Reti di Telecomunicazione
a.y. 2015-2016

Software-Defined Networking (SDN)
Overview

Ing. Luca Davoli
Ph.D. Student
Network Security (NetSec) Laboratory

davoli@ce.unipr.it
Overview

- From classical network architectures to SDN
- SDN concepts
- OpenFlow: a SDN-oriented protocol
- SDN applications: TE & SR
- SDN controllers & emulator
- SDN in the future: Cloud & NFV
Classical network architecture

- Distributed control plane
- Distributed routing protocols: OSPF, IS-IS, BGP, etc.
- Huge complexity in integration among different HW vendors
- Closed equipments
- Slowness in innovation
Classical network architecture [cont.]

- Vertically integrated, closed, proprietary
- Many complex functions baked into the infrastructure
- **OSPF, BGP, multicast, differentiated services, NAT, firewalls, redundant layers, etc.**
- Little ability for non-TLC network operators to get what they want
- Functionalities defined by standards and put in hardware

- Million of source code lines, barrier to entry
- Billions of gates, complex, power hungry
Evolution in network architectures

Feature

Simple Packet Forwarding HW

Well-defined open API

NorthBound Interface (NBI)

Open protocol

SouthBound Interface (SBI)

Feature

Simple Packet Forwarding HW

Simple Packet Forwarding HW

Simple Packet Forwarding HW

Luca Davoli – davoli@ce.unipr.it
Network OS

- Distributed system that creates a consistent and up-to-date network view

- Runs on servers (controllers) in the network

- Uses an open protocol to:
  - Get state information from forwarding elements
  - Give control directives to forwarding elements
Software Defined Networking (SDN)

- Key computing trends: changing traffic patterns, rise of Cloud services, Big Data
- Current network constrained: complexity that leads to stasis, inability to scale, vendor dependence
- Allows to decouple control layer, with traffic control function, from the underlying infrastructure layer, composed by network hardware
- Virtualization
- Networks must be able to adapt in terms of security, scalability and manageability
Software Defined Networking (SDN) [cont.]

- Directly programmable, programmatically configured, centrally managed
- Network intelligence and state are logically centralized
- The underlying network infrastructure is abstracted from the applications

- Execute or run control plane software on general-purpose hardware
  - Use commodity servers
- Maintain, control and program data plane state from a central entity
- An architecture to control not just a networking device but an entire network
Software Defined Networking (SDN) [cont.]

- Better QoS, separating the control plane from the data plane, and abstracting the underlying hardware complexity

- Interaction with APIs at both Northbound (NBIs) and Southbound interfaces (SBIs)

- Pure solution, with SDN protocol-enabled devices only

- Hybrid solution, combining SDN protocols IP, to maintain IP features, increasing devices functionalities
Software Defined Networking (SDN) [cont.]

- Different behaviors at data plane due to different nodes
- Different applications atop the control plane
OpenFlow: a SDN-oriented protocol [cont.]

- SDN Controller
  - OpenFlow protocol
  - Secure channel
    - SW
    - Flow table
      - HW
  - Add/delete flow entries
  - Encapsulated packets
  - Controller discovery

Luca Davoli – davoli@ce.unipr.it
OpenFlow: a SDN-oriented protocol [cont.]

- Controls how packets are forwarded
- Implementable on COTS hardware
- Make deployed networks programmable (not just configurable)
- Goal
  - No more special purpose testbeds
  - Validate experiments on deployed hardware with real traffic at full-line speed

SDN ≠ OpenFlow
OpenFlow: a SDN-oriented protocol [cont.]

- More speed, scale and fidelity of vendor hardware
- More flexibility and control of software and simulation
- Vendors do not need to expose their own implementation
- Leverages hardware inside most switches today
Traditional network node: Switch

- Typical networking software
  - Management plane → vendor-dependant protocols
  - Control plane → decision handler
  - Data plane → packet forwarder
Traditional network node: Router

- Router can be partitioned into Control and Data plane
  - Management plane / configuration
  - Control plane / decision: Open Shortest Path First (OSPF) protocol
  - Data plane / forwarding

![Diagram showing the partitioning of a router into different planes: Management, Control, and Data (Forwarding)]
OpenFlow basics

OpenFlow Controller

OpenFlow protocol (SSL/TCP)

Control plane (SW)

OpenFlow

Data plane (HW)
OpenFlow basics [cont.]

