Deployment of SCION on Bare Metal Switches

Lars-Christian Schulz Cecil Benjamin Leonard

March 30, 2020

Introduction

- SCION is an internet architecture that is capable of routing control, failure isolation, and explicit trust information for end-to-end communication
- Bare-metal switch is a network device without network operating system.We can decide on an operating system.It is cheaper and features based on operating system can be customized when in comparison to a closed switches.
- We aim to implement the SCION on an Edgecore Network's bare metal switches

Outline

Introduction

Switch Hardware

Choosing a NOS

Running SCION on Open Network Linux

Switch Fabric APIs

Summary and Future Work

Switch Hardware

- SCION is deployed on two Edgecore switches an AS4610-54T and an AS5812-54T.
- Switch AS4610-54T utilizes a Broadcom BCM56340 Helix4 switching ASIC and a dual-core ARM Cortex A9 at 1 GHz with 2 GB of DDR3 SDRAM.
- Switch AS5812-54T has a Intel Atom C2538 quad-core 2.4 GHz x86 processor. Broadcom BCM56864 Trident2 switching ASIC.

Outline

Introduction

Switch Hardware

Choosing a NOS

Running SCION on Open Network Linux

Switch Fabric APIs

Summary and Future Work

Choosing a Network Operating System

- SCION can run on multiple platforms such as Ubuntu, ARM mini computers, Android devices.
- Network Operating Systems - PICA OS, Cumulus OS. ONL, OpenSwitch (OPX)
- We chose Open Network Linux because it is an open source operating system
- It is supported and can be implemented on most of the available processors.
- We are using Debian9 based armel as the operating system in AS4610-54T
- Debian9 AMD64 is used as an operating system in AS5812-54T.

Outline

Introduction

Switch Hardware

Choosing a NOS

Running SCION on Open Network Linux

Switch Fabric APIs

Summary and Future Work

Running SCION on Open Network Linux

ARM

- SCION and its dependencies are compatible with Ubuntu environment and ONL is Debian based operating system.
- Installation of SCION in Debian 8 based armhf ONL
  - Python has to be upgraded. As the default version is too old.
  - Capnproto and libcapnp-dev packages are unavailable in Debian 8 hence installed it from Jessie-backports
- Installation of SCION in Debian9 based armhf ONL
  - Error when installing cryptography. So upgraded the version of cryptography
  - Error with ZLOG files.
  - Setcap command failed because of kernel configuration.
  - SCION was unable to find if docker is running or not.

