

GIMI/LabWiki Tutorial

Mike Zink, Divyashri Bhat,
Cong Wang, Thierry Rakotoarivelo
GEC20

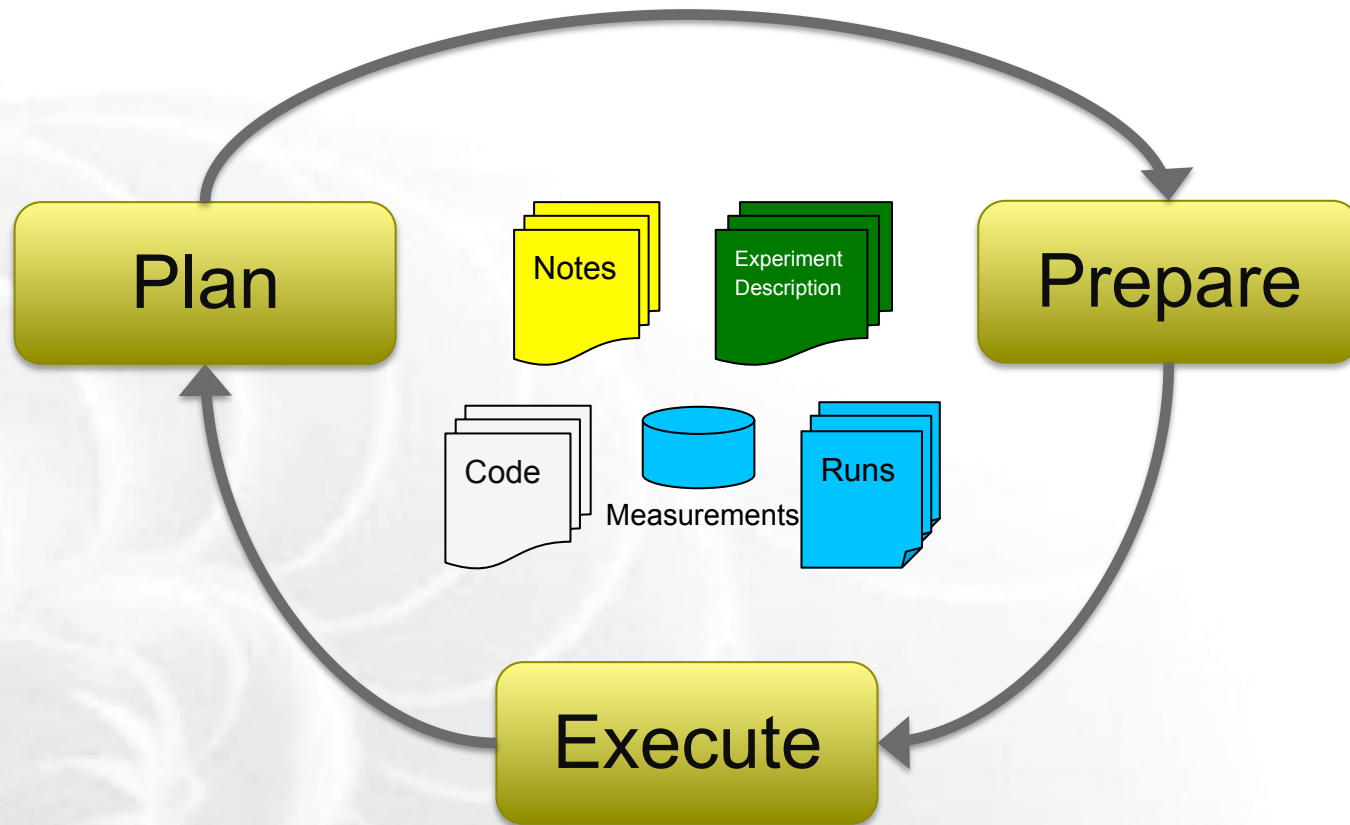
March 22nd 2014, UC Davis, CA



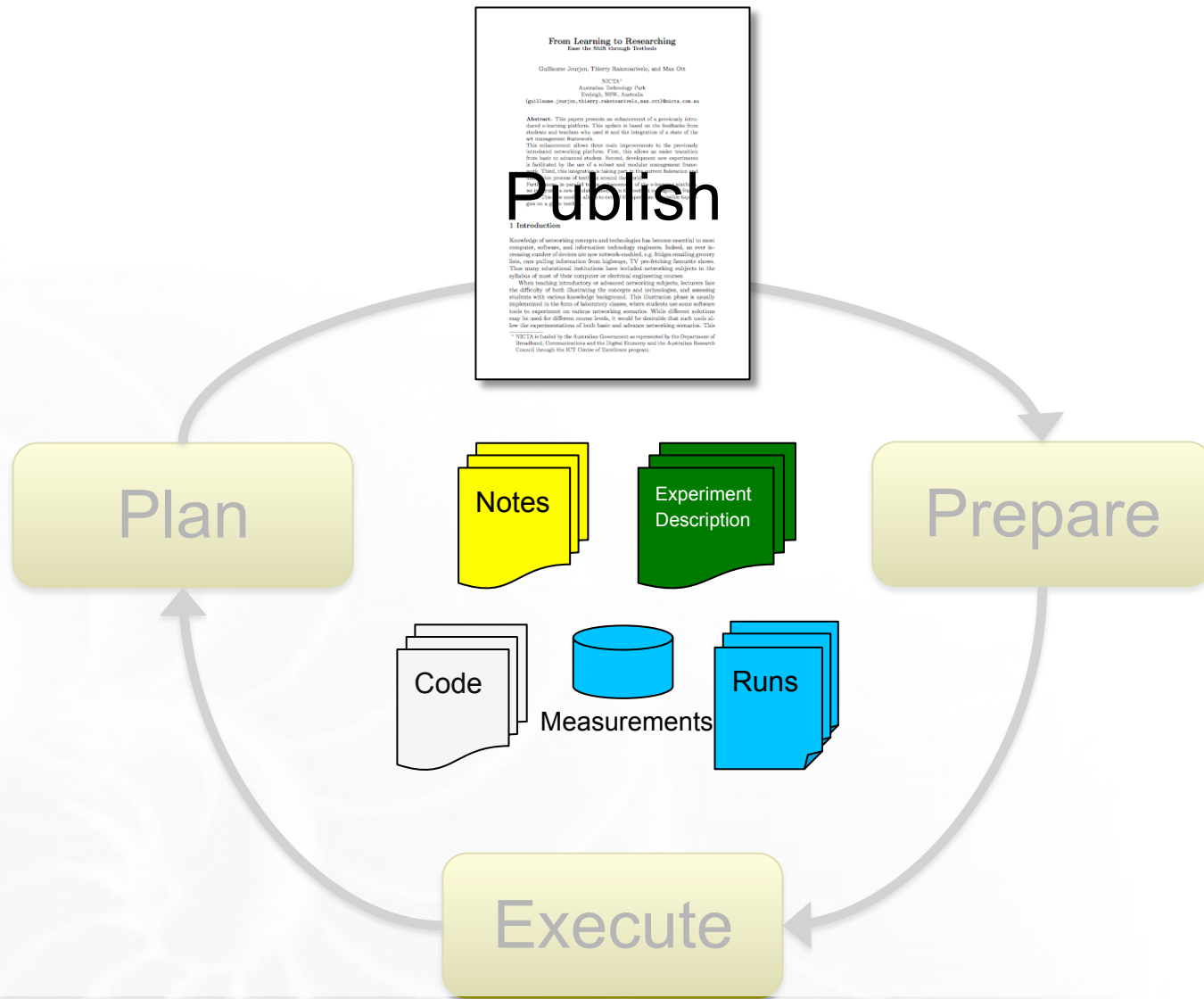
“Perform basic measurements on GENI testbeds”

- ✓ GENI slice (network & compute resources)
- ✓ Repeatable experiment
- ✓ Ability to monitor and document experiment

The “Experiment Cycle”



The “Successful Experiment Cycle”



localhost:4000/labwiki
6444

LabWiki
user1 Log out


Plan

Prepare

Execute

Tutorial: First Experiment

As mentioned before, we want to configure an experiment as shown below:



The first step is to describe the experiment in OEDL, the OMF Experiment Description Language. To see how this looks for this experiment, open the '1_hello.rb' file in the **Prepare** column.

