Experiences from Deployment of LAMP/PerfSONAR to explore "Infrastructure Measurement Slices" in GENI

Shriram R Ramamurthy, shriram@oar.net; Prasad Calyam, pcalyam@oar.net

(Work in Progress)

January 2012

1. Purpose of this Document

This document describes findings from our experiments to deploy LAMP/perfSONAR (http://groups.geni.net/geni/wiki/LAMP) on the GENI infrastructure and to explore the instrumentation and measurement related capabilities for setting up "infrastructure measurement slices".

Although a LAMP tutorial exists at: http://groups.geni.net/geni/wiki/LAMP/Tutorial, we initially faced difficulties in re-producing the tutorial steps. In the course of our experiments between October 2011 to December 2011, we worked closely with the LAMP project team to get our issues resolved, and also provided feedback to them to update their documentation and scripts to help future experimenters.

2. Findings Summary

- **Resolved Issues**

1. In the first steps, we encountered a Hardware Error problem; there was a hardware error being thrown during the process of creating sliver or getting a ticket to the sliver creation. This error stated: OS 'GeniSlices/UBUNTU91-LAMP' (OS-2283) does not run on this hardware type! This error was reported to the LAMP team and they sent us a patch update that resolved this problem.

2. LAMP does not support the version 2 RSpec format. The lamp-sendmanifest.py script's execution threw an error indicating the incompatible RSpec version. Also, when the code was manually modified to use the correct RSpec version, the PEM Passphrase request did not pop up. This problem was reported and finally we got a corrected script from the LAMP team. The corrected script has now been posted on the main LAMP wiki.

3. The error-handling messages were not informative; the wrong script that was hosted at the main LAMP wiki, when used, did not throw an error, when the PEM Passphrase was not prompted. The process continued and the lamp-getcertificate.py script threw an error.

4. The documentation for bundle certificates was not mentioned in the main LAMP wiki, this resulted in a problem where the tests on being scheduled and pushed to the UNIS, cannot be pulled to view the results. This bug was fixed after the LAMP team got back saying that the 'getacerts script' at /usr/local/etc/protogeni/ssl/getcacerts needs to be run before the systems are

rebooted. This was not run earlier as the main LAMP wiki said that the bootstrap command invokes this script, but in our case as well as in other cases as mentioned to us by the LAMP team, it failed to do so.

- **Open Issues**

1. The slice renewal is a problem. The slice currently expires automatically after around 6 hours and we lose all the data, and we have to follow all of the installation steps again to continue running LAMP in our slice. We are trying to automatically kick off a renewal script run via the crontab feature in Linux to see if we can keep the slice up and running for extended durations.

2. BWCTL (throughput) results were not available after the tests were pushed to the UNIS server, and the LAMP team suggested that we get back to see the results after some considerable time has passed by, but still the results are not available. We plan to follow-up with the LAMP team about getting this issue resolved.

3. Technical Terms

What is a LAMP Portal?

A useful resource for experimenters, which enables configuration, query and visualization of I&M services data. When a slice with LAMP on multiple-nodes is setup, there is a LAMP image deployed on a separate node that hosts the portal, which can then be used as a starting point to interact with the other nodes and the services in the slice.

What is UNIS?

Unified Network Information Service provides a combined service: Lookup Service, Topology Service. The topology data of the slice (specified in the RSpec) needs to be uploaded to the UNIS. After the upload, users can interact with the perfSONAR services within the slice.

4. Step-by-step Process that worked for us

We now describe the steps we took to create a slice, getting a LAMP certificate for the slice, uploading the topology to UNIS, and finally enabling the instrumentation and measurement services for the slice.

