Becoming a Contributor to SONiC

By Wajahat RaziMarch 18, 2024No Comments

SONiC is a vibrant open-source project that has an active community. The community invites contributors to join hands and contribute towards its development through designing, testing, feature enhancement, documentation, or active participation in various workgroups.

Governance

The project’s governance is focused on maintaining a technical meritocracy. It consists of multiple repositories, each led by a maintainer. Contributors can submit various types of work, and maintainers have the authority to merge pull requests. Conflict resolution is primarily handled within each repository, with Project Leaders intervening when necessary.

Community Mailing List

The community mailing list is known as “sonic-dev@lists.sonicfoundation.dev”. It serves as a platform for developers, contributors, and members interested in the development and ongoing activities related to SONiC like attending the weekly discussions.

Community Sub-groups

The community consists of several subgroups, each with a specific focus. These include Testing, Management, VRF, MCLAG, EVPN, Breakout, SAI Security, SONiC Application, SONiC Chassis, SONiC Platform, SONiC Kubernetes, SONiC MPLS, SONiC Yang, SAI IPSEC, SONiC-Documentation, SONiC-PINS, SONiC-OTN, SONiC DASH, and SONiC-Routing. These subgroups focus on various aspects, including testbed improvement, management standardization, VRF enhancement, MLAG options, EVPN design, port break-related features, security-related APIs, documentation, SDN interfaces, optical transport support, disaggregated APIs, and routing performance and scale.

Community Calendar

Subscribing to a SONiC calendar is a useful way to stay up-to-date with important events, updates, and releases. The calendar is hosted online and can subscribed to using calendar applications, such as Google Calendar, or Microsoft Outlook.

Weekly Meetings

Community meetings are held every week and each workgroup conducts its meetings separately. community meeting occurs every Tuesday from 8:00 a.m. to 9:00 a.m. PST, the test-subgroup meeting occurs every alternate Wednesday from 8:00 a.m. to 9:00 a.m. PST, SONiC EVPN subgroup convenes the same alternating Wednesdays but from 9:00 a.m. to 10:00 a.m. PST, SAI ECMP workgroup gathers every Friday from 2:30 pm to 3:15 pm PST, SONiC Chassis Subgroup meets every Wednesday from 9:00 am to 11:00 a.m. PST, SONiC Kubernetes Subgroup, and SAI IPSEC Subgroup both meet every alternate Friday and Wednesday, respectively, from 8:00 am to 9:00 am PST.

How to Contribute

Members can contribute by developing a feature or enhancing an existing feature. To do so, the members must inform the community of their intent, provide a rough plan about the design document availability date, code availability date, and also identify reviewer(s). Finally, ensure that your feature is merged into the codebase before the release.

Development

Creating the design document using the template provided at SONiC HLD Template, sending the design document as a pull request to the SONiC GitHub repository and the community will review the design, where feedback and comments may be provided during the review. Once the design is approved by the community and merged into the SONiC GitHub repository.

Design: Members will code features as per the approved design, create a pull request in the relevant repository, follow code nomenclature, and use appropriate programming language format (e.g., Python, C/C++, Ruby). The community will review and merge the code upon approval.

Code: Members can actively participate in SONiC's review process by reviewing design documents, code, test cases/scripts, and related materials. They can attend design review meetings, offer feedback, and comment on pull requests to help ensure timely feature development for the roadmap.

Testing & Bug-Fixing

Members are encouraged to test SONiC on their platforms and report issues in the relevant SONiC repositories. They can also make valuable contributions by reviewing existing test cases and introducing new ones to the sonic-mgmt repository. Bug fixing is encouraged by the SONiC community. Members should join the issue triage subgroup and contribute to root cause analysis and reviews to resolve issues.

Documentation

Members should review and enhance SONiC technical documents, including the Configuration Guide, CLI Guide, and Wiki pages. They are also welcome to document deployment scenarios and use cases on the SONiC Wiki. Some relevant resources are High-level design documents, Test plans, SONiC release notes, SONiC command reference, SONiC YANG models, SONiC YANG model guidelines, and Technical Charter.

Fork & Clone

Fork the SONiC repository, clone it, and create a private branch. Make changes, add, commit, and push them. Create a pull request to the original repository with labels and details. For CLI changes, update Command-Reference.md in sonic-utilities. Optionally, use an HLD PR to track code PRs.

Release Process

The SONiC release process includes Planning (feature commitments and plans), Design (detailed document review), Coding and testing (coding, PRs, testing), and Release (branching, stabilization, release notes). Features unfinished by the branching date are deferred to the next cycle to maintain a structured development process. Regular communication is also emphasized.

Conclusion

SONiC with its active and diverse community, offers various ways for contributors to get involved. The community's communication hub provides a platform for discussions and updates. SONiC encompasses multiple subgroups, each focused on specific aspects, offering ample opportunities for participation. Regular community meetings, workgroup sessions, and guidelines for contribution and collaboration are emphasized, promoting a structured development process for SONiC.