Network OS

«if header match \( r_1 \rightarrow action_1 \)»
«if header match \( r_2 \rightarrow action_2 \)»
«if header match \( r_3 \rightarrow action_3 \)»
OpenFlow basics [cont.]

- Match arbitrary bits in headers
  - Match on any header, or new header (in theory….)
  - Allows any flow granularity

- Action
  - Forward to port(s), drop, send to controller
  - Overwrite header with mask, PUSH or POP
  - Forward at specific bit-rate

<table>
<thead>
<tr>
<th>Header</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match: 1000x01xx0101001x</td>
<td></td>
</tr>
</tbody>
</table>
OpenFlow basics [cont.]

- Flow tables entry

1. Forward packet to zero or more ports
2. Encapsulate and forward to SDN controller
3. Send to normal processing pipeline
4. Modify fields
5. Any other extension
OpenFlow example

Flow Table

<table>
<thead>
<tr>
<th>MAC src</th>
<th>MAC dst</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>port 1</td>
</tr>
</tbody>
</table>

port 1  port 2  port 3  port 4

5.6.7.8

Controller

Luca Davoli – davoli@ce.unipr.it
OpenFlow example [cont.]

Switching

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</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></td>
<td></td>
<td>00:1f...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>port6</td>
</tr>
</tbody>
</table>

Flow Switching

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>port3</td>
<td>00:20..</td>
<td>00:1f..</td>
<td>0800</td>
<td>vlan1</td>
<td>1.2.3.4</td>
<td>5.6.7.8</td>
<td>4</td>
<td>17264</td>
<td>80</td>
<td>port6</td>
</tr>
</tbody>
</table>

Firewall

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22</td>
<td>drop</td>
</tr>
</tbody>
</table>
OpenFlow example [cont.]

### Routing

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</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>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>Port6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mac dst = xxx</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mac src = yyy</td>
</tr>
</tbody>
</table>

### VLAN Switching

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</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>vlan1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6, port7, port9</td>
</tr>
</tbody>
</table>
Reactive vs Proactive

- Reactive
  - First flow packet triggers controller to insert flow entries
  - Efficient use of flow tables
  - Every flow incurs small additional flow setup time
  - If connection’s control is lost, switch has limited utility
  - Extremely simple fault recovery

- Proactive (pre-populated)
  - Controller pre-populates flow table in switch
  - Zero additional flow setup time
  - Loss of connection’s control connection does not disrupt traffic
  - Essentially requires aggregated rules (wildcards)
Microflow vs Aggregated flow

- **Microflow**
  - Every flow is individually set-up by controller
  - Exact-match flow entries
  - Flow table contains one entry per flow
  - Good for fine grain control, policy, and monitoring (e.g., campus)

- **Aggregated flow**
  - One flow entry covers large groups of flows
  - Wildcard flow entries
  - Flow table contains one entry per category of flows
  - Good for large number of flows (e.g., backbone)
Traffic Engineering (TE)

- Optimization of the performance of data network by dynamically analyzing, predicting and regulating the behavior of transmitted data
  - QoS enhancement
  - Packet loss and delay minimization
  - Throughput enforcement
  - Resource utilization and network congestion management
  - Overloaded and congested scenarios investigation
Segment Routing (SR)

- Segment Routing (SR): based on Source Routing
  - Enhanced packet forwarding without any topological restrictions and additional signaling requirements
  - The state of the system goes from network to single packet
  - Segments based on MPLS
  - Segments based on IPv6 (ordered list of IPv6 addresses in routing extension header)
OpenDayLight

- Cisco, Juniper, Ericsson, IBM, NEC and other network vendors are joining up to standardize SDN with OpenDayLight project (April 2013)

http://www.opendaylight.org
OpenDayLight [cont.]

