

Hardware OpenFlow Tutorial using jFed Experimenter

Version: 1.0

Author: Frederic Francois

Email: f.francois@bristol.ac.uk

Contents

Tutorial Overview.....	3
Prerequisites	3
Tutorial Steps	3
Step 1: launching jFed Experimenter	3
Step 2: creating the virtual machines	4
Step 3: logging into the virtual machines	6
Step 4: finding the IP of the leftvm virtual machine to act as controller.....	6
Step 5: reserving the network flowspace	6
Step 6: configuring the virtual machines for the reserved flowspace.....	9
Step 7: starting the POX controller with L2 switch learning	10
Step 8: running a ping test between the 2 virtual machines	10
Step 9: terminating your experiment	12
Conclusions	12
Appendix	13
VTAM Reservation RSpec.....	13
FOAM Reservation RSpec	14

Tutorial Overview

In this tutorial, you will be taught how to use jFed Experimenter to setup a simple network topology and run an OpenFlow L2 switch application. Figure 1 shows the network topology that will be used. It consists of two OpenFlow packet switches which will inter-connect two separate virtual machines that the experimenter will create on the virtualization servers of the testbed. The POX controller with the simple L2 switch application will be used to provide the forwarding logic to the OpenFlow packet switches.

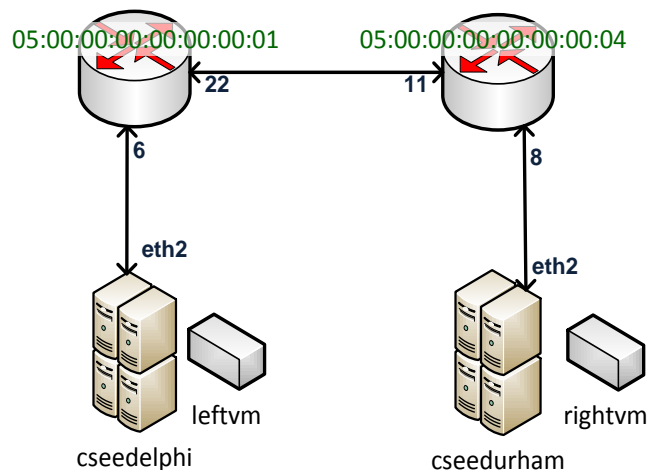


Figure 1: Simple network topology for the tutorial.

Prerequisites

Before an experimenter can successfully perform the following steps of the tutorial, it is expected that the experimenter is in possession of a credential of one of the Fed4FIRE authorities. If you don't have one, you can follow the instructions on <http://doc.fed4fire.eu/getanaccount.html> to get one. You should download the credential to your computer once your account is created.

Tutorial Steps

Step 1: launching jFed Experimenter

Use an internet browser to browse to <http://ifed.iminds.be/releases/r2042/> and launch jFed Experimenter by following the instructions on the webpage. A login window for jFed, similar to Figure 2, will appear when jFed is launched.



Figure 2: jFed login window.

Once the login window appears, click on [1] to browse to the location of your credential. After your credential is successful loaded by jFed, your “Username” and “Authority” should appear in the relevant fields. You then have to fill in the passphrase of your credential in [2] and click [3] to finish the login phase of jFed.

Step 2: creating the virtual machines

An already-prepared reservation RSpec (a RSpec is an XML file to describe resources at the testbed) is going to be used to create two virtual machines, **leftvm** and **rightvm** on the two virtualization servers **cseedelphi** and **cseedurham** at UNIVBRIS OFELIA. These virtual machines will act as the source and sink of traffic. The **leftvm** virtual machine will also host the POX controller along with the L2 switch application.

To start the creation process of the virtual machines, click “Open URL” [1] in Figure 3 and paste the following URL in the pop-up window (also available in the appendix of this document):

http://univbrisofeliaf4f.blogs.ilrt.org/files/2014/10/of_tutorial_vtam.rspec_.txt

After jFed reads the URL, you will see a new tab in jFed as shown in Figure 4. You can click the green play button [2] in Figure 4 to send the resource request to the testbed and start the creation of the virtual machines. A pop-up window will ask you to name a slice and duration. You can set these to any valid value that you wish.

After the virtual machines are ready, they will become green in colour (see [1] in Figure 5) and a green tick will appear next to “waiting for nodes from ... to become ready” (see [2] in Figure 5).

