Overview of Software Defined Networking

Ronald van der Pol <Ronald.vanderPol@SURFnet.nl>
Network Management Automation

SDN approaches
- Underlay/Overlay networking
- Logically centralized control of distributed control plane protocols
- Logically centralized control of programmable dataplanes

Programmable Dataplanes
- Motivation
- OpenFlow
- OpenFlow/SDN example: Google inter-DC network
- P4
- Open Compute Project

Summary
How do we manage networks today?

Configuration of each individual network nodes via the CLI

Error prone
• Manual work
• Need to get every detail correct
• Each protocol has many parameters
• Often similar configuration must be configured on several nodes
It is better to automate this
• Use templates
• Use puppet, chef, ansible, etc
• Preferably with transactions and rollback support

Automation is common aspect of all SDN solutions.

But, not all automation is SDN
• Automating some parts of the configuration is not SDN
• SDN automates a complete end-to-end service
SDN Approaches

Underlay/Overlay networking

Logically centralised control of distributed control plane protocols

Logically centralised control of programmable dataplanes
Underlay/Overlay Networking
Used in data centres, but also in WAN

VXLAN often used as tunneling protocol

Open vSwitch and hypervisors at the edges of the tunnels

Virtualisation and multi-tenancy support

Controller often uses vendor proprietary protocols and APIs

Examples
  • Dell (VMware/Nicira) NSX
  • Nuage Networks (Nokia)
Logically Centralised Control of Distributed Control Protocols

Overview of SDN, Amsterdam, 29 April 2016
Logically Centralised Control of Programmable Dataplanes

Programmable OpenFlow/P4 Switches

OpenFlow
Programmable Dataplanes

Motivation
• disaggregation

Examples
• OpenFlow
• Example: Google
• P4
• Open Compute Project
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

Specialized Applications
Specialized Operating System
Specialized Hardware

App
--- Open Interface ---
Windows (OS) or Linux or Mac OS
--- Open Interface ---
Microprocessor

(slid by Nick McKeown, Stanford University)
Vertically integrated
Closed, proprietary
Slow innovation

Vertically integrated
Closed, proprietary
Slow innovation

Horizontally integrated
Open interfaces
Rapid innovation
Computing vs Networking

Closed Systems

Closed hardware
Workstations + UNIX
UNIX System Call API
Start of Open Source Software
Portable applications

Open Hardware
Open Software

Open Hardware
Open Firmware

Closed hardware
OpenFlow API
Open Source Applications
Portable applications
Disaggregation

From

Closed vendor proprietary routers/switches
• Vendor decides which features to support
• Vendor decides when these feature become available
• Dependent on innovation strength of vendor

To

Split between hardware and firmware (disaggregation)
• Choose best vendor for hardware and best vendor for firmware
• Use open source firmware (more control over features and innovation)
Current way to handle new functionality in networking is to define a new protocol.

Exponential growth in network protocol standards.

Standards seem to become larger and more complex.

Vendors implement all standards, which increases costs and decreases stability.

Do you need all those standards?
Total Number of RFCs Published

Data by Jari Arkko

Overview of SDN, Amsterdam, 29 April 2016
Simple VLAN standard?

Not really, original version amended by at least 14 additional standards.

802.1Q-1998 had 211 pages.

802.1Q-2011 has 1365 pages, and includes:
802.1u, 802.1v, 802.1s (multiple spanning trees), 802.1ad (provider bridging), 802.1ak (MRP, MVRP, MMRP), 802.1ag (CFM), 802.1ah (PBB), 802.1ap (VLAN bridges MIB), 802.1Qaw, 802.1Qay (PBB-TE), 802.1aj, 802.1Qav, 802.1Qau (congestion management), 802.1Qat (SRP)
OpenFlow is the protocol between controller and switch. Standardised protocol.

Switch forwarding tables can be populated directly via OpenFlow.

Many commercial available OpenFlow switches available. Traditional Ethernet switches with an OpenFlow API. Dedicated OpenFlow switches.

Several open source OpenFlow controllers available. (OpenDaylight, ONOS, Ryu, NOX/POX, …)
OpenFlow Components

- OpenFlow Application
- OpenFlow Controller
- OpenFlow Protocol

Overview of SDN, Amsterdam, 29 April 2016
OpenFlow is standardised by the Open Networking Foundation (ONF).

ONF is a non-profit consortium.

Founded in March 2011 by Deutsche Telecom, Facebook, Google, Microsoft, Verizon and Yahoo!

Most vendors in ICT and networking are members now.

Mission: The Open Networking Foundation (ONF) is a user-driven organization dedicated to the promotion and adoption of Software Defined Networking (SDN) through open standards development.
OpenFlow Protocol

Insert flow forwarding entries in switch forwarding tables.

