Leveraging SONiC Functionalities in Disaggregated Network Switches

Ali AlSabeh*, Elie Kfoury*, Jorge Crichigno*, Elias Bou-Harb†
* Integrated Information Technology Dept., University of South Carolina (USC), Columbia, South Carolina, USA
†The Cyber Center For Security and Analytics, Information Systems and Cyber Security Dept.
University of Texas at San Antonio (UTSA), San Antonio, Texas, USA
Email: *aalsabeh@email.sc.edu, *ekfoury@email.sc.edu, *jcrichigno@cec.sc.edu, †elias.bouharb@utsa.edu

***Abstract*—Ever since the inception of the networking industry, routing and switching devices have been limited to tightly-coupled hardware and software components. Vendors provide closed source proprietary stacks, restraining network operators from utilizing customized features, and hence hindering innovation. This aggregated model is costly, time consuming, and unscalable as changes in the devices require vendor's intervention. As a result, the industry started manufacturing white-box switches and developing Network Operating Systems (NOSs) that support multiple vendors and Application Specific Integrated Circuits (ASICs). This model is referred to as "disaggregated" as the software and hardware are decoupled; essentially, vendors' switching silicons (e.g., Broadcom) are compatible with different NOS (e.g., SONiC). In this paper, we discuss the lessons learned while designing and implementing a testbed that consists of disaggregated network devices. We iterate over several open source Internet Protocol (IP) routing suites and NOSs that are vendor-agnostic. Additionally, we highlight a novel type of forwarding data planes that are programmable and explore their features. The testbed consists of two white-box switches provided by Edgecore that use programmable switching silicon (Tofino) manufactured by Barefoot Networks, an Intel Company. We installed SONiC NOS on top of the switches and tested static and BGP routing protocols. We report the configuration process and the prerequisites needed to deploy a working disaggregated environment. Finally, we discuss how open source NOSs and programmable switches can be extended to support campus networks, rather than being data center-oriented only.**

***Keywords*—Network operating system, SONiC, white-box switches, programmable ASICs, routing protocols.**

I. Introduction

Network switches were introduced as hardware devices that connect multiple computers in a Local Area Network (LAN) and forward frames using the physical address. The network layer operates on top of the data-link layer and connects computers over the Internet. Typically, routers perform network layer functionalities, where they run routing protocols such as Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) [1].

The functionality of conventional switches has been limited to data plane forwarding since they run closed source, proprietary Network Operating Systems (NOSs), which are conceptualized, designed, developed, and sold by a specific company. The vendor provides the locked-in hardware with a pre-installed NOS, preventing the user from tampering it or installing third-party software. This behavior is beneficial among traditional networks where vendors have extensively tested their software before distributing it among clients. However, when it comes to adopting new technologies and scaling the network, vendors become cautious and reluctant due to security concerns, financial costs, and downtime drawbacks that might follow [2].

Bare-metal/white-box switches provide network engineers the capability of installing an operating system independent from the hardware. Thus, they give users the flexibility to customize switches based on their needs. Customization has attracted the interest of network engineers and large manufacturers such as Dell and HP [3]. As a result, open source NOSs are being developed and their popularity is increasing due to their role in leveraging switches functionalities. However, they still need extensive experimentation and documentation, especially if companies are aiming to replace their current closed source networks by open source technologies.

In this paper, we set up a testbed consisting of two programmable switches with SONiC, a pioneering open source NOS that supports a full-suite of network functionality [4]. We perform experiments in the context of programmable white-box switches and open source NOS. The main goal is to build a testbed in a lab environment prior to incrementally deploy this technology in the campus enterprise network and Science Demilitarized Zone (DMZ) [5] at the University of South Carolina (UofSC). This effort is aligned with the development of custom protocols to enhance TCP performance [6] and relay services for media traffic [7] on campus. The rest of the paper is organized as follows. In Section 2, we present background information about open source routing stacks and NOSs, specifically Free Range Routing (FRR) [8] and SONiC [4]. In addition, we introduce programmable switches and present a number of powerful features they possess. In Section 3, we demonstrate the preparatory phase to deploy our testbed and the experiments that were performed. In Section 4, we discuss how a campus network can benefit from open source NOSs and programmable white-box switches. We conclude this work in Section 5 and provide a road map for future work.

