



[http://groups.geni.net/geni/wiki/GEC13Agenda/
ExperimenterWorkflowTutorial/Tutorial](http://groups.geni.net/geni/wiki/GEC13Agenda/ExperimenterWorkflowTutorial/Tutorial)

Tutorial: Experimenter Workflow and Topologies using GENI

Niky Riga, Sarah Edwards
GENI Project Office
13 March 2012



- Overview (tutorial, resources, Flack, Omni)

... hands on ...

- Choosing your physical topology

... hands on ...

- Building a layer 2 Experiment

... hands on ...

- Using Openflow

... hands on ...

- Wrap Up

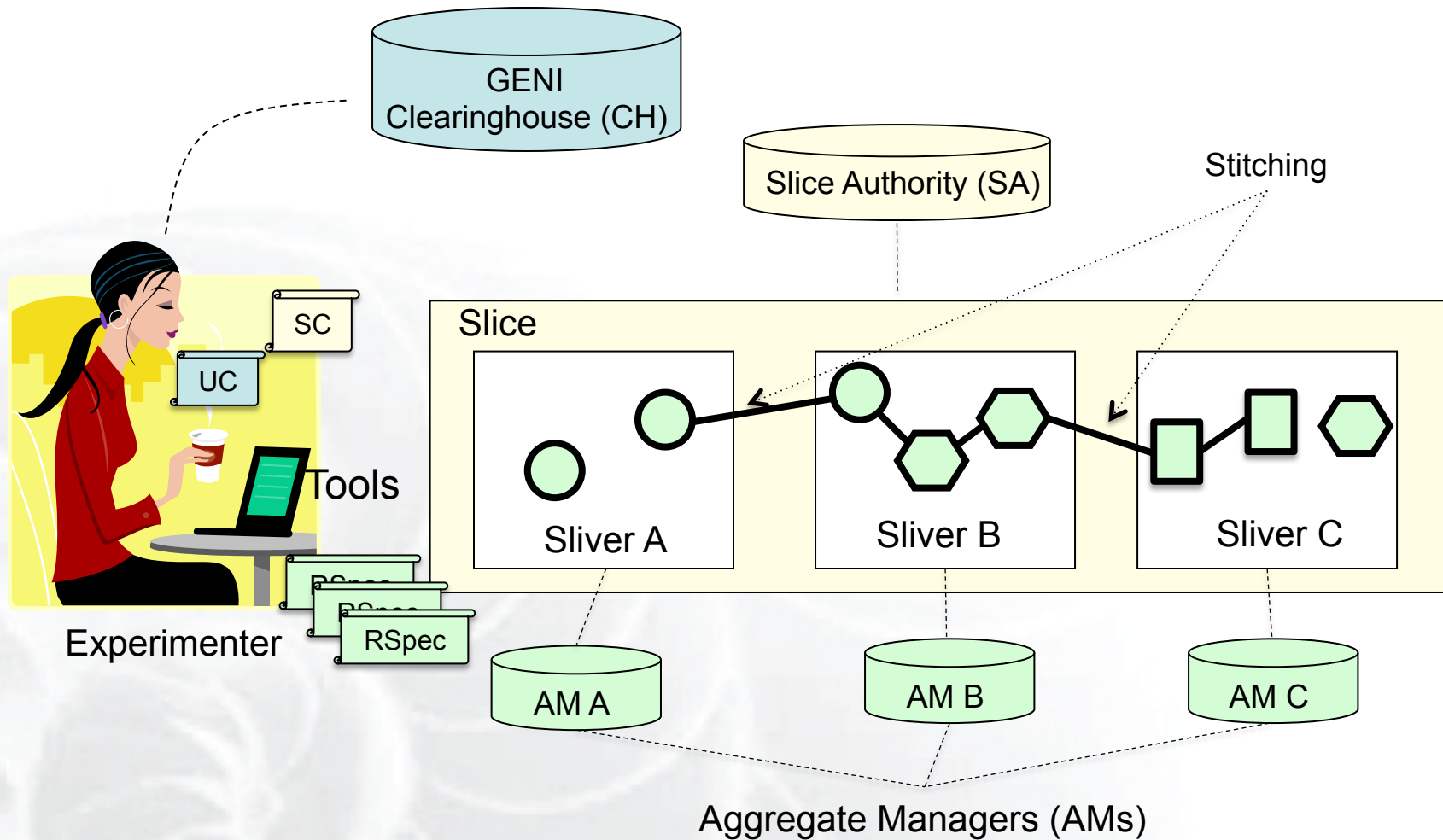
... hands on ...

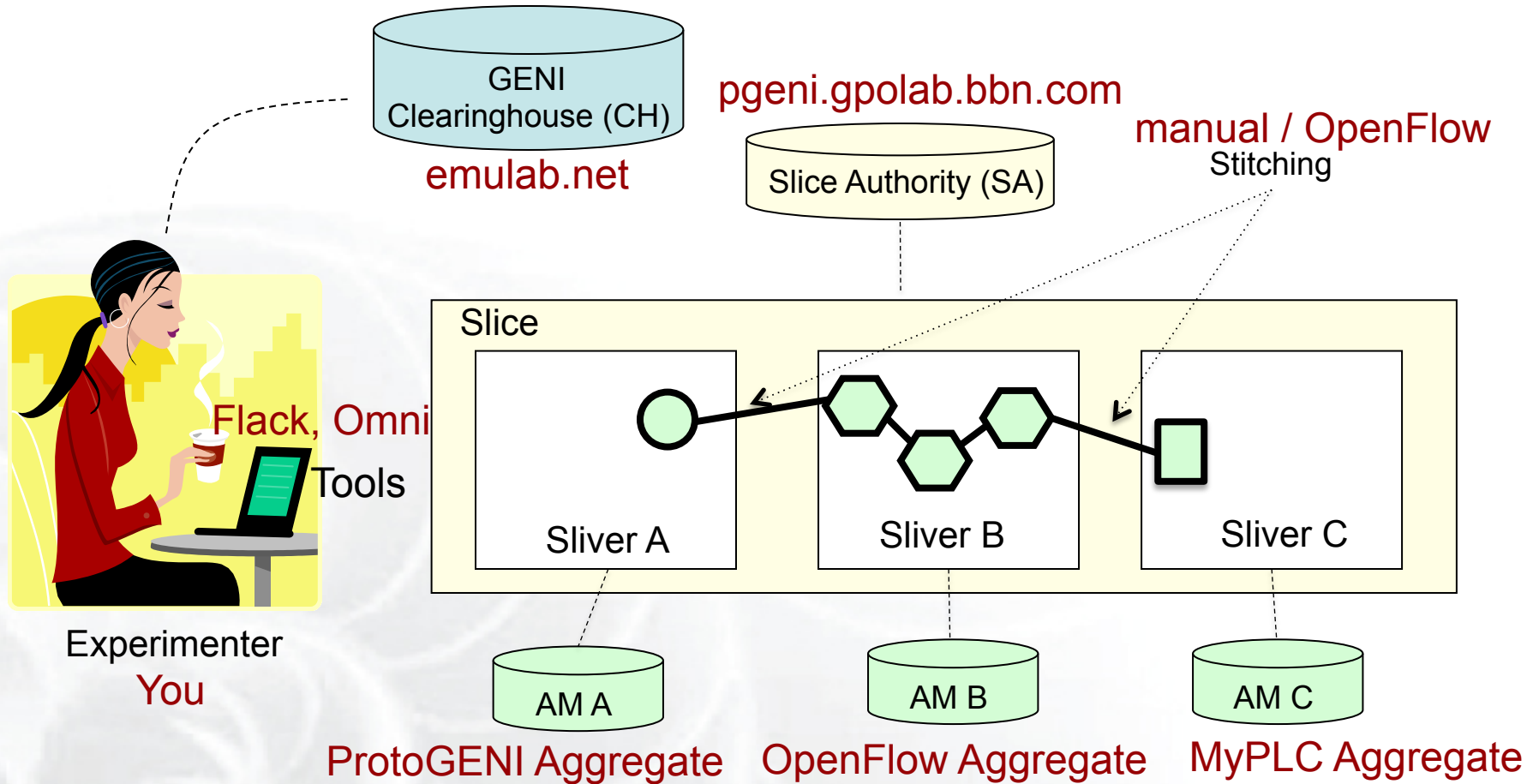
- Learn how an experiment in GENI works
 - How is an experiment setup ?
 - What are the key participants?
 - How do different tools work?
- Use Mesoscale for your experiment
 - Control your topology
 - Run Layer 2 (and up) experiments across the country

Looking Behind the Scenes Disclaimer

- GENI is evolving fast
 - What is true today, might change tomorrow
- Use tutorial resources as a pointer
- Best place to get up-to-date info is the GENI wiki
<http://groups.geni.net/geni/wiki/GeniExperiments>
- Email us with questions (help@geni.net)