Send packets to OpenFlow switch data path.

Receive packets from OpenFlow switch data path.

Retrieve data path traffic statistics from OpenFlow switch.

Retrieve flow tables from OpenFlow switch.

Retrieve parameters from OpenFlow switch.
E.g. number and properties of ports.
OpenFlow Components

- OpenFlow Controller
- OpenFlow Protocol
- TCP
- SSL
- TLS
- OpenFlow Channel
- OpenFlow Switch
  - Flow Table
  - Group Table
  - pipeline

Overview of SDN, Amsterdam, 29 April 2016
# Flow Table

<table>
<thead>
<tr>
<th>Matching rule #1</th>
<th>Counter</th>
<th>Action #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching rule #2</td>
<td>Counter</td>
<td>Action #2</td>
</tr>
<tr>
<td>Matching rule #3</td>
<td>Counter</td>
<td>Action #3</td>
</tr>
<tr>
<td>Matching rule #4</td>
<td>Counter</td>
<td>Action #4</td>
</tr>
<tr>
<td>Matching rule #5</td>
<td>Counter</td>
<td>Action #5</td>
</tr>
<tr>
<td>Matching rule #6</td>
<td>Counter</td>
<td>Action #6</td>
</tr>
<tr>
<td>Matching rule #7</td>
<td>Counter</td>
<td>Action #7</td>
</tr>
</tbody>
</table>
Table Pipeline

Overview of SDN, Amsterdam, 29 April 2016
## Header Matching (OF 1.3)

**Input port**
Metadata passed between tables

**Ethernet source/destination address**

**Ethernet type**
- VLAN ID
- VLAN priority
- IP DSCP (6 bits in ToS field)
- IP ECN (2 bits in ToS field)

**IP protocol**

**IPv4/IPv6 source/destination address**

**TCP/UDP/SCTP source/destination port**
- ICMP/ICMPv6 type/code
- ARP opcode
- ARP src/tgt IPv4/hardware address
- IPv6 flow label, extension header
- ND target address
- ND src/tgt link layer address
- MPLS label, traffic class, bottom of stack bit
- PBB I-SID
- Logical port metadata
Actions

Output *port_nr*
Group *group_id*
Drop
Set-Queue *queue_id*
Push-Tag/Pop-Tag *ethertype*
Set-Field *field_type value*
Change-TTL *ttl*
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Group Type</th>
<th>Counters</th>
<th>Action Buckets</th>
</tr>
</thead>
</table>

Overview of SDN, Amsterdam, 29 April 2016
Group Types

**Indirect: Execute the single bucket in this group**
Usage: multiple flow entries can point to this group ID, bucket action can be IP routing next hop

**ALL: Execute all buckets**
Used for multicast and broadcast

**Select: Execute one bucket in the group**
Used for load balancing

**Fast Failover: Execute the first live bucket**
Each action bucket is associated with a port
Flow Insertion

**Proactive**
Flow entries are inserted in the OpenFlow switches before packets arrive.

**Reactive**
Packets arriving at an OpenFlow switch without a matching flow entry are sent to OpenFlow controller. They examined by the controller after which flow entries are inserted in the switches.
**Example of Proactive Flow Entries**

*Forward all packets between port 1 and 2*

```bash
ovs-ofctl add-flow br0 in_port=1,actions=output:2
ovs-ofctl add-flow br0 in_port=2,actions=output:1
```

*Forward all packets between access port 4 and trunk port 6 using VLAN ID 42*

```bash
ovs-ofctl add-flow br0 in_port=4,
actions=push_vlan:0x8100,set_field:42->vlan_vid,output:6

ovs-ofctl add-flow br0 in_port=6,
actions=strip_vlan,output:4
```
Google has two networks:
I-Scale: User facing services (search, YouTube, Gmail, etc), high SLA
G-Scale: Data centre traffic (intra and inter), lower SLA, perfect for OpenFlow testing

OpenFlow introduced in G-Scale network since mid 2010

Experience/benefits of introducing OpenFlow:
Better Traffic Engineering (global view of network)

Centralised Traffic Engineering much faster on a 32 core server (25-50 times as fast) than on slow CPUs inside switches

Software development for a high performance server with modern software tools (debuggers, etc) much easier and faster and produces higher quality software than development for an embedded system (router/switch) with slow CPU and little memory
Built from merchant silicon
  ○ 100s of ports of nonblocking 10GE
OpenFlow support
Open source routing stacks for BGP, ISIS
Does not have all features
  ○ No support for AppleTalk...
Multiple chassis per site
  ○ Fault tolerance
  ○ Scale to multiple Tbps
Google’s OpenFlow Deployment

G-Scale WAN Usage

- Exit testing "opt in" network
- SDN rollout
- SDN fully Deployed
- Central TE Deployed

Overview of SDN, Amsterdam, 29 April 2016
Multiple controllers.
3, 5, 7 with Paxos election system.

