Dynamic Slice Creation HOWTO

Brent Chun

This document describes how to create, manage, and delete dynamic slices on PlanetLab using dslice (http://dslice.planet-lab.org), a prototype implementation of dynamic slice creation. Currently, only users with principle investigator privileges are allowed to perform these operations. But don't let this deter you from reading on (and offering feedback), since this will be changing in the near future.

Table of Contents

1. Overview

dslice (http://dslice.planet-lab.org) is a prototype implementation of dynamic slice creation as described in PDN-02-005 (http://www.planet-lab.org/pdn/pdn02-005.pdf). (Please read the PDN-02-005 before continuing.) It consists of an agent daemon, a node manager daemon, and a service manager (a command-line program) which includes an integrated broker. In the current deployment, an agent running on a well-known machine (`dslice.planet-lab.org`) issues tickets for all nodes. Each node runs a node manager that implements the node's admission control policy and handles lease and sliver management requests. Slices are created using a service manager. Creating a slice entails: (1) obtaining a set of tickets for a set of nodes for a slice through a broker (which in turn contacts an agent), (2) redeeming the tickets for leases by contacting each node manager directly, and (3) using the leases to create virtual machines for each node in the slice. Once created, a slice is further managed by a service manager to perform operations such as adding/removing SSH keys, renewing leases for the slice, etc. Once the appropriate SSH keys have been added, a slice can then be accessed remotely by users using SSH. The slice itself also has a per-slice SSH key pair that allows users to SSH between any node in the slice without a password. This document describes how to install the dslice software and use it to create, manage, and delete dynamic slices on PlanetLab.

2. Installing the Software

All the software required on the user's end is contained in a single Linux RPM (dslice) and can be downloaded off of the dslice web page: http://dslice.planet-lab.org. The RPM depends on python2.2, expect, and openssl. On Redhat 7.3, these RPMs should already be installed. If not, they can downloaded here (http://dslice.planet-lab.org/misc/).

Once you are ready to install the dslice RPM, use the `rpm` command as shown below. (Note that the example below is for dslice 0.1.0. However, we recommend that you always use the latest version of the software. The latest version can always be found at http://dslice.planet-lab.org.)

```
root# rpm -ivh dslice-0.1.0-1.i386.rpm
Preparing...                ########################################### [100%]
    1:dslice                ########################################### [100%]
```

The RPM installs all of its files in `/usr/local/planetlab`. To make the programs and documention in this directory accessible, add `/usr/local/planetlab/bin` to your PATH and `/usr/local/planetlab/man` to your MANPATH. For bash/zsh users (you may want to add this to the appropriate dotfile):

```
# export PATH=$PATH:/usr/local/planetlab/bin
# export MANPATH=$MANPATH:/usr/local/planetlab/man
```

For csh/tcsh users (you may want to add this to the appropriate dotfile):

```
# setenv PATH ${PATH}:/usr/local/planetlab/bin
# setenv MANPATH ${MANPATH}:/usr/local/planetlab/man
```

3. Setting up your .planetlab Directory

The first thing that needs to be done before anything else is to create your `.planetlab` directory. This directory contains your RSA key pair and X509 certificate for authentication along with important files related to your slices (e.g., tickets, leases, etc.). This directory is automatically set up for you using the `plkeygen` command as shown

below. This command will create your .planetlab directory, populate it, and send an X509 certificate request to the PlanetLab CA for approval. Use your email/passwd from your PlanetLab web account (https://www.planet-lab.org/db/login/login.php) when prompted for this information. Currently, only users with principle investigator privileges are allowed to create, delete, and manage dynamic slices on PlanetLab. This will be changing in the near future.

```
# plkeygen
email: llp@cs.princeton.edu
password: ******
Certificate request sent to bnc@intel-research.net.
Do a 'plkeygen --refresh' once you receive a reply.
```

At this point, the PlanetLab CA will receive an email containing your certificate request. Upon inspecting it and verifying its contents, an X509 certificate will be created and an email will be sent back to you confirming the availability of your new X509 certificate.

Once you receive an email confirming the availability of your X509 certificate, your certificate can then be downloaded as shown below. The `plkeygen` command will retrieve your certificate and save it in the appropriate place in your `.planetlab` directory.

```
# plkeygen --refresh
email: llp@cs.princeton.edu
Certificate downloaded.
```

4. Creating and Deleting Slices

4.1. Creating a Slice

Slices are created by (1) creating an XML file that describes the slice's requirements and (2) using the service manager, `svm`, to contact the relevant entities in the system to obtain tickets and leases and create a network of virtual machines on a set of nodes. Before doing either of the above, you might want to first query the system to see how many nodes are available from the default agent running on `dslice.planet-lab.org`. You can see how many nodes are being advertised by this agent by using the `svm` command as shown below:

```
# svm getads dslice.planet-lab.org
150.135.65.2
150.135.65.3
131.215.45.71
131.215.45.72
128.232.103.203
....
```