Running SCION on Open Network Linux

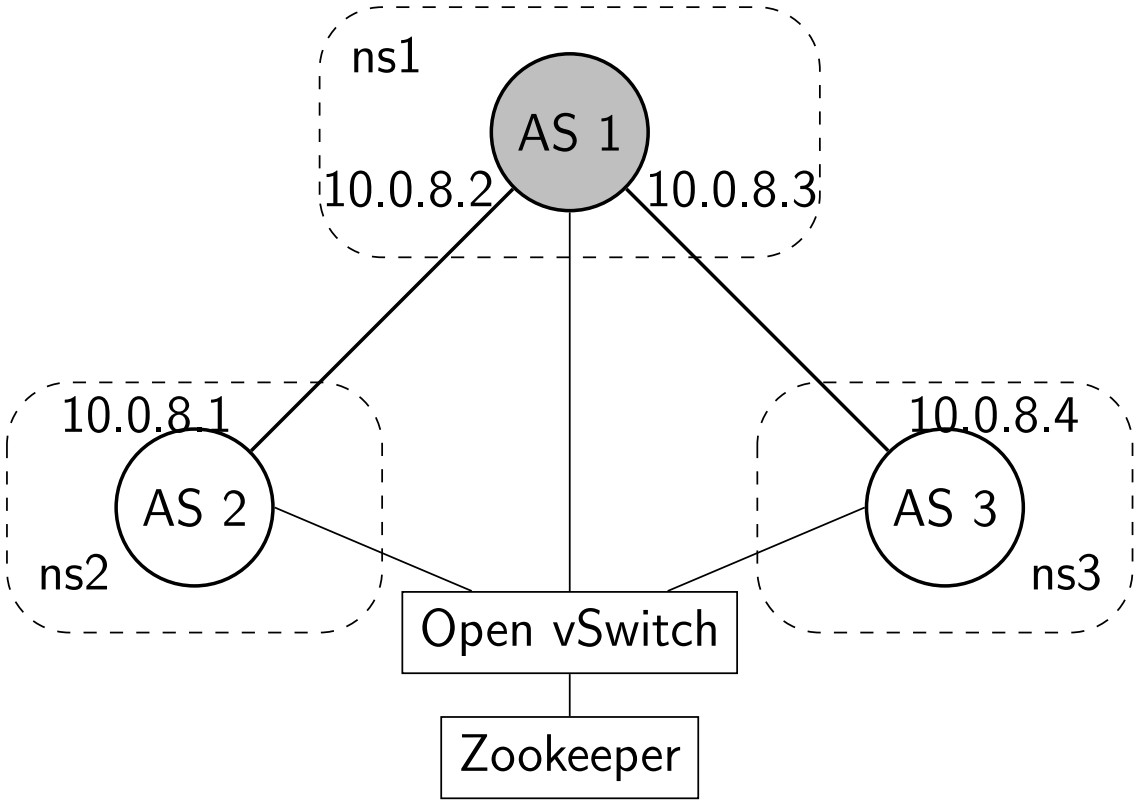
x86

- Installation of SCION in Debian9 based x86 ONL
  - Issue with Cryptography installation. Cryptography has a dependence on pyopenssl. When cryptography was upgraded and pyopenssl was also upgraded. Pip3 stopped working so pyopenssl was reinstalled with lower version.
  - Setcap command failed so installed libcap2-bin.
  - Bandwidth test and sensor fetch were tested.
  - SCION is moved to a USB drive because SCION logs files could cause flash memory full in the switch. Better solution would be to use an network file storage system.

Running SCION on Open Network Linux

SCION traffic on a physical loop connection

- Map SCION UDP overlay connections to switch ports



- Isolate ASes in network namespaces to force packets out the physical network interfaces
- Open vSwitch connections for infrastructure management

Outline

Introduction

Switch Hardware

Choosing a NOS

Running SCION on Open Network Linux

Switch Fabric APIs

Summary and Future Work

Open Switch APIs (Broadcom)

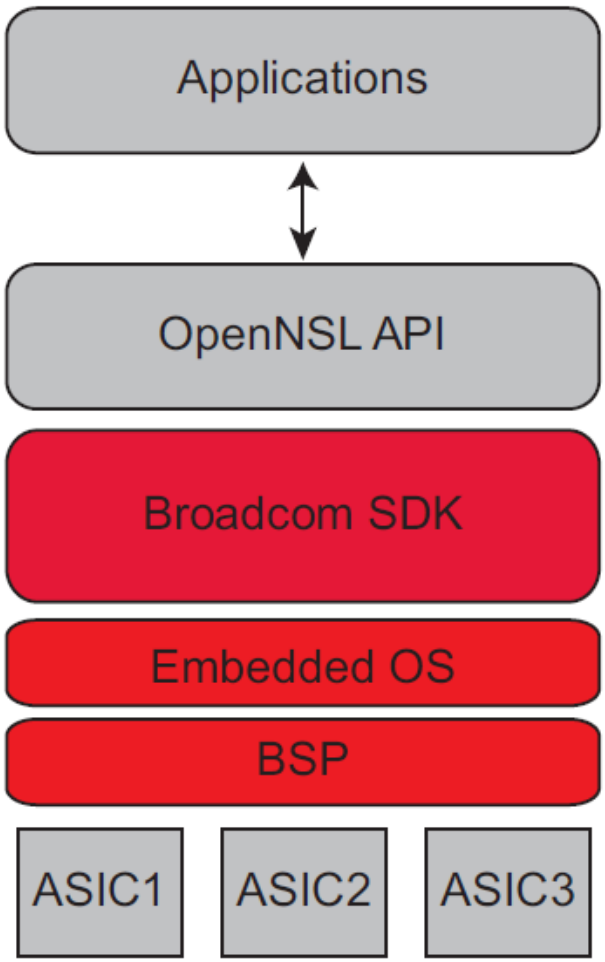
- At this point the switches are just slow general-purpose computers running Linux
- Need a way to control and configure the switching chip

