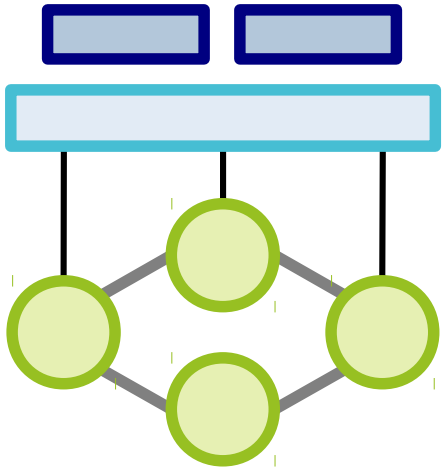
Software Defined Networking and the OpenDaylight Controller



Profile – Dr. Michael Bredel

- Studied electrical engineering at the Technische Universität Darmstadt
- PhD at the Leibniz Universität Hannover in the field of performance evaluation of computer networks
- Network research engineer at the California Institute of Technology and CERN
- Professor for Web Communications and Information Systems at the university of applied sciences in Kufstein, Austria
- Senior researcher at NEC Labs Heidelberg, Germany

Educational Objectives

- Know the basic principles of Software Defined Networking
- Know about the OpenFlow protocol and its primitives
  - Know about flow matching
  - Know about OpenFlow actions
- Know some tools to experience SDN
- Know about the OpenDaylight controller

Outline

- Presentation
  - The evolution of servers and networks
  - Software Defined Networking
  - The OpenFlow protocol
  - The OpenDaylight SDN controller
- Hands-On Exercises
  - Installing, configuring, and running OpenDaylight
  - Network emulation with MiniNet
  - ODL features
    - L2Switch
    - Virtual Tenant Network

Evolution of Computing and Networking

- Evolution of servers

1990

2014

Chef

puppet labs

vmware

ownCloud

openstack CLOUD SOFTWARE

- Evolution of networking – the control plane

1990

```
R1>?
Exec commands:
  access-enable     Create a temporary Access-List entry
  access-profile    Apply user-profile to interface
  call              Voice call
  clear             Reset functions
  connect           Open a terminal connection
  crypto            Encryption related commands.
  disable           Turn off privileged commands
  disconnect        Disconnect an existing network connection
  enable            Turn on privileged commands
  exit              Exit from the EXEC
  help              Description of the interactive help system
  lock              Lock the terminal
  login             Log in as a particular user
  logout            Exit from the EXEC
  modemui           Start a modem-like user interface
  mrinfo            Request neighbor and version information from a multicast
                    router
  mstat             Show statistics after multiple multicast traceroutes
  mtrace            Trace reverse multicast path from destination to source
  name-connection   Name an existing network connection
  pad               Open a X.29 PAD connection
  ping              Send echo messages
  ppp               Start IETF Point-to-Point Protocol (PPP)
  release           Release a resource
  renew             Renew a resource
  resume            Resume an active network connection
  rlogin            Open an rlogin connection
  set               Set system parameter (not config)
  show              Show running system information
  slip              Start Serial-line IP (SLIP)
  ssh               Open a secure shell client connection
  systat            Display information about terminal lines
  tclquit           Quit Tool Command Language shell
  telnet            Open a telnet connection
  terminal          Set terminal line parameters
  traceroute        Trace route to destination
  tunnel            Open a tunnel connection
```

Telnet

2014

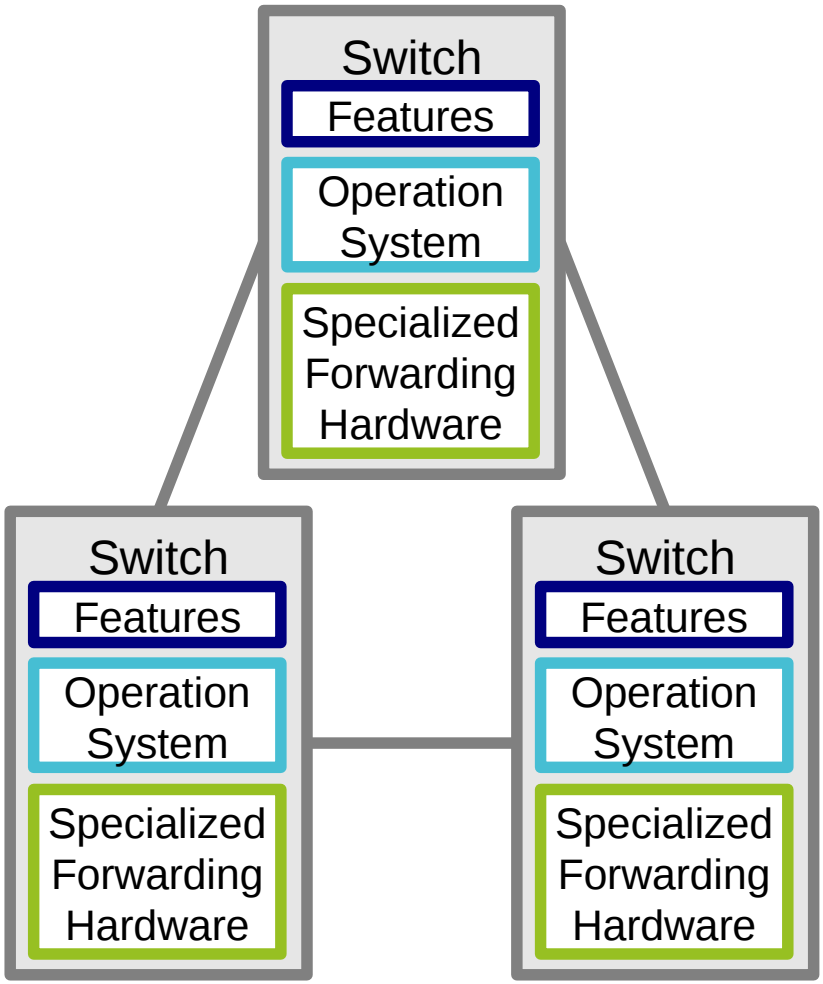
```
R1>?
Exec commands:
  access-enable     Create a temporary Access-List entry
  access-profile    Apply user-profile to interface
  call              Voice call
  clear             Reset functions
  connect           Open a terminal connection
  crypto            Encryption related commands.
  disable           Turn off privileged commands
  disconnect        Disconnect an existing network connection
  enable            Turn on privileged commands
  exit              Exit from the EXEC
  help              Description of the interactive help system
  lock              Lock the terminal
  login             Log in as a particular user
  logout            Exit from the EXEC
  modemui           Start a modem-like user interface
  mrinfo            Request neighbor and version information from a multicast
                    router
  mstat             Show statistics after multiple multicast traceroutes
  mtrace            Trace reverse multicast path from destination to source
  name-connection   Name an existing network connection
  pad               Open a X.29 PAD connection
  ping              Send echo messages
  ppp               Start IETF Point-to-Point Protocol (PPP)
  release           Release a resource
  renew             Renew a resource
  resume            Resume an active network connection
  rlogin            Open an rlogin connection
  set               Set system parameter (not config)
  show              Show running system information
  slip              Start Serial-line IP (SLIP)
  ssh               Open a secure shell client connection
  systat            Display information about terminal lines
  tclquit           Quit Tool Command Language shell
  telnet            Open a telnet connection
  terminal          Set terminal line parameters
  traceroute        Trace route to destination
  tunnel            Open a tunnel connection
```

