FlowFuzz
A Framework for Fuzzing OpenFlow-enabled Software and Hardware Switches

Nicholas Gray, Manuel Sommer, Thomas Zinner, Phuoc Tran-Gia
About us

Modeling, Performance Analysis & Optimization, Measurement, Experimentation, Simulation

Software-defined Networking & Cloud Networks
Future Internet & Smartphone Applications
Network Dynamics & Control
QoE Modeling & Resource Management

Chair of Communication Networks
comnet.informatik.uni-wuerzburg.de

SarDiNe
Sardine-project.org
Agenda

• Software-defined Networking (SDN)
  • SDN Basics
  • Enhancing Network Security with SDN
  • Overview of the SDN Attack Surface
  • OpenFlow

• FlowFuzz
  • Architecture
  • Evaluation of Software Switches
  • Investigation of Feedback Sources for Hardware Switches
  • Evaluation of Hardware Switches
Speed of Innovation

Edge

Core

Data Center
Speed of Innovation
Speed of Innovation
Speed of Innovation

Ethernet, IPv4, BGP...

Edge

Core

Data Center
Speed of Innovation

Core

Ethernet, IPv4, BGP...

Data Center

Edge

Xbox Live, Netflix, YouTube, Facebook, Microsoft Azure, VMware, OpenStack
Innovation Barrier

Specialized Hardware

Control Plane
Data Plane
Innovation Barrier

- Specialized Hardware
- Proprietary Firmware

Diagram:
- Control Plane
- Data Plane
Innovation Barrier

- Specialized Hardware
- Proprietary Firmware
- Over Specification

Control Plane
Data Plane
Innovation Barrier

- Specialized Hardware
- Proprietary Firmware
- Over Specification
- Few Vendors

Control Plane

Data Plane
Software-defined Networking (SDN)

Separation of Control and Data Plane
Software-defined Networking (SDN)

- Separation of Control and Data Plane
- Control Plane
- Data Plane
- Southbound API
Software-defined Networking (SDN)

Separation of Control and Data Plane

Logically Centralized Control Plane

Control Plane

Southbound API

Data Plane
Software-defined Networking (SDN)

- Separation of Control and Data Plane
- Logically Centralized Control Plane
- Open Interfaces

Control Plane

Southbound API

Data Plane
Software-defined Networking (SDN)

- Separation of Control and Data Plane
- Logically Centralized Control Plane
- Open Interfaces
- Programmability

Control Plane

Data Plane

Southbound API
SDN – Packet Handling & Table Structure

Rule → Action → Stats
### SDN – Packet Handling & Table Structure

**Rule** → **Action** → **Stats**

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>Switch Phy Port</th>
<th>Meta data</th>
<th>ETH Dst</th>
<th>ETH Src</th>
<th>ETH Type</th>
<th>VLAN VID</th>
<th>VLAN PCP</th>
<th>IP DSCP</th>
<th>IP ECN</th>
<th>IP Proto</th>
<th>IPv4 Src</th>
<th>IPv4 Dst</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMPv4 Type</td>
<td>ICMPv4 Code</td>
<td>TCP Src</td>
<td>TCP Dst</td>
<td>UDP Src</td>
<td>UDP Dst</td>
<td>SCTP Src</td>
<td>SCTP Dst</td>
<td>ARP OP</td>
<td>ARP SPA</td>
<td>ARP TPA</td>
<td>ARP SHA</td>
<td>ARP THA</td>
<td>...</td>
</tr>
</tbody>
</table>

Mask for match fields
SDN – Packet Handling & Table Structure

- Rule
- Forward packet to zero or more ports
- Encapsulate and forward to controller
- Send to normal processing pipeline
- Modify Fields
- Any extensions you add!

