

SpringOS

Version 1.5

Manual

StarBED Project
2011 Spring

This document is compiled at March 25, 2011.

SpringOS Version 1.5 Manual
by StarBED Project

Copyright ©2011 StarBED Project. All rights reserved.

Preface

Network products should be tested practically like other industrial products. SpringOS helps these test. It improves the time of development and deployment for your products.

Audience

This book expects three type audience like followings:

[Network Experimenters]

SpringOS is developed to you. You should read Part I and Part III

[Testbed Administrators]

Part II is written for you. You also read Part I to test that SpringOS runs well.

[Testbed Tool Developers]

You should read source and all of this document. Especially you can find fractions of SpringOS design in Part IV.

Acknowledgments

SpringOS has been supported up by its users. It could not get today's prosperity without their cooperation.

We would be thank Mr. Masayuki Sano. He was a maintainer of our facility since 2002 (beginning of the facility) until 2009. Especially, he maintained the facility over 512 PCs alone in 2006–2007. He retired in November of 2009.

StarBED Project has been supported by JAIST and NICT. We thank for their supports and understanding.

Contact Point

Questions and comments are welcome. Please send what happened and the version or file name which you use (e.g. `springos-v1.5beta-r2840.tar.gz`) by e-mail to following address:

`info@starbed.org`

Further information, see also our WWW site:

`http://www.starbed.org/`

License

Copyright (C) 2002–2011 StarBED Project All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ‘‘AS IS’’ AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Terminology

experiment: a human activity using network equipments to design and analysis for network software and/or hardware.

node:

- 1) a entity of network.
- 2) an experiment entity program executable. In most case, it is PC.

host: a network entity. It is equal to node, except host is program executable certainly.

scenario: a sequence of statements which means experiment procedures/steps.

WoL: Wake on LAN; a PC booting method in NIC’s BIOS.

VLAN: Virtual Local Area Network; is not Video LAN.

NIC: Network Interface Card;

IA: INTEL Architecture; i386 compatible CPU.

resource:

- 1) physical item: actor hosts and switch ports.
- 2) logical item: IP addresses and VLAN IDs.

StarBED:

- 1) The research project of Internet simulation/emulation in Japan Advanced Institute of Science and Technology(JAIST).
- 2) The facility for network experiment (testbed). It consists of 1070 PCs and 8 switches. Its building and most of facility are found by National Institute of information and Communications Technology (NICT) according to the fundamental design by StarBED project. See also *Hokuriku Research Center* .

National Institute of Information and Communications Technology (NICT): A national research organization of Japan government.

Japan Advanced Institute of Science and Technology (JAIST): A national university of Japan government.

Hokuriku Research Center: The official name of the building for Internet simulator facility of NICT.

Contents

Preface

i

I User's Manual

1	Introduction	I-1
1.1	Executive Summary	I-1
1.2	GOAL — Automation of Network Experiment	I-2
1.3	Multi-user Experiment Facility	I-2
1.4	Structures	I-2
1.5	Driving Styles	I-6
1.6	The Navigation of Node Configuration	I-7
1.7	Loader Switching	I-8
1.8	OS installation	I-8
1.9	A Life of Actor Host	I-8
1.10	User's Work Flow	I-9
2	Built Programs	I-11
2.1	Required and Recommended	I-11
2.2	Retrieve Files	I-15
2.3	Structure of Source Directories	I-16
2.4	Compile	I-16
3	Manual Operations	I-17
3.1	Node Up/Down by sbpsh	I-17
3.2	Disk Backup/Restore by pickup/wipeout	I-19
3.3	Network Setup by bswc.pl	I-21
3.4	File Uploading via Kiyomitsu	I-22
4	Automatic Running	I-25
4.1	Site-aware Definitions	I-25
4.2	Main-body of Experiment	I-26
4.3	sns	I-30
4.4	Phases Skipping of kuma	I-30
4.5	Hints of File Writing	I-32

II Administrator's Manual

5	Boot-Loader Setup	II-1
5.1	coil	II-1
6	Daemon Setup	II-3
6.1	dhcpd.conf	II-3
6.2	RDB: Resource Database File	II-4
6.3	UDB: User Database File	II-7
6.4	ACL: Access Control List File	II-7

6.5	RULE: Rule File	II-8
6.6	swmg.conf	II-11
6.7	pwmg.conf	II-12
6.8	Start Daemons	II-13
6.9	Check Service Ports	II-15
7	Diskimage Making for Disk Operations	II-19
7.1	ni	II-19
7.2	SNMPmine	II-19
7.3	Setup Diskless OS via PXE	II-19
7.4	Diskimage Customize for ni	II-21
	References	II-23
	Historical Issues	II-25

III Command Reference

ENCD	III-3
bswc	III-4
dman	III-6
erm	III-7
fncp	III-9
ifscan	III-10
kiyomitsu	III-11
kuma	III-12
kusa	III-13
mlog	III-14
ni	III-15
pickup	III-16
pqerm	III-19
pwmg	III-20
sbpsh	III-21
snmpmine	III-22
sns	III-23
swmg	III-24
wolagen	III-25

IV Protocol Manuals

8	Experiment Resource Reservation Protocol Version 0.6 (ERRP/0.6)	IV-1
8.1	Overview	IV-1
8.2	Terminology	IV-1
8.3	Connections	IV-1
8.4	Name Space	IV-2
8.5	Resource Category	IV-3
8.6	Property of Resource	IV-3
8.7	Node State	IV-5
8.8	Communication Syntax	IV-6
8.9	Commands	IV-7
8.10	Sample of Command Sequence	IV-13
8.11	Administrator	IV-13
8.12	Literals	IV-14
	Appendix	IV-15
9	Experiment Structure Query Protocol Version 0.5 (ESQP/0.5)	IV-17
9.1	Overview	IV-17

9.2	Commands	IV-17
9.3	Name of Entities	IV-21
9.4	See Also	IV-21
10	Switch Configuration Protocol Version 1.2 (SWCP/1.2)	IV-23
10.1	Overview	IV-23
10.2	Commands	IV-24
10.3	Identifier of Switch Ports	IV-27
10.4	Tagged VLAN Expression	IV-27
10.5	Example	IV-28
10.6	See Also	IV-28
10.7	Hisotory	IV-28
11	Power Configuration Protocol Version 0.1 (PWCP/0.1)	IV-29
11.1	Overview	IV-29
11.2	Command	IV-30
11.3	Example	IV-31
11.4	See Also	IV-32
	Appendix	IV-32
12	Directory Manipulation Protocol Version 0.4 (DMP/0.4)	IV-33
12.1	Overview	IV-33
12.2	Command	IV-33
12.3	Response	IV-35
12.4	Example	IV-35
12.5	See Also	IV-36
	Appendix	IV-36
13	Wake on LAN agent Protocol Version 1.0 (WOLAP/1.0)	IV-37
13.1	Overview	IV-37
13.2	Command	IV-37
13.3	Example	IV-38
13.4	See Also	IV-38
	Appendix	IV-38
14	HTTP Extentions for ENCD	IV-39
14.1	Overview	IV-39
14.2	X-NSS	IV-39

Part I

User's Manual

1

Introduction

SpringOS is a program toolkit for network experiments. It makes the experiment efficiency. This chapter describes SpringOS's concept and structure.

1.1 Executive Summary

This sections shows the outline of SpringOS by listing.