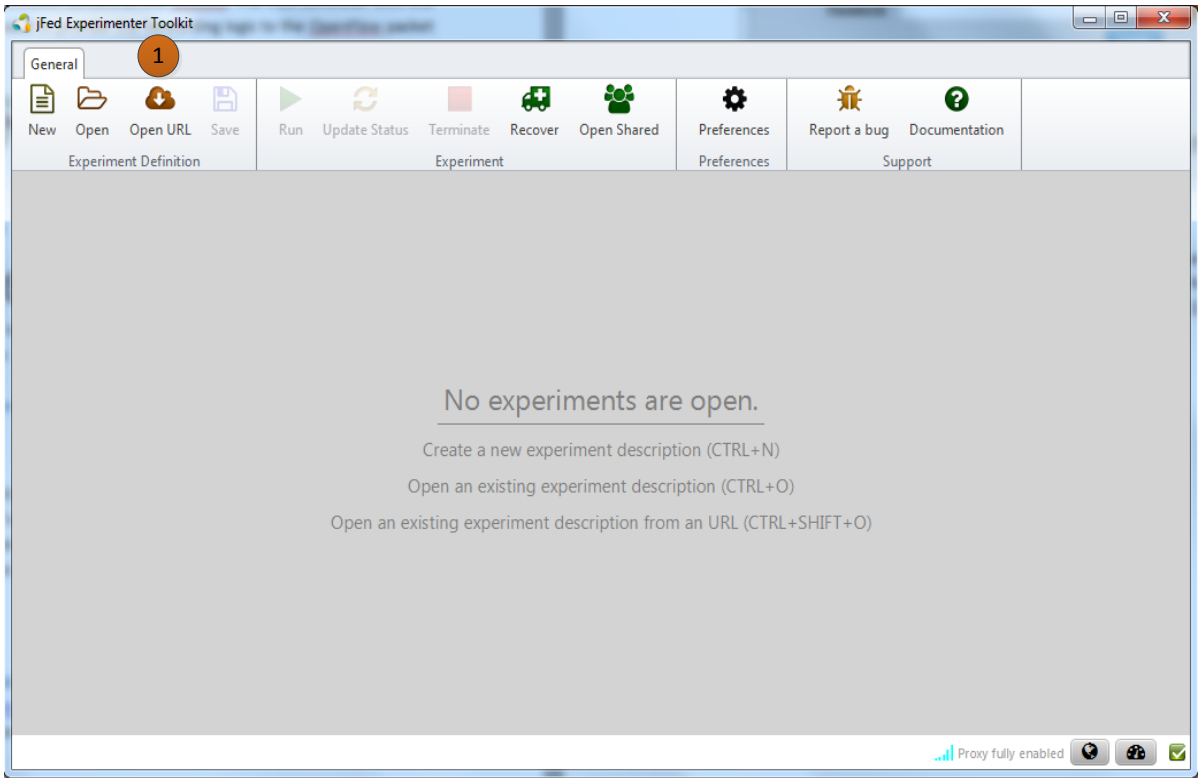


Figure 3: jFed window after login.

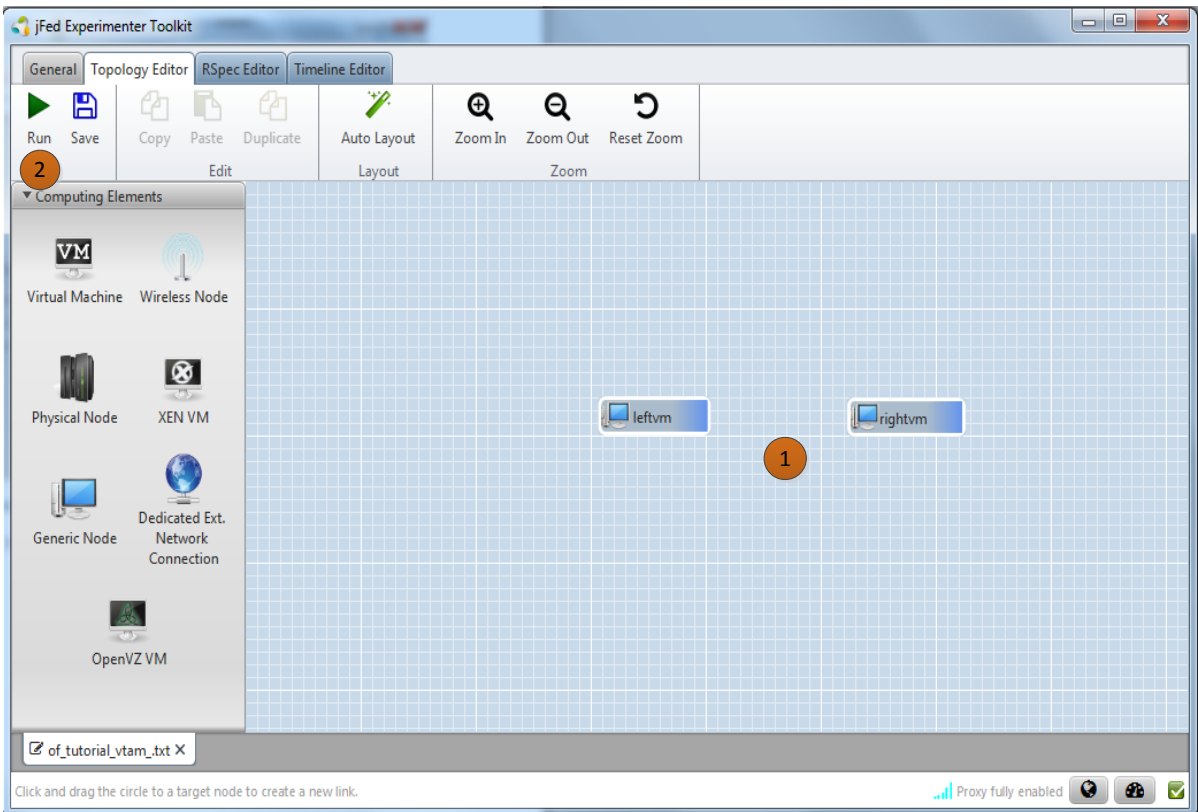


Figure 4: jFed window with virtual machines loaded.