- Action

- Stats

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>Switch Phy Port</th>
<th>Meta data</th>
<th>ETH Dst</th>
<th>ETH Src</th>
<th>ETH Type</th>
<th>VLAN VID</th>
<th>VLAN PCP</th>
<th>IP DSCP</th>
<th>IP ECN</th>
<th>IP Proto</th>
<th>IPv4 Src</th>
<th>IPv4 Dst</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMPv4 Type</td>
<td>ICMPv4 Code</td>
<td>TCP Src</td>
<td>TCP Dst</td>
<td>UDP Src</td>
<td>UDP Dst</td>
<td>SCTP Src</td>
<td>SCTP Dst</td>
<td>ARP OP</td>
<td>ARP SPA</td>
<td>ARP TPA</td>
<td>ARP SHA</td>
<td>ARP THA</td>
<td>…</td>
</tr>
</tbody>
</table>

Mask for match fields
**SDN – Packet Handling & Table Structure**

**Rule**
- Forward packet to zero or more ports
- Encapsulate and forward to controller
- Send to normal processing pipeline
- Modify Fields
- Any extensions you add!

**Action**

**Stats**
- Packet + Byte Counters

**Table Structure**

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>Switch Phy Port</th>
<th>Meta data</th>
<th>ETH Dst</th>
<th>ETH Src</th>
<th>ETH Type</th>
<th>VLAN VID</th>
<th>VLAN PCP</th>
<th>IP DSCP</th>
<th>IP ECN</th>
<th>IP Proto</th>
<th>IPv4 Src</th>
<th>IPv4 Dst</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMPv4 Type</td>
<td>ICMPv4 Code</td>
<td>TCP Src</td>
<td>TCP Dst</td>
<td>UDP Src</td>
<td>UDP Dst</td>
<td>SCTP Src</td>
<td>SCTP Dst</td>
<td>ARP OP</td>
<td>ARP SPA</td>
<td>ARP TPA</td>
<td>ARP SHA</td>
<td>ARP THA</td>
<td>...</td>
</tr>
</tbody>
</table>

Mask for match fields
SDN Example

Control Plane (CP)

Reactive

Southbound API

Data Plane (DP)

A

B

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>.</em></td>
<td>CP</td>
</tr>
</tbody>
</table>
SDN Example

Control Plane (CP)

Reactive

Southbound API

Data Plane (DP)

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>.</em></td>
<td>→ CP</td>
</tr>
</tbody>
</table>
SDN Example

Control Plane (CP)

Reactive

Southbound API

Data Plane (DP)

Match | Action
--- | ---
*.* | → CP

A ➜ B
SDN Example

Control Plane (CP)

Reactive

Southbound API

Data Plane (DP)

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*. *</td>
<td>CP</td>
</tr>
</tbody>
</table>

A ➔ CP

B ➔ CP
SDN Example

Control Plane (CP)

Reactive

Southbound API

Data Plane (DP)

Match | Action
--- | ---
*.* | CP

A

B
SDN Example

Control Plane (CP)

Reactive

Southbound API

Data Plane (DP)

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td><em>.</em></td>
<td>CP</td>
</tr>
</tbody>
</table>
**SDN Example**

### Control Plane (CP)

#### Reactive

- Southbound API
- Data Plane (DP)
- A → CP
- B

#### Proactive

- Southbound API
- Data Plane (DP)
- A
- B

### Match Action Table

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>B</td>
</tr>
<tr>
<td><em>.</em></td>
<td>CP</td>
</tr>
<tr>
<td><em>.</em></td>
<td>CP</td>
</tr>
</tbody>
</table>

**Reactive Proactive**
SDN Example

Control Plane (CP)

Reactive

Southbound API

Data Plane (DP)

Match | Action
--- | ---
A | B
*.* | CP

Proactive

Southbound API

Data Plane (DP)

Match | Action
--- | ---
*.* | CP
SDN Example

Control Plane (CP)

Reactive

Southbound API

Match | Action
--- | ---
*.* | CP

Data Plane (DP)

Proactive

Match | Action
--- | ---
*.* | CP

Data Plane (DP)

A

B

A

B
SDN Example

Control Plane (CP)