SpringOS Features

- Automatic driving of network experiment by script language (Chapter 4)
⇒ User can write experiment procedures/steps like a shell script
- Manual driving of network experiment by command line (Chapter 3)
- Resource management; search and exclusive lock of resources
- Automatic construction of experiment situation
- PC power control by several protocols
- Software installation for PCs
- Switch control in multi platform

SpringOS Merit

- Organization of experiment steps
- Reproduction of experiment
- Reducing miss-operation
- Reducing preparation of experiment
- Running of multi experiments in one facility

Indexes of Merit

- Human cost
- Financial cost
- Time
- The scale of experiment entity

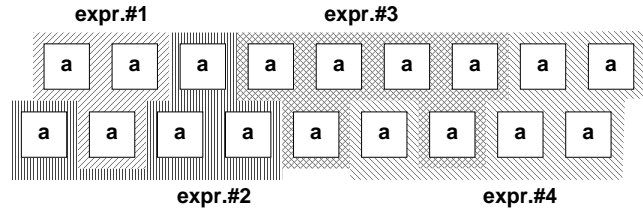


Figure 1.1: The test-bed is divided by multi experiments

1.2 GOAL — Automation of Network Experiment

SpringOS aims to reduce overheads of human operation in network experiments. The gathering of equipments (PC, switches and others) spends long time. The configuration of network devices makes tester confuse. The determination of IP addresses and VLAN ID is tired operation. As the scale of network-experiment increases, these overheads increases dramatically.

To solve them is our goal. Installation and execution of programs, and network configuration can be solved by program. Complex steps of experiment can be progress by programming language. Such approaches reduce human and financial cost, and overhead time. Improvement of cost and time efficiently increase the scale of experiment and the frequency of the repetition. Moreover, the automation brings improvement of experiment quality. Program language can replay the experiments repeatedly. It reduces miss-operations.

1.3 Multi-user Experiment Facility

SpringOS is developed to run over multi user testbed. In multi user testbed, many users do experiment(s) independently. Then, SpringOS have to support and identify individual users. Figure 1.1 depicts multi experiments divide a testbed. Furthermore, we introduce the concept of "project". User can performer multiple experiments. each experiment is project. So, SpringOS can handle multi-user and multi-project over one facility.

1.3.1 Resource Binding

To arbitrate the user's request for resource, we designed the resource manager. The implementation of that is experiment resource manager (ERM.) User and its program employ resources via ERM. ERM uses the protocol experiment resource reservation protocol (ERRP; Chapter 8).

1.4 Structures

1.4.1 Host Categories

Hosts are divided into 3 categories.

- actor host (many)
target programs for experiment run on them
- commander host (1 and more)
experiment driving program runs on the host
- facility side management host (1 and more)
resource management, boot management

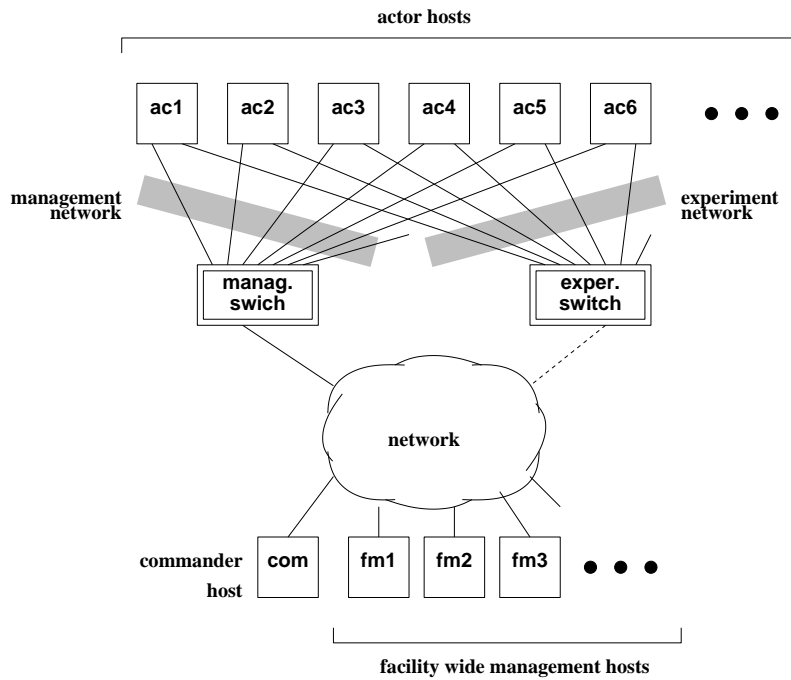


Figure 1.2: Conceptual Topology of Testbed

1.4.2 Program Categories

Programs are separated 2 region.

- facility wide management server programs
 - `dhcpcd` (not included SpringOS)
 - `tftpd` (not included SpringOS)
 - `ftpd` (not included SpringOS)
 - `fnccp`
 - `erm`
 - `dman`
 - `wolagent`
 - `swmg`
 - `pwmg`
 - `kiyomitsu`
- experiment programs
 - driver (commander)
 - * `kuma`
 - * `sbpsh`
 - * `pickup/wipeout`
 - * `bswc.pl`
 - be driven (actor)
 - * `kusa`
 - * `ifscan`

```

* ni
* snmpmine
* coil

```

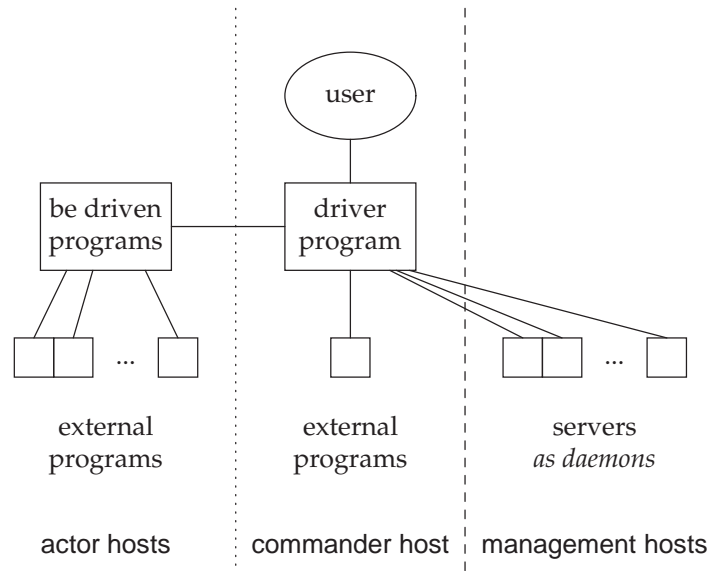


Figure 1.3: Program Rough Locations

Figure 1.3 shows the rough locations of them. Driver programs on commander host drive experiments with 1) calling of external programs on actor and/or commander host, 2) issuing of request to daemons upon management hosts and 3) dispatching the role for each actor hosts.

Following table shows location directory and role of these programs.

dir	role	programs
libexec	facility	dman erm pqerm fncp swmg pwmg wolagent kiyomitsu
libexec	actor	ni coil snmpmine
bin	commander	kuma sbpsh pickup wipeout sns bswc.pl
bin	actor	kusa ifscan

NOTE In early version of SpringOS, **kuma** and **kusa** are called **master** and **slave** respectively. Old names are straight forward. However very generic/simple name often cause another confuse because those name are appear in other programs. Then, we change those name.

1.4.3 Program Relationship

Figure 1.4 shows the relationship of them. By the number of link, you can recognize that **kuma** and **erm** are the key of SpringOS. **kuma** is a principle driver program. It have language processing feature. The program includes K language evaluation. **pickup** and **wipeout** are derivations of **kuma** for disk operation. **bswc.pl** controls network operations. **sbpsh** is a shell for network experiment. The program apples various operations (power, boot and others) according to user command.

In left column, **coil** is a boot loader. It is retrieved from **tftpd** when actor host wakes. **ni** is the disk operation program according to driver instructions from **kuma**, **pickup** and

wipeout. **kusa** is a slave program of **kuma**. The program evaluates K language script for each actor hosts. The program often calls **ifscan**, a network interface scanner. **snmpmine** is a terminator of actor hosts.

In right column, **kiyomitsu** is the reception program for file uploading. **dman** is a yet another file processing program. The program makes symbolic links for boot loader and removes them. **fncp** navigates **ni** to ENCD. **erm** is the resource manager. The program provides resource specification and manages ownership of them. **swmg** controls switches on the experiment facility. **wolagent** issues WoL magic packet for L2 networks. Filesystem sharing among **tftpd**, **dman** and **kiyomitsu** is required because **dman** and **kiyomitsu** aim the changing of **tftpd**'s files. Those file are boot loaders, kernel files and diskimages for OSs. Moreover, local disk and/or local filesystem is better for that sharing because using distributed filesystem for that sharing may cause race conditions.

