SONiC Architecture

Embracing a New Networking Paradigm

Introduction to SONiC

Overview of SONiC

- Developed by Microsoft for Azure data centers
- Open-sourced in 2016
- Based on Debian Linux

Key Features

- Utilizes SAI for multi-vendor ASIC support
- Supports a wide range of networking protocols
- Decouples software from hardware
- Supports functionality to all layers

Benefits and Use Cases

Advantages of SONiC

- Vendor-agnostic platform
- Cost-effective due to open-source nature
- Modular architecture for scalability and flexibility

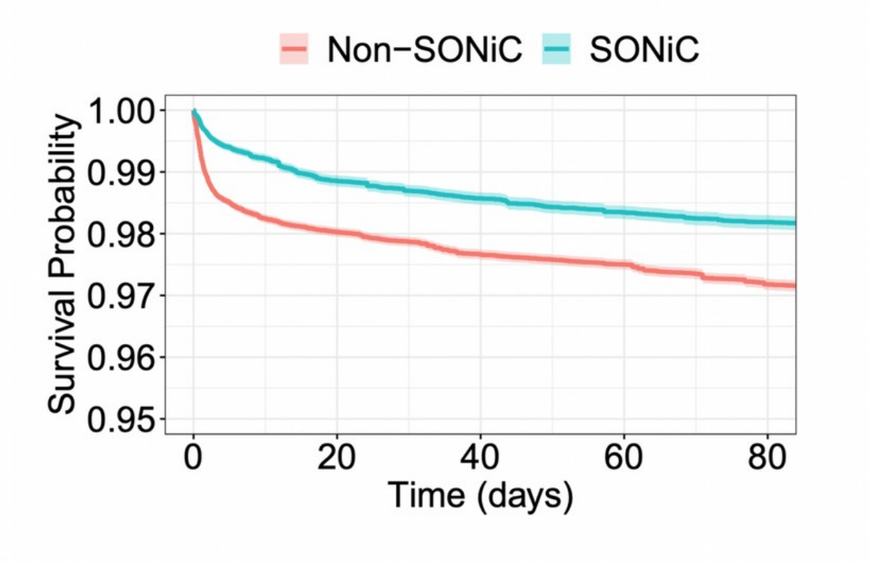
Use Cases

- Increasing adoption in large data center networks
- Seamless scalability and automation
- Standardized protocols for easy management and configuration

Benefits and Use Cases

Reliability of SONiC

- Study demonstrating reduced switch failure rates in Azure data centers with SONiC
- High survivability and stability in demanding production environments



Adoption and Future Trends

- Gartner's prediction of SONiC becoming analogous to Linux for networking OS within years
- Expected widespread adoption across large-scale data center networks by 2025

Why Choose SONiC Over Other Operating Systems?

- Vendor Agnostic
- Open Source
- Decoupled Software from Hardware
- Modular Architecture
- Standardized Protocols
- Scalability and Automation
- Reliability
- Community Support

SONiC System Architecture Overview

- Various modules interact through a centralized and scalable infrastructure
- Relies on Redis database engine for key-value storage and messaging
- Utilizes publisher/subscriber messaging paradigm for data distribution
- Modules are containerized using Docker for modularity and isolation
- Complete Network Operating System with a collection of software components
- Offers serviceability, extensibility, development agility, and resource efficiency
- Encloses all major subsystems within Docker containers for IPC (Inter-Process Communication)

Interactions Between Components

- Blue arrows indicate interactions with the Redis engine
- Black arrows represent other interactions (e.g., netlink, file-system)

SONiC Architecture

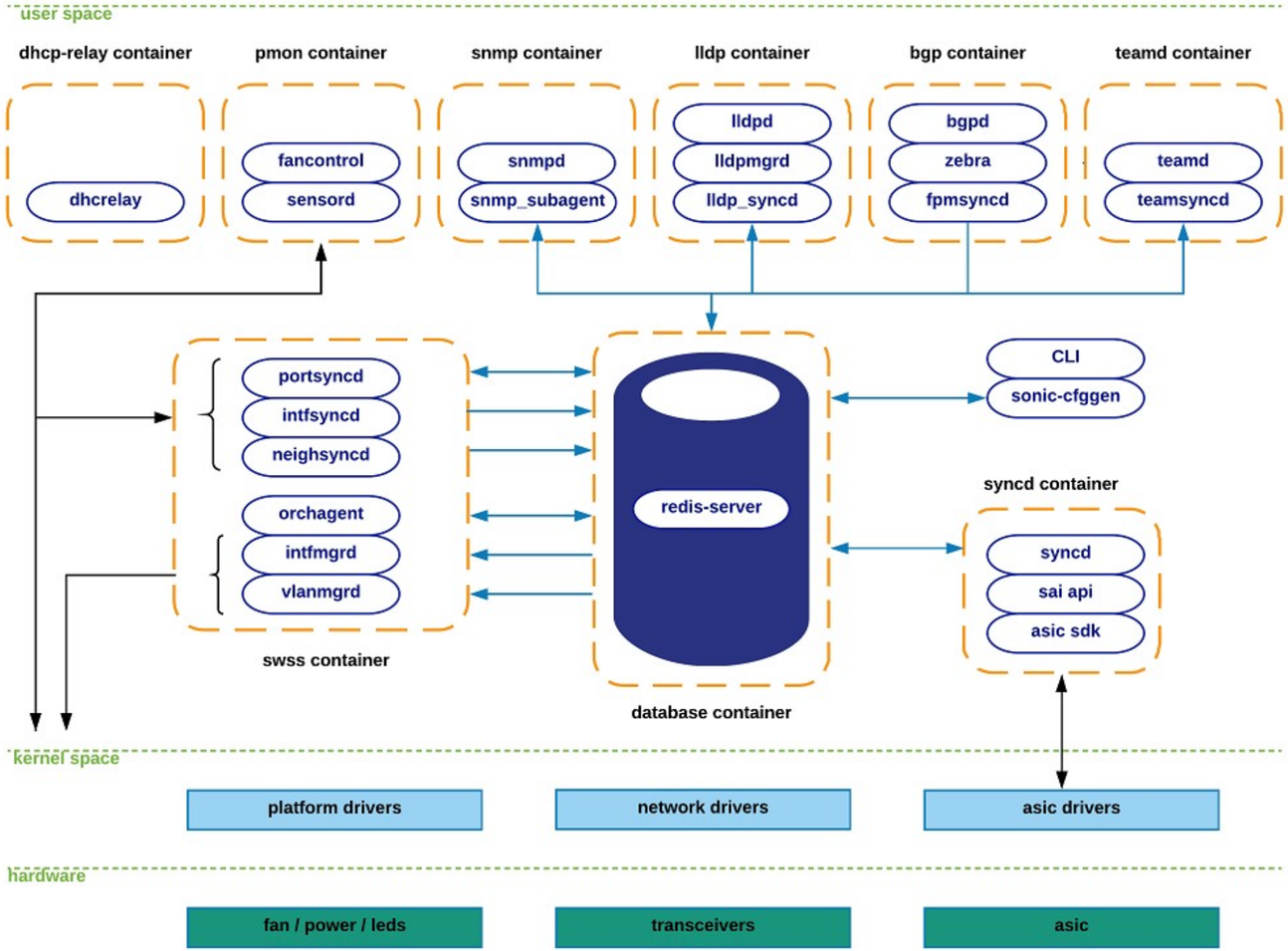
Components in Docker Containers

Application Containers

- Dhcp-relay
- Pmon
- Snmp
- Lldp
- Bgp
- Teamd

Infrastructure Containers

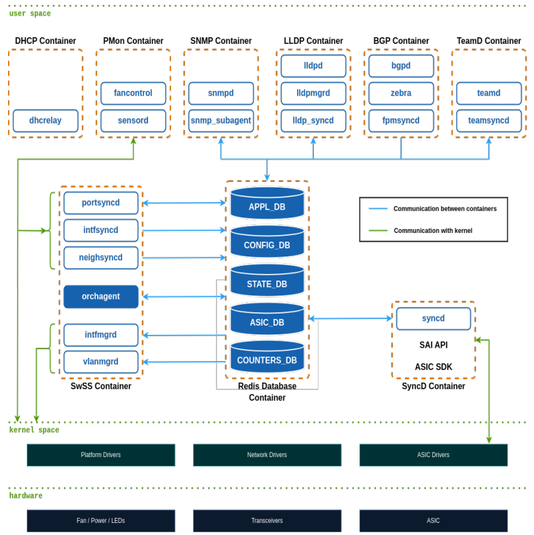
- Database
- Swss
- Syncd



SONiC Subsystems Description

Teamd (teaming daemon) container

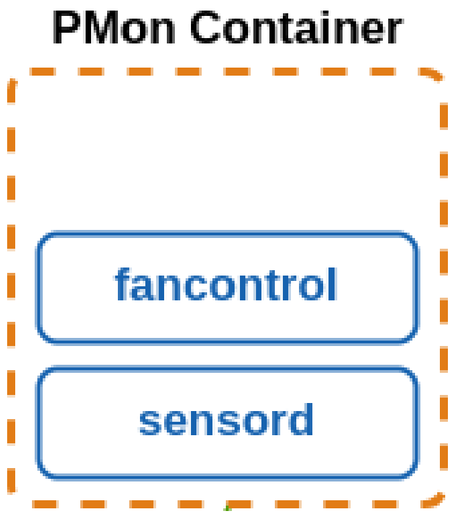
- Manages Link Aggregation (LAG) functionality in SONiC devices
- Implements LAG protocols such as LACP (Link Aggregation Control Protocol) and static LAG
- Creates and manages team interfaces to aggregate multiple physical ports into a single logical interface
- Enables higher bandwidth, load balancing, and fault tolerance in networking
- Utilizes the "teamd" tool, a Linux-based open-source implementation of LAG protocols, to handle link aggregation
- Interacts with south-bound subsystems through the "teamsyncd" process for low-level configuration and hardware interaction



SONiC Subsystems Description

Pmon (Process Monitor) container:

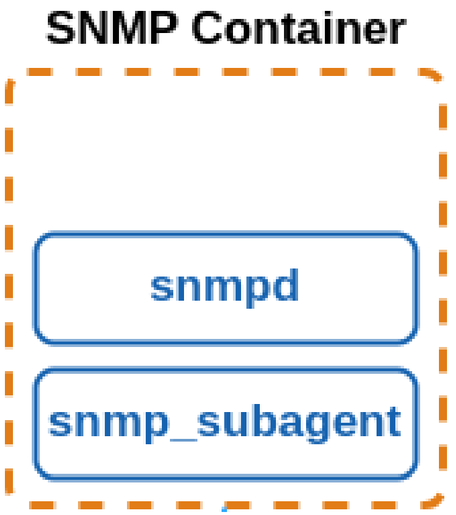
- Manages hardware monitoring and sensor data logging in SONiC devices
- Runs the "sensord" daemon to periodically collect sensor readings (e.g., temperature, voltage, fan speed) from hardware components
- Alerts system administrators when critical alarms or thresholds are triggered based on sensor data
- Hosts the "fancontrol" process to monitor and manage fan-related states using platform-specific drivers
- Proactively maintains system stability and reliability through real-time health monitoring and management of hardware conditions



SONiC Subsystems Description

Snmp (Simple Network Management Protocol) container

- Simple Network Management Protocol that allows administrators to monitor and manage network devices from a centralized management station
- SNMP Daemon (snmpd): Handles incoming SNMP requests from network elements
- AgentX (SNMP Extension): Facilitates communication between SNMP manager and subagents
- Subagents (sonic_ax_impl): SONiC's implementation of SNMP subagents that gather data from SONiC databases by using MIB
- Communication Flow:
  SNMP manager → AgentX → Subagents for data gathering



SONiC Subsystems Description

DHCP Relay Container

- DHCP-Relay Agent forwards DHCP requests from subnets without access DHCP servers
- Enables devices in one subnet to obtain IP addresses from DHCP servers in other subnets
- Listens for DHCP messages from local clients and forwards them to designated DHCP servers
- Receives DHCP responses from servers and relays them back to requesting clients

SONiC Subsystems Description

Lldp (Link Layer Discovery Protocol) container:

LLDP enables devices to automatically discover and identify neighboring devices connected to the same network segment

LLDP Daemon ("lldp"):

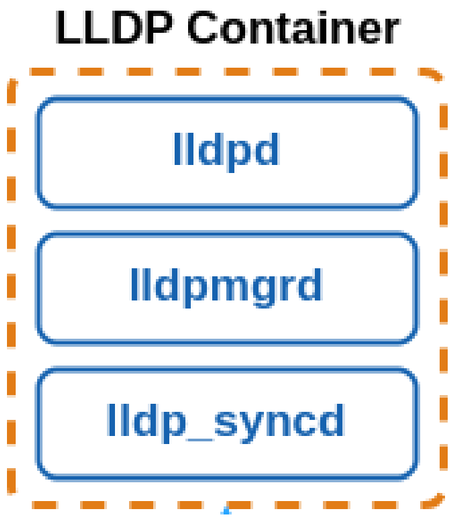
- Manages LLDP functionality within SONiC
- Establishes LLDP connections with neighboring devices to exchange system capabilities and network information

LLDP Syncd ("lldp_syncd"):

- Uploads LLDP-discovered state to the centralized message infrastructure (Redis engine APPL_DB)
- Ensures LLDP state is accessible to other applications interested in consuming this information, such as SNMP

LLDP Manager ("lldpmgr"):

- Provides incremental configuration capabilities to the LLDP daemon
- Subscribes to the STATE_DB within Redis to receive real-time updates on network state



SONiC Subsystems Description

Bgp (Border Gateway Protocol) container:

BGP Daemon (bgpd):

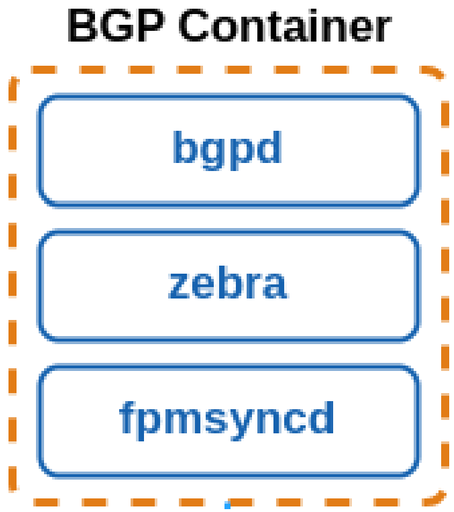
- Implements the Border Gateway Protocol (BGP) for exchanging routing information with external devices
- Receives BGP routing updates from peers over TCP connections
- Updates the BGP routing table based on received updates
- Sends routing updates to the forwarding plane (data plane) for packet forwarding decisions

Zebra:

- Acts as a routing manager within SONiC, managing the kernel's routing table
- Performs interface lookups to determine outgoing network interfaces
- Maintains the Forwarding Information Base (FIB) used for packet forwarding decisions
- Handles route redistribution between different routing protocols (e.g., OSPF, ISIS, BGP)

fpmsyncd:

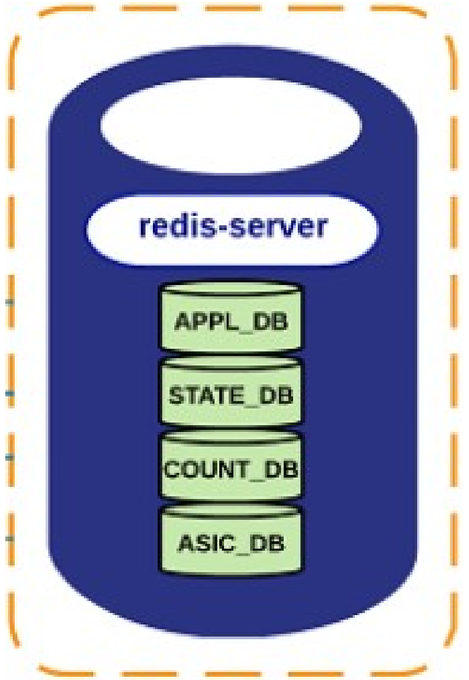
- Collects the Forwarding Information Base (FIB) state from Zebra
- Stores the FIB state in the Application Database (APPL_DB) within Redis
- Ensures synchronization of forwarding information across SONiC devices for efficient packet forwarding



SONiC Subsystems Description

Database Container

- Database Container in SONiC:
  - Hosts Redis database engine for storing network state and configuration information
- Databases within Redis:
  - APPL_DB:
    - Stores operational state information from SONiC applications.
    - Accessed by routing daemons (e.g., bgpd) for routing information and forwarding agents (e.g., zebra) for forwarding state
  - CONFIG_DB:
    - Manages configuration settings created by SONiC applications
    - Interacted with by configuration daemons and VLAN management services
  - STATE_DB:
    - Stores essential operational state data for coordinating subsystem interactions
    - Used by LAG management and VLAN membership services

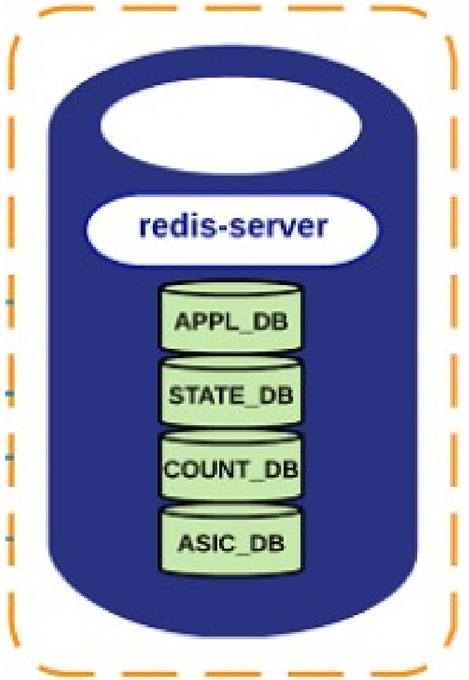


database container

SONiC Subsystems Description

Database Container

- ASIC_DB:
  - Contains ASIC-specific configuration and operational data
  - Interacted with by syncd component for ASIC management
- COUNTERS_DB:
  - Stores port-level counters and statistics for monitoring and analysis
  - Accessed by telemetry services and CLI tools for network diagnostics
- Container Interactions:
  - Each SONiC container accesses specific Redis databases based on its role and functionality
  - Organized use of separate databases facilitates efficient network management and operations within SONiC ecosystem



database container

SONiC Subsystems Description

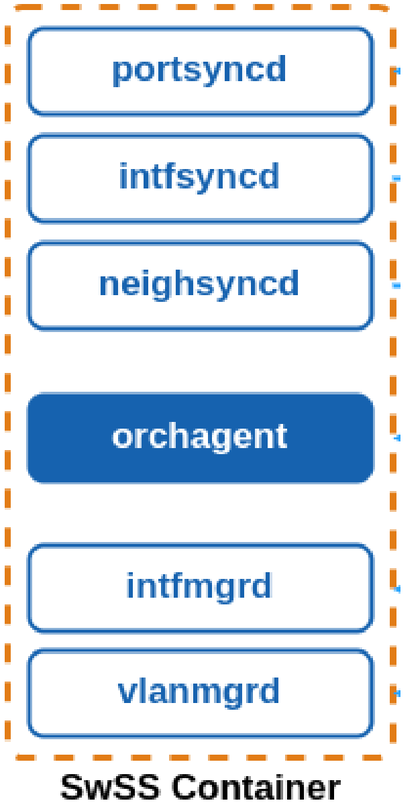
SWSS (Switch State Services) container:

Overview:

- Hosts processes facilitating northbound interactions between SONiC applications and the Redis message infrastructure
- Acts as intermediaries between SONiC applications and network state databases
- Listens for network events, collects relevant state, and updates databases to ensure accurate network information for applications

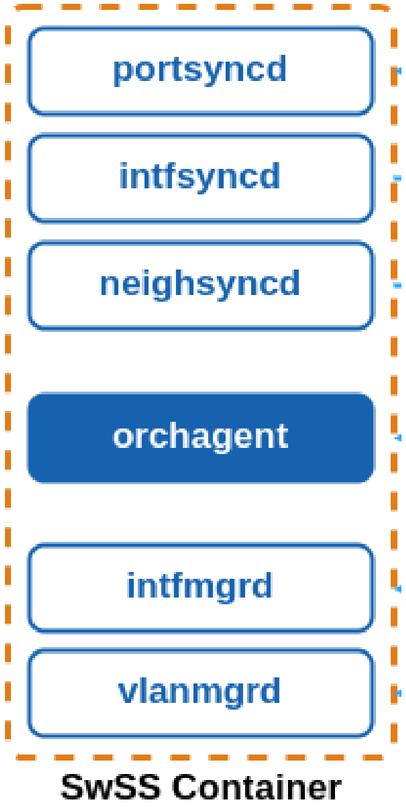
- Northbound Interface:
  - Higher-level (e.g., applications, controllers) → Lower-level (e.g., infrastructure, network devices)

- Southbound Interface:
  - Lower-level (e.g., infrastructure, network devices) → Higher-level (e.g., applications, controllers)



State producers in SWSS

1. Portsyncd:
   a. Monitors port-related events using netlink
   b. Collects physical port information during boot-up from system hardware profiles
   c. Updates APPL_DB and STATE_DB within Redis with collected port state
2. Intfsyncd:
   a. Listens to interface-related netlink events
   b. Manages changes in IP addresses associated with network interfaces
   c. Updates APPL_DB with interface state changes
3. Neighsyncd:
   a. Monitors neighbor-related netlink events triggered by ARP processing
   b. Gathers information about newly discovered neighbors (e.g., MAC addresses)
   c. Populates APPL_DB with neighbor state for L2 rewrite operations
4. Teamsyncd:
   a. Manages Link Aggregation Group (LAG) state within the teamd container
   b. Collects and updates LAG-related state in APPL_DB
5. Fpmsyncd:
   a. Manages Forwarding Information Base (FIB) state within the bgp container
   b. Collects forwarding information from routing daemons
   c. Updates APPL_DB with routing and forwarding state
6. Lldp_syncd:
   a. Manages LLDP state within the lldp container
   b. Handles LLDP-related events and updates APPL_DB with LLDP state



Subscribers in SWSS

- Orchagent
  - Orchagent receives state information from *syncd daemons
  - Orchagent processes and organizes the received state based on predefined rules
  - Orchagent sends the processed data towards a "south-bound interface"
  - Orchagent acts as a consumer when receiving state from sources like APPL_DB
  - It acts as a producer when distributing this processed information to ASIC_DB

- IntfMgrd Process:
  - Manages network interface configurations in the Linux kernel
  - Reacts to state changes from Redis databases (APPL_DB, CONFIG_DB, STATE_DB)
  - Configures network interfaces based on received state
  - Checks for conflicts or inconsistencies before applying changes