A slice XML file currently specifies a slice name, the number of nodes requested, and a lease length (in seconds). The slice name will be the login used to SSH into the nodes in your slice. The number of nodes requested is a hard requirement on the number nodes in your slice. The lease length is the amount of time (in seconds) the resources for your slice will be allocated. (Leases for a slice can be renewed as described in Section 5.2.) The example below creates a slice called `oceanstore` on 64 nodes with a lease length of two weeks, the current maximum lease length.

```
<?xml version="1.0" ?>
```

```
<slice>
     <name>oceanstore</name>
     <numnodes>64</numnodes>
     <leaselen>1209600</leaselen>
</slice>
```

To request that a slice span a certain set of nodes, use the optional `ips` tag as shown below. Given a list of n whitespace-delimited node IP addresses in a slice XML file and a request for m nodes (n >= m), the system will attempt to allocate m of the n nodes starting from the beginning of the list and working towards the end until m nodes have been found. (If m nodes cannot be allocated, an error will be reported.) The example below is for a 2-node slice where the two nodes should come from a set of three preferred IP addresses.

```
<?xml version="1.0" ?>
<slice>
     <name>oceanstore</name>
     <numnodes>2</numnodes>
     <leaselen>1209600</leaselen>
     <ips>141.213.4.201 216.165.109.81 216.165.109.82</ips>
</slice>
```

Once your slice XML file is created, you then create your slice using the `svm` command. Suppose your slice XML file is named `oceanstore.xml`. Your slice would then be created as shown below. Note that this command can take *several minutes* to complete for all the nodes in your slice. The main reason for this delay is the time it takes to create a new virtual machine on a node. In the future, we plan to add preallocation of virtual machines to speed this up.

```
# svm createslice oceanstore.xml
Success on 12.155.161.147
Success on 152.3.136.3
Success on 128.111.52.62
Success on 131.215.45.71
....
```

`svm createslice` does three things: (1) it obtains a set of tickets for a set of nodes for your slice through a broker (which in turn contacts an agent), (2) redeems the tickets for leases by contacting each node in the slice, and (3) uses the leases to create virtual machines for each node in the slice.

Two additional flags that can be passed to `svm` are the `-p` flag and the `-t` flag. The `-p` flag controls the amount of parallelism used by the service manager, specifically the number of threads used. The default value is 16. Increasing this value increases the amount of virtual memory used but can speed things up significantly when creating a slice on many nodes. The `-t` flag controls the timeout (in seconds) used to establish TCP connections to each of the nodes. If a node is completely unresponsive, using the `-t` flag can speed things up by returning a quick error, as opposed to waiting for TCP to time out. Here is an example that uses both flags. It specifies that up to 32 threads may be used for communicating with remote nodes and that each TCP connection should wait at most 15 seconds to be established:

```
# svm -p 32 -t 15 createslice oceanstore.xml
Success on 12.155.161.147
Success on 152.3.136.3
Success on 128.111.52.62
Success on 131.215.45.71
....
```

4.2. Deleting a Slice

Deleting an existing slice is also done using the `svm` command. Simply name the slice you wish to delete (i.e., the *name* of your slice, not the XML file used to create the slice) and `svm` will authenticate with each node and delete each sliver that is part of your slice. For example:

```
# svm deleteslice oceanstore
Success on 12.155.161.147
Success on 152.3.136.3
Success on 128.111.52.62
Success on 131.215.45.71
....
```

5. Managing Slices

5.1. Adding and Removing SSH keys

Remote access to slices is done using SSH. SSH keys must be in the *OpenSSH key format*. (A sample key in the OpenSSH key format can be found here (http://dslice.planet-lab.org/identity.pub). Note that it a *single line of text*.) Keys for authorized users for a slice are added using the `svm` command. Once an SSH key is added, the corresponding user should be able to SSH into any node in the slice using the slice's name as the login. For example:

```
# svm addkey oceanstore ~/.ssh/identity.pub
Success on 12.155.161.147
Success on 152.3.136.3
Success on 128.111.52.62
Success on 131.215.45.71
....

# ssh 12.155.161.147 -l oceanstore
```

SSH keys are also removed using the `svm` command.

```
# svm delkey oceanstore ~/.ssh/identity.pub
Success on 12.155.161.147
Success on 152.3.136.3
Success on 128.111.52.62
Success on 131.215.45.71
....
```

To remove *all* SSH keys for a slice, use the `svm` command as shown below.

```
# svm nukekeys oceanstore
Success on 12.155.161.147
Success on 152.3.136.3
Success on 128.111.52.62
Success on 131.215.45.71
....
```