Ignoring some of the details we can see the definition of two resource groups, **Sender** in line 6 and **Receiver** in line 21.

```
6: defGroup('Sender', ...
...
21: defGroup('Receiver', ...
```

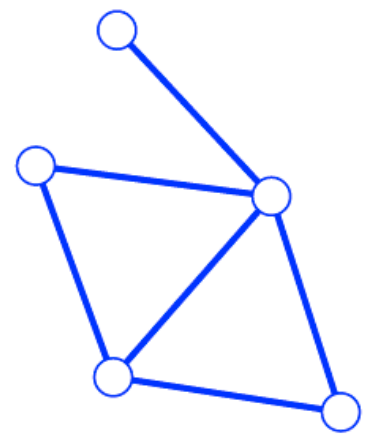
There will be more on groups later, but in this

git:default:repo/oid/tutorial/1_hello.o

```

1 defProperty('res1', 'omf.nicta.node1', "
2 defProperty('res2', 'omf.nicta.node2', "
3 essid = (0...8).map{65.+(rand(25)).chr}.
4 channel = rand(11)+1
5
6 defGroup('Sender', property.res1) do |no
7   node.addApplication("test:app:otg2") d
8   app.setProperty('udp:local_host', '1
9   app.setProperty('udp:dst_host', '192
10  app.setProperty('udp:dst_port', 3000
11  #app.measure('udp_out', :interval =>
12  app.measure('udp_out', :samples => 1
13  end
14  node.net.w0.mode = "adhoc"
15  node.net.w0.type = 'g'
16  node.net.w0.channel = channel
17  node.net.w0.essid = essid
18  node.net.w0.ip = "192.168.0.2"
19  end
20
21 defGroup('Receiver', property.res2) do |
22  node.addApplication("test:app:otr2") d
23  app.setProperty('udp:local_host', '1
24  app.setProperty('udp:local_port', 30
25  #app.measure('udp_in', :interval =>
```

Slice 70256298659140





Plan

Prepare

Execute



Search



Slice 7...5298659140

Tutori

As mentioned before, we want to configure an experiment as shown below:



Wiki

The first step in configuring an experiment in OEDL, the OEDL description Language.

For this experiment, open the '1_hello.rb' file in the Prepare column.

Ignoring some of the details we can see the definition of two resource groups, Sender in line 6 and Receiver in line 21.

```

6: defGroup('Sender', ...
...
21: defGroup('Receiver', ...
  
```

There will be more on groups later, but in this

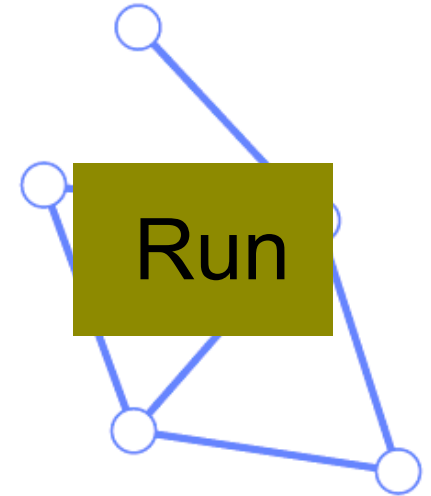
git:default:repo/oidl/tutorial/1_hello.o



```

1 defProperty('res1', 'omf.nicta.node1', "
2 defProperty('res2', 'omf.nicta.node2', "
3   ssid = (0...8).map(65.+(rand(25)).chr).
4   channel = rand(11)+1
5
6 defGroup('Sender', property.res1) do |no
7   node.addApplication("test:app:otg2") d
8   app.s
9   app.s
10  app.s
11  #app.
12  app.n
13 end
14 node.net.w0.mode = "adhoc"
15 node.net.w0.type = 'g'
16 node.net.w0.channel = channel
17 node.net.w0.essid = ssid
18 node.net.w0.ip = "192.168.0.2"
19 end
20
21 defGroup('Receiver', property.res2) do |
22   node.addApplication("test:app:otr2") d
23   app.setProperty('udp:local_host', '1
24   app.setProperty('udp:local_port', 30
25   #app.measure('udp_in', :interval =>
  
```

Edit



Run

LabWiki Core

Plan

Plugin



Prepare



Your Plugin

Execute



GENI
CH/AM

OMF

GIMI
Services

iRODS

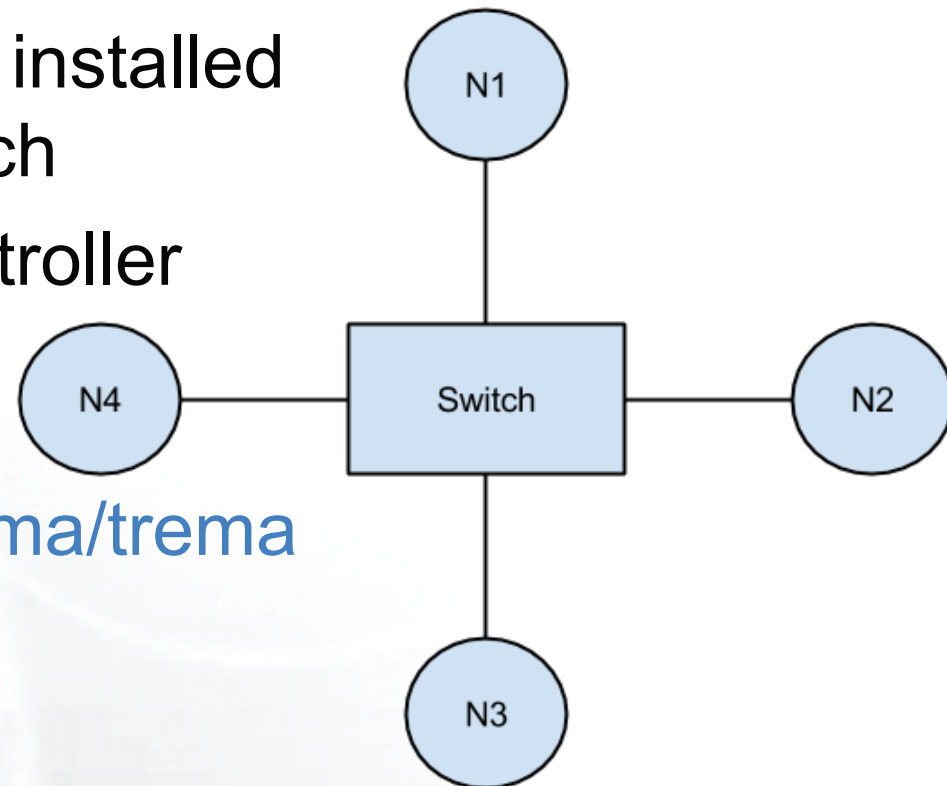
Your
Service

<https://github.com/mytestbed/labwiki>

- In earlier tutorials:
 - GENI account and credentials
 - iRODS account (set up via portal)
 - Know what resources are available
 - How to create a slice request (Rspec)
 - How to reserve a slice
 - How to log into VMs of your slice
- Familiar with GENI Portal!