This work was supported by the U.S. National Science Foundation (NSF), Office of Advanced Cyberinfrastructure (OAC), award #1925484.

II. Background

A. Open source Internet routing protocol suites

FRR [8] is an IP routing protocol suite for Linux and Unix platforms that implements protocol daemons for BGP, OSPF, and other protocols. In addition to supporting the full range of L3 configuration, FRR supports a number of L2 functionalities such as Layer Distribution Protocol (LDP). Each protocol in FRR is implemented as a daemon, and all the daemons communicate with an orchestration daemon called zebra, which coordinates routing decisions and communicates with the data plane. The modular architecture of FRR and the separation of daemons make it highly resilient, flexible, and extensible.

Other open source software solutions that implement Internet routing protocols are listed as follows. BIRD [9] supports multiple routing protocols, such as BGP and OSPF. It is tested on Linux and ported to FreeBSD, NetBSD, and OpenBSD. OpenBGPD [10] fairly provides a complete BGP implementation and it is regularly tested on OpenBSD, Linux, and FreeBSD. eXtensible Open Router Platform (XORP) [11] supports a wide variety of routing protocols such as Internet Group Management Protocol (IGMP), OSPF, BGP. Mac OS X, Linux, and Windows are among the platforms supported by XORP.

B. Open source NOSs

Disagreggating the hardware from the software in white-box switches has pushed open source NOS to be developed and maintained constantly. SONiC [4] is a Linux-based open source NOS developed by Microsoft. It offers a full-suite of network functionality, like BGP and Remote Direct Memory Access (RDMA) and runs on switches from multiple vendors and ASICs. SONiC consists of several modules that exist either in docker containers or in the Linux-host system itself. A container is a lightweight, standalone, executable package of software that contains the code, runtime, system tools, system libraries, and settings needed to execute the application [12]. The high-level architecture of SONiC is shown in Figure 1, where it operates in the user space. Each component in SONiC handles a specific job, such as relaying the DHCP requests, handling Link Layer Discovery Protocol (LLDP) functionalities, providing Command Line Interface (CLI) and system configuration capabilities, and running FRR or Quagga routing stacks.

Facebook has developed its own NOS for its data centers called FBOSS [13]. Contrary to SONiC, FBOSS is not a separate Linux distribution, and its design is specific to Facebook's data center infrastructure. Thus, FBOSS may not generalize to any data center. Open Network Linux (ONL) [14] is a Linux distribution for bare-metal switches that was developed by big switch networks. Additionally, they incorporated SONiC with ONL to create an open source NOS stack for collaborative development in networking. Figure 2 summarizes the high-level design of the aforementioned NOSs and compares them to legacy switches.

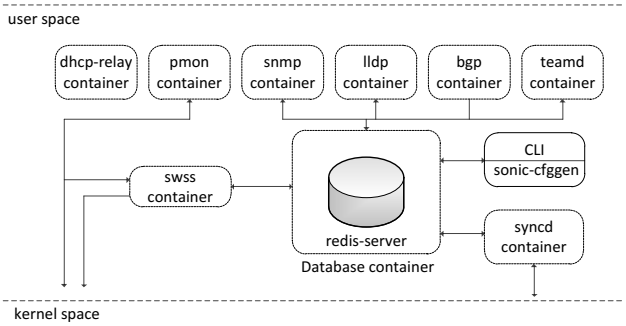

Fig. 1. High-level architecture of SONiC.

C. Enterprise Networks Migrating to SONiC

Apart from Microsoft, which has acquired SONiC since 2016 and outsourced it to the community, Linkedin is working on OpenFabric [15], a web-scale protocol for data center fabric. Alibaba [16], a multinational Chinese company specializing in e-commerce, retail, Internet, and technology, has deployed SONiC as its NOS and become a major contributor in the open source community of SONiC. Alibaba added features like Virtual Local Area Network (VLAN) Trunk, Terminal Access Controller Access Control System Plus (TACACS), Link Aggregation Control Protocol (LACP) fallback, and Maximum Transmission Unit (MTU) setting. Other data center enterprises with large-scale networks such as Tencent and Baidu are providing resources to test, build, and deploy SONiC in their infrastructure [17].