To simplify the management of pushing SSH public keys out to slices, users can use the `pluserinfo` command to look up SSH public keys on `www.planet-lab.org`. SSH public keys are available for all users who have registered (https://www.planet-lab.org/db/web_accounts/add_web_account.php?PHPSESSID=a93214fd0dc1de9d866a0e39c9fe3622) at www.planet-lab.org (http://www.planet-lab.org) and have had their accounts enabled by their site's principle investigator. Like PlanetLab web accounts, `pluserinfo` identifies accounts for SSH keys by email address, as shown in the example below. (Note that the key below is printed with newlines in the example just for formatting. Your actual SSH public key should consist of a single line of text.) The `pluserinfo` command most often is used to look up a key, redirect it to a file, then pass the file to `svm` to push the key out to all nodes in a slice.

```
# pluserinfo --key llp@cs.princeton.edu
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA3Xv8nHAS+qRrvQ7DKxoH77ZKSAhaqbQtjAib6o1h6+9u
qhDqdMGiRivu1gKyYjj7DGPSoaqQ/LxMIbwFbNwRpkAM+rFRIYPjC0kWE2pBGAaf3TsIZZUMdGxPI5a/
qSpMrb8WFtab1CN+xzVj9jauX/pEjfPzcHZQTSH7PTA0M2k= llp@gentoo
```

`pluserinfo` can also be used to look up additional details on a particular user. For example:

```
# pluserinfo llp@cs.princeton.edu
EMAIL: llp@cs.princeton.edu
NAME: Larry Peterson
SITE_NAME: Princeton
TITLE: Professor
IS_ADM: f
IS_PI: t
ADDRESS: 35 Olden Street
CITY: Princeton
STATE: NJ
ZIP: 08544
COUNTRY: US
PHONE: 609-258-6077
URL:
SSH_PUBKEY: ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA3Xv8nHAS+qRrvQ7DKxoH77ZKSAhaqbQt
jAib6o1h6+9uqhDqdMGiRivu1gKyYjj7DGPSoaqQ/LxMIbwFbNwRpkAM+rFRIYPjC0kWE2pBGAaf3TsI
ZZUMdGxPI5a/qSpMrb8WFtab1CN+xzVj9jauX/pEjfPzcHZQTSH7PTA0M2k= llp@gentoo
```

Using www.planet-lab.org (http://www.planet-lab.org) as a lookup service for SSH keys is more a convenience than anything else. As shown in the initial example, any SSH key (e.g., `~/.ssh/identity.pub`) may be added to a slice.

5.2. Renewing Slices

Leases for slivers in a slice can be renewed using the `svm` command. All lease renewals are for same lease length as specified in the original slice XML file when the lease was created. Upon renewing a lease for a sliver on a given node, the old lease is replaced with the new lease. Hence, when renewing a lease at time t, the new lease will be valid until time t + leaselen. Using `svm`, leases for all slivers in a slice can be renewed with a single command as shown below.

```
# svm renewslice oceanstore
Success on 150.135.65.2
Success on 150.135.65.3
Success on 131.215.45.71
Success on 131.215.45.72
```

```
Success on 128.232.103.203
....
```

5.3. Node Lists and Expirations

The `svm` command can also be used to list the nodes in a particular slice. The list of nodes returned are nodes where virtual machines have been successfully created.

```
# svm nodelist oceanstore
150.135.65.2
150.135.65.3
131.215.45.71
131.215.45.72
128.232.103.203
...
```

The `svm` command can also be used to list the expiration times for the leases in a particular slice. For example:

```
# svm expirations oceanstore
Sliver for slice oceanstore expires 2003-03-05 23:11:01 UTC on 150.135.65.2
Sliver for slice oceanstore expires 2003-03-05 23:11:03 UTC on 150.135.65.3
Sliver for slice oceanstore expires 2003-03-05 23:11:02 UTC on 131.215.45.71
Sliver for slice oceanstore expires 2003-03-05 23:11:03 UTC on 131.215.45.72
...
```

5.4. Cleaning up your .planetlab Directory

As slices are created, deleted, and expire over time, information for certain slices in your `.planetlab` directory will become out of date. In some cases, the `svm` command will delete the relevant files for you. For example, if a slice is expicitly deleted, all files related to that slice will be automatically deleted. In other cases, however, files will persist. For example, if a slice's leases simply expire, information about that slice will still persist in your `.planetlab` directory.

To purge your `.planetlab` of old files, use the following command. This command essentially will look for slices that do not contain any valid (i.e., non-expired) tickets or leases. If a slice does not contain any valid tickets or leases, all files related to that slice will be deleted. Note that since this command does comparisons based on time, the machine where you run `svm` must maintain *reasonably accurate time*. If your clock is ahead by several days, for example, files will get blown away several days before they should have been.

```
# svm clean
```

6. Handling Partial Failures

As mentioned, `svm createslice` does three things:

- (1) Obtains a set of tickets for a set of nodes from a broker/agent.

