Open. Together.
LinkedIn Adoption of OCP
SONiC

Zhenggen Xu, Software Engineer, LinkedIn
LinkedIn Infrastructure

- **5 DCs**: Global Datacenter footprint
- **20 PoPs**: Global Edge PoP presence
- **>200K servers**
- **>1.5TB**: Inter-DC Next Generation BB
Infrastructure Growth

34% infrastructure growth every year…
High bandwidth demand due to the organic growth.

For every single byte, thousands bytes of east-west traffic:

- Application Call Graph
- Kafka Tracking
- Hadoop / Offline Processing
- Machine Learning
Why Build Own NOS?

- **Freedom and Choice**
  - Flexibility
  - Customization
  - Modularity

- **Move Fast**
  - Growth!
  - Scale
  - Evolve
  - Code & Innovate

- **Independence**
  - Channel Procurement
  - Build Strategy
  - Ownership

- **Control**
  - Quality
  - Maintenance
  - Risks
  - Security
Why SONiC?

-- Linux based
Reuse networking stack, leverage same tools, experience.

-- Containerized
Pick the best components for the jobs at each layer.
Incremental upgrade

-- Platform agnostic
Switch Abstraction Interface (SAI)

-- Large community
LinkedIn SONiC development

• FRR protocol stack integration
• Fully IPv6 support – IPv6 ACL, IPv6 link local etc
• BGP convergence – FIB acceleration, SAI improvements.
• BGP docker warm restart
• SONiC system warm reboot and SWSS warm restart (collaboration effort)
• White-box onboarding
• AAA --- manage switches like servers
• ZTP integration
• Open19 onboarding
• Incremental upgrade of dockers and packages
• LinkedIn tools integration (telemetry and more)

* Green ones are contributed back to the community
SONiC lessons: Routes installation convergence
SONiC testing

Test Statistics

<table>
<thead>
<tr>
<th>Critical Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Tests</td>
<td>82</td>
<td>77</td>
<td>5</td>
<td>00:55:43</td>
<td>100%</td>
</tr>
</tbody>
</table>

Statistics by Tag

<table>
<thead>
<tr>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>

No Tags

<table>
<thead>
<tr>
<th>Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>

SONiC

<table>
<thead>
<tr>
<th>Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>

Network

<table>
<thead>
<tr>
<th>Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>

SONiC

<table>
<thead>
<tr>
<th>Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>

Network

<table>
<thead>
<tr>
<th>Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>

SONiC

<table>
<thead>
<tr>
<th>Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>

Network

<table>
<thead>
<tr>
<th>Tests</th>
<th>Total</th>
<th>Pass</th>
<th>Fail</th>
<th>Expanded</th>
<th>Pass / Fail</th>
</tr>
</thead>
</table>
SONiC upgrade at scale

- version checks
- traffic drain
- upgrade commands
- upgrade status check
- error handling

Network diagram:
- File server
  - Dev test framework
  - System test framework
  - sync
- Build server
  - github
  - Linkedin
- Anycast http servers
  - ONIE images
  - Docker images
  - Deb packages
- Get packages with dependencies
- Switch
- Upgrade framework
- Switch
- Switch
SONiC monitoring: 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 applying actions back to the devices.
SONiC telemetry demo snapshot

Rules:

InterfaceFlaps (1 active)

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

Email alerts received:
SONiC feature WIP: configuration management

- SONiC customized Yang based schema – no backend translation
- Syntax and dependencies validation
- Data store and rollback
- Plugin and play
- Programmable northbound interface, preferably gNMI
SONiC feature WIP: dynamic port breakout

• Fully flexible to delete and add ports at run time
• Port breakout domain and naming can be defined and validated per platform
• Utilize the configuration management system for configuration integrity validation
• Remove and add port related dependencies automatically.
• Empower the flexibility for platforms like Open19 platform to the full extent
Call to Action

SONiC links:

Website: https://azure.github.io/SONiC/
Mailing list: sonicproject@googlegroups.com
Source code: https://github.com/Azure/SONiC/blob/gh-pages/sourcecode.md
Wiki: https://github.com/Azure/SONiC/wiki