Lab :: ONOS SDN Controller

Basic Intro & Hands-On Tutorial

2S/2019

Prof. Christian Rothenberg (FEEC/UNICAMP)

Agenda

- ONOS Introduction
- Exercise 1: CLI commands/Reactive Forwarding
- Exercise 2: ONOS GUI
- Exercise 3: Intent Reactive Forwarding
- (Optional) Exercise 4: SDN-IP

ONOS Introduction

What is ONOS?

- **ONOS** stands for **O**pen **N**etwork **O**perating **S**ystem. ONOS provides the control plane for a software-defined network (SDN), managing network components, such as switches and links, and running software programs or modules to provide *communication services* to end hosts and neighboring networks.

- The most important benefit of an operating system is that it **provides a useful and usable platform for software programs** designed for a particular application or use case. ONOS applications and use cases often consist of customized communication routing, management, or monitoring services for software-defined networks.

- ONOS can run as a **distributed system across multiple servers**, allowing it to use the CPU and memory resources of multiple servers while providing fault tolerance in the face of server failure and potentially supporting live/rolling upgrades of hardware and software without interrupting network traffic.

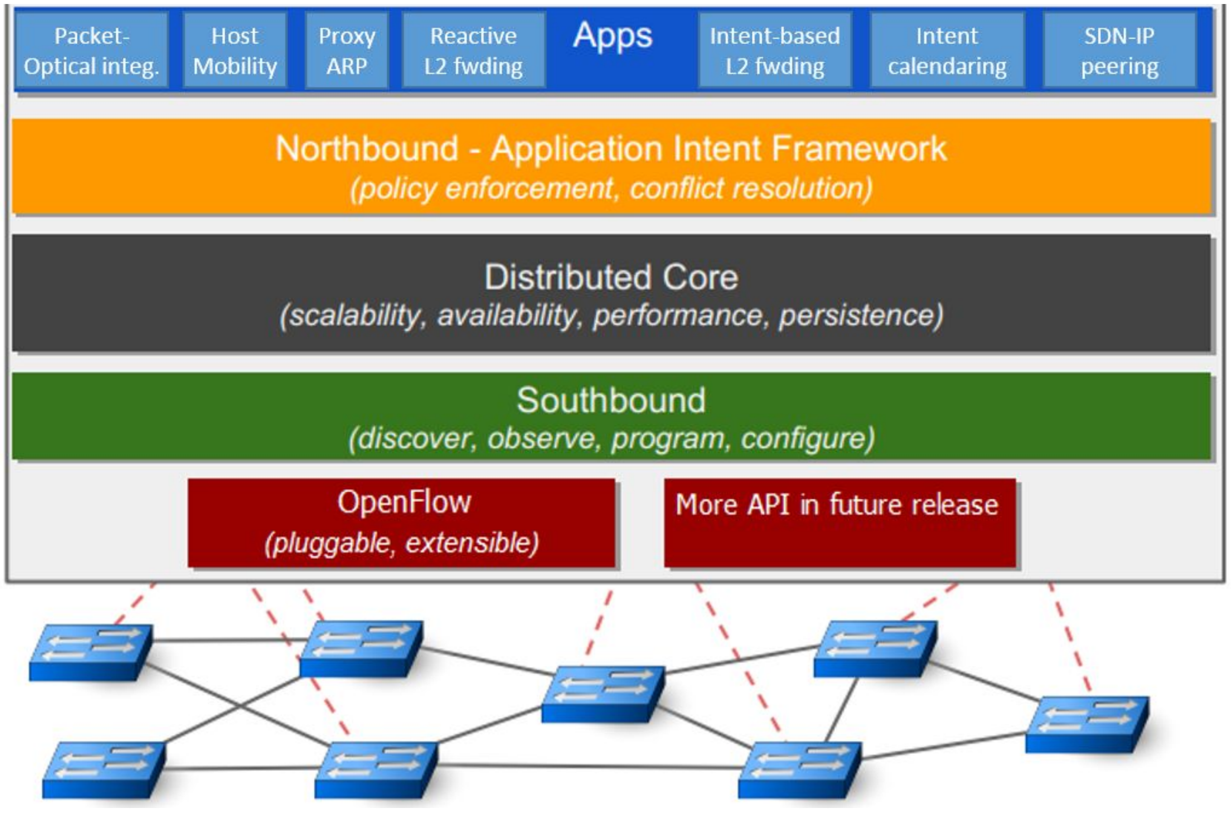
- The ONOS kernel and core services, as well as ONOS applications, are written in Java as bundles that are loaded into the Karaf OSGi container.

- ONOS is an **open source project** backed by an expanding community of developers and users

https://wiki.onosproject.org/display/ONOS/Wiki+Home

Architectural Tenets

- High availability, scalability and performance
- Strong abstractions and simplicity
- Protocol and device behavior independence
- Separation of concerns and modularity



Packet-Optical integ.
Host Mobility
Proxy ARP
Reactive L2 fwding
Apps
Intent-based L2 fwding
Intent calendaring
SDN-IP peering
Northbound - Application Intent Framework
(policy enforcement, conflict resolution)
Distributed Core
(scalability, availability, performance, persistence)
Southbound
(discover, observe, program, configure)
OpenFlow
(pluggable, extensible)
More API in future release

https://wiki.onosproject.org/display/ONOS/System+Components

Hands-on

Exercise 1:

- CLI commands
- Reactive Forwarding

Set up the environment (DONE!)

# Pre-requisites

- You will need a computer with at least 2GB of RAM and at least 5GB of free hard disk space. A faster processor or solid-state drive will speed up the virtual machine boot time, and a larger screen will help to manage multiple terminal windows.
- The computer can run Windows, Mac OS X, or Linux – all work fine with VirtualBox, the only software requirement.
- To install VirtualBox, you will need administrative access to the machine.
- The tutorial instructions require prior knowledge of SDN in general, and OpenFlow and Mininet in particular.

# Install required software

- You will need to acquire two files: a
    - VirtualBox: https://www.virtualbox.org/
    - Tutorial VM: https://intrig.dca.fee.unicamp.br:8840/index.php/s/ilUsYFLcVcPmOMv (psw: **onos**)

# Create Virtual Machine

- Import the OVA file (Lab6-ONOS.ova)
- When the import is finished start the VM and log in using:
    - USERNAME: **tutorial1**
    - PASSWORD: **tutorial1**

Máquinas do Laboratório

As máquinas (computadores) do laboratório devem ser acessadas com as contas fornecidas para cada aluno. Caso a máquina não inicie corretamente, peça auxílio ao responsáveis pelo laboratório.

Para iniciar a VM utilizada nessa prática, abra um terminal e execute o comando a seguir.