Experiment Workflow and Terminology

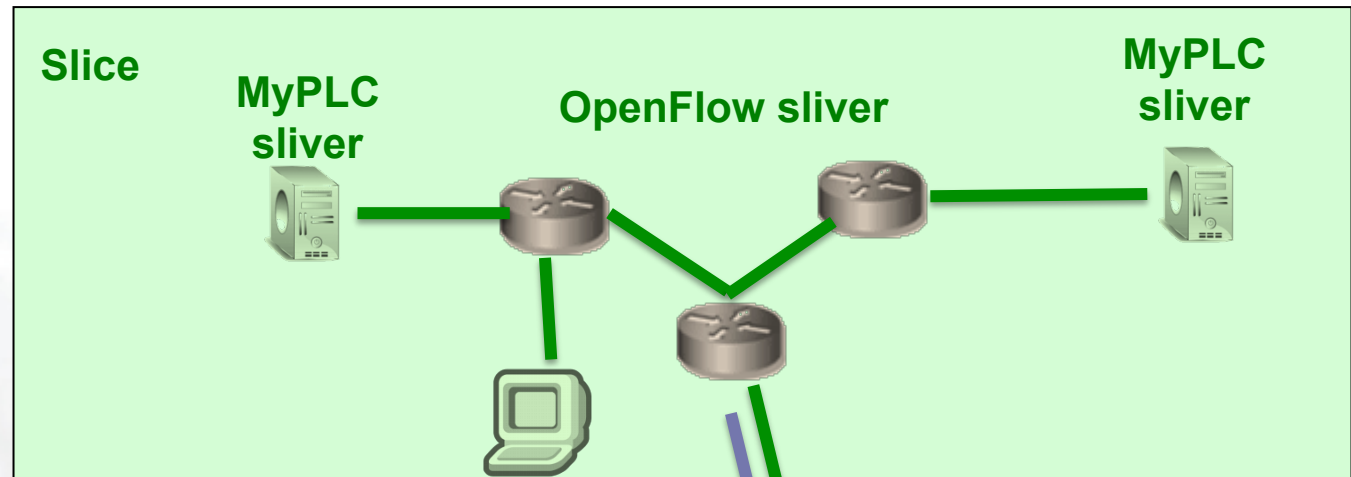




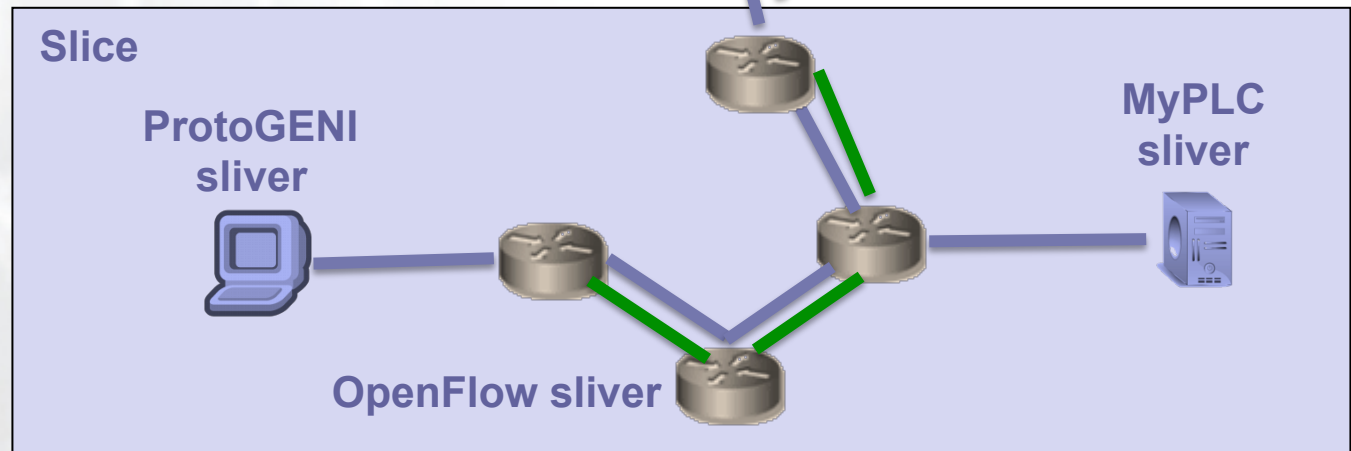
Today's GENI Experiments



Flack, Omni



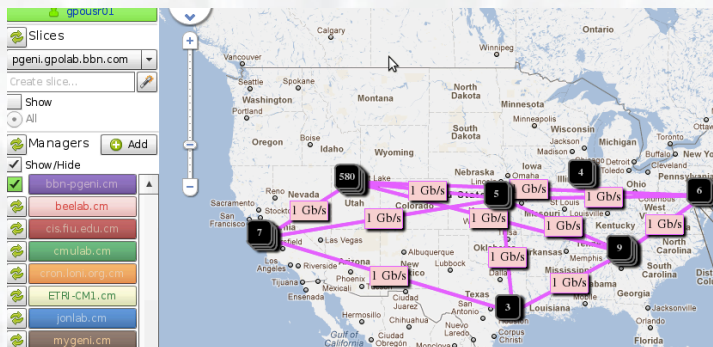
Flack, Omni



Most Experimenter Tools :

- provide a different interface to the user
- Implement the same functionality under the covers
- Interact with the CHs using the appropriate APIs
- Interact with the AMs using the GENI AM API

Graphical Tools

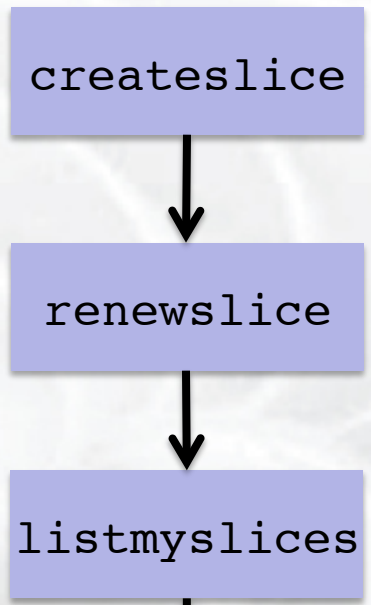


Command Line Tools

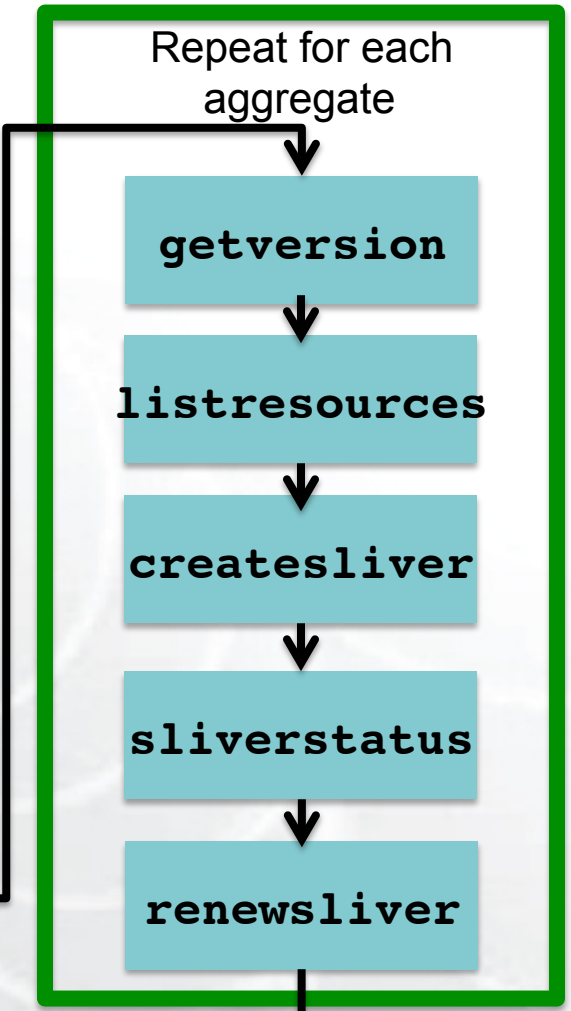
```
$ omni.py createsliver aliceslice myRSpec.xml
INFO:omni:Loading config file omni_config
INFO:omni:Using control framework pgeni
INFO:omni:Slice urn:publicid:IDN+pgeni.gpolab.
        expires within 1 day on 2011-07-07
INFO:omni:Creating sliver(s) from rspec file
INFO:omni:Writing result of createsliver for
INFO:omni:Writing to 'aliceslice-manifest-rspe
INFO:omni:-----
```


