Tutorial: OpenFlow and GENI

Niky Riga
GENI Project Office
ICDCS13

Sponsored by the National Science Foundation

- **Part I: Design/Setup**
  - **Obtain Resources**
  - What is OpenFlow, what can I do with Openflow?
  - Demo: Using OpenFlow in GENI
- Part II: Execute
  - Configure and Initialize Services
  - Execute Experiment
- Part III: Finish
  - Teardown Experiment

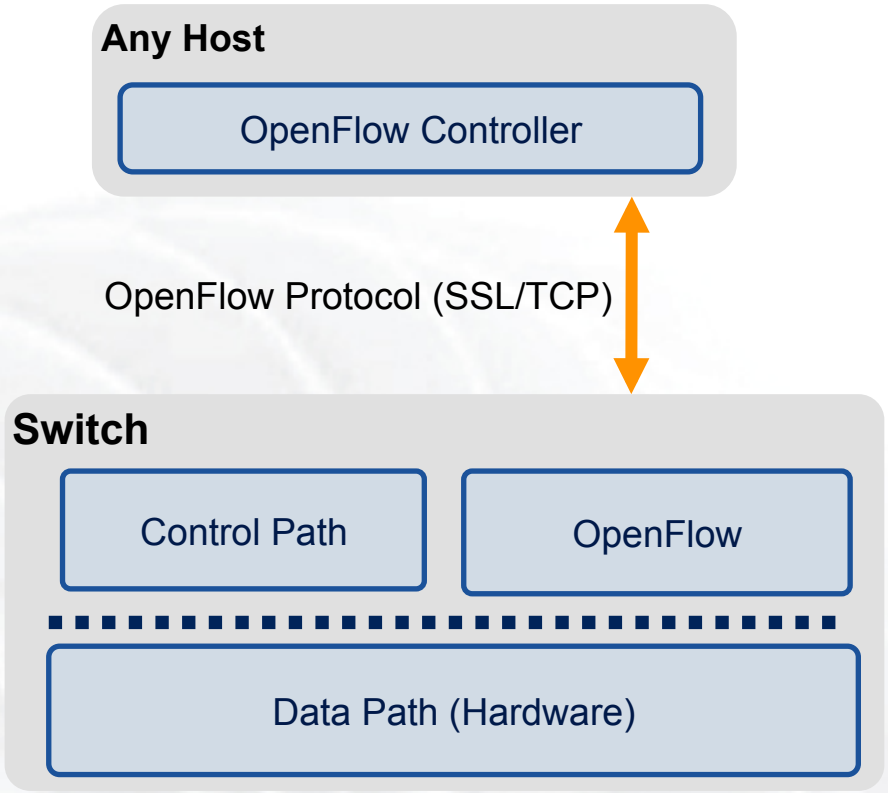
- **Part I: Design/Setup**
  - Obtain Resources
  - **What is OpenFlow, what can I do with Openflow?**
  - Demo: Using OpenFlow in GENI
- Part II: Execute
  - Configure and Initialize Services
  - Execute Experiment
- Part III: Finish
  - Teardown Experiment

OpenFlow is an API

- Control how packets are forwarded
- Implementable on COTS hardware
- Make deployed networks programmable
  - not just configurable
- Makes innovation easier

Modified slide from : http://www.deutsche-telekom-laboratories.de/~robert/GENI-Experimenters-Workshop.ppt

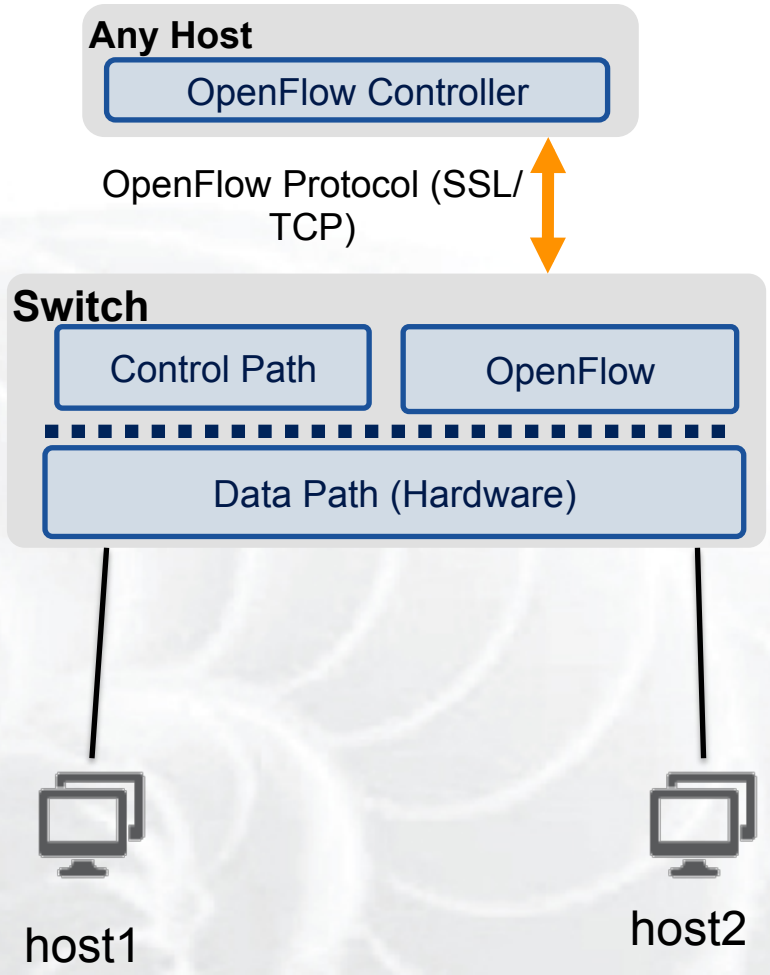
OpenFlow



- The controller is responsible for populating forwarding table of the switch
- In a table miss the switch asks the controller

Modified slide from : http://www.deutsche-telekom-laboratories.de/~robert/GENI-Experimenters-Workshop.ppt