Example Experiment Topology

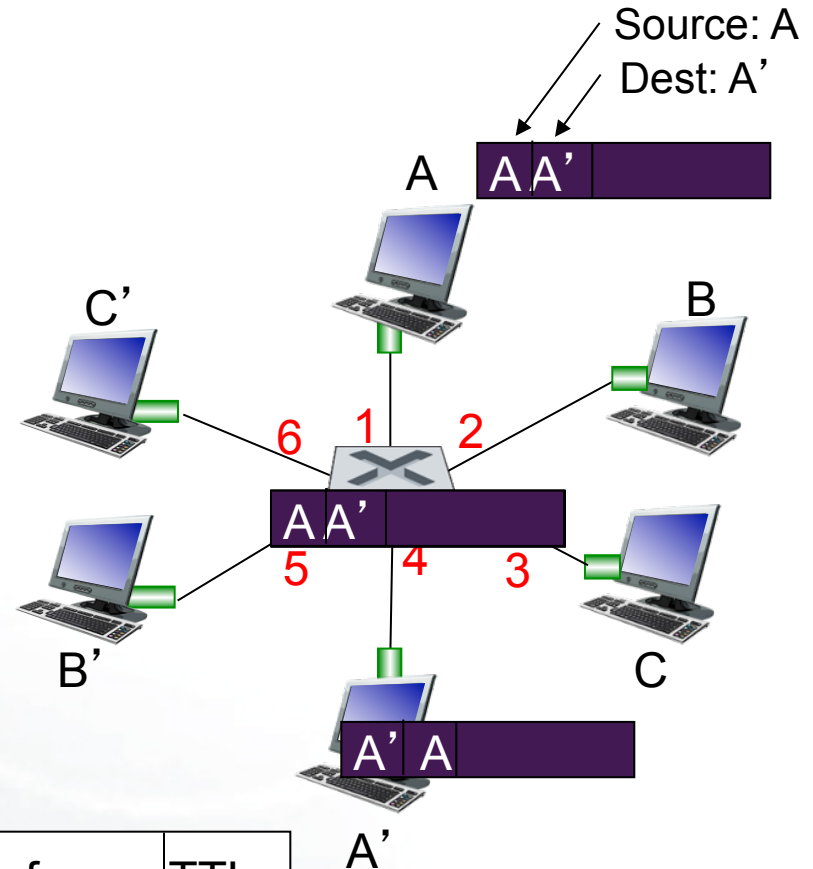
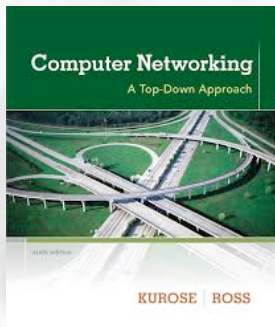
- N1 – N4: regular end systems
- Switch: VM with OVS installed
→ sw-based OF switch
- Trema-based OF controller running on Switch



<https://github.com/trema/trema>

Goal: Implement learning switch and verify its correct behavior

- frame destination unknown: *flood*
- ❖ destination A location known: *selective send*
- ❖ More info in chapter 6 of “Computer Networks”, Kurose & Ross



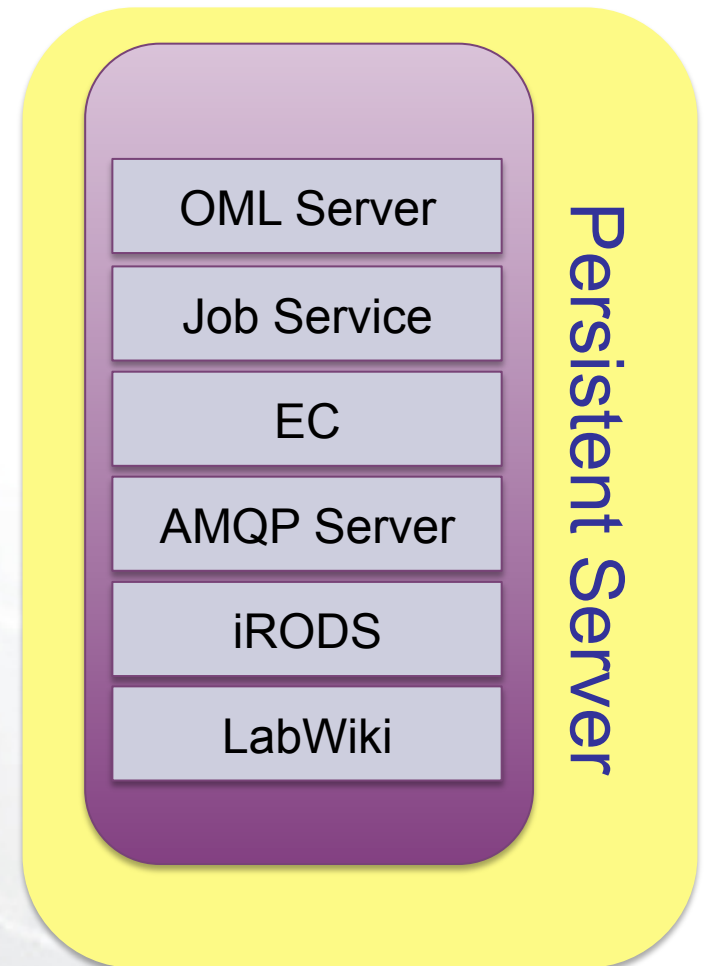
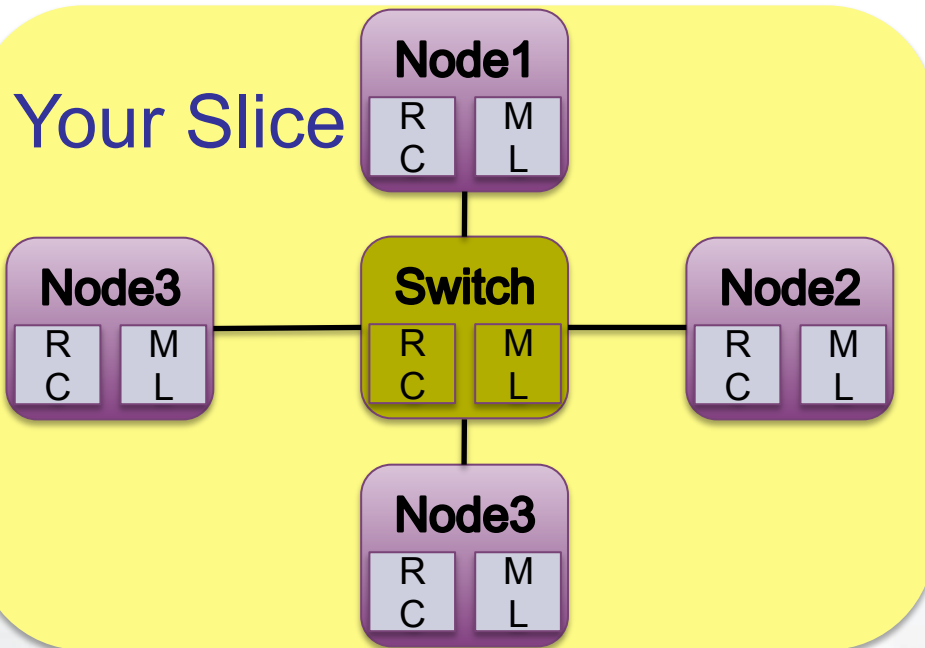
MAC addr	interface	TTL
A	1	60
A'	4	60

