

IBM Blades ModelNet Environment

Cosmin Arad
cosmin@sics.se

June 1, 2007

1 Introduction

One of the goals of the Evergrow project in its final year is to bring together the results obtained in WP1, concerned with observing the Internet and measuring its properties, and WP3, concerned with developing scalable algorithms for Internet-based systems, by creating an evaluation testbed for large-scale dynamic distributed systems. The testbed allows participants to evaluate their systems developed as part of WP3 in emulated realistic settings available from WP1.

We have decided to use ModelNet [2, 1], the large-scale network emulator most widely used by the research community. The ModelNet network emulator is implemented as a FreeBSD¹ kernel module. ModelNet was developed in 2003, for FreeBSD 4.5, and hasn't been maintained since. The current stable FreeBSD version is 6.2, and the kernel code is SMP aware, i.e., more than one processor can run in the kernel at one time, which was not true for FreeBSD 4.5.

Each Evergrow participant site has 14 IBM blades with 2 hyperthreaded processors and 4GB of RAM memory each. We have decided to use some of these machines for the evaluation testbed. However, these machines have new devices for which there exist no FreeBSD 4.5 drivers. However, we have succeeded in installing FreeBSD 6.2 on the IBM blades.

ModelNet was written for FreeBSD 4.5 which had no multicore support so it relied on blocking interrupts for synchronization, namely on the `sp1x` API. This API has been deprecated in FreeBSD 6.2, all calls being essentially no-ops. Therefore we needed to port the ModelNet code to FreeBSD 6.2, which mainly consisted of replacing the old `sp1x` synchronization primitives with new kernel mutexes.

This document presents the process of setting up the ModelNet environment on the Evergrow IBM blades, so that it can be reproduced by other partners. The next section discusses the process of installing FreeBSD 6.2 on the blades, as it turned out to involve some configuration tricks. Then we discuss ModelNet installation followed by a description of the process of porting it from

¹A Linux alpha release is available but it does not contain all the features of the FreeBSD version and hasn't been as widely used as the FreeBSD version.

FreeBSD 4.5 to FreeBSD 6.2. We conclude with the performance we have obtained from the ported ModelNet by repeating the performance evaluation from the ModelNet paper [2].

2 FreeBSD 6.2 Installation

This is to overcome problems that arise because of (1) the FreeBSD boot CD does not boot from the blade cd drive, and (2) the blade network cards (Broadcom 5704) fail to negotiate link parameters with the blade switch and as a result, they report that the link is down (no carrier). First step is to download the FreeBSD boot diskette images from here:

```
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/6.2-RELEASE/floppies/boot.flp
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/6.2-RELEASE/floppies/kern1.flp
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/6.2-RELEASE/floppies/kern2.flp
ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/6.2-RELEASE/floppies/kern3.flp
```

Then, you need to create the floppies using these images. For instance, on a linux system you do that by running:

```
dd if=boot.flp of=/dev/fd0
```

Boot from the boot floppy. The boot loader will ask for the `kern*` floppies. When the bootloader is started you are presented with a menu. Select option **6**, to get a boot console. In the console type:

```
set hw.bge.fake_autoneg=1
boot
```

The `sysinstall` program is starting now. After language selection you are presented with a large menu. Select the `FixIt` option and then the `Shell` option. This will start a console on terminal 4. Switch to terminal 4 and type:

```
ifconfig bge0 up
ifconfig bge1 up
```

Now you can go on with the installation steps and you can install FreeBSD from FTP. The network will be configured by DHCP. For the package distribution, select `Kern-Developer`. This will help later with compiling the ModelNet code. When the installation is complete, the system will reboot. Do not forget to change the boot device priority so that Hard Disk 0 is before Network, otherwise you'll still boot PlanetLab. Again you have to get into the boot console and type:

```
set hw.bge.fake_autoneg=1
boot
```

Now FreeBSD starts for the first time. You can run:

```
pkg_add -r <my_favourite_editor>
pkg_add -r bash
```

and edit the `/boot/loader.conf` file to contain this line:

```
set hw.bge.fake_autoneg=1
```

Edit the `/etc/ssh/sshd_config` file to `PermitRootLogin yes` and make sure that the network interfaces are set to be autoconfigured by DHCP. Check that the `/etc/rc.conf` file contains the lines:

```
ifconfig_bge0="DHCP"
ifconfig_bge1="DHCP"
sshd_enable="YES"
gateway_enable="YES"
usbd_enable="YES"
```

From now on, you should be able to ssh to this blade, and reboot it without visiting the server room. To install and use ModelNet you need to do the following:

```
pkg_add -r perl
pkg_add -r sudo
pkg_add -r p5-XML-Simple linuxthreads libgnugetopt boost xerces-c2
cpan install XML::Simple
cpan install Graph
```

Accept to install all the dependencies when asked.