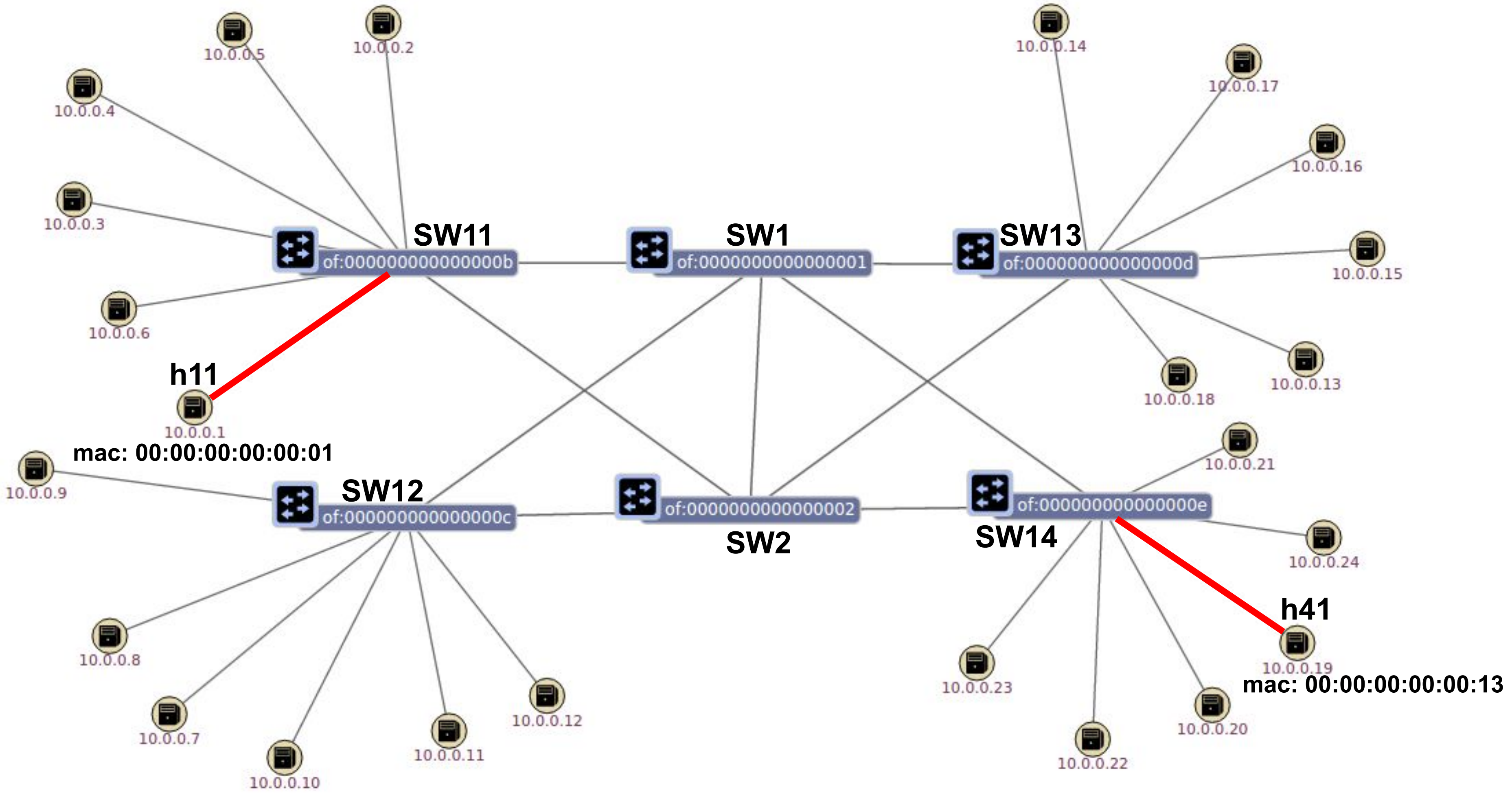
```
$ vm-inf556-lab7-ONOS
```

Para acessar a VM, utilize as seguintes credenciais:

USERNAME: tutorial1
PASSWORD: tutorial1

Reactive Forwarding

- Double-click the **“Reset”** icon on the desktop now. This will ensure the environment is clean and will start up ONOS.
- To start mininet with this topology, simply double click on the **“Mininet”** icon on your desktop.



Reactive Forwarding

# Start ONOS

- Start by opening a console onto ONOS, you can do this by double clicking on the **"ONOS"** icon.

# we are going to run a sample app shipped with ONOS. Reactive Forwarding is a simple application which installs flows for every packet in that arrives at the controller.

- Go to your mininet prompt and do the following:

```
mininet> h11 ping -c5 h41
```

Q1: the ping fails? Why?

Tip: Try to list the loaded applications

```
onos> apps -a -s
```

Reactive Forwarding

# In your ONOS window, do

- onos> ***app activate org.onosproject.fwd***

# Then, in a mininet window run the ping again, just this time don't limit the number of pings.

```
mininet> h11 ping h41
PING 10.0.0.19 (10.0.0.19) 56(84) bytes of data.
64 bytes from 10.0.0.19: icmp_req=1 ttl=64 time=9.12 ms
64 bytes from 10.0.0.19: icmp_req=2 ttl=64 time=0.892 ms
64 bytes from 10.0.0.19: icmp_req=3 ttl=64 time=0.075 ms
64 bytes from 10.0.0.19: icmp_req=4 ttl=64 time=0.068 ms
```

# Start stop start stop

- You have now seen that you can load applications into ONOS dynamically. Actually you can also interrupt applications while they are running so, for example, let's stop the reactive forwarding application.
  - onos> ***app deactivate org.onosproject.fwd***
- Observe that the ping has now stopped. This is because when the reactive forwarding application is unloaded, it cleans up after itself by removing the rules that it has pushed.
- For now, let's restart the reactive forwarding application.
  - onos> ***app activate org.onosproject.fwd***

ONOS CLI commands

# ONOS has many CLI commands. We will go through some of the most useful commands. While we will explain some of the ONOS CLI commands here you can find an exhaustive list by running:

- onos> ***help onos***
- More information about an individual command adding ***--help*** to any command. Also most commands have autocompletion to help you find the parameters quickly and easily.

# Devices command

- An SDN Controller would be nothing without devices to control. ONOS has a convenient command to list the device currently known in the system. Running:
  - onos> ***devices***
  - It will return information which consists of a device id, and a boolean value which indicates whether this devices is currently up. You also get the type of device and well as it's role relationship with this ONOS instance.

```
id=of:0000000000000001, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10
id=of:0000000000000002, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10
id=of:000000000000000b, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10
id=of:000000000000000c, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10
id=of:000000000000000d, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10
id=of:000000000000000e, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.1.3, serial=None, protocol=OF_10
```

ONOS CLI commands

# Links command

- The links command is used to list the links detected by ONOS. At the ONOS prompt run:
    - onos> *links*

```
src=of:000000000000000e/1, dst=of:0000000000000001/5, type=DIRECT, state=ACTIVE
src=of:000000000000000d/1, dst=of:0000000000000001/4, type=DIRECT, state=ACTIVE
src=of:000000000000000e/2, dst=of:0000000000000002/5, type=DIRECT, state=ACTIVE
src=of:000000000000000c/1, dst=of:0000000000000001/3, type=DIRECT, state=ACTIVE
src=of:000000000000000d/2, dst=of:0000000000000002/4, type=DIRECT, state …
...
```

- The output shows you the list of discovered links. Reported links are formatted by source device-port pair to destination device-port pair. The 'type' field indicates whether the link is a direct connection between two devices or not.