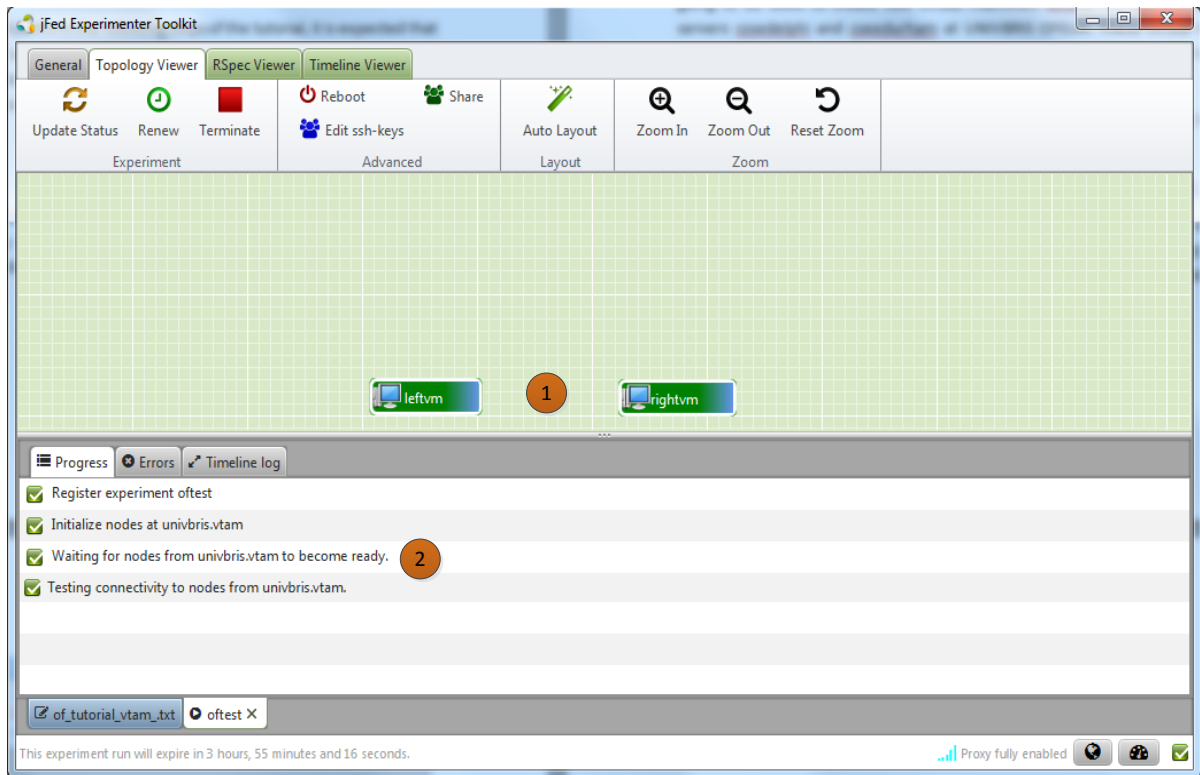


Figure 5: jFed window after the virtual machines are ready to be logged into.

Step 3: logging into the virtual machines

You can SSH into your virtual machines by right-clicking the virtual machine icon and selecting “Open SSH terminal” in the menu. A terminal window will appear and ask for the passphrase of your jFed credential. You will log in as root user and can perform actions with root privileges.

Step 4: finding the IP of the leftvm virtual machine to act as controller

In the terminal of the **leftvm** virtual machine, you can type *ifconfig* to get the control IP of **leftvm**—which is the IP of *eth0*. In this particular case, it is *10.216.22.53* but it may be different when you are running your experiment.

Step 5: reserving the network flowspace

A pre-configured RSpec will be used to reserve a VLAN on the ports of the two OpenFlow packet switches shown in Figure 1. In jFed Experimenter, click on the top “General” tab and click “Open URL” and paste the following link to get the network/OpenFlow RSpec (also available in the appendix of this document):

http://univbrisofeliaf4f.blogs.ilrt.org/files/2014/10/of_tutorial_foam.rspec.txt

```

10.216.22.53 - PuTTY
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jul  8 21:59:13 2014 from 10.216.22.251
root@leftvm:~# ifconfig
eth0    Link encap:Ethernet  HWaddr 02:06:00:00:00:02
        inet addr:10.216.22.53  Bcast:10.216.23.255  Mask:255.255.252.0
        inet6 addr: fe80::6:ff:fe00:2/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:1678 errors:0 dropped:0 overruns:0 frame:0
        TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:137741 (134.5 KiB)  TX bytes:11983 (11.7 KiB)
        Interrupt:76

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@leftvm:~#

```

Figure 6: ifconfig result of the leftvm virtual machine.