- Public APIs available from Broadcom:
  - OpenNSL (Open Network Switch Layer)
  - OF-DPA (OpenFlow Data Plane Abstraction)
  - SAI (Switch Abstraction Interface)
  - SDKLT (Logical Table Software Development Kit)

OpenNSL

Overview

- Abstracts a wide range of Broadcom switch ASICs
- General switch operation, configuring and monitoring ports
- Warm boot (restart without disrupting the data plane)
- Control L2 switching and L3 routing tables
- VLANs, QoS, L2 multicast, trunking, tunneling, ...



Source: OpenNSL White Paper (June 2016)

OpenNSL

Overview

- Packet Tx/Rx API
  - Bypass the Linux network stack $\rightarrow$Could improve border router speed

OpenNSL

Overview

- Packet Tx/Rx API
  - Bypass the Linux network stack →Could improve border router speed
- Kernel Network (KNET)
  - Creates virtual network interfaces connecting switch fabric to the CPU
  - Switch ports become accessible from Linux
  - Can match arbitrary packet data →Could match hop fields in the SCION header

OpenNSL

Overview

- Packet Tx/Rx API
  - Bypass the Linux network stack →Could improve border router speed
- Kernel Network (KNET)
  - Creates virtual network interfaces connecting switch fabric to the CPU
  - Switch ports become accessible from Linux
  - Can match arbitrary packet data →Could match hop fields in the SCION header
- Field Processor API
  - Match header fields, classify packets and take according actions
  - Might be able to match arbitrary user-defined header fields

OpenNSL

Overview

- C interface and usage examples published on GitHub[1]
  - "Community Development Package"
- Implementation is closed source and binaries are provided by switch vendors
  - "OEM/ODM Development Package" available under a license agreement
  - Some platform customization through configuration scripts without recompiling
- OpenNSL exposes a subset of the full Broadcom SDK

[1] `https://github.com/Broadcom-Switch/OpenNSL`

OpenNSL

Compiling the Examples

- Binaries supplied by Edgecore
- Example source code from GitHub supplied by Broadcom

OpenNSL

Compiling the Examples

- Binaries supplied by Edgecore
- Example source code from GitHub supplied by Broadcom
- x86-64: compiles and works

OpenNSL

Compiling the Examples

- Binaries supplied by Edgecore
- Example source code from GitHub supplied by Broadcom
- x86-64: compiles and works
- ARM: missing symbols

```
bin/as4610/libopennsl.so.1: undefined reference to 'bcm_ether_atoe'
bin/as4610/libopennsl.so.1: undefined reference to 'bcm_ether_ntoa'
```

- Slight version mismatch between examples source code (3.5.0.1) and OpenNSL binary (3.5.0.3)
- bcm_ether_atoe and bcm_ether_ntoa are utility functions used in some Broadcom code
- Can link them in from another source

OpenNSL

Compiling the Kernel Modules

- OpenNSL relies on three modules loaded in the Linux kernel: `linux-kernel-bde`, `linux-user-bde` and `linux-bcm-knet`
- Edgecore provides OpenNSL only for ONL based on the armel port of Debian
- User space library for armel can run on armhf too
- Kernel modules do not work with a kernel they have not been compiled for
- GPL-licensed kernel module source code is available

OpenNSL

Compiling the Kernel Modules

- Makefiles supplied with OpenNSL seem quite outdated
- Compiling for an x86-64 architecture and Linux 4.4 (Ubuntu 16.04) worked, but we are running Kernel 4.14 on ARM (armhf)

