GENI: Slivers and Slices in a Diverse, Outdoor, Mobile Network Environment (DOME) Testbed

Milestone 1b Status and Documentation

(Revision 1.0, January 30, 2009)

Project Number 1599

University of Massachusetts, Amherst
Brian Levine, Mark Corner, Brian Lynn

Overview

This purpose of this document is to provide information on the DOME Xen virtualization environment. It details the status of DOME Milestone 1b: implement, integrate and document the Xen operating system and its virtualization features for the DOME mobile nodes ("bricks").

Status

Milestone 1b is complete. We have ported a Linux kernel with Xen support to our bricks. Additionally, we now have the ability to a initiate Xen guest domains on the bricks. The guest domains have access to the Atheros WiFi card, 3G cellular link, Ethernet subnet and GPS device (which is effort toward milestone 1c, due June 1, 2009). The instantiation of guest domains is currently manual; automation is a 1d deliverable (also June 1, 2009).

Implementation

The DOME control plane and devices under the control of DOME execute in domain 0 on a brick (Figure 1). GENI experiments will execute within a domU guest domain. The details on accessing devices and the scheduling of experiments are future deliverables (milestones 1c and 1d, respectively). This deliverable describes the Xen-based environment for hosting GENI experiments on the mobile nodes.

This document reflects the current state of DOME for GENI. It is subject to change.

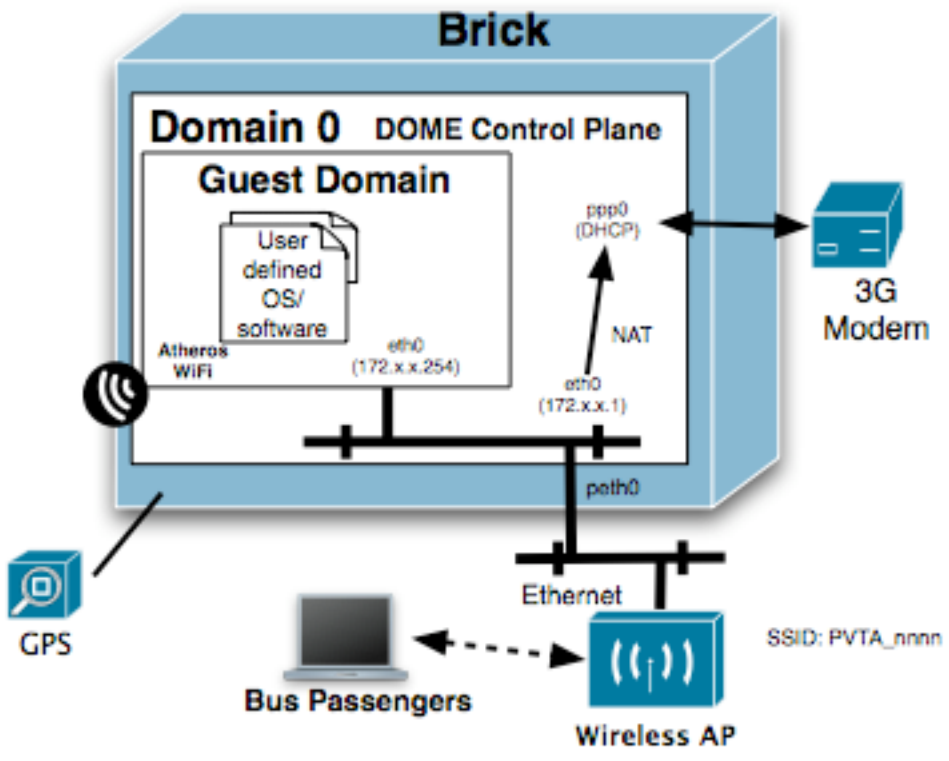

Figure 1: DOME Brick

Linux Kernel, Xen and Drivers

We use an Ubuntu 8.04 Server distribution, with a Linux 2.6.24-19 kernel. We install the Xen software and kernel source using the standard Ubuntu apt-get process. The current version of Xen is 3.2. For the Atheros PCI WiFi device, we have built and installed the madwifi-0.9.4 driver. By default, we use the identical kernels for both domain 0 and the guest domain (domU). We do support user-supplied kernels (see the GENI Guest Domains section).

To provide access to 3G links, we have built and installed the Sierra v1.2.10b driver. The GPS device uses the pl2303 module, a USB serial driver that is included with the Ubuntu distribution. Both the Sierra and pl2303 drivers need only execute in domain 0.

