Deploying OpenFlow experiments on the Virtual Wall testbed

Maxim Claeys
Jeroen Famaey
Niels Bouten

Ghent University
iMinds
Outline

1. The Virtual Wall network emulation testbed
2. Fed4FIRE: Federation of Future Internet testbeds
3. Executing a basic Virtual Wall experiment using jFed
4. OpenFlow protocol basics
5. Running OpenFlow experiments on the Virtual Wall
The Virtual Wall network emulation testbed
Doing large test setups – The hard way

Creating large test setups

- Can be very time consuming
- Requires a lot of hardware
- No shared usage of hardware
After a few years (or months?)
The iMinds Virtual Wall

- Automated topology creation and device configuration
- Full automatic install of OS & other software
- Fast swap-in and swap-out of experiments
- Experiment management tools
The Virtual Wall

Displays for visualization & demos

1.5 – 1.8 Tbps Non-Blocking Switch

Server nodes
Emulab

- University of Utah testbed (hard- and software)
- Predecessor of the Virtual Wall
- Testbed management software freely available
The Virtual Wall Concept
Virtual Wall: Topology Control
Virtual Wall: Topology Control
Virtual Wall: Topology Control
Virtual Wall: Node Functionality

- Client (Windows)
- Client (Windows CE)
- Client (Linux)
- Client 2
- Client 3
- Network Element
- Impairment Node
- Server (Windows)
- Server (Linux)
- Server
- Server 2
- Server 3
Virtualization

- Client (Windows)
- Client (Windows CE)
- Client (Linux)
- Network Element
- Impairment Node
- Server (Windows)
- Server (Linux)
- S 1 | S 2 | S 3 | S 4
- Cl 1 | Cl 2 | Cl 3 | Cl 4
- Cl 1 | ... | Cl n
Two Wall Setups: Virtual Wall 1

- 200 servers, “older” nodes
  - Dual CPU, quad or eight cores
  - 4 – 12 GB RAM
  - 2 – 6 network interfaces

- Central switch: Force 10 networks
  - 336 x Gb/s port
  - 8 x 10 Gb/s port
  - 1.53 Tb/s backplane

- Wall 2
  - Runs latest version of testbed mgmt software
  - Local features need to be ported
  - No displays
  - Newer hardware

- Wall 3
  - In beta
  - Soon publicly released
Two Wall Setups: Virtual Wall 2

- 100 servers, “newer” nodes
  - 12 cores @ 2.40GHz
  - 24GB RAM
  - 2-5 network interfaces

- 1 Super computational node
- 1 Super graphical node
- Central switch: Force 10 networks
Connection to the iMinds wilab.t

- Virtual Wall can be linked to Wireless testbed
- Limited external interference
- 60 fixed nodes and 20 mobile node carriers
  - Zotac embedded PC
  - Rmoni sensor node
  - Bluetooth
  - Subset equipped with Webcam
Wrap-up: Virtual Wall

- Automate time-consuming manual testbed setup
  - Automatic topology, network and device configuration
  - Reuse of hardware
- Connected to
  - iMinds wireless wilab.t testbed
  - Other European FIRE facilities
- Publicly available
  - For iMinds project partners
  - Through FIRE initiatives (BonFIRE, Fed4FIRE)
Fed4FIRE: Federation of Future Internet testbeds
Fed4FIRE – general info

- IP project coordinated by iMinds
- 10/2012 - 9/2016
- Total budget: 7.75 MEUR
- 28 partners
Fed4FIRE facilities

- EPCC BonFIRE testbed
- UNIVBRIS OFELIA island
- Grid'5000 Europe
- PlanetLab
- iLab.t Virtual Wall
- iLab.t Wireless lab
- FuSeCo playground
- i2CAT OFELIA island
- UPC community lab
- UMA LTE performance lab
- Smart Santander facility
- UC3M optical access testbed
- Sydney: NORBIT testbed
- Korea: KOREN testbed
- Stanford optical access testbed
- Outside EU
- NITOS wireless testbed
- NETMODE wireless testbed
What’s in it for the experimenter?