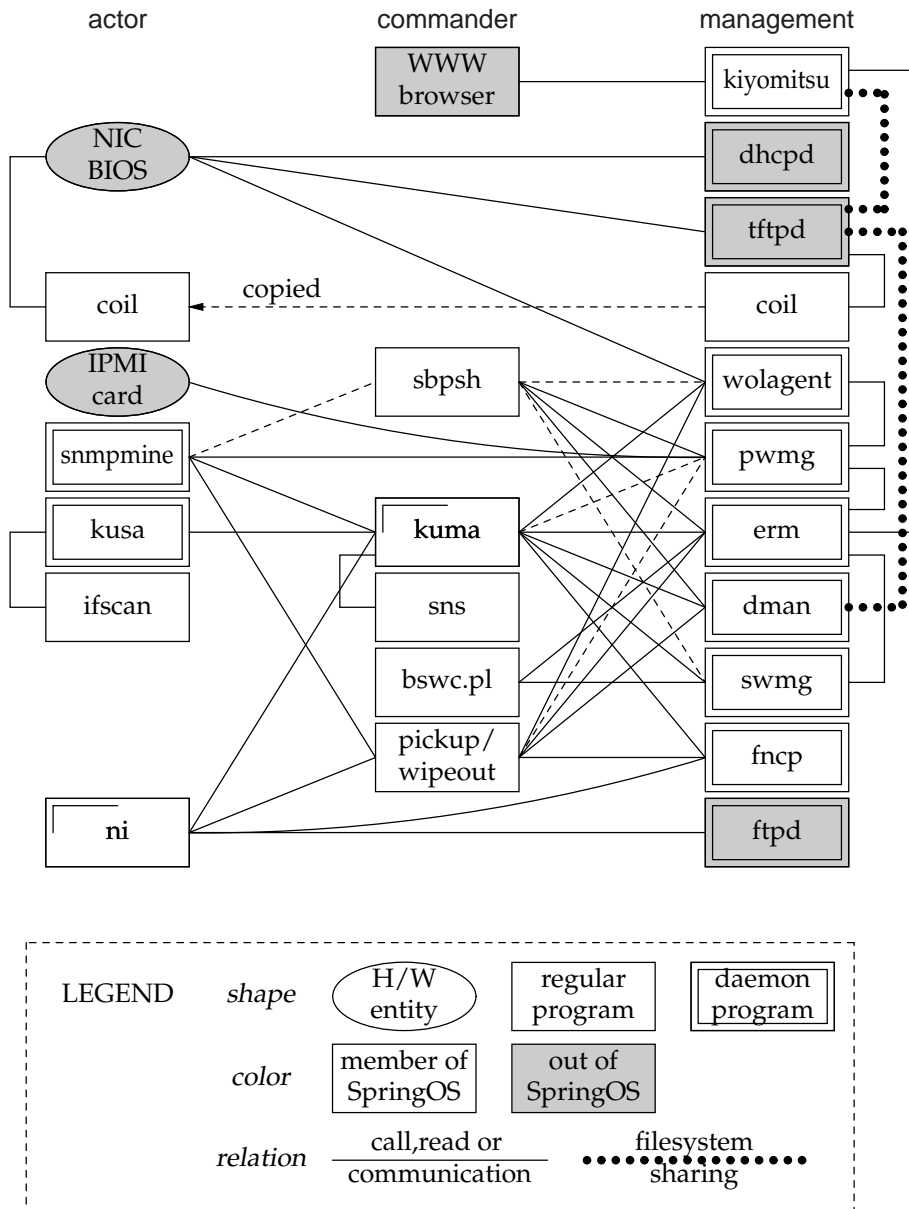


Figure 1.4: Program Relationship

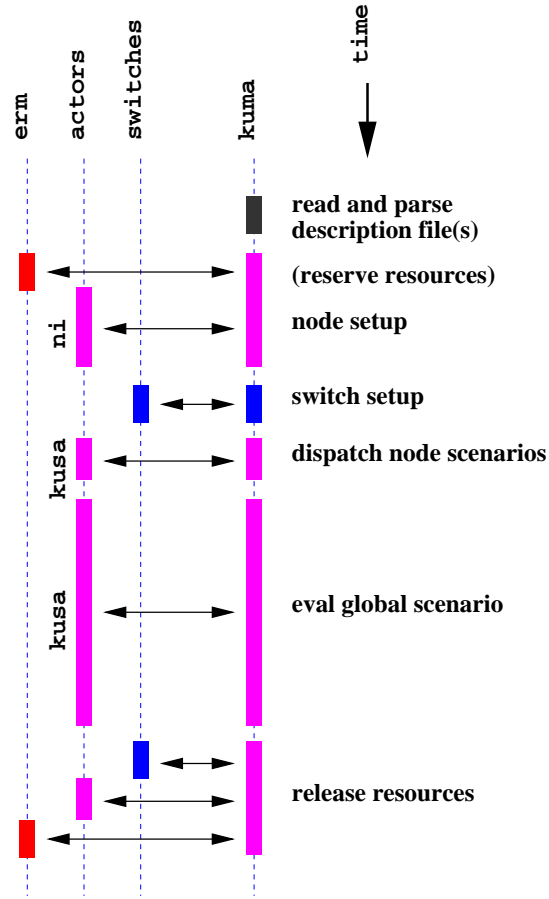


Figure 1.5: A Process flow of Automatic Driving

1.5 Driving Styles

There are 2 driving style of experiments using SpringOS. First is batch (auto), second is interactive (manual). Following two sub sections describe of them. See also Chapter 4 and Chapter 3.

1.5.1 Batch (Auto) Processing

Users can drive all steps of their experiments automatically. Program **kuma** is the key of that. Understanding of **kuma**'s behavior is important to use and debug SpringOS. Figure 1.5 depicts its process flow. In the beginning, **kuma** reads and parses experiment description file(s). Content of these files are shown later chapters.

After parsing file, the program enters communication of **erm** to decide actor host(s) as experiment node(s) and their spare(s). Spares are allocated to avoid insignificant errors. It will be discussed Section 4.2.1. According to result of **erm** communication, **kuma** tries to contact to actor hosts. When connection for **ni** is established, **kuma** issues OS installation command sequence. **kuma** waits until the number of nodes reaches to enough experiment running. Furthermore, timeout routine is ready to avoid forever waiting,

Switch(es) are configured after node setup via **swmg**. Because **kuma** knows the board and port number on switches which the host connected through **erm**'s database, **swmg** can configure VLANs on switch(es). VLAN ID(number)s are allocated from **erm**.

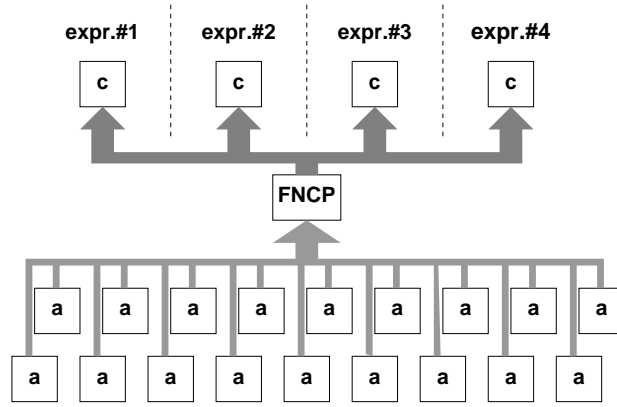


Figure 1.6: FNCP navigates actor hosts to commander host

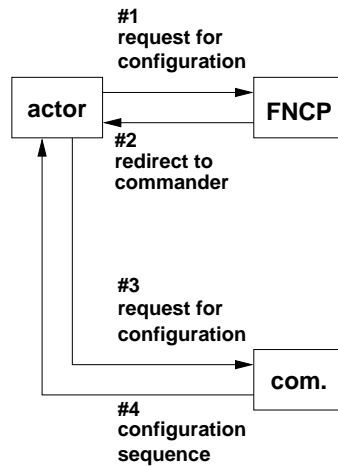


Figure 1.7: The navigation by HTTP redirection

When nodes and switches are ready, it is time to run scenarios. **kuma** sends node scenarios to **kusas** along node definitions. It is dispatching roles to actor hosts.