D. Programmable forwarding plane

Traditionally, switches implement fast path forwarding, i.e., the forwarding of the packets at high speeds, using switching ASICs that contain one or more forwarding tables. These ASICs have enabled the transfer of terabits of aggregate traffic every second. However, they are designed with fixed functionalities that allow a limited number of network protocols. As a result, in order to adopt new protocols, the emergence of new ASIC generation is needed, such that the performance capabilities are not sacrificed to keep up with the ever growing network services. Barefoot networks countered this problem by introducing Tofino, the world's

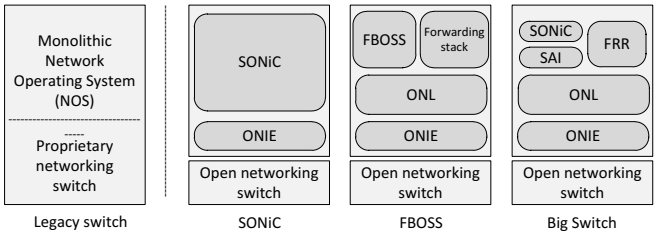

Fig. 2. Open and closed source NOS design.

first end-user fully-programmable Ethernet switch. The Tofino chip is loaded with a P4 program that provides the logic for handling all supported protocols. Thus, making the Tofino switch protocol independent, and whenever a new protocol is required, the only entity that needs to be updated is the P4 program.
The programmable switches support a set of powerful features that allow network operators to drive and control their network with more freedom. Among these features is enabling network administrators to have more visibility by exporting metadata through various channels. For example, using In-band Network Telemetry (INT) to query switch-internal state, such as queue size, link utilization, and queuing latency [18]. A noteworthy breakthrough is that the additional features of the programmable switch do not come on the expense of performance, nor on the power consumption and price [19].

III. Experimentation

In this section, we discuss the preparatory stage to acquire the switches. Then, we present our testbed and the steps followed to build the environment. Furthermore, we test the routing protocols that are supported by SONiC while providing technical details regarding the configuration done.

A. Preparatory Phase

Before acquiring any networking devices, the specifications of the device and its role in the network should be clear. We adopt programmable switches due to their power in reshaping the network.
Programmable switches are gaining more attraction, and vendors are competing in this field. We chose Barefoot Networks, an Intel Company, that are the initiators of programmable ASIC switches. Barefoot manufactures these programmable circuits, and they are now supported by multiple vendors, such as Edgecore, Inventec, Stordis, and WNC. Our white-box switches are manufactured by Edgecore, and they are equipped with Barefoot Tofino 3.3T ASIC. To acquire these switches, a Non-disclosure agreement (NDA) had to be signed.
Initially, these switches are loaded with Open Network Install Environment (ONIE), which defines an open install environment and enables an open networking hardware ecosystem where users can choose and deploy a variety of NOSs [20].

B. Building the Testbed

Figure 3 shows our testbed which consists of two Wedge 100BF-32X switches. On top of ONIE, we installed the required files to build the Software Development Environment (SDE) 9.1 and the P4 profiles. Barefoot's SDE was also used to build the baseline `switch.p4` program on the device. This program implements various networking features needed for typical cloud data centers, including Layer 2/3 functionalities, Access Control List (ACL), Quality of Service (QoS), etc. After building the SDE, SONiC was installed and the two switches were connected through a fiber optic cable of 100 GB bandwidth. SONiC is built on the Switch Abstraction Interface

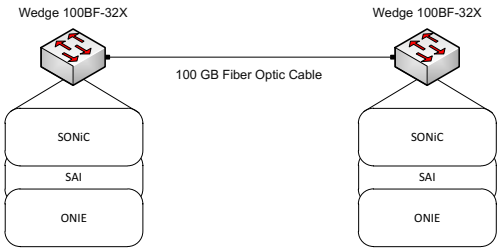

Fig. 3. Our testbed consisting of two Wedge 100BF-32X switches through 100 GB fiber optic cable.

