

Software for Open Networking in the Cloud (SONiC)

A Reference Open Networking Software for OCP

# Executive Summary

Software for Open Networking in the Cloud (SONiC) is a collection of software that form a fully operational network switch device. SONiC runs on multiple switch hardware platforms and ASIC’s through its use of the Switch Abstraction Interface (SAI). SONiC’s modular architecture and lean software stack keep the base software reliable, fast to test, debug and fix software bugs. Sonic comes with complete functionality for a simple layer 3 switch, but is designed to be extensible for community needs.

Contents

[Software for Open Networking in the Cloud (SONiC) 1](#_Toc459025460)

[A Reference Open Networking Software for OCP 1](#_Toc459025461)

[Executive Summary 2](#_Toc459025462)

[Contents 2](#_Toc459025463)

[Figures 3](#_Toc459025464)

[Revision History 4](#_Toc459025465)

[Overview 4](#_Toc459025466)

[License 5](#_Toc459025467)

[Background 5](#_Toc459025468)

[Design 6](#_Toc459025469)

[**System Device Interface (SDI)** 6](#_Toc459025470)

[**Platform Adaptation Services (PAS)** 7](#_Toc459025471)

[**SONiC Object Library** 7](#_Toc459025472)

[**Switch State Service (SwSS)** 8](#_Toc459025473)

[**Network Adaptation Service (NAS)** 8](#_Toc459025474)

[**Switch Abstraction Interface (SAI)** 9](#_Toc459025475)

[Test Plan 9](#_Toc459025476)

[Checklist for Maintenance 9](#_Toc459025477)

[Checklist for Governance 10](#_Toc459025478)

[Roadmap 10](#_Toc459025479)

[Supporting Documents 11](#_Toc459025480)

Figures

[Figure 1 SONiC target architecture 4](#_Toc458981332)

[Figure 2 SONiC current architecture 5](#_Toc458981333)

[Figure 3 System device control path 8](#_Toc458981334)

[Figure 4 Switch State Service architecture 9](#_Toc458981335)

[Figure 5 NAS integration in SONiC 10](#_Toc458981336)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Version | Description |
| Xin Liu | 8/25 | V1 |  |
|  |  |  |  |

Overview

SONiC delivers a simple layer 3 switch software stack which enables cloud operators and enterprises to take advantage of network switch hardware innovation with a framework to build upon an open source code for apps on the network switch and the ability to integrate with multiple platforms. We believe it’s the final piece of the puzzle in delivering an open sourced switch platform that can share the same software stack across hardware from multiple switch vendors. SONiC is led by Microsoft and includes contributions from Dell, Mellanox, Arista, Accton, Broadcom, and Barefoot.

Figure 1 describes the Sonic software architecture.



Figure 1 - SONiC Architecture

# License

Sonic is a suite of many open source components, under various licenses. The Microsoft, Dell and Mellanox contributed code is licensed under the Apache License, Version 2.0 where compatible and under other licenses where required, such as GPL v2 for Linux kernel drivers. Complete license details can be found at the list of [SONiC source code repositories](https://github.com/Azure/SONiC/blob/gh-pages/sourcecode.md).

# Background

All cloud operators rely on high-speed, highly available networks to power their services, which is what makes SDN so critical. It’s vital that network operators be able to rapidly add networking features they need while managing any feature changes that increase risk and complexity.

We believe there are many excellent switch hardware platforms available on the market, with healthy competition between many vendors driving innovation, speed increases, and cost reductions. However, it is challenging to integrate the different software running on each different type of switch into a cloud-wide network management platform. Ideally, one would like all the benefits of the features that have implemented and the bugs that have fixed to remain intact, even as one rides the tide of newer switch hardware innovation.

Through open sourcing SONiC, contributions to and adoptions from the broader networking community will continue to improve SONiC’s quality. With this initial contribution we aim to form a robust community that can improve, customize, extend and evolve this over time.

# Design

SONiC consists of the following components:

## **System Device Interface (SDI)**

A system device refers to a hardware component, such as:

* Fans/cooling devices
* Power supplies
* Temperature sensors
* LEDs
* EEPROM
* Programmable devices.
* Transceivers

All hardware components except NPUs are abstracted as system devices.

The SDI API defines a low-level platform-independent abstraction for all types of system devices. Only system device drivers that implement SDI API are hardware-specific; the API itself is hardware-independent. The implementation of SDI API can use any approach suitable for a given platform or vendor:

* 'Sysfs' access to Linux kernel device drivers
* user space devices drivers using UIO or other methods
* new vendor specific kernel modules, accessible through sysfs, netlink or ioctl calls
* combination of any of the above methods

In general, other approaches not mentioned above are also possible, as long as the implementation supports the SDI API.

## **Platform Adaptation Services (PAS)**

PAS provides a higher-level abstraction and aggregation of the functionality provided by the System Device Interface (SDI) component. In addition, the PAS implements the object models associated with system devices. PAS monitors the status of system devices and reports (publishes) status changes or faults as events. It also allows user applications to retrieve current status information and set the control variables of system devices.

An application will use the PAS Object model through the Object Library. PAS service use SDI to access the drivers which will (as an option) communicate with the Sysfs file system which in turn communicates with the actual hardware device drivers.



Figure 3 System device control path

## **SONiC Object Library**

As of this writing, the object library is being updated as PAS and NAS are integrated into the project. As this occurs, this document will be updated.

