Software Defined Networking & OpenFlow

SNE COLLOQUIUM 6 MAY 2014

Ronald van der Pol <Ronald.vanderPol@SURFnet.nl>
Outline

Vision behind Software Defined Networking (SDN)

OpenFlow

OpenStack & Open vSwitch

Google & OpenFlow

Network Functions Virtualisation (NFV)

Open Compute Project

Some of the OpenFlow Players

SURFnet OpenFlow project

Summary
Separation between Control Plane (policy) and Data Plane (packet forwarding).

**Logically centralised policy (control plane).**
Program (controller) that reads policy configurations, compiles it to forwarding rules and sends those to network elements → Software Defined Networking.

**“Dumb” switches (data plane).**
Firmware does packet frame forwarding only.

**Standardised protocol between switches and controllers (e.g. OpenFlow).**

**Possibility of different vendors for switches and controllers.**
More competition.
Traditional Networking

Routing and Switching Protocols
SDN Architecture
Benefits of SDN

Accelerate network innovation.

Reducing operational costs (OPEX).

Alternative for “protocol soup”.

Applying computing model to networking.
Accelerate Network Innovation

Much easier to upgrade software running on a few servers, compared to firmware upgrade of many switches and routers. User is in control, no need to wait for vendors to implement new functionality in firmware.

Network functions not limited by slow embedded processors in switches, software runs on fast servers. This makes it possible to run computationally intensive algorithms.
OPEX in Networking

Adding routers and switches to your network increases the operational cost.

Each new device needs to be configured manually via the CLI.

Changes usually involves configuration actions on several devices. Easy to make mistakes.

Firmware updates on routers and switches with slow CPUs takes a long time.

Problems are hard to debug; routers and switches are mostly black boxes. Send ping and traceroute and try to figure out what is going on.
Operational tasks in computing are usually automated.
Chef, Puppet, CFEngine, etc.
Configurations usually done on many servers at a time.

Grid and cloud clusters have powerful scaling properties.
Adding additional servers has little operational cost
Centralised policy (define once, applied consistently on all servers)

Middleware software with centralized policy (OpenStack, etc) controls the servers.

Configure the policy once and push the button to apply the changes to all servers.
## OPEX, Networking vs Computing

<table>
<thead>
<tr>
<th>Networking</th>
<th>Computing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual configuration</td>
<td>Automated configuration</td>
</tr>
<tr>
<td>Time consuming to add switches/routers</td>
<td>Easy to add servers</td>
</tr>
<tr>
<td>Firmware upgrade slow</td>
<td>Upgrades automated and fast</td>
</tr>
<tr>
<td>Configuration per device</td>
<td>Configuration on multiple servers</td>
</tr>
<tr>
<td>Policy configured per device</td>
<td>Centralised policy</td>
</tr>
<tr>
<td>Difficult to debug</td>
<td>Many debugging tools</td>
</tr>
</tbody>
</table>
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

SNE SDN & OpenFlow Colloquium, Amsterdam, 6 May 2014
IEEE 802.1Q

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)
Specs of a Modern Ethernet Switch

(random example, but they are all the same)
(slide by Nick McKeown, Stanford University)

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry
Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

(slides by Nick McKeown, Stanford University)
Computing vs Networking

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

Open Hardware
- Hypervisor API
- Portable VMs

1970
- Closed Systems
1980
1990
2000
2010
- Closed hardware
- OpenFlow API
- Open Source Applications
- Portable applications
OpenFlow is an example of SDN.

OpenFlow is the protocol between controller and switch. Standardised protocol.

Many commercial available OpenFlow switches available. They are all (modified) Ethernet switches. Many are traditional Ethernet switches with an OpenFlow API. Some are dedicated OpenFlow only switches.