Experiment Command Workflow

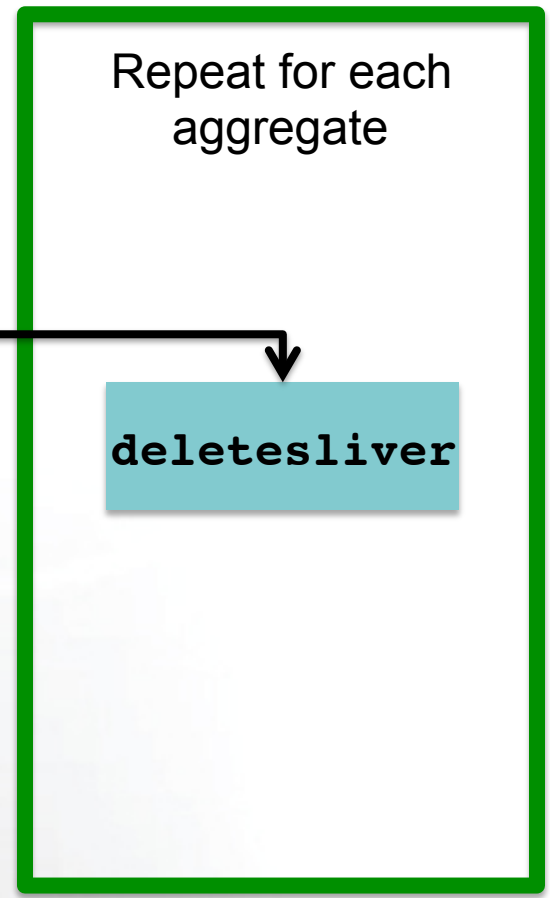
Create Slice



Create Sliver

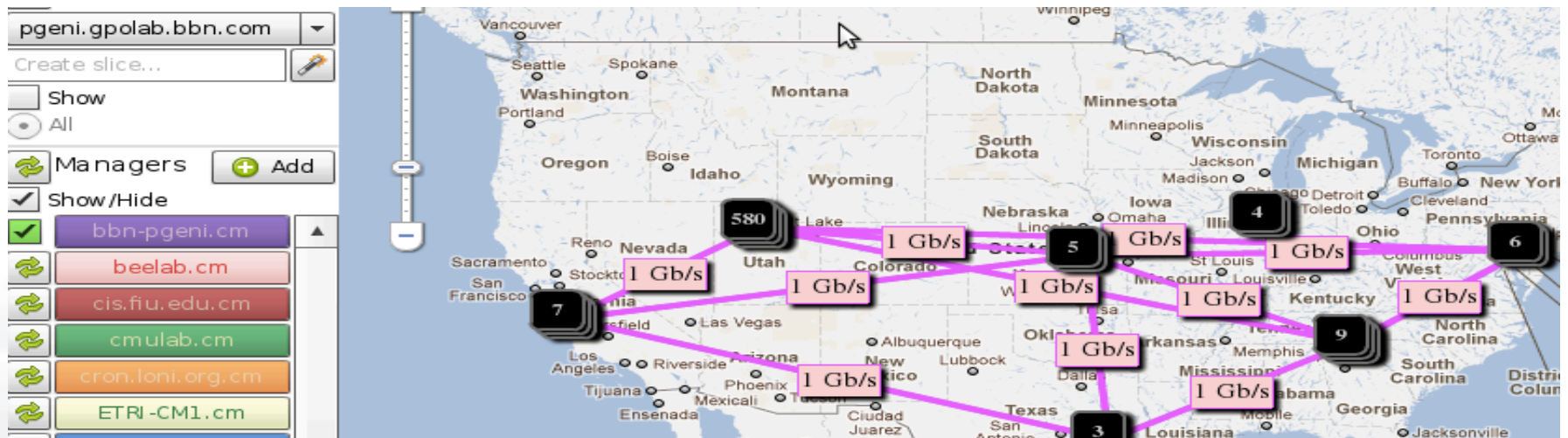


Cleanup



Legend: **AM API command**

Flack : Web-based Reservation Tool



- Web based graphical Tool
- Uses the GENI AM API
- Easy to use and get started with experiments
- Use your existing accounts (ProtoGENI, PlanetLab)
- Does not give access to Mesoscale Ams
- Is not easily scriptable
- For help use protogeni-users@googlegroups.com

Omni: Command Line Reservation Tool

- A command line experimenter tool
- Create slices and slivers using the GENI AM API
- Written in and scriptable from Python
- Use existing accounts
 - ProtoGENI
 - PlanetLab
- **Works with aggregates that implement the GENI AM API**
 - ProtoGENI, PlanetLab, OpenFlow, ...

```
$ omni.py createsliver aliceslice myRSpec.xml
INFO:omni:Loading config file omni_config
INFO:omni:Using control framework pgeni
INFO:omni:Slice urn:publicid:IDN+pgeni.gpolab.
        expires within 1 day on 2011-07-07
INFO:omni:Creating sliver(s) from rspec file
INFO:omni:Writing result of createsliver for
INFO:omni:Writing to 'aliceslice-manifest-rspe
INFO:omni: -----
INFO:omni: Completed createsliver:

Options as run:
                aggregate: https://www.emulab.
                framework: pgeni
                native: True

Args: createsliver aliceslice myRSpec.xml

Result Summary: Slice urn:publicid:IDN+pgeni
Reserved resources on https://www.emulab.net/p
Saved createsliver results to aliceslice-man
INFO:omni: =====
```

<http://trac.gpolab.bbn.com/gcf/wiki/Omni>

```
[omni]
default_cf = pgeni      # Tutorial accounts are on GPO's PG
users = gpousr21       # gpousr21's keys loaded on the VM to
allow login

# ----- Users -----
[gpousr21]
urn = urn:publicid:IDN+pgeni.gpolab.bbn.com+user+gpousr21
# Really important to get the keys correct!!!
#key to load on VM
keys = ~/Tutorials/Omni/gpousr20/ssh/gpousr20_key.pub

# default aggregates to run omni commands on
aggregates = http://emulab.net/protogeni/xmlrpc/am,
https://pgeni.gpolab.bbn.com/protogeni/xmlrpc/am, ...
```

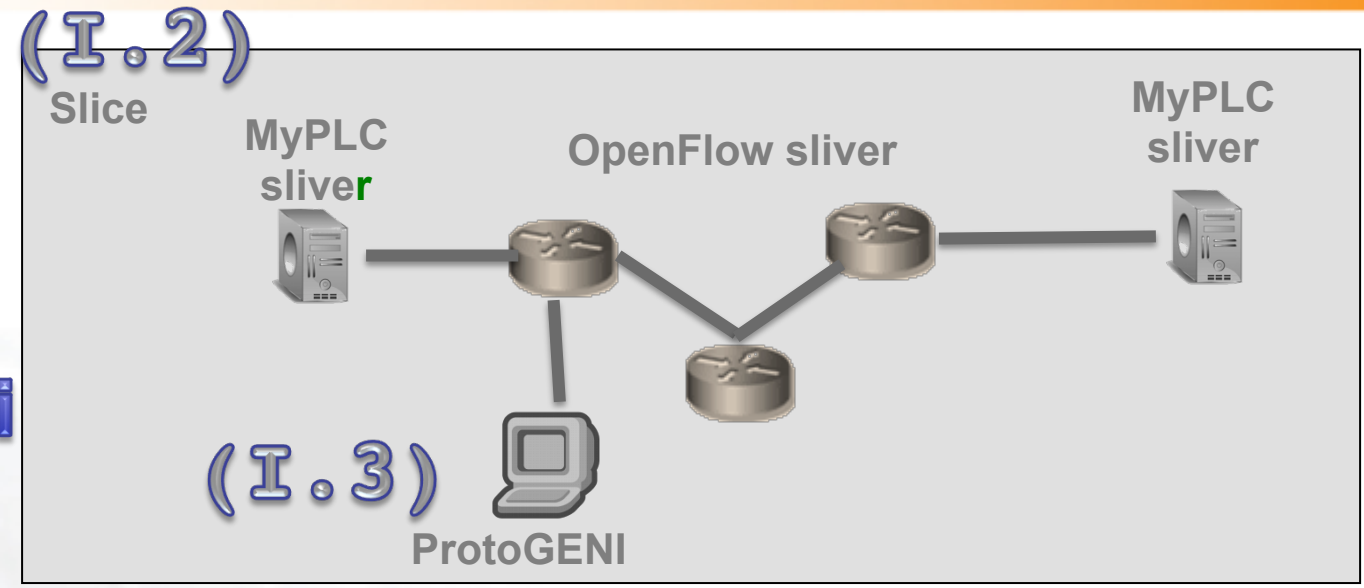
```
# ----- Frameworks -----  
[pgeni]  
type = pg  
ch = https://www.emulab.net:443/protogeni/xmlrpc/ch  
sa = https://www.pgeni.gpolab.bbn.com:443/protogeni/xmlrpc/  
sa  
  
# Tutorial certificate and key  
cert = ~/omni_tutorial/ssh/gpousr21_cert_ct.pem  
key = ~/omni_tutorial/ssh/gpousr21_cert_ct.pem  
  
#----- AM nicknames -----  
[aggregate_nicknames]  
pg-gpo=urn:publicid:IDN+pgeni.gpolab.bbn.com+authority  
+am,https://pgeni.gpolab.bbn.com/protogeni/xmlrpc/am  
plc=,https://www.planet-lab.org:12346
```

- `omni.py -h`
 - Lists all commands and their arguments
 - Lists all command line options
 - Lists Omni version
 - Lists url to find out more information about Omni
- Omni Troubleshooting page:
<http://trac.gpolab.bbn.com/gcf/wiki/OmniTroubleShoot>