At last, **kuma** release resources by communications.

1.5.2 Interactive (Manual) Processing

SpringOS readies to progress each step. Some experiments suit and match this approach. Some user like that. User can control the power and boot loader of actor hosts by **sbpsh.pickup** makes the diskimage of actor host. Reversely, **wipeout** distributes a diskimage to actor host(s). **bswc.pl** is a client of SWCP. Using the program, user changes the topology of experiment. If you want to execute scenario, you should **kuma** with disabling of other features.

1.6 The Navigation of Node Configuration

Mention above, SpringOS have a purpose to support multi experiments in one test-bed, many experiment driver programs — **kuma** is one of them, are running on the test-bed. However, actor hosts don't know how many experiments are running and where are commander hosts of those in initially.

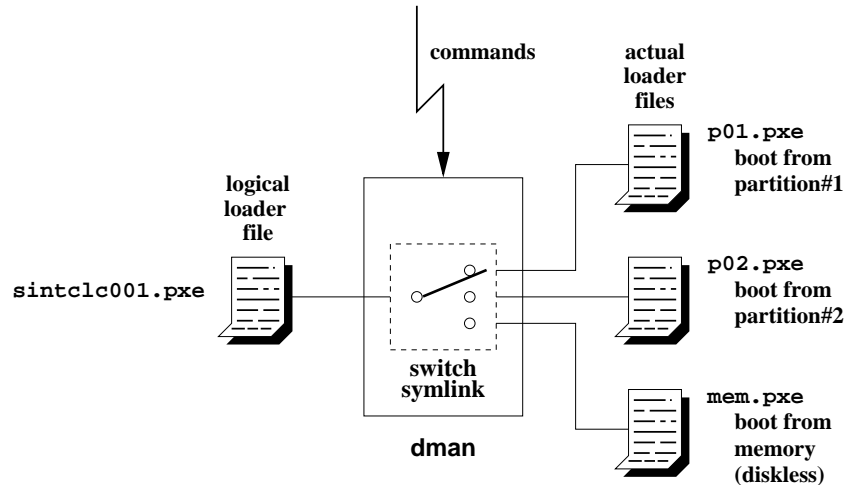


Figure 1.8: DMAN — symlink switcher

FNCP (facility node configuration pilot) have a role to solve the issue. **kuma** notifies a commander host for each actor host to FNCP. FNCP memorize those relations. Actor hosts can know commander host by asking to FNCP (Figure 1.6). The configuration of actor host is only to store FNCP's contact points. Other configuration (information about experiment, user, scenario, resources) are not required.

This navigation is a kind of routing. By HTTP redirection, FNCP notices a contact point to actor host. Figure 1.7 shows a sequence of its communication. So, a node configuration program has to communicate server at least twice.

1.7 Loader Switching

SpringOS changes OSs on actor hosts by symlink (symbolic link) on the TFTP server's contents directory. DMAN is an implementation of symlink switcher (Figure 1.8). By commands on its protocol DMAP from foreign program (Chapter 12), DMAN switches symlink for each actor host.

1.8 OS installation

Using program disk operation program **ni** with special OS, SpringOS backups and restores disk contents in actor hosts. Figure 1.9 shows this operations. Program **pickup** gives orders for **ni** to read and upload a diskimage of actor host's drive/partition to file server. The diskimage is a raw data like that of **dd** program in actor host. In file server it is a binary file on file system. Diskimage file is often compressed.

Conversely, **wipeout** gives orders that **ni** download the diskimage and write it into specified disk/partition of actor hosts. **wipeout** can such operation in parallel.

1.9 A Life of Actor Host

Figure 1.10 shows a typical life of actor host. Power-on by WoL magic packet. See Chapter 13 about protocol about wolagent. PXE booting by DHCP and TFTP. After **ni** waking, OS installation is started. **ni** reboot OS itself.

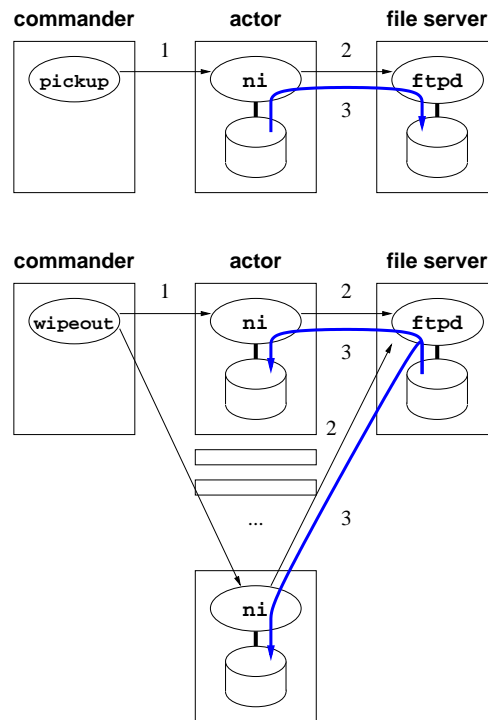


Figure 1.9: the operation flow of pickup/wipeout

Installed OS boots in a moment. **kusa** wakes and waits polling from **kuma**. To accept connection, **kuma** count this actor host as a candidate for experiment nodes. **kuma** sends a node scenario when the actor host are employed as node. Then, **kusa** evaluates the scenario to drive experiment.

1.10 User's Work Flow

- Design your experiment

You should think following items and more.

 - What nodes do you desire ?
 - How many actor-hosts to play these nodes ?
 - What network do you use ?
 - What program does be called and when it happen ?
- Make program and data set for actor hosts
 - (standard) setup new disk images for disk boot
 - * install OS
 - * install **kusa**, **ifscan**, and **snmpmine**
 - * extract disk images by **pickup**
 - * (optional) distribute these disk images into actor hosts by **wipeout**
 - (optional) setup new disk images for PXE boot
 - (optional) custom installed OS on actor hosts

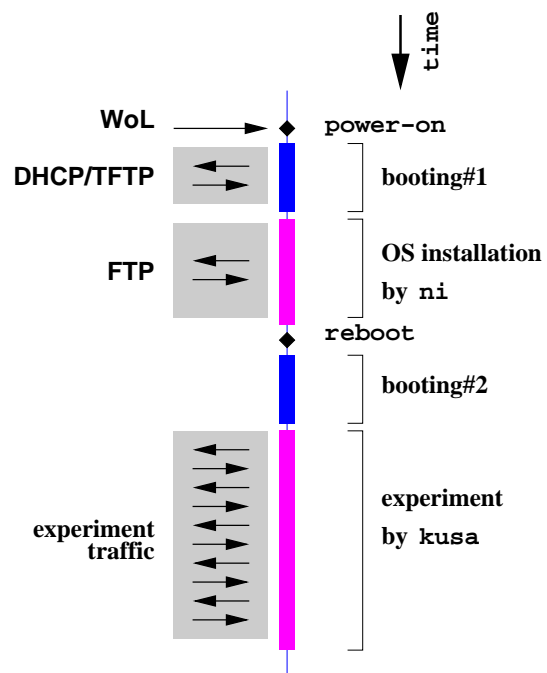


Figure 1.10: A Life of Actor Host

2

Built Programs

Since SpringOS is distributed in source file, user have to compile them before using. This chapter describes the requirements for compiling and running. Moreover, hints of its compiling appears.

2.1 Required and Recommended

This section describes requirements and recommendation of equipments (hosts(PCs), switches and others .) Table 2.1 shows results of our test.

2.1.1 PCs for Compile

To compile SpringOS, it has no requirement for hardware. Care only software.

Programs

Some OSs do not includes development tools in default install. You should install them.