OpenNSL

Compiling the Kernel Modules

- Makefiles supplied with OpenNSL seem quite outdated
- Compiling for an x86-64 architecture and Linux 4.4 (Ubuntu 16.04) worked, but we are running Kernel 4.14 on ARM (armhf)
- Combined existing makefiles for different kernel versions and architectures
- No success yet, because of removed/changed kernel APIs

OpenNSL

Compiling the Kernel Modules

- Makefiles supplied with OpenNSL seem quite outdated
- Compiling for an x86-64 architecture and Linux 4.4 (Ubuntu 16.04) worked, but we are running Kernel 4.14 on ARM (armhf)
- Combined existing makefiles for different kernel versions and architectures
- No success yet, because of removed/changed kernel APIs
- Updated code for Kernel 4.14 $\rightarrow$Modules compile successfully

OpenNSL

Compiling the Kernel Modules

- Makefiles supplied with OpenNSL seem quite outdated
- Compiling for an x86-64 architecture and Linux 4.4 (Ubuntu 16.04) worked, but we are running Kernel 4.14 on ARM (armhf)
- Combined existing makefiles for different kernel versions and architectures
- No success yet, because of removed/changed kernel APIs
- Updated code for Kernel 4.14 $\rightarrow$Modules compile successfully
- Loading the modules still fails, not enough memory for DMA
- Lowering amount of memory allocated for DMA from 4 MB to 2 MB works

OpenNSL

Compiling the Kernel Modules

- Makefiles supplied with OpenNSL seem quite outdated
- Compiling for an x86-64 architecture and Linux 4.4 (Ubuntu 16.04) worked, but we are running Kernel 4.14 on ARM (armhf)
- Combined existing makefiles for different kernel versions and architectures
- No success yet, because of removed/changed kernel APIs
- Updated code for Kernel 4.14 $\rightarrow$Modules compile successfully
- Loading the modules still fails, not enough memory for DMA
- Lowering amount of memory allocated for DMA from 4 MB to 2 MB works
- Result: `linux-kernel-bde` fails to detect the switch hardware

OpenNSL

Summary

- OpenNSL working on both switches
- Can compile examples and our own programs on both switches
  - Send and receive packets, configure VLANs, get statistics
  - Create KNET interfaces and filters

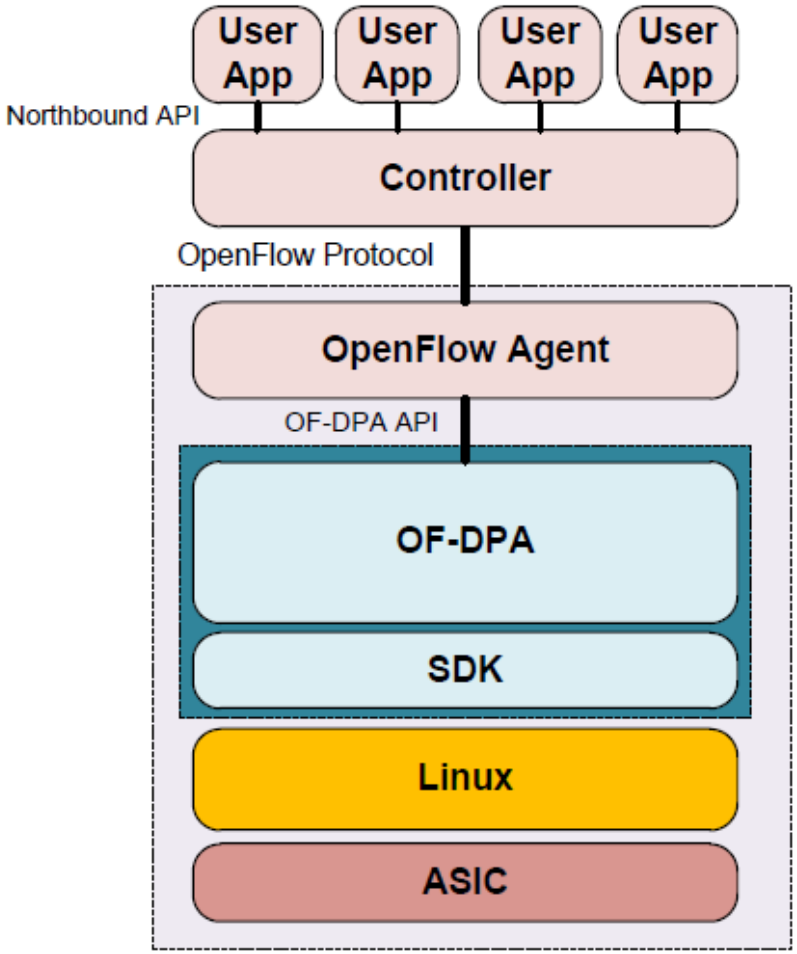
```
>add netif 1 port1
Created interface port1 ID: 1
Created filter port1 ID: 2
>ls netif
ID: 1, Port: 1, VLAN: 0, Name: "port1", MAC: 02:10:18:00:00:01
>ls filter
ID: 2, Priority : 100, Ingress port: 1, DestID: 1, Description : "port1"
ID: 1, Priority : 255, Ingress port: 0, DestID: 0, Description : "
    DefaultRxAPI"
```