- Many people will be accessing the resources, so some calls might fail. Wait a bit and try again!
- Omni is a command line tool, copy-paste is your friend
- You can copy-paste between your computer and the VM.
- There are other tutorials and demos happening, please **don't click "Submit"** when using Flack



Flack, Omni



I.1 Get to know the Tools

```
omni.py getversion
```

I.2 Make a slice

```
omni.py createslice slicename
```

```
omni.py renewslice slicename date
```

```
omni.py listmyslices username
```

I.3 Make a ProtoGENI sliver

```
omni.py createsliver slicename reqRSpec
```

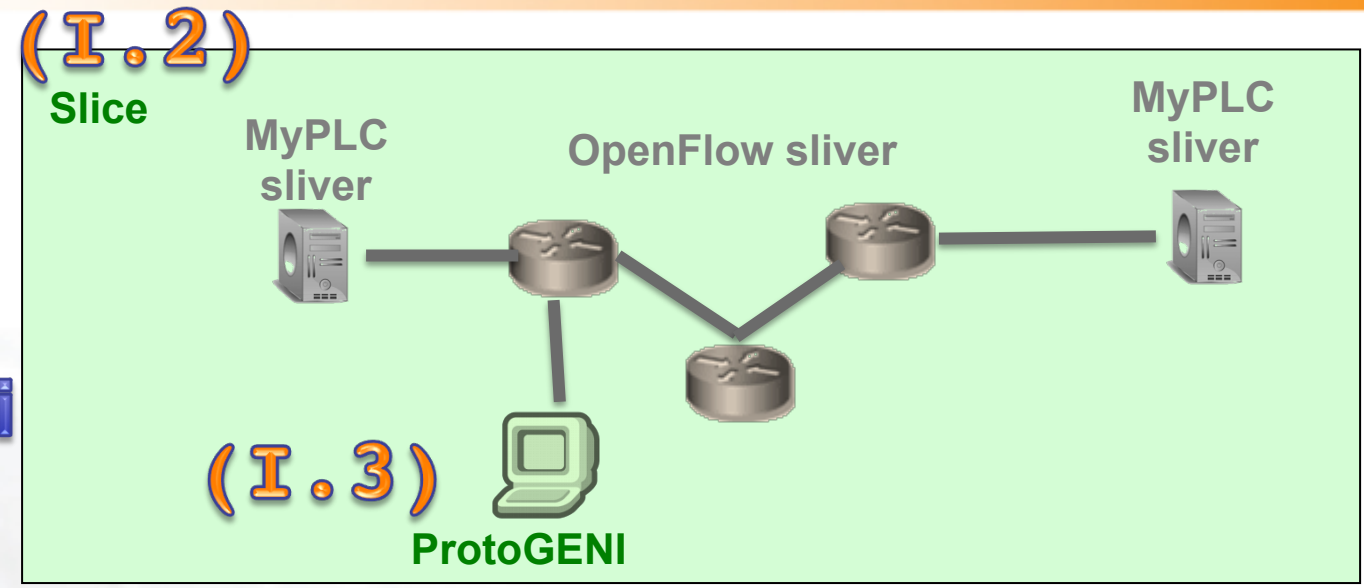
```
omni.py sliverstatus slicename
```

Note:

- a *aggregate*
to specify an aggregate manage
- o
to save the output to a file



Flack, Omni



I.1 Get to know Omni

`omni.py getversion`

I.2 Make a slice

`omni.py createslice slicename`

`omni.py renewslice slicename date`

`omni.py listmyslices username`

I.3 Make a ProtoGENI sliver

`omni.py createsliver slicename reqRSpec`

`omni.py sliverstatus slicename`

Note:

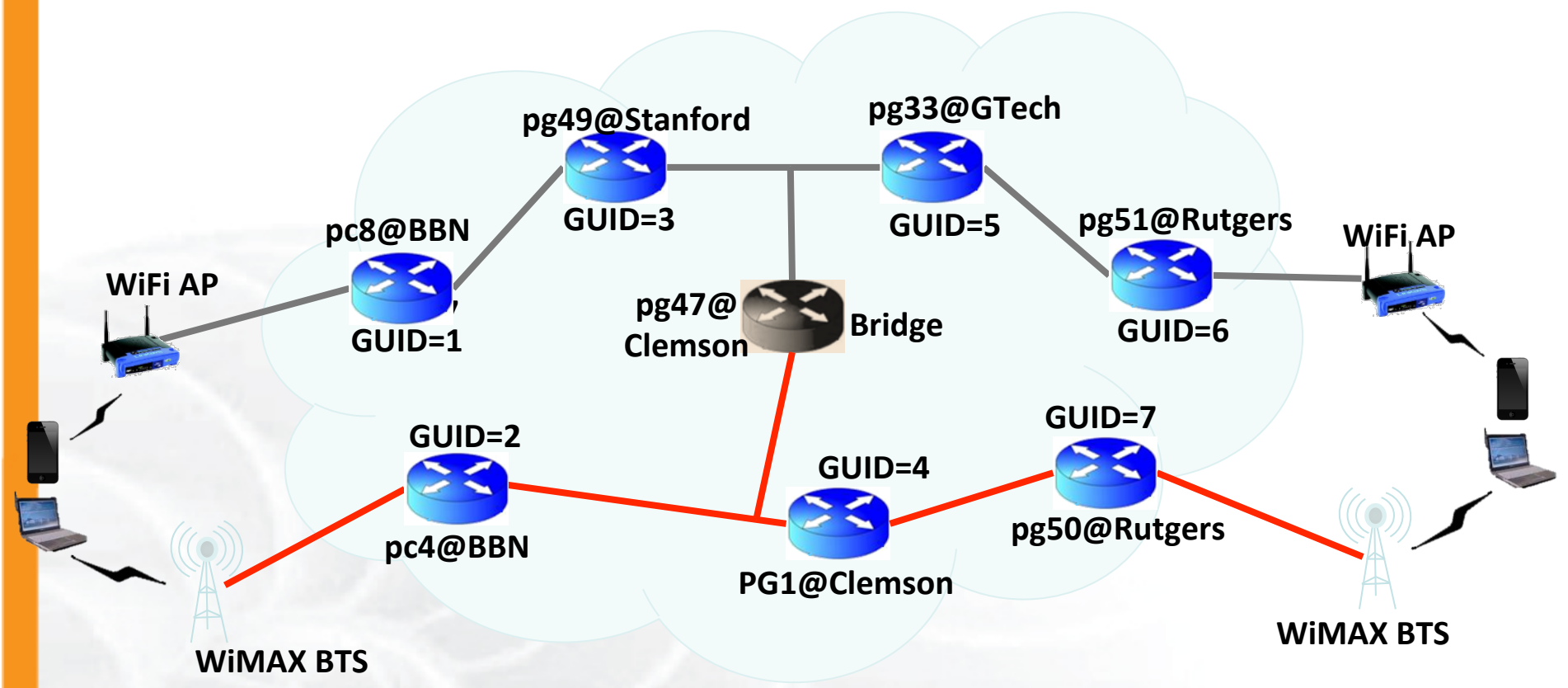
- a *aggregate*
to specify an aggregate manage
- o
to save the output to a file

OpenFlow Mesoscale deployment :

- is a prototype GENI infrastructure
- spans multiple sites connected over Layer 2
 - 2 backbone, 7 regionals, 8 campuses
- is open to experimenters that want to gain early access to a Layer 2 infrastructure that combines multiple aggregates.
- includes :
 - OpenFlow aggregates
 - Private PlanetLab aggregates (MyPLC)
 - ProtoGeni aggregates

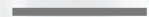
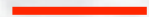

- Separate control and data plane
 - Control plane over commodity Internet
 - Data plane is Layer 2 over GENI backbone
- All hosts have one interface directly connected to an OpenFlow switch
- Pre-provisioned ~60 Dataplane IP subnets
- Out of band reservation of IP subnet
- Out of band reservation of eth_types for L2 experiments.
- **2 VLANs over the same resources, providing different topologies**

Example Experiment MobilityFirst: Mapping onto Backbone Hosts



BBN Wireless Edge

Rutgers Wireless Edge

-  NLR path using VLANs 3716, 3799 (Clemson)
-  I2 path using VLANs 3715, 3745(BBN), 3798 (Clemson)
-  ProtoGENI host running MF Router

Resource **Spec**ification Document (**RS**pec)

- XML document that describes resources
 - hosts, links, switches, etc
- today 2 different RSPEC versions are used