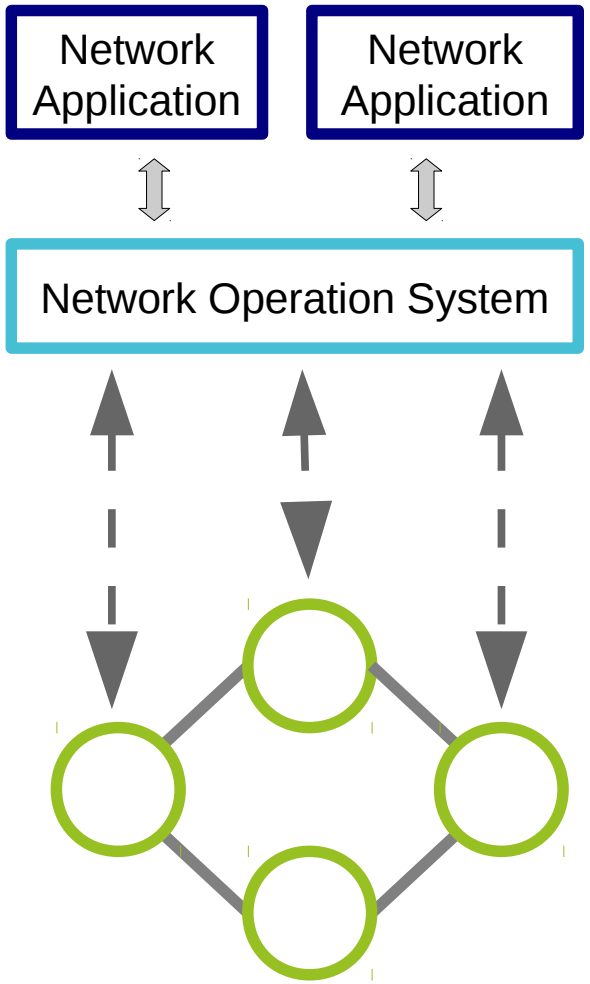
SSH

Problems due to the Limited Control Plane

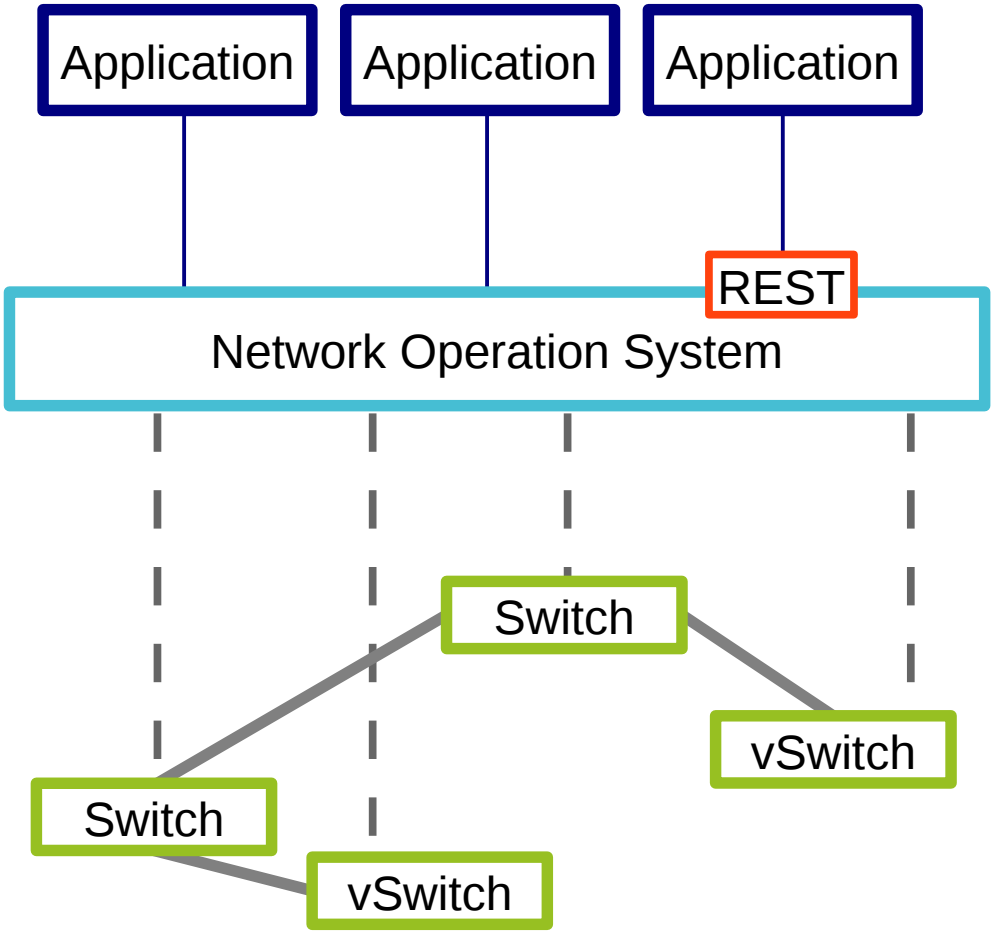
- Management interfaces
  - How to manage a large number of switches and routers?
  - How does it scale?
  - How do you guarantee persistence, e.g. in ACL's?
  - How do you guarantee traffic separation?
- In-band traffic control
  - How to optimize traffic flows globally?
  - How to use multiple paths?
    - How to get rid of Spanning Tree Protocol?
  - How to minimize convergence time?
  - How to obtain deterministic behavior?