  - Use the field processor to classify packets

```
>ls field_group
Field Group: priority : -2147483647 to 2147483647, entries: 1/16384,
    counters: 0/16384, meters: 0/16384
```

OF-DPA

- C API enabling implementation of OpenFlow 1.3.4
- Uses the same software infrastructure as OpenNSL
- Reference implementation of an OpenFlow agent and examples on GitHub[2]



Source: OF-DPA: Abstract Switch Specification Version 2.01

[2] `https://github.com/Broadcom-Switch/of-dpa`

OF-DPA

- Since OpenFlow 1.2: Extensible Match (OXM) format to describe match rules
  - Supports “experimenter” extensions
  - OF-DPA specification describes some experimenter header fields
  - No user-defined header fields

OF-DPA

- Since OpenFlow 1.2: Extensible Match (OXM) format to describe match rules
  - Supports “experimenter” extensions
  - OF-DPA specification describes some experimenter header fields
  - No user-defined header fields
- Test application to send packets between switches

```
>add flow rx 1
>dump table 60 10
Table  60: ACL Policy
Entries: 1 / 3840
Entry 0:  inPort:mask = 1:0 xffffffff  srcMac:mask =
      0000.0000.0000:0000.0000.0000 destMac:mask =
      0000.0000.0000:0000.0000.0000 etherType:mask = 0x0000:0x0000 | GoTo = 0 (
      None) outPort = CONTROLLER (Reserved) | priority = 0 hard_time = 0
      idle_time = 0 cookie = 0
>rx 1
No more packets available
```

SAI and SDKLT

- SAI (Switch Abstraction Interface)
  - Vendor-independent switch API under the umbrella of the Open Compute Project[3]
  - Broadcom's implementation[4] is a wrapper around OpenNSL
- SDKLT (Logical Table Software Development Kit)
  - Similar aim to OpenNSL
  - More direct access to forwarding tables
  - Low-level API is documented
  - Implementation is open source[5]
  - Currently only available for BCM56960 Tomahawk devices

[3] https://github.com/opencomputeproject/SAI
[4] https://github.com/Broadcom-Switch/SAI
[5] https://github.com/Broadcom-Network-Switching-Software/SDKLT

Outline

Introduction

Switch Hardware

Choosing a NOS

Running SCION on Open Network Linux

Switch Fabric APIs

Summary and Future Work

Summary and Future Work

- Selected a suitable OS for the bare-metal switches
- Installed SCION on Debian 8 (Jessie), Debian 9 (Stretch)
- SCION running on a bare-metal switch
- Evaluated switch fabric APIs for use in SCION border router
  - OpenNSL might be able to help, but not sure yet