Exercise for Lecture Software Defined Networking

Prof. Dr. David Hausheer
Julius Rückert, Leonhard Nobach

Winter Term 2015/16
Exercise No. 2
Published at: 27.10.2015
Submission exclusively via Moodle, Deadline: 10.11.2015
Contact: `[rueckert|lnobach]@ps.tu-darmstadt.de`
Web: `http://www.ps.tu-darmstadt.de/teaching/ws1516/sdn/`
Submission: `https://moodle.tu-darmstadt.de/enrol/index.php?id=6349`

– Example Solution –

Problem 2.1 - SDN Relatives to OpenFlow

a) Name three concrete implementations/instantiations of the SDN concept, other than OpenFlow and ForCES.

Solution: Active Networks, 4D, Sane, Ethane, . . .
We highly recommend further reading (also named in the lecture): Nick Feamster, Jennifer Rexford, Ellen Zegura: The Road to SDN: An Intellectual History of Programmable Networks

b) The Forwarding and Control Element Separation (ForCES) Framework was standardized by the IETF in 2004 (RFC 3746). Please revisit the concepts presented in the lecture on ForCES by reading Section 1 and 2 of the standard document and answer the following questions. RFC 3746: https://tools.ietf.org/html/rfc3746

I) Explain the responsibilities of Control Elememts (CEs) and Forwarding Elements (FEs) in ForCES.

Solution:

- *Forwarding Element (FE)* - "A logical entity that implements the ForCES Protocol. FEs use the underlying hardware to provide per-packet processing and handling as directed by a CE via the ForCES Protocol. FEs may happen to be a single blade (or PFE), a partition of a PFE, or multiple PFEs."

- *Control Element (CE)* - "A logical entity that implements the ForCES Protocol and uses it to instruct one or more FEs on how to process packets. CEs handle functionality such as the execution of control and signaling protocols. CEs may consist of PCE partitions or whole PCEs."

Source: RFC 3746, Section 1.

II) What does the standard state as reason for the separation between FEs and CEs?
Solution:

- Aim of ForCES: “define a framework and associated protocol(s) to standardize information exchange between the control and forwarding plane”
- Reason: As a result of this standardization, the two elements can be physically separated components. This implies also most benefits that are drivers for the separation between data and control plane in OpenFlow.

Source: RFC 3746, Section 2.

Problem 2.2 - SDN and OpenFlow Basics

a) Separation of Concerns: Briefly explain in your own words the different responsibilities of the different layers of the commonly used SDN controller architecture: *SDN Control Program*, *Network Virtualization* layer, and *Network OS*.

Solution:
SDN Control Program:

- Express desired behavior without specifying the details on how to implement it on physical network structure.
- Examples: ACLs, high-level requirements on routing, QoS requirements.

Network Virtualization:

- Provide a logical network view to the control program. This view is virtual and can differ from the physical network.
- Different control programs can work on different virtual views.

Network OS:

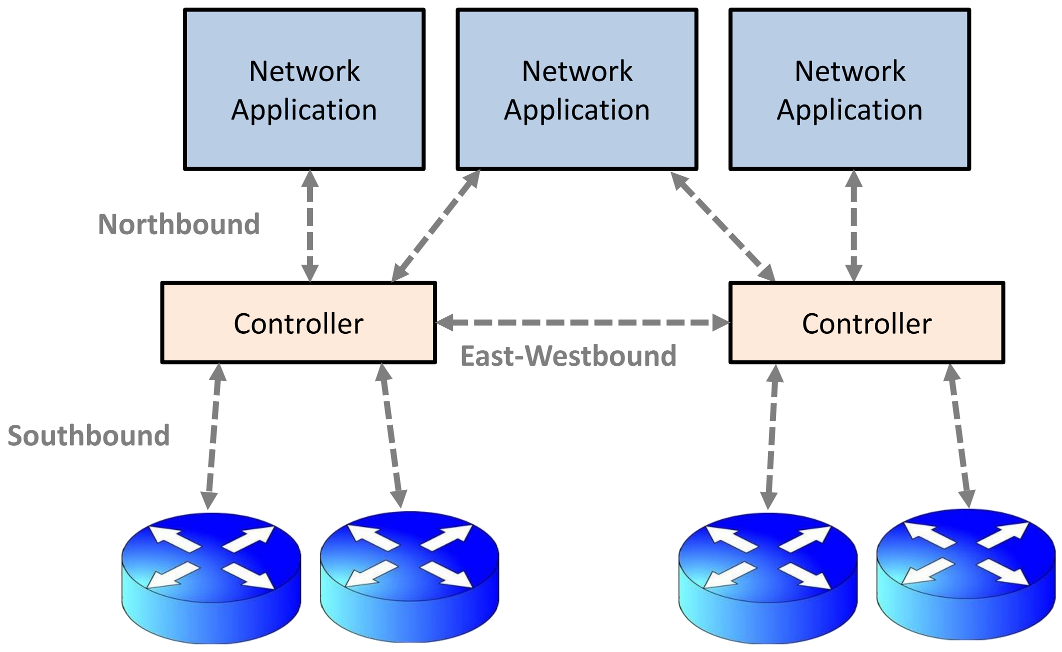
- Provide the global network view to the upper layers.
- Translate abstract behavior as specified by the control program to physical configurations on hardware boxes.