- C compiler and standard library
SpringOS is designed for C99. Some files can compile under ANSI C. The compiler should includes mathematics library (**libm.a.**)
- make
- flex (or lex compatibles)
- bison (or yacc compatibles)
- ar
- ranlib
- awk
- perl
- (for **pgsql**) PostgreSQL
- (for **coil**) nasm

Libraries and Headers

- BSD sockets (build-in or **libsocket.a**)
SpringOS use inter process communication via BSD socket.

- POSIX thread (or its compatibles; build-in or **libpthread.a**)
kuma and **kusa** are implemented as multi-threaded via POSIX Thread API. So, POSIX thread library is required. You may find it as **libpthread.a** in your system(s). Most experiment does not required serious scheduling. You can employ some compatible libraries, maybe.
- zlib (**libz.a**)
 SpringOS use Zlib to compress data. You may find it as **libz.a** in your system(s).
- (optional; for **pqerm**) PostgreSQL's libraries and headers
- (optional) ncurses (or curses compatible; **libncurses.a**)
- (optional) readline (**libreadline.a**)

Some OSs separate headers from libraries. For example, zlib (**libz.a**) is installed in CentOS's host but its headers (**zlib.h** and others) are often not installed. You should install **zlib-devel** into the host like following.

```
% yum install zlib-devel
```

See also COLUMN on Page I-13.

2.1.2 PCs for Running

Actor Hosts

Hardware:

- IA base CPU
- two and more NICs
 Host is connected to two networks, management and experiment (Figure 1.2). Then, Host has NICs at least two.
 - WoL acceptable and PXE bootable NIC for management
 - * network acceptable IPMI (recommended)
 - one and more NIC for experiments
 We used following network interface card.
 - Intel Pro 100 (100Base-T)
 - Intel Pro 1000 (1000Base-T)
 - DEC related DE500 and its compatible (a.k.a "Tulip")
 There are many variation. We cannot promise you what are acceptable.
- half GB and more main memory
- several GB disk space
- compact chassis (recommended)
 To gather a lot of PCs, compact chassis is recommended. rack-mount/blade type PC is more better.

NOTE *Sorry, we did not test the threshold of memory and disk. A user reports that the virtual machine with 24MB memory cannot execute **kusa**.*

NOTE *A user reports that **ni** works well on Ubuntu 10.*

Software:

COLUMN	
In CentOS 5.5 we installed followings to compile SpringOS.	
for principal programs	for demo programs
<ul style="list-style-type: none"> • gcc • binutils • flex • bison • perl • zlib-devel • ncurses-devel • readline-devel 	<ul style="list-style-type: none"> • libX11-devel • libXt-devel • libXaw-devel • libICE-devel • libSM-devel • libXau-devel • libXdmcp-devel • libXmu-devel • libXpm-devel • xorg-x11-protocol-devel • xorg-x11-util-macros
for coil	
<ul style="list-style-type: none"> • nasm 	
for pqlrm	
<ul style="list-style-type: none"> • postgresql • postgresql-server • postgresql-devel 	
They are followings on the Debian. Please check other document to installation procedures.	
<ul style="list-style-type: none"> • build-essential • flex • bison • zlib1g-dev • ncurses-dev • libreadline-dev 	

- UNIX like OS

Our primary developing platform is CentOS 5.4 .

- C compiler and standard libraries

SpringOS is designed for C99. Some files can compile under ANSI C.

- POSIX thread library

kuma and **kusa** are implemented as multi-threaded via POSIX Thread API. So, POSIX thread library is required. You may find it as **libpthread.a** in your system(s). Most experiment does not required serious scheduling. You can employ some compatible libraries, maybe.

- Zlib

SpringOS use Zlib to compress data. You may find it as **libz.a** in your system(s).

Table 2.1: PC Architecture/OS Availability

Category		CentOS 5.4 i386	FreeBSD 4.2 i386	SunOS 5.9 SPARC	SunOS 5.10 i386
roles	actor	✓	✓		
	commander	✓	✓	✓	✓
	manage	✓		✓	✓
programs	fncp	✓		✓	✓
	erm	✓		✓	✓
	dman	✓		✓	✓
	wolagent	✓		✓	✓
	kuma/kusa	✓	✓	✓	✓
	ifscan	✓	✓	N/A	N/A
	ni		✓		

✓: Available, N/A: Not Available, Blank: unknown

Commander host

Hardware:

- Architecture/OS
No requirement, however UNIX like OS over IA base CPU is recommended.

Software:

- Architecture/OS
No requirement, however UNIX like OS over IA base CPU is recommended.
- POSIX thread library
- Zlib

NOTE You can use laptop/note PC to commander. However, experiment driving program have to run and connect to entities during experiment. When you take long experiment, you may want to leave test-bed. Commander host cannot disconnect from test-bed in the case. Consider this point when you employ laptop PC as commander host.

Management Hosts

Hardware:

- architecture/OS
No requirement, however UNIX like OS over IA base CPU is recommended. Because we never use other architecture/OSs (except **ftpd**, Solaris/SPARC and TurboLinux/i386 are used as **ftpd**) and this document expects them. You may use some architecture/OSs, probably. However this document can not cover them. please study your self.
- (**dhcpcd**, **wolagent**) connected management network of actor hosts in L2 (data-link) view.
DHCP(a kind of BOOTP) and WoL packets only flow on data-link layer. You should put several hosts when you use several hundreds actor hosts.

Table 2.2: Supported Network Switches

Maker	H/W Model	S/W	M.net	E.net
D-Link	DGS-3426, 3427 and 3450	*noname*	✓	-
Cisco	Catalyst 6509	IOS	-	✓
Brocade	BigIron MG8	IronWare	-	✓
(Foundry)	BigIron RX4, 16 and 32	IronWare	-	✓

- (**tftpd**, **ftpd**) large storage
diskimages often reach many tens GB. at least 40GB required, 160GB and more recommended.

Software:

- bootloader
SpringOS expects filenames of bootloader. Disk booting named **boot_p** and digit (c.f., boot_p1 and boot_p5 .) See Page II-25.
- (for **pqerm**) PostgreSQL

2.1.3 Switches

SpringOS supports Cisco, Brocade(Foundry) and D-Link switches (Table 2.2). To avoid disturbance, you should reduce the connection to other switches. Isolated switches are more better. However you have to keep connection(s) to control switches between the commander host and switches.

You have to check physical wire connection of switches to actor hosts. The result is used to make resource datafile of **erm** (Section 6.2) and configuration file of **swmg** (Section 6.6 .)

2.1.4 Network Topologies

1) management network

To management all PC, one NIC per PC connected this network. This network is managed with DHCP.

2) experiment network

All NIC of each PCs except management network have to join this network.

2.2 Retrieve Files

Access WWW page [**http://www.starbed.org/software/**](http://www.starbed.org/software/) or download page. Source files are archived as **springos-v1.5.tar.gz**. You may find files under following style:

- **springos-v1.5-rNNNN.tar.gz** (NNNN is revision number)
- **springos-v1.5beta-rNNNN.tar.gz** (NNNN is revision number)

2.3 Structure of Source Directories

[lib] the library for SpringOS

[prog] programs

COLUMN
<p>KNOWN BUGS</p> <ul style="list-style-type: none"> MD5 related code includes bug under 64bits (x86_64) CentOS. This code is used in IPMI operations on pwmng and sheepdog. You may meet that on other 64bit OSs. <p>In CentOS, you can avoid that by compiling with -m32. You should install related packages for libraries and headers.</p>

[**lib-wx**] (optional) the library for SpringOS with X

[**demo**] (optional) demo programs

[**self-test**] (optional) test tool and dummy data

2.4 Compile

According to typical way, you can compile SpringOS.

```
% tar xzf springos-v1.5.tar.gz
% cd springos-v1.5
% ./configure
% make
```

In some case, you would not compile programs in directories **lib-wx** and **demo**. However, these directories are optional. You can run SpringOS when you can compile programs in **lib** and **prog** directories.

```
% tar xzf springos-v1.5.tar.gz
% cd springos-v1.5
% ./configure
% cd lib
% make
% cd ../prog
% make
```

configure can accept typical options. For example, you can change the place to install by **--prefix** as **/opt**.

```
% ./configure --prefix=/opt
```

You can see options and arguments by run **configure** with **--help**.

```
% ./configure --help
```

NOTE *If you are administrator, read Part II to setup management programs and their related.*

3

Manual Operations

SpringOS consists of various features. User can call these feature by both manual and automatic. In this chapter, we show manual operations.

