OpenFlow - the key standard of Software-Defined Networks

Dmitry Orekhov, Epam Systems
# Software-defined network

## The Need for a New Network Architecture

<table>
<thead>
<tr>
<th>Limitations of Current Networking Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity that leads to stasis</td>
</tr>
<tr>
<td>Inconsistent policies</td>
</tr>
<tr>
<td>Inability to scale</td>
</tr>
<tr>
<td>Vendor dependency</td>
</tr>
</tbody>
</table>

- Changing traffic patterns
- The rise of cloud services
- “Big data” means more bandwidth
- The “consumerization of IT”

## The key idea of SDN

Network control is decoupled from forwarding and is directly programmable.
OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture.
Packet forwarding inside OpenFlow switch

### Flow Tables

- **Flow Table 0**
  - Match, Instruction
  - ...

- **Flow Table 1**
  - Match, Instruction
  - ...

- **Flow Table N**
  - Match, Instruction
  - ...

- Packet may be transferred to other table
- Packet header may be modified
- Packet may be forwarded to given port or just dropped
- Packet may be applied to given QoS
OpenFlow Switch: key elements

OpenFlow tables
Pipeline
Ports
OpenFlow Channel
Flow table entry: key elements

<table>
<thead>
<tr>
<th>Match Fields</th>
<th>Priority</th>
<th>Counters</th>
<th>Timeout</th>
<th>Cookies</th>
<th>Instruction set</th>
</tr>
</thead>
</table>

**Match criteria:**

- Ingress-port
- Ethernet MAC
- ARP
- IPv4 and IPv6
- TCP ports
- VLAN, MPLS etc.

**Instruction:**

- Go-To Table
- Modify Metadata
- Action Set {forward, apply QoS, drop}
OpenFlow examples

<table>
<thead>
<tr>
<th>Switching</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>00:1f</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Port6</td>
</tr>
<tr>
<td>Flow switching</td>
<td>Port3</td>
<td>00:20..</td>
<td>00:1f 0800 Vlan1 1.2.3.4 5.6.7.8 4 17264 Port6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firewall</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>22 Drop</td>
<td></td>
</tr>
<tr>
<td>Routing</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>Port6</td>
</tr>
<tr>
<td>VLAN switching</td>
<td>*</td>
<td>*</td>
<td>00:1f Vlan1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>* Port6, port7, port8</td>
<td></td>
</tr>
</tbody>
</table>

OpenFlow can be compared to the instruction set of a CPU. It specifies basic primitives that can be used by an external software application to program the forwarding plane of network devices, just like the instruction set of a CPU would program a computer system.
OpenFlow Protocol: key messages

- Handshake
- Configuration
- Modify
- Statistics
- Error

- Asynchronous messages: Packet-In
- Symmetric messages
OF Controller – Switch: Feedback

- Packet-In – Packet-Out: Controller learns Switch based on information about incoming packets sent by Switch

- Error messages: Switch sends to controller messages about malformed or inappropriate packets.
OpenFlow is evolving fast

- Active and professional community
- Support of academic people and majors (IBM, HP, Huawei etc... )
- Close communication between attendees: mailing lists, JIRA, face-to-face events (interop plugfest)

As a result - moving from 1.0 to 1.3.1!
All necessary information is stored in openflow.h

```c
/* Header on all OpenFlow packets. */
struct ofp_header {
    uint8_t version;   /* OFP_VERSION. */
    uint8_t type;      /* One of the OFPT_ constants. */
    uint16_t length;   /* Length including this ofp_header. */
    uint32_t xid;      /* Transaction id */
};
OFP_ASSERT(sizeof(struct ofp_header) == 8);
```
Open Source implementations

- CPqD (http://www.cpqd.com.br/) + Ericsson Research: NOX Controller (C), POX Controller (Python). OpenFlow library for switch (C, python) OpenFlow 1.2, 1.3
- Open vSwitch (http://openvswitch.org/) Used as a switch in Xen, VirtualBox. OpenFlow 1.0
- FloodLight controller (http://floodlight.openflowhub.org/). OpenFlow 1.0, LOXI announced.
LINC switch

OF Configuration Point

OF-Config

OF Controller

OF Protocol

LINC

Userspace implementation

API (gen-switch)

HW

Kernel mode implementation
Apache Avro

- Serialization library like Protobuf or Thrift
  Schema written on JSON-based specification
  Also you can use IDL

- Schema may be parsed in runtime.

Resulting Java-objects contains all information about data.
No metainformation put into a wire.
Dynamic parsing.

But encoding still exists. Also, some lack in Avro schema specification
Reference

- OpenNetworking Foundation (OpenFlow documents)
  https://www.opennetworking.org/about/onf-documents

- FlowForwarding (LINC Switch)
  http://www.flowforwarding.org/

- Floodlight OpenFlow controller
  http://floodlight.openflowhub.org/

- Apache Avro
  http://avro.apache.org

- And me, Dmitry Orekhov (Dmitry_Orekhov@epam.com)
Thank you!