About the Author:

Wajahat Razi, a Network Design Engineer at xFlow Research Inc., is a professional specializing in crafting robust network architectures. With a core focus on Network Design, his expertise lies in shaping efficient and scalable

network solutions. With a keen interest in cutting-edge technologies, he has extensively researched and comprehensively understands the architecture of SONiC, particularly its relevance in modern data centers. #xFlowResearch #SONiC #Networking

Demystifying SONiC: Its Importance in Modern Data Centers
By Wajahat RaziMarch 4, 2024No Comments

SONiC – Software for Open Networking in the Cloud – was developed by Microsoft for its Azure data centers and open-sourced in 2016. It is based on the Linux distributive Debian. SONiC uses SAI to decouple software from the underlying hardware that allows it to run on multi-vendor ASICs. According to Gartner, with the increase in interest for SONiC, there is a strong possibility that SONiC will become analogous to Linux for networking operating systems in the next three to six years. Analysts predict that SONiC switching could exceed $5 billion in revenue by 2026.

Over the last half-decade or so, there has been a notable increase in the level of interest and engagement surrounding SONiC. Data centers are rapidly expanding and evolving to an architecture that is disaggregated and Software-defined to meet the exponential growth in AI, Big Data, and high-performance computing. SONiC offers serviceability and adaptability with the potential to cope with this demand in Data Centers. Gartner also predicts that by 2025, 40% of organizations that operate large data center networks (greater than 200 switches) will run SONiC in a production environment.

Proprietary network devices installed in data centers use exclusive operating systems leading to vendor lock. Moreover, these NOS solutions incur high costs due to license fees and support contracts. SONiC’s openness gives customers the flexibility to switch platforms without changing the software stack. Customers can seamlessly scale and automate their network by taking advantage of the open-source OS, resulting in savings and more control. Moveover, it is multi-vendor supported and it uses open & standardized protocols like REST APIs for configuration and monitoring. This means the network can be managed and configured using common tools regardless of the underlying hardware brand.

The modular architecture of SONiC is well-suited for data center environments due to its ability to provide serviceability, enhancements, and support for zero downtime. For example, Individual software components like the OS updates, drivers etc. can be upgraded, or new protocols or applications can be added dynamically without affecting other modules or bringing down the whole switch. Hence, customers are not dependent on the vendor’s product roadmap for innovation and the addition of new features. Instead, they have control over the system’s functionality, allowing them to seamlessly incorporate important features such as automation and applications or address issues like bug fixes and upgrades. This modular approach enables efficient failure recovery and in-service upgrades, allowing for the replacement or enhancement of specific components without disrupting the overall system’s operation. This design empowers customers to maintain and evolve their SONiC deployment in a data center with flexibility and scalability, all while keeping the system running smoothly.

SONiC has the capability to operate from core to edge network, helping streamline deployments, operations, and monitoring in data centers and even in ISPs or enterprise networks. It provides L2, L3 and L4 functionality, serving as the solid foundation for the overlay network services and applications running in the data center.

SONiC is a reliable NOS. As per a study conducted by a team, led by Dave A. Maltz, Technical Fellow & CVP, Azure Networking, Microsoft & Chair of the Governing Board for the SONiC Foundation, that tracked 180,000 switches in Azure data centers for three months, only two percent of network switches will fail

within three months, but that figure is cut in half if the vendor’s operating software is replaced by SONiC. The study found these were one percent more likely to survive three months – nearly halving the failure rate.

Deep dive into SONiC Architecture & Design
By Wajahat RaziMarch 11, 2024No Comments

Unlike traditional and proprietary network operating systems, SONiC – Software for Open Networking in the Cloud – has emerged as a groundbreaking open-source network operating system (NOS) designed to redefine how network infrastructure is administered. Among the reasons behind its popularity are its capabilities to provide L2, L3 & L4 functionalities. Its modular structure is based on Debian Linux, and the SAI – Switch Abstraction Interface – layer which essentially acts as a translator between the silicon and the software. It makes SONiC a disaggregated NOS that has the ability to work with multi-vendor silicons reducing vendor lock-in and giving a glimpse into the future of networks.

SONiC stands out as a complete Network Operating System. The architecture comprises a collection of software components & tools that work in harmony to offer serviceability, extensibility, development agility, and resource efficiency. All major subsystems are enclosed within Docker containers and facilitate Inter-Process Communication within the centralized system.

Application Containers

SONiC architecture employs user-space services within containerized modules, each serving specific networking functions. These services include FRR routing stack (BGP container) for routing, LLDP for topology mapping & device discovery, SNMP server (based on NGINX) for monitoring & configuration management, and PMON for hardware management & sensor monitoring. The Teamd container ensures network robustness, managing LAG groups and link status for redundancy. The DHCP-Relay container facilitates DHCP communication between clients and servers, ensuring IP assignments and configuration. Modularity allows tailored configurations to meet network needs.

Infrastructure Containers

Switch State Services (SWSS), Database, and Synchronization daemon (SyncD) serve different roles in a SONiC as compared to application containers. They are responsible for the fundamental management and configurations like switch configuration, Port updates, ASIC management & maintaining key-value configurations.

