Rise of the Open NOS

Open networking Innovations are largely driven by an industry need to protect network platform investments, maximize supply chain diversification, reduce operating costs, and build a homogenous operational and management framework that can be consistently applied across platforms running standardized software. By virtue of its adoption by cloud scale operators and its most recent inclusion in the Linux Foundation, SONiC has gained tremendous momentum across different market segments. This blog outlines key factors relevant to SONiC adoption, its evolution in the open network operating system (NOS) ecosystem, and Cisco’s value proposition with the SONiC platform validation and support.

Why use an open NOS?

Disaggregation enables decoupling hardware and software, giving customers the ability to fully exercise plug-and-play. An open-source NOS like SONiC can provide a consistent software interface across different hardware platforms, allowing for supply chain diversity and avoiding vendor lock in, further leveraged by in-house custom automation frameworks that don’t have to be modified on a per-vendor basis. A DevOps-centric model can accelerate feature development and critical bug fixes, which in turn reduces dependency on vendor software release cycles. The open-source ecosystem can provide the necessary support and thought leadership to enable snowflake use cases prevalent in many network deployments. The freedom to choose can protect investment across both hardware and software, thus leading to significant cost savings that further reduce total cost of ownership (TCO), operating expenditures (OpEx), and capital expenditures (CapEx).

What is SONiC?

SONiC (Software for Open Networking in the Cloud) was created by Microsoft in 2016 to power their Azure cloud infrastructure connectivity. SONiC is Debian based and has a microservice based containerized architecture where all major applications are hosted within independent Docker containers. In order to abstract the underlying hardware and ASIC, SONiC is built on SAI (Switch Abstraction Interface)which is a standardized vendor neutral hardware abstraction API. The NOS provides north bound interfaces (NBIs) to manage the device and these NBIs are based on gNMI, ReST, SNMP, CLI, and Openconfig Yang models so it’s easily integrated with automation frameworks.

Figure 1. A conceptual overview of SONiC

Why SONiC?

With so many open-source options out there, why consider SONiC? This NOS is gaining strong community leverage with growing industry traction through its adoption by prominent players spanning different market segments such as enterprise, hyperscale data center, and service providers. Open-source contributions have honed SONiC for focused use cases, enriching feature delivery while holistically enabling different architectures. Below are a few factors that emphasize open NOS benefits as applicable to SONiC:

Open Source:

- Vendor independence – SONiC can run on any compatible vendor hardware
- Feature velocity – Custom feature additions/modifications and self-driven bug fixes
- Community support – Upstream code contributions benefit all SONiC consumers
- Cost savings – Reduced TCO, OpEx, and CapEx

Disaggregation:

- Modular components – Multiple independent containerized components for increased resiliency and easier plug-and-play
- Decoupling software functions – Individual components can be customized based on use case

Uniformity:
- Abstraction – SAI abstraction layer to normalize underlying hardware intricacies
- Portability – Feature portability as the SAI normalizes hardware complexity

DevOps:
- Automation – Unified orchestration/monitoring for compute and common NOS across platforms
- Programmability – SONiC provides options that can leverage ASIC capabilities to the fullest

Figure 2. The value proposition of SONiC

Where does SONiC fit in various use cases?

At a high level, the existence of a software feature on a SONiC-enabled system depends on the following three components:
1. SONiC operating system support – Community driven
2. SAI API support – Community driven
3. SDK support – Vendor driven

For a software feature to be built into SONiC, it needs to be facilitated at all the above layers to be fully productized. The current SONiC ecosystem is comprehensively built for IP/VxLAN and BGP based architectures. These technology components can be cross-leveraged to create any architecture of choice – whether it is a data center fabric or a CDN ToR. SONiC deployments today are predominantly observed in data centers and enterprises but can be easily extended to other networks that leverage similar technology components. Commonly deployed network roles and use cases with SONiC are outlined below:

Data center fabric and DCI – IP/VxLAN and BGP based:

1. Leaf (single and dual homed)
2. Spine
3. Super spine

These data center deployments are spread across different customer segments ranging from Tier1/Tier 2 hyperscalers, service providers, and larger enterprises.

Due to its strong community support, many working groups are collaborating on how to further extend SONiC for core and backbone use cases, amongst others. For example, the SONiC MPLS working group is looking at enabling MPLS and SR/SRv6 support for SONiC that are more applicable to WAN use cases.

SONiC in the real world

With all the benefits of an open-source NOS, network operators have many questions such as “Is SONiC the right fit for my use case?”, “How does support work?”, “How do I ensure code quality?”, “How do I train my team to build the skill set to manage SONiC?”, and the list goes on. Product adoption is always driven by customer experience. Any product or solution, open-source or not, will be successful only if it provides a seamless user experience. While the many merits of an open-source NOS are attractive, operators still want the security and partnership of a vendor NOS when it comes to support and field deployments. So how do we achieve the best of both worlds?

Network operators assessing SONiC either have a very strong self-driven ecosystem equipped to handle an open NOS or they’re trying to understand the deployability of an open NOS. Operators with a self-sufficient ecosystem tend to gravitate towards customized SONiC to suit their specific network requirements. This might involve customizing community SONiC to create a private distribution (BYO – build your own) or they can rely on external vendors that create commercial distributions built from community SONiC. On the other hand, operators trying to gain more experience with open NOS for relatively simpler use cases might want to rely on community SONiC, where there's a fine balance in retaining the open-source nature of SONiC along with its validation on vendor hardware.

Figure 3. SONiC consumption model

While assessing a network rollout, there are certain evaluation criteria that an operator needs to consider. These evaluation criteria are independent whether the network solution in place is open or closed but depending upon the target ecosystem the responses to these criteria might differ.

Table 1. SONiC deployment evaluation criteria

The Cisco 8000 Series advantage

The high-performance Cisco 8000 Series of routers and switches is based on the Cisco Silicon One ASIC, making these devices three times more power efficient and twice as dense as industry incumbents. A wide variety of fixed and modular form-factors are available, while its power savings, run-time completion efficiency, and SDK portability offer unique advantages of the Cisco 8000 that greatly facilitate SONiC onboarding. As a strategic investment, every new platform is compatible with SONiC for the ability to leverage one silicon and one software end-to-end in different roles across use cases.

Figure 4. SONiC – The Cisco advantage

Support

The saying *“With great power comes great responsibility”* aptly applies to any open-source ecosystem. When deploying a production network, every operator is looking for holistic triage, faster resolution, predictable SLAs, and accountability. So how does this apply to SONiC?

Operationalizing SONiC on vendor hardware can be visualized as three layers. The bottom two layers consist of vendor-specific components – hardware systems at the very bottom followed by the infrastructure software that consists of SAI APIs, SDK, BSP/platform drivers, and other glue logic to seamlessly abstract hardware intricacies from the overlying operating system. By itself, SONiC looks like a constellation of open-source components and custom code, depending on whether customized SONiC is in play or not. With plug and play, accountability still sits with respective stakeholders for their components, leading to a shared responsibility support model. For Cisco-validated SONiC, every shipping platform will go through intensive customer and use case centric testing, with major and minor release cadence for community SONiC. Major releases will support newer features while minor releases provide bug fixes.