3.1 Node Up/Down by sbpsh

The program-**sbpsh** is a shell for the SpringOS environment. The program requires knowledge of environment parameters to you. You have to check those parameters to use the program. It is a good drill for to write the experiment description file which will appear in Chapter 4.

```
% ./sbpsh
command>
```

3.1.1 Parameters

You can get parameter's current state/value by **set** command.

```
command> set
trace off
prompt "command> "
user starbed
project starproj
rmpassword *none*
rm host localhost port 1234
swmg host localhost port 1240
pwmg host localhost port 1242
snmpport 161
snmpinterval 100
tftpdman undef
nikernel ni-kernel
nidiskimage ni-diskimage
pxeloder recover_system/pxeboot
wolinterval 100
command>
```

When you want to know commands, type **help**.

```
command> help
commands:
  help
  sample
  trace on|off
```

```

what <word>
info <category> <name>
poweron <hosts>
reboot <hosts>
poweroff <hosts>
setbootpxe <hosts> <path>
setdiskboot <hosts> <partition>
setniboot <hosts>
setpxelinux <hosts> <kern> <image>
setpxelinuxcfg <hosts> <config>
set
set rm <host> <port>
set swmg <host> <port>
set pwmg <host> <port>
set user <user>
set project <user>
quit

```

syntax:

```

hosts := hosts,host           ex. c7,c10-12
hosts := [a-b][0-9]+         ex. a23
      | [a-b][0-9]+--[0-9]    ex. c2-17
      | localhost

```

command>

Since **sbpsh** reads **.sbpshrc** before the command reception, you can set parameters automatically. Following is a sample of **.sbpshrc**.

```

#
# .sbpshrc - sample for sbpsh
#
#trace on
set rm 172.16.3.101 1234
set swmg 172.16.3.101 1240
set pwmg 172.16.3.101 1242
set tftpdman 172.16.3.101 1236
set wolagent 172.16.1.101 5959 172.16.1.0/23
set wolagent 172.16.3.101 5959 172.16.3.0/23

```

You can disable this feature by -n option.

```

% ./sbpsh -n
command>

```

3.1.2 Manipulation

Using **setdiskboot**, you can change bootloader of actor host as disk booting. The second arguments 2 means second partition. You should set name of user and project before first command. And the program prompt to enter password. You have to enter your password (e.g. 'ajapa'). When you do not know password, you must contact and ask this to the administrator of facility you used. See also Section 6.2 if you are administrator.

```

command> set user john
command> set project diskbench
command> setdiskboot sintcle001 2
password:

```

You can register name of user and project in **.sbpshrc**. To protect privacy, the program ignores registration of password. However, the program check env.var.-**KUROPASSWD**. If the variable is present, the program uses it as password.

NOTE You should apply `setdiskboot` before `poweron`, when you want to change OS.

Commands `poweron`, `poweroff` and `reboot` apply these procedures to actor hosts. The command for power operations requires running `pwmg`.

```
command> poweron sintcle001
```

3.2 Disk Backup/Restore by pickup/wipeout

Disk backup and restore programs are named `pickup` and `wipeout`. In this section, we will introduce these programs.

3.2.1 The Parameter File for pickup/wipeout

A traditional running example of `pickup` is following.

```
% ./pickup -u john -p ajapa -j diskbench -r 172.16.3.101:1234 \
-k 172.16.3.101:1236 -f 172.16.3.101:1238 -s 172.16.220.118 \
-K FreeBSD/ni02.fs:recover_system/kernel_recover \
-X ftp://install:install@172.16.210.9/ sintclc004:2
```

It requires many options and arguments (Table 3.1). Command line operations may cause mistakes. Then, it supports the parameter file. In that case, you can use a parameter file `pickup.opt` like following:

```
-r 172.16.3.101:1234
-k 172.16.3.101:1236
-f 172.16.3.101:1238
-s 172.16.220.118
-K FreeBSD/ni02.fs:recover_system/kernel_recover
-X ftp://install:install@172.16.210.9/
```

Next running is equal to above but it use `-F`. The option `-F` specify the parameter file. The argument `sintclc004:2` means *2nd partition of host 'sintclc004'*.

```
% ./pickup -F pickup.opt -u john -p ajapa -j diskbench \
sintclc004:2
```

3.2.2 Backup – pickup

Using `-u`, `-p` and `-j`, set parameters to access `erm`. These parameter should not be written into the parameter file. because they include privacy of user. You may omit `-x`, because it often includes some privacy, also.

```
% ./pickup -F pickup.opt -u john -p ajapa -j diskbench \
-X ftp://install:install@172.16.210.9/ sintcle001:2
```

This example expects that user `install` is accessible 172.16.210.9 with password `install`.

3.2.3 The Path of Target Partitions and Destination files

`pickup/wipeout` have the pattern table for device path. It is according to the FreeBSD manner, the program guesses device path by disk type and partition number. *D* and *P* are number of drive and partition. But *D* is always 0.

Table 3.1: Options of pickup

option	description
-u <i>user</i>	user name
-p <i>passwd</i>	password
-j <i>project</i>	project name
-F <i>filename</i>	configuration file
-r <i>host:port</i>	RM's info
-f <i>host:port</i>	FNCP's info
-s <i>host:port</i>	ENCD's info
-k <i>host:port</i>	DMAN's info for tftpd
-P <i>path</i>	PXE boot loader
-K <i>diskimage:kernel</i>	NI's info
-X <i>str</i>	(pickup) prefix of destination (wipeout) diskimage
-U <i>path</i>	device path
-V <i>disktype:num=dev,num=dev</i>	device name dictionary
-z	disable compression
-Q	disable autoreboot

type	device name
IDE(ATA)	/dev/radDsP
SCSI	/dev/rdaDsP

If you use devices which is out of above pattern, use -U like following:

```
% ./pickup -F pickup.opt -u john -p ajapa -j diskbench \
-X ftp://install:install@172.16.210.9/ -U /dev/ad0s4 \
sintclf001:4
```

To set pattern, use -V like following:

```
% ./pickup -F pickup.opt -u john -p ajapa -j diskbench \
-X ftp://install:install@172.16.210.9/ \
-V IDE:1=/dev/rda0s1,2=/dev/rda0s2 \
-V SCSI:1=/dev/rda0s1,2=/dev/rda0s2 \
sintclf001:4
```

The path of destination is determined by hostname, device path and date like following:

disk type	argument	date	filename
IDE(ATA)	sintcle001:2	9 Aug 2004 13:17	sintcle001-rad0s2-200408091317.gz
SCSI	sintclc001:2	5 Jul 2004 11:33	sintclc001-rda0s2-200407051133.gz

The date part is generate by local time.

3.2.4 Restore – wipeout

The options of **wipeout** is same as that of **pickup** except -x. In **wipeout**, -x is the name of diskimage to restore. Moreover, you can specify multiple targets by listing them as arguments.

```
% ./wipeout -F pickup.opt -u john -p ajapa -j diskbench \
-X ftp://install:install@172.16.210.9/sintcle001-rad0s2-200408091317.gz \
sintcle001:2 sintcle002:2 sintcle003:2
```

3.3 Network Setup by bswc.pl

bswc.pl is a tiny sample client for SWCP(Chapter 10). This program applies switch configuration in batch. The configuration is written in file. **bswc.pl** read them and apply the configuration to switches via **swmg**. The program access **erm** to book resources (nodes and VLANs) also.

After you write configuration file, you can run the program like following.

```
% perl bswc.pl conf-file
```

Following list is a sample of configuration file. The line which starts by #(shape) is comment. To simplify, this sample split into 2 parts. First part describes situation. **rm** sets resource manager's IP address and port. **rmuser** and **rmproject** set user and project for resource manager. The password for that is set by **rmpasswd** or env.var.-**BSWCPASSWD**. **sm** sets switch manager's IP address and port.