– **GENI RSPECs v3**

- including OpenFlow RSPECs
extension to GENI v3

– **ProtGENI Rspec v2**

```
<?xml version="1.0" encoding="UTF-8"?>
<rspec xmlns="http://www.protogeni.net/
resources/rspec/2"
xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xsi:schemaLocation="http://
www.protogeni.net/resources/rspec/2
http://
www.protogeni.net/resources/rspec/2/
request.xsd" type="request" >
  <node client_id="my-node"
exclusive="false">
    <sliver_type name="emulab-openvz" />
  </node>
</rspec>
```

omni.py getversion

```
geni@geni-vm:~/omni_tutorial$ omni.py getversion -a http://www.planet-lab.org:12346
INFO:omni:Loading config file omni_config
INFO:omni:Using control framework pgeni
INFO:omni:AM URN: unspecified_AM_URN (url: http://www.planet-lab.org:12346) has version:
INFO:omni:{  'ad_rspec_versions': [  {  'extensions': [  'http://www.protojeni.net/resource
                                                    'http://www.protojeni.net/resources/rspec/ex
                                                    'namespace': 'http://www.protojeni.net/resources/rspec/2',
                                                    'schema': 'http://www.protojeni.net/resources/rspec/2/ad.xsd',
                                                    'type': 'ProtoGENI',
                                                    'version': '2'},
          {  'extensions': [],
            'namespace': None,
            'schema': None,
            'type': 'SFA',
            'version': '1'}],
  'code_tag': '1.0-27',
  'code_url': 'git://git.onelab.eu/sfa.git@sfa-1.0-27',
  'default_ad_rspec': {  'extensions': [],
                        'namespace': None,
                        'schema': None,
                        'type': 'SFA',
                        'version': '1'},
```

```
INFO:omni: -----
INFO:omni: Completed getversion:

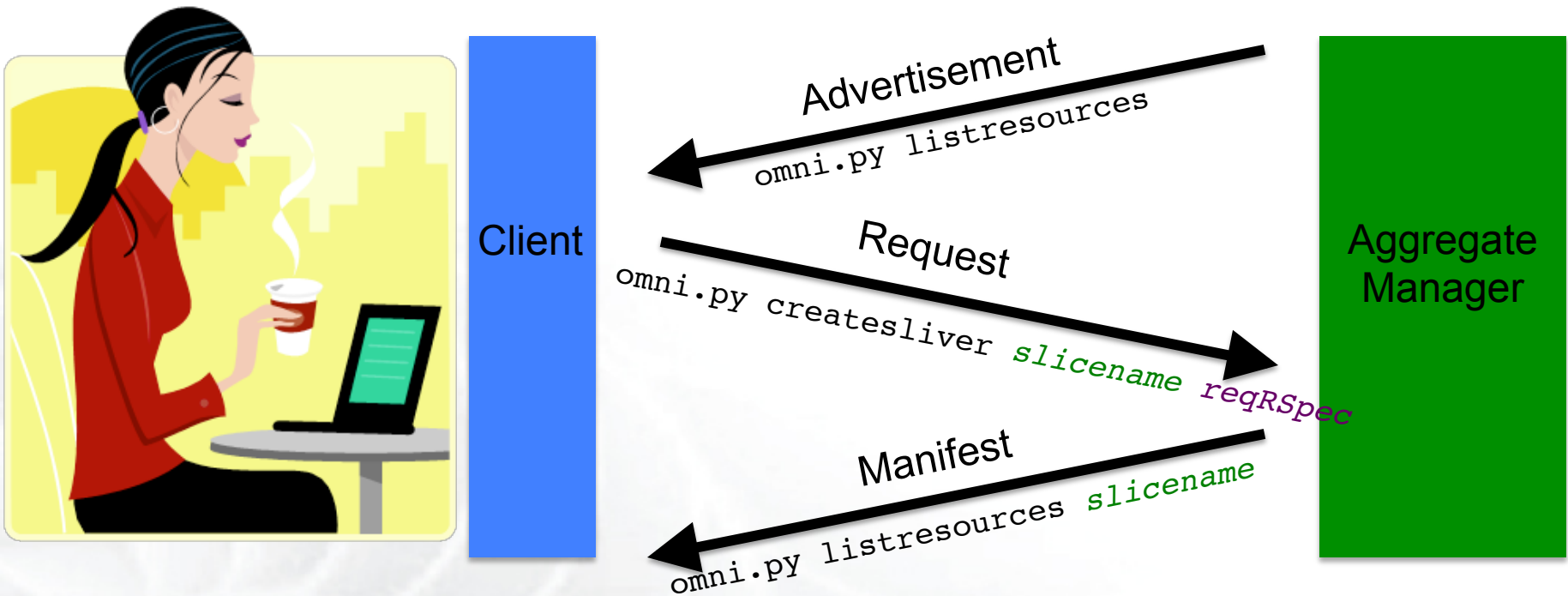
Options as run:
    aggregate: http://www.planet-lab.org:12346
    framework: pgeni
    native: True

Args: getversion

Result Summary:
Got version for 1 out of 1 aggregates

INFO:omni: =====
```


The Three Types of RSpecs



Advertisement RSpec : What does the AM have?

Request RSpec : What does the Experimenter want?

Manifest RSpec: What does the Experimenter have?

Too many RSpecs

- There is an art in writing well formed RSpecs
- Do not try to write one from scratch
 - Find example RSpecs and use them as your base
 - Use tools, like Flack, to generate sample RSpecs for you
 - When appropriate modify advertisement RSpecs



```
<rspec type="request" xsi:schemaLocation="... " xmlns="http://  
www.geni.net/resources/rspec/3">  
  <node client_id="..." component_manager_id="urn:..."  
component_id="urn:..." component_name="..." exclusive="true">  
  <sliver_type name="raw-pc">  
    <disk_image name="urn:...">  
  </sliver_type>  
  <services>  
    <execute command="..." shell="..." />  
    <install install_path="..."  
      <url="..."  
      <file_type="..." />  
  </services>  
</node>  
</rspec>
```

PlanetLab: Modifying an ad RSpec

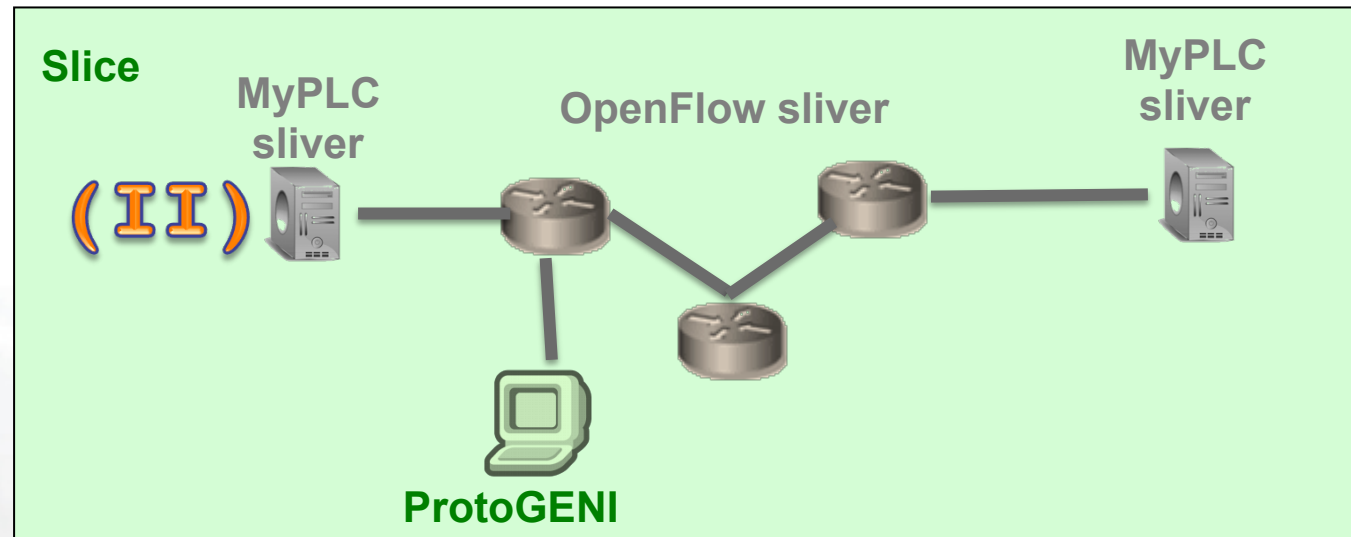
```
<RSpec type="SFA">
  <network name="plc">
    <site id="s1">
      <name>MyPLC</name>
      <node id="n1">
        <hostname> host1.geni.net </hostname>
        <sliver/>
      </node>
      <node id="n2">
        <hostname> host2.geni.net </hostname>
      </node>
    </site>
  </network>
</RSpec>
```

Insert a **<sliver/>** tag in the node tag, for the node you want to reserve

- You can write custom Python scripts
 - Call existing Omni functions
 - Parse the Output
- Example: readyToLogin.py
 - Calls sliverstatus
 - Parses output of sliverstatus
 - Determines ssh command to log into node
- More examples distributed with Omni