(SAI), which defines the API to provide a vendor-independent way of controlling forwarding elements, such as a switching ASIC [21].

C. SONiC supported IP routing protocols

Figure 4 shows our topology that consists of two switches directly connected through a fiber optic cable on interface `Ethernet124` of both switches. Additionally, both switches have their interface `Ethernet120` up and configured with an IP address that will be advertised to the switch on the other end. The interfaces `Ethernet120` and `Ethernet124` have logical numbers, and they map to the physical interfaces `Ethernet31` and `Ethernet32`, respectively.
The two routing protocols that are supported by SONiC are static and BGP routing protocols. For BGP configuration, switch 1 and switch 2 are in autonomous systems 65000 and 65100, respectively.

SONiC manages its configuration using a single source of truth, referred to as ConfigDB. This file is located in `/etc/sonic/config_db.json` and it consists of multiple tables that SONiC uses in its configuration. For example, the table `INTERFACE` contains the IP addresses of the interfaces. To change the IP addresses, we modified the `INTERFACE` table manually and configured the IP addresses of interfaces Ethernet120 and Ethernet124 according to our architecture. Figure 5 shows the modified tables in ConfigDB file. For the modification to take effect, the configuration must be loaded using the command `sudo config load -y`, and the switch must be rebooted using the `reboot` command. We were able to configure and run static routing successfully using the command line interface `vtysh` that provides a combined frontend to all FRR daemons in a single combined session [8].
For BGP, SONiC uses the loopback address configured in the `config_db.json` file as the router ID in BGP sessions. Thus, we changed the IP address of the loopback interface on both switches so that the routers do not have the same ID in a BGP session. The configuration of the loopback interface lies in `LOOPBACK_INTERFACE` table. BGP was also configured using the command line interface `vtysh`, and the ping was successful between the the advertised networks.

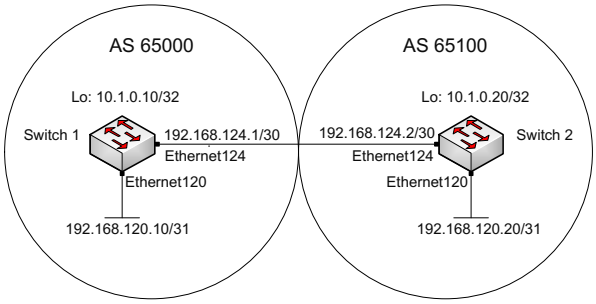

Fig. 4. Our network topology to test SONiC supported routing protocols.

Switch 1 config_db.json file	Switch 2 config_db.json file

"INTERFACE": {
"Ethernet120|192.168.120.10/31": {},
"Ethernet124|192.168.124.1/30": {},
},
"LOOPBACK_INTERFACE": {
"Loopback0|10.1.0.10/32": {},
},
	"INTERFACE": {
	"Ethernet120|192.168.120.20/31": {},
	"Ethernet124|192.168.124.2/30": {},
	},
	"LOOPBACK_INTERFACE": {
	"Loopback0|10.1.0.20/32": {},
	},

Fig. 5. Modified tables in the configuration file of switches 1 and 2.

IV. Discussion

Although most of the open source NOSs are mainly targeted to data centers, the features that they support allow them to be deployed on campus networks. Campus network contains multiple LANs that reside within a geographic area. It has a hierarchical design that is divided into layers. The access layer performs layer 2 functionalities. On top of it, the distribution layer resides to perform layer 3 functionalities.

In a proprietary closed-source network design, the access layer has limited capabilities, and the best actions that network operators can do is to replace the current layer 2 switches with layer 3 switches. However, layer 3 switches are costly and their features are limited and closed to the vendor [22]. With open source NOS, the aforementioned limitations would not exist, and the access layer capabilities will be boosted based on the requirements of the campus network, without being limited to the vendor.

Furthermore, P4 switches allow the implementation of a new class of high performance data-plane applications, such as INT, layer 4 load balancing, and in-network DDoS detection. Therefore, combining an open-source NOS with powerful hardware is promising for a new era of network designs.