b) Explain what the “Southbound API" of an SDN controller refer to? Can you also imagine what the “Northbound” and “East-/Westbound API” could refer to?

Solution:

- Southbound API: This is the interface between the SDN controller and the physical network devices. The OpenFlow protocol is an instantiation of a southbound API. It is the protocol that is used for the configuration of the network devices as well as the signalling between the controller and the network devices.

- Northbound API: This is the interface between the controller and any type of network application running on top of it. This interface is very hard to define in a precise manner as any kind of network applications should be supported. Therefore, for now, the northbound interface is more of a high-level concept and not yet well-defined.
- East-/Westbound API: This is the interface between different controllers. Just as the northbound API, currently, the east-/westbound is more like an abstract concept, with many different possibilities to implement it. The idea is that this interface is used, e.g., for inter-domain coordination, route exchanges, etc.



c) What are implications of running the OpenFlow client as part of the software layer of a switch?

Solution: There are several implications of this design aspect. Most importantly it is to be highlighted that processing of any kind that happens at the software layer has a much lower performance than pure harware-based processing, e.g., for forwarding of packets. The performance impact can be immense whenever OpenFlow features are used that are not supported by the switch hardware and thus are to be performed in software.

d) Taking your above answer into consideration, why is it hard to realize flow matching up to ISO/OSI layer 7?

Solution: Line-rate processing of flows and thus packets requires hardware support as mentioned above. Above the “waist of the Internet protocol stack”, it becomes increasingly complex to support the large number of available protocols. Harware-supported processing up to layer 7 thus would be very costly to be realized.

Problem 2.3 - The OpenFlow Protocol

For the following questions you have to use the official OpenFlow specifications. You are not intended to read the whole documents! Get familiar with their structure and use them to look up details. Knowing how to navigate and find details within the specifications is essential for several later tasks and the lab. If we do not specify the version to be used, we assume version v1.5.0 in the following.
Important OpenFlow Specifications:

- v1.0.0: `https://goo.gl/G9UJP4`
- v1.1.0: `https://goo.gl/7JISrB`
- v1.3.0: `https://goo.gl/3WCVZz`
- v1.5.0: `https://goo.gl/VDxZN2`

Besides, we recommend the following webpage that can help to investigate differences between versions of the OpenFlow protocol etc.: `http://flowgrammable.org/sdn/openflow/`

a) Up to which ISO/OSI layer is packet matching supported in OpenFlow 1.0 without extensions?

Solution: layer 4/transport layer > Filtering based on L4 port

b) Why does OpenFlow support switches to connect to multiple controllers?

Solution: This is a feature to improve the reliability in case of connection failures. Besides, it can be used for load balancing between controllers. (cf. p.43 in OpenFlow v1.5.0 specification).

c) What is the difference between *Instructions* and *Actions* as defined by the OpenFlow standard?

Solution: "Each flow entry contains a set of instructions that are executed when a packet matches the entry. These instructions result in changes to the packet, action set and/or pipeline processing." (p.26, OpenFlow v1.5.0 specification)
Flow entries point to instructions that specify pipeline operations (required for using pipelining as introduced in OpenFlow v1.1.0) or a set of actions to be performed on the matching packets. Thus, actions are operations directly on the packets.

d) One very essential action that can be applied to packets of a flow is called *output*. What is the purpose of this action and what can the reserved port "CONTROLLER" be used for in this action's context?

Solution: The specification describes its purpose in the following manner: "output: . . . forward the packet on the port specified by the output action" (p.27, OpenFlow v1.5.0 specification)
The reserved port "CONTROLLER" can be used to send an incoming packet to the controller: "When used as an output port, encapsulate the packet in a packet-in message and send it using the OpenFlow switch protocol " (p.16, OpenFlow v1.5.0 specification)