Software Defined Networking





Software Defined Networking



1 **Open Source**
Integration layer

2 **Open APIs**
to program the network

3 **Open Standards**
such as OpenFlow

Software Defined Networking

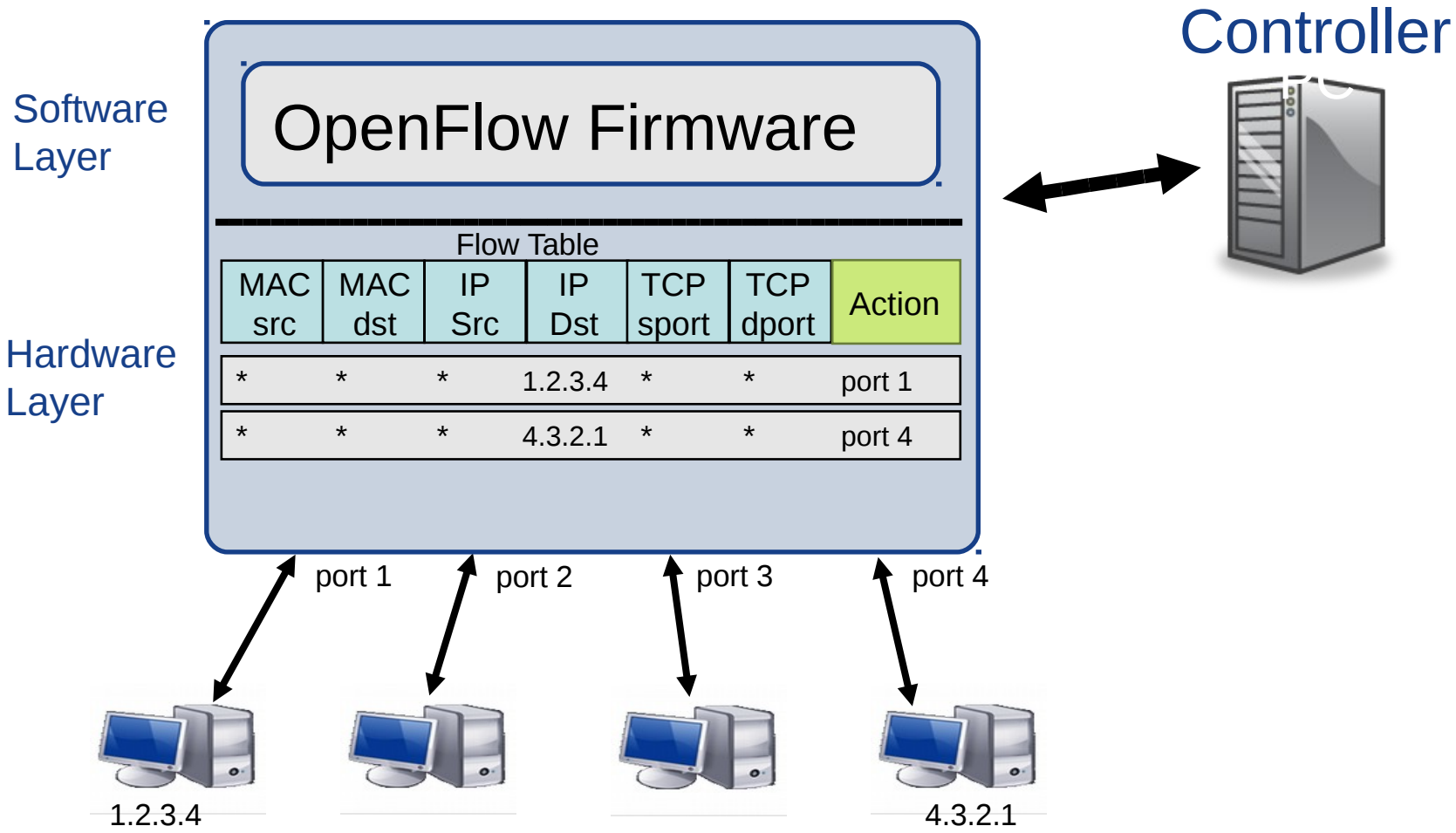
- An approach to computer networking that allows for managing network services through abstractions of lower level functionality

- Decoupling of
  - Data (or forwarding) plane
    - Typically hardware (a circuit) that forwards packets from an input port to the respective output port at line rate
  - Control plane
    - The logic, typically software, that controls the packet forwarding

OpenFlow Protocol

- OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over a control interface
  - Control how packets are forwarded
  - Implementable on COTS hardware
  - Make deployed networks programmable
- Maintained by the Open Networking Foundation (ONF)

OpenFlow Protocol

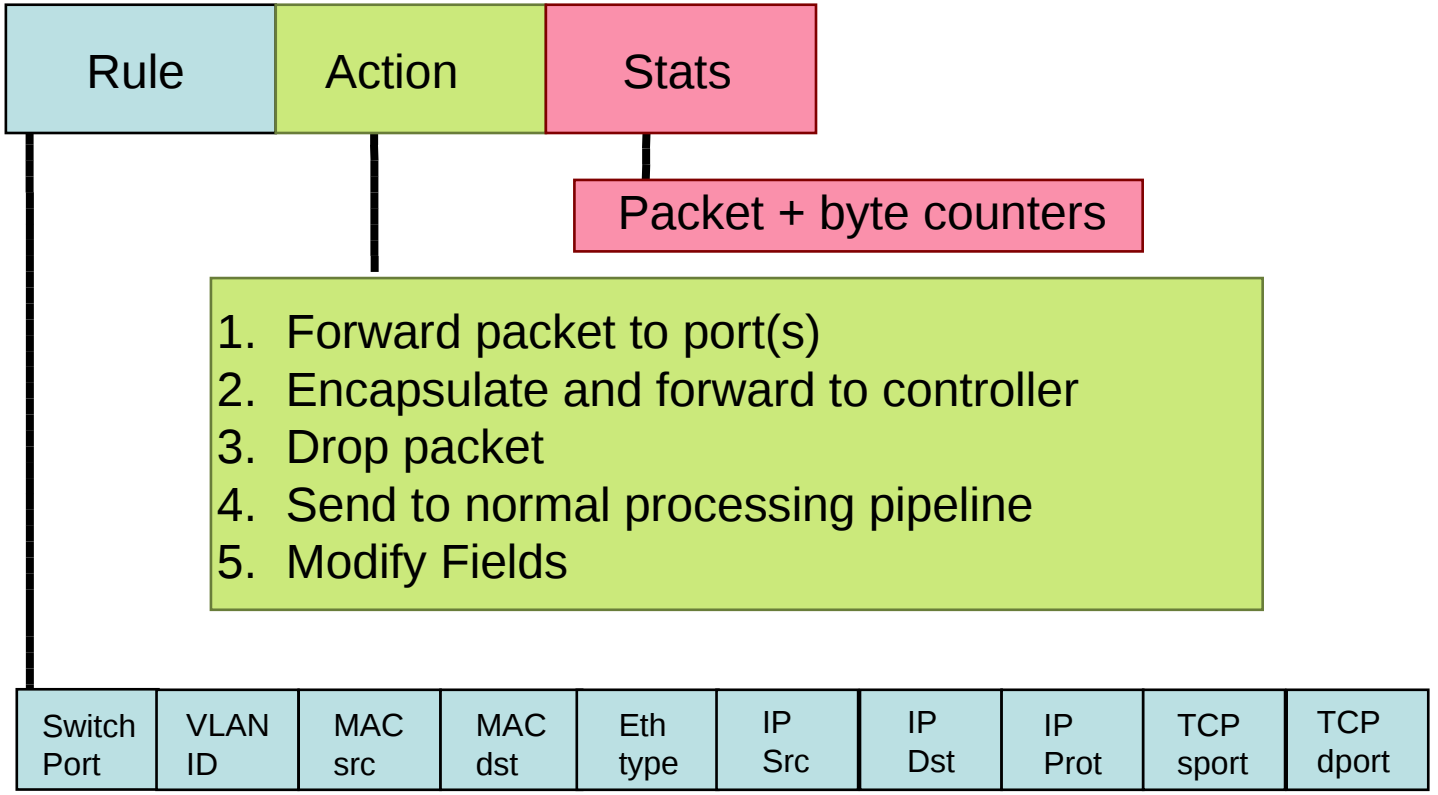


OpenFlow Protocol

- How the protocol works (in reactive mode)
  - On packet arrival, match the header fields with flow entries in a table
  - If no entry matches
    - Send the packet to the OpenFlow SDN controller
  - If any entry matches
    - Update the counters indicated in that entry
    - Perform indicated actions
  - Idle timeout
    - Remove entry if no packets received for this time
  - Hard timeout
    - Remove entry after this time
  - If both are set, the entry is removed if either one expires