Reactive

Southbound API

Match | Action
--- | ---
![](green-check) | B
*.* | CP

Data Plane (DP)

A → B

Control Plane (CP)

Proactive

Southbound API

Match | Action
--- | ---
![](green-check) | B
*.* | CP

Data Plane (DP)

A → B

Reactive vs. Proactive:

- Reactive: 
  - Match: ![](green-check)
  - Action: B
  - Match: *.*
  - Action: CP

- Proactive: 
  - Match: ![](green-check)
  - Action: B
  - Match: *.*
  - Action: CP
SDN Ecosystem

SDN Control Plane

Network Control Module
Application Control Interface
Network Control Module

Switch

SDN WAN
SDN Ecosystem

Application Control Plane

Northbound API

SDN Control Plane

Switch

Switch

SDN WAN
SDN Ecosystem

Application Control Plane

Northbound API

Westbound API

Eastbound API

Southbound API

SDN Control Plane

Legacy Network Control Plane

Application Control Module

Network Control Module

Application Control Interface

Network Control Module

Hypervisor

vSwitch

Cloud

Switch

SDN WAN

Legacy WAN
SDN Use Cases

- Cloud Orchestration
- Application Awareness
- Routing/Load Balancing
- Network Monitoring
- Network Management
- Network Security

Software-defined Networking
SDN Use Cases

- Cloud Orchestration
- Application Awareness
- Routing/Load Balancing
- Network Monitoring
- Network Management
- Network Security

Software-defined Networking
Can we enhance network security with SDN?
Can we enhance network security with SDN?
Can we enhance network security with SDN?
Can we enhance network security with SDN?

External Network → Internal Network
Can we enhance network security with SDN?
Can we enhance network security with SDN?
Can we enhance network security with SDN?
Can we enhance network security with SDN?
Can we enhance network security with SDN?
SDN Omni-Present Firewall

SDN Controller

Network Management System

Cloud Management System

Internal Network

SDN Switch

SDN Switch

AAA

FW VNF

Services

Private Cloud
Available Services
- Portal Service
- Business Intelligence
- SSH to Webserver
- Internet
- Email
- File Transfer
- Print

Available Services
- AAA
- FW VNF
- Services

SDN Omni-Present Firewall
- SDN Controller
- Network Management System
- Cloud Management System

Internal Network
- SDN Switch
- SDN Switch

Private Cloud
Internal Network

SDN Omni-Present Firewall

SDN Controller → Network Management System → Cloud Management System

Available Services
- PortalService
- Business Intelligence
- SSH to Webserver
- Internet
- Email
- File Transfer
- Print

Internal Network

SDN Switch → SDN Switch

Private Cloud

Shared State

AAA
FW VNF
FW VNF
Services

Private Cloud
SDN Attack Surface

SDN Control Plane

Northbound API

Application Control Module

Application Control Interface

Network Control Module

Westbound API

Legacy Network Control Plane

Eastbound API

Legacy WAN

Southbound API

Switch

Switch

vSwitch

Cloud

SDN WAN
SDN Attack Surface

Application Control Plane
- Input Validation & Representation
- API Abuse
- Security Features
- Time and State
- Errors
- Code Quality
- Encapsulation
- Environment
SDN Attack Surface

Northbound API
- No standardization
- Controller dependent
- Bi/Uni-directional communication
- Often RESTful Webservices
SDN Attack Surface

**SDN Network Control Plane**

- 25+ controller implementations
- 250000+ lines of code
- Centralized & distributed controllers
- Open Source and proprietary solutions
- Often lack of basic security features
**Eastbound API**

- No standardization
- Flexible vs. static nature of devices
- Synchronization issues
- Integration challenges
SDN Attack Surface

Westbound API
- No standardization
- Controller dependent
- Various aggregation levels
- Synchronization issues
SDN Attack Surface