<u>Step-1: Creating the RSpec</u>

We need to create an RSpec, which contains various resource specifications and topology. The LAMP uses a variant of the UBUNTU image. This wiki follows the usage of the default RSpec that has been provided in the LAMP wiki.

```
<rspec xmlns="http://protogeni.net/resources/rspec/0.2"
  xmlns:lamp="http://protogeni.net/resources/rspec/0.2/ext/lamp/1">

  <node virtual_id="node1" virtualization_type="raw" exclusive="1"
    startup_command="/usr/local/etc/lamp/bootstrap.sh
urn:publicid:IDN+emulab.net+slice+slice_name
```

```
urn:publicid:IDN+emulab.net+user+user_name">
    <node_type type_name="pc" type_slots="1"/>
    <disk_image name="urn:publicid:IDN+emulab.net+image+GeniSlices//UBUNTU91-
LAMP" />
        <lamp:config />
        <interface virtual_id="iface0"/>
  </node>
    <node virtual_id="node2" virtualization_type="raw" exclusive="1"
    startup_command="/usr/local/etc/lamp/bootstrap.sh
urn:publicid:IDN+emulab.net+slice+slice_name
urn:publicid:IDN+emulab.net+user+user_name">
    <node_type type_name="pc" type_slots="1"/>
    <disk_image name="urn:publicid:IDN+emulab.net+image+GeniSlices//UBUNTU91-
LAMP" />
    <lamp:config />
    <interface virtual_id="iface0"/>
  </node>
    <link virtual_id="link1" >
    <interface_ref virtual_node_id="node1" virtual_interface_id="iface0"/>
    <interface_ref virtual_node_id="node2" virtual_interface_id="iface0"/>
    <link_type type_name="ethernet" />
    <latency>100</latency>
    <packet_loss>0.05</packet_loss>
  </link>
    <node virtual_id="lamp" virtualization_type="raw" exclusive="1"
    startup_command="/usr/local/etc/lamp/bootstrap.sh
urn:publicid:IDN+emulab.net+slice+slice_name
urn:publicid:IDN+emulab.net+user+user_name">
    <node_type type_name="pc" type_slots="1"/>
    <disk_image name="urn:publicid:IDN+emulab.net+image+GeniSlices//UBUNTU91-
LAMP" />
        <lamp:config>
      <lamp:service type="lamp_portal" enable="true" />
    </lamp:config>
  </node>
</rspec>
```

NOTE: The RSpec has a bootstrap script i.e., the getacerts script, which needs to be run once the system boots. If we do not specify this, we need to run it every time we restart the perfSONAR services. As pointed out in the Section 2 (Resolved Issues), often this script has to be run manually even though you specify it in the bootstrap.

NOTE: After completion of Step-1, you should download the protogeni-testscripts bundle for further progress. These scripts are available at - http://www.emulab.net/downloads/protogeni-tests.tar.gz. *Then make sure that your Mac/PC is running Python version 2.6.x. This is a strict requirement*. The protogeni-

testscripts are compatible with Python 2.6.x alone. And you may get error throws if you try with Python 2.7 even though it's an enhanced release. *There must also be an M2Crypto Python Library present on your Mac/PC*. Absence of this library may cause Certificate Errors. Make sure you are using version 0.20.1 or above of M2Crypto as recommended by the official website – http://chandlerproject.org/bin/view/Projects/MeTooCrypto.

NOTE: You may need to resolve a set of dependencies when trying to import the M2Crypto library for Python2.6. Many OS may have the latest version or some other versions of Python, so once the Python2.6 tar ball has been downloaded and installed, we need to go ahead and do `ln` command for changing the hard link to python. Usually the executable of Python is in /usr/bin or /usr/local/bin, `which python` command will tell the path, after getting the path do `ln –s path/to/python2.6 existing/python/path`. M2Crypto is dependent on **SWIG** that has a Perl regular expression parser and hence will require **PCRE**. The PCRE latest builds can be obtained at http://www.pcre.org/. Once PCRE has been installed, then we can go ahead and install SWIG. The latest builds can be found at http://www.swig.org/download.html. Now make sure that the **OpenSSL** that is installed on your Mac/PC has a version of at least 0.9.8. OpenSSL might require the `openssl-devel install` for perfect compiling of M2Crypto. M2Crypto uses the SWIG features, which are essential for building the M2Crypto package. They are: -python -I/usr/local/include/python2.6 -I/usr/include/openssl -includeall -D__i386__ -cpperraswarn. *These flags are essential for a graceful build of M2Crypto*. For Fedora users, try the `sh fedora_setup.sh build` and `sh fedora_setup.sh install`. This will fix the include flag dependency, by using the proper SWIG features. If this still throws dependency errors of missing config files of SWIG, try uncommenting the `finalize_options` function's code parts of the setup.py file, in the M2Crypto untar directory. Please reboot your Mac/PC at the end to make the changes effective.

