>
</tr>
<tr>
<td></td>
<td>• Modify Redis DB upon changes in peripheral state (e.g., SFP transceiver presence)</td>
</tr>
<tr>
<td></td>
<td>• Modify peripheral state upon changes to Redis DB (e.g., Front panel LED state)</td>
</tr>
</tbody>
</table>
Client Example
xcvrd
Testing

SONiC community

- Client applications
- Unit tests
- Automated tests
- lm-sensors

Platform vendor

- Kernel modules
- Implemented plugins
Porting SONiC to a New Platform

• Platform vendor responsibilities:
  1. Design kernel modules which simply expose necessary registers to meet SONiC requirements for peripheral interaction
  2. Implement platform-specific plugins which interact with exposed hardware
  3. Test modules and plugins to ensure proper functionality
Roadmap / Call for Contribution

• Develop remaining plugins
  • Fans
  • System status registers
• Develop command line utilities to interact with the above
• Develop daemons
  • xcvrd
  • psud
  • fand
• Develop tests
Further Reading

- SONiC Platform Porting Guide
  - Contains current implementation details
Q & A