Southbound API & SDN-enabled Devices
- Standardized protocols
- Focal point of information exchange
- Potential pivot point for an attacker
- Virtual and hardware SDN-enabled switches
- Directly and indirectly exposed to attackers
OpenFlow

- De-facto standard Southbound API protocol
- Maintained by the Open Networking Foundation
- First release in December 2009
- Most current version 1.5.1 (April 2015)
- Supported by 120+ industrial members
OpenFlow – Channel Initialization
OpenFlow – Channel Initialization

OpenFlow Switch

Controller

MAC Resolution

ARP Request

ARP Reply

TCP Handshake

TCP SYN

SYN ACK

ACK
OpenFlow – Channel Initialization

MAC Resolution
- ARP Request → ARP Reply
- MAC Resolution

TCP Handshake
- TCP SYN → SYN ACK → ACK

Hello
- Hello
- Feature Request → Feature Reply

OpenFlow Switch → Controller

OpenFlow Switch

Controller
OpenFlow – Message Structure & Types

OpenFlow Message Header

- Version
- Type
- Length
- XID

...Payload...
OpenFlow – Message Structure & Types

OpenFlow Message Header
- Version
- Type
- Length
- XID

...Payload...

Asynchronous  Controller-to-Switch  Symmetric
OpenFlow – Message Structure & Types

**OpenFlow Message Header**
- **Version**
- **Type**
- **Length**
- **XID**

...Payload...

**Asynchronous**
- Packet-In
- Flow Removed
- Port Status
- Error

**Controller-to-Switch**
- Feature Request, Get Config Request, Set Config,
  Packet-Out, Flow Modification, Group
  Modification, Port Modification, Table
  Modification, Meter Modification, Statistics
  Request, Barrier Request, Queue Get Config
  Request, Role Request, Get Asynchronous
  Request, Set Asynchronous

**Symmetric**
- Hello
- Echo Request
- Echo Reply
- Experimeter
Fuzzing

Automated Fuzzer → Random Input → Unexpected Input → System/Device Under Test

Check for Crash
Fuzzing

Automated Fuzzer

Mutation
Mutates valid input

Generation
Generates valid/invalid input

Random Input

Invalid Input

Unexpected Input

System/Device Under Test

Check for Crash
Fuzzing

Feedback Loop
- Invalid Input
- Unexpected Input
- Random Input

Check for Crash

Automated Fuzzer
- Mutation: Mutates valid input
- Generation: Generates valid/invalid input

System/Device Under Test
Open vSwitch (OvS)

- Production quality, multilayer open virtual switch
- Integrated into OpenStack, Xen, Pica8...
- Fully supports OpenFlow up to v1.4
- Operates either as software switch or as control stack for dedicated hardware

Diagram:
- User Space: ovs-vswitchd, openvswitch.ko, ovsdb, Virtual Switch
- Kernel Space: Virtual Switch, Virtual Switch, openvswitch.ko
Open vSwitch Fuzzer – A First Try

Ryu OpenFlow Controller

Mutation Fuzzer

Open vSwitch
Open vSwitch Fuzzer – A First Try

Ryu OpenFlow Controller

Mutation Fuzzer

Open vSwitch

× Lack of control
Open vSwitch Fuzzer – A First Try

- Ryu OpenFlow Controller
- Open vSwitch
- Mutation Fuzzer

- Lack of control
- Controller needs to be actively triggered
Open vSwitch Fuzzer – A First Try

- Lack of control
- Controller needs to be actively triggered
- Hard to integrate a feedback loop
Open vSwitch Fuzzer – A First Try

- lack of control
- controller needs to be actively triggered
- hard to integrate a feedback loop

→ simple and fast but no promising approach
FlowFuzz

- Protocol Aware
- Python Based
- Supports OF v1.0/1.3
- Corpus of Valid Inputs
- Directed and Random Input Generation
- Various Sources as Feedback Loop
FlowFuzz – Architecture & Stages
FlowFuzz – Architecture & Stages