- Access a wide range of FIRE testbeds.
- Create experiments that break the boundaries of the different individual FIRE domains (wireless, wired, OpenFlow, cloud computing, smart cities, services, etc.)
- Easily access all the required resources with a single account.
- Focus on your core task of experimentation, instead of on practical aspects such as learning to work with different tools for each testbed, requesting accounts on each testbed separately, etc.
Example of the experiment lifecycle

**Resource discovery**
- Show me all resources available in the Fed4FIRE federation

**Resource requirements**
- Limit to nodes that have 2 IEEE 802.11n interfaces

**Resource reservation**
- Reserve me 30 nodes on testbed X for 10 hours

**Resource provisioning**
- Make sure they will be deployed with Ubuntu 12.04 LTS

**Experiment control**
- After 10 s, start data stream of 10 Mbps with source node 1, after 30 s start second data stream of 5 Mbps with source node 5.

**Monitoring**
- Facility monitoring: crucial servers up and running? → testbed up and running
- Infrastructure monitoring: CPU load, number of transmit errors
- Experiment measurement: measure end-to-end throughput, delay and jitter.

**Permanent storage**
- Store measurements on the storage server of testbed X for later analysis

**Resource release**
- I’m done with them after 5h already, release my resources so they can be used by other experimenters.
How to configure a Wall experiment

Emulab configuration (AIMS 2013)

jFed Experimenter (Fed4FIRE)
Common tool: jFed

- Java based framework supporting SFA testbed federation client tools
- Includes automated testing tools and an experimenter tool
Resource Specifications (RSpec)

- RSpecs are XML documents that describe resources
  - Machines, VMs, links, etc.

