

DeKiRu StarBED

— Experience SpringOS using Virtual Machine —

StarBED Project

Version 0.2

1 Introduction

It is quite normal that, before starting your experiments at StarBED, you would want to obtain some basic knowledge about StarBED and SpringOS by using the same environment.

First of all, many people do not even know what kind of things can be done with StarBED/SpringOS without trying it out.

Therefore, we will use the Virtual Machine as a quick and convenient environment in which SpringOS can run. However, some functions are not available in the Virtual Machine Environment, because it utilizes various mechanisms of infrastructure in StarBED/SpringOS. In this tutorial, the instructions do not cover the functions that are not available. We will describe the kinds of operations that are available in StarBED/SpringOS as much as possible.

2 Operational Procedures of StarBED

First, let's review the procedures that should be taken when conducting experiments using StarBED. The following is an overview for the experiment procedure, excluding administrative tasks, such as conclusion of Collaborative Research Agreement:

1. Reserve the facility
2. Setup the experiment management node for the SpringOS
3. Create disk image
4. (Start up ERM)
5. Scenario Description
6. Execute SpringOS

If you reserve the facility through the administrative staff, you can use the reserved nodes during the reserved time period. Please plan ahead and make sure that you reserve the facility.

Before actually starting the experiment, you have to install the OS and the various modules for SpringOS to the "node used to manage an experiment".

Second, decide what the OS and applications will be used in each “Experiment Node” and install them into one node (Also, install various modules for SpringOS).

Then, save it as a disk image so that you can use it as a template. You will copy this image to other nodes to create nodes that have the same settings.

Next, run the resource manager (ERM). This procedure was originally unnecessary; however, as of May, 2008, the ERM, which is supposed to be running smoothly in StarBED, is not functioning in a normal way. This is why you have to run ERM every time you start an experiment.

ERM is the abbreviation for Experiment Resource Manager. It maintains information on the physical resources that exist in StarBED. Judging from the number and types of the network interfaces required by your experiment node, assign the appropriate number of physical nodes and VLAN IDs as needed by the network.

When you are done up to this point, the next thing to do is to describe an experiment scenario. In the experiment scenario, describe the experiment topology, the disk image to use, and the commands that will be executed at each experiment node.

Execute kuroyuri master (called “master” hereinafter), which is an actor that drives the experiment by using this scenario as an input. The master will assign nodes, install the OS, and execute commands based on the scenario description. This way, one experiment will be finished.

Originally, users do not need to configure the ERM, since only one ERM needs to be running for the StarBED experiment facility. However, currently, this is not the case, since there is an operational problem. When this problem is solved, the executor should be able to execute the experiment by only preparing the environment where the master is running, creating a disk image and a scenario file, and then executing the master! You would be able to start the experiment right away after you have finished a simple test of the scenario using the Virtual Machine.

3 Basic Functions of SpringOS and Environment using Virtual Machine

First of all, I will introduce you to the basic functions of StarBED and SpringOS, and review the functions that must be set up in the Virtual Machine Environment.

Before we start the detailed explanation, please remember the expression “Experimental setup”. “Experimental setup” is a collective term we use for the experiment nodes and the other nodes that manage them. Meanwhile, we call the nodes on which the master to manage the experiment will be running, “management node for experimental setup”. This node needs to be ready before the experiment starts. The experiment starts by running the master in this node.

3.1 Basic Functions of SpringOS

SpringOS controls the nodes of StarBED, configures the settings required for the experiment, and executes commands in the nodes. The typical functions are as follows:

- Power on, shut down, and reboot of nodes
- Change the boot method of nodes
- Install the OS in each node

- Configure switches (VLAN)
- Scenario execution by executing commands at nodes
- Manage experiment resources

In StarBED, a management network is provided and is set up as a static network. By using this network (so long as there is no failure in the management network and no OS hang-up occurs), you can access the experiment nodes at any time.

3.1.1 Power on, Shut down, and Reboot of Nodes

SpringOS uses Wake on LAN (WoL) for booting nodes.

WoL is a booting method in which a magic packet is sent to the target node residing in the same segment.

Paradoxically, it can boot only the nodes connected to the same segment. Because the management network for the StarBED is divided into multiple segments, the WoL segment must be relayed to the other segments.

This task is accomplished by the WoL Agent.

The only thing WoL will do is turn the power on.

Therefore, shutdowns and reboots are accomplished by controlling an SNMP agent which is running in the experiment node.

For this purpose, `snmpmine` is provided as an application running in the node.

3.1.2 Changing the Boot Method of Nodes

It would take all day to change the BIOS settings and configure boot methods such as bootloader for every single node used in your experiment.

The PC nodes of StarBED always go through a network boot via PXE to change all these settings remotely at one time. The boot methods and partitions can be changed by switching the bootloader obtained during this network boot.

The first bootloader is forwarded via TFTP.

DMAN switches the bootloader for each node located under the `tftpboot` directory.

These bootloaders are provided as COIL.

Use DHCP to specify the information which is needed to obtain the IP address and bootloaders during the node boot.

3.1.3 Installing the OS in each node

As a method to install the OS in each node, we use either applying the disk image or booting via network.

The disk image method works like this: we install the OS and all applications that are necessary into one node. Then, make a binary format disk image from the applicable partition of this disk, and save it in an FTP server, from which this image will be distributed to the other nodes. The reason why we use this method is that most of the experiment nodes have the same configuration. For the settings that vary in different nodes, you can configure each node by running the commands as a part of a scenario. Use `pickup` to backup the disk image of the OS installed in the first machine. Use `wipeout` when you want to restore the disk image. Use `master` to install the OS as a part of your experiment. From now

on, I will call the group of elements (such as pickup, wipeout, and master) the ENCD (Experiment Node Configuration Driver). ENCD executes experiments by being the user's interface and by calling various functions of SpringOS. When reading and writing the disk image, the target node first boots using its exclusive diskless OS. Then, a hard disk operation program (hereinafter called NI) which is running on this node manipulates the disk image as instructed by ENCD. Since the NI does not know where the ENCD (which is supposed to manage it) is running, it uses a matching program (hereinafter called FNCP) which is managing both NI and ENCD. FNCP is continuously running in the facility. ENCD records in the FNCP the IP addresses of the nodes under its responsibility and its own IP address. When NI is started, the NI can connect to ENCD by asking the FNCP, "What is the IP address of the ENCP which is in charge of my IP address?"

When booting the nodes using the diskless method, you have to prepare a necessary image and boot it from there. However, the diskless system via PXE is not supported in all the OS's, since the configuration varies depending on the OS.

3.1.4 Configuring Switches (VLAN)

In StarBED, the physical wiring will not be changed. Instead, the L2 topology is generated virtually using VLAN and ATM. The change is accomplished by master issuing a topology change request to the switch management program (hereinafter called "SWMG"). The current SpringOS supports only VLAN.

3.1.5 Scenario Execution

Scenarios are executed by the coordination between master and kuroyuri slave (hereinafter called "slave"). The master runs on the management node for experimental setup and the slave runs on the experiment nodes. Both master and slave execute the scenario (command list) for the experiment. The coordination between nodes is established by messages exchanged between slave and master. The message is always exchanged through the master even when a slave is taking some action activated by another slave's action. There is another scenario executed in master. This scenario is basically used to control this type of message exchange.

To configure the network, you have to set up the IP address specifying the experiment side network interface at each experiment node. This is treated as a part of the scenario execution. Device names of each network interface vary depending on the OS. Since the device name may change every time it boots up, SpringOS will configure the IP address based on the MAC address of each interface. To map the MAC address and device name, use ifscan.

3.1.6 Managing Experiment Resources

To conduct the experiment, you have to assign as much resources as needed for the experiments, including the nodes and VLAN IDs. For this purpose, ERM manages nodes and VLAN IDs, and assigns resources to satisfy the demands of the master. The ERM also keeps track of the assignment status so that it does not assign a resource to multiple experiments. When programs such as master are requesting node information, it issues queries to the ERM.

3.2 Functions that can be realized in Virtual Machine Environment

In this section, we will configure nodes that manage your experiment. These are the services that are provided as part of the StarBED facility in the actual environment. However, you have to set these up,

Table 1: Modules related to the SpringOS

Category	Module name	Purpose
Facility	FTPd	Save disk image
	TFTPd	Provide data such as bootloader during node booting
	DHCPd	Provide management-side IP address and information regarding network boot
	ERM	Assign resource management
	SWMG	Configure switch
	DMAN	Change bootloader of experiment nodes
	COIL	bootloader used to boot from each partition
	wolagent	Relay WoL packets
	FNCP	Notify applicable ENCD during NI start up
	NI	Disk read and write client that works on experiment nodes
Experimental setup management	kuroyuri master	Experiment executing actor
	pickup	Save disk image
	wipeout	Write disk image
Experiment node	sbpsh	CLI used for power supply management and booting method change
	kuroyuri slave	Execute scenario
	ifscan	Mapping of MAC address and NIC device names
	snmpmine	SNMP Agent for shut down and reboot

since they are not available in the virtual environment.

Currently, compatibility with the environment is confirmed using Parallels and VMware. Since VMware provides a wider range of functions than Parallels, it is unusually preferable.

First, we have to check the prerequisites. In the real environment, you need modules in Table 1 shown below:

With VMware, you cannot execute Node Boot using WoL. Also, you cannot configure VLANs, since the setting interface (CLI, etc.) such as a general switch is not provided. Therefore, you don't need to start wolagent and SWMG related to this feature. However, the experiment-side network settings need to be finished before the experiment. In addition, PXE network boot is not supported with Parallels. In other words, you don't need DMAN and COIL, since booting will always be from HDD. Besides, when installing a disk image in SpringOS, the diskless boot is always used. Therefore, neither NI nor FNCP is necessary in Parallels.

4 Setting up the Management Node for the Facility

In the StarBED environment, application programs including ERM and FNCP are operated from the facility side. Let us configure the nodes so that we can provide the following functions:

- Configure Virtual Machine
- Install the OS
- Configure network

- Start up DHCPd
- Start up TFTPd
- Start up FTPd
- Obtain and compile SpringOS source
- Start up ERM
- Configure start up of NI
- Start up DMAN
- Start up FNCP

4.1 Configuring Virtual Machine

In this section, we will not go into details of configuration for individual Virtual Machines. However, note that a minimum of 1 network interface is necessary for each management node for the facility. SpringOS does not require very much memory if you are conducting a small-scale experiment. A total of 128 MB of memory should suffice.

However, please adjust the memory size depending on what other application software you want to use. More disk capacity will be required to conduct a large-scale experiment. Even so, configure the memory without worrying about it too much, since using a Virtual Machine allows you to change the configuration at a later time.

The disk capacity required for SpringOS itself is very small. Disk size will be determined by the OS you are going to install and how much logging you need.

4 GB will probably be sufficient for starter.

However, if you are going to use the disk as a file server to save OS disk image, you will need more capacity.

4.2 Installing the OS

Start up the Virtual Machine and install the OS by following the Virtual Machine's procedure.

Currently, SpringOS supports UNIX-like system OSs.

SpringOS's operation capabilities are confirmed in Linux and FreeBSD. There are not long enough track records about compatibility with other OSs. Moreover, there may be some other distributions and versions that cannot support compilation even in Linux or FreeBSD.

In this document, we used FreeBSD 7.0-Release.

The tasks after this section do not need to be done using root. However, use the users that have sudo and su, because some tasks may need root privileges.

In this document, we will use username starbed to work on the tasks using sudo.

4.3 Configuring Network

Pick the IP address to use for the management network. This time, let's use 10.211.55/24. Of course, you can use other IP addresses. For this node, we will use 10.211.55.5. Configure it in the rc.conf file so that the address will be set during boot.

The network interface on FreeBSD 7 will be recognized as ed? in parallels and as le? in VMware.

Edit the rc.conf file to describe the setting as follows:

The following is the case for VMware. Fill after ifconfig_ with the device name for each Virtual Machine.

```
ifconfig_le0="10.211.55.5 netmask 255.255.255.0"
```

Also, do not forget to specify the host name. We will use the name "master" here, but you can use anything else. Try to use a name that is easy to understand.

Add the following description to the rc.conf as well.

```
hostname = "master"
```

4.4 Starting up DHCPd

Next, we will configure DHCPd.

This is used to delivery addresses automatically to the nodes and hand over the options for PXE booting.

Therefore, if your environment is small scale and is not using network boot, you can specify static IP addresses.

I used ISC-dhcpd here. In FreeBSD, you should be able to easily install it from ports; even in Linux, you can easily install it using packages.

An example of the configuration file is as follows:

A complete set will be installed under /usr/local/ if you use ports in FreeBSD.

Then, create /usr/local/etc/dhcpd.conf as follows:

In this file, the network interface information that is connected to the management side network is described.

Check and write the network interface MAC Address of the management side (not the experiment side) for the experiment node.

```
default-lease-time 600;
max-lease-time 7200;

ddns-update-style none;

server-identifier 10.211.55.5;

option domain-name "mannot";
option ip-forwarding off;
option dhcp-server-identifier 10.211.55.5;
use-host-decl-names on;
```

```

group {
    next-server 10.211.55.5;
    option subnet-mask 255.255.255.0;
    option broadcast-address 10.211.55.255;

    subnet 10.211.55.0 netmask 255.255.255.0 {
    }

    host node001 {
        hardware ethernet 00:0c:29:84:ff:71;
        fixed-address 10.211.55.20;
        filename "node001.pxe";
    }
    host node002 {
        hardware ethernet 00:0c:29:84:ff:72;
        fixed-address 10.211.55.21;
        filename "node002.pxe";
    }
}

```

Then, configure it so that DHCPd will start when the node starts up. In FreeBSD, add the following two lines in `/etc/rc.conf`. Using `dhcpd_flags`, specify the device name of the network interface which provides the DHCP service as an argument of `dhcpd`.

```

dhcpd_enable="YES"
dhcpd_flags="le0"

```

Now all you need to do is to start up the DHCPd.

`/usr/local/etc/rc.d/isc-dhcpd` is a start up script. Copy this file under `/etc/rc.d` and execute it using the following command:

```
$ sudo /etc/rc.d/isc-dhcpd start
```

4.5 Starting up TFTPd and FTPd

In this section, we will configure TFTPd, which provides bootloader, etc. and FTPd, which is the interface to be used as the file server.

We will utilize `inetd`, since FreeBSD can provide TFTP/FTP service through it very easily.

You can apply the same method with Linux without problems by just adding a package.

In FreeBSD, uncomment the entries for `tftpd` and `ftpd` that are commented out in the `/etc/inetd.conf`, then restart `inetd` as follows:

```
$ sudo /etc/rc.d/inetd restart
```

4.6 Obtaining and Compiling SpringOS Source

Now, let's move on to the configuration of SpringOS.

We will use the installation destination directory `/usr/local/springos`.

The binary files will be saved under `/usr/local/springos/bin`, so we have to set the path to this directory.

Obtain the source code of SpringOS from <http://www.starbed.org/>.

In this document, we will use `SpringOS-Release-Ix-rc6.tar.gz`, `fncp-080508.tar.gz`, and `coil-20050902.tar.gz` under snapshot of the download page. Please use this FNCP version, although FNCP is also included in the `SpringOS-Release-Ix-rc6`, since a bug has been found in this version.

Now, decompress the SpringOS that you just downloaded and compile it.

You may save the source code anywhere you wish. In this document, we will create `/usr/local/springos/src` directory and store it under this directory.

```
$ cd /usr/local/springos/src
$ sudo tar zxf SpringOS-Release-Ix-rc6.tar.gz
$ cd SpringOS-Release-Ix-rc6
$ ls
dck13-080411.tar.gz      fncp-071116.tar.gz      pqerm-080411.tar.gz
dman-071116.tar.gz      ifsetup-071116.tar.gz   swmg-080411.tar.gz
erm-071116.tar.gz       mine-071116.tar.gz
wolagent-071116.tar.gz
```

These are the parts of the modules that SpringOS consists of.

The modules currently required in the management node for the facility are FNCP, DMAN, ERM, and COIL.

Create the directory `/usr/local/springos` before installation.

```
$ sudo mkdir /usr/local/springos
```

Decompress the files and compile them using `configure` and `make` commands.

The following shows the compile method of `erm`. You can compile them the same way by changing tarball names and decompressed directory names.

```
$ tar zxf erm-071116.tar.gz
$ cd erm-071116/src
$ ./configure
$ make
```

Then, create a symbolic link to the binaries of `erm`, `dman`, and `fncp` from `/usr/local/springos/bin/`.

Note that you do not need to compile `wolagent`, since it is a perl script (`wolagent.pl`).

As described above, most Virtual Machines seem not to be started via WoL. However, it may be useful to start up `wolagent` if you are going to control the actual nodes by creating management nodes in the Virtual Machine.

Decompress the new FNCP using the individual tar program and then compile and install it using the above-mentioned method.

`nasm` is required to invoke "make COIL". Please install it separately.

If you invoke "make COIL", binary files from `p1.bin` to `p6.bin` will be created.

Copy them under `/tftpboot`, giving them names from `boot_p1.bin` to `boot_p6.bin` respectively.

4.7 Setting up erm

In this section, we will set up ERM, which is the database server.

Three configuration files are required for ERM.

There are sample files in the data directory located directly under the directory created by decompressing the ERM source.

The names of the sample files are in parentheses.

It is OK to change the file names to any name you like, since you will specify the files at the ERM option.

- Resource file (starbed-resources): This contains the resource information, such as node.
- Account file (starbed-project): SpringOS manages resources using user and project. Configurations related to these programs will be written in this file.
- Access Control File (starbed-acl): Specifies the resources that user/project can use.

In this example, let's create `/usr/local/springos/etc` directory and then create an erm subdirectory. Save the files under the erm subdirectory. Each file will be saved as follows:

- `/usr/local/springos/etc/erm/demo-resource`
- `/usr/local/springos/etc/erm/demo-account`
- `/usr/local/springos/etc/erm/demo-acl`

4.7.1 Description of Resource File

In the sample resource files, all the resources of StarBED are described. Using the same format as in this sample, prepare the resource files for the virtual environment.

The resource file sample that is used in this case is shown in 1.

In this file, the VLAN range and two nodes are described. Please consider this an extra tip, since we do not configure VLANs in the virtual environment. Since there is no attribute in the VLAN IDs for the moment, it is specified using only a range.

Each node needs a respective entry, since the nodes have different specifications.

Nodes are defined in a block starting in “node”.

In this example, two nodes called node001 and node002 are defined.

Name nodes using the naming convention “character string + numbers”. Use three-digit numbers. (Neither two-digit nor four-digit numbers are accepted.)

The most important entries in the virtual environment are the network interface description.

This description is written in two lines that start from “net” in each node definition.

The “manage” in second column on the first line indicates a management interface. The “experiment” in the second line indicates the experiment interface. In the StarBED environment, each node has at least two network interfaces. One of the two is static and connected to management network.

The reason for this is that even if the network used for the experiment is failing because of some problems, you can check the status and troubleshoot the node by connecting to it via this management network. The third column specifies the network interface type, such as FastEthernet and GigabitEthernet. The fourth column is used to specify MAC address of each interface. The fifth column is used to

```

vlan 800..899

node node001 [
  bootdisk,IDE
  helth,ICMP,SSH
  power,SNMP-NECMIB
  net,manage,FastEthernet,00:55:5c:52:46:d9,"mgsw1:1/1",10.211.55.20,"Linux"eth0"FreeBSD"ed0"
  net,experiment,FastEthernet,00:1c:42:2c:bb:82,"exsw1:1/1",,"Linux"eth1"FreeBSD"ed1"
]

node node002 [
  bootdisk,IDE
  helth,ICMP,SSH
  power,SNMP-NECMIB
  net,manage,FastEthernet,00:55:5c:52:46:d7,"mgsw1:1/2",10.211.55.21,"Linux"eth0"FreeBSD"ed0"
  net,experiment,FastEthernet,00:1c:42:2c:bb:88,"exsw1:1/2",,"Linux"eth1"FreeBSD"ed1"
]

```

Figure 1: Resource Description for erm

specify the connected switch and its port. This information is used to set up the switch. It is OK to leave this empty, since a switch currently does not exist. To use SWMG, you need to add an entry for the switch by using the name specified here.

The sixth column is used for the IP address of the interface. The management interface requires assignment of a static address. Use the same IP address you just specified in DHCPd.

You can ignore the other definitions for now.

4.7.2 Description of Account File

Current SpringOS versions specify the resources that a user can access by using an account file. Figure 2 shows an example of the contents.

```
starbed:starpass:starproj
```

Figure 2: User Description for erm

In this file, each line has three columns divided by ":". The first column refers to user name, the second column refers to password, and the third column refers to project name. You can reserve the facility using the user name. One user can execute multiple experiments by dividing them into different projects. One line represents one entry. When there are multiple users, write one entry per line for each user.

4.7.3 Description of Access Control File

Information for all resources that exist in the environment is included in the ERM. However, most of the time, it indicates just part of the resources that a certain user can use. Therefore, you can specify the resources that are available to each user in this file. In Figure 3, node1 to node2 and VLAN IDs 800 to 810 are assigned to the user “starbed”.

You can specify the nodes in different formats: node1-node2, node001-002, or by delimitating them with commas.

```
permitrange node starbed node1-2
permitrange vlan starbed 800-810
```

Figure 3: Access Control Description for erm

4.7.4 Starting up ERM

To start up ERM, enter the following command:

```
$ /usr/local/springos/bin/erm -D /usr/local/springos/etc/erm/demo-resource \\  
-U /usr/local/springos/etc/erm/demo-account -A /usr/local/springos/etc/erm/demo-acl
```

If you save logs in a file, troubleshooting will be easier when a trouble occurs.

Methods to keep logs vary depending on the shell type. In sh, for example, you can keep logs using the following command:

Create the directory called springos under /var/log and store the log in it.

```
$ sudo mkdir /var/log/springos
```

```
$ sudo -c 'sh /usr/local/springos/bin/erm -D /usr/local/springos/etc/erm/demo-resource \\  
-U /usr/local/springos/etc/erm/demo-account \\  
-A /usr/local/springos/etc/erm/demo-acl > /var/log/springos/erm.log 2>&1'
```

Once you make sure erm can start up, exit erm using the kill command. It is useful if you create an entry in /etc/rc.local that instructs erm to start during node start-up.

Edit rc.local using root privilege to add the following entries: If there is no rc.local file yet, create a new one.

```
/usr/local/springos/bin/erm -D /usr/local/springos/etc/erm/demo-resource \\  
-U /usr/local/springos/etc/erm/demo-account \\  
-A /usr/local/springos/etc/erm/demo-acl > /var/log/springos/erm.log 2>&1'
```

4.8 Configuring Start up of NI

As mentioned above, to duplicate nodes in SpringOS, you can save the OS and applications that are installed in an existing node as a binary format disk image, and then copy it to other nodes.

When you are saving and writing the disk image to the target node, boot the target node as a diskless system and then read and write the disk using NI which is running in the system.

This is why we need to configure the environment where NI can be running.

There are many methods to configure this environment. However, I will not explain each one of them in this document, since all of these methods should be covered in other documents.

The following conditions are required for the environment:

1. The target node boots using a diskless system using PXE
2. On the above node, ni is running
3. Boot up can be configured using DMAN

What you can configure using DMAN is changing the bootloader of each node, located under tftpboot.

For the diskless system which is being accomplished by using general TFTP and NFS on FreeBSD or NetBSD, you can change it by just changing the target file. However, Linux is not supported, since you also have to change other files on it.

In addition, if you are booting up the node using a diskless system, at least the management network's interface and the hard disk where data will be written need to be ready and available.

Please set up the kernel so that these devices will be available.

Also, make entries so that the IP address of the management network will be obtained via DHCP, and ni and snmpmine will be started during NI start up.

You can obtain NI from the snapshot page mentioned above. snmpmine should be included in the package with dck that you have already obtained.

In this example, for your reference the rc.local is shown when the node is running on FreeBSD.

However, this rc.local would be the same even with Linux or NetBSD. Please note that this is not the rc.local file which is used for the management node for the facility, but the rc.local for the OS which will be running in experiment node.

```
#!/bin/sh
```

```
/usr/local/springos/bin/snmpmine > /dev/null 2>&1 &  
/usr/local/springos/bin/ni http://10.211.55.5:1238/nodeconfig.txt
```

In the code shown above, the URI of the FNCP is handed over as an option of the ni program. The URI is needed, since the FNCP and NI are actually communicating with each other via HTTP.

Make sure that this node can communicate with the management node for experimental setup through the management network by using the ping command.

4.9 Starting up the other modules

Now start up dman, fncp, and wolagent, which you just installed in the above section.

Start up dman by specifying the tftboot location as an option.

```
$ /usr/local/springos/bin/dman -D /tftpboot > /var/log/springos/dman.log 2>&1
```

You can start up fncp and wolagent without any options.

```
$ /usr/local/springos/bin/fncp > /var/log/springos/fncp.log 2>&1
```

```
$ /usr/bin/perl /usr/local/springos/bin/wolagent > /var/log/springos/wolagent.log 2>&1
```

Make entries for each of them in the rc.local file. Replace the perl path in the example according to your specific environment.

```
/usr/local/springos/bin/dman -D /tftpboot > /var/log/springos/dman.log 2>&1 &  
/usr/local/springos/bin/fncp > /var/log/springos/fncp.log 2>&1 &  
/usr/bin/perl /usr/local/springos/bin/wolagent.pl > /var/log/springos/wolagent.log 2>&1 &
```

When you are finished up to this point, reboot the node and verify that the program groups written in rc.local start up correctly.

5 Setting up management node for experimental setup

In addition to the above-mentioned programs, users have to prepare the management node for experimental setup which drives the programs, such as master.

Normally, in general StarBED use, users have to prepare only the images of this node and the experiment nodes.

Here, we will run the programs which are prepared by the facility and the programs which should be prepared specially for the experimental setup, in one node simultaneously.

Configure the management node for experimental setup by using the following procedure:

1. Compile kuroyuri master
2. Set up sbpsh

sbpsh is the interface that receives instructions from the command line. The instructions include experiment node's boot up, shut down, reboot, and change of boot method.

Using sbpsh makes the various operations easier.

Install kuroyuri master and sbpsh to the management node for the facility that you have just configured in the above section.

Compile the dck that was included in the above-mentioned tarball like the other programs.

Considering the future, create symbolic links to master, sbpsh, pickup, and wipeout, which are required binary files.

For the time being, let's create the configuration file for sbpsh only.

Figure 4. shows an example of the sbpsh.rc file. The first line represents the IP address of the node in which ERM is running.

This time, we will specify it as 127.0.0.1, since the ERM is running on the node on which sbpsh will be executed. 1234 is port number. Using tftpdman, specify the node on which DMAN will be running. We basically use port No. 1236. So, you don't need to change it.

```
set rm 127.0.0.1 1234
set tftpdman 10.211.55.5 1236
set wolagent 10.211.55.5 5959 10.211.55.0/24

set user starbed
set project starproj
```

Figure 4: Sample of sbpsh.rc file

In addition, if you are running WoL Agent, specify the IP address and the range of IP addresses that you have assigned to the management network. Leave 5959, since it is the port number.

In the last two lines, define the user name and project name; namely, starbed and starproj.

Now, you should be able to use it. I will explain how to use it later.

Now we have completed setting up the management node for experimental setup. Next, let us configure experiment nodes.

6 Setting up Experiment Nodes

The experiment nodes are the nodes that you are going to actually use in your experiment.

There should be different requirements depending on the executor of the experiment.

For an experiment node, you need at least two network interfaces - one is for the management side, the other is for the experiment side. Set up the required number of interfaces according to your experiment.

Currently, SpringOS supports only Unix-like OSs. Please use the OS that you think would be compatible with SpringOS.

If the OS you thought that was compatible did not work, try to contact the StarBED development team; they might be able to help you. Of course, we welcome you to fix the problem and send us a patch.

6.1 Installing the OS

Install the OS that you want to use. In this case, we use FreeBSD 7.0-Release, which is the same as what is used in the management node for experimental setup.

The reason we use this OS is that it is easier to copy the disk image which you have used in the management node above, since we are using Virtual Machines.

Of course, you may use a different OS from the management node for experimental setup. This is closer to the reality of the actual situation.

6.2 Downloading SpringOS

Use the same SpringOS that was included in the tarball you have downloaded while you were configuring the management node for experimental setup.

Copy it from the management node.

6.3 Compiling SpringOS

The method for compiling the Spring OS is the same as when you did it for the management node for experimental setup.

The package that needs compilation is dck (slave that is included in it), mine, and ifsetup.

Invoke “make” and then create symbolic links from `usr/local/springos/bin` to each binary file (slave, snmpmine, ifscan).

Since the configure program is not provided for ifsetup, just invoke “make” without executing configure. Please note that the binary file you would use is not ifsetup but ifscan.

6.4 Configuring slave and snmpmine

kuroyuri slave and snmpmine are client programs that operate upon receiving requests from the server.

You can just start them up without any special configuration. Create entries in the `/etc/rc.local` that have instructions to kuroyuri slave and snmpmine to start up during the OS boot.

Do not forget to create the `/var/log/springos` in this node too.

```
#!/bin/sh
```

```
/usr/local/springos/bin/slave -t -d > /var/log/springos/slave.log 2>&1 &  
/usr/local/springos/bin/snmpmine &
```

6.5 Configuring network

First, configure the network interface on the management side. You don’t need to set this value if you have set up the addresses to be assigned using DHCP.

Here, use the address that you have written in the ERM configuration file. Create the following entries in the `rc.conf` in the same way as you did for the management node for experimental setup.

```
ifconfig_le0="10.211.55.20 netmask 255.255.255.0"
```

Now may be a good time to for you to specify the host name by writing it into `rc.conf`.

```
hostname="node001"
```

The name in this entry is applicable for the first node’s setting. Change it to the applicable name for each node accordingly. In this environment, we will prepare two experiment nodes.

The IP address and host of each node are shown in Table 2. The management node for experimental setup that we have already created is also included in this table.

7 Creating Disk Image

Now, let’s create a disk image by using `ni` and `pickup`. `pickup` can be running on the management node for experimental setup. I think you remember that we compiled it with the other programs, including `master`.

To use this program, you have to prepare the configuration file first. You can specify all configurations as an option argument. However, since there are settings that never change, and to prevent typographical errors, it is better to put them together and save them in a configuration file.

Table 2: Node Names and IP Addresses

Node Name	IP address
node1	10.211.55.20
node2	10.211.55.21
master	10.211.55.5

```
-u starbed
-p starpass
-j starproj
-r 127.0.0.1:1234
-k 127.0.0.1:1236
-f 127.0.0.1:1238
-s 10.211.55.5
-K FreeBSD/ni04b.fs:recover_system/kernel_recover
-P recover_system/pxeboot
-w 10.211.55.5:5959:10.211.55.5/24
```

-u,-p, and -j represent user information that I suppose you are already familiar with.

-r is used to specify the node on which the resource manager (ERM) will be running; -k is used to specify the node on which DMAN will be running; and -f is used to specify the node on which FNCP will be running.

-s specifies the node and port that are running ENCD (in this case, pickup).

In the other options, you can use values such as 127.0.0.1 and localhost. However, in the value which is specified with -s, the expressions such as localhost and 127.0.0.1 are not supported. (This is to make it accessible from other nodes, since ni will try to get access to it after registration to FNCP.)

In -K option, the first segment divided by ":" is the required disk image for diskless boot. The latter segment means kernel.

-P is a bootloader. -w is used to specify wolagent. Now, save the file as pickup.opt.

I think -K and -P are more difficult. However, you can just set up -P in the FreeBSD environment which is using TFTP and NFS.

However, just place any value, since pickup will be abnormally terminated if you do not setup -K.

Now, start up the pickup.

There is something you have to do before executing pickup. pickup instructs the NI to save the device name of the partition. Therefore, the node's NI requires the device name and partition number of the disk. Actually, since the device name and partition name are hard-coded in the actual pickup, you have to modify source code.

Please rewrite the two lines (in default, "/dev/rad0s%c") of node.c under dck source directory according to the OS environment where you are using.

The partition number you want to specify will replace "%c".

```
/usr/local/springos/bin/pickup -F /usr/local/springos/etc/pickup.opt\
-X ftp://starbed:starpass@10.211.55.5/ node001:1
```

With `-F`, specify the configuration file you just created above and with `-X`, specify the target location to store the image in it.

In this example, it will be saved to the management node for experimental setup via FTP.

At the end, the target partition number of the target node is specified. Now, you only need to restart the target node. If you are using an environment where WoL is available, the node will be automatically rebooted.

Wait for a while, then the disk image will be saved into the management node for experimental setup.

The file name will be as follows:

```
node001-ad0s1-200805291811.gz
```

In short, the format of the file name's elements indicates the following.

```
Node name-device name-date time.gz
```

It may be convenient for you to change it to a name that is easier for you to understand. When you want to perform a write test of the disk image or you want to write the test image, you can accomplish it using the wipeout command. Run the wipeout at the management node for experimental setup, just like pickup.

The command shown below will write the file you just saved into partition No. 1 of the node002.

You don't need to change the configuration file, since you can use the same name as that of pickup. So, this avoids a typographical error.

```
/usr/local/springos/bin/wipeout -F /usr/local/springos/etc/pickup.opt\  
-X ftp://starbed:starpass@10.211.55.5/node001-ad0s1-20080591811.gz node002:1
```

However, wipeout will not modify the partition table. Therefore, create the same partition table as that of the node you just used as a template. You can retrieve MBR using the dd command; it would be easier for you to just copy the disk image of the template node as it is.

Regarding this technique, you may feel, "after all, NI is not needed in the Virtual Machine Environment". But be patient and think of it as a good way to experience the real environment.

After adding and deleting some files, write the disk image that you have created in the above by using pickup. Then, check if these files are back in place.

8 Scenario Description

Since the preparation is finished, it is time to describe the scenario and execute the experiment. However, from here, it will be the same procedure as for the general users of the actual environment. I will leave the explanation to other documents.

Please note that configurations of the SWMG, which sets up switches, have been omitted from the Virtual Machine Environment. When reading these instructions, please keep it in mind, like "OK, only this part is a little bit different from the scenario description in the real environment".

9 Conclusion

You just finished creating a StarBED-like environment. Now, please test it out and conduct many different kinds of experiments. It's good idea to collect various tips before working on the StarBED. You may have other wishes and requests after these experiments. Please share them with us.

Moreover, you may have requests that something should be improved or done in a different way in the Virtual Machine Environment.

We believe that this Virtual Machine Environment as well as the StarBED environment can be utilized as a very effective tool; please let us know your opinion by all means. If possible, please give us a hand in development.

I wish you a wonderful experimental life.