Omni



II.1 Make a MyPLC sliver

`omni.py createsliver slicename reqRSpec`

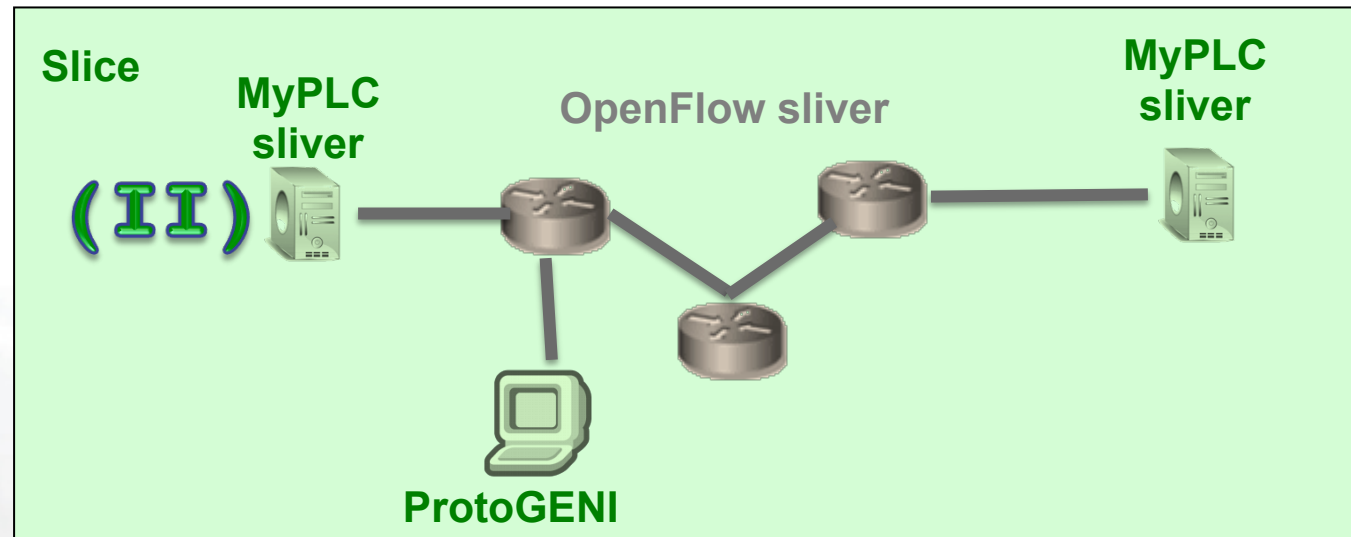
II.2 Login to the nodes

`readyToLogin.py -a AM_NICKNAME slicename`

II.3 Test Different Topologies



Omni



II.1 Make a MyPLC sliver

```
omni.py createsliver slicename reqRSpec
```

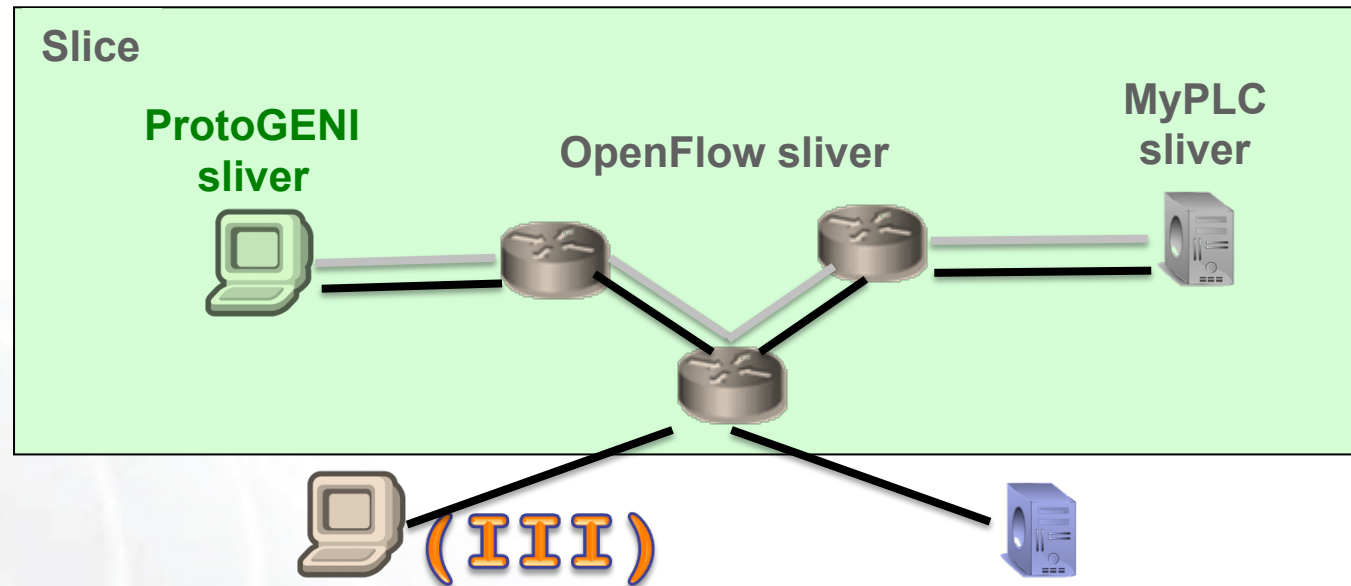
II.2 Login to the nodes

```
readyToLogin.py -a AM_NICKNAME slicename
```

II.3 Test Different Topologies

- GMOC is the GENI meta operation center which is responsible for GENI wide monitoring
 - AMs report monitoring data to GMOC
 - GMOC saves to a central DB and they are publicly available through GMOC's API
- <http://monitor.gpolab.bbn.com/tango/> is an example visualization of these statistics that you may find useful
- GPO is also running periodic health checks to mesoscale AMs and visualizes them here :

<http://monitor.gpolab.bbn.com/nagios/cgi-bin/status.cgi?host=all>



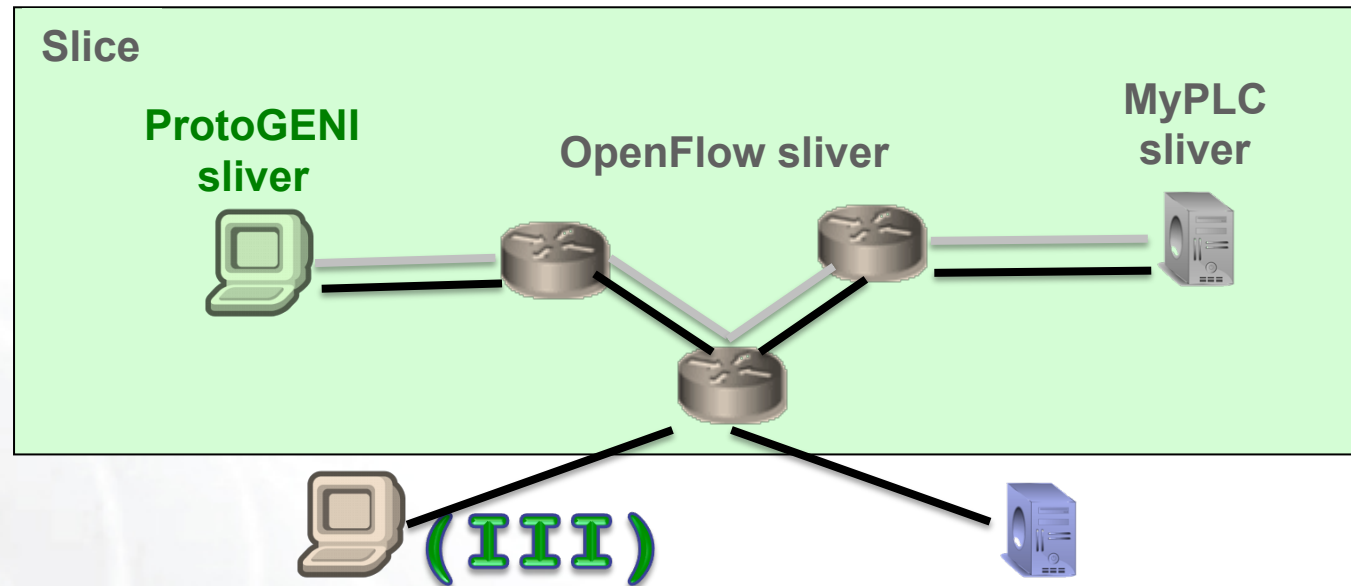
III.1 Login to your ProtoGENI node

III.2 Check the dataplane interface

III.3 Run a Layer 2 ping



Omni



III.1 Login to your ProtoGENI node

III.2 Check the dataplane interface

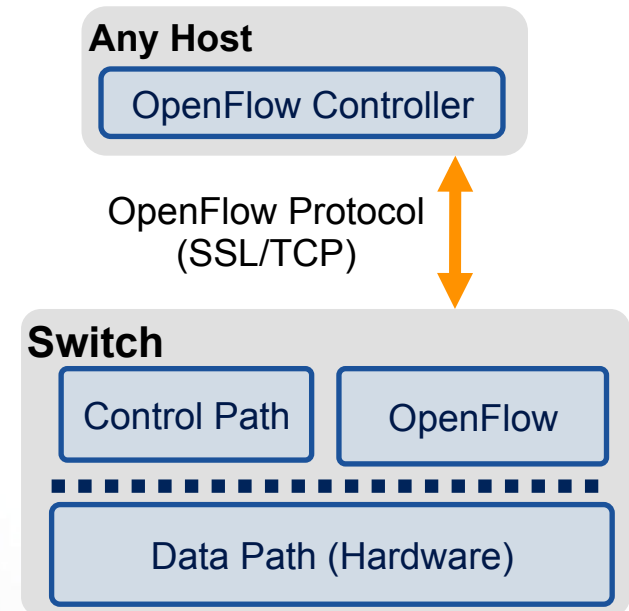
III.3 Run a Layer 2 ping

OpenFlow is an API

- Controls how packets are forwarded
- Implemented on COTS hardware
- Make deployed networks programmable

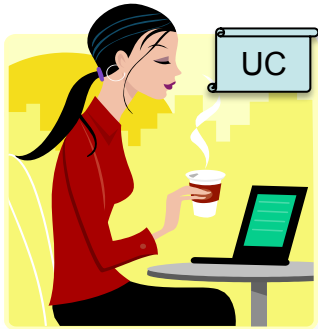
FlowSpace describes packet flows :

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

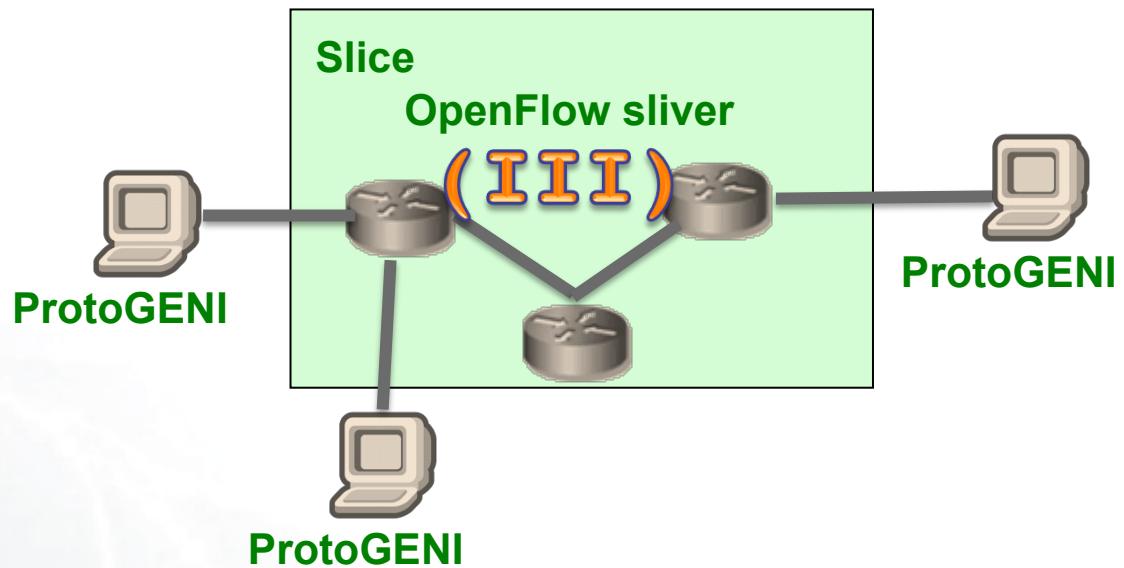


An **experimenter** can control multiple FlowSpaces

- Use OpenFlow to route your packets in the network
- Use OpenFlow to rewrite header fields :
 - `OFPAT_SET_VLAN_VID,` `/* Set the 802.1q VLAN id. */`
 - `OFPAT_SET_VLAN_PCP,` `/* Set the 802.1q priority. */`
 - `OFPAT_STRIP_VLAN,` `/* Strip the 802.1q header. */`
 - `OFPAT_SET_DL_SRC,` `/* Ethernet source address. */`
 - `OFPAT_SET_DL_DST,` `/* Ethernet destination address. */`
 - `OFPAT_SET_NW_SRC,` `/* IP source address. */`
 - `OFPAT_SET_NW_DST,` `/* IP destination address. */`
 - `OFPAT_SET_NW_TOS,` `/* IP ToS (DSCP field, 6 bits). */`
 - `OFPAT_SET_TP_SRC,` `/* TCP/UDP source port. */`
 - `OFPAT_SET_TP_DST,` `/* TCP/UDP destination port. */`
- Caveat, not all actions are done in hardware (OVS)



Omni



III.1 Make an OpenFlow sliver

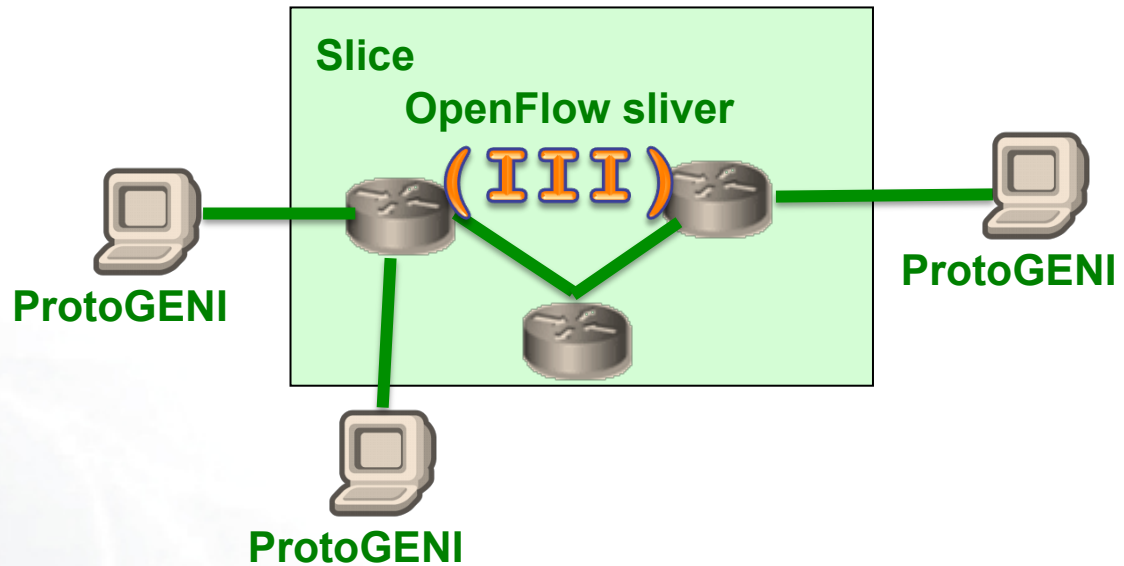
`omni.py createsliver slicename reqRSpec`