V. Conclusion

In this paper, we surveyed a number of open source networking software systems that are reshaping the network by disaggregating its components and making it vendor-agonistic. The architecture of the open source network, in a bottom-up view, starts from the white-box forwarding hardware, in which the operating system is not limited to the vendor. Instead, network administrators choose which open source NOS to deploy based on the needs of their network. Furthermore, open source IP routing stacks are being integrated with NOSs, thus, transforming a switch from a layer 2 hardware, to layer 3 hardware. In our experiments, we had two programmable switches running SONiC as a NOS, where we tested static and BGP routing protocols and validated their correctness. Although open source NOSs and white-box switches are mainly targeted to large data centers, they have strong potentials to replace closed source proprietary switches used in campus networks. For future work, we plan to deploy our programmable switches that run SONiC in our campus network and closely observe their effects on the network behavior.

References

[1] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*. Pearson, 6 ed., 2017.

[2] X. Yang, Z. Sun, J. Li, J. Yan, T. Li, W. Quan, D. Xu, and G. Antichi, "Fast: enabling fast software/hardware prototype for network experimentation," in *Proceedings of the International Symposium on Quality of Service*, pp. 1–10, 2019.

[3] E. Puijk, "Open-source network operating systems: feature evaluation of sonic." https://esc.fnwi.uva.nl/thesis/centraal/files/f1729148638.pdf. Accessed: 2020-01-03.

[4] "Software for open networking in the cloud - sonic." https://azure.github.io/SONiC/. Accessed: 2019-11-15.

[5] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on science dmz," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 2041–2078, 2019.

[6] E. Kfoury, J. Crichigno, E. Bou-Harb, D. Khoury, and G. Srivastava, "Enabling tcp pacing using programmable data plane switches," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 273–277, IEEE, 2019.

[7] E. Kfoury, J. Crichigno, and E. Bou-Harb, "Offloading media traffic to programmable data plane switches," in *IEEE International Conference on Communications (ICC)*, IEEE, June 2020.

[8] "Free range routing - frr." https://frrouting.org/. Accessed: 2020-01-10.

[9] "Bird internet routing daemon." https://bird.network.cz/?get$_docv$ $=$ $20f = bird - 1.htmlss1.1.Accessed$: $2020 - 02 - 05$.

[10] "Openbgpd." http://www.openbgpd.org/. Accessed: 2020-02-05.

[11] "extensible open router platform." http://www.xorp.org/. Accessed: 2020-02-05.

[12] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[13] S. Choi, B. Burkov, A. Eckert, T. Fang, S. Kazemkhani, R. Sherwood, Y. Zhang, and H. Zeng, "Fboss: building switch software at scale," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 342–356, 2018.

[14] "Big switch networks." https://www.bigswitch.com/. Accessed: 2020-02-06.

[15] R. White, S. Hedge, and S. Zandi, "Is-is optimal distributed flooding for dense topologies draft-white-distoptflood-03," April, 2020.

[16] H. Wang, "Sonic development and deployment at alibaba." https://www.opencompute.org/files/Alibaba-NOS-Development.pdf. Accessed: 2020-01-17.

[17] A. Raveh, "Sonic is making open ethernet a dream come true." https://blog.mellanox.com/2018/10/sonic-open-ethernet-dream-come-true/. Accessed: 2020-01-15.

[18] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, 2015.

[19] Barefoot Networks, an Intel company, "World's fastest p4-programmable ethernet switch asics." https://www.barefootnetworks.com/products/brief-tofino/, 2020.

[20] "Open network install environment (onie)." https://opencomputeproject.github.io/onie/. Accessed: 2019-11-15.

[21] "Open network install environment (onie)." https://github.com/opencomputeproject/SAI. Accessed: 2019-11-20.

[22] "Campus lan and wireless lan design guide." https://www.cisco.com/c/dam/en/us/td/docs/solutions/CVD/Campus/CVD-Campus-LAN-WLAN-Design-Guide-2018JAN.pdf. Accessed: 2020-02-08.