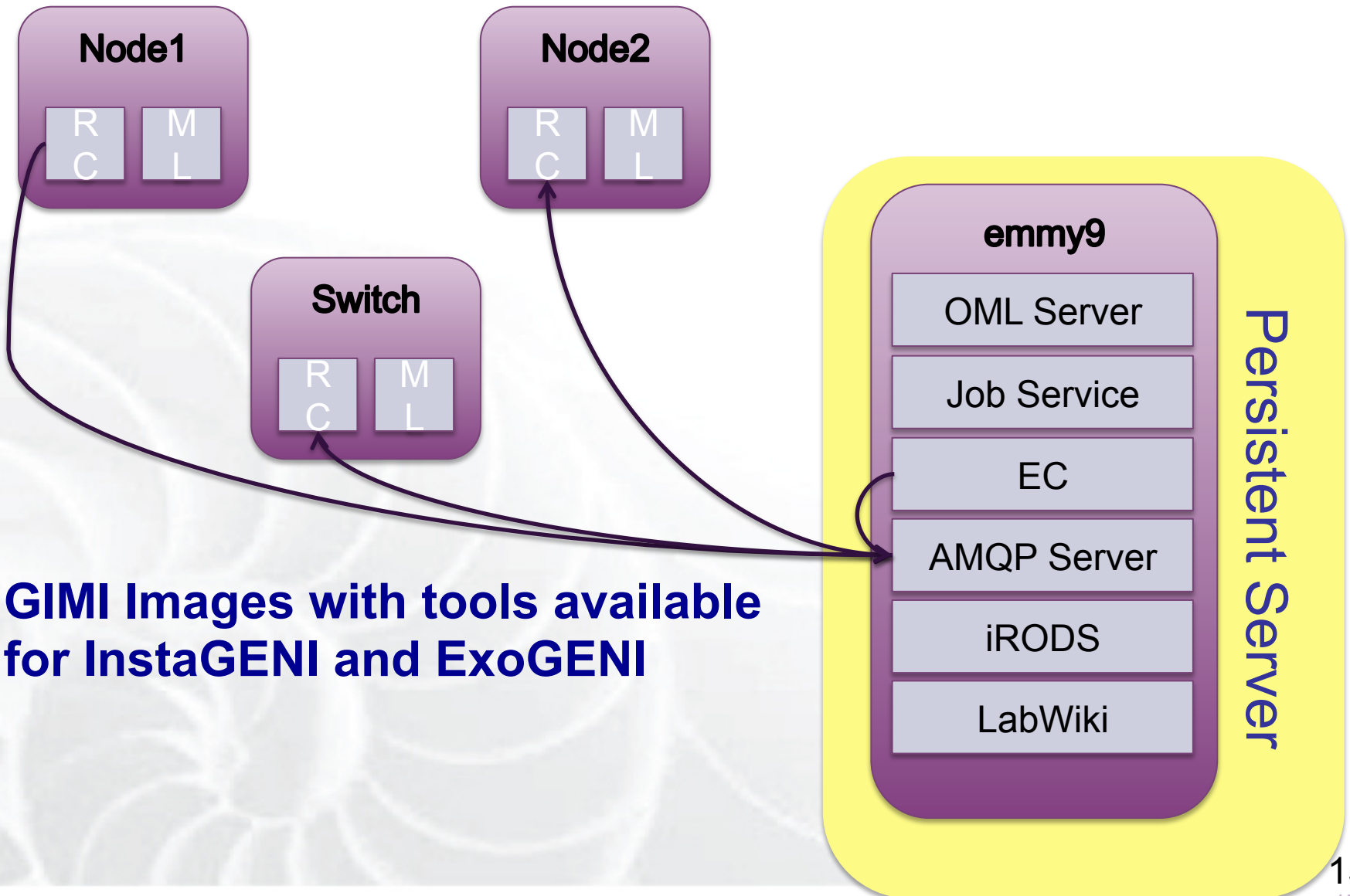
*Switch table
(initially empty)*

Module A

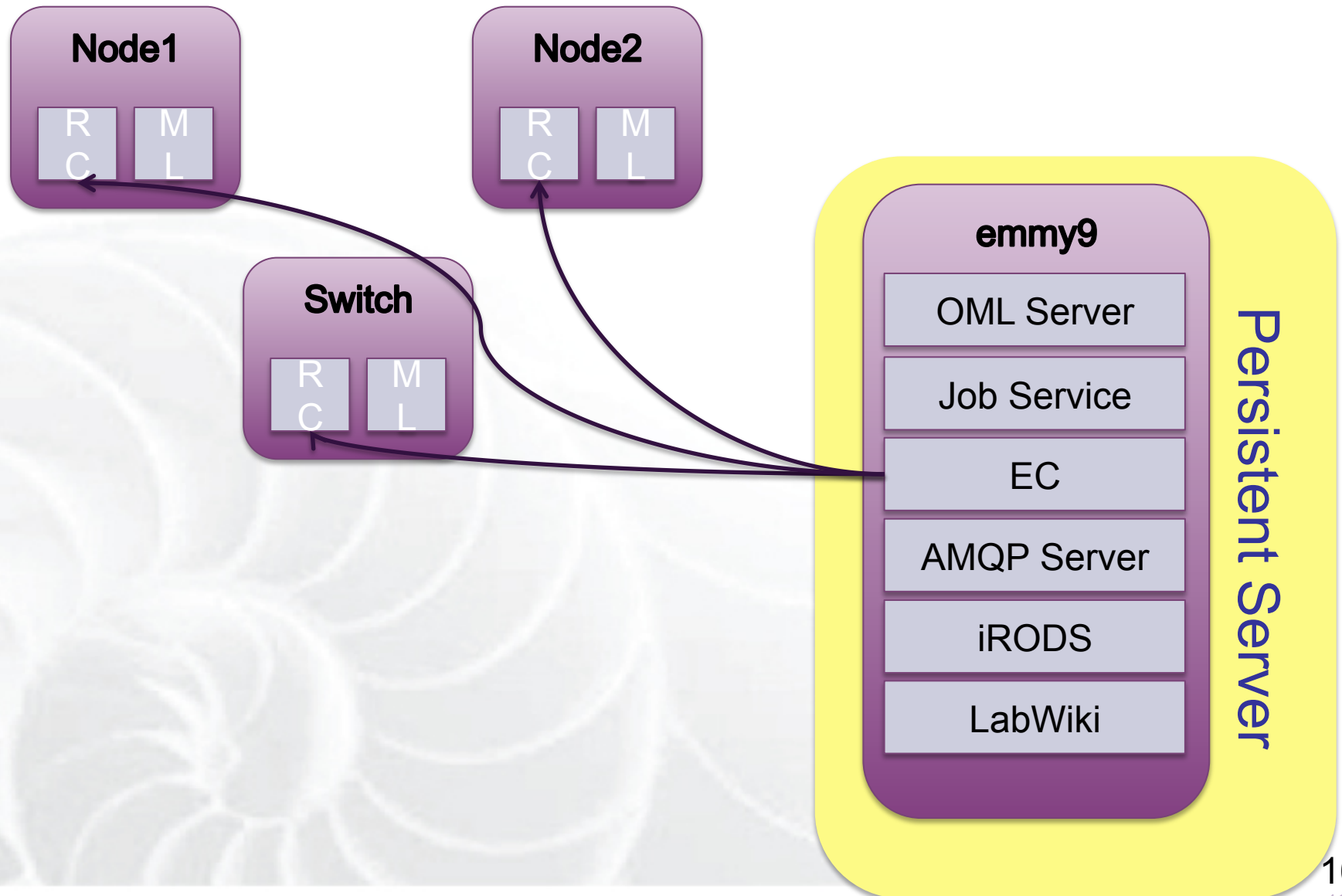
[http://groups.geni.net/geni/wiki/
GEC20Agenda/LabWiki/ModuleA](http://groups.geni.net/geni/wiki/GEC20Agenda/LabWiki/ModuleA)