# Hosts command

- ONOS has the ability to list the hosts currently in the system:
    - onos> ***hosts***

```
id=00:00:00:00:00:01/-1, mac=00:00:00:00:00:01, location=of:000000000000000b/3, vlan=-1, ip(s)=[10.0.0.1]
id=00:00:00:00:00:13/-1, mac=00:00:00:00:00:13, location=of:000000000000000e/3, vlan=-1, ip(s)=[10.0.0.19]
```

    - It will display the hosts' id as well as its mac address and where in the network it is connected. The '-1' in the id field is used to display the vlan information, in this case there is no vlan.

ONOS CLI commands

# Flows command

- The flows command allows you to observe which flow entries are currently registered in the system. So let's start some traffic by going to the mininet window and running
    - `mininet>` ***h11 ping h41***
- Then in the ONOS window let's run the flows command
    - `onos>` ***flows***

```
        deviceId=of:0000000000000001, flowRuleCount=1
  id=30000b889cb32, state=ADDED, bytes=8722, packets=89, duration=89, priority=10, appId=org.onlab.onos.fwd
    selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13},
IN_PORT{port=2}]
    treatment=[OUTPUT{port=5}]
        deviceId=of:0000000000000002, flowRuleCount=1
  id=30000b889cf4d, state=ADDED, bytes=8624, packets=88, duration=88, priority=10, appId=org.onlab.onos.fwd
    selector=[ETH_TYPE{ethType=800}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01},
IN_PORT{port=5}]
    treatment=[OUTPUT{port=2}] ...
```

- As you can see from the above output, ONOS provides many details about he the flows at the switches.
    - Each flow entry defines a **selector** and **treatment** which is the set of traffic matched by the the flow entry and how this traffic should be handled.
    - Notice as well that each flow entry it tagged by an appId (application id), this appId identifies which application installed this flow entry. This is a useful feature because it can help an admin identify which application may be misbehaving or consuming many resources.

ONOS CLI commands

# Paths command

- Given a network topology, ONOS computes all the shortest paths between any two nodes.
- This is especially useful for your applications to obtain path information for either flow installation or some other use.
- The paths command takes two arguments, both of them are devices.
- To make things easy for you ONOS provides CLI autocompletion by simply hitting the <*TAB*> key.

```
onos> paths <TAB>

of:0000000000000001    of:0000000000000002    of:000000000000000b

of:000000000000000c    of:000000000000000d    of:000000000000000e
```

- ONOS lists device options for you, thereby making it easier to find the devices you would like. For example, the output of the command below shows two paths of equal costs.

```
onos> paths of:000000000000000b of:000000000000000e

of:000000000000000b/1-of:0000000000000001/2==>of:0000000000000001/5-of:000000000000000e/1; cost=2.0
of:000000000000000b/2-of:0000000000000002/2==>of:0000000000000002/5-of:000000000000000e/2; cost=2.0
```

Exercise 2:
ONOS GUI

ONOS Graphical User Interface

# Links command

- ONOS comes with a GUI. The GUI allows you to manipulate your network in a simple way.
- To open the UI simply click on the **“ONOS GUI”** icon on the desktop.
- Let's make all the hosts appear in the UI, we can do this by making the hosts talk on the network. The best way to do this is to run the pingall command at mininet.

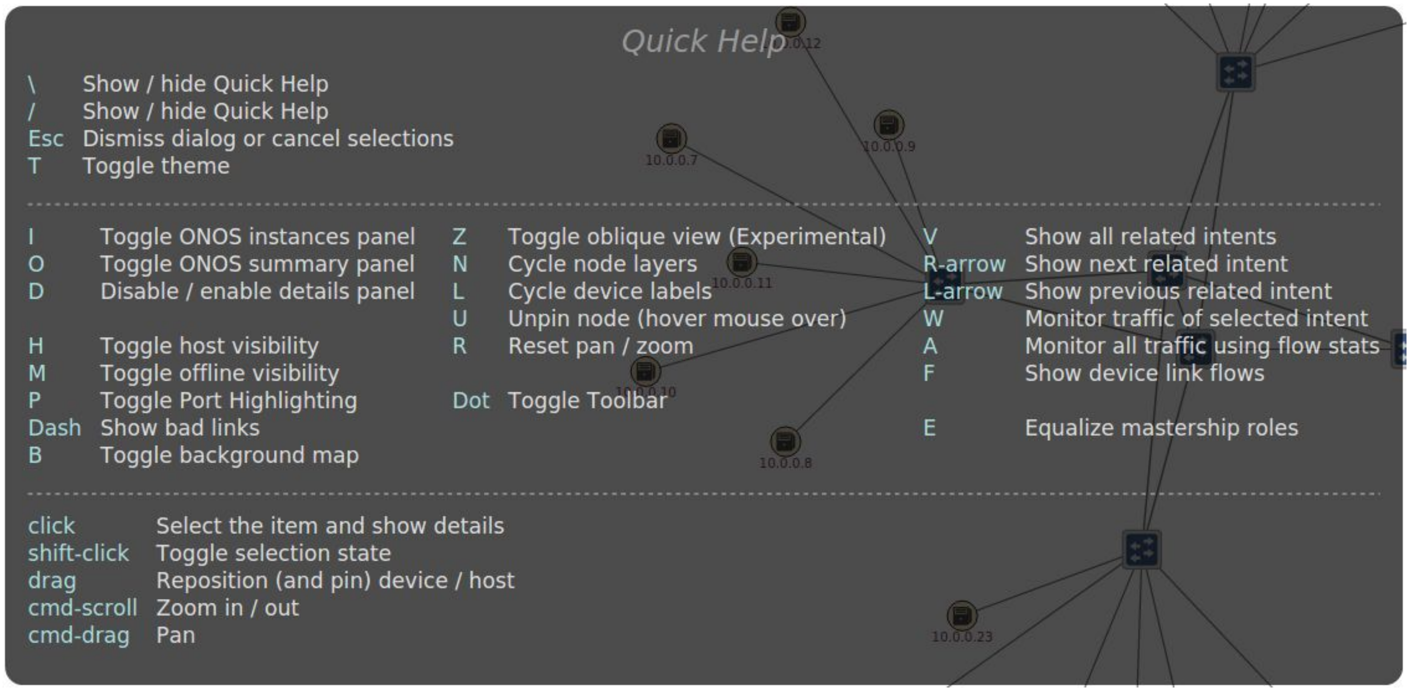
```
mininet> pingall
*** Ping: testing ping reachability
h11 -> h12 h13 h14 h15 h16 h21 h22 h23 h24 h25 h26 h31 h32 h33 h34 h35 h36 h41 h42 h43 h44 h45 h46
h12 -> h11 h13 h14 h15 h16 h21 h22 h23 h24 h25 h26 h31 h32 h33 h34 h35 h36 h41 h42 h43 h44 h45 h46
h13 -> h11 h12 h14 h15 h16 h21 h22 h23 h24 h25 h26 h31 h32 h33 h34 h35 h36 h41 h42 h43 h44 h45 h46
...
```

- The hosts will not appear initially, simply type ***'h'*** in your browser window and they will appear. At this point you should see something roughly similar to the image in the slide 9.