OpenFlow Protocol

- OpenFlow flow table entries



+ mask what fields to match

OpenFlow Protocol

- Flow table example

Port	DL_SRC	DL_DST	DL_VLAN	Priority	ETHER_TYPE	NW_SRC	NW_DST	NW_PROTO	NW_ToS	TP_SRC	TP_DST	Action	Counter
*	*	0a:c8:*	*	*	*	*	*	*	*	*	*	port 1	401
*	*	*	*	*	IP	*	192.168.*.*	*	*	*	*	port 2	306
*	*	*	*	*	*	*	*	*	*	21	21	drop	200
*	*	*	*	*	*	*	*	0x806	*	*	*	local	444
*	*	*	*	*	*	*	*	0x1*	*	*	*	controller	1

- Flow descriptions should be in normal form
  - A flow may only specify a value for an L3 field if it also specifies a particular L2 protocol
    - For example, if the L2 protocol type dl_type is wildcarded, then L3 fields nw_src, nw_dst, and nw_proto must also be wildcarded

OpenFlow Protocol

- OpenFlow actions
  - Forward to physical port or to virtual port
    - All: To all interfaces except incoming interface
    - Controller: Encapsulate and send to controller
    - Local: Send to its local networking stack
    - Table: Perform actions in the flow table
    - In_Port: Send back to input port
    - Normal: Forward using traditional Ethernet
    - Flood: Send along minimum spanning tree except the incoming interface
  - Drop
  - Modify Field
    - Add or remove VLAN tags, ToS bits
    - Change TTL
    - Change L2 and L3 addresses

OpenFlow Versions

- OpenFlow 1.0
  - As explained above
  - Perform action on a match
  - Ethernet/IP only

- OpenFlow 1.1
  - Introduced table chaining and group tables
    - Group Tables: each entry has a variable number of buckets
      - All: Execute each bucket. Used for broadcast and multicast
      - Indirect: Execute one *predefined* bucket
      - Fast Failover : Execute the first live bucket →live port
  - Added MPLS label and MPLS traffic class to match fields
  - Added support for Q-in-Q, tunnels, and multipath

OpenFlow Versions

- OpenFlow 1.2
  - Added IPv6 support
    - Matching fields include IPv6 source address, destination address, protocol number, traffic class. ICMPv6 type, ICMPv6 code, IPv6 neighbor discovery header fields, and IPv6 flow labels
  - Extensible Matches
    - Type-Length-Value (TLV) structure
    - Previously the order and length of match fields was fixed
  - Experimenter extensions
    - through dedicated fields and code points assigned by ONF

OpenFlow Versions

- OpenFlow 1.3
  - Per-Connection Event Filtering: Better filtering of connections to multiple controllers
  - Cookies: A cookie field is added to messages containing new packets sent to the controller
    - This helps controller process the messages faster than if it had to search its entire database
  - Meter: Switch element that can measure and control the rate of packets/bytes
    - Meters are attached to a flow entry not to a queue or a port
    - Meter Band: If the packet/byte rate exceeds a pre-defined threshold the meter has triggered the band
    - A meter may have multiple bands
    - If on triggering a band the meter drops the packet, it is called rate limiter

OpenFlow Deployment Models

- Reactive vs. proactive flow control

Reactive

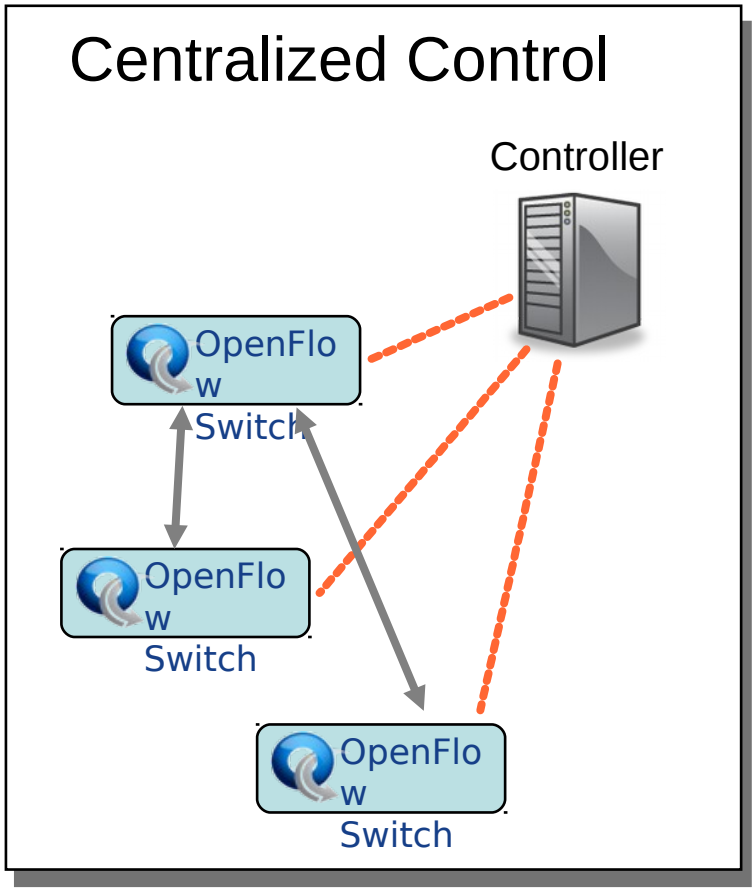
- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

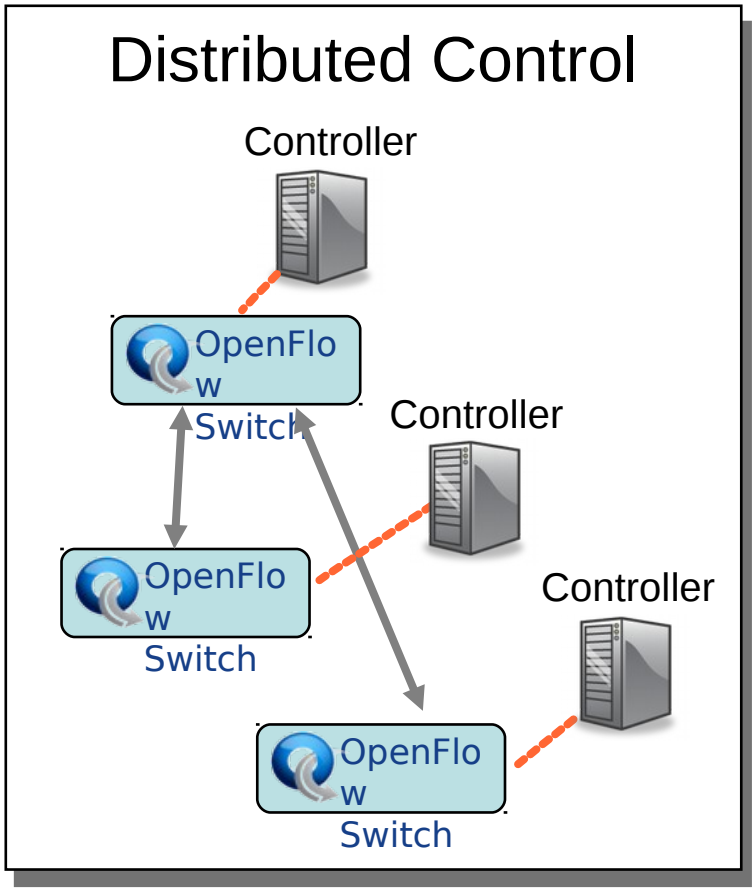
Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules

OpenFlow Network Design

- Centralized vs. decentralized control





Bootstrapping OpenFlow Networks

- Switches require initial configuration
  - Switch IP address
  - Controller IP address
  - Default gateway
- Switches connect to the controller
- Switch provides configuration information about ports
- Controller installs a rule to forward LLDP responses to controller and then sends a LLDP request which is forwarded to all neighbors
- Controller determines the topology from LLDP responses

OpenDaylight SDN Controller

- OpenDaylight (ODL)
  - An open platform for network programmability meant to enable SDN and NFV for networks at any size and scale
  - OpenDaylight mainly consists of software designed to be run on top of a Java Virtual Machine (JVM) and can be run on any operating system and hardware as there is a Java Runtime Environment (JRE) available for it
  - Open Source project under The Linux Foundation, funded April 8, 2013

OpenDaylight History

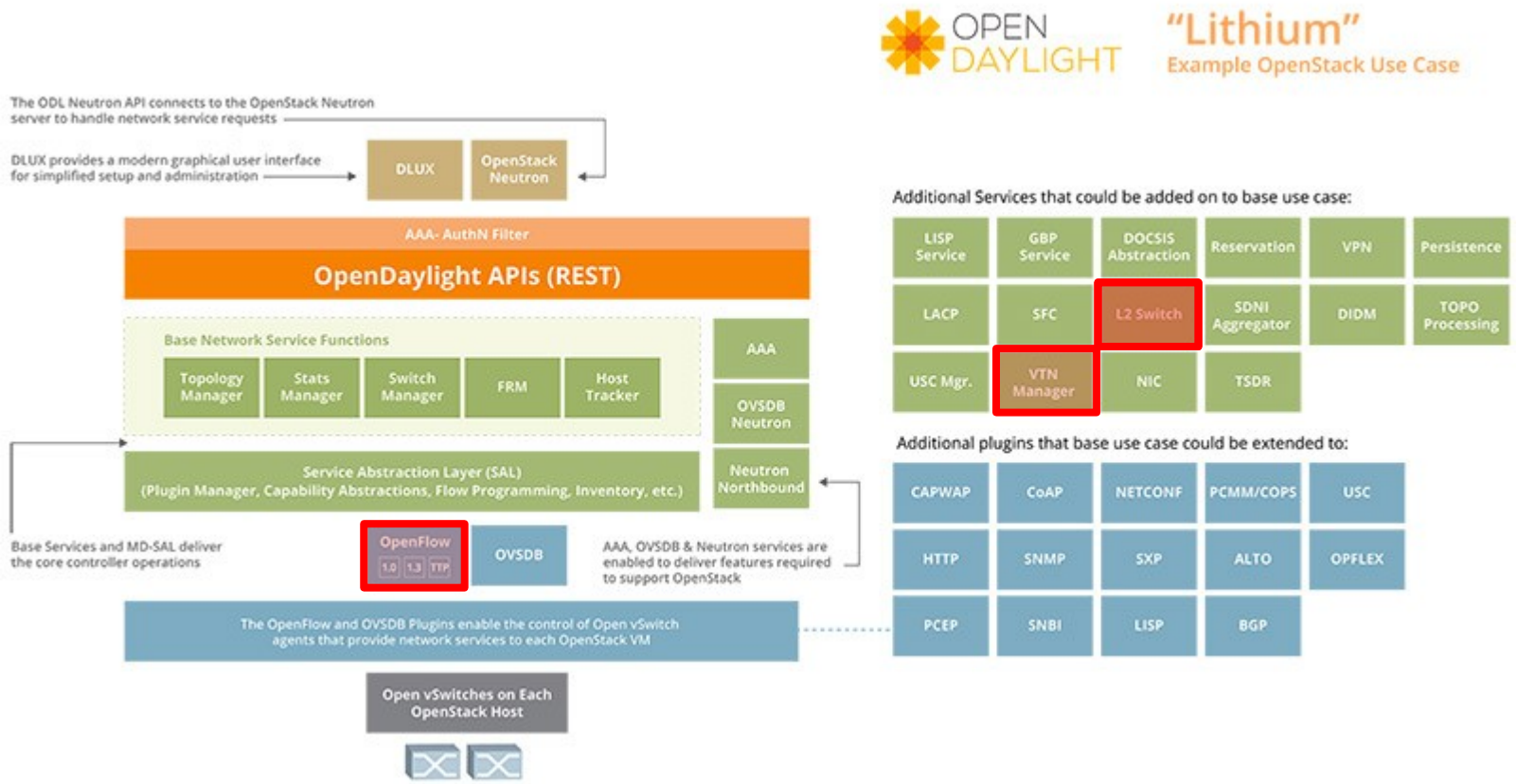
- The community operates around a time-based release cycle of roughly every six months with frequent development milestones
- The naming convention for each OpenDaylight release follows the atomic number of elements in the periodic table
- Releases
  - Hydrogen, February 2014
  - Helium, October 2014
  - Lithium, June 2015
  - Beryllium, February 2016 (Planned)

OpenDaylight History

- The community operates around a time-based release cycle of roughly every six months with frequent development milestones
- The naming convention for each OpenDaylight release follows the atomic number of elements in the periodic table
- Releases
  - Hydrogen, February 2014
  - Helium, October 2014
  - Lithium, June 2015
  - Beryllium, February 2016 (Planned)

OpenDaylight Architecture (Lithium)