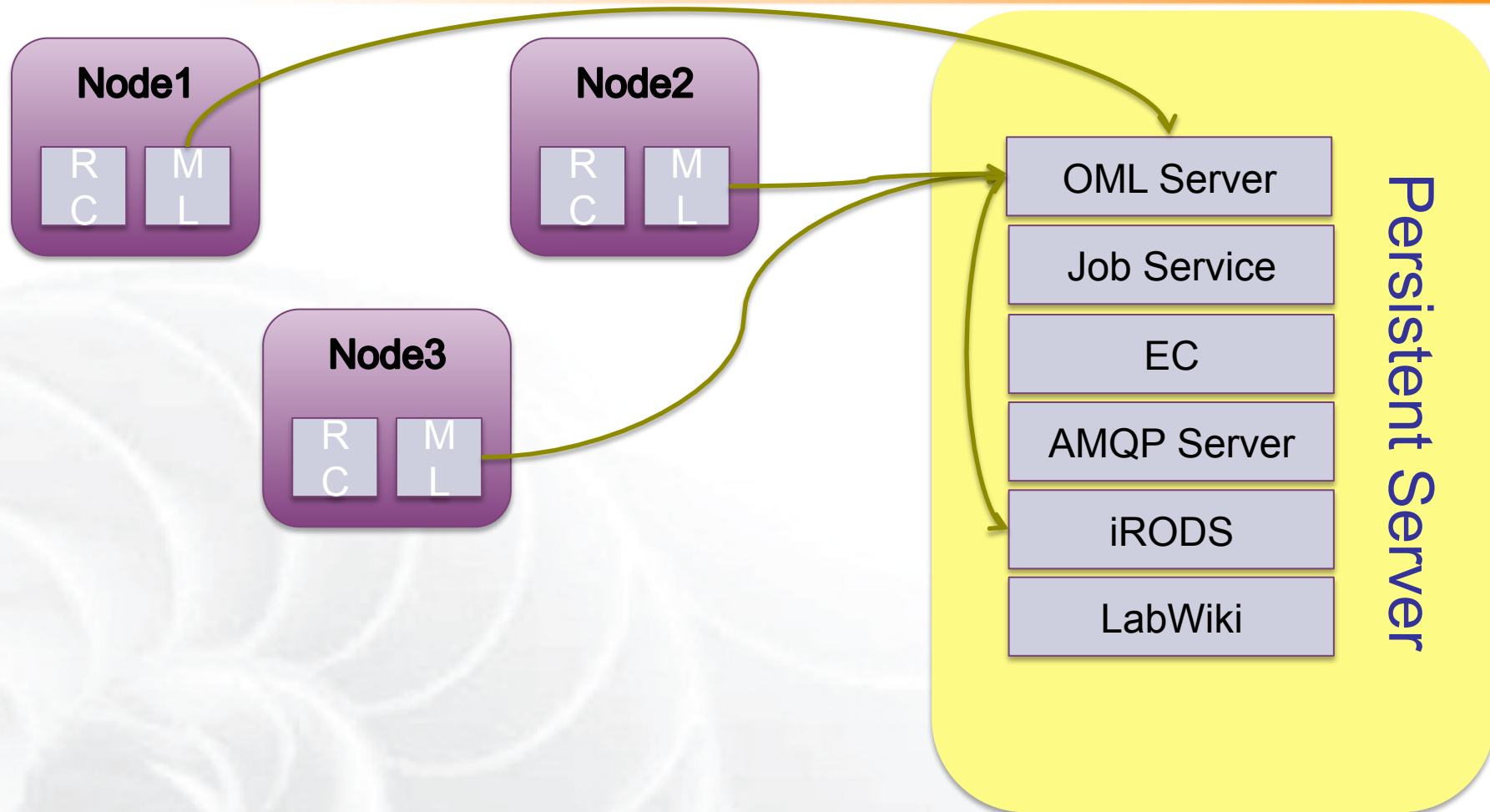
Module B

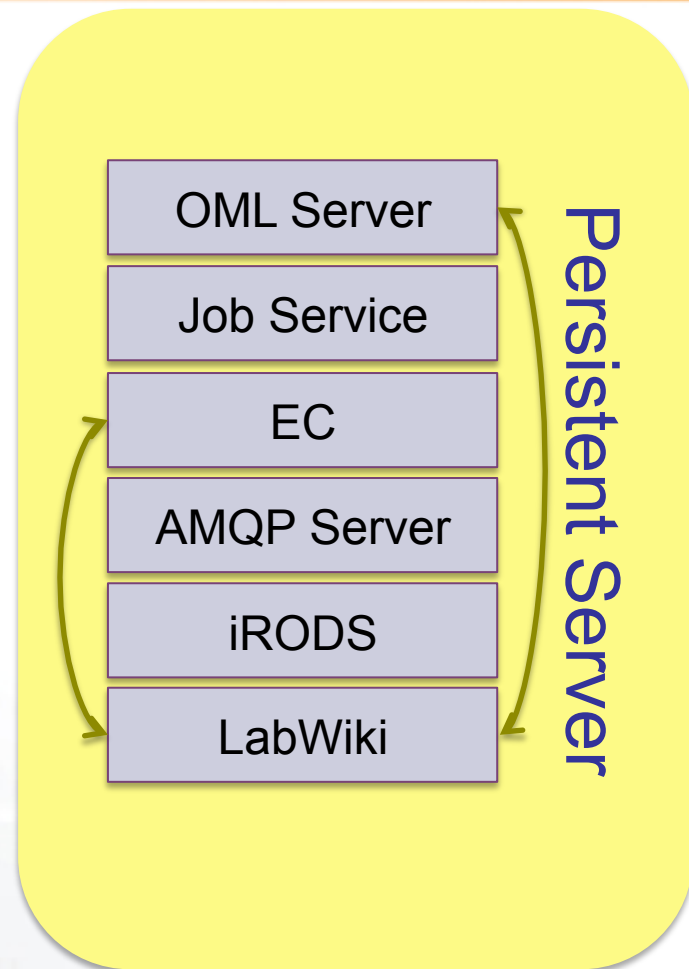
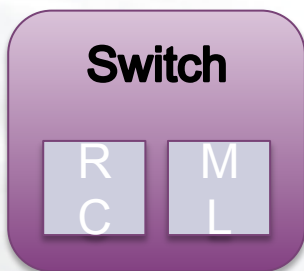
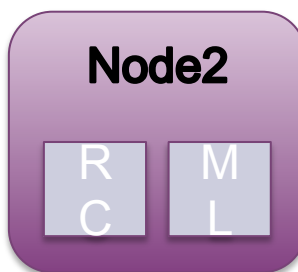
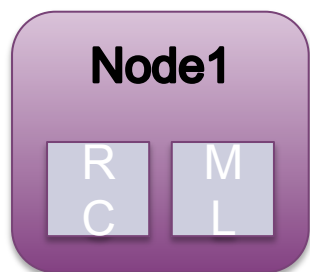