III.2 Run your OpenFlow controller

III.3 Run Layer 2 pings



Omni



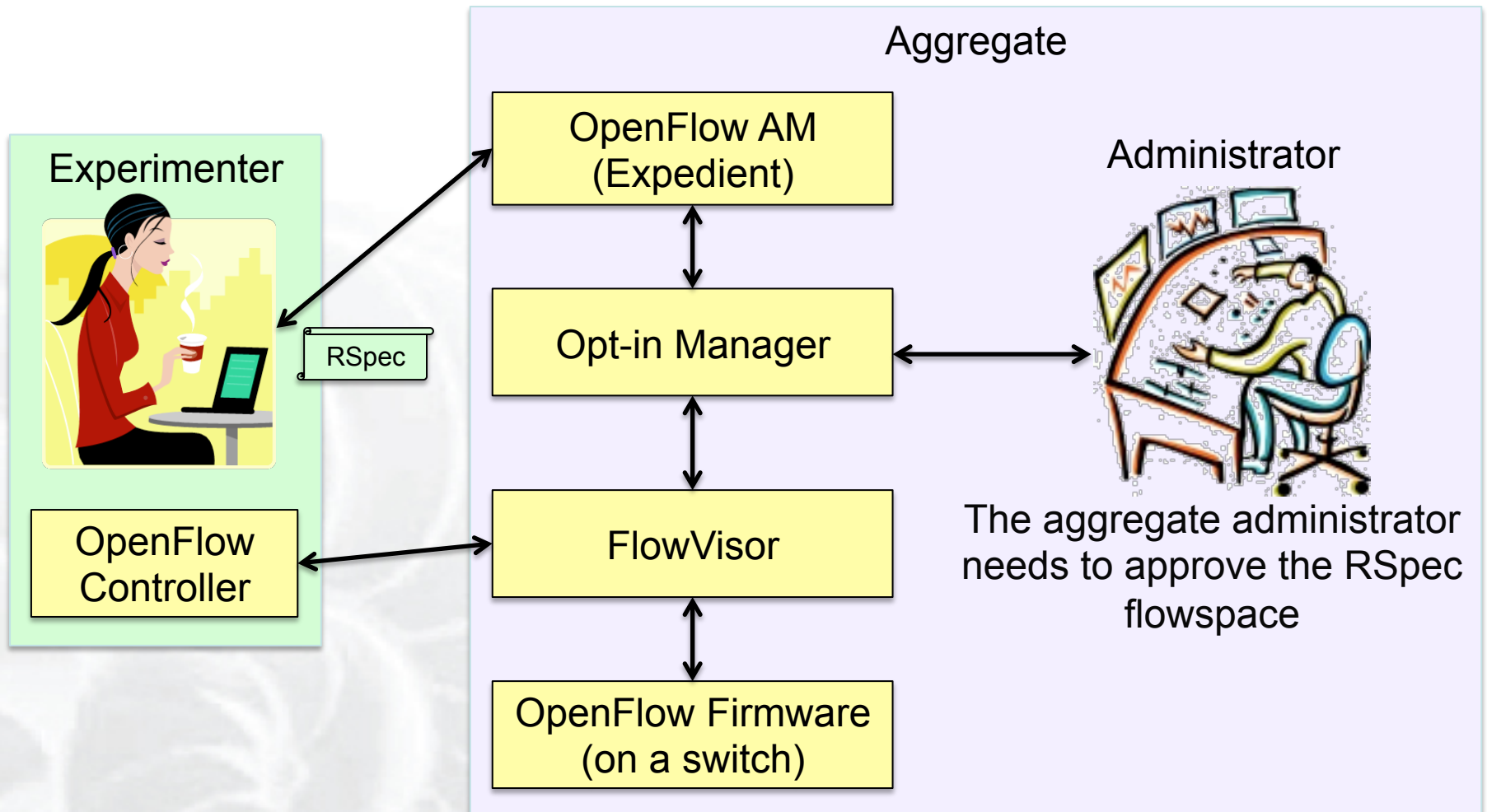
III.1 Make an OpenFlow sliver

`omni.py createsliver slicename reqRSpec`

III.2 Run your OpenFlow controller

III.3 Run Layer 2 pings

OpenFlow in GENI



```
<resv_rspec type="openflow" version="2">  
  <user ... />  
  <project ... />  
  <slice controller_url="tcp:host:port" expiry="1326139200" />  
  <flowspace>  
    <switch urn="urn: ... " />  
    <port urn="urn: ... " />  
    <pkt_field from="..." to="..." />  
  </flowspace>>
```

expiry is in Unix timestamp

pkt_field can be :

- dl_src, dl_dst, dl_type, dl_vlan
- nw_src, nw_dst, nw_proto, nw_tos
- tp_src, tp_dst

- When your experiment is done, you should always release your resources.
 - Archive your data
 - Delete all your slivers
 - OpenFlow slivers might outlive your slice, make sure you delete them before your slice expires
 - When appropriate delete your slice

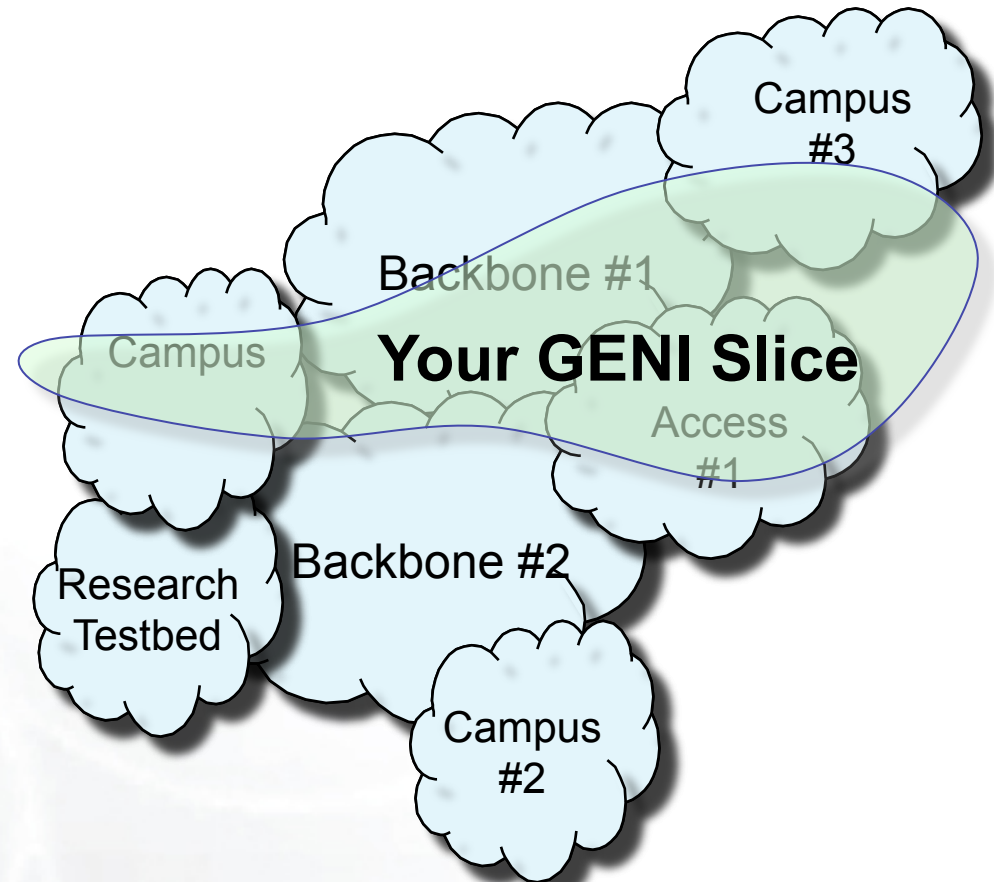


Running Experiments on GENI

- Get an account to run experiments on GENI

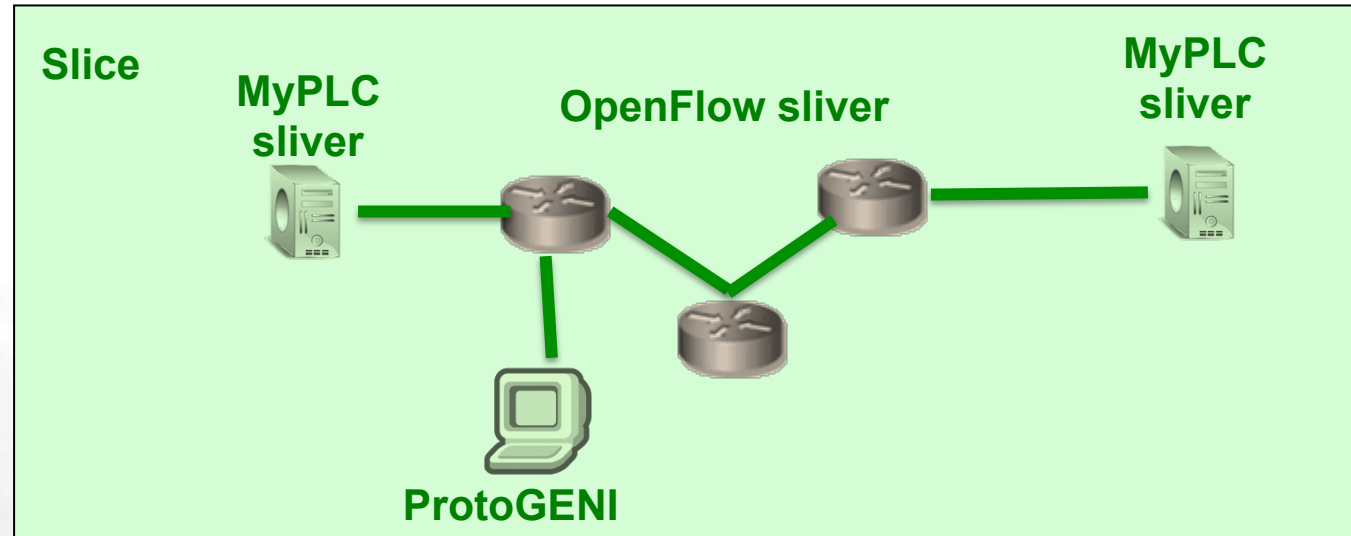
- Contact us at help@geni.net

- More information on Experimenter Portal:
 - <http://groups.geni.net/geni/wiki/ExperimenterPortal>





Omni



V. Cleanup resources

`omni.py deletesliver slicename`

VI. Request your own account

– We will help you get GENI credentials



Flack



Omni

Happy experimenting!

Backup Slides

- Primary Information
 - `omni.py -h`
 - Omni Troubleshooting page:
<http://trac.gpolab.bbn.com/gcf/wiki/OmniTroubleShoot>
 - For Omni specific help: gcf-dev@geni.net
 - For general GENI help: help@geni.net
- Omni Wiki (install instructions, documentation, bug reporting):
<http://trac.gpolab.bbn.com/gcf/wiki/Omni>
- For an overview of GENI Experimentation using Omni:
 - <http://groups.geni.net/geni/wiki/GENIExperimenter>
- Example experiment walk-through:
 - <http://groups.geni.net/geni/wiki/GENIExperimenter/ExperimentExample>
- Example script walk-throughs:
 - <http://trac.gpolab.bbn.com/gcf/wiki/OmniScriptingWithOptions> and
<http://trac.gpolab.bbn.com/gcf/wiki/OmniScriptingExpiration>

- `omni.py getversion`
- `omni.py createslice slicename`
- `omni.py renewslice slicename date`
- `omni.py listmyslices username`
- `omni.py createsliver slicename requestRSpec`
- `omni.py sliverstatus slicename`
- `omni.py listresources [slicename]`
 - t ProtoGENI 2 to request PGV2 Rspecs
- `omni.py deletesliver slicename`

Other Omni command line arguments

- c *omni_config* to use another *omni_config*
- f *plc* to use a different framework
- t ProtoGENI 2 to specify the version of the Rspec