Networking

Figure 1 illustrates the hardware components attached to the brick, as well as the networking configuration. The guest domain will have exclusive access to the Atheros PCI WiFi card. This card can act as a WiFi client, accessing WiFi access points (APs) on other buses, as well as connecting to open access points such as the UMass and Town of Amherst meshes. The Atheros WiFi cards may also be configured to act as access points, permitting direct communication between Atheros devices on buses. We use the madwifi driver, and GENI experiments will have full access to the device. Customization of the driver will be permitted. This will be further documented in the future as part of the Milestone 1c deliverable.

The GENI guest domain's Ethernet interface (eth0) will be assigned a private (non-routable), static class B IP address, defined by RFC 1918 to be in the range of 172.16.0.0 through 172.31.255.255. We use a subnet mask of 255.255.255.0. Each brick has a unique, deterministic Ethernet subnet. For each brick with subnet 172.x.x.n, the IP addresses are allocated as follows:

172.x.x.1	Domain 0, gateway
172.x.x.254	GENI guest domain
172.x.x.255	Broadcast address
172.x.x.2-127	DHCP range, assigned to bus riders
172.x.x.128-245	Static range, e.g. guest domains on other buses
172.x.x.246-253	Reserved for future guest domains

Given its Ethernet subnet, a GENI experiment would be able to determine its IP address, the IP address of domain 0, and the bus number assigned to the bus the brick is installed on. (The mechanism for mapping a subnet to a bus number will be documented in the future.) This information will also be made available to the guest domain when the virtual machine (VM) is booted. (See section GENI Guest Domains).

The Ethernet subnet has an 802.11 access point connected to it, identifiable by its SSID. We have implemented a DHCP server in domain 0. A client of the 802.11 access point, such as a bus rider, can request via DHCP an IP address for joining the Ethernet subnet. Thus, bus riders can be on the same subnet as the GENI guest domain. GENI researchers will have the opportunity to offer experiments involving user opt-in of bus riders.

The GENI guest domains can use their Atheros WiFi cards to connect to 802.11 Access Points, in addition to other Atheros cards that are in AP mode. If connecting to an 802.11 Access Point, the guest domain must use IP. GENI guest domains can either use DHCP to obtain an IP address, or simply assign themselves an IP address within the static range.

Each brick also has a 3G modem attached to it. Our data has shown the cellular link to have approximately 90% availability. The 3G link is the DOME control plane; it is used to download experiments, upload logs files and provide status. All routable traffic sent on the Ethernet subnet is routed by domain 0 through the cellular link. This means that the GENI guest domains -- as well as bus riders -- have access to the Internet while the 3G link is operational. We NAT the Ethernet addresses, so connections must be established from the bus rather than into the bus. (The restriction is moot, since AT&T firewalls inbound connections on the 3G modems.)

The 900MHz XTend radios are not shown because they are a Year 2 deliverable.

GPS

A GPS daemon (gpsd) will execute in domain 0, and it will be accessible from the guest domain. A GPS device with a lock updates its coordinates approximately every second, and the raw sentences will be available in real-time to the guest domain. This will be further documented as part of Milestone 1c.

GENI Guest Domains

We have defined each GENI guest domain to consist of five separate disk partitions. We have implemented disk partitions using files formatted as disks, typically using the ext3 file system. We use separate files for each disk partition. Furthermore, we use the Xen blktap (tap:aio) access method for efficiency. The defined partitions are:

- A user partition. A GENI experiment is packaged as a disk partition that can be distributed to the nodes in the DOME mobile network. This user partition is downloaded from a server (Milestone 1d), and should contain all non-OS files necessary to execute one or more GENI experiments. This partition will be persistent. Once installed on a bus, the partition is expected to remain on the bus until the experiment is no longer scheduled (assuming no logical disk errors or space constraints). Therefore, files written by the guest domain during an instantiation of the experiment will still exist when the VM is re-instantiated. The user partition is the only partition that a GENI researcher is required to submit. This partition is mounted as /dome.
- A root partition. This partition contains the Linux kernel, default loadable modules, GNU/Linux executables and configuration files. The root partition contains the directories /bin, /boot, /dev, /etc, /home, /lib, /mnt, /root, /sbin, /tmp, /usr and /var, as well as the mount points for the pseudo file systems procfs and sysfs, and the mount points for the other partitions described here. We use Xen's pygrub bootloader so that kernels may be embedded within the root partition in the /boot directory. A user-supplied root partition is optional; there will be a default OS installed if it's not supplied. It is recommended that you do not supply a root partition unless you need to customize the OS in a way that precludes using a file installed in the user partition. Like the user partition, the root partition is persistent across instantiations of the experiment.
- Logging partition. The logging partition is a special partition mounted at /genilogs. The only files written to the partition should be logging files conforming to the DOME log file format (see below). Log files written to this partition are persistent, but each instantiation of the guest domain is given a newly created logging partition. Log files written to previous logging partitions are uploaded to a server by domain 0. The purpose of the logging partition is to allow log files written by an experiment to be uploaded even though the experiment is not executing.
- Information partition. This partition, mounted at /domeinfo, will be written by domain 0 each time the guest domain is booted. This directory will contain information about the bus, domain 0 and the current instance of the guest domain.

Examples of the information in this partition are IP addresses, subnet masks and bus numbers. This partition will be read-only.

- Swap partition. A swap partition is created for each instantiation of the VM.

The bricks are headless systems that run unattended. Therefore, each GENI guest domain must be able to execute without human intervention. The GENI guest domains will be configured to execute a script named /dome/geni.sh, i.e., geni.sh located in the parent directory of the user partition, when the guest domain boots. The GENI researcher submitting the experiment defines the contents of geni.sh, and its purpose is to bootstrap the experiment. This script will execute as root, and it will be executed by /etc/rc.local.

GENI researchers will have root access to their guest domains. We have several proposals for configuring root passwords, but the final method has not yet been decided upon. One approach would be to disable root logins by default, and to provide a script within the VM to set the password and enable logins. Another approach is to disable root logins and provide a script for the guest domain to add a user with sudo privileges. Of course, since the experiments begin life as root, they are free to do whatever they want within the VM. Furthermore, a user supplied root partition would need to ensure root had been properly configured.

Since the bricks run unattended, logins -- root or otherwise -- are typically not required. A VM can, however, configure a reverse-ssh tunnel to permit logging into the guest domain. The more important consideration for a researcher is to ensure bus riders and others do not obtain unauthorized access to guest domains.

Creating Partitions

It is a simple matter to generate a file for use as a user partition. You first need to create and format the file. For example, to create a 1GB partition:

```
# dd if=/dev/zero of=my_partition bs=1k seek=1024k count=1
# mke2fs -T ext3 -F my_partition
```

The first command says to create a file consisting of 1024K 1K blocks. We do not need to initially allocate all of the blocks, hence the count of 1. The second command formats the file using the ext3 file system. You can then mount your partition and put files on it.

```
# mkdir /mnt/mypart
# mount -o loop my_partition /mnt/mypart
```

Once the partition has been populated, unmount it:

```
# umount /mnt/mypart
```

We will place a TBD limit on the size of the user partition.

User-Supplied Root Partitions

If you wish to customize the kernel, you'll need to supply a customized root partition. Note that if you are only loading custom kernel modules, you may be able to install the modules in the user partition and load them from there. Since bandwidth and disk space are limited, we discourage the use of customized root partitions unless absolutely necessary.

We will make available copies of the default DOME root partition. User-supplied root partitions must be based on the default root partition. To make the custom root partition as small as possible, we have developed a tool, dome_trimfs, to reduce the size of custom root partitions.

To use the trmifs tool after creating your custom root partition, first mount both partitions. For example:

```
# mount -o loop dome_root.img /mnt/dome
# mount -o loop my_root.img /mnt/custom
```

Then execute dome_trimfs, specify the two mount directories:

```
# dome_trimfs -b /mnt/dome -u /mnt/custom -d -M -t -l -z
```

The -d option instructs dome_trimfs to remove all files from the user's custom root partition that also exist on the base partition. The -M option indicates that a manifest file should be written to the custom root partition. When DOME first attempts to start a guest domain using the custom root partition, it will read the manifest file and restore all deleted files to the custom partition. The -t and -l options tell dome_trimfs to get rid of temporary and log files, respectively, that might have been created on the custom root partition during testing. The -z option tells dome_trimfs to write zeros to all unused blocks. This is done to improve compression. The complete help text to dome_trimfs is provided at the end of this document.