3 ModelNet Installation

ModelNet is comprised of the emulator proper and a set of helper scripts. The process of installing all the prerequisite software packages and all the necessary configuration steps is described in a howto document available here: <http://modelnet.ucsd.edu/howto.html>.

We need to install ModelNet from source and then we apply a patch that includes the porting changes, and compile the ModelNet kernel module, against the FreeBSD kernel header. Alternatively, we can distribute the ModelNet kernel module precompiled for FreeBSD 6.2.

4 Porting ModelNet from FreeBSD 4.5 to 6.2

There were two steps in porting ModelNet from FreeBSD 4.5 to 6.2. The first one was to make the ModelNet emulator kernel module compile against the 6.2 kernel headers. The second step was to replace the mechanism used for synchronization between the hopclock code, running from the timer interrupt

and updating packet queues of the emulated virtual hops (links) on the one hand, and the code running when new packets are received and put in their first hop queue on the other hand. Both access the model data, which is basically the paths and hops in the emulated virtual network, and their associated queued packets.

Making the ModelNet code compile against the 6.2 kernel headers involved changing how IP header length was accessed in IP address data structures, as these changed when FreeBSD switched to version 5. Also, ModelNet depends on the `ipfw` module which is the packet filtering/firewall module for FreeBSD. Since 2003 `ipfw` changed versions from 1 to 2 and the old ModelNet code strictly depended on version 1.

Replacing the `splx` based synchronization involved creating a kernel mutex on module initialization and using it to protect critical sections. The FreeBSD kernel mutex is a blocking mutex which spins for a short while if already locked, thus being optimized for cases when the critical section is short enough. After deploying the ported version we have conducted stress tests heavily loading the emulator with traffic to make sure there were no synchronization bugs.

With multiple emulator cores, the emulation load is divided among multiple physical machines by splitting the hops in the emulated virtual network among them. A packet traversing the virtual network might need to be handed over from one emulator core machine to another one, where the packet's next hop is emulated. ModelNet provides two enhancements: packet caching and aggregation. Packet caching means that the packets are cached on the first emulator machine and only their header is handed over to other emulator machines. When the packet has traversed all the virtual hops, the header returns to the first emulator machine, it is reunited with its payload and sent to the destination edge machine. Aggregation means that if there is more than one packet that needs to be handed over from one emulator machine to another in one time unit (jiffy), all of these will be aggregated into a single UDP packet, up to the MTU limit. Both packet caching and aggregation are enabled by default.

We have experienced some problems when using ModelNet in multiple emulator cores. We have found out that these problems disappear by disabling the aggregation feature. We haven't investigated the matter further.

5 Performance Evaluation

We tried to conduct the same performance evaluation as the one in the ModelNet paper [2] as much as possible. We do not have the exact same hardware and in our tests we used fewer machines than they did. Basically we used 6 IBM blades, 2 for emulators and 4 as edge machines. They used 4 emulators and 5 edge machines for the single core experiment and 20 edge machines for the multiple cores experiment. Due to our experiments being restricted to fewer machines, we believe that in some cases the performance was limited by one edge machine's bandwidth capacity.

We don't not know for sure what virtual topologies they have used. We

derived very simple virtual topologies based on the description in the paper. The results we have obtained are similar to theirs, and we tend to conclude that in both cases the network is a bottleneck as the tests fully use the 1Gbps physical bandwidth.

The performance measurements and graphs are available in an attached spreadsheet.

References

- [1] ModelNet. <http://modelnet.ucsd.edu>, 2002-2007.
- [2] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostic, Jeff Chase, and David Becker. Scalability and accuracy in a large-scale network emulator. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 271–284, New York, NY, USA, 2002. ACM Press.