Many open source and proprietary OpenFlow controllers available.
Data and Control Plane Separation

Control Plane

Data Plane

Ethernet Switch

OpenFlow Switch

Control Plane

OpenFlow protocol

Data Plane
OpenFlow Controlled Network

OpenFlow Application

OpenFlow Controller

OpenFlow Protocol

SNE SDN & OpenFlow Colloquium, Amsterdam, 6 May 2014
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 Standards

OpenFlow 1.0.0 (December 2009)
Most widely used version

OpenFlow 1.1.0 (February 2011)

OpenFlow 1.2 (December 2011)
IPv6 support, extensible matches

OpenFlow 1.3.0 (June 2012)
Flexible table miss, per flow meters, PBB support

OpenFlow 1.4.0 (October 2013)

OF-Config 1.0 (December 2011)

OF-Config 1.1 (January 2012)

OF-Config 1.2 (2014)

OpenFlow Test
Interoperability Event technical papers
OpenFlow Protocol

Insert flow forwarding entries in switches.

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
pipeline
Flow Table
Group Table
## 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

packet in → Ingress Port → Action Set = {} → table 0 → Packet + Ingress Port + metadata → Action Set → table 1 → table n → Packet → Execute Action Set → packet out
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`
### Group Table

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Group Type</th>
<th>Counters</th>
<th>Action Buckets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Identifier: 29/65
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 are 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**

```
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**