**GIMI Images with tools available
for InstaGENI and ExoGENI**



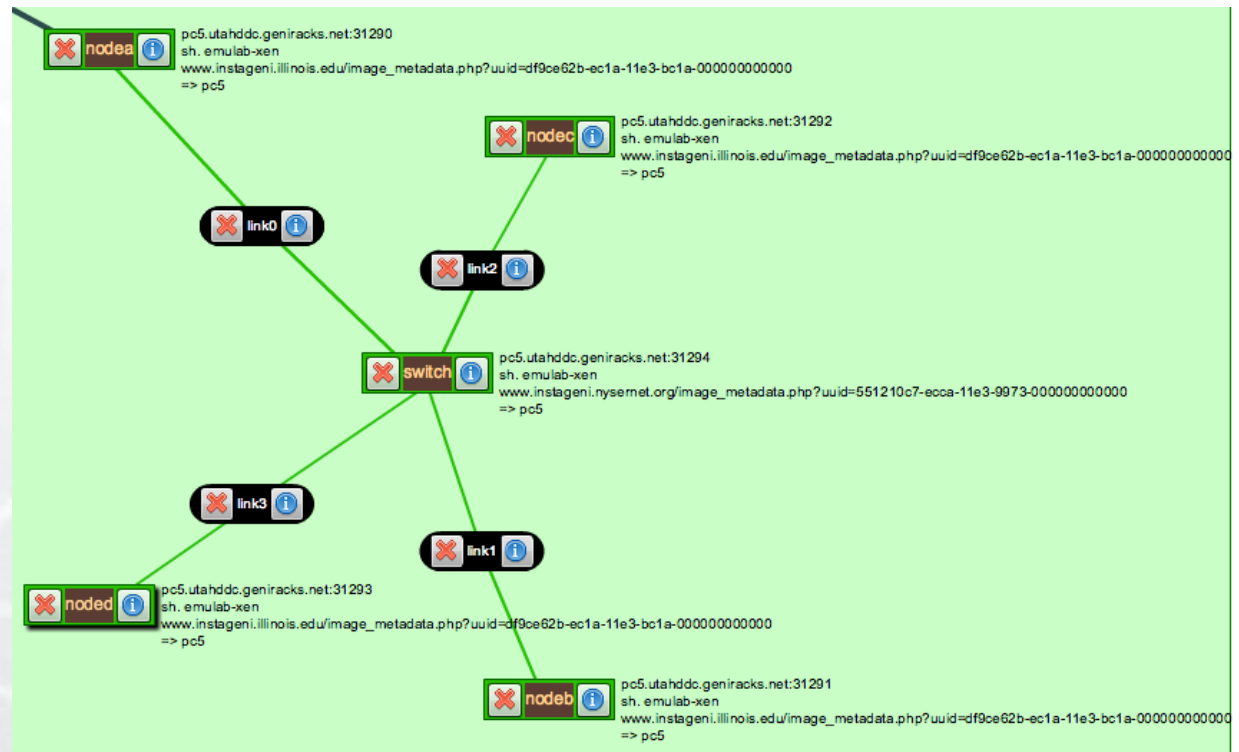




- OML: <http://oml.mytestbed.net/projects/oml>
- Job Service: https://github.com/mytestbed/omf_job_service
- OMF: <http://oml.mytestbed.net/projects/omf>
- AMQP: <http://www.rabbitmq.com/>
- iRODS: <http://irods.org/>
- LabWiki: <https://github.com/mytestbed/labwiki>

1. Reserve slice with VMs that boot GIMI image
2. Verify that slice is up
3. Define experiment to be executed in LabWiki
4. Run and observe experiment
5. Archive results
6. Document results
7. Delete slice (be a good GENI citizen by sharing!!)

- Image
- Postboot script
- ssh into node



- ExoGENI

- Debian 6
- OMF
- OML client & apps
- URL to image:
[http://
emmy9.casa.umass.edu/
Disk_Images/ExoGENI/
deb6ovsgimiv4.xml](http://emmy9.casa.umass.edu/Disk_Images/ExoGENI/deb6ovsgimiv4.xml)

- InstaGENI

- Ubuntu 12.04
- OMF
- OML client & apps
- URL to image:
[https://
www.instageni.illinois.ed
u/image_metadata.php?
uuid=df9ce62b-
ec1a-11e3-
bc1a-000000000000](https://www.instageni.illinois.edu/image_metadata.php?uuid=df9ce62b-ec1a-11e3-bc1a-000000000000)

Nodes 1-4: Postboot Script IG

- Postboot script (<http://emmy9.casa.umass.edu/GEC-20/gimidev.tar.gz>)

```
#!/bin/bash
read -r slice1 </var/emulab/boot/nickname
slicename=$(echo $slice1 | cut -f2 -d.)
host1=$(hostname)
host=$(echo $host1 | cut -f1 -d.)
slice=`ruby -e "print '$slicename'[[^+]*$]"`
echo $host > /etc/hostname
/bin/hostname -F /etc/hostname
```

```
echo "---
:uid: $host-$slice
:uri: amqp://gimi3.casa.umass.edu
:environment: production
:debug: false" > /etc/omf_rc/config.yml
restart omf_rc
```

Initiates connect to
AMQP server

Identical for both
PB scripts

```
wget http://emmy9.casa.umass.edu/GEC-20/fdb.rb -P /root/
wget http://emmy9.casa.umass.edu/GEC-20/Insta_post/learning-switch_copy.rb -P /root
chmod +x /root/learning-switch_copy.rb
wget http://emmy9.casa.umass.edu/GEC-20/Insta_post/learn_ofcollect.rb -P /usr/local/bin
chmod +x /usr/local/bin/learn_ofcollect.rb
wget http://emmy9.casa.umass.edu/GEC-20/Insta_post/ovs-setup2.sh -P /root
chmod +x /root/ovs-setup2.sh
```