RSpec for a physical machine with one interface:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rspec type="request" xsi:schemaLocation="http://www.geni.net/resources/rspec/3 ... xmlns=http://www.geni.net/resources/rspec/3">
  <node client_id="node0" component_manager_id="urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm" exclusive="true">
    <sliver_type name="raw-pc"/>
  </node>
</rspec>
```
RSpecs

- RSpec documents are exchanged by experimenter tools (e.g. jFed) and testbeds
  - Testbeds use RSpecs to describe what they have – Advertisement RSpecs
  - Experimenters use RSpecs to describe the resources they want – Request RSpecs
  - Testbeds use RSpecs to describe the resources allocated to an experimenter – Manifest RSpecs
The AM API

- Experimenter tools and testbeds talk to each other using the Aggregate Manager API (AM API)
Working today

Internet2 is running a GENI AM for ION Project

Belnet

Geant

250 vlans

10 vlans

ProtoGENI Utah

PG RACK

BBN RACK

SLICE A

SLICE B

SLICE C

SLICE D

PG RACK

GENI Rack

Regional Network

Campus Network

GENI Aggregate

Wide Area Network

ProtoGENI CRON

LONI

KyRON

UMD

ProtoGENI UKY

LSU

UKY

MAX RACK

MAX

NOX
Conclusions

- A wide range of available European testbed facilities circumvent need for time-consuming manual testbed configuration and setup
- Virtual Wall available for iMinds project partners (or get in touch with us for other options)
- Federation of European testbeds: Fed4FIRE: Wide range of wired and wireless testbeds
Executing a basic Virtual Wall experiment using jFed
Create SSH key

- **Linux/Unix**
  - Create SSH key using `ssh-keygen -t rsa`
- **Windows**
  - Use PuTTYgen to create SSH keys
  - Download from: http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.63-installer.exe
Create SSH key
Create SSH key
Create SSH key
Create SSH key
Create User

Emulab is a network testbed, giving researchers a wide range of environments in which to develop, debug, and refine both to a facility and to a software system. The primary Emulab installation is run by the Flux Group, part of Utah. There are also installations of the Emulab software at more than two dozen sites around the world, ranging to testbeds with hundreds of nodes. Emulab is widely used by computer science researchers in the fields of network design to support education, and has been used to teach classes in those fields.

Emulab is a universally available time- and space-shared network emulator which achieves new levels of ease combined with secure, user-friendly web-based tools, and driven by ns-compatible scripts or a Java GUI, allow you to design and link down to the hardware level. Packet loss, latency, bandwidth, queue sizes—all can be user-defined and securely replaced with custom images by any experimenter. Emulab can load tens or a hundred disks in less than the control and ease of use of simulation, without sacrificing the realism of emulation and live network experiments.

Links to help you get started:

- Authorization Scheme, Policy, and "How To Get Started"
- Overview of Installed Software
- Hardware Overview, "Emulab Classic"
- Security Issues
- Administrative Policies and Disclaimer
Create User

Request a New Emulab Account

If you already have an Emulab account, please log on first!

Join an Existing Project

or

Start a New Project.

If you are a student (undergrad or graduate), please do not try to start a project!
Your advisor must do it.
Create User

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>nbouten</td>
</tr>
<tr>
<td>Full Name</td>
<td>Niels Bouten</td>
</tr>
<tr>
<td>Job Title/Position</td>
<td>PhD Student</td>
</tr>
<tr>
<td>Institutional Affiliation</td>
<td>Ghent University – iMinds</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>UGent (e.g. MIT)</td>
</tr>
<tr>
<td>Home Page URL</td>
<td><a href="http://intec.ugent.be">http://intec.ugent.be</a></td>
</tr>
<tr>
<td>Email Address[1]</td>
<td><a href="mailto:nbouten@intec.ugent.be">nbouten@intec.ugent.be</a></td>
</tr>
<tr>
<td>Postal Address: Line 1</td>
<td>Gaston Crommenlaan 8</td>
</tr>
<tr>
<td>Line 2</td>
<td></td>
</tr>
<tr>
<td>City</td>
<td>Ledeberg</td>
</tr>
<tr>
<td>State/Province</td>
<td></td>
</tr>
<tr>
<td>ZIP/Postal Code</td>
<td>9050</td>
</tr>
<tr>
<td>Country</td>
<td>Belgium</td>
</tr>
<tr>
<td>Phone #</td>
<td>+32498330533</td>
</tr>
<tr>
<td>Password[1]</td>