ONOS Graphical User Interface

# GUI Features

- At anytime you can pull up the GUI's cheat sheet by typing *'/'* and you will get a pane that looks like below.



Quick Help

\	Show / hide Quick Help
/	Show / hide Quick Help
Esc	Dismiss dialog or cancel selections
T	Toggle theme

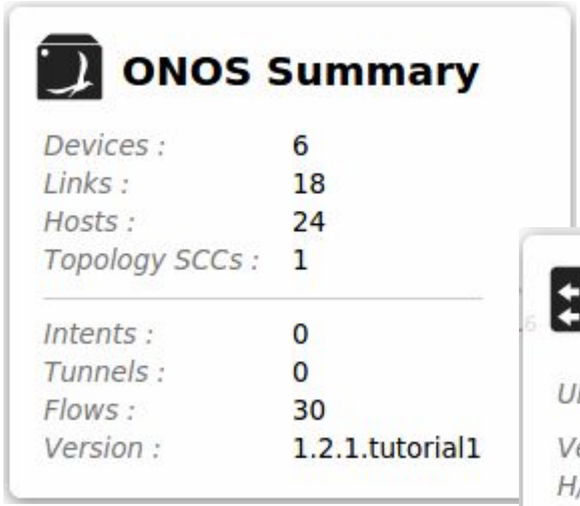
I	Toggle ONOS instances panel	Z	Toggle oblique view (Experimental)	V	Show all related intents
O	Toggle ONOS summary panel	N	Cycle node layers	R-arrow	Show next related intent
D	Disable / enable details panel	L	Cycle device labels	L-arrow	Show previous related intent
		U	Unpin node (hover mouse over)	W	Monitor traffic of selected intent
H	Toggle host visibility	R	Reset pan / zoom	A	Monitor all traffic using flow stats
M	Toggle offline visibility			F	Show device link flows
P	Toggle Port Highlighting	Dot	Toggle Toolbar		
Dash	Show bad links			E	Equalize mastership roles
B	Toggle background map				

click	Select the item and show details
shift-click	Toggle selection state
drag	Reposition (and pin) device / host
cmd-scroll	Zoom in / out
cmd-drag	Pan

ONOS Graphical User Interface

# Summary pane

- The GUI comes with a very useful summary pane. It shows you a summary of what is going on at this ONOS cluster.

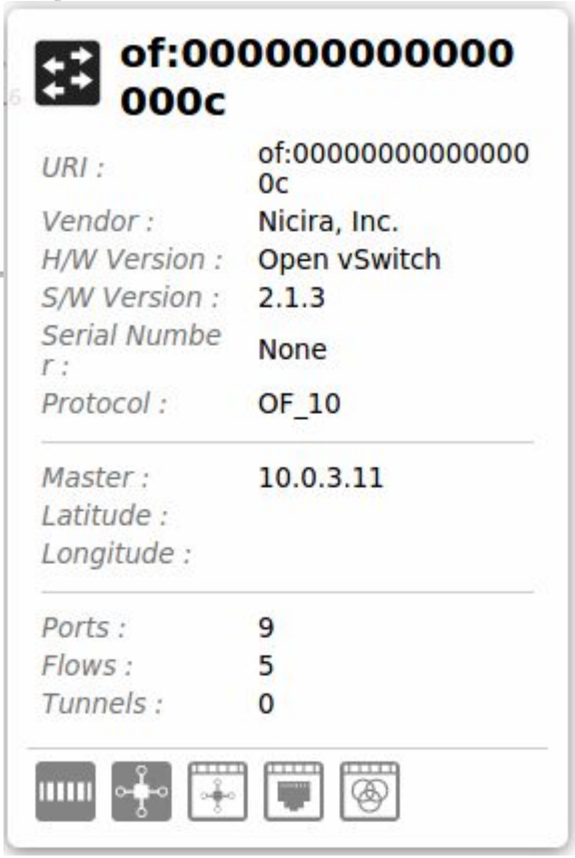


ONOS Summary

Devices :	6
Links :	18
Hosts :	24
Topology SCCs :	1
Intents :	0
Tunnels :	0
Flows :	30
Version :	1.2.1.tutorial1

# Switch details

- When you click on a switch a pane appears on the right hand side. This pane gives information about the switch as shown in the image below.



of:000000000000000c

URI :	of:000000000000000c
Vendor :	Nicira, Inc.
H/W Version :	Open vSwitch
S/W Version :	2.1.3
Serial Number :	None
Protocol :	OF_10
Master :	10.0.3.11
Latitude :	
Longitude :	
Ports :	9
Flows :	5
Tunnels :	0

NOTE: In the next exercise, we will continue to explore the ONOS GUI options.