As previously noted, we use Xen's pygrub to boot a Linux kernel found on the root partition, rather than using a kernel located in domain 0. Our default root partition places the kernel image in /boot and the configuration file in /boot/grub/menu.lst. A custom-built root partition with a modified kernel would need to ensure that pygrub is appropriately configured.

DOME Logging

GENI researchers are encouraged to upload as much logging data as possible to their servers during normal execution of their guest domains. However, we do provide a mechanism for asynchronously uploading logging data even if an experiment is not executing. We have defined a DOME logging file format. Any file found in a prior logging partition will be assumed to be a log file. Files will be deleted after they have been uploaded, or after they have expired (criteria TBD). We will also likely cap the number of log files and the amount of data that can be uploaded (TBD).

Invoking HTTP URLs embedded in the log file is the mechanism for uploading logging data. The log file format is very simple. At a minimum, the log file must contain a directive specifying the URL to be invoked. For example:

```
<uri> http://node.edu/log.php </uri>
```

Any other directives are interpreted as URL parameter names and their respective values. For example, a log file may have the following as its contents.

```
<uri> http://node.edu/log.php </uri><test>wifi</test><val>12 54</val>
```

The above would cause the DOME logger to invoke:

```
http://node.edu/log.php?test=wifi&val=12%2054
```

We have created a tool, dome_writegenilog, which can be used by the guest domain to generate log files. We have created another tool, dome_uploadgenilog, which the guest domain can run as a daemon to upload log files while the guest domain is active. The help text for the utilities can be found at the end of this document.

Source Code

Milestone 1b states that source code is to be made available. All UMass code developed for GENI uses the GENI unrestricted usage license. Though of little value until the mobile network is available to the GENI researchers, source is available upon request, as is a snapshot of a brick's disk image. However, DOME is still under development and will continue to evolve.

The OS source code includes:

- GNU/Linux Ubuntu 8.04 Server, http://www.ubuntu.com/getubuntu/download.
- Madwifi driver, http://sourceforge.net/project/showfiles.php?group_id=82936&package_id=85233&release_id=576121
- Sierra driver, http://www.sierrawireless.com/faq/ShowFAQ.aspx?ID=607

When we make the DOME mobile network available we will document the process for creating custom-built kernels.

Tools

The following is help text for some user-visible programs that we have developed.

Dome_trimfs

```
Overview:

This program is typically called to "trim" files from within a file
that represents a file system. The file would later be mounted as
a partition to a Xen domain. The assumption is that the "user" FS
was created from a "base" FS and that they have files in common.

Typical use of this program is when the user FS will be downloaded
to another system, and that system has a copy of the base FS. In
order to reduce the size of the FS being downloaded, files also in
the base FS are removed. The program outputs a list of deleted files.
Using the list, the files can be restored onto the user FS from the
base FS before starting the Xen domain.

This program contains other options for trimming the FS. If the user
FS is a root file system then /tmp and log files can be deleted.
There is also an option to ensure unused blocks contain zeros to
maximize compression of the FS file.

Additionally, the program supports a restore option, allowing files
to be copied from the base FS to the user FS.

Normally a user and base file system has been mounted with "mount
-o loop" before calling this program. The -u and -b options identify
the mount points. This program is typically run as root to ensure
access to all files in the mounted FS. Mounting a file requires
root privileges unless the information is in /etc/fstab.

It is also common to create a "manifest" in the user file system.
The manifest file is a specially formatted file written to the user
FS retaining information such as the names of the files that have
been deleted, and the name of the base FS was used when selecting
files to be deleted.

Syntax:

   dome_trimfs [-d -b dir][-B base][-c][-l][-M][-o file][-q][-t][-z]
               -u dir
   dome_trimfs -r -b dir [-f][-i file][-l][-m][-q][-z] -u dir
   dome_trimfs -? | -h

Where:

   -b Specifies the directory of the base FS for comparing to the
      user FS. Typically, this is the mount point of the base FS.
      This option is used with the -d or -r option.
   -B Causes the supplied base name to be written to the manifest
      or output file when used in conjunction with -d.
   -c Continue with the -d option even if neither -i nor -m
   -d Remove duplicate files from the user FS. Any files in
      the user FS that are also in the base FS will be deleted.
      A list of deleted files will be written to stdout.
   -f Force files to be overwritten on a restore. The default is
      not restore a file if it already exists in the user FS.
   -h Print this help text and exit.
   -i File containing a list of files to be restored.
   -l Remove known log files from /var/log of the user FS.
   -m Restore using names from the user FS manifest file. Use of
      this option conflicts with the -i option.
   -M Create a manifest file in the user directory. The manifest
      is identical to the file created with the -o option, but
      it becomes part of the user FS. The manifest file will be
      named ".DOME_TRIMFSMANIFEST" and be in the root directory of
      of the user file system.
   -o Output the names of deleted files to this file.
   -q Enable quiet mode suppressing warnings.
   -r Restore files to the user FS by copying them from he base
      file system.
   -t Remove all files in /tmp of the user FS.
   -u The directory where the user FS is mounted. This is the FS
      that is to be be trimmed.
   -z Write zeros to unused blocks in the user FS. This improves
      compression, and it is useful if decompressing to a sparse
      file.
   -? Print this help text and exit.

Copyright (c) 2008,2009 University of Massachusetts, Amherst
```