<td>••••••</td>
</tr>
<tr>
<td>Retype Password</td>
<td>••••••</td>
</tr>
<tr>
<td>Geni Account</td>
<td></td>
</tr>
<tr>
<td>Geni SSL Pass Phrase[3]</td>
<td>••••••</td>
</tr>
<tr>
<td>Retype Geni Pass Phrase</td>
<td>••••••</td>
</tr>
</tbody>
</table>

Password for wall2 portal
Password for Geni certificate
Create User

<table>
<thead>
<tr>
<th>Project Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Name:</strong> AIMSTutorial</td>
</tr>
<tr>
<td><strong>Group Name:</strong> (Leave blank unless you know the group name)</td>
</tr>
</tbody>
</table>

(See our Users page)

<table>
<thead>
<tr>
<th>Will you add a link on your project page to <a href="http://www.wall2.ilabt.iminds.be">www.wall2.ilabt.iminds.be</a>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Funding Sources and Grant Numbers: (Type &quot;none&quot; if not funded)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fed4Fire</td>
</tr>
</tbody>
</table>

| Estimated # of Project Members[2]: |
|----------------|---|
| 4 |

| Estimated # of PCs[2]: |
|----------------|---|
| 30 |

<table>
<thead>
<tr>
<th>Please describe how and why you'd like to use the testbed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>open call experiment for Fed4Fire</td>
</tr>
</tbody>
</table>

Submit
Dear Brecht Vermeulen (bvermeu5):

This is your account verification key: 81383e282f9bc5191a095f1d729958c2

Please use this link to verify your user account:


You will then be verified as a user. When you have been both verified and approved by Testbed Operations, you will be marked as an active user and granted full access to your account. You MUST verify your account before your project can be approved!

Thanks,
Testbed Operations
Create User

Confirm Verification

Done!

You have now been verified. However, your application has not yet been approved. You will receive email when that has been done.
Add SSH key to emulab

Enter ssh public keys for user aims01 [1,2]
(We strongly encourage the use of Protocol 2 keys only! [6])

Upload Public Key[3,4]: (4K max)
Password[5]:

Add New Keys

Last Users Login: N/A
Last Node Login: N/A
Download Fed4FIRE certificate

Download SSL Certificate for user: bvermeu5

Download your certificate and private key in PEM format, and then save it to a file in your .ssl directory.

You can also download it in pkcs12 format for loading into your web browser (if you do not know what this means, or you wish to load your certificate using your web browser).

We have also created a SSH key pair for you, derived from your new ssl certificate, using the same pass phrase. You will need to use this key pair for certificate validation. The private key is typically placed in your .ssh directory on your desktop machine. If you are running an agent such as SSH, some programs might also need to know about the corresponding public key.
jFed tool installation

- You need a recent version of Java 7 (http://www.java.com/verify)
- Go to http://jfed.iminds.be
jFed tool installation

For OS X, you may need to change security settings
jFed tool installation

Browse for Geni .pem file

Enter Geni password
jFed tool installation

Welcome to the jFed Experimenter Toolkit!

Initial configuration required

In order to successfully run your an experiment, you need to configure your Public-Key Authentication settings.

Please complete your settings before continuing.

Ok
jFed tool installation (Windows)
jFed tool installation (Unix/Mac)
jFed tool

No experiments are open.

Create a new experiment description (CTRL+N)
Open an existing experiment description (CTRL+O)
Open an existing experiment description from an URL (CTRL+SHIFT+O)
Create Experiment – Select testbed
Create Experiment – Select testbed
Throughput / Latency / Packet loss

2 ways

- By using tc at both endpoints of the links
- By using impairment nodes with OpenBSD, configured as a bridge
Link configuration
Adapt settings link

- tc script on each node located at
  
  ```bash
  % /var/emulab/boot/rc.linkdelay
  ```

- Modify settings
  
  ```bash
  % sudo /sbin/tc class change ...  
  % sudo /sbin/tc qdisc change ...
  ```
Bridge configuration
Adapt settings bridge

- Login to bridge using jFed

- List settings
  
  ```
  % sudo ipfw pipe show
  ```

- Adjust settings
  
  ```
  % sudo ipfw pipe nr config bw 10Mbit/s plr 0.0 delay 40ms
  ```
Advanced use with RSpec

Install Specific Disk Image

```xml
<sliver_type name="raw-pc">
  <disk_image name="urn:publicid:IDN+wall2.ilabt.iminds.be+image+emulab-ops//DEB60_64-VLAN"/>
</sliver_type>
```

Install Software when provisioning, run script after provisioning

```xml
<sliver_type name="raw-pc"/>
<services>
  <execute command="sudo apt-get install dstat" shell="sh"/>
  <install install_path="/local" url="http://download.videolan.org/pub/videolan/vlc/0.5.1/vlc-0.5.1.tar.gz"/>
</services>
```

Specify link configurations

```xml
<link client_id="link2">
  <component_manager name="urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm"/>
  <link_type name="lan"/>
  <property source_id="node1:if0" dest_id="node0:if0" capacity="10000" latency="20" packet_loss="0.01"/>
  <property source_id="node0:if0" dest_id="node1:if0" capacity="20000" latency="20" packet_loss="0.01"/>
  <interface_ref client_id="node1:if0"/>
  <interface_ref client_id="node0:if0"/>
</link>
```
Shared Storage (per testbed)

- Shared folder of nodes is available via
  
  /groups/wall2-ilabt-iminds-be/projectname

  - This share is mounted on all nodes automatically
  - This share is permanent, data stored here will remain available when experiment is done
  - This share is shared with all people in the project
Advanced use Alpha jFed

- Support for command timeline
- Add barriers to command execution
- Execute instant commands
Create timed commands
Add Barriers
Start execution and follow output
Execute instant commands
Execute Instant commands
Follow output of instant commands
OpenFlow protocol basics
Traditional network switch
OpenFlow-enabled switch

Externally controlled Switch

OpenFlow Controller

Build information
Programmatic

Store information
Policy, Topology

Forwarding Decision

OpenFlow Interface API

Data Plane

Forwarding Path

Message Bus

Port 1

Port 2

BRAD HEDLUND .com
The OpenFlow protocol and controller

- Controller is responsible for populating flow table of the switch
- In case of a table miss the switch can, for example
  - Forward packet to controller
  - Drop the packet
OpenFlow ports

- Network interfaces for passing packets to and from OpenFlow switch
- Packets enter the switch through ingress ports
  - Ingress ports can be used to match packets
- Packets leave the switch through output ports
  - Packets can be sent to specific output port as action
- Several port types available
  - Physical: Correspond to specific hardware interface
  - Logical: e.g., tunnels, aggregation groups, loopback
  - Reserved: e.g., controller, flooding, normal switching
5.1 Pipeline Processing

In this case pipeline processing is greatly simplified. A switch must have at least one flow table. An OpenFlow switch with only a single flow table is valid, with those flow tables (see Figure 2). An OpenFlow switch is required to have at least one flow table.

The OpenFlow pipeline processing defines how packets interact with those flow tables, and can optionally have more flow tables. The OpenFlow pipeline processing is used for ACL and QoS processing. Those switches should provide a classification mechanism outside of OpenFlow-hybrid switches, i.e. traditional L2 Ethernet switching, VLAN isolation, L3 routing (IPv4 routing, IPv6 routing...), and other classification mechanisms.

OpenFlow-hybrid switches support both OpenFlow operation and Ethernet switching operation. In those switches all packets are processed by the OpenFlow pipeline, and can not be processed otherwise.

OpenFlow-compliant switches come in two types: OpenFlow-only switches support only OpenFlow operation, in those switches all packets are processed by the OpenFlow pipeline or the other, or it may direct all packets to the OpenFlow pipeline. This classification mechanism is outside the scope of this specification.

An OpenFlow-hybrid switch may also allow a packet to go from the OpenFlow pipeline to the normal pipeline through the classification mechanism.

Figure 2:

- Packets are matched against multiple tables in the pipeline:
  - (a) Per-table packet processing
  - (b) Find highest-priority matching flow entry

- Execute instructions:
  - i. Modify packet & update match fields (apply actions instruction)
  - ii. Update action set (clear actions and/or write actions instructions)
  - iii. Update metadata

- Send match data and action set to next table
## OpenFlow table entries

<table>
<thead>
<tr>
<th>Match Fields</th>
<th>Priority</th>
<th>Counters</th>
<th>Instructions</th>
<th>Timeouts</th>
<th>Cookie</th>
<th>Flags</th>
</tr>
</thead>
</table>

1. Write Metadata
2. Goto Flow Table
3. Write action(s) to action set
   1. Output: Send packet to specified port
   2. Drop
   3. Set-Queue: Assign packet to specified queue
   4. Set-Field: Modify packet header field(s)
   5. Change-TTL

### Ingress port

<table>
<thead>
<tr>
<th>Packet header fields</th>
<th>Pipeline Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OpenFlow software components

- Controller
  - NOX
  - POX
  - Beacon
  - Floodlight

- Slicing
  - FlowVisor

- Switches
  - Open vSwitch
  - Pica8
  - Pantou
OpenFlow controller implementations

- **NOX**
  - Efficient C++ based controller for Linux
  - [http://www.noxrepo.org/](http://www.noxrepo.org/)

- **POX**
  - Python-based version of NOX
  - Less efficient but useful for rapid prototyping

- **Beacon**
  - Modular Java-based OpenFlow controller
  - [https://openflow.stanford.edu/display/Beacon/Home](https://openflow.stanford.edu/display/Beacon/Home)

- **Floodlight**
  - Java-based, similar to Beacon
  - [http://www.projectfloodlight.org/floodlight/](http://www.projectfloodlight.org/floodlight/)
Switch slicing with FlowVisor

- Normally: One controller per switch
- FlowVisor is a proxy that lets multiple controllers manage one switch
- Supports virtualization by splitting switch into slices, each managed by different controller
OpenFlow software switches

- **Open vSwitch**
  - C/Python software switch with OpenFlow support
  - Integrated into the GNU/Linux kernel
  - http://openvswitch.org/

- **Pica8**
  - C-based hardware-agnostic switch operating system with OpenFlow support

- **Pantou**
  - OpenFlow implementation for OpenWRT-based wireless routers
OpenFlow simulation with Mininet

- Mininet creates a virtual network on one or multiple computers
- The network consists of virtual machines running real code on top of GNU/Linux
- Supports OpenFlow through integration of Open vSwitch
- Many OpenFlow controllers can be used, including NOX/POX
- Developed controller code can be easily transferred to testbed or real deployment
Running OpenFlow experiments on the Virtual Wall
Loading RSpec

- Rspec at http://jfed.iminds.be/ovs.rspec
- Topology

Diagram:
- POX
- OVS
- Host 1
- Host 2
- Host 3
Loading RSpec

Open URL

Open Experiment Definition

Please enter the URL:
http://jfed.iminds.be/ovs.rspec

Find wall2 & Replace by wall1 in RSpec
Configure switch

- Ethernet bridge acting as software switch was added during configuration
  
  ```
  % sudo ovs-vsctl list-br
  ```

- Add interfaces to this bridge that will act as ports of the software switch
  
  ```
  % sudo ifconfig eth1 0
  % sudo ovs-vsctl add-port br0 eth1
  % sudo ovs-vsctl list-ports br0
  ```

- Check interface-host mapping with ping
Point switch to controller

- The controller can be hosted anywhere, here it is on localhost
  
  `% sudo ovs-vsctl set-controller br0 tcp:127.0.0.1:6633`

- Standalone vs secure mode
  
  `% sudo ovs-vsctl set-fail-mode br0 secure`
  `% sudo ovs-vsctl set-fail-mode br0 standalone`

- What is the difference?
Learning controller

- Try to ping between hosts
- Start learning controller

```bash
% cd /local/pox
% ./pox.py --verbose SimpleL2Learning
```

```
node2:~$ ping node3
PING node3-link6 (10.0.1.3) 56(84) bytes of data.
From node2-link5 (10.0.1.2) icmp_seq=1 Destination Host Unreachable
From node2-link5 (10.0.1.2) icmp_seq=2 Destination Host Unreachable
From node2-link5 (10.0.1.2) icmp_seq=3 Destination Host Unreachable
From node2-link5 (10.0.1.2) icmp_seq=4 Destination Host Unreachable
From node2-link5 (10.0.1.2) icmp_seq=5 Destination Host Unreachable
From node2-link5 (10.0.1.2) icmp_seq=6 Destination Host Unreachable
64 bytes from node3-link6 (10.0.1.3): icmp_req=7 ttl=64 time=2008 ms
64 bytes from node3-link6 (10.0.1.3): icmp_req=9 ttl=64 time=0.428 ms
64 bytes from node3-link6 (10.0.1.3): icmp_req=8 ttl=64 time=1038 ms
64 bytes from node3-link6 (10.0.1.3): icmp_req=10 ttl=64 time=0.440 ms
64 bytes from node3-link6 (10.0.1.3): icmp_req=11 ttl=64 time=0.479 ms
```
Soft vs Hard Timeouts

- **Soft Timeout**
  - If no matching packets received, how long will flow remain in forwarding table

- **Hard Timeout**
  - Total time a flow will remain in forwarding table, independent of matching packets are received

```python
msg.match = of.ofp_match.from_packet(self.packet, self.event.port)
msg.idle_timeout = 10
msg.hard_timeout = 30
```
Traffic duplication

- Check traffic on each interface of switch using tcpdump
  
  ```
  % sudo tcpdump -i <data_interface_name>
  ```

- Duplicate all traffic and send to host3
  
  ```
  % ./pox.py --verbose DuplicateTraffic
  --duplicate_port=[eth_host3]
  ```
Traffic duplication

class DuplicateTrafficSwitch(SimpleL2LearningSwitch):
    def __init__(self, connection, duplicate_port):
        SimpleL2LearningSwitch.__init__(self, connection, False)
        self._connection = connection;
        self._duplicate_port=duplicate_port
        self._of_duplicate_port=getOpenFlowPort(connection, duplicate_port)

    def _handle_PacketIn(self, event):
        log.debug("Got a packet : " + str(event.parsed))
        self.packet = event.parsed
        self.event = event
        self.macLearningHandle()
        out_port = self.get_out_port()
        self.forward_packet([out_port, self._of_duplicate_port])

class DuplicateTraffic(object):
    def __init__(self, duplicate_port):
        core.openflow.addListeners(self)
        self._duplicate_port= duplicate_port

    def _handle_ConnectionUp(self, event):
        log.debug("Connection %s" % (event.connection,))
        DuplicateTrafficSwitch(event.connection, self._duplicate_port)

    def launch(duplicate_port="eth4"):
        log.debug("DuplicateTraffic" + duplicate_port);
        core.registerNew(DuplicateTraffic, duplicate_port)
Port forward controller

- Redirect traffic to different port
- Run netcat on port 5000 and 6000 on host2
  
  ```
  % nc -l 5000
  ```

- Check with learning controller if connection works
  
  ```
  % nc 10.0.1.2 5000
  ```

- Update ext/port_forward.config
- Run portforwarding controller
  
  ```
  % ./pox.py --verbose PortForwarding
  ```

- Check if port forwarding works
Server proxy controller

- Redirect traffic to other host
- Run netcat on port 5000 and 7000 on host2 and host3 respectively
  
```
  % nc -l 5000
```

- Update ext/proxy.config
- Run proxy controller
  
```
  % ./pox.py --verbose Proxy
```

- Check if proxy works
Further reading and contact information

- Fed4FIRE
  - Website: http://www.fed4fire.eu
  - Documentation: http://doc.fed4fire.eu
  - Portal: http://portal.fed4fire.eu
  - jFed: http://jfed.iminds.be

- Maxim Claeys: maxim.claeys@intec.ugent.be
- Jeroen Famaey: jeroen.famaey@intec.ugent.be
- Niels Bouten: niels.bouten@intec.ugent.be