The whole network is emulated in a simulator.
New software revisions can be tested in the simulator.
Network events (e.g. link down) are sent to production servers + testbed.
Testing in simulator but with real network events.
Google OpenFlow Architecture

Mixed SDN Deployment

- Ready to introduce new functionality, e.g., TE

Overview of SDN, Amsterdam, 29 April 2016
Almost 100% Link Utilization

Sample Utilization

Overview of SDN, Amsterdam, 29 April 2016
Experience/benefits:

Software development for a high performance server with modern software tools (debuggers, etc) much easier and faster and produces higher quality software than development for an embedded system (router/switch) with slow CPU and little memory.

Centralised Traffic Engineering much faster on a 32 core server (25-50 times as fast).
P4 Language

P4: Programming Protocol-Independent Packet Processors

Domain Specific Language for programmable dataplanes

P4 program → P4 compiler → target code

Target code is loaded on P4 switch
  • Consists of packet parser and lookup tables
P4 Switch

Switch Configuration

Parse Graph → Control Program → Match+Action Table Config

Ingress

INPUT → PARSER → Match Action → Queues and/or Buffers → Match Action → OUTPUT

Egress

Source: The P4 Language Specification Version 1.0.2

Overview of SDN, Amsterdam, 29 April 2016
header_type ethernet_t {
    fields {
        dstAddr : 48;
        srcAddr : 48;
        etherType : 16;
    }
}

header_type ipv4_t {
    fields {
        version : 4;
        ihl : 4;
        diffserv : 8;
        totalLen : 16;
        identification : 16;
        flags : 3;
        fragOffset : 13;
        ttl : 8;
        protocol : 8;
        hdrChecksum : 16;
        srcAddr : 32;
        dstAddr : 32;
    }
}
```
header_type vlan_t {
    fields {
        pcp : 3;
        cfi : 1;
        vid : 12;
        ethertype : 16;
    }
}
```
parser start {
    return parse_ethernet;
}

parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        ETHERTYPE_IPV4 : parse_ipv4;
        default: ingress;
    }
}

parser parse_ipv4 {
    extract(ipv4);
    return ingress;
}

Overview of SDN, Amsterdam, 29 April 2016
table ipv4_fib_lpm {
  reads {
    ingress_metadata.vrf : exact;
    ipv4.dstAddr : lpm;
  }
  actions {
    on_miss;
    fib_hit_nexthop;
  }
  size : IPV4_LPM_TABLE_SIZE;
}

action fib_hit_nexthop(nexthop_index) {
  modify_field(ingress_metadata.nexthop_index, nexthop_index);
  subtract_from_field(ipv4.ttl, 1);
}

control ingress {
  if (valid(ipv4)) {
    apply(port_mapping);
    apply(bd);
    apply(ipv4_fib) {
      on_miss {
        apply(ipv4_fib_lpm);
      }
    }
    apply(nexthop);
  }
}
Table Types

**Exact:** value == table entry
E.g. IPv4 host route

**Ternary:** value AND mask == table entry
Wildcard

**LPM: Longest Prefix Match**
Special case of ternary (1111....11110000.....0000)

**Range:** low entry <= value <= high entry

**Valid:** table entry = {true, false}
True: header field is valid
False: header field is not valid
header ipv4_t ipv4;

field_list ipv4_checksum_list {
  ipv4.version;
  ipv4.ihl;
  ipv4.diffserv;
  ipv4.totalLen;
  ipv4.identification;
  ipv4.flags;
  ipv4.fragOffset;
  ipv4.ttl;
  ipv4.protocol;
  ipv4.srcAddr;
  ipv4.dstAddr;
}

field_list_calculation ipv4_checksum {
  input {
    ipv4_checksum_list;
  }
  algorithm : csum16;
  output_width : 16;
}

calculated_field ipv4.hdrChecksum {
  verify ipv4_checksum;
  update ipv4_checksum;
}

header_type ipv4_t {
  fields {
    version : 4;
    ihl : 4;
    diffserv : 8;
    totalLen : 16;
    identification : 16;
    flags : 3;
    fragOffset : 13;
    ttl : 8;
    protocol : 8;
    hdrChecksum : 16;
    srcAddr : 32;
    dstAddr : 32;
  }
}
Checksum Algorithms

XOR16
CSUM16
CRC16
CRC32

Programmable_CRC
Arbitrary CRC polynomial
Additional P4 Features

Counters
Type: bytes or packets
Min-width
Saturating: stop counting; default is wrap

Meters

Registers

Resubmit (original packet + metadata)

Recirculate (packet after egress modifications)