```
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
```
OpenStack is an open source cloud computing platform.

**Computing**
OpenStack Compute (Nova).  
OpenStack Image service (Glance)

**Networking**
OpenStack Networking (Neutron)

**Storage**
OpenStack Object Storage (Swift)  
OpenStack Block Storage (Cinder)

http://www.openstack.org/
OpenStack Architecture
OpenStack Networking: Neutron

OpenStack + Quantum Integration Architecture
Open vSwitch

Software switch that implements the OpenFlow protocol
- Open Source project
- Included in the Linux kernel, OpenStack, OpenNebula, …

Developed by Nicira (startup founded in 2007)
- Martin Casado (Stanford University)
- Nick McKeown (Stanford University)
- Scott Shenker (UC Berkeley)

Nicira was acquired by VMware in 2012 for USD 1.26 billion
Open vSwitch in a Cloud Environment

- **Security**: VLAN isolation, traffic filtering
- **QoS**: Traffic queuing and traffic shaping
- **Monitoring**: Netflow, sFlow, SPAN, RSPAN
- **Automated Control**: OpenFlow, OVSDB mgmt. protocol
Google Data Network

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
Google Data Network
Google OpenFlow Switch (source Google)

- 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

Traffic

Google Data Network

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.
Mixed SDN Deployment

- Ready to introduce new functionality, e.g., TE
Almost 100% Link Utilization

Sample Utilization

Utilization (percent)
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).

Goal:
Provide Network Functions through virtualisation techniques using general purpose servers and storage devices.

How:
Replace proprietary hardware network appliances by consolidating the network functions as applications running on virtual machines.
Open Compute Project

Started by Facebook in April 2011.
Build servers and data centres following the Open Source model.

Open Networking Project announced in May 2013.
Build an open network switch.

Current Projects:
• Intel: switch specification.
• Mellanox: switch specification.
• Cumulus Networks: ONIE (Open Network Install Environment).
• Broadcom: switch specification.
Some OpenFlow Players

Startups
Big Switch Networks
Nicira
Pica8

ONRC

ON.LAB

OpenDaylight consortium
Stanford University Startups

**Big Switch Networks** ([http://www.bigswitch.com/](http://www.bigswitch.com/))
FloodLight (open source OpenFlow controller), Big Switch, Big Tap.
Guido Appenzeller (CEO/Co-Founder)
- Former head of Stanford University Clean Slate program
Kyle Forster (Co-Founder, ex-Cisco, ex-Joost)
Rob Sherwood (CTO)

**Nicira** *(acquired by VMware)* ([http://nicira.com/](http://nicira.com/))
Open vSwitch (open source software switch).
Steve Mullaney (CEO)
Martin Casado (CTO, Co-Founder)
- SDN was graduate work at Stanford University, supervised by Nick McKeown & Scott Shenker
Nick McKeown (Co-Founder)
- Former faculty director of Stanford University Clean Slate program
Scott Shenker (Chief Scientist, Co-Founder)
- University of California at Berkeley
Pica8

Founded in 2008.

Open the switch and router platforms.

High quality software with merchant silicon.

PicOS based on:
XORP (open source routing project).
Open vSwitch (open source OpenFlow switch).

http://www.pica8.com/
Pica8 Switch Architecture

PicOS Architecture

Open Source
Future add-on
Proprietary (protected)

XORP
CLI
Web
SNMP

IPC Broker / XRL

L2
Switch Mgr
IPv6
CoS
L3
Route Mgr
Multicast
NVGRE
FCoE
TRILL
New App

FEA
FDB
RIB
PortMgr

PICA8 RPC

Linux 2.6

PICA8 Driver API
Broadcom SDK

Open vSwitch (OpenFlow)
Pronto Switches

**Pronto 3290**
48x 10/100/1000 BASE-T RJ45 & 4x 10GE SFP+
USD 2,750

**Pronto 3780**
48x 10GE SFP+
USD 9,950

**Pronto 3920**
48x 10GE SFP+ & 4x 40GE QSFP
USD 13,500
Open Networking Research Center

Located at Stanford University & UC Berkeley

Sponsors: CableLabs, Cisco, Ericsson, Google, Hewlett Packard, Huawei, Intel, Juniper, NEC, NTT Docomo, Texas Instruments, VMware

People:
Nick McKeown @ Stanford University
Scott Shenker @ UC Berkeley

http://onrc.stanford.edu/
Headed by Guru Parulkar
Professor at Stanford University

Build open source OpenFlow tools and platforms
Beacon, NOX, FlowVisor, Mininet

http://onlab.us/
Developed by an open consortium as a Linux Foundation Collaborative Project.

Many large vendors as members.

Most active members: Cisco, IBM, NEC, Ericsson, VMware, Red Hat.

First release: February 2014 (hydrogen)
- Base edition (controller + OpenFlow)
- Virtualisation edition (base + affinity + OpenDOVE + OVSDB + VTN + Defense4All)
- Service Provider edition (base + affinity + BGP/PCEP + LISP FlowMapping)
OpenDaylight Architecture

First Code Release “Hydrogen”
GN3plus Open Call Project (CoCo).

October 2013 – March 2015 (18 months).

Budget Eur 216K.

16.4 person months.

Partners: SURFnet (NL) & TNO (NL).

Five work packages:
WP1: use cases & market demand
WP2: architecture, design & development
WP3: experimental validation
WP4: dissemination
WP5: project management
SURFnet OpenFlow Testbed

- 5 sites
- Co-located at SURFnet7 PoPs at major universities
- Interconnected via SURFnet7
- 10GE nearly full mesh
- Each site (probably) co-located with a small OpenStack cloud cluster
- OpenDaylight controller
- Initial OF application will offer functionality for IP forwarding and multiple private multi-point networks
Goal of CoCo service:
On-demand private multi-domain, multi-point networks.
Connect laptops, VMs, storage, instruments, eScience resources.
Each eScience community group can easily setup their own private CoCo instance via web portal.

Based on OpenFlow programmable network infrastructure.
Example CoCo Instance
Forwarding in CoCo

CoCo OpenFlow application inserts flow forwarding entries.

Forwarding based on MPLS label matching and forwarding.
Label stack with two MPLS labels.
Outer MPLS label used to identify egress PE switch.
Inner MPLS label used to identify CoCo instance.

Adding and removing MPLS labels done at edges (PE).
CoCo Inter-Domain Forwarding

customer c1
OF
CE
PE
P
PE
domain d1

OF
CE
PE
P
PE
customer c3
domain d3

domain d2
VPN
PE
P
PE
CE

SNE SDN & OpenFlow Colloquium, Amsterdam, 6 May 2014
Summary

SDN is a new way to control networks.

Central role for software; users get more control over their network.

Less dependent on vendors for new features.

Many possibilities for open source and open hardware in networking.

OpenFlow is used in production. Access to forwarding tables.
Ronal van der Pol
Ronald.vanderPol@SURFnet.nl