The SONiC Object Library (Object Library) mediates interactions between SONiC applications and external applications. The Object Library infrastructure defines two types of application roles: clients and servers. Client applications execute create, set, get, and delete operations on objects. Server applications execute operations requested by clients.

In addition, the Object Library infrastructure supports a publisher/subscriber model. Server applications publish relevant events; client applications can subscribe (register) for specific events and objects. Client applications can register for events generated when objects are created, modified, or deleted. The publisher/subscriber approach and object-centric operations allow for the completely independent operation of client and server applications. Custom-written applications use the Object Library API to communicate with the SONiC components.

## **Switch State Service (SwSS)**

The Switch State Service (SwSS) is a collection of software that provides a database interface for communication with and state representation of network applications and network switch hardware. Network applications read and write to APP\_DB. Example applications include a netlink route syncer, quagga FPM route syncer, access control list (ACL) control, QoS control, load balancer, telemetry control and so on.

The Orchestration agent reads and writes data between APP and ASIC databases. The Orchestration agent is responsible for any necessary logic to verify and transform the data into SAI objects which can be processed by syncd.

The syncd process reads and writes SAI objects between the ASIC\_DB and the SAI SDK.



Figure 4 Switch State Service architecture

## **Network Adaptation Service (NAS)**

The Network Adaptation Service (NAS) manages the high level network processor (NPU) abstraction and adaptation. The NAS abstracts and aggregates the core functionality required for networking access at Layer 1 (physical layer), Layer 2 (VLAN, link aggregation), Layer 3 (routing), ACL, QoS and network monitoring. The NAS enables adaptation of the low level switch abstraction provided by the Switch Abstraction Interface to:

* Standard Linux networking APIs and Linux Interfaces
* SONiC specific Object Library API functionality.

In addition, the NAS is responsible for providing packet I/O services, using the Linux kernel IP stack.



Figure 5 NAS integration in SONiC

## **Switch Abstraction Interface (SAI)**

The switch abstraction interface is a standardized C API to the switching ASIC. This API is normally implemented by an SDK specific to the Switch ASIC. More information on SAI is available at the [SAI GitHub repository](https://github.com/opencomputeproject/SAI).

# Test Plan

The [SONiC build system](https://sonic-jenkins.westus.cloudapp.azure.com/) executes automated tests for both unit and integration tests. We are investigating utilizing UNH IoL to perform open hardware integration testing for SONiC.

# Checklist for Maintenance

Currently the code is maintained in GitHub and the development uses GitHub-based best practices. All code changes are reviewed publicly (using GitHub’s online code review tools) and approved by a repository maintainer. Sonic consists of a list of repositories each of which have their own maintainers. The SONiC project’s build repository defines the complete list of sources used to create the project and is maintained by Microsoft.

It is mandatory that all entities (including the ones listed above) with code approval or commit capability, i.e., are either committers/maintainers into the SONiC project be OCP members. We are open to expanding the committers list as other contributors/authors emerge. New contributors/authors cannot become committers/maintainers without first being an OCP member.

In the event that all maintainers are permanently unavailable, a duly appointed representative of the Open Compute Project may take over the project.

Software releases will be made as time and major features are committed. While many open source projects with regular committers have a time-based release model, at least for the near future until the projects popularity increases, we will follow a feature-based release schedule.

# Checklist for Governance

This is the list of current governance sites which may change with acceptance into OCP:

Website: <https://azure.github.io/SONiC/>

Mailing list: sonicproject@googlegroups.com

IRC: N/A

Mirror: N/A

GitHub: https://github.com/Azure/SONiC

Wiki: <https://github.com/Azure/SONiC/wiki/SONiC-Documentation>

SONiC Governance document: <https://github.com/Azure/SONiC/blob/gh-pages/governance.md>

# Roadmap

Features which are implemented, tested and deployed in production.

* Compatible with image install using ONIE and Aboot (Arista bootloader)
* Incremental software update
* BGP
* ECMP
* QOS - ECN
* Priority Flow Control (PFC - 802.1Qbb)
* WRED
* COS
* SNMP
* Syslog
* LLDP
* NTP
* LAG
* tcpdump for packets sent to CPU

Features currently in development

* Fast reboot (reboot with less than 30 seconds data plane impact)
* Control plane packet rate limiting (in code review)
* VLAN (in code review and testing)
* ACL permit/deny
* Netbouncer (tunnel decap)
* SNMP subagent hardening
* LLDP update for SwSS
* Warm boot (ISSU-like feature, less than 1 second data plane impact)

Committed roadmap

* Open source build (make it easier to build SONiC)
* Open source mgmt repo (high priority)
* Open source test (high priority)
* DHCP relay agent
* Buffer monitoring
* ERSPAN/Everflow (mirror packet via ACL and encap in GRE)
* IPV6
* Hardware table usage and capacity reporting (FIB, ACL, etc)

# Supporting Documents

The majority of the technical documents live in the [GitHub](https://github.com/Azure/SONiC) directory in the source, including:

* Getting Started Guide
* [SONiC Architecture](https://github.com/Azure/SONiC/blob/gh-pages/architecture.docx)
* SONiC Code
* Link to SAI Code
* Monitoring and Management Tools
* Imaging and Building tools
* SONiC Feature List