Second part describes operations for network. **activate** activates switch ports. It is known as "no shutdown" in some switch command line interface. **deactivate** is inverse of **activate**. **joinvlan** makes that switch ports join a certain VLAN. **exit** terminates the program.

```
rm localhost 1234
rmuser starbed
rmpasswd foobar
rmproject starproj
sm localhost 1240
#
activate devpc2:1
activate devpc3:1
joinvlan 800 devpc2:1 devpc3:1
exit
```

Switch ports are specified by the combination of node name and network interface number with a zero origin. "devpc2:1" means second interface of node-"devpc2". The program accepts a range expression in nodename. "devpc2-3:1" is recognized "devpc2:1" and "devpc3:1". However, the program does not accept a comma-expression like "devpc2:1,devpc3:1".

VLAN is specified by number. In generally, it is includes 1-4095. However it is restrict by configuration of **erm** and switch ability. Some switches support it only 1-1023. Most case of switch configuration cannot allow to use VLAN#1 because most switch use VLAN#1 as default VLAN.

Experiments often requires a timing tuning. To support them, **bswc.pl** has a command-**sleep**. This waits during specified time. Following list makes connectivity with 3 phases.

- 1) unreachable within 30 seconds
- 2) reach within 30 seconds
- 3) unreachable

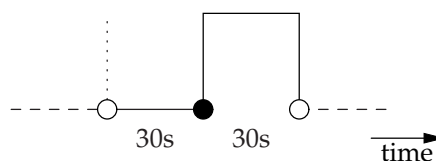




Figure 3.1: "Top" page of Kiyomitsu

```

deactivate devpc2:1
deactivate devpc3:1
sleep 30
#
activate devpc2:1
activate devpc3:1
joinvlan 800 devpc2:1 devpc3:1
sleep 30
#
deactivate devpc2:1
deactivate devpc3:1
exit

```

3.4 File Uploading via Kiyomitsu

Although user want to upload files on file servers (e.g., boot-loader, OS diskimage), access(login, file uploading and other procedures) to servers often are limited on many facility in security aspect. **kiyomitsu** is a way to upload them. The program runs with TCP 1244 port as HTTP server. User can upload files without user account on the host. You should access it using WWW browser (Firefox and Internet Explorer.) Figure 3.1 is the screen shot of Kiyomitsu's top page.

Click "Upload" button of the top page if you want to upload. Figure 3.2 is the screen shot of Kiyomitsu's upload page. Fill user, project and password for SpringOS and click "Browse..."-button. File selection window expose over WWW browser like Figure 3.3. Listed files are located on your local machine. Select file by click of filename and close the window by click of "Open"-button. Then, selected file will appear on the upload page. Actual uploading, file transmission starts by click of "Upload"-button.

Furthermore, user want to change symbolic links also. **kiyomitsu** can handle it. Figure 3.4 shows the screen shot of this operation. First, fill user, project and password. Second, fill hostname of target in this operation and Choice boot loader file. **kiyomitsu** has built-in 3 boot loader files for each partitions according to StarBED situation. "symlink"-button applies symlink the file to the host. If the host is symlinked other file, the symlink is overwritten new symlink.

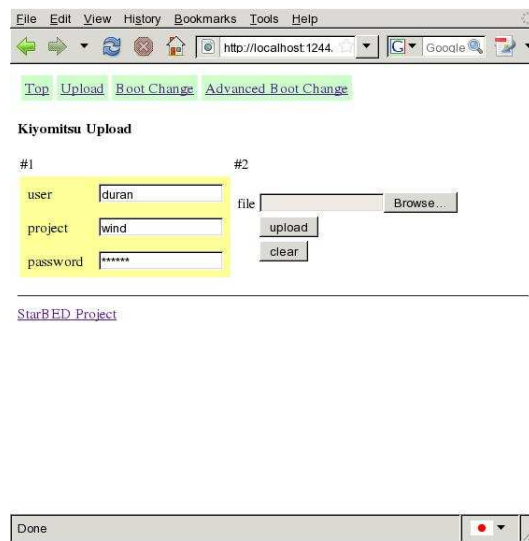


Figure 3.2: "Upload" page of Kiyomitsu

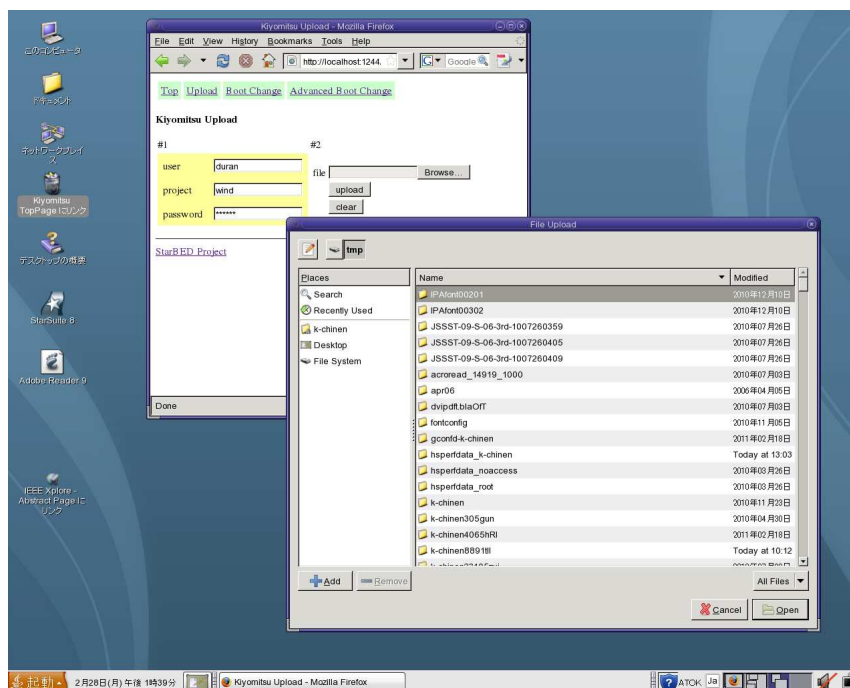


Figure 3.3: Selection of local file for uploading

The screenshot shows a web browser window with the address bar set to `http://localhost:1244`. The browser's menu bar includes File, Edit, View, History, Bookmarks, Tools, and Help. Below the address bar, there are navigation buttons: Top, Upload, Boot Change (highlighted in green), and Advanced Boot Change (highlighted in green). The main content area is titled "Kiyomitsu Boot Change". It features two sections: #1 and #2. Section #1 contains three input fields: "user" with the value "jduan", "project" with the value "jwind", and "password" with masked characters "*****". Section #2 contains a "hosts" field with the value "a033" and a "partiton" field with a list box containing three entries: "boot_p1.bin // Windows", "boot_p2.bin // Fedora", and "boot_p5.bin // Fedora". Below the list box are two buttons: "symlink" and "clear". At the bottom of the form, there is a link labeled "StarBED Project". A status bar at the very bottom of the browser window shows the word "Done" and a small red dot icon.

File Edit View History Bookmarks Tools Help

[Top](#) [Upload](#) [Boot Change](#) [Advanced Boot Change](#)

Kiyomitsu Boot Change

#1

user

project

password

#2

hosts

partiton

- boot_p1.bin // Windows
- boot_p2.bin // Fedora
- boot_p5.bin // Fedora

[StarBED Project](#)

Done

Figure 3.4: "Boot Change" page of Kiyomitsu

4

Automatic Running

Experiment driving program, **kuma** runs according to the experiment description file. The file consists of the site information and the definition of nodes, network and their behaviors. This chapter describe parts of experiment description file and how to run the program.

4.1 Site-aware Definitions

To drive experiment, **kuma** must know the IP address and port number of various daemons. Such information is site-aware. It is independent from the contents of experiment. Then, we recommend that you split experiment description into many files (e.g., site-aware definitions, main-body of experiment and others.) Such files are often named with suffix-".k". It stands the file of K language script.