Exercise 3:
Intent Reactive Forwarding

Introduction

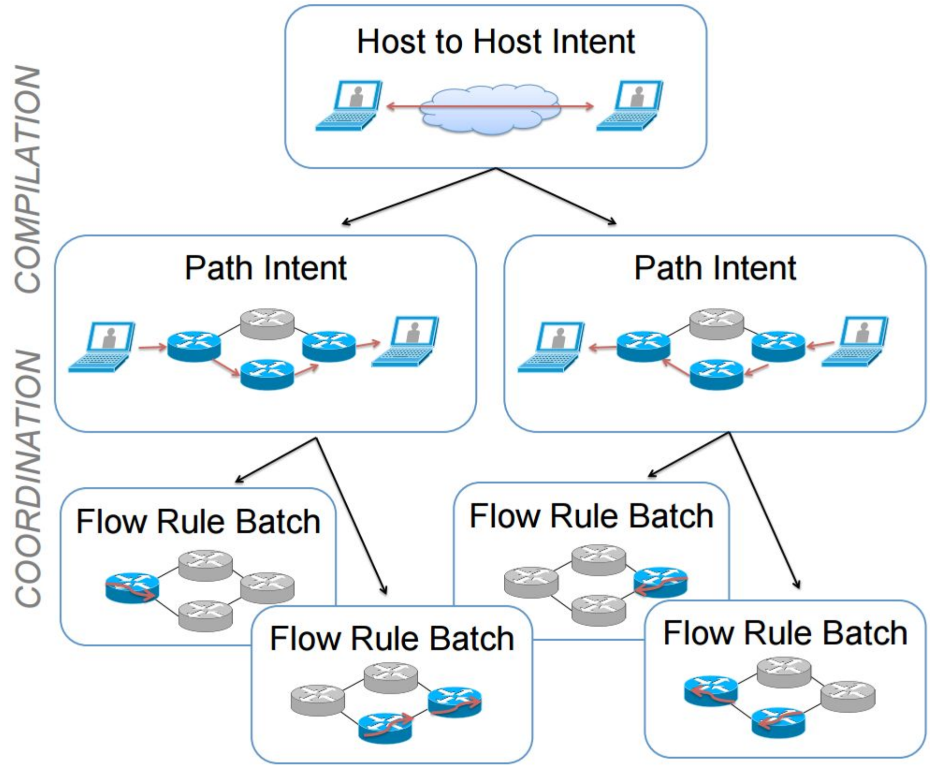
- **INTENTS**
  - Provide a flexible, high-level interface for declaring global network policy
  - Allow applications and operators to **describe policy** rather than **device instructions**
  - Abstract network complexity from higher layers
  - Extend easily to produce more complex functionality through combinations of other intents

- **ONOS Intent Framework**
  - Translates intents into device instructions (in accordance with network state and policy)
  - Coordinates and ensures installation of instructions
  - Reacts to changing network conditions
  - Permits optimization across intents
  - Extends dynamically to add or change functionality (compilers or coordinators)

https://wiki.onosproject.org/display/ONOS/Intent+Framework

http://onosproject.org/wp-content/uploads/2014/11/ONOS-Intent-Framework.pdf

Intent Compilation and Coordination



- **Intent Compilers**
  - Translate intents into other intents that are more specific to the network environment
  - (Recursively) Produce Intents that are installable into the network

- **Intent Coordinators**
  - Determine how the network must be programmed at a device/resource level
  - Reserve resources as required
  - Plan the order of installation of device instructions

http://onosproject.org/wp-content/uploads/2014/11/ONOS-Intent-Framework.pdf

Intent Reactive Forwarding

A sample application in ONOS is the **INTENT REACTIVE FORWARDING APPLICATION**. Rather than pushing flow entries for each packet it sees, the intent reactive forwarding application provisions an intent. In particular, it provisions a host to host intent which is a simple connectivity intent which enables the connectivity between two hosts.

# First let's start by removing the old reactive forwarding application and load the intent reactive forwarding application.

```
onos> app deactivate org.onosproject.fwd
onos> app activate org.onosproject.ifwd
```

- Notice the different bundle names: onos-app-fwd vs. onos-app-**ifwd**

# let's just make sure it is loaded correctly:

```
onos> apps -s -a
*   5 org.onosproject.drivers    1.2.1 Builtin device drivers
*  10 org.onosproject.ifwd       1.2.1 Reactive forwarding application using intent service (experimental)
*  30 org.onosproject.openflow   1.2.1 OpenFlow protocol southbound providers
```

- So we can see that the intent forwarding application is correctly active. The appId for the Reactive Forwarding application remains, so if you reload that application it will get the same appId it had previously.

Intentionally React

# Let's forward some traffic.

```
mininet> h11 ping -c3 h41
PING 10.0.0.19 (10.0.0.19) 56(84) bytes of data.
64 bytes from 10.0.0.19: icmp_req=1 ttl=64 time=9.12 ms
64 bytes from 10.0.0.19: icmp_req=2 ttl=64 time=0.892 ms
64 bytes from 10.0.0.19: icmp_req=3 ttl=64 time=0.075 ms
--- 10.0.0.19 ping statistics ---
```

# So by using the flows command you can list the flows (by running the flows command) that the intent installed. So how is this different than the other application? Well the end result is the same but the process by which it was obtained it radically different. The intent reactive forwarding application has installed an intent between h11 and h41, as you can see by running the intents command:

```
onos> intents
```

or in detail:

```
onos> intents -i
```

- So the intent forwarding app has pushed the intent. The intent is a host to host intent which details the path along which the flows have been installed. If you would like to know more about intents or the intent framework in general have a look at this ***page***.

Intentionally React

# Before we continue, let's remove the intent you just installed so that it doesn't get in the way of the intent gymnastics we are going to do in the next section.

```
onos> remove-intent org.onosproject.ifwd <TAB>
onos> purge-intents
```

# The <tab> key will autocomplete the id that was associated the intent we just pushed. You should now see that there are no intents in the system:

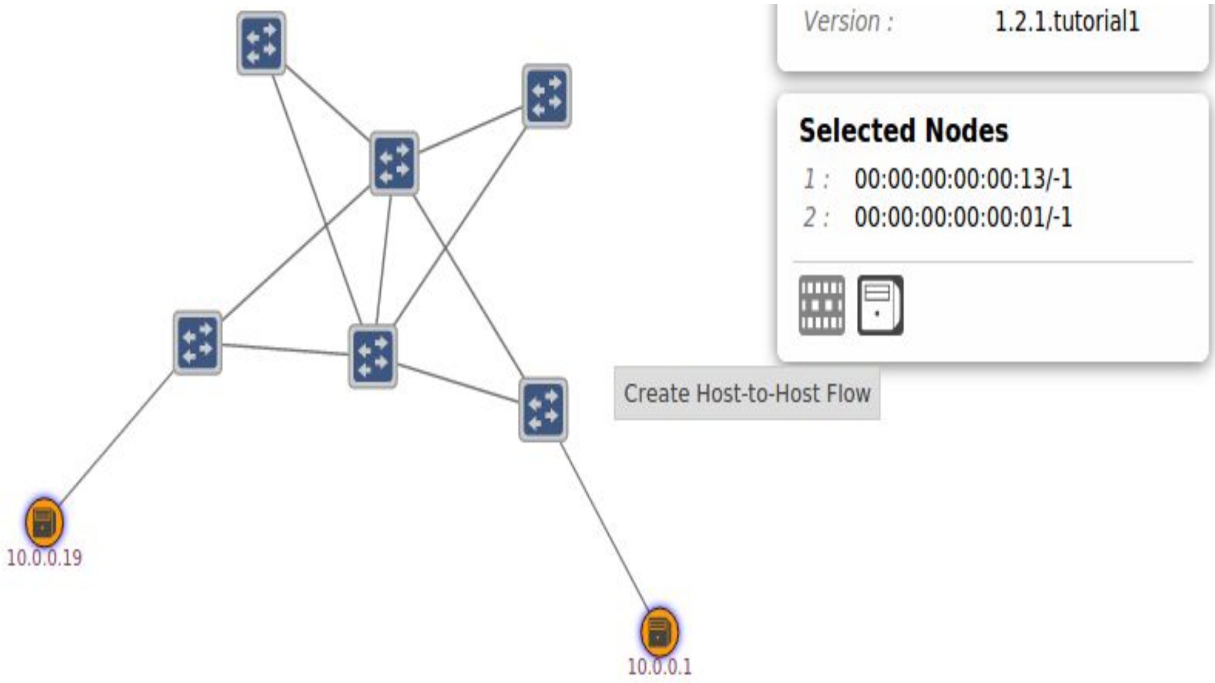
```
onos> intents
```

- This command will return nothing as there are no intents in the system.