Graphical User Interface, Application, and ToolKit (DLUM / NeXT UI)

AAA Authorization

OpenDaylight APIs REST/RESTCONF/NETCONF/AMQP

Base Network Functions
- Host Tracker
- L2 Switch
- OpenFlow Forwarding Rules MP
- OpenFlow State Manager
- OpenFlow Switch Manager
- Topology Processing

Enhanced Network Services
- AAA
- Centralized Accounting Data Store
- Controller Shield
- DevOps Discovery, Management & Monitoring
- DOCSIS Abstraction
- Eth Aggregation CEs Protocol
- GTP Service

SNMPv3
- Time Series Data Repository
- Unified Services Channel MP
- User Network Interface MP
- Virtual Private Network
- Virtual Tenant Network MP

Network Abstractions (Policy/Intent)
- ACI Protocol Manager
- Fabric as a Service
- Group-Based Policy Service
- NEMO
- Network Intent Composition

Controller Platform Services/Applications

Service Abstraction Layer/Core

Messaging (Notifications/RPCs)

Data Store (Config & Operational)

OpenFlow: 1.3, 1.4, 1.5, 1.6, 1.7, 1.8

OK-Config

Open vSwitches

Additional Virtual & Physical Devices

Southbound Interfaces & Protocol Plugins

Luca Davoli – davoli@ce.unipr.it
Open Network Operating System (ONOS)

- First open-source SDN network operating system targeted specifically at the Service Provider and mission critical networks

- Provides the high availability (HA), scale-out, and useful NBI and SBI APIs to enable easier development and to allow for control of OpenFlow-ready and legacy devices

- [http://onosproject.org](http://onosproject.org)
Ryu

- Is a component-based SDN framework
- Open-sourced network operating system
- Provides software components with well-defined API
- Supports various protocols for managing network devices (e.g., OpenFlow v1.0-v1.4)
- Supported by OpenStack → Thousands of virtual networks
- Provides a GUI topology viewer
- Written in Python language

https://osrg.github.io/ryu/
NOX/POX

- **NOX**
  - First OpenFlow controller
  - In 2008 it has been released to the community and it has been the basis in SDN space
  - Provides a C++ OpenFlow 1.0 API, fast, asynchronous IO
  - Includes components for topology discovery, learning switch, network-wide switch

- **POX**
  - Python-based
  - Supports the same GUI and visualization tools as NOX

- [http://www.noxrepo.org/](http://www.noxrepo.org/)
Mininet

- Provides a simple and inexpensive network testbed for developing OpenFlow applications
- Enables complex topology testing, without the need to wire up a physical network
- Includes a CLI that is topology-aware and OpenFlow-aware, for debugging purposes
- Supports arbitrary custom topologies, also providing extensible Python APIs
- Mininet networks run real code including standard Unix applications and network stack
- Developed code can move to a real system with minimal changes, for real-world deployment

http://mininet.org/

> sudo mn
SDN and Cloud Computing

- Cloud Computing service providers face the issue of multi-tenancy at the network level.
- IP and Ethernet each have virtual network capability, but limited in terms of:
  - how many tenants can be supported
  - how isolated each tenant
  - configuration and management complexity
- SDN is increasingly accepted as the path to "cloud networking"
Network Function Virtualization (NFV)

- Network entities traditionally uses dedicated hardware → now replaced with computers on which software runs to provide the same functionality

- Easier to expand and modify the network, flexibility, reduced costs

- Entire classes of network node functions can be set up as building blocks that can be connected to create overall telecommunications networks

- Traditional virtualization techniques, but now VMs running different software and providing different processes are able to provide the network functions (e.g., load balancer, firewall, IDS, routers, access control)

NFV ≠ SDN
Network Function Virtualization (NFV) [cont.]

- SDN deals with the replacement of standardized networking protocols with centralized control.

- SDN promises to reduce the complexity of distributed networking control protocols with the simplicity of programming an overall controller → improved flexibility as only a single instance needs updating to reflect a change.

- SDN separates the network control and data planes, providing a central view for more efficient implementation and running of the network services.

- NFV replaces the network hardware elements with software that runs on standard servers, focusing on optimizing the network services → decoupling of network functions from proprietary hardware.

Comparing SDN & NFV

SDN and NFV are complementary with overlapping features.

Network Functions Virtualization

Software Defined Networking

Flow Managed Network

© Greg Ferro 2015
Reti di Telecomunicazione
a.y. 2015-2016

Software-Defined Networking (SDN)
Overview

Ing. Luca Davoli
Ph.D. Student
Network Security (NetSec) Laboratory

davoli@ce.unipr.it