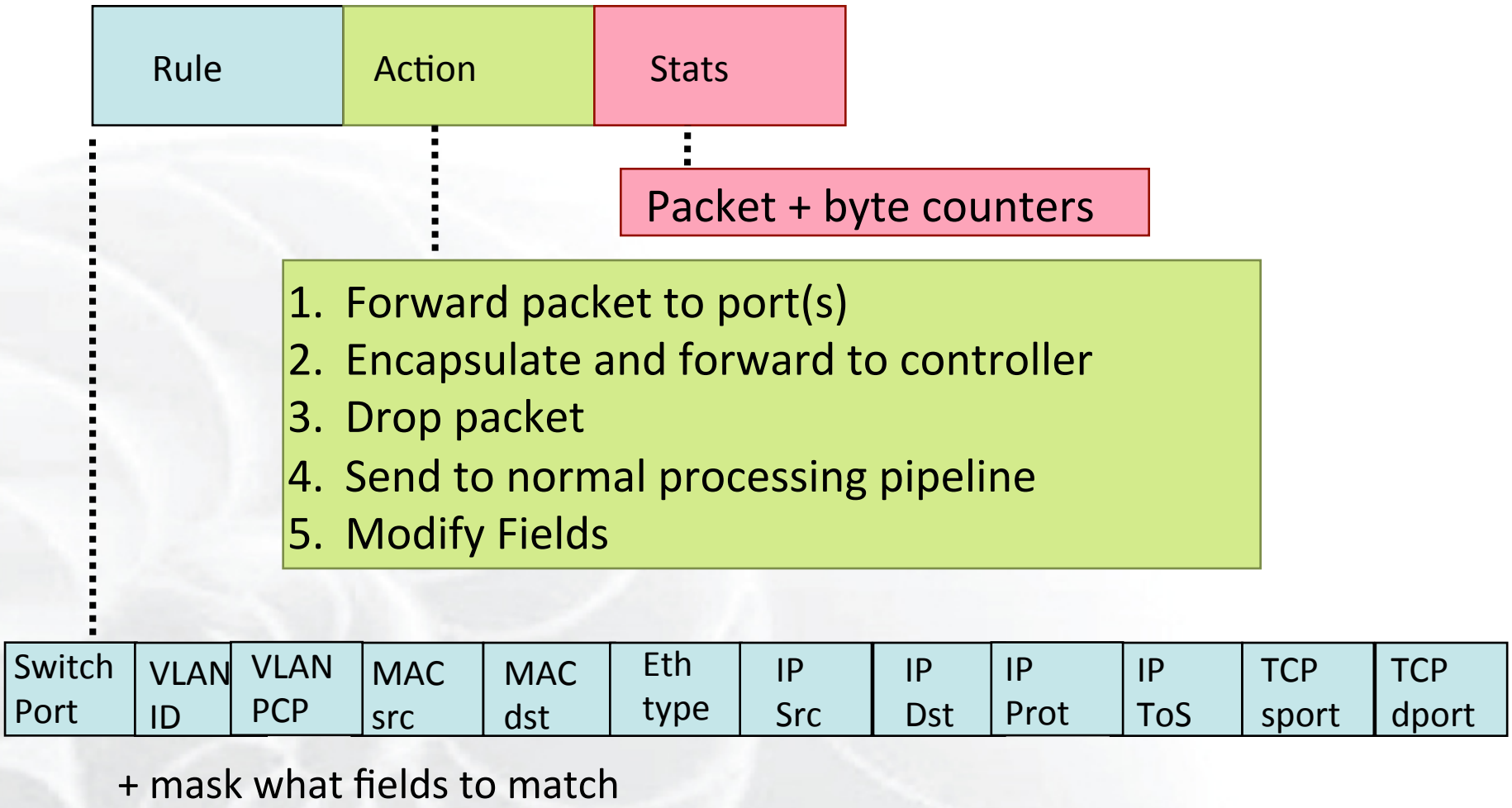
OpenFlow in action



- Host1 sends a packet
- If there are no rules about handling this packet
  - Forward packet to the controller
  - Controller installs a flow
- Subsequent packets do not go through the controller

Modified slide from : http://www.deutsche-telekom-laboratories.de/~robert/GENI-Experimenters-Workshop.ppt

OpenFlow Basics
Flow Table Entries



slide from : http://www.deutsche-telekom-laboratories.de/~robert/GENI-Experimenters-Workshop.ppt

Use Flow Mods

- Going through the controller on every packet is inefficient
- Installing Flows either proactively or reactively is the right thing to do:
- A Flow Mod consists off :
  - A match on any of the 12 supported fields
  - A rule about what to do matched packets
  - Timeouts about the rules:
    - Hard timeouts
    - Idle timeouts
  - The packet id in reactive controllers

OpenFlow common PitFalls

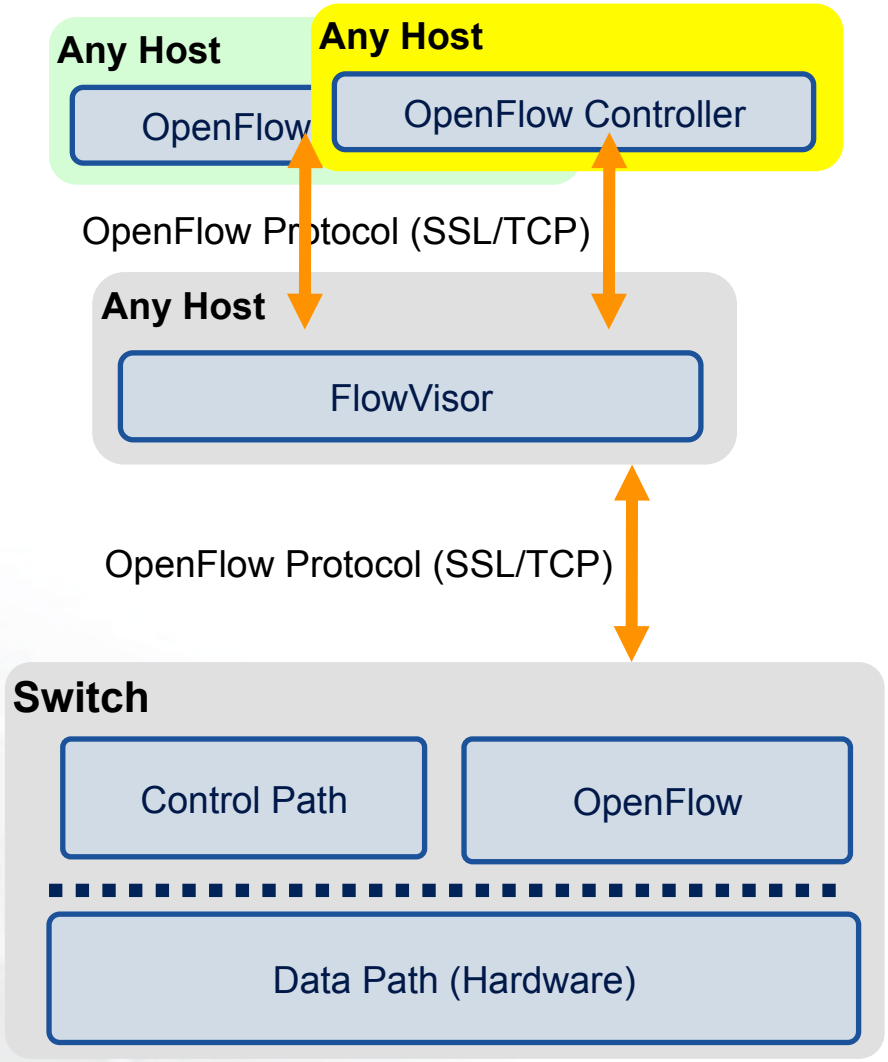
- Controller is responsible for all traffic, not just your application!
  - ARPs
  - DHCP
  - LLDP
- Reactive controllers
  - UDP
- Performance in hardware switches
  - Not all actions are supported in hardware
- No STP
  - Broadcast storms

FlowVisor

- Only one controller per switch
- FlowVisor is a proxy controller that can support multiple controllers

FlowSpace describes packet flows :

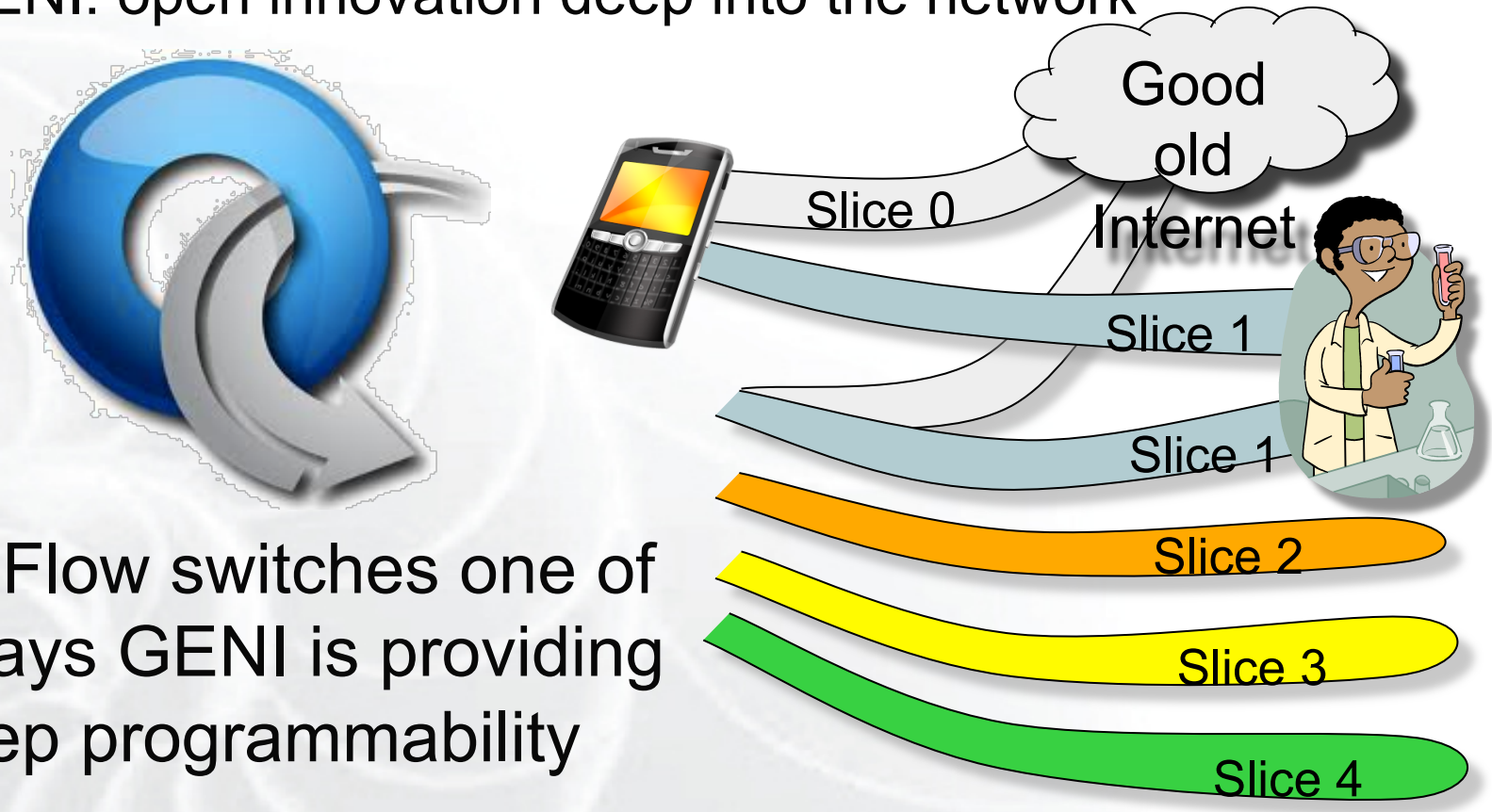
- Layer 1: Incoming port on switch
- Layer 2: Ethernet src/dst addr, type, vlanid, vlanpcp
- Layer 3: IP src/dst addr, protocol, ToS
- Layer 4: TCP/UDP src/dst port



- **Part I: Design/Setup**
  - Obtain Resources
  - What is OpenFlow, what can I do with Openflow?
  - **Demo: Using OpenFlow in GENI**
- Part II: Execute
  - Configure and Initialize Services
  - Execute Experiment
- Part III: Finish
  - Teardown Experiment

GENI Programmable Network

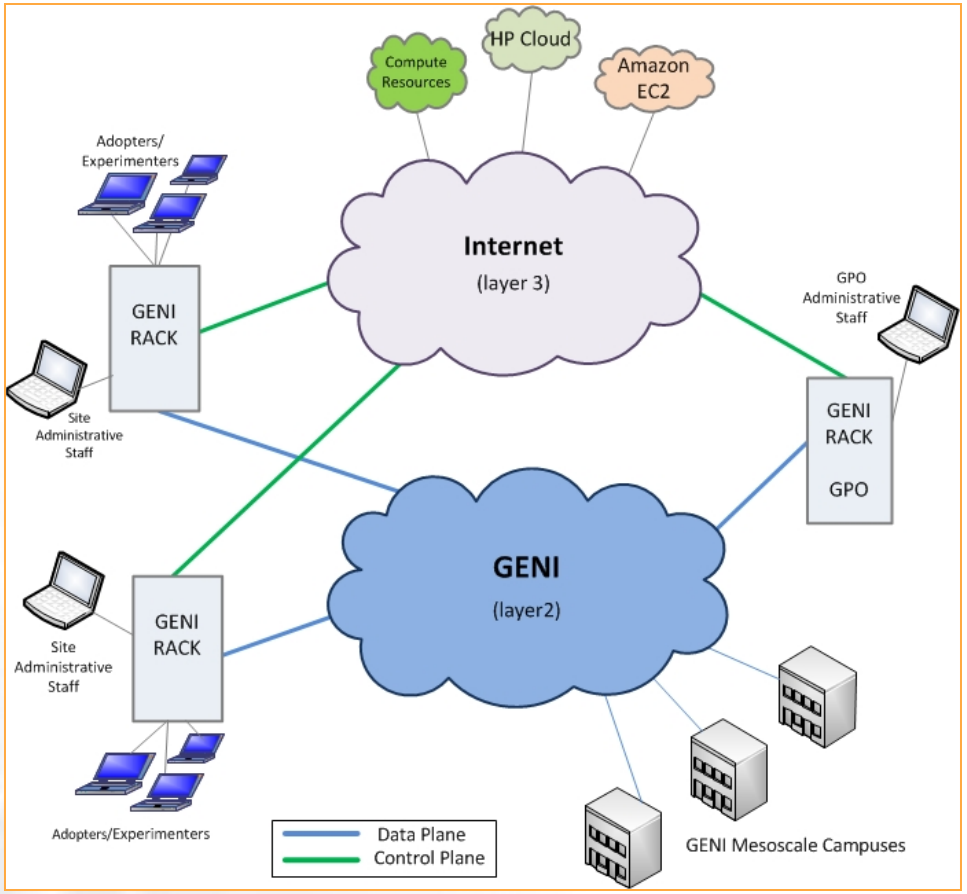
- Key GENI concept: slices & deep programmability
  - Internet: open innovation in application programs
  - GENI: open innovation deep into the network



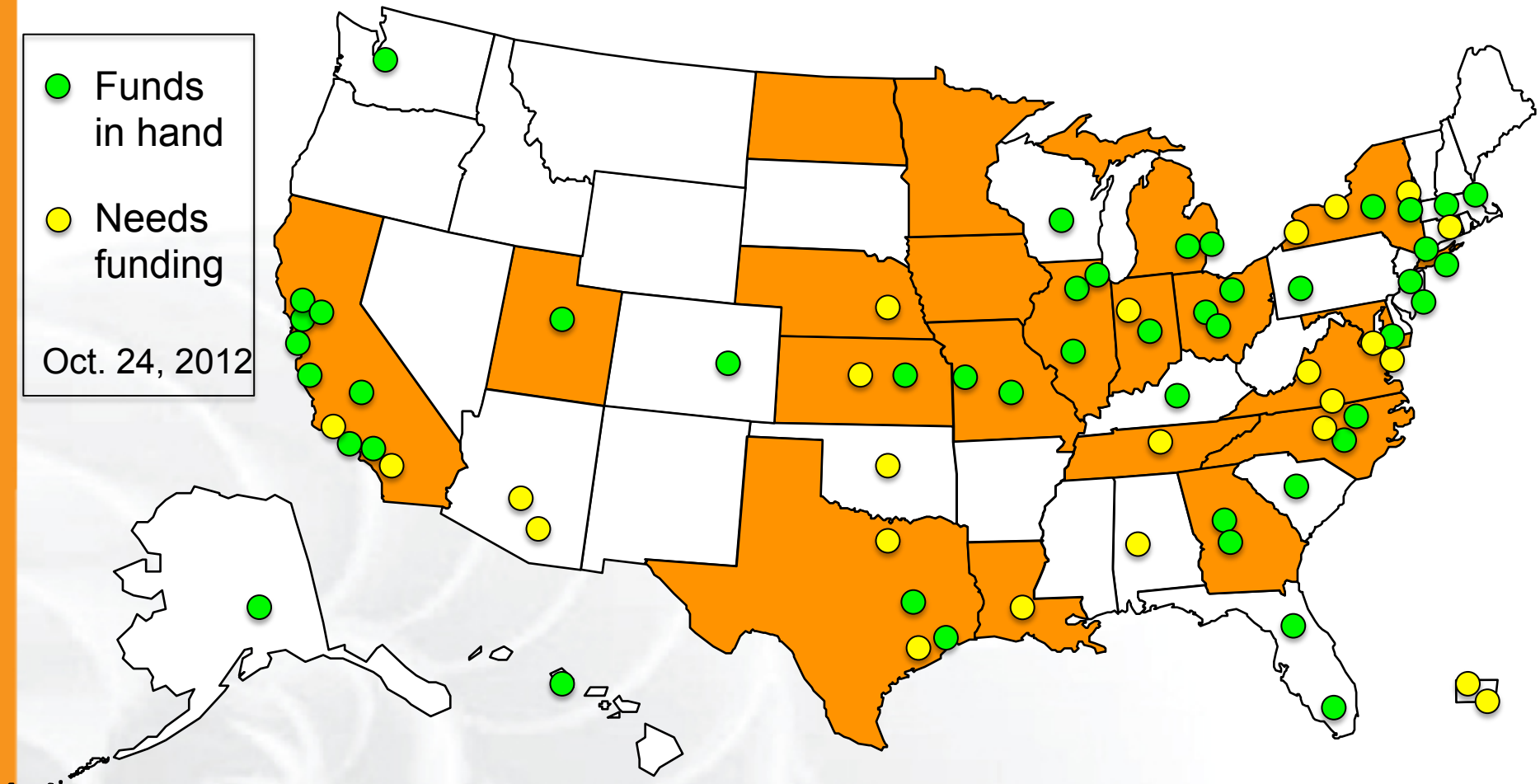
OpenFlow switches one of the ways GENI is providing deep programmability

Racks and Campuses

- GENI Rack projects are expanding available GENI infrastructure in the US.
- Racks provide reservable, sliceable compute and network resources using Aggregate Managers.
- GENI AM API compliance

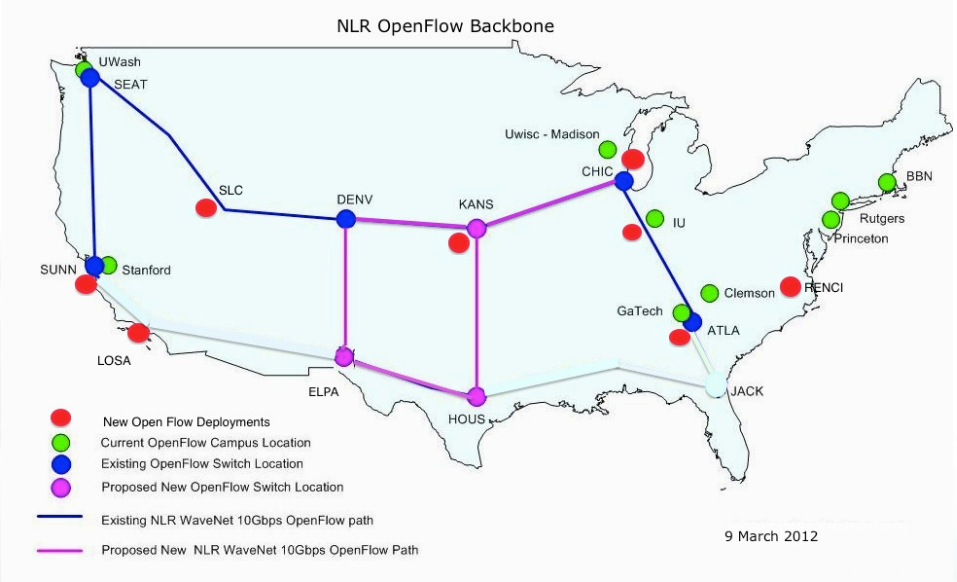


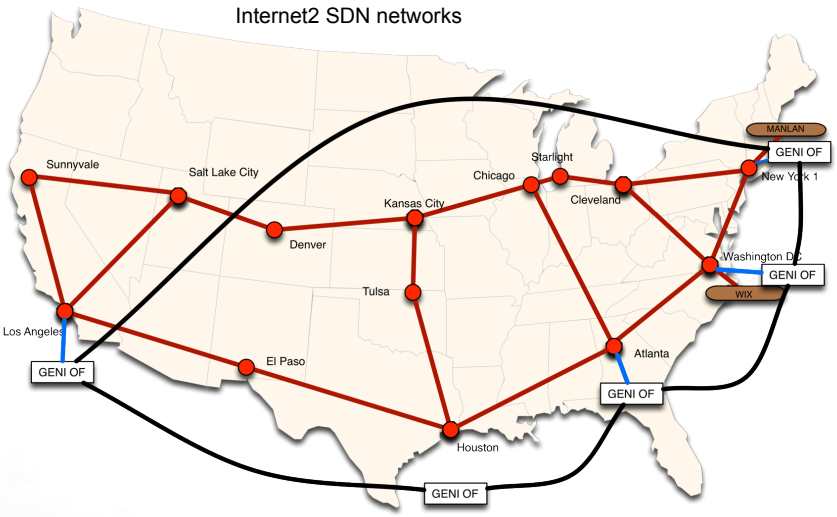
GENI Rack Campuses



- 43 racks planned this year
- Each rack has an **OpenFlow-enabled** switch

Core Networks





- NLR committed to 2013 meso-scale expansion following reorganization
- Internet2 adding 10GbE paths to Advanced Layer 2 Services (AL2S) at 4 of 5 OpenFlow meso-scale/ProtoGENI Pops
- GENI Aggregate Manager in Internet2 AL2S and dynamic stitching with GENI coming in Spiral 5

FOAM

- An OpenFlow Aggregate Manager
- It's a GENI compliant reservation service
  - Helps experimenters reserve flowspace in the FlowVisor
- Speaks AM API v1
- Rspecs GENI v3, openflow v3 extension

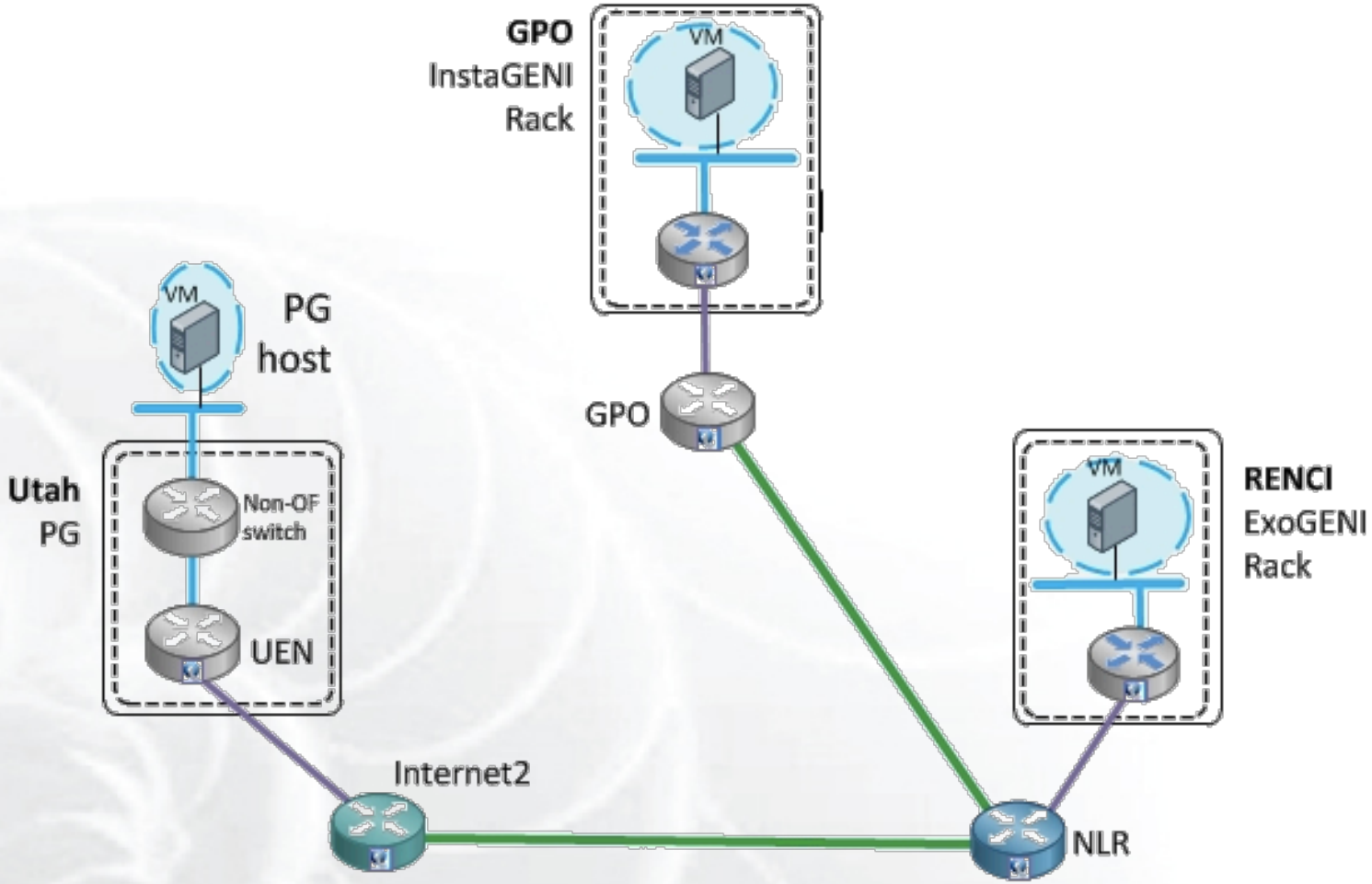
OpenFlow Experiment

Experiment will demonstrate OpenFlow in GENI using:

- InstaGENI, ExoGENI and ProtoGENI sites OpenFlow resources.
- GENI OpenFlow backbone and Regional resources.
- InstaGENI, ExoGENI and PG sites compute resources.
- This experiment is available at:
  http://groups.geni.net/geni/wiki/GENIExperimenter/ExperimentExample-OF

OpenFlow Experiment

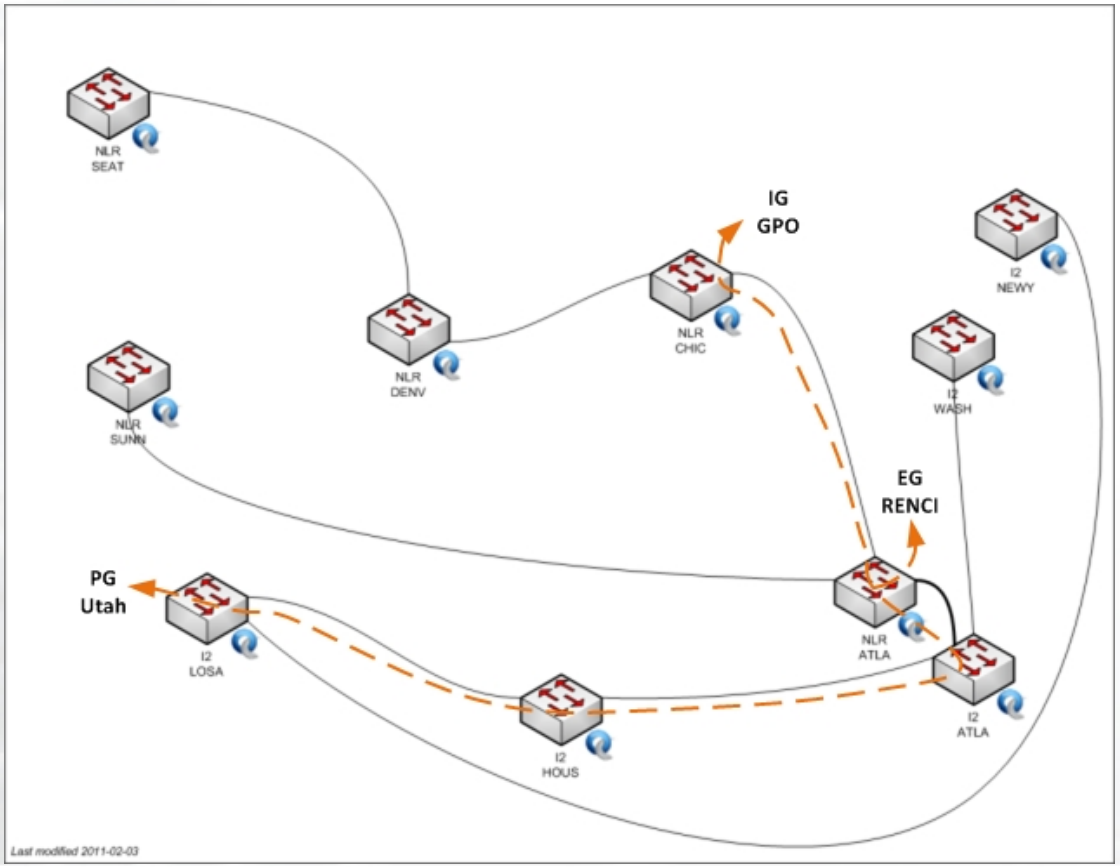
Experiment topology



- **Part I: Design/Setup**
  - **Obtain Resources**
- Part II: Execute
  - Configure and Initialize Services
  - Execute Experiment
- Part III: Finish
  - Teardown Experiment

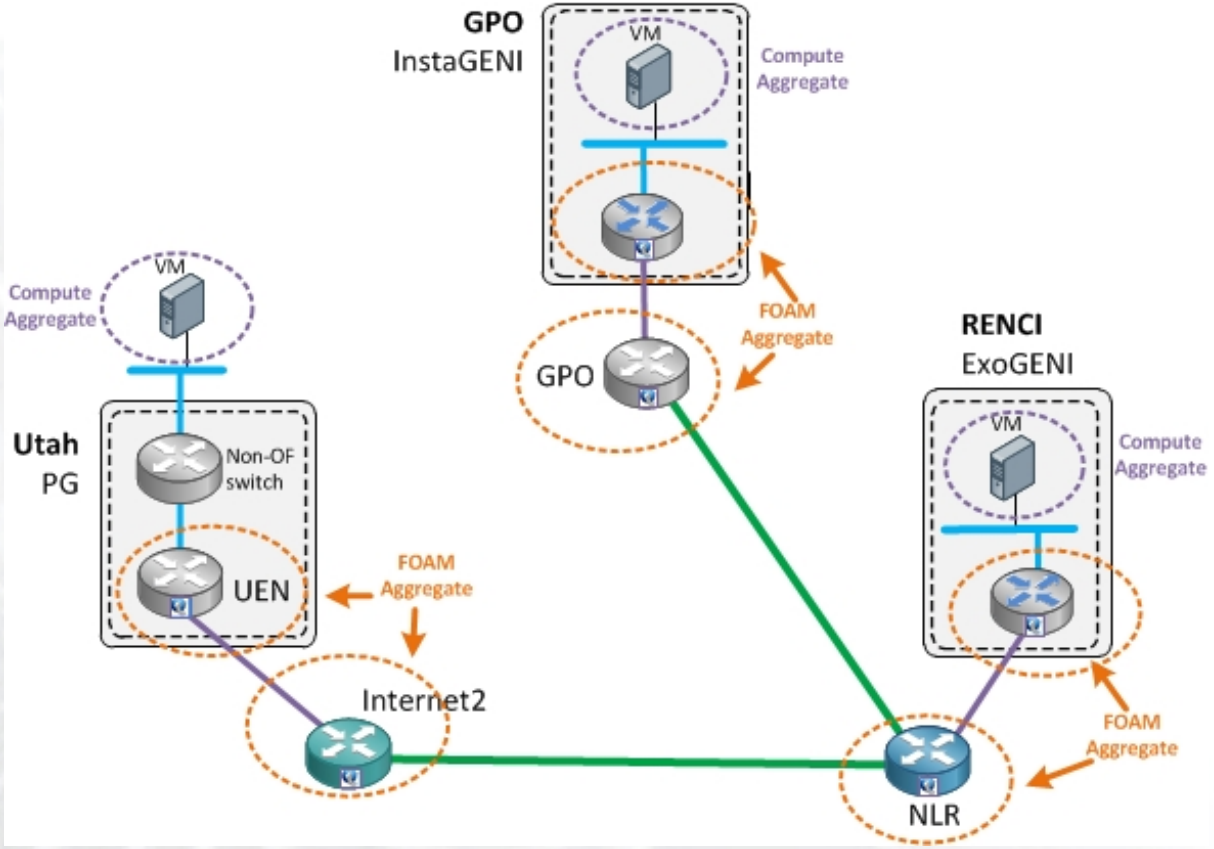
Obtain Resources

- **Determine OpenFlow resources for the experiment sites:**
  - http://groups.geni.net/geni/wiki/GeniAggregate
- **Determine Core Network to use:**
  - http://groups.geni.net/geni/wiki/NetworkCore



Obtain Resources

- Write OpenFlow request RSpecs (GPO InstaGENI, RENCI ExoGENI, PG Utah UEN, NLR and Internet2).
- Write compute resources request RSpecs (GPO InstaGENI, RENCI ExoGENI, Utah PG)



Note: Request IP address range http://groups.geni.net/geni/wiki/NetworkCore/SubnetReservations

Request Resources:

1. **Create a slice:**

   $ omni.py createslice 3sites-OF

2. **Request resources at each FOAM aggregate:**

   $ for aggregate in gpo ig-gpo eg-renci uen nlr i2
   > do
   > omni.py –a of-$aggregate createsliver 3sites-OF $aggregate-of.rspec
   > done

   Note: Approval email is sent from each FOAM site, some auto-approve.

3. **Request compute resources:**

   $ for aggregate in ig-gpo eg-renci pg-utah
   > do
   > omni.py –a $aggregate createsliver 3sites-OF $aggregate-cr.rspec
   > done

- Part I: Design/Setup
  - Obtain Resources
- **Part II: Execute**
  - **Configure and Initialize Services**
  - Execute Experiment
- Part III: Finish
  - Teardown Experiment

Configure and Initialize Services

Determine login information to connect to hosts:

$ *readyToLogin.py -a ig-gpo 3sites-OF*

….

User lnevers logins to gpo-ig using:

xterm -e ssh -p 30522 -i /home/lnevers/.ssh/id_rsa lnevers@pc1.instageni.gpolab.bbn.com &

$ *readyToLogin.py –a eg-renci 3sites-OF*

….

User root logins to renci-eg using:

xterm -e ssh -i /home/lnevers/.ssh/id_rsa root@152.54.14.17 &

$ *readyToLogin.py –a pg-utah 3sites-OF*

….

User lnevers logins to utah-pg using:

xterm -e ssh -p 30010 -i /home/lnevers/.ssh/id_rsa lnevers@pc522.emulab.net &

- Part I: Design/Setup
  - Obtain Resources
- **Part II: Execute**
  - Configure and Initialize Services
  - **Execute Experiment**
- Part III: Finish
  - Teardown Experiment

Execute Experiment

Experiment is a simple ping to show connections are possible between the sites:

- Login into each host and start a ping to a remote site – This should fail, as no controller is running!
- Start your OpenFlow controller, in this example the NOX controller is used.
- Review the windows were pings had been failing and now you will see ping traffic is flowing!

- Part I: Design/Setup
  - Obtain Resources
- Part II: Execute
  - Configure and Initialize Services
  - Execute Experiment
- **Part III: Finish**
  - **Teardown Experiment**

Teardown Experiment

When the experiment is done, archive your data and release the resources by deleting the slivers at each aggregate:

$ for aggregate in of-gpo of-nlr of-i2 of-uen eg-of-renci ig-of-gpo ig-gpo eg-renci pg-utah
> do
> omni.py -a $aggreate deletesliver 3sites-OF
> done

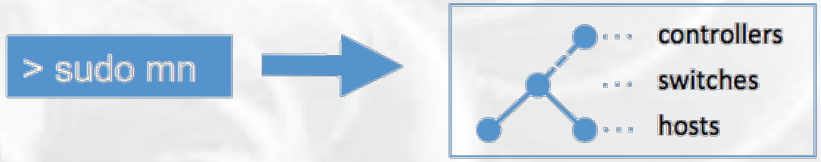
The resources have been released, you are now done!

OpenFlow Experiments

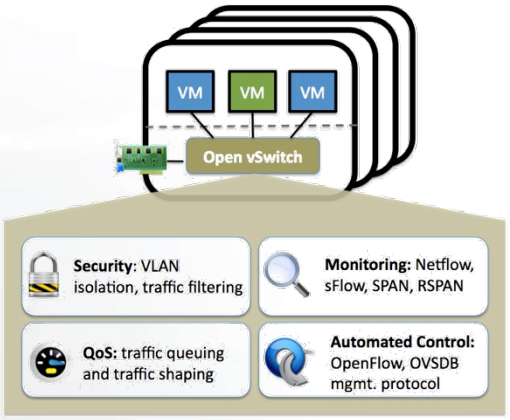
Debugging OpenFlow experiments is hard:

- Network configuration debugging requires coordination
- Many networking elements in play
- No console access to the switch

Before deploying your OpenFlow experiment test your controller.



http://mininet.github.com/

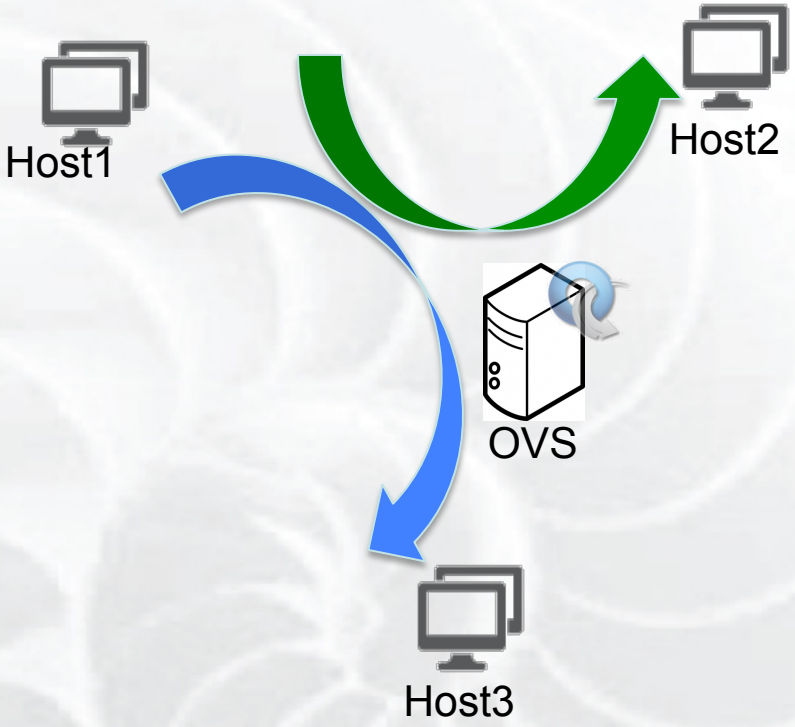


http://openvswitch.org/

Run an OpenFlow experiment

1 host as OVS switch

3 VMs connected to OVS



- Setup OVS
- Write simple controllers
  - e.g. diverge traffic to a different server
  - use python controler PoX

Before we start

- Many people will be accessing the resources, so some calls might fail. Wait a bit and try again!
- There will be a lot of commands to run, copy paste is your friend
- You can copy-paste between your computer and the VM.

- Part I: Design/Setup
  - Obtain Resources
  - What is OpenFlow, what can I do with Openflow?
  - Demo: Using OpenFlow in GENI
- **Part II: Execute**
  - **Configure and Initialize Services**
  - Execute Experiment
- Part III: Finish
  - Teardown Experiment

Configure OVS

OVS is a virtual switch running on a bare metal node.

- The interfaces of the node are the ports of the switch
  - Configure an ethernet bridge
  - add all dataplane ports to the switch
- Can be an OpenFlow switch
  - Need to specify the controller (for convinience on the same host but it can be anywhere)
- Userspace OVS for this exercise

- Part I: Design/Setup
  - Obtain Resources
  - What is OpenFlow, what can I do with Openflow?
  - Demo: Using OpenFlow in GENI
- **Part II: Execute**
  - Configure and Initialize Services
  - **Execute Experiment**
- Part III: Finish
  - Teardown Experiment

Experiments (1/4)

1. Verify connectivity with using a learning switch
    1. See the flow between host start and stop based on the controller
    2. Soft versus hard timeouts

Experiments (2/4)

2. Write a controller that will duplicate traffic to a different port on the switch
    1. Use tcpdump to see the duplication

Experiments (3/4)

3. Write a controller that will do port forwarding on your server
    1. Use netcat to run two servers on host2

Experiments (4/4)

3. Write a controller that will redirect packets to a proxy
   1. What fields do you need to overwrite?
   2. Which packets needs special handling?
   3. Use netcat to see the deflection

- Part I: Design/Setup
  - Obtain Resources
  - What is OpenFlow, what can I do with Openflow?
  - Demo: Using OpenFlow in GENI
- Part II: Execute
  - Configure and Initialize Services
  - Execute Experiment
- **Part III: Finish**
  - **Teardown Experiment**

Part III: Finish Experiment

When your experiment is done, you should always release your resources.

- Normally this is when you would archive your data
- Delete your slivers at **each** aggregate