Database – REDIS

Database container hosts the redis-database engine. Redis – an in-memory database serves as a critical datastore for SONiC components, enabling efficient storage and retrieval of network states, configuration, and operational data in key-value pairs. The redis-server exposes a unix socket & binds to the host filesystem allowing SONiC components to access the database. The database container maintains several databases to organize data according to its nature and use, such as Application, Configuration, State, Counter, and ASIC-specific DBs.

Switch States Services

SWSS functions as the central communication hub for SONiC’s modular components, serving as a bridge between the high-level configuration and control interfaces of SONiC, and the underlying switch hardware. Its primary role is to maintain state consistency among various modules and applications. SWSS connects to the centralized Redis engine to retrieve and update data that is stored in key-value pairs. This includes data related to VLANs, ports, routes, etc. Orchagent is an essential component of the SWSS container, translating configurations from the APPL_DB as per the SAI API and publishing them into the ASIC_DB. These

instructions are communicated with the switch’s ASIC driver via the SAI interface to apply configuration changes and update the switch hardware accordingly.

Synchronization Daemon – The way to talk to hardware!

SAI – Switch Abstraction Interface – is designed to be vendor-agnostic, which means SONiC can work with a wide range of switch hardware without having to undergo extensive modifications or adaptations. It acts as an intermediary between SONiC and Hardware-specific switch ASICs (Application Specific Integrated Circuits). The primary role of Syncd is to synchronize and push network state & configuration changes to the hardware and simultaneously collect real-time information from the hardware. Its role involves setting up and configuring ASICs to match the desired network configuration, including data plane forwarding, switch port configuration, VLANs, and other networking functionalities. SyncD leverages the SAI API to interact with the hardware components using ASIC SDK. Hardware vendors typically provide standard SAI interfaces, implemented using vendor specific SDKs (Software Development Kit) required to drive their ASICs. These drivers are essential to harness the full capabilities of the underlying hardware as they enable hardware acceleration and optimization, improving network performance and reducing the load on the main CPU.

Linux Kernel

SONiC leverages linux kernel capabilities like creating Network namespaces, separate network stacks with their own interfaces, routing tables, firewall rules, and settings to achieve network isolation, useful for managing multiple ASICs), Inter-Process Communication mechanisms which SONiC uses for communication between different network services and daemons, enabling seamless cooperation among SONiC components, ACL functionalities like firewalling, packet mirroring, Policy Based Routing (PBR), and control plane ACL, and Switch memory management, including network-specific functionalities, such as managing network configurations, Quality of Service (QoS) policies, packet forwarding, and other switch-related tasks. SONiC defines and controls how memory is allocated and used for networking purposes, including switch memory management for QoS maps, shared buffers, policers, schedulers, and queueing algorithms. It also integrates various features related to switch memory management, like Weighted Random Early Detection (WRED) and Explicit Congestion Notification (ECN). It supports NAT tools like SONiC supports NAT, tools like iptables and nftables . SONiC can use Berkeley Packet Filter (BPF) to filter and selectively process Netlink messages. BPF filter programs can be attached to specific Netlink sockets in the kernel, and they are executed for each incoming Netlink message, deciding whether to accept or drop the message.

Configuration Management

There are several ways to configure the SONiC NOS via the SONiC-CLI, REST-API and minigraph. Free Range Routing (FRR) CLI for routing configurations. The SONiC-CLI, which is based upon a CLICK Python library, provides a SONiC-centric and user-friendly way to interact with the operating system, making it easier to configure network modules, view status information, troubleshoot network settings, and access operational data such as interfaces, routing tables, and system logs.

With SONiC, you can use SSH to log in to the CLI from a remote location. The REST API in SONiC offers a programmatic way to interact with the network operating system, following REST principles and using standard HTTP methods like GET, POST, PUT, and DELETE. It employs a client-server architecture for seamless communication. Additionally, YANG models play a crucial role in SONiC by defining network configurations and operational data. SONiC also uses a JSON file named “config_db.json” to store configuration data. This file can be edited to configure various aspects of the NOS, useful for making bulk configuration changes or for scripting configuration updates.

Conclusion

SONiC, a transformative NOS, revolutionizes network management with its modular, vendor-agnostic design. It offers broad hardware support,and programmability through user and kernel space, enhancing network performance and scalability. SONiC’s open-source, containerized architecture empowers network operators to tailor their infrastructure, fostering automation and adaptability in the dynamic data center landscape. It represents a future where open-source, extensible solutions simplify modern networking for administrators.

About the Author:

Wajahat Razi, a Network Design Engineer at xFlow Research Inc., is a professional specializing in crafting robust network architectures. With a core focus on Network Design, his expertise lies in shaping efficient and scalable network solutions. With a keen interest in cutting-edge technologies, he has extensively researched and comprehensively understands the architecture of SONiC, particularly its relevance in modern data centers. #xFlowResearch #SONiC #Networking