Software Defined Networking
and the design of OpenFlow switches

Paolo Giaccone

Notes for the class on “Packet Switch Architectures”

Politecnico di Torino

December 2015
Outline

1. Introduction to SDN
2. OpenFlow
3. Design of OpenFlow switches
Section 1

Introduction to SDN
Network Architecture

Control plane
- how to handle traffic (e.g. routing)

Data plane
- forward the traffic based on the control plane decision
Traditional networks

Internet architecture

- vertically integrated
  - control and data planes within each device
  - control plane distributed across the switches/routers
- complex and hard to manage
  - due to distributed control plane
  - difficult to understand the state of the network and its history
- vendor-specific commands to manage switches/routers
## Software Defined Networking (SDN)

**Emerging networking paradigm**

**Separation between control and data plane**
- routers and switches just acting as forwarding elements

**Flow-based forwarding decision**
- instead of destination-based forwarding
- **flow** definition: a set of packet field values acting as a *match rule* and a set of *actions* to operate on all packets belonging to the same flow
- unify the behavior of routers, switches, firewalls, load-balancers, traffic shapers, etc.
Software Defined Networking (SDN)

Logical centralization of the control plane

- unique abstract view of the network state (e.g. topology)
- control logic moved to an external entity
  - SDN controller or Network Operating System (NOS)

Network programmability

- network applications run on the SDN controller
  - similar to computer applications running on computer operating system
- standard or ad-hoc languages
SDN controller

- software platform running on commodity servers
- logically centralized
  - for scalability and reliability reasons, can be distributed in different servers
- open-source examples: NOX, POX, Ryu, ONOS, OpenDaylight, etc.

![SDN Architecture Diagram](image-url)

Fig. 1. Simplified view of an SDN architecture.

Image taken from [Kr15]
### Northbound Interface (NBI) API

- APIs available to application developers
  - CLI (Command Line Interface), GUI (Graphical User Interface), REST APIs
  - e.g. "`curl http://controllerIP:controllerPort/command`"
- abstracts the low-level instructions to program forwarding devices
- needed to develop the network applications with any programming language

### Southbound Interface (SBI) API

- protocol to access the switches and to send them commands
- enables the interaction between control and data plane
Adoption of SDN

- Google uses SDN
  - in WAN to interconnect data centers
  - within the data center
- network operators and ISP are also considering to move to SDN networks
- next generation 5G networks are based on SDN in all the layers
Section 2

OpenFlow
OpenFlow

- initial idea proposed at Stanford University in 2006
- definition by Open Networking Foundation (ONF)
  - ONS promotes the adoption of SDN through open standards
  - OpenFlow 1.0 released on Dec 2009
  - OpenFlow 1.5.1 released on Mar 2015
- defines the protocol adopted in the southbound interface (SBI)

Image taken from [OF15]
OpenFlow messages

- transported on TLS or TCP connections

**Packet-in**
- from the switch to the controller
- to transfer the control of a received packet to the controller
- carries a copy of the packet (possibly, only the header)
- generated by default in case of forwarding table misses

**Packet-out**
- from the controller to the switch
- to send packets out of a specified port on the switch
- carries the full packet or an ID buffer of the switch

**Flow-mod**
- from the controller to the switch
- to modify the flow tables
- carries the match-action rule to install in the switch
OpenFlow example
### Ethernet switch (e.g. to reach 68:a8:6d:00:81:42)

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>IP src</th>
<th>IP dst</th>
<th>TCP src port</th>
<th>TCP dst port</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>68:a8:6d:00:81:42</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Forward P1</td>
</tr>
</tbody>
</table>

### IP router (e.g. direct delivery on 130.192.9.0/24)

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>IP src</th>
<th>IP dst</th>
<th>TCP src port</th>
<th>TCP dst port</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0x0800</td>
<td>130.192.9.*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Forward P2</td>
</tr>
</tbody>
</table>

### Firewall (e.g. block BitTorrent from all Politecnico’s hosts)

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>IP src</th>
<th>IP dst</th>
<th>TCP src port</th>
<th>TCP dst port</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0x0800</td>
<td>130.192.*</td>
<td>*</td>
<td>*</td>
<td>6969</td>
<td>Drop</td>
</tr>
</tbody>
</table>
OpenFlow fields

<table>
<thead>
<tr>
<th>Version</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenFlow 1.0</td>
<td>12</td>
</tr>
<tr>
<td>OpenFlow 1.3</td>
<td>40</td>
</tr>
<tr>
<td>OpenFlow 1.5</td>
<td>44</td>
</tr>
</tbody>
</table>

Examples: switch input port, Ethernet frame type and addresses, VLAN id and priority, IP DSCP/ECN flags and protocol and addresses, TCP ports and flags, UDP ports, ICMP type and code, ARP op code and addresses, MPLS labels and flags, metadata passed between tables.
Section 3

Design of OpenFlow switches
Flow (or Forwarding) Tables

Each rule is a match-action pair
- **Match**
  - binary exact match (0,1)
  - ternary match (0,1, *)
- **Action**
  - drop, forward, modify, goto another table

Flexible

Enables to implement routers, switches, firewalls, load balancers, traffic shapers, etc.
Packet processing within each flow table

Matching based only on the header fields

Figure 4: Matching and Instruction execution in a flow table.

Image taken from [OF15]
Implementation on NetFPGA

Implementation in FPGA used as reference model for hardware architecture of any OpenFlow switch

![Diagram of OpenFlow switch pipeline](image_taken_from_[Na08](image_taken_from_[Na08]))
- New programmable architecture compatible with any packet processing
- Compatible with OpenFlow

(a) RMT model as a sequence of logical Match-Action stages. Image taken from [Bo13]
### OpenState
- state machine within each switch
- delegates some control from the controller to the switch
- refer to [BC14]

### P4
- protocol-independent packet processors
- new programming language
- internal state machines within each switch
- possible candidate for OpenFlow 2.0
- refer to [Bo14]
Bibliography