NOTE *In early versions of SpringOS use suffix-".sc". Because early versions only support scenario but OS diskimage and network configuration. Since least version can handles other many things, we change the suffix. However evaluators do not care suffix. It is only the issue of sense, currently.*

Next script fraction is a sample of site-aware definitions.

```
#
# facility oriented information
#
# for StarBED 2003, 2004.
#
rmanager ipaddr "127.0.0.1" port "1234"
smanager ipaddr "127.0.0.1" port "1240"
pmanager ipaddr "127.0.0.1" port "1242"
wolagent ipaddr "172.16.1.101" port "5959" ipaddrrange "172.16.1.0/23"
wolagent ipaddr "172.16.3.101" port "5959" ipaddrrange "172.16.3.0/23"
fncp ipaddr "172.16.3.101"
tftpdman ipaddr "172.16.3.101"

nidiskimage "FreeBSD/ni02.fs"
nikernel "recover_system/kernel_recover"
pxeloder "recover_system/pxeboot-nohang"
setuptimeout total 3600 warm 5
```

See syntax manual for furthermore information. Here, we describe short description. "S" means one definition is acceptable. Do not define twice or more. "M" means multi definition are acceptable, if necessary.

item	#	description
rmanager	S	resource manager (erm)
smanager	S	switch manager (swmg)
pmanager	S	power manager (pwmg)
wolagent	M	wolagent
fncp	S	facility node configuration pilot (fncp)
tftpdman	S	directory manipulator (dman) for TFTP
nidiskimage	S	file path of diskimage for ni
nikernel	S	file path of kernel for ni
pxeloder	S	file path of PXE loader
setuptimeout	S	node setup timeout
swtype	M	switches

4.2 Main-body of Experiment

This section shows an example of experiment. Image an experiment with 3 pair of **netperf** / **netserver** over one network (Figure 4.1.) In this experiment, we assign program **netperf** and **netserver** to single node (actor host) to simplify. These nodes are divided into 2 roles as server and client. Then, we should define nodeclass and its instance for servers and clients.

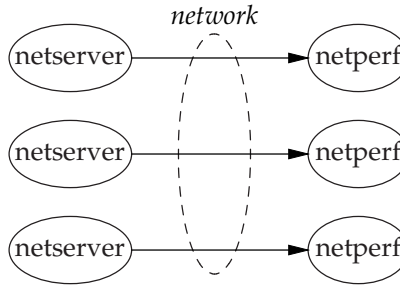


Figure 4.1: The Network Flow of Example Experiment

Figure 4.2 shows the relationship of nodes and network. Server and client nodes connect to network **ethnet** by network interface **netif[0]**. Statement **attach** can define such connection. The range of IP address of **ethnet** is 192.168.3.0/24.

Next topic is the scenario of nodes and global. As you know, server have to start before client start. One hand, client node waits server start by receiving of message. After receiving, client node starts **netperf**. Other hand, server node starts **netserver** at beginning of server node. If it received message "quit", terminates the program by **kill**. Commander side (global) scenario 1) it sends server's IP address to clients after enough time to start server. 2) it sleeps enough time to program termination. Figure 4.3 shows these sequence. This example shows well that the experiment is driven by events of message. Message driven can describe the dependency of processes. It accepts the rendezvous of flows over multiple nodes. The language can express serial and parallel flow.

Here, the time of first sleep in global scenario is 60 seconds. It includes network setup time of switch and each node. Because some switch takes several tens seconds to change port VLANs. At node side, the scenario retrieves the tuple of device name, MAC address and IP address by statement **netifit**. The statement fetches MAC address and device name by program **ifscan**. To verify reachability between client and server, **ping** is called in client node scenario. The time of second sleep in global scenario is 10 seconds. Figure 4.4 shows full sequence of this experiment.

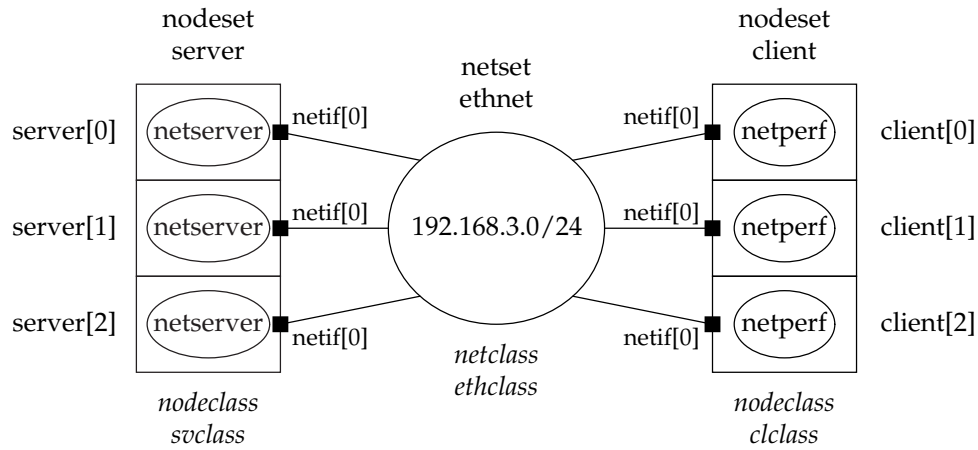


Figure 4.2: The Relationship of Example Experiment

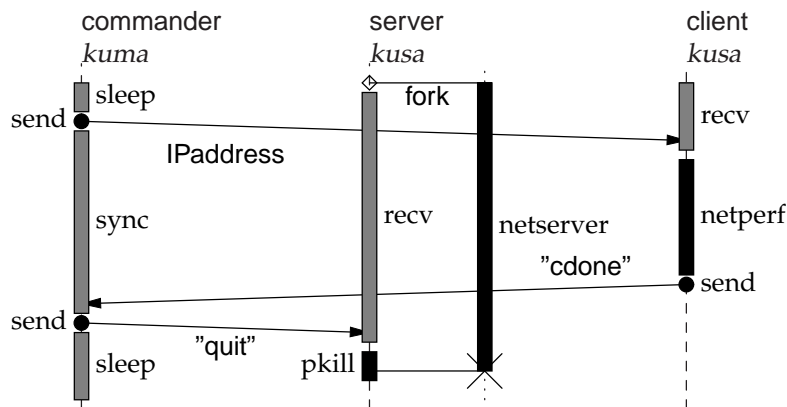


Figure 4.3: The Timing of Example Experiment (Essence)

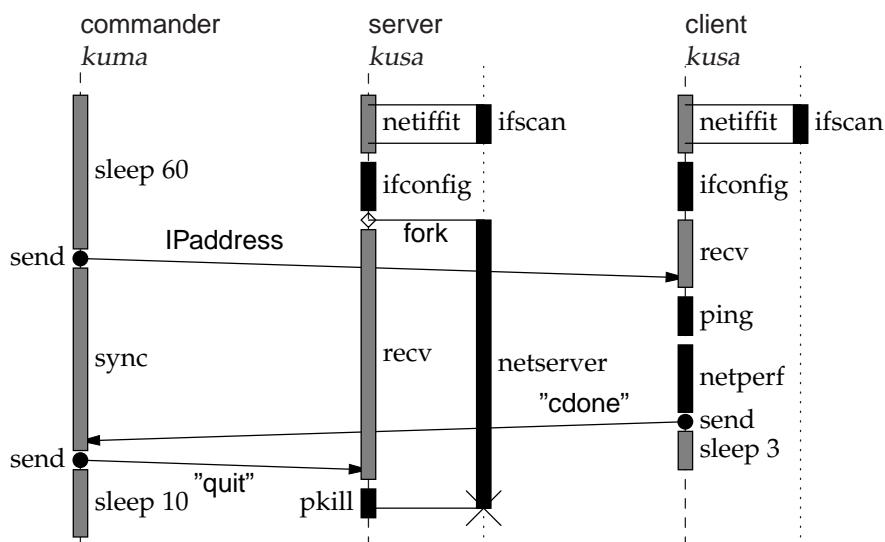


Figure 4.4: The Timing of Example Experiment

Following experