FlexSwitch

A Reference Network Stack for OCP

Executive Summary

FlexSwitch is a reference network stack for Open Compute Project (OCP) compatible network hardware. FlexSwitch is designed to run on top of a Network Operating System (NOS) – such as OCP's Open Network Linux (ONL). FlexSwitch's goal is to provide the much needed protocol suite to allow OCP network hardware to be deployed into production environments with an entirely OCP-based software solution. This suite comprises of Open Network Install Environment (ONIE) as the bootloader, ONL as the NOS, Switch Abstraction Interface (SAI) as the hardware abstraction layer (HAL), and a complete layer 2 and layer 3 control-plane provided by FlexSwitch. We believe that an open, modern network protocol suite will invigorate the OCP community and lead to innovation that will allow for OCP network hardware and software to become a true alternative to traditional network OEMs.

Contents

Figures

Revision History

Name	Date	Version	Description
Glenn Sullivan	2016-06-13	0.1	Initial Release

Overview

FlexSwitch is a modern network protocol suite providing complete layer 2 and layer 3 functionality designed to run on OCP hardware in conjunction with other OCP software offerings – such as ONIE, ONL, and SAI. Written from scratch in Go, FlexSwitch is an alternative to legacy open source routing and switching stacks that require stitching multiple projects together - in order to build a complete stack. Operating in user-space with an underlying NOS, such as Open Network Linux – FlexSwitch provides a control-plane for network protocol features, as well as the ability to program the underlying hardware. With a focus on scale, zero-touch operations, and consumability – FlexSwitch is designed with the core mantra that everything must have an API. The goal of FlexSwitch is to have a highly modular, reusable code-base that runs on any combination of OCP hardware and software – with every element accessible to view or change via a REST API. This modular software architecture is shown in Figure 1.

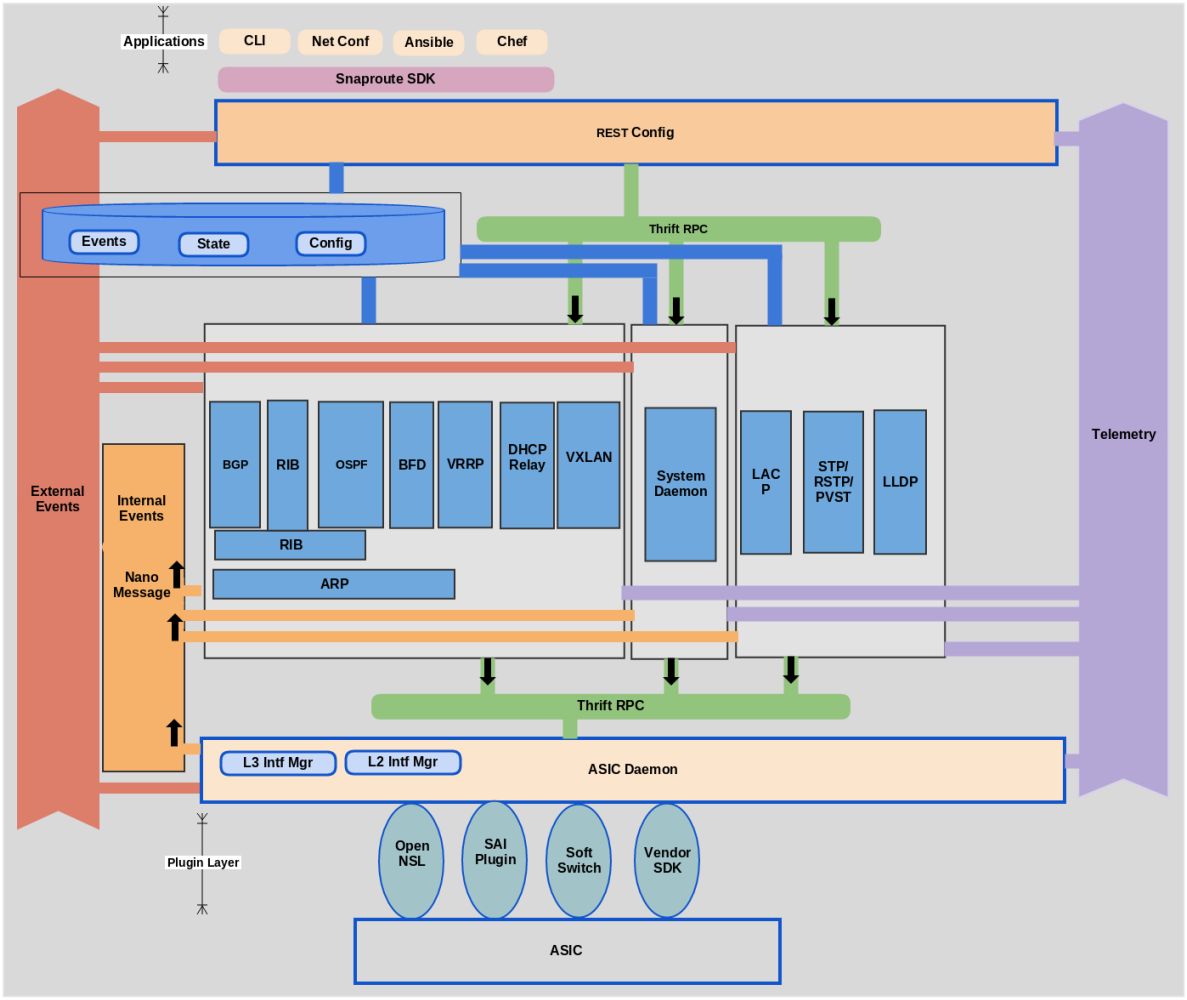

Figure 1: FlexSwitch architecture

License

Background

The current software offerings from the Open Compute Project (OCP) serve as an excellent platform for driving adoption of OCP hardware and have laid the groundwork for upper-layer protocol stacks to provide an alternative to the traditional network OEM offerings. The Open Network Install Environment (ONIE) is a robust boot-loader that has seen widespread adoption by all participating OCP switch vendors. Open Network Linux (ONL) has become the reference standard NOS with a Hardware Compatibility List that encompasses a wide range of devices. Finally, the Switch Abstraction Interface (SAI) has become an industry-backed hardware abstraction layer (HAL) for programming merchant silicon.

The missing piece, preventing an operator from deploying a complete OCP hardware and software solution, is a network protocol stack that implements industry standard routing and switching protocols to program underlying packet-forwarding hardware. Such a network stack needs to be offered as part of the OCP software suite and written with both developers and operators in mind allowing it to appeal to as many OCP adopters as possible.

To this end, an OCP network stack must:

- Reside in user-space without reliance on specific Linux kernels or patches
- Be comprised of new code implementations of standards-based features – without integrating legacy open source protocol stacks
- Interop appropriately with traditional OEM protocols and hardware – allowing for deployment in mixed environments
- Demonstrate agnostic principles by working on as many different hardware and software platforms as possible
- Focus on DevOps with a Golang code-base and Python SDK
- Be designed with a modular component base allowing for easy swap out of key components (i.e. database, RPC/IPC, etc.)
- Have an API for every configuration and state condition – eliminating the need for "screen scraping" information from a network device
- Have the ability to be packaged in a flexible manner, allowing for the user to only deploy the components and features that are required

Design

FlexSwitch is a suite of layer 2 and layer 3 network protocol user-space applications, providing for a full control-plane which programs the underlying software and hardware packet forwarding infrastructure. FlexSwitch is designed to function entirely in user-space, allowing for a highly-portable code-base that can be utilized across a multitude of platforms. Lending to this portability, FlexSwitch complements other OCP projects, such as Open Network Linux (ONL) and Switch Abstraction Interface (SAI) – as ONL serves as the base Network Operating System

(NOS) and SAI is used as a means of programming switching silicon.

FlexSwitch is comprised of feature components run as independent daemons, acting as individual micro servers. Each daemon provides APIs using RPC mechanisms – Apache Thrift is leveraged for this interaction. Configuration, state, and event information is stored in a central Redis database. Events notification between components is handled by Nanomsg. These infrastructure components for RPC, database, and messaging are implemented in a way to allow for easy replacement – should needs change.

Here is an overview of major FlexSwitch components:

Configuration Manager

Initialized as "ConfigD" – the configuration manager serves as the ingress point for all interactions with the rest of the system. ConfigD provides REST interfaces to a pool of objects, which are defined in the Model Repository. All REST API requests and responses for ConfigD are performed via HTTP. User authentication is achieved using a proxy, such as NGINX or Apache.

System Daemon

SysD is responsible for monitoring the overall health of the system as well as the health of individual protocol and infrastructure daemons. Utilizing ongoing keep-alive mechanisms SysD can determine a hung or crashed process and initiate a control restart and recovery of a protocol or feature. Additionally, SysD also controls global configuration parameters such as System Name, Router ID, Management IP, etc.

Routing Information Base

All IPv4 and IPv6 routes are stored and processed by RibD, resulting in a centralized point for the processing of all routing information. This centralized approach alleviates the need for routing protocol daemons to interact directly with each other – allowing for both protocol-dependent and generic policies to be applied in a holistic fashion. The policy-based functionality is a core component of RibD and serves as the mechanism to filter, redistribute, or manipulate routes.

ASIC Daemon

AsicD is a functional abstraction layer between the underlying hardware and the rest of the FlexSwitch stack. The core functionality of AsicD is to provide a common northbound API interface to all protocol daemons. This allows for a high level of reusability of configurations across all hardware platforms. Currently, AsicD support the use of OpenNSL and direct SDK access for Broadcom platforms – as well as SAI for Mellanox, Barefoot, and Cavium ASICs. In addition to programming hardware, AsicD is also responsible for publishing data to Linux – creating routes, interfaces, and statistics via netlink.

Layer 2 and Layer 3 Protocols

Each protocol acts as an autonomous control-plane with a dedicated daemon for each protocol. This highly segmented approach allows for deployment customization – with only the required features and components running on a system. An added benefit of the “one protocol – one daemon” method is that it greatly limits the impact of issues to within that process or protocol, making it difficult for one protocol to impact another.

FlexSwitch SDK

Along with the ability to configure FlexSwitch via REST APIs, a Python-based SDK is integrated into the code base. Taking the form of a Python module, the FlexSwitch SDK can be imported into any Python script and its methods perform the function of every available API on the target device.

Test Plan

Unit tests are written and submitted as part of the source. Additionally, integration testing is performed against multiple functional code units (ex. BFD interacting with BGP). Finally, a nightly end-to-end functional test is performed against every build running the code on live hardware in a multitude of intricate topologies. This functional test is run across various hardware platforms on different ASICs, different base NOSs, and includes traditional OEM hardware for interop validation.

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 someone with commit rights. The current list of committers/maintainers includes:

- SnapRoute

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 FlexRoute 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: http://www.snaproute.com

Mailing list: opencompute-networking@lists.opencompute.org

IRC: N/A

Mirror: N/A

GitHub: https://github.com/OpenSnapRoute

Wiki: http://www.opencompute.org/wiki/Networking

Roadmap

The initial feature list for FlexSwitch reflects the needs of data center operators with a heavy focus on scalable features. Follow-up releases are targeted for enterprise and service provider needs.

Current features:

- BGP
- BFD
- OSPF
- VRRP
- ECMP
- RIB
- Routing Policy Engine
- DHCP Relay
- ARP
- VLANs
- SVIs
- STP, RSTP, PVST
- BPDU Guard
- Bridge Assurance
- LACP

Future enterprise features:

- IPv6
- MLAG
- Data Plane ACLs
- Port Mirroring
- TACACS
- SNMP
- Control Plane Policing (CoPP)
- sFlow
- QoS Classification and Scheduling
- Data Plane and Management VRFs
- VxLAN
- Gradeful Restart and Hitless Upgrades

Future service provider features:

- LFIB for MPLS
- LDP
- RFC 3107
- VPLS and HVPLS
- EVPN
- VPWS
- L3VPN

Supporting Documents

The majority of the technical documents live in the OpenSnapRoute/docs directory in the source on Github. This code is rendered on GitHub Pages and includes:

- Quick Start Guide
- Build Environment Overview
- FlexSwitch Architecture
- Binary Package Usage
- Configuration Examples
- API Reference