Nodes 1-4: Postboot Script EG

- Postboot script (<http://emmy9.casa.umass.edu/GEC-20/gimidev.tar.gz>)

```
#!/bin/bash
export PATH=/usr/local/bin/rubydir/bin:$PATH
host=$1
slice=`ruby -e "print '$2'[/[^\+]*$/]"`
echo $host > /etc/hostname
/bin/hostname -F /etc/hostname
apt-get install psmisc
echo "---
```

```
:uid: $host-$slice
:uri: amqp://gimi3.casa.umass.edu
:environment: production
:debug: false" > /etc/omf_rc/config.yml
echo $PATH > /tmp/curpath.out
if grep 'Ubuntu' /etc/issue; then
restart omf_rc
else
/etc/init.d/omf_rc restart
fi
```

Initiates connect to
AMQP server

Identical for both
PB scripts

- Image
- Postboot script
- ssh into node

- ExoGENI

- Debian 6
- OMF
- OML
- OVS
- URL to image:
[http://
emmy9.casa.umass.edu/
Disk_Images/ExoGENI/
deb6ovsgimiv4.xml](http://emmy9.casa.umass.edu/Disk_Images/ExoGENI/deb6ovsgimiv4.xml)

- InstaGENI

- Ubuntu 12.04
- OMF
- OML
- OVS
- URL to image:
[https://
www.instageni.nysernet.
org/
image_metadata.php?
uuid=551210c7-
ecca-11e3-9973-000000
000000](https://www.instageni.nysernet.org/image_metadata.php?uuid=551210c7-ecca-11e3-9973-000000000000)

- Postboot script:
- Rspec:

```
<execute command="wget http://emmy9.casa.umass.edu/GEC-20/fdb.rb -P /
root/; wget http://emmy9.casa.umass.edu/GEC-20/learning-switch.rb -P /
root/; sh /gimidev/gimibot.sh $self.Name() $sliceName; wget http://
emmy9.casa.umass.edu/GEC-20/ovs-setup.sh -P /root/; sh /root/ovs-setup.sh"
shell="sh"/>
```

- ovs-setup.sh:

```
#!/bin/bash
ethnum=$(ifconfig | grep 'Link encap:' | awk '{print $1}')
while [[ "$sethnum" != *eth1* || "$sethnum" != *eth2* || "$sethnum" != *eth3* || "$sethnum" != *eth4*]]; do
    /usr/local/bin/neuca-netconf
    sleep 10
    ethnum=$(ifconfig | grep 'Link encap:' | awk '{print $1}')
done
ovs-vsctl add-br test
ovs-vsctl add-port test eth1
ovs-vsctl add-port test eth2
ovs-vsctl add-port test eth3
ovs-vsctl add-port test eth4
#ovs-vsctl set-controller test tcp:127.0.0.1:6653 ptcp:6633:127.0.0.1
ovs-vsctl set-controller test tcp:127.0.0.1:6653
```

- ssh into node

```
ssh -i ~/.ssh/yourkey management_ip_of_node
```

- On the node:

```
/var/log/omf_rc.log
```

```
[2014-06-20 17:48:03 +0000] INFO OmfRc::Runner: Starting OMF Resource Controller version '6.1.2.pre.5'  
[2014-06-20 17:48:04 +0000] INFO OmfRc::Runner: Connected using  
{:proto=>:amqp, :user=>"guest", :domain=>"127.0.0.1"}  
[2014-06-20 17:49:05 +0000] WARN OmfCommon::Comm::AMQP::Communicator: Lost connectivity.  
Trying to reconnect...  
[2014-06-20 17:49:05 +0000] INFO OmfRc::Runner: Starting OMF Resource Controller version '6.1.2.pre.5'  
[2014-06-20 17:49:05 +0000] INFO OmfRc::Runner: Connected using  
{:proto=>:amqp, :user=>"guest", :domain=>"127.0.0.1"}  
[2014-06-20 17:56:06 +0000] INFO OmfRc::ResourceProxy::AbstractResource: App Event from  
'ping_cxt_0' (#0) - STARTED: 'env -i /usr/local/bin/ping-oml2 -a 192.168.1.7 -c 30 --oml-  
config /tmp/854849b2-19a2-475d-9588-278271d894aa-1403286966.xml'  
[2014-06-20 17:56:06 +0000] INFO OmfRc::ResourceProxy::AbstractResource: App Event from  
'ping_cxt_1' (#0) - STARTED: 'env -i /usr/local/bin/ping-oml2 -a 192.168.1.8 -c 30 --oml-  
config /tmp/96c56027-f62e-41b1-873d-ba39fe429ab9-1403286966.xml'  
[2014-06-20 17:56:06 +0000] INFO OmfRc::ResourceProxy::AbstractResource: App Event from  
'ping_cxt_2' (#0) - STARTED: 'env -i /usr/local/bin/ping-oml2 -a 192.168.1.9 -c 30 --oml-  
config /tmp/53a86677-61d6-4b31-8f65-4e272d7b2343-1403286966.xml'
```


LabWiki by NICTA
Tools - Add experiment context Michael zink Log out

Plan
Prepare
Execute

Learning switch Assignment

git:mzink/wiki/GEC20-learning-switch.md

Learning switch Assignment

Learning switch Assignment

In this assignment you will learn about to implement the learning switch capability that is used by Ethernet switches by using a software-based OpenFlow switch. The functionality of the learning switch will both be explored with the LabWiki script.

Overall we will do the following:

- Set-up the learning switch
- Verify the correctness of the learning-switch

Preparation for Routing Assignment

To learn about learning-switch, check out this link: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>.

To learn about the trema controller, check out this link: <http://trema.github.io/trema/>.

Also you should understand the topology you will be utilizing for this tutorial. It can be found [here](#).

Preparation for utilizing Geni Environment

Make sure that before starting you have setup the correct topology and have the proper setup of SSH keys. The following videos will walk you through these steps.

- Part 1: Understanding the Geni Portal
- Part 2: Setting up your SSH Keys
- Part 3: Reserving Resources

In the case of this assignment the only thing that is different is the name of the RSpec you have to choose. So instead of using "ECE3743Node" (as shown around 1:50 minutes into the video) you now have to use RSpec titled "learning-switch"

Experiment

The goal of this assignment is to implement the learning switch capability that is used by Ethernet switches by using a software-based OpenFlow switch. In the case of the topology shown in the figure, this software switch is to be implemented in node "switch". All the other nodes represent regular hosts. To realize this implementation of a learning switch it is your task to implement and trema-based OpenFlow controller. You will have to verify the correct functionality of the learning switch by creating an experiment script in which any node A pings nodes B – node E in LabWiki.

Working

In the topology shown in the figure, the switch node will act as the learning switch, which connect nodes A-E with each other. Whenever a regular node pings any regular node for the first time, the switch does not know the destination address of the node. At this point, the switch node floods the packet to all the nodes it is connected to. When it receives a reply to the flood, this information will be used to populate its switching table. Hereafter, whenever any node wants to ping any other nodes, the switch node will fetch the information from the table. For this purpose, two scripts are already preloaded in the switch node, namely learning-switch.rb and fdb.rb. The second script offers rudimentary database functionality that is used to maintain the switching table.