- ODL Lithium



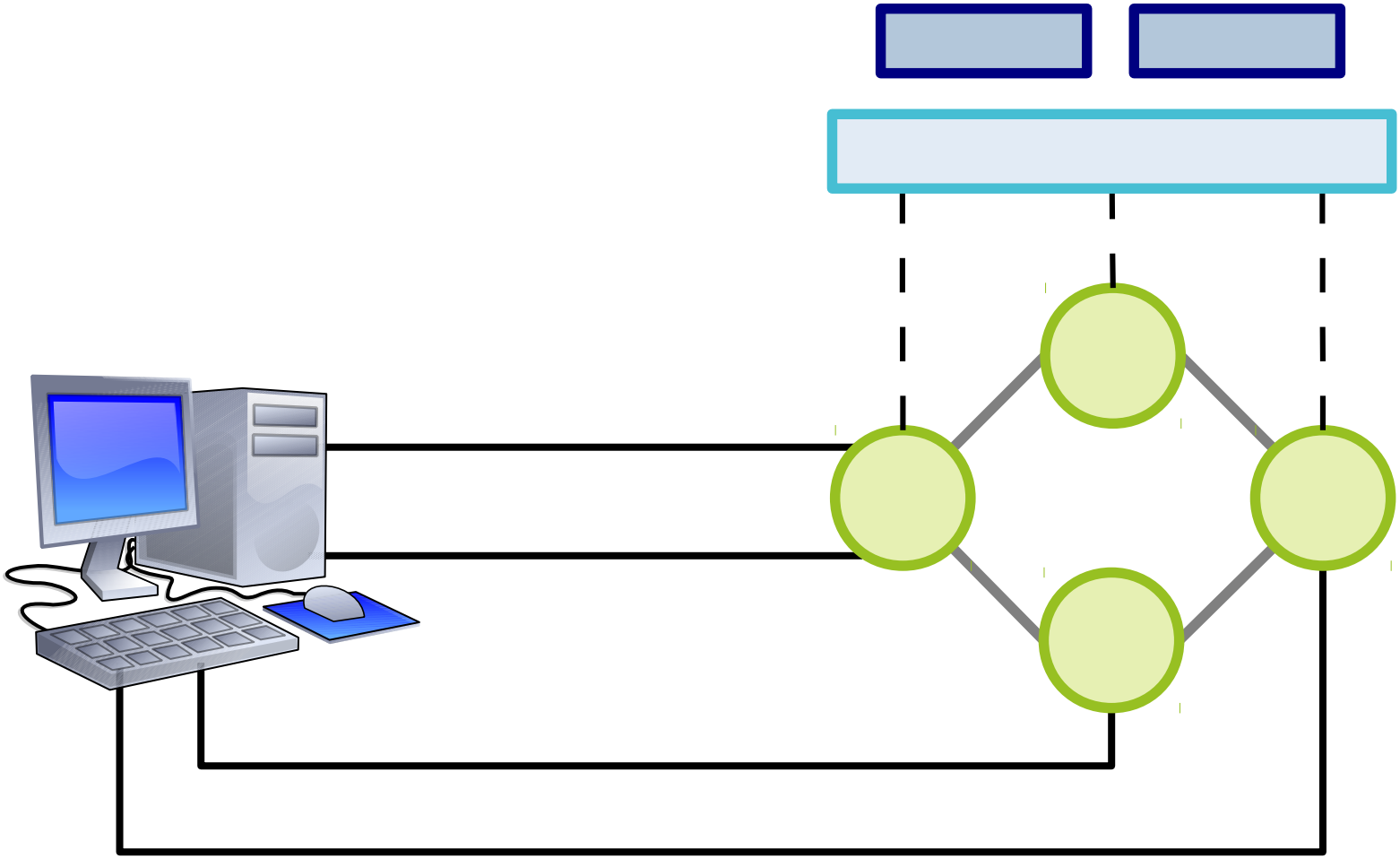
OpenDaylight Features (Lithium)

- AAA
  - Standards-compliant Authentication, Authorization and Accounting Services
  - Support for persistent data stores, Federation and SSO with OpenStack Keystone
- CAPWAP
  - Theh CAPWAP plugin project enables the Opendaylight Controller to manage CAPWAP compliant wireless termination point network devices.
  - Intelligent applications (e.g. radio planning, etc) can be developed by tapping into the operational states, made available via REST APIs, of WTP network devices.

OpenDaylight Features (Lithium)

- DLUX (openDayLight User eXperience)
  - DLUX provides a modern and intuitive graphical user interface for OpenDaylight based on the AngularJS framework
- LACP
  - The LACP Project within OpenDaylight implements Link Aggregation Control Protocol (LACP) as an MD-SAL service module and will be used to auto-discover and aggregate multiple links between an OpenDaylight controlled network and LACP-enabled endpoints or switches

SDN LACP



OpenDaylight Features (Lithium)

- Time Series Data Repository
  - The TSDR project in OpenDaylight creates a framework for collecting, storing, querying, and maintaining time series data in then OpenDaylight SDN controller
  - With the capabilities of data collection, storage, query, aggregation, and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering, and network analytics with automated intelligence

OpenDaylight Features (Lithium)

- Virtual Tenant Network
  - VTN is an application that provides multi-tenant virtual network on an SDN controller
  - VTN allows the users to define the network with a look and feel of conventional L2/L3 network
  - Once the network is designed on VTN, it will automatically be mapped into underlying physical network, and then configured on the individual switch leveraging SDN control protocol.
  - The definition of logical plane makes it possible not only to hide the complexity of the underlying network but also to better manage network resources
  - It achieves reducing reconfiguration time of network services and minimizing network configuration errors.

Hands-on Exercises

Outline

- Log in to your VM
- Install and run OpenDaylight
  - Start and Stop OpenDaylight
  - Install basic features
- Install and run MiniNet to emulate an OpenFlow network
- Use basic OpenDaylight features to operate the OpenFlow network
  - L2-Switch
  - Virtual Tenant Network

The Student VM's

- Instance: SDN-1
  - address: 141.52.228.133
  - pass: ~dwZ58teD
- Instance: SDN-2
  - address: 141.52.229.252
  - pass: 4mDCm4ty(
- Instance: SDN-3
  - address: 141.52.229.253
  - pass: 0sVow2lO@

Installing OpenDaylight

- Requirements for running OpenDaylight
  - Java 7+ installation (we use Java 8)
  - Network connectivity

- Download OpenDaylight
  - http://www.opendaylight.org/software/downloads
  - The Karaf distribution has no features enabled by default. However, all of the features are available to be installed
    - For compatibility reasons, you cannot enable all the features simultaneously

Running OpenDaylight

- To run the Karaf distribution
  - Unpack the .zip or .tar.gz file
  - Navigate to the directory
  - run ./bin/karaf
  - Be patient !!!

- Example (for Linux)

```
$ tar -zxvf distribution-karaf-0.3.1-Lithium.tar.gz
distribution-karaf-0.3.1-Lithium-SR1/configuration/
distribution-karaf-0.3.1-Lithium-SR1/data/
...
distribution-karaf-0.3.1-Lithium-SR1/bin/stop.bat
$ cd distribution-karaf-0.3.1-Lithium-SR1
$ ./bin/karaf
```

Configure OpenDaylight

- Configuration can be done by some config files
  - ./etc/...
  - ./configuration
- logback.xml
  - Logging in ODL is done by Logback
  - By default logging messages are appended to
    - stdout of the java process
    - File data/log/karaf.log
  - When debugging a problem it might be useful to increase logging level

```
<logger name="org.opendaylight.controller" level="DEBUG"/>
```

Running OpenDaylight

- Once, OpenDaylight has been startet …
  - Press tab for a list of available commands
  - Typing [cmd] --help will show help for a specific command
  - Press ctrl-d or type system:shutdown or logout to shutdown OpenDaylight

- OpenDaylight starts with no feature installed by default
- To list the installed features type

```
$ feature:list -i
```

Installing Components

- To install a feature use the following command in the Karaf console

```
$ feature:install
```

- For Example

```
$ feature:install <feature-name>
```

- For a list of available features

```
$ feature:list
```

Installing Components

- OpenDaylight starts with no feature installed by default

- For a start, install the following features
  - feature:install odl-base-all
  - feature:install odl-aaa-authn
  - feature:install odl-restconf
  - feature:install odl-dlux-core odl-dlux-node

  - feature:install odl-openflowplugin-all
  - feature:install odl-l2switch-all odl-l2switch-switch-ui
  - feature:install odl-vtn-manager odl-vtn-manager-rest

Postman REST Client

- *Postman* is an alternative to *curl*
- Google Chrome App Postman
  - Postman is a generic REST client that runs in the browser
  - Easy to use (compare, e.g. to curl)
  - https://www.getpostman.com
    - Available at the Chrome Web Store

OpenDaylight – RestConf for Inventory

- The REST URL for listing all the nodes

  ```
  http://{ODL_IP}:8181/restconf/operational/opendaylight-inventory:nodes/
  ```

- You will need to set the Accept header

  ```
  Accept: application/xml
  ```

- You will also need to use HTTP Basic Auth with

  ```
  Username: admin, Password: admin
  ```

- Alternately, if you have a node's id

  ```
  http://{ODL_IP}:8181/restconf/operational/opendaylight-inventory:nodes/node/<id>
  ```

  ```
  http://{ODL_IP}:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:1
  ```

MiniNet

- Mininet creates a virtual network, running real kernel, switch and application code, on a single machine with a single command:

```
$ sudo mn
```

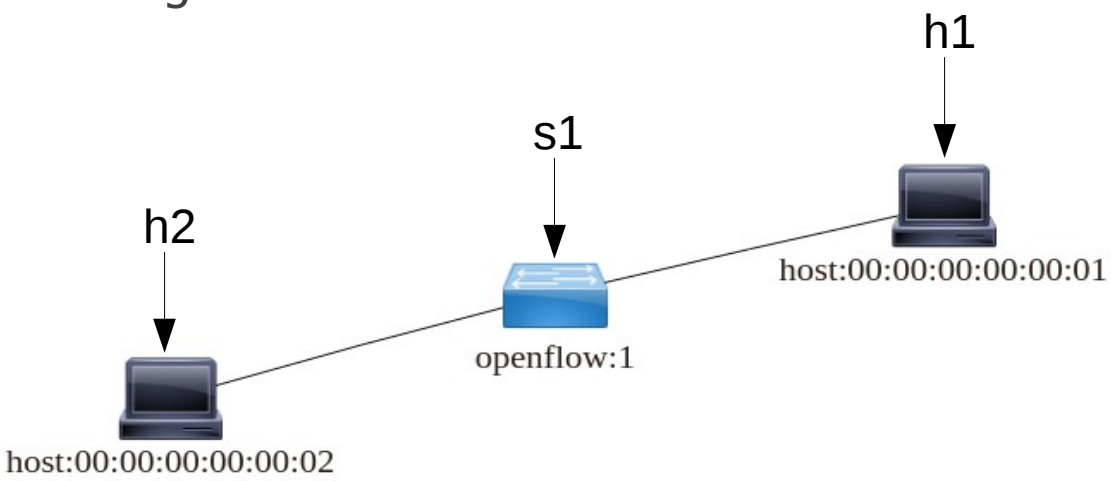
- Creating topologies
  - tree, linear, single, minimal, torus

- Show OpenFlow flows on a switch (s1)

```
$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1
```

- Install the following features
  - feature:install odl-openflowplugin-all
- To create a very simple topology

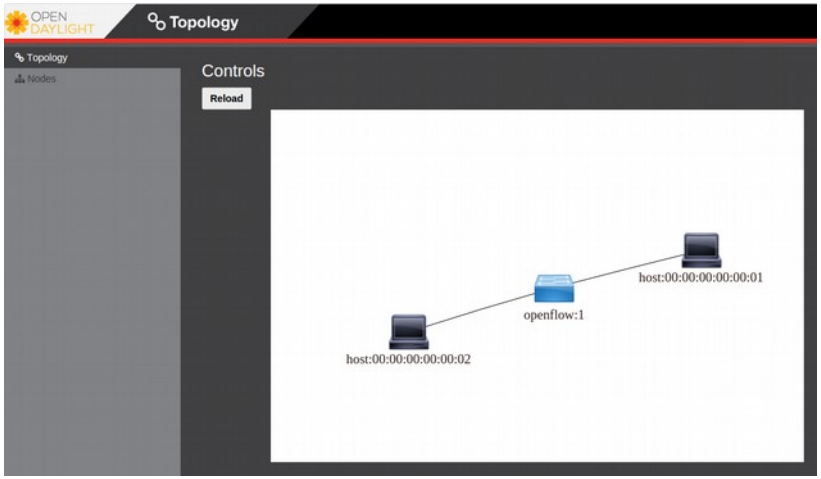
```
$ sudo mn
 --mac --switch=ovsk,protocols=OpenFlow13
 --controller=remote,ip=${ODL_IP},port=6653
 --topo=single
```



DLUX – openDayLight User eXperience

- Install the following features
  - feature:install odl-dlux-core
  - feature:install odl-dlux-node
- Have a look at
  - http://${ODL_IP}:8181/index.html
- Try to ping between the hosts
  - In MiniNet type:

```
$ h1 ping h2
```



OpenDaylight – RestConf to Push Flows

- The REST URL to push a flow
  - Operation: PUT
  - URI:

    ```
    http://${ODL_IP}:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/1
    ```

- You will need to set the Accept header

  ```
  Accept: application/xml
  ```

- Content-type

  ```
  application/xml
  ```

- Body (have a look at the wiki)

  ```
  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <flow xmlns="urn:opendaylight:flow:inventory">
  ...
  </flow>
  ```

OpenDaylight – RestConf to Push Flows

Institut für Kommunikations-Technik

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <output-action>
                        <output-node-connector>1</output-node-connector>
                        <max-length>60</max-length>
                    </output-action>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <table_id>0</table_id>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
    <match>
        <in-port>0</in-port>
    </match>
    <hard-timeout>12</hard-timeout>
    <cookie>4</cookie>
    <idle-timeout>34</idle-timeout>
    <flow-name>P0_TO_P1</flow-name>
    <priority>2</priority>
    <barrier>false</barrier>
</flow>
```

OpenDaylight – RestConf to Remove Flows

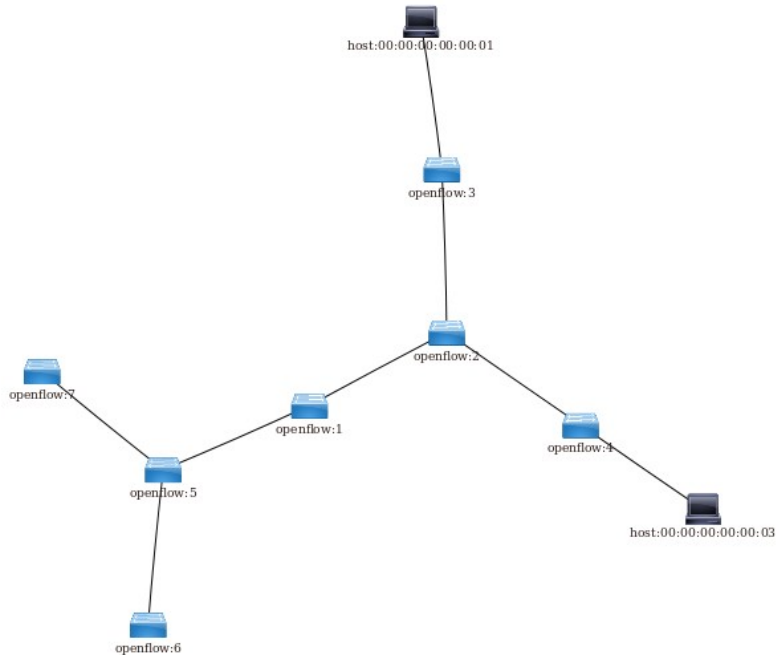
- The REST URL to remove a flow
  - Operation: DELETE
  - URI:

```
http://${ODL_IP}:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:1/table/0/flow/1
```

- To create a more complex topology

```
$ sudo mn
  --mac --switch=ovsk,protocols=OpenFlow13
  --controller=remote,ip=${ODL_IP},port=6653
  --topo=tree,3
```

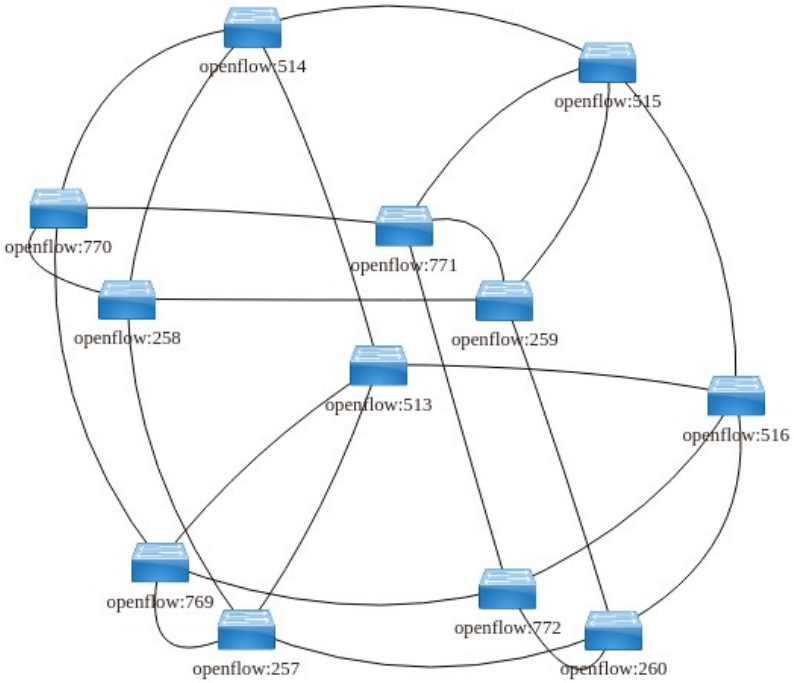
creates a three layered topology of 7 switches and two attached hosts



L2Switch

- Install the following feature
  - feature:install odl-l2switch-all
  - feature:install odl-l2switch-switch-ui

- Components of the L2Switch
  - Packet Handler
  - Loop Remover
  - Arp Handler
  - Address Tracker
  - Host Tracker
  - L2Switch Main (Flow Installer)



L2Switch – Configuration

- L2Switch can be configured using the file
  - ./etc/opendaylight/karaf/52-loopremover.xml
  - ./etc/opendaylight/karaf/54-arphandler.xml
  - ./etc/opendaylight/karaf/56-addresstracker.xml
  - ./etc/opendaylight/karaf/58-l2switchmain.xml

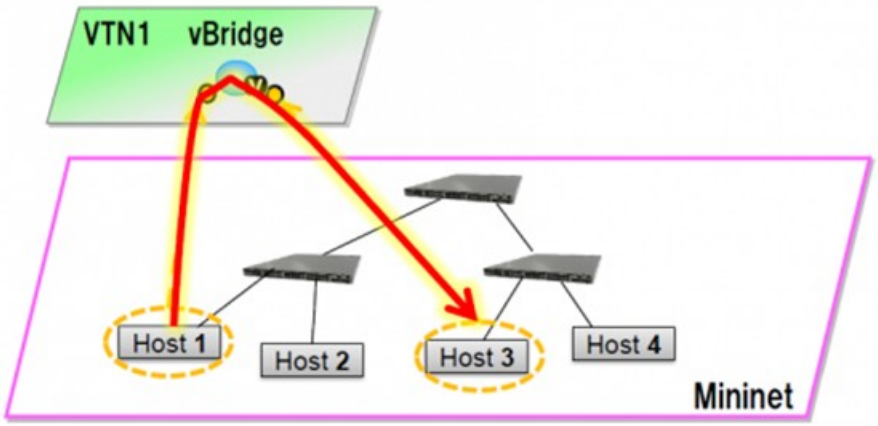
- Ping between two hosts

```
$ h1 ping h2
```

- Verify the flows on the switch (in MiniNet)
  - `$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1`

Virtual Tenant Network

- Virtual Tenant Network service provides multi-tenant virtual network on an SDN controller
- It enables the complete separation of logical plane from physical plane
- It is implemented as two major components
  - VTN Manager
  - VTN Coordinator



Virtual Tenant Network – Installation

- *VTN Manager* is a set of OSGI bundles running in the OpenDaylight Controller
  - Install the following features
    - feature:install odl-vtn-manager
    - feature:install odl-vtn-manager-rest
    - feature:install odl-vtn-manager-neutron
- *VTN Coordinator* is an application running outside the controller

```
$ cd ./externalapps
$ sudo tar -C/ -jxvf distribution.vtn-
coordinator-6.1.2-SNAPSHOT-bin.tar.bz2
```

Virtual Tenant Network – VTN Coordinator

- Configuring database for VTN Coordinator

```
$ sudo /usr/local/vtn/sbin/db_setup
```

- Start the VTN Coordinator

```
$ sudo /usr/local/vtn/bin/vtn_start
```

- Stop the VTN Coordinator

```
$ sudo /usr/local/vtn/bin/vtn_stop
```

- NOTE! For security reason, VTN Coordinator does not run if installation directory is group or world writable

```
$ sudo chmod go-w /usr/local/vtn /usr/local /usr
$ sudo chown -R root:root /usr/local/vtn
```

Virtual Tenant Network – VTN Coordinator

- Check the VTN Coordinator

```
$ curl --user admin:adminpass -H 'content-type:
application/json' -X GET
http://<VTN_COORDINATOR_IP_ADDRESS>:8083/vtn-
webapi/api_version.json
```

The result should look like this:

```
{"api_version":{"version":"V1.2"}}
```