Overview of SDN, Amsterdam, 29 April 2016
P4 Control Flow

If/else

+, *, -, <<, >>, &, |, ^

~, -

OR, AND

>, >=, ==, <=, <, !=
Work Flow

Write P4 program, typically these source files:
- foo.p4
- headers.p4
- parser.p4

Convert P4 program to JSON configuration

Run P4 software switch with JSON config
**Open Compute Project**

**Started by Facebook in April 2011.**
Share design of servers, data centres, etc. and collectively improve them.

**Open Networking Project announced in May 2013.**
Open design for a network switch.

**Current Projects:**
- Specs for open hardware switches (Accton/Edge-core, Facebook, Alpha, Broadcom/Interface Masters, Mellanox, Intel)
- SAI: Switch Abstraction Interface (Microsoft, Dell, Facebook, Broadcom, Intel, Mellanox)
- ONIE: Open Network Install Environment (Cumulus Networks)
- Open Network Linux (Big Switch Networks)
Emerging Open Switch Ecosystem

Overview of SDN, Amsterdam, 29 April 2016
Open Network Install Environment

ONIE

ONIE is an Open Compute Project open source initiative contributed by Cumulus Networks that defines an open “install environment” for bare metal network switches.
Edge-Core White Label Switches

Product List

Quick Search:
- Select:
  - Data Center
  - Core Networks
  - Distribution Networks
  - Access Wired
  - Access Wireless
  - Network Software
  - Switch Accessories
  - Unmanaged Switches
  - Switch Function selector
  - EOL

Search keyword:

Data Center
- Bare Metal Hardware
  - AS6700-32X with ONIE: 40GbE Data Center Switch
  - AS6701-32X with ONIE: 40GbE Data Center Switch
  - AS5710-64X with ONIE: 10GbE Data Center Switch
  - AS5712-54X with ONIE: 10GbE Data Center Switch
  - AS5610-52X with ONIE: 10GbE Data Center Switch
  - AS5600-52X with ONIE: 10GbE TOR or Spine Switch
  - AS4600-54T with ONIE: 1GbE TOR Switch

- White-box Switch with DCSS SwitchOS
  - AS6700-32X with DCSS: 40GbE Data Center Switch
  - AS5600-52X with DCSS: 10GbE TOR or Spine Switch with DCSS L2 L3 Software
  - AS4600-54T with DCSS: 1GbE TOR Switch with L2 L3 Software

Overview of SDN, Amsterdam, 29 April 2016
Open your choices, innovate faster.
Maximize agility and choice with a full portfolio of 1/10GbE and 10/40GbE open networking switches, disaggregated OS including third-party OS, and software options.

Your choice of third-party operating systems and software
Enable business agility and avoid vendor lock-in with third-party OS and software options, including:
- Cumulus Linux OS: Simplify network management, orchestration and automation, and leverage a broad ecosystem of Linux applications.
- Big Switch Networks Switch Light™ OS: Enable a range of SDN-controller-based fabric solutions, and help reduce cost and complexity.
- Big Switch Networks Big Tap™ Monitoring Fabric™: Tap traffic everywhere in the network for exceptional visibility with an SDN-based
OpenNSL

OpenNSL Switch Specification and Software

Overview

Open Network Switch Library (OpenNSL) is a library of network switch APIs that is openly available for programming Broadcom network switch silicon based platforms. OpenNSL is a software interface with a set of open APIs that enable the development of new applications on top of Broadcom StrataXGS switches, giving customers the flexibility to tailor their network equipment and meet their unique infrastructure requirements.
OF-DPA

OpenFlow 1.3.1 Switch Pipeline Specification and Software

Overview

Broadcom’s OpenFlow Data Plane Abstraction (OF-DPA) is an application software component that implements an adaptation layer between OpenFlow and the Broadcom Silicon SDK. OF-DPA enables scalable implementation of OpenFlow 1.3 on Broadcom switch devices

Documentation
White Label Switch Firmware

PicOS L2/L3 & OpenFlow, commercial, Pica8

Cumulus Linux, L2/L3, commercial, Cumulus Networks

Switch Light OS, OpenFlow, commercial, Big Switch Networks

Open Network Linux, L2/L3 (Quagga, BIRD, FBOSS, SONiC), open source project

FBOSS, L2/L3, open source, Facebook

Azure SONiC, L2/L3, open source, Microsoft
Three approaches to Software Defined Networking
- Underlay/overlay networking
- Logically centralised control of distributed protocols
- Logically centralised control of programmable dataplanes

Disaggregation
- Hardware: white label switches (Open Compute Project)
- Software: various network operating systems

Programmable dataplanes
- OpenFlow
- P4

Goals
- Operator in control, less dependent on vendors
- Easier and less error prone network operation by automation and software driven networking
- Lower operating cost, faster innovation