After the RSpec is opened, you need to customize the RSpec for your specific experiment. In order to do so, you need to click the “RSpec Editor” tab in jFed Experimenter to get the editor.

```

jFed Experimenter Toolkit
General Topology Editor RSpec Editor Timeline Editor
Run Save Format Code Search Search & Replace Code
Command: http://jfed.iminds.be/rspec/ext/jfed-command/1 xmlns:of="http://www.w3.org/2004/xmlschema-instance"
xmlns:jfedBonfire="http://jfed.iminds.be/rspec/ext/jfed-bonfire/1"
xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
xmlns:delay="http://www.protogeni.net/resources/rspec/ext/delay/1" xmlns:jfed="http://jfed.iminds.be/rspec/ext/jfed/1"
xmlns:sharedvlan="http://www.protogeni.net/resources/rspec/ext/shared-vlan/1" xmlns:jfed-ssh-
keys="http://jfed.iminds.be/rspec/ext/jfed-ssh-keys/1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.geni.net/resources/rspec/3 http://www.geni.net/resources/rspec/3/request.xsd
http://www.geni.net/resources/rspec/ext/openflow/3/of-rev.xsd"
3 <openflow:sliver email="f.francois@bristol.ac.uk" description="OF request example">
4 <openflow:controller type="primary" url="tcp:10.216.22.51:6633"/>
5 <openflow:group name="fs1">
6 <openflow:datapath component_id="urn:publicid:IDN+openflow:ofam:univbrist+datapath+05:00:00:00:00:00:01"
component_manager_id="urn:publicid:IDN+openflow:ofam:univbrist+authority+cm" dpid="05:00:00:00:00:00:00:01">
7 <openflow:port num="6" name="GBE0/6"/>
8 <openflow:port num="22" name="GBE0/22"/>
9 </openflow:datapath>
10 <openflow:datapath component_id="urn:publicid:IDN+openflow:ofam:univbrist+datapath+05:00:00:00:00:00:04"
component_manager_id="urn:publicid:IDN+openflow:ofam:univbrist+authority+cm" dpid="05:00:00:00:00:00:00:04">
11 <openflow:port num="11" name="GBE0/11"/>
12 <openflow:port num="8" name="GBE0/8"/>
13 </openflow:datapath>
14 </openflow:group>
15 </openflow:match>
16 <openflow:use-group name="fs1"/>
17 <openflow:packet>
18 <openflow:dl_type value="0x800,0x806"/>
19 <openflow:dl_vlan value="56"/>
20 </openflow:packet>
21 </openflow:match>
22 </openflow:sliver>
23 </rspec>
of_tutorial_vtam_txt oftest of_tutorial_foam_txt X
Proxy fully enabled

```

Figure 7: RSpec Editor in jFed Experimenter.

In the RSpec Editor, you need to make the following modifications:

- (a) modify the email attribute of the RSpec in line 3 to specify your email address, and
- (b) modify the IP of the controller in line 4 from 10.216.22.51 to the IP of your **leftvm** as found in Step 4.

After doing the two modifications above, you can run the experiment by clicking the green play button as before and completing the usual prompts.