Pre-condition → Test Execution → Validation → Post-condition

Test Manager
Log Module
Testcase Loader
Replay Module
TCP Connection Handler
FlowFuzz – Architecture & Stages

- Table Initiation
- OF Handshake
- Pre-condition
- Test Execution
- Validation
- Post-condition

- Test Manager
  - Log Module
  - Testcase Loader
  - Replay Module
  - TCP Connection Handler
FlowFuzz – Architecture & Stages

- **Table Initiation**
- **OF Handshake**
- **Pre-condition**
- **Transmission**
- **Input Generation**
- **Test Execution**
- **Validation**
- **Post-condition**

**Modules:**
- Test Manager
- Log Module
- Testcase Loader
- Replay Module
- TCP Connection Handler
FlowFuzz – Architecture & Stages

**Stage 1:** Pre-condition

- Table Initiation
- OF Handshake

**Stage 2:** Test Execution

- Transmission
- Input Generation

**Stage 3:** Validation

- Logging
- Sanity Checks

**Stage 4:** Post-condition

**Supporting Modules:**

- Test Manager
  - Log Module
  - Testcase Loader
  - Replay Module
- TCP Connection Handler
FlowFuzz – Architecture & Stages

Table Initiation → OF Handshake → Pre-condition

Transmission → Input Generation → Test Execution

Logging → Sanity Checks → Validation

Reset → Evaluation

Pre-condition → Test Execution → Validation → Post-condition

Test Manager

Log Module → Testcase Loader → Replay Module

TCP Connection Handler
FlowFuzz – Architecture & Stages

- Table Initiation
- OF Handshake
- Pre-condition
- Transmission
- Input Generation
- Test Execution
- Sanity Checks
- Validation
- Logging
- Test Manager
  - Log Module
  - Testcase Loader
  - Replay Module
  - TCP Connection Handler
- Reset
- Evaluation
- Post-condition
Open vSwitch – Test Bed

All compiled with AddressSanitizer
Open vSwitch – Fuzzer Evaluation

- Test duration of one week
- Targeted OpenFlow version 1.0
- Crafted and random inputs
- Code coverage as main feedback source

<table>
<thead>
<tr>
<th></th>
<th>v1.5</th>
<th>v2.0</th>
<th>v2.5</th>
<th>v2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Anomalies</strong></td>
<td>2538</td>
<td>2986</td>
<td>2263</td>
<td>2047</td>
</tr>
<tr>
<td><strong>Crashes</strong></td>
<td>13</td>
<td>10</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>
Open vSwitch – Fuzzer Evaluation

- Test duration of one week
- Targeted OpenFlow version 1.0
- Crafted and random inputs
- Code coverage as main feedback source

⇒ High number of false positives due to switch reconnects

<table>
<thead>
<tr>
<th>Results</th>
<th>v1.5</th>
<th>v2.0</th>
<th>v2.5</th>
<th>v2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anomalies</td>
<td>2538</td>
<td>2986</td>
<td>2263</td>
<td>2047</td>
</tr>
<tr>
<td>Crashes</td>
<td>13</td>
<td>10</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>
Open vSwitch – Fuzzer Evaluation

- Test duration of one week
- Targeted OpenFlow version 1.0
- Crafted and random inputs
- Code coverage as main feedback source

\[\rightarrow\] High number of false positives due to switch reconnects
\[\rightarrow\] Crashes due to environment setup and could not be reproduced

<table>
<thead>
<tr>
<th>Results</th>
<th></th>
<th>v2.0</th>
<th>v2.5</th>
<th>v2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>v1.5</td>
<td>2538</td>
<td>2986</td>
<td>2263</td>
</tr>
<tr>
<td>Anomalies</td>
<td>13</td>
<td>10</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>
Open vSwitch – Fuzzer Evaluation

- Test duration of one week
- Targeted OpenFlow version 1.0
- Crafted and random inputs
- Code coverage as main feedback source