Dome_writegenilog

```
Syntax:
  dome_writegenilog [-b][-d dir][-p #][-q][-t][-v] [--] parameter ...
  dome_writegenilog -r [-b][-d dir][-p #][-q][-t][-v] [--] text
  dome_writegenilog -?

where:
  -b Add the bus number using a bus parameter.
  -d Define the logging directory. The default is: /genilogs.
  -p Purge policy, defined by a value between 0 and 3. See below
     for more information.
  -q Quiet mode: suppress error messages.
  -r Raw mode: simply write text to a log file.
  -t Add the local time (YYYY-MM-DD hh:mm:ss) using a localtime.
     parameter.
  -v Verbose mode.
  -- Terminates options; parameters or text follows.
  -? Print this help text and exit.

In default (not raw) mode the parameters after the command line options
must be in the form variable=value. One of the variables must be the
uri. For example:

  dome_writegenilog -q -- 'uri=http://node.com/log.php' 'code=5'

In raw mode, the above would be accomplished by:
  dome_writegenilog -q -r
    -- <uri>http://node.com/log.php</uri><code>5</code>'
```

The uri parameter defines the URI to be invoked by some logging program. The other parameters define parameters to the URI. For example, the above would result in an http get:

```
   http://node.com/log.php?code=5
```

The purge policy (-p) option is an internal directive for the utility responsible for invoking the URI. It tells the utility when it should remove the log file. Valid values are:

```
  0 delete after any attempt to invoke the URI
  1 delete only if there were no communication or other failures
  2 delete only if an HTTP 200 (OK) is received
  3 do not delete after uploading
```

The default value is 1.

Dome_uploadgenilog

Syntax:

```
  dome_uploadgenilog [-a] [-c #] [-k] [-m #] [-o file] [-p #] [-q]
                [-r #] [-v] [-w string] [-x string] [--] file ...

  dome_uploadgenilog -?
```

Where

```
  -a Append to the file defined by -o. Useful if there are multiple log
     files.
  -c Curl connect timeout. The default is 5.
  -k Keep files that have a problem, e.g. ill-formed logs. The default
     is to purge bad files.
  -m Curl maximum time issuing the command. The default is 15.
  -o File in which to write curl's output.
  -p Overrides a log file's purge priority. Permitted values are:
       0 unconditionally delete files
       1 delete if no communication or other failure
       2 delete only if an HTTP 200 (OK) is received
       3 don't remove any files
     Note that this is separate from the -k option since the -k option
     applies to invalid log files, while the -p option applies to log
     files that appear to be valid.
  -q Quiet mode, suppressing error and warning messages.
  -r Curl retry value. The default is 1.
  -v Verbose mode, enabling informative messages to stderr.
  -w String used with the curl -w option to write variables to the
     output file defined by the -o option.
  -x Extra curl options. The string is included in the curl command
     line.
  -- Terminates the command line options.
```

Overview

This program reads log files written by dome_writegenilog and then uses curl to invoke the URI with parameters, as defined by the log file.

Notes

If you want to use the curl -w/--write-out option you should specify the variables using this program's -w option rather than adding it via this program's -x option. This is because dome_uploadgenilog internally
makes use of the curl -w option to determine the HTTP return code.

Exit Code

The exit code will be 0 if the curl command succeeded. If the curl command returns a non-zero value then that value will be returned. Note that if multiple log files are specified then this program will return the result of the last curl command. If this program detects a user error, it will return a value of 99.