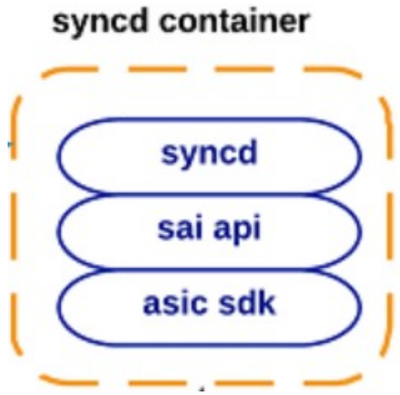
- VlanMgrd Process:
  - Configures VLAN interfaces in the Linux kernel
  - Reacts to state changes from Redis databases (APPL_DB, CONFIG_DB STATE_DB)
  - Configures VLAN interfaces based on received state
  - Ensures dependent conditions are met before applying interface configuration

SONiC Subsystems Description

Syncd Container

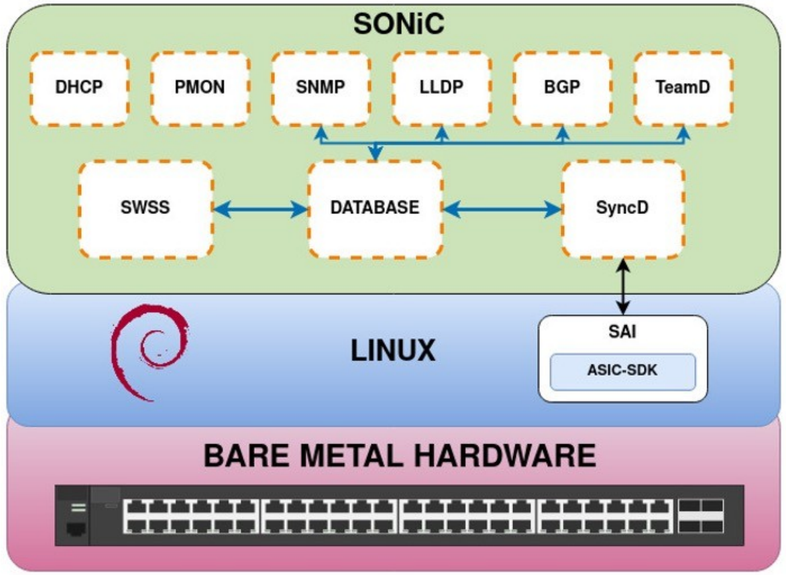
Overview:

- Manages synchronization between network state in software and the hardware ASIC (Application-Specific Integrated Circuit) of the switch
- Executes synchronization logic to initialize, configure, and monitor the switch's ASIC status
- Functionality of Syncd:
  - Integrates with ASIC SDK (Software Development Kit) at compilation time to interact with the hardware
  - Subscribes to ASIC_DB in SWSS (SONiC Switch State Service) to receive state updates from other SONiC components
  - Registers as a publisher to push hardware state updates back to SONiC components



SAI (Switch Abstraction Interface):

- SAI enables SONiC to work with diverse switch hardware seamlessly without extensive modifications
- Acts as a bridge between SONiC (network OS) and hardware-specific switch ASICs
- Provides a standardized API for controlling ASICs, NPUs, or software switches
- syncd uses SAI to interact with hardware components via ASIC SDK
- Supports hardware acceleration and reduces CPU load by leveraging vendor-specific SDKs



ASIC SDK (Software Development Kit):

- Provided by ASIC vendors to manage and control their ASICs.
- Implements the SAI interfaces to interact with hardware-specific features of the ASIC.
- Typically packaged as a dynamic-linked library (DLL) used by syncd to interface with the ASIC.

CLI (Command Line Interface):

- The CLI is built using Python's Click library, which helps create user-friendly and customizable command line tools.
- Users can execute commands and configurations by typing specific commands into the terminal.
- The CLI translates these commands into actions that SONiC modules can understand and execute.

Sonic-cfggen:

- Sonic-cfggen is a component that works closely with SONiC's CLI to manage system configurations and perform related actions.
- When you enter commands through SONiC's CLI, sonic-cfggen is invoked to process these commands.
- Sonic-cfggen performs configuration changes or interacts with SONiC modules based on the commands received.

THANK YOU

CONTACT INFORMATION

www.xflowresearch.com