Install Intent

# One major advantage of using intents over simply using flow entries to program your network is that intents track the state of the network and reconfigure themselves in order to satisfy your intention. For example, if link were to go down the intent framework would reroute your intent (ie. your flows) onto an alternative path.

# Let's install an intent using the ONOS GUI. First select two hosts by clicking on one host then shift-click on another. Let's pick 10.0.0.1 (h11) and 10.0.0.19 (h41). Now a pane will appear on the right and side of the screen as here:


Version : 1.2.1.tutorial1

Selected Nodes

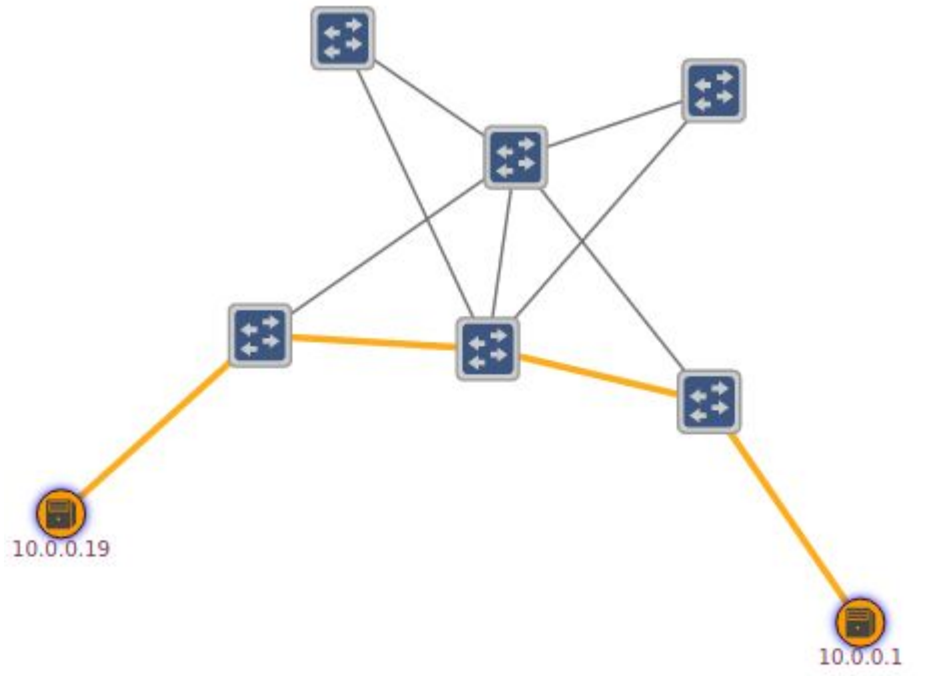
1 : 00:00:00:00:00:13/-1

2 : 00:00:00:00:00:01/-1

Install Intent

# Now click on 'Create Host-to-host Flow', this actually provisions a host to host intent

# If you click on 'Show Related Traffic', it lights up the path used by the intent.



Q2: Using the ONOS GUI, how can I see all intents installed in the network?

Install Intent

# So now that the intent is installed let's have a look what path it is using. Be careful here as the output from the tutorial and what you see may vary slightly as all alternate paths here have equal cost and therefore ONOS is free to pick either one.

```
onos> flows
deviceId=of:0000000000000001, flowRuleCount=7
   id=10000c364dd58, state=ADDED, bytes=0, packets=0, duration=1781, priority=123,
appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=2}, ETH_SRC{mac=00:00:00:00:00:01}, ETH_DST{mac=00:00:00:00:00:13}]
      treatment=[OUTPUT{port=5}]
   id=10000c364ddb2, state=ADDED, bytes=0, packets=0, duration=1781, priority=123, appId=org.onlab.onos.net.intent
      selector=[IN_PORT{port=5}, ETH_SRC{mac=00:00:00:00:00:13}, ETH_DST{mac=00:00:00:00:00:01}]
      treatment=[OUTPUT{port=2}]
      …
deviceId=of:0000000000000002, flowRuleCount=5
      …
```

Q3: Based on the flow entries information, could you say which switches are traversed by the packets between h11 (10.0.0.1) and h41 (10.0.0.19)?

Install Intent

# let's teardown the link between s1 and s11, you may have to teardown the link between s2 and s11 so pay attention to the flows command output. This can be done in mininet by running:

```
mininet> link s1 s11 down
```

# let's have a look at the flows again.

```
onos> flows
```

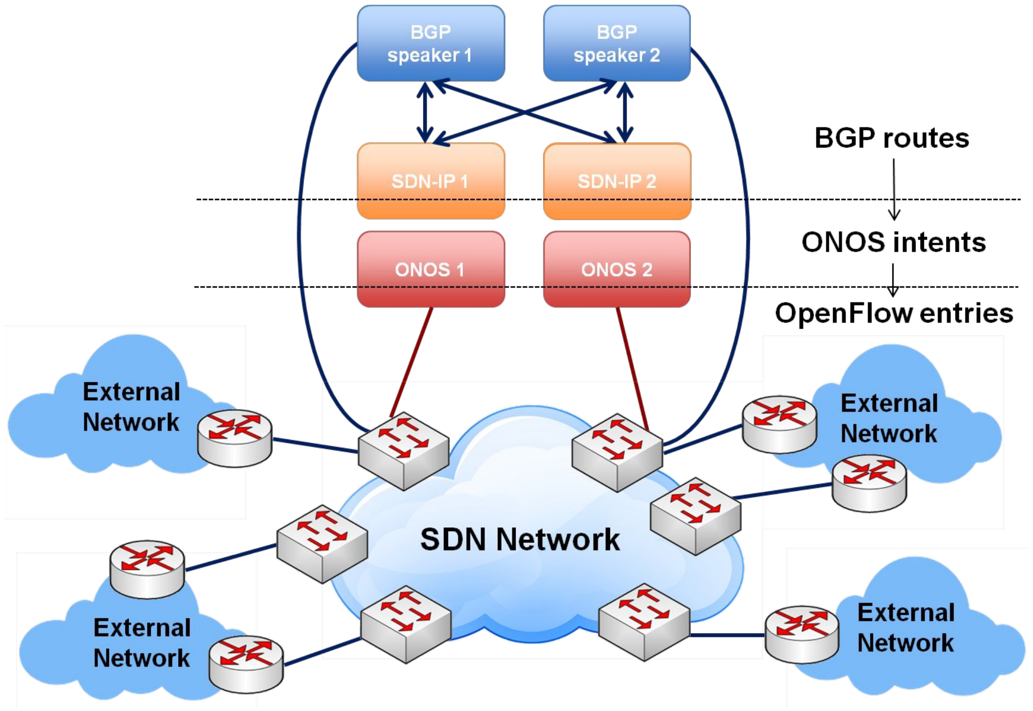
Q4: Now, packets between h11 (10.0.0.1) and h41 (10.0.0.19) traversed the same switches that Q3?