Step-2: Slice Creation

Next we need to create a slice. Before starting the process, please make sure you have your public key uploaded to the Emulab profile and also have the public/private pair in ~/.ssl directory. Then also make sure that you have the SSL certificate generated in the Emulab profile and have it download into the ~/.ssl/encrypted.pem.
For more information in the slice creation and certificate generation in Emulab, please visit: http://www.protogeni.net/trac/protogeni/wiki/Tutorial.

Step-3: Get Credential

We need to get the credential from the Slice Authority (SA) that grants the user a signed document containing the privileges and permissions. This is a credential provided to the user by the local SA i.e., Emulab in our case. Please run the getcredential script from the protogeni-testscripts bundle.

```
python getcredential.py
```

This provides us with the signed credential.

Step-4: Register a Slice

Now we need to register a slice name. Please run the `registerslice` script.

python registerslice.py –n slice_name

The –n option if overridden, we get a default slice created as mytestslice. So please make sure you provide the –n option else don't use the –n option along with any other script.

Step-5: Allocating Resources

Now we need to allocate the resources. This can be done in two ways. We can either request a ticket and redeem a ticket and then renew a ticket or we can go ahead with the `createsliver` option (recommended option!)

Once you create a slice, we next need to establish a sliver which can hold the resources. Please run the createsliver script.

python createsliver.py –n slice_name rspec_file.xml

This returns the manifest returned for this instance, which we can store in the lamp-manifest.xml file. You may not get a manifest return here, as createsliver.py is a bundled script, which does all the operations of requesting and redeeming a ticket. So in this case please use the `getmanifest.py` script, which is a part of the protogeni-testscripts bundle, to get the output throw of the manifest.

Once we have created the sliver, we need to renew it before we start any experiments. The `createsliver` script, creates the sliver for a specific amount of time, which usually is not enough to do prolonged work, and suddenly you lose the sliver which causes the work to stop abruptly. So we also renew it side by side.

python renewslice.py –n slice_name time_to_renew_in_minutes

NOTE: Renewing the slice automatically renews the containing sliver too. You can find it in the output throw

Step-6: Upload to UNIS

After creating the slice and the sliver and renewing the sliver too, we need to upload the topology to the UNIS. The topology, which is represented as a RSpec, needs to be converted into a UNIS schema. This is taken care by the script `lamp-sendmanifest.py`. But this script has some requirements. The UNIS will always require the slice credential for identification, and also the manifest which we obtained earlier when we allocated the resources.

We use the `getslicecredential` script for the obtaining the slice credential.

python getslicecredential.py –n slice_name > lamp-credential.xml

NOTE: We need to remove the first line in the stored file lamp-credential.xml, so that the signature check sees it as a signature.

python lamp-sendmanifest.py lamp-manifest.xml
urn:publicid:IDN+emulab.net+slice+slice_name lamp-credential.xml

We get an output throw, which shows the message we sent to UNIS. Then we get a prompt for pass phrase. Upon the correct pass phrase enter; we get the data elements successfully replaced message. This means that the topology is now uploaded successfully in the UNIS. Use the `lamp-getmanifest.py` script to obtain the lamp-manifest.xml file's content.

Step-7: LAMP Certificate

We now have to upload the LAMP certificate to all the nodes in our slice. We need to upload the certificate to the /usr/local/etc/protogeni/ssl/lampcert.pem. *The file name in the nodes must for sure be lampcert.pem*. But you may have any file name in your local directory.

python lamp-getcertificate.py –n slice_name

The resulting certificate throw, needs to be stored in a file and then uploaded to the location as specified above. [Assume it is stored in lampcert.pem in the local directory also].

grep "login" lamp-manifest.xml

bash-3.2$ grep "login" lamp-manifest.xml
<services><login authentication="ssh-keys" hostname="pc117.emulab.net" port="22" username=user_name/></services></node>
<services><login authentication="ssh-keys" hostname="pc104.emulab.net" port="22" username=user_name/></services></node>
<services><login authentication="ssh-keys" hostname="pc142.emulab.net" port="22" username=user_name/></services></node>
bash-3.2$

We need to upload the lampcert.pem in our local directory to the /usr/local/etc/protogeni/ssl/lampcert.pem in the nodes we have obtained above. Now all we need to do is to ssh to the nodes above and load this lampcert.pem file into the specified location.