- (2) Redeems the tickets for leases by contacting each node in the slice.
- (3) Uses the leases to create virtual machines for each node in the slice.

In each of these three steps, errors may occur due to nodes that are unreachable (e.g., a node has crashed, the network is down, etc.). In the current implementation, (1) involves communication with a single agent daemon, so this either succeeds or fails. For (2) and (3), however, the service manager, `svm`, must communicate with multiple nodes in a slice, some of which may be unreachable. An example of a failure in case (2) is shown below. In general, errors are reported with an "Error on www.xxx.yyy.zzz" message while success is reported with a "Success on www.xxx.yyy.zzz" message for each node. (The error message is printed with an extra newline below simply for formatting.)

```
# svm createslice oceanstore.xml
Error on 12.155.161.147: (<class socket.error at 0x824196c>,
<socket.error instance at 0x835bb4c>, <traceback object at 0x83595ec>)
Success on 169.229.51.250
Success on 169.229.51.252
Success on 169.229.51.251
```

Partials failures for (2) and (3) can be handled using the `svm` command. Please make sure to read PDN-02-005 (http://www.planet-lab.org/pdn/pdn02-005.pdf) before continuing here. If (2) fails, this means that tickets for one or more nodes were not redeemed for leases. If (3) fails, this means that leases for one or more nodes were not successful in creating virtual machines. For each slice, there is a directory that manages the transitions from steps (1) to (2) to (3) (i.e., from tickets to leases to virtual machines) for each node in the slice. For example, for the `oceanstore` slice:

```
# cd ~/.planetlab/slices/oceanstore
# ls -l
total 20
drwxr-xr-x    2 bnc      dusers       4096 Feb 19 15:08 leases
drwxr-xr-x    2 bnc      dusers       4096 Jan 23 15:02 slicekeypair
drwxr-xr-x    2 bnc      dusers       4096 Jan 24 12:41 sshkeys
drwxr-xr-x    2 bnc      dusers       4096 Jan 23 15:05 tickets
drwxr-xr-x    2 bnc      dusers       4096 Jan 23 15:05 vms
```

The `tickets` directory contains unused tickets for the slice (i.e., for nodes where (2) failed). The `leases` directory contains unused leases (i.e., for nodes where (3) failed). The `vms` directory contains currently active leases for nodes in a slice where virtual machines have been successfully created. Each directory contains a set of files named by IP address. In doing an `svm createslice`, tickets for nodes are placed in the `tickets` directory, tickets get deleted as leases are obtained and placed in the `leases` directory, and finally, leases are moved from the `leases` directory to the `vms` directory as VMs are created.

To handle partial failures for cases (2) and (3), you will need to use `svm` to issue lower-level commands. In fact, `svm createslice` really is identical to performing a (a) an `svm newtickets` operation, (b) an `svm newleases` operation, and (c) an `svm newvms` operation, each with the appropriate arguments. As mentioned, (1) always either succeeds or fails since it involves contacting a single node. To handle failures to create leases or create virtual machines (using previously issued leases), you must use these lower-level commands.

To handle failures in (2) (i.e., redeeming tickets for leases), use the `svm` command to explicitly retry this operation on a set of nodes. For example, suppose the service manager tried to redeem tickets for leases on nodes 150.135.65.3 and 131.215.45.71 for slice `oceanstore`. The ticket files for these two nodes would thus live in `~/.planetlab/slices/oceanstore/tickets`. To retry all failed ticket to lease conversions, do the following:

```
# svm newleases oceanstore ~/.planetlab/slices/oceanstore/tickets/*
```

To handle failures in (3) (i.e., using leases to create VMs), use the `svm` command to explicitly retry this operation on a set of nodes. For example, suppose the service manager tried to use leases to create VMs on nodes 150.135.65.3 and 131.215.45.71 for slice `oceanstore`. The lease files for these two nodes would thus live in `~/.planetlab/slices/oceanstore/leases`. To retry all failed lease to VMs conversions, do the following:

```
# svm newvms oceanstore ~/.planetlab/slices/oceanstore/leases/*
```

Handling failures in all other cases (e.g., adding a key, removing a key, renewing leases in a slice, etc.) is simply done by retrying the command until it succeeds on all nodes.

7. Feedback and Contributions

Feedback? Contributions? Problems getting things working? Send email to Brent Chun (bnc (at) intel-research.net). If you have general questions/feedback on dynamic slice creation in general, please send email to planetlab-slices (at) lists.sourceforge.net.