- When we tore down the link between s1 and s11, ONOS detected this change and informed all people interested by this event that the link went down. Therefore the intent service receives this information and realises that one of its intents is affected by this change and thus it recompiles the intent in light of this change which causes the intent to be installed on a different path.
- This simple example shows that using intents is more powerful than simply installing flows. Intents maintain your intention (hence the name!) while retaining the ability to install them as is possible or most efficient.

(OPTIONAL) Exercise 4: SDN-IP

Introduction

- SDN-IP is an ONOS application that **allows a Software Defined Network to connect to external networks on the Internet using the standard Border Gateway Protocol (BGP)**.
- Externally, from a BGP perspective, the SDN network appears as a single Autonomous System (AS) that behaves as any traditional AS. Within the AS, the SDN-IP application provides the integration mechanism between BGP and ONOS.
- At the protocol level, SDN-IP behaves as a regular BGP speaker.
- From ONOS perspective, it's just an application that uses its services to install and update the appropriate forwarding state in the SDN data plane.



REFERENCES:

Architecture: https://wiki.onosproject.org/display/ONOS/SDN-IP+Architecture

SDN-IP Deployment Guidelines: https://wiki.onosproject.org/display/ONOS/SDN-IP+Deployment+Guidelines

Starting the exercise

- If you're already logged, please log out by clicking the bottom-left icon, clicking **“Logout”**, then click **“Logout”** again.

- This tutorial in the VM is presented as a different user. Log in to the SDN-IP tutorial user with the following credentials:
  - Username: **sdnip**
  - Password: **sdnip**

- You'll be deposited on a desktop with a bunch of icons that will be used for the tutorial.

Network Topology

- We've prepared a simple emulated Mininet topology, which contains some OpenFlow switches to make up the SDN network.
- Connected around the edges of the SDN network are emulated routers. The routers run a piece of software called **Quagga**, which is an open-source routing suite.
    - Note that it is not mandatory to use Quagga; any software/hardware capable of speaking BGP will do. In our case we run the BGP part of Quagga on them, to simulate external BGP routers belonging to other administrative domains.
- The goal of SDN-IP is to be able to talk BGP with these routers in order to exchange traffic between the different external ASes.

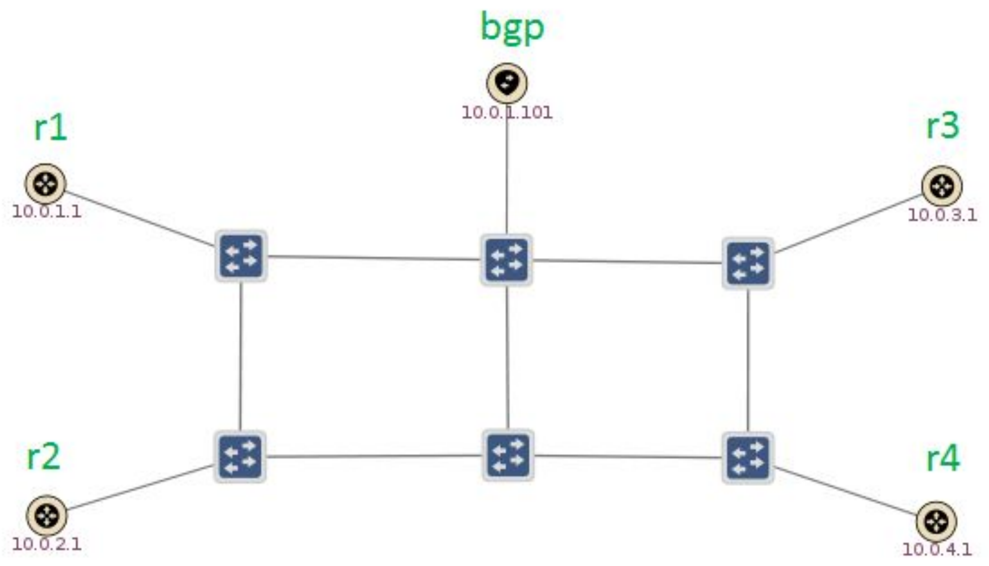
Network Topology

This figure shows the topology as observed by ONOS. We can see 6 blue OpenFlow switches, and 5 peripheral nodes with yellow icons.

- The node labelled "bgp" is our Internal BGP Speaker. It sits inside our SDN network and its job is to peer with all the External BGP Routers, learn BGP routes from them, and relay those routes to the SDN-IP application running on ONOS.
- The other four nodes, labelled r1 through r4, are the External BGP Routers. They are the border routers that reside in other networks that want to exchange traffic with us.
- Behind each router is a host. These are labelled h1 through h4 in Mininet. ONOS can't see these hosts, because they reside in other networks that are not controlled by ONOS.

- **Note:** The miminet code can be found in:

  /home/sdnip/sdnip/tutorial.py



Start up ONOS

- First double-click the **“Reset”** icon on the network to clean the environment.

- Start up the network
  - Double-click the **“SDN-IP Mininet”** icon on the desktop to start up the network.
  - We can look at the configuration of the hosts in the Mininet terminal.

```
mininet> h1 ip addr show
...
    inet 192.168.1.1/24 brd 192.168.1.255 scope global h1-eth0
...

mininet> h2 ip addr show
...
    inet 192.168.2.1/24 brd 192.168.2.255 scope global h2-eth0
…
```

- Each host is in a different IP subnet. When SDN-IP is up and running, these hosts will be able to communicate with one another despite being in different networks. This is because the SDN network is able to route traffic based on BGP routes.

Start up ONOS

- Also, double-click the **“ONOS”** icon on the desktop to start up the ONOS console. If you run the ***“devices”*** command, you should see the network has started up and connected to ONOS.

```
onos> devices
id=of:00000000000000a1, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch,
sw=2.3.0, serial=None, protocol=OF_10
id=of:00000000000000a2, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch,
sw=2.3.0, serial=None, protocol=OF_10
id=of:00000000000000a3, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch,
sw=2.3.0, serial=None, protocol=OF_10
id=of:00000000000000a4, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch,
sw=2.3.0, serial=None, protocol=OF_10
id=of:00000000000000a5, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch,
sw=2.3.0, serial=None, protocol=OF_10
id=of:00000000000000a6, available=true, role=MASTER, type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch,
sw=2.3.0, serial=None, protocol=OF_10
```

Running SDN-IP for the first time

- If you try and ping between any two hosts right now, you'll notice nothing is working.

```
mininet> h1 ping h2
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
From 192.168.1.254 icmp_seq=1 Destination Net Unreachable
From 192.168.1.254 icmp_seq=2 Destination Net Unreachable
From 192.168.1.254 icmp_seq=3 Destination Net Unreachable
```

- Even though ONOS is running and connected to switches, there are no applications loaded so there is nothing to tell ONOS how to control the network.

Install the application

- First we need to install some helper applications that SDN-IP relies on. These apps let ONOS read in various configuration files and respond to ARP requests between the external routers and internal BGP speakers.

```
onos> app activate org.onosproject.config
onos> app activate org.onosproject.proxyarp
```

- Now, let's install the SDN-IP application so we can get some traffic flowing between our networks.

```
onos> app activate org.onosproject.sdnip
```

- A lot happens as soon as we install the SDN-IP application. The first thing it does is install point-to-point intents to allow the external BGP peers to communicate with our internal BGP speaker. This allows the external BGP routers to relay the routes that are capable of forwarding through to SDN-IP.

Install the application

- We can see the routes that SDN-IP has learnt with the ***"routes"*** command.

```
onos> routes
   Network          Next Hop
   192.168.1.0/24      10.0.1.1
   192.168.2.0/24      10.0.2.1
   192.168.3.0/24      10.0.3.1
Total SDN-IP IPv4 routes = 3
   Network          Next Hop
Total SDN-IP IPv6 routes = 0
```

  - Don't worry if you don't see all of the routes straight away - sometimes it takes a minute or so for the BGP sessions to establish and advertise the routes to ONOS.
- Now that ONOS has learned some routes, it has programmed those routes into the switches using the intent API. If we look at the intent summary, we can see the different intents that SDN-IP is using.

```
onos> intents -s
```

  - We see a total of 27 intents.
  - The 24 PointToPoint intents are simple end-to-end flows which allow the external BGP routers to communicate with our internal BGP speaker.
  - The 3 MultiPointToSinglePoint intents are flows which allow external BGP routers to communicate directly between then through the OpenFlow data plane.

Install the application

- Now that the intents are installed, we can ping through the network. Go back to the Mininet console and try ping between a pair of hosts.

  ```
  mininet> h1 ping h2
  PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
  ...
  ```

  - **TROUBLESHOOTING NOTE:** If the ping doesn't work straight away, try to quit the Mininet network and restart it by double-clicking the "SDN-IP Mininet" icon. There is sometimes a bug with the script where the routes do not get put into the routing table of the linux routers correctly. Restarting Mininet often fixes this.

  Q5: What is the TTL value? Why the default value is not 64?

- You can try pinging between some of the other hosts such as h1, h2 and h3. We can't ping h4 yet, but we'll address that in the next slide.

Advertising a new route

- Now that we've got the system up and running, let's see what happens when there's a change in the BGP routes. We're going to make one of the external routers advertise a new route, which will allow us to talk to a new host. Right now r4 is not advertising any routes, so we can't talk to h4. Let's verify this by trying to ping h4.

```
mininet> h1 ping h4
PING 192.168.4.1 (192.168.4.1) 56(84) bytes of data.
From 192.168.1.254 icmp_seq=1 Destination Net Unreachable
From 192.168.1.254 icmp_seq=2 Destination Net Unreachable
From 192.168.1.254 icmp_seq=3 Destination Net Unreachable
From 192.168.1.254 icmp_seq=4 Destination Net Unreachable
```

- To make r4 advertise a new route, we have to change the configuration of the BGP router. In our case, the BGP router is a Quagga process, so we'll connect to the Quagga CLI and configure r4 to advertise a new route. (The Quagga CLI is complex and includes lots of options, but considering this is not a Quagga tutorial we won't go into much detail here. If you're interested, there's Quagga CLI documentation **here**).

Advertising a new route

- First, from the Mininet CLI we can start up new terminal on the r4 router so we can connect to the Quagga CLI.

```
mininet> xterm h4
```

- A new terminal window will pop up which gives us a terminal on the r4 router node. The next few commands will by typed into this window - pay attention to the command prompt to ensure you're typing commands into the correct place.

- We can use telnet to connect to the Quagga BGP CLI.

```
root@mininet-vm:~# telnet 10.0.4.1 2605
Trying 10.0.4.1...
Connected to 10.0.4.1.
Escape character is '^]'.
Hello, this is Quagga (version 0.99.23).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
User Access Verification
Password:
```

- It will prompt for a password, which is **sdnip**. Once the password is entered, we'll drop into a prompt r4>

Advertising a new route

- Now we can begin to configure the router to advertise a new network.

```
r4> enable
r4# configure terminal
r4(config)# router bgp 65004
r4(config-router)# network 192.168.4.0/24
r4(config-router)# exit
r4(config)# exit
r4# exit
Connection closed by foreign host.
```

- Now our external router r4 has advertised a new route to the SDN network. We're done with our r4 terminal window, so you can close it (it's the one titled "root@mininet-vm: ~").

- Let's go back to our ONOS terminal and see if ONOS has received the new route.

```
onos> routes
   Network           Next Hop
   192.168.1.0/24       10.0.1.1
   192.168.2.0/24       10.0.2.1
   192.168.3.0/24       10.0.3.1
   192.168.4.0/24       10.0.4.1
Total SDN-IP IPv4 routes = 4
   Network           Next Hop
Total SDN-IP IPv6 routes = 0
```

Advertising a new route

- Also, when SDN-IP received the route it installed a new MultiPointToSinglePoint intent into the network.

```
onos> intents -s
...
MultiPointToSinglePoint  total=              4   installed=         4
MultiPointToSinglePoint  withdrawn=          0   failed=            0
MultiPointToSinglePoint  submitted=          0   compiling=         0
MultiPointToSinglePoint  installing=         0   recompiling=       0
MultiPointToSinglePoint  withdrawing=        0
...
```

- Our number of MultiPointToSinglePoint intents has increased to 4. Now let's see if we can ping to our new network.

```
mininet> h1 ping h4
PING 192.168.4.1 (192.168.4.1) 56(84) bytes of data.
64 bytes from 192.168.4.1: icmp_seq=1 ttl=62 time=0.595 ms
64 bytes from 192.168.4.1: icmp_seq=2 ttl=62 time=0.182 ms
64 bytes from 192.168.4.1: icmp_seq=3 ttl=62 time=0.164 ms
…
```

- Now we can ping to h4 which is in the network we just received through BGP. This shows that whenever the routes learned through BGP are updated, SDN-IP reacts to the update and programs the data plane accordingly.