<table>
<thead>
<tr>
<th>Results</th>
<th>v1.5</th>
<th>v2.0</th>
<th>v2.5</th>
<th>v2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anomalies</td>
<td>2538</td>
<td>2986</td>
<td>2263</td>
<td>2047</td>
</tr>
<tr>
<td>Crashes</td>
<td>13</td>
<td>10</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>

→ High number of false positives due to switch reconnects
→ Crashes due to environment setup and could not be reproduced
→ No security flaws detected – yet!
Hardware Switch – Feedback Sources

- NEC PF5240
- Pronto 3290
- HP 2920-24G
- Quanta T1048-LB9
Hardware Switch – Feedback Sources

Traditional guided fuzzing mechanisms cannot be applied!
Hardware Switch – Feedback Sources

- NEC PF5240
- Pronto 3290
- HP 2920-24G
- Quanta T1048-LB9
Hardware Switch – Feedback Sources

- Protocol Errors
- Debug Mode
- Device Log

Black Box?

- NEC PF5240
- Pronto 3290
- HP 2920-24G
- Quanta T1048-LB9
Traditional guided fuzzing mechanisms cannot be applied!
Traditional guided fuzzing mechanisms cannot be applied!

Combine all sources to create an unique signature per input.
Traditional guided fuzzing mechanisms cannot be applied!

Combine all sources to create an unique signature per input.
Feedback Sources – Measuring Response Times

Start = Intended Message

End = Barrier Reply

→ Timediff = End - Start
Feedback Sources – Evaluation of Response Times

**HP 2920-24G**

![Graph for HP 2920-24G](image)

**Pronto 3290**

![Graph for Pronto 3290](image)
Hardware Switch – Fuzzer Evaluation

- Test duration of 12h
- Targeted OpenFlow version 1.0
- Crafted and random inputs
- Response times as main feedback source

→ High number of false positives due to switch reconnects
→ No security flaws detected – yet!

<table>
<thead>
<tr>
<th>Version</th>
<th>NEC</th>
<th>HP</th>
<th>Quanta</th>
<th>Pronto</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomalies</td>
<td>2133</td>
<td>1735</td>
<td>1915</td>
<td>2643</td>
</tr>
<tr>
<td>Crashes</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Flow Fuzz – Next Steps & Future Extension

Measurements
• Reduce false positive rate
• Increase test duration
• Fuzz OpenFlow v1.3

Extensions
• Support higher OF versions
• Optimize feedback loop
• Agents for DP fuzzing

Corpus Generation
• Categorized by OF version
• Derived from code coverage
Sound Bytes

- SDN is coming – Be prepared!
- SDN can enhance the security of networks
- FlowFuzz – A protocol-aware OpenFlow fuzzing framework
- De-blackboxing black boxes by using alternative feedback sources
Sources

• Michael Jarschel, Thomas Zinner, Tobias Hoßfeld, Phuoc Tran-Gia, and Wolfgang Kellerer, Interfaces, Attributes, and Use Cases: A Compass for SDN, IEEE Communications Magazine, 52, 2014


Sources


Sources

• Changhoon Yoon, Seungsoo Lee, *Attacking SDN Infrastructure: Are We Ready for the Next-Gen Networking?*, *Black Hat USA 2016*

• Jennia Hizver, *Taxonomic Modeling of Security Threats in Software Defined Networking*, *Black Hat USA 2015*

• Gregory Pickett, *Abusing Software Defined Networks*, *Black Hat Europe 2014*

Sources

• Open Networking Foundation, https://www.opennetworking.org, called on 2017-07-14

• Open Networking Foundation, OpenFlow Switch Specification Version 1.3.5, called on 2017-07-14


• OpenVSwitch - Linux Foundation, https://openvswitch.org, called on 2017-07-14
Sources

• Ryu SDN Framework Community
  https://osrg.github.io/ryu/,
  called on 2017-07-14

• OpenStack – Open Source Cloud Computing Software
  https://www.openstack.org/
  called on 2017-07-14