NOTE: You may get a permission denied when you try to ssh into the nodes. Use the –i flag of the ssh to direct the ssh to read from the ~/.ssl/your_private_key.

Now after uploading the lampcert.pem from the local directory to the location in the nodes [this can either be achieved by scp or by simply copy pasting after ssh to the nodes], we need to run the following shell script:

bash-3.2$ cat shell.sh
for node in node1 node2 node3; do
ssh -i ~/.ssl/your_private_key user_name@$node "sudo mv lampcert.pem /usr/local/etc/protogeni/ssl/lampcert.pem"
ssh -i ~/.ssl/your_private_key user_name@$node "sudo chown root.perfsonar /usr/local/etc/protogeni/ssl/lampcert.pem"

ssh -i ~/.ssl/your_private_key user_name@$node "sudo chmod 440 /usr/local/etc/protogeni/ssl/lampcert.pem"
ssh –i ~/.ssl/your_private_key user_name@$node "sudo /usr/local/etc/lamp/bootstrap.sh urn:publicid:IDN+emulab.net+slice+geni1 urn:publicid:IDN+emulab.net+user+shriram"
ssh -i ~/.ssl/your_private_key user_name@$node "sudo perl /usr/local/etc/protogeni/ssl/getcacerts"
ssh -i ~/.ssl/your_private_key user_name@$node "sudo /etc/init.d/psconfig restart"
done
bash-3.2$

Step-8: Browser Access
Now after restarting the perfSONAR-ps Services in the nodes, we can access the LAMP portal.

NOTE: To access the LAMP portal, we need to upload a certificate to the browser. Follow the below steps to upload a certificate if you are using the Firefox web-browser. The portal access works fine with Firefox and Google Chrome. It does not work well with Safari.

- Go to your user profile page on the Utah Emulab.
- Click 'Generate SSL Cert' on the left.
- Enter a password and remember it.
- Immediately click on the pkc12 format download and store it locally.
- Then feel free to click on the Download [1st option] option and copy the certificate and paste it in the ~/.ssl/encrypted.pem. This is to update your local certificate as you have changed it now.
- Open Firefox, select Preferences [command+, for mac]
- Select Advanced Tab and choose Encryption sub tab.
- Check the Select One Automatically radio button.
- Click View Certificates and select Your Certificates tab.
- Import the pkc12 format you downloaded locally and save the settings.
- Now open the https://pc142.emulab.net/lamp/ [Here pc142 is the lamp node]
- **Only the lamp node, see the Rspec, can be accessed via the web portal.**
- Then Select the Add Exception, and you will reach the lamp portal of the node.

Step-9: Completing LAMP Deployment
The rest of the steps are as specified in the main LAMP wiki:
http://groups.geni.net/geni/wiki/LAMP/Tutorial from **Step 6.**

5. Conclusion

In the above, we have described the complete details of the first five steps along with the **[NOTE] tags,** which are likely the error possibilities in getting LAMP successfully deployed in your GENI slice. Although the main LAMP wiki is very helpful, and the LAMP team is very responsive to help requests, our overall experience showed that the documentation in the main LAMP wiki needs to be more informative. Also, due to the dependencies of the LAMP software, there might be other challenges that may arise across Mac/PC platforms.

6. References

[1] http://groups.geni.net/geni/wiki/LAMP/Tutorial
[2] http://www.protogeni.net/trac/protogeni/wiki/Tutorial
[3] https://www.protogeni.net/trac/protogeni/wiki/FlashClientSetup
[4] http://groups.geni.net/geni/wiki/GIR3.2_LAMP
[5] Inputs from: Matthew Jaffee mjaffee@indiana.edu; lamp@damsl.cis.udel.edu