Set up the learning switch

extended-learningswitch.oedl

git:mzink:oed/extended-learningswitch.oedl

```

1 defProperty('source1', 'nodea-mz-ls-extended', "ID of a resource")
2 defProperty('source2', 'switch1-mz-ls-extended', "ID of a resource")
3 defProperty('source3', 'switch2-mz-ls-extended', "ID of a resource")
4
5
6 defProperty('sinkaddr12', '192.168.1.7', "Ping destination address")
7 defProperty('sinkaddr13', '192.168.1.8', "Ping destination address")
8
9 defProperty('sinkaddr21', '192.168.1.13', "Ping destination address")
10 defProperty('sinkaddr23', '192.168.1.14', "Ping destination address")
11 defProperty('sinkaddr24', '192.168.1.15', "Ping destination address")
12
13
14 defApplication('ping') do [app]
15   app.description = 'Simple Definition for the ping-oml2 application'
16   # Define the path to the binary executable for this application
17   app.binary_path = '/usr/local/bin/ping-oml2'
18   # Define the configurable parameters for this application
19   # For example if target is set to foo.com and count is set to 2, then the
20   # application will be started with the command line:
21   # /usr/bin/ping-oml2 -a foo.com -c 2
22   app.defProperty('target', 'Address to ping', '-a', {:type => :string})
23   app.defProperty('count', 'Number of times to ping', '-c', {:type => :integer})
24   # Define the OML2 measurement point that this application provides.
25   # Here we have only one measurement point (MP) named 'ping'. Each measurement
26   # sample from this MP will be composed of a 4-tuples (addr,ttl,rtt,rtt_unit)
27   app.defMeasurement('ping') do [m]
28     m.defMetric('dest_addr',:string)
29     m.defMetric('ttl',:uint32)
30     m.defMetric('rtt',:double)
31     m.defMetric('rtt_unit',:string)
32   end
33 end
34
35 defApplication('trema') do [app]
36   app.description = 'This app runs trema from command line'
37   app.binary_path = '/usr/bin/trema run /root/learning-switch.rb'
38 end
39 defGroup('source2', property.source2, property.source3) do [node]
40   node.addApplication("trema")
41 end
42 defGroup('source1', property.source1) do [node]
43   node.addApplication("ping") do [app]
44     app.setProperty('target', property.sinkaddr12)
45     app.setProperty('count', 30)
46     #app.setProperty('interval', 1)
47     app.measure('ping', :samples => 1)
48   end
49   node.addApplication("ping") do [app]
50     app.setProperty('target', property.sinkaddr13)
51     app.setProperty('count', 30)
52     #app.setProperty('interval', 1)
53     app.measure('ping', :samples => 1)
54   end
55   node.addApplication("ping") do [app]
56     app.setProperty('target', property.sinkaddr21)

```

mzink-ls_extended_test1-2014-06-11T19-30-04

Finished

sinkaddr13	192.168.1.8
sinkaddr21	192.168.1.13
sinkaddr23	192.168.1.14
sinkaddr24	192.168.1.15

Graphs

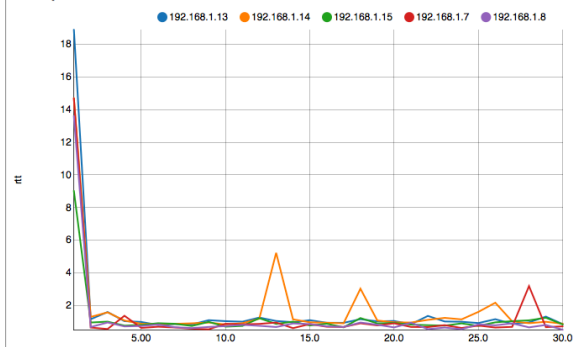


Figure: RTT of received packets.

Logging

98.0	Disconnecting...
97.0	OMF Experiment Controller 6.1.2.pre.5 - Exit.
93.0	(4493a08d-eeee-4291-9af7-2d09a6ed4a4a) Distributing message to [#<Proc:0x00000049cab90@home/gimladmin/omf_job_service/omf_ec/vendor/bundle/ruby/1.9.1/gems/omf_ec-6.1.2.pre.5/lib/omf_ec.rb:49>]
93.0	(4493a08d-eeee-4291-9af7-2d09a6ed4a4a) Message type [':inform',:message,:released'] (Om/Common::Message::Json::Message)
93.0	(4493a08d-eeee-4291-9af7-2d09a6ed4a4a) Deliver message 'inform': JsonMessage: {src=>:ampq://gim3.casa.umass.edu/frcp.4493a08d-eeee-4291-9af7-2d09a6ed4a4a,:type=>:application,:itype=>:RELEASED,:op=>:inform,:mid=>:6f145486-3cfd-4dc2-b43d-ac1a3bc01e7e',:ts=>:1402515105,:props=>{:res_id=>:ampq://gim3.casa.umass.edu

- We will look at this in LabWiki
- More info on OEDL scripts:
<http://omf.mytestbed.net/projects/omf6/wiki/OEDLOMF6>

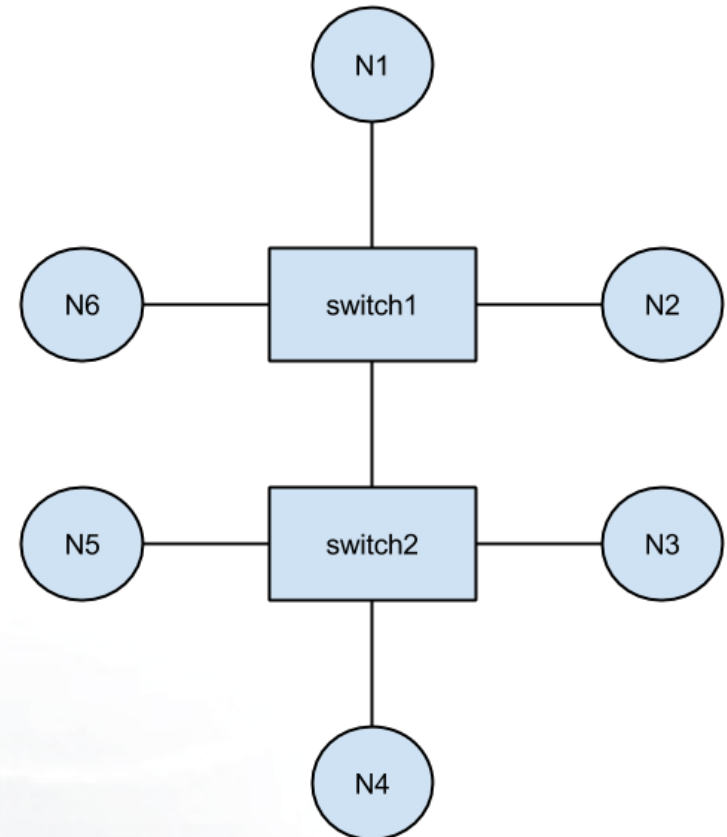
- We will look at this in LabWiki

Module C

**[http://groups.geni.net/geni/wiki/
GEC20Agenda/LabWiki/ModuleC](http://groups.geni.net/geni/wiki/GEC20Agenda/LabWiki/ModuleC)**

Extended Experiment Topology

- N1 – N6: regular end systems
- Switch 1-2: VM with OVS installed → sw-based OF switch
- Trema-based OF controller running on Switch



Goal: Implement learning switch and verify its correct behavior

- Observe performance of flows
- Done via `flow_stats` in OF controller
- OML app to *capture* information and transmit to OMLserver
- Use LabWiki to observe FlowStats live

- **Sample assignments:**
<http://groups.geni.net/geni/wiki/GENIEducation/SampleAssignments>
- **Demo Night, immediately after this session**