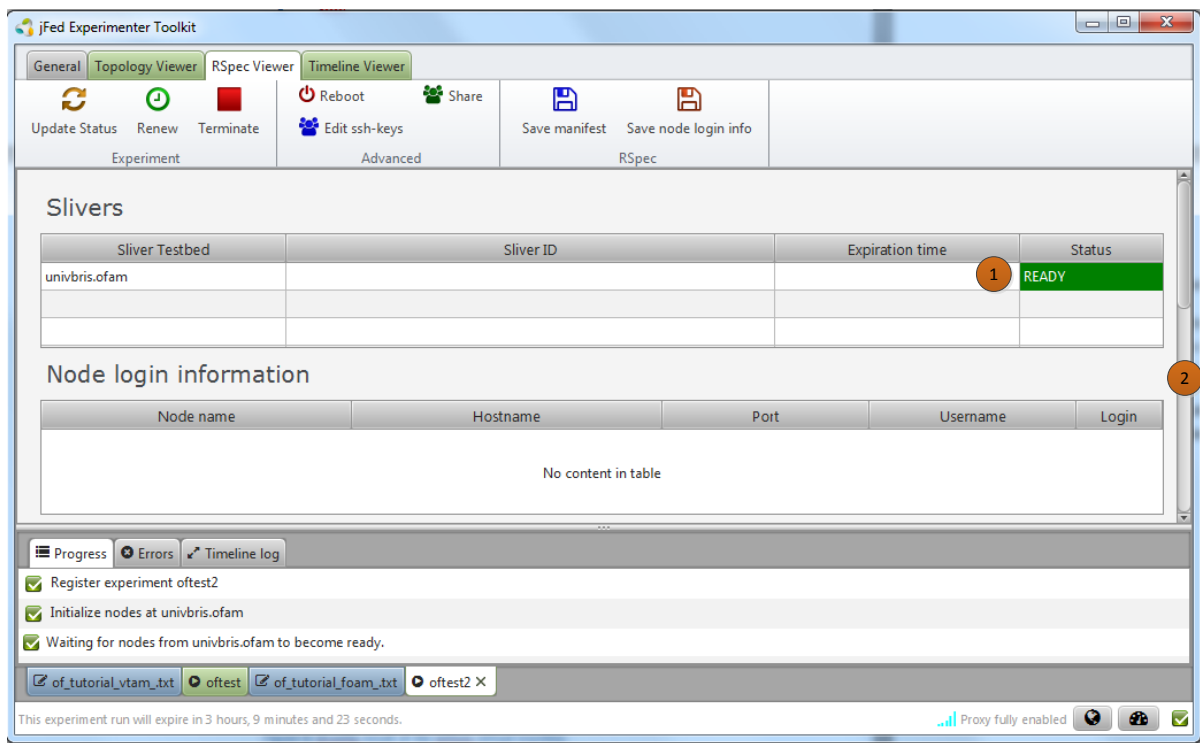


Figure 8: RSpec Viewer in jFed Experimenter after the network resources are reserved.

After the network resources goes into ready state (see [1] in Figure 8), you can scroll down to see the Manifest XML by using the scroll bar in [2] in Figure 8.

The Manifest XML allows you to see exactly which VLAN is reserved for your experiment. You can find the VLAN by clicking on the radio button “specific” (see [1] in Figure 9) and selecting the SFA authority “urn:publicid:IDN+openflow:ofam:univbris+authority+cm” in the drop-down menu (see [2] in Figure 9). You then need to scroll down in the textbox (see [3] in Figure 9) to look at the specific reservation allocated to you.

You will find the line 18 (this may be another nearby line for you), the line “<openflow:dl_vlan value="56"/>”. Here, the VLAN allocated is 56 but this may be different for you. You need to make a note of this VLAN so that you can configure your virtual machine appropriately. From now on, all the packets tagged with the VLAN allocated to you and passing through the ports and switches specified in the manifest RSpec will be under your control.

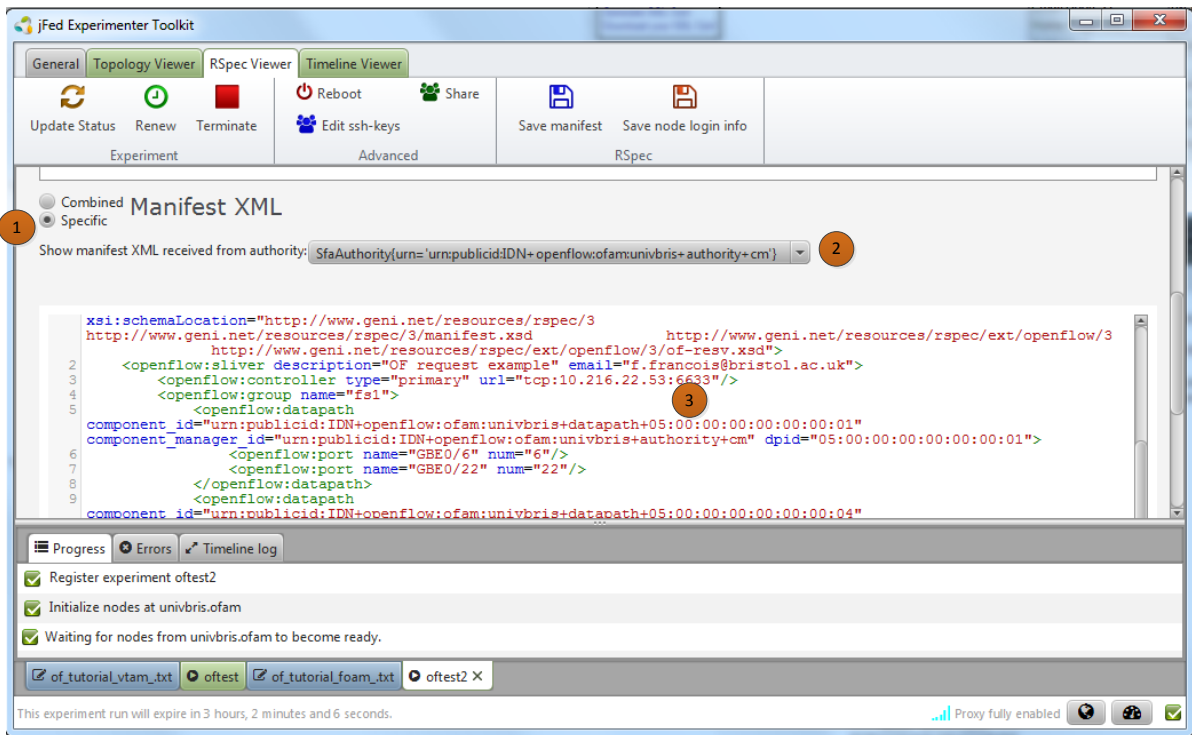


Figure 9: RSpec Viewer in jFed Experimenter to see the manifest RSpec.

Step 6: configuring the virtual machines for the reserved flowspace

Before the experiment can start, the network interfaces to the OpenFlow packet switches need to be configured. In this experiment, the interface is eth1 and is down by default.

You can run the following commands in the terminal of each of your virtual machines:

leftvm virtual machine

- (a) bring eth1 interface up: *ifconfig eth1 up*
- (b) configure allocated VLAN on interface eth1, replace <allocated_vlan> with the vlan allocated in step 5: *vconfig add eth1 <allocated_vlan>*
i.e. *vconfig add eth1 56*
- (c) give an IP to the new VLAN interface: *ifconfig eth1.<allocated_vlan> 192.168.1.1/24 up*
- (d) set the maximum transmission unit to 1496 bytes: *ifconfig eth1.<allocated_vlan> mtu 1496*

rightvm virtual machine

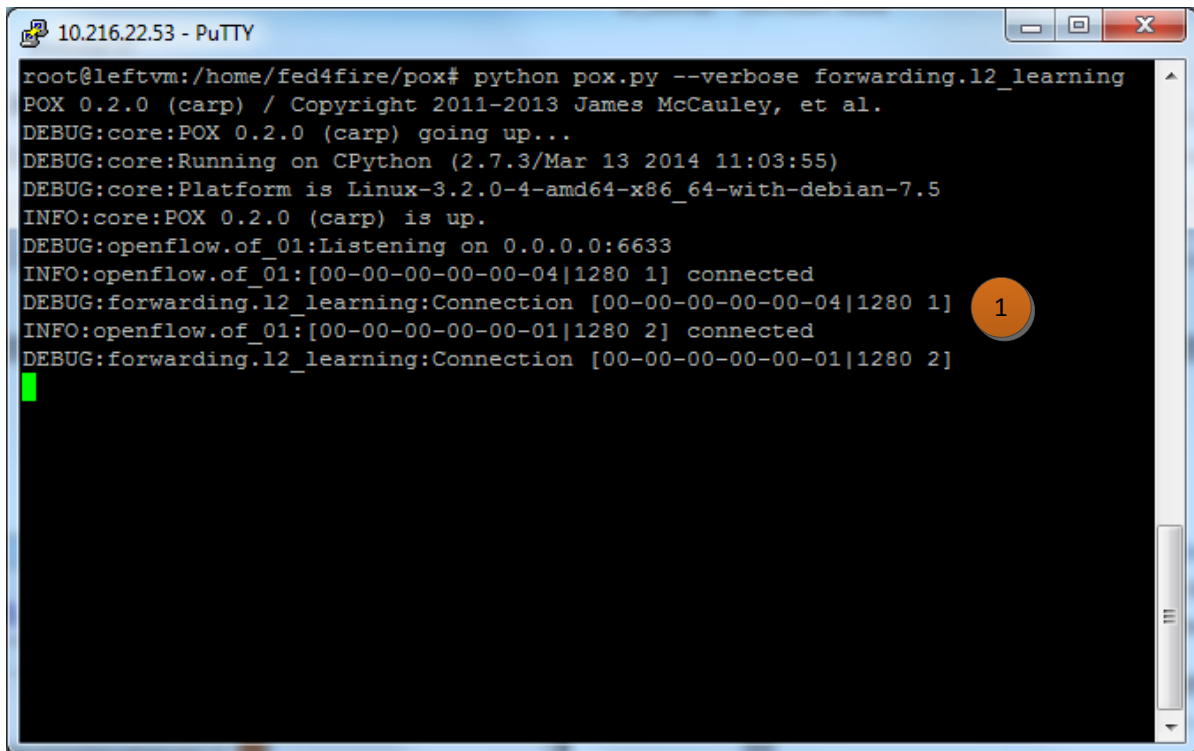
- (a) bring eth1 interface up: *ifconfig eth1 up*
- (b) configure allocated VLAN on interface eth1, replace <allocated_vlan> with the vlan allocated in step 5: *vconfig add eth1 <allocated_vlan>*
i.e. *vconfig add eth1 56*
- (c) give an IP to the new VLAN interface: *ifconfig eth1.<allocated_vlan> 192.168.1.2/24 up*
- (d) set the maximum transmission unit to 1496 bytes: *ifconfig eth1.<allocated_vlan> mtu 1496*

Step 7: starting the POX controller with L2 switch learning

The POX controller can be launched by opening a new SSH terminal for the **leftvm** virtual machine and using the following commands:

- (a) `cd /home/fed4fire/pox`
- (b) `python pox.py --verbose forwarding.l2_learning`

After the POX has started, you will see 2 switches being connected to the controller as shown by [1] in Figure 10.



```
10.216.22.53 - PuTTY
root@leftvm:/home/fed4fire/pox# python pox.py --verbose forwarding.l2_learning
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.3/Mar 13 2014 11:03:55)
DEBUG:core:Platform is Linux-3.2.0-4-amd64-x86_64-with-debian-7.5
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-04|1280 1] connected
DEBUG:forwarding.l2_learning:Connection [00-00-00-00-00-04|1280 1]
INFO:openflow.of_01:[00-00-00-00-00-01|1280 2] connected
DEBUG:forwarding.l2_learning:Connection [00-00-00-00-00-01|1280 2]
```

Figure 10: Information given by the POX controller after the switches connect to it.

Step 8: running a ping test between the 2 virtual machines

You can run a ping test between the two virtual machines by typing the following command in the terminal of **leftvm**: `ping 192.168.1.2 -c5` where 192.168.1.2 is the IP of **rightvm**.

You can see in Figure 11 that the ping is successful. One interesting observation is that the first ping latency is much larger than the subsequent pings. This is normal since initially the switches do not have a rule in order to handle the new ping packets and they need to send it to the POX controller so that a decision is made about the forwarding of these packets. After the controller installs the rules in the switches, subsequent packets no longer have to be sent to the controller and the ping latency reduces to sub-milliseconds. Figure 12 shows some of the actions performed by the L2 switch application of the POX controller, i.e. installing rules into the switches.

```
10.216.22.53 - PuTTY
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:468 (468.0 B)

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

root@leftvm:~# ping 192.168.1.2 -c5
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_req=1 ttl=64 time=74.0 ms
64 bytes from 192.168.1.2: icmp_req=2 ttl=64 time=0.365 ms
64 bytes from 192.168.1.2: icmp_req=3 ttl=64 time=0.413 ms
64 bytes from 192.168.1.2: icmp_req=4 ttl=64 time=0.307 ms
64 bytes from 192.168.1.2: icmp_req=5 ttl=64 time=0.357 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.307/15.102/74.072/29.485 ms
root@leftvm:~#
```

Figure 11: Ping test from the virtual machine leftvm to the rightvm one.

```
10.216.22.53 - PuTTY
DEBUG:forwarding.l2_learning:Connection [00-00-00-00-00-04|1280 1]
INFO:openflow.of_01:[00-00-00-00-00-01|1280 2] connected
DEBUG:forwarding.l2_learning:Connection [00-00-00-00-00-01|1280 2]
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:09.8 -> 02:06:00:00:00:03.11
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:09.22 -> 02:06:00:00:00:00:03.6
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:03.6 -> 02:06:00:00:00:00:09.22
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:03.11 -> 02:06:00:00:00:00:09.8
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:09.8 -> 02:06:00:00:00:00:03.11
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:09.22 -> 02:06:00:00:00:00:03.6
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:09.8 -> 02:06:00:00:00:00:03.11
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:09.22 -> 02:06:00:00:00:00:03.6
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:03.6 -> 02:06:00:00:00:00:09.22
DEBUG:forwarding.l2_learning:installing flow for 02:06:00:00:00:03.11 -> 02:06:00:00:00:00:09.8
```

Figure 12: POX controller installing new rules for the ping packets.

You can check whether the controller is responsible for the connectivity between the virtual machines by killing the controller by pressing “Ctrl + C” and running the ping test after waiting for 1-2 minute for the existing rules to expire in the switches.

Step 9: terminating your experiment

To terminate your experiment, you can click on the red “Terminate” button in the top menu of jFed as shown in [1] in Figure 13. The icon of the virtual machine should become black after they become uninitialized in the testbed. For the network resources, after you click the terminate button, the status of [1] in Figure 8 will go to uninitialized with a black background.

It should be noted that your experiment will also terminate automatically after your sliver has expired at the testbed. You can set the expiry date of the sliver when first creating the experiment by clicking the green play button. You can see the expiry date of the sliver by looking at the bottom status bar, i.e. [2] in Figure 13.

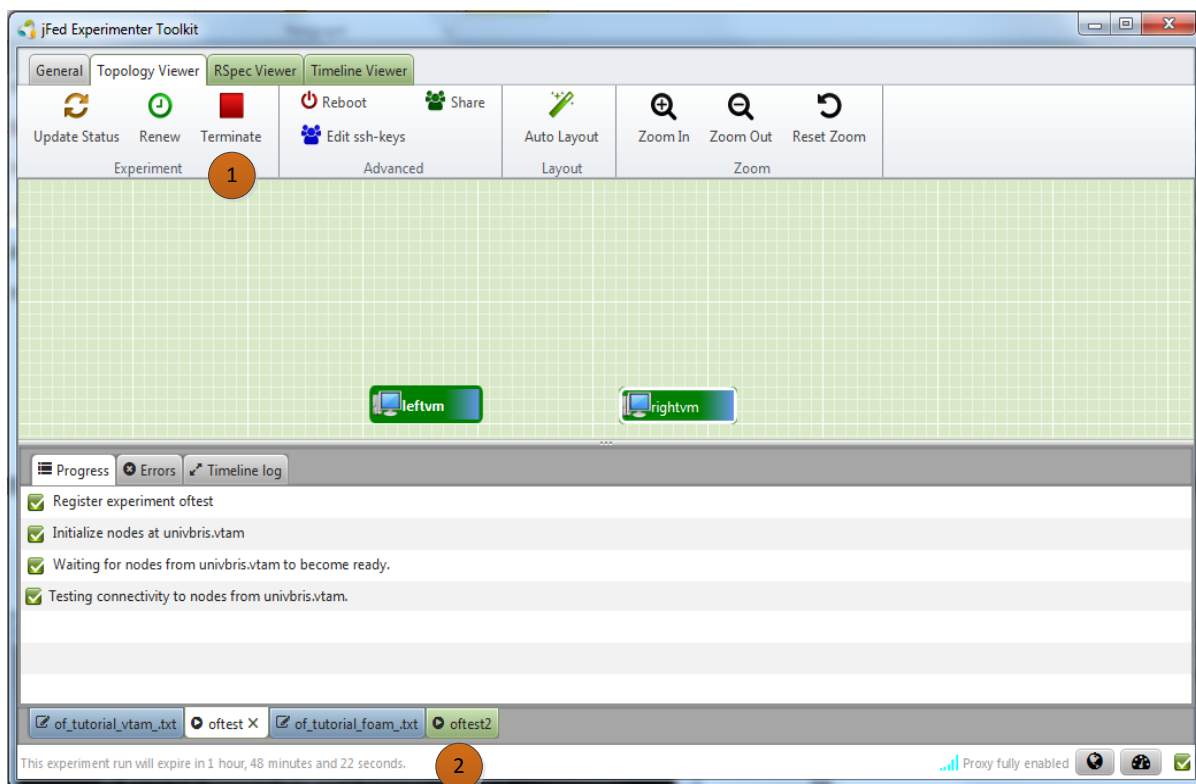


Figure 13: Terminating the experiment in jFed Experimenter.

Conclusions

Now that you have successfully run your experiment, you can look at the inner workings of the testbed from an experimenter point-of-view at <https://univbrisofeliaf4f.blogs.ilrt.org/user-manual-introduction/>. While the user manual on the website uses the OMNI reservation tool, you can use the information on the website to create your own customized RSpecs. You can then use these RSpecs in jFed by pasting the RSpec in the RSpecs Editor.

Appendix

VTAM Reservation RSpec

```
<rspec xmlns="http://www.geni.net/resources/rspec/3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" expires="2014-09-
15T16:18:38Z" generated="2014-09-15T16:18:38Z" type="request"
xsi:schemaLocation="http://www.geni.net/resources/rspec/3
http://www.geni.net/resources/rspec/3/manifest.xsd">
  <node client_id="leftvm"
component_id="urn:publicid:IDN+vtam.univbris+node+cseedelphi"
component_manager_id="urn:publicid:IDN+vtam.univbris+authority+cm"
exclusive="false">
    <sliver_type name="ofelia-vm"/>
  </node>
  <node client_id="rightvm"
component_id="urn:publicid:IDN+vtam.univbris+node+cseedurham"
component_manager_id="urn:publicid:IDN+vtam.univbris+authority+cm"
exclusive="false">
    <sliver_type name="ofelia-vm"/>
  </node>
</rspec>
```

FOAM Reservation RSpec

```
<rspec xmlns="http://www.geni.net/resources/rspec/3"
  xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
  xs:schemaLocation="http://www.geni.net/resources/rspec/3
    http://www.geni.net/resources/rspec/3/request.xsd
    http://www.geni.net/resources/rspec/ext/openflow/3
    http://www.geni.net/resources/rspec/ext/openflow/3/of-
resv.xsd"
  type="request">
  <openflow:sliver email="f.francois@bristol.ac.uk" description="OF
request example">
    <openflow:controller url="tcp:10.216.22.51:6633" type="primary"/>
    <openflow:group name="fs1">
      <openflow:datapath
component_manager_id="urn:publicid:IDN+openflow:ofam:univbris+authority+cm"

component_id="urn:publicid:IDN+openflow:ofam:univbris+datapath+05:00:00:00:
00:00:00:01"
          dpid="05:00:00:00:00:00:00:01">
            <openflow:port name="GBE0/6" num="6"/>
            <openflow:port name="GBE0/22" num="22"/>
          </openflow:datapath>
        <openflow:datapath
component_id="urn:publicid:IDN+openflow:ofam:univbris+datapath+05:00:00:00:
00:00:00:04"
          dpid="05:00:00:00:00:00:00:04">
            <openflow:port name="GBE0/11" num="11"/>
            <openflow:port name="GBE0/8" num="8"/>
          </openflow:datapath>
        </openflow:group>
      <openflow:match>
        <openflow:use-group name="fs1" />
        <openflow:packet>
          <openflow:dl_type value = "0x800,0x806"/>
          <openflow:dl_vlan value= "56"/>
        </openflow:packet>
      </openflow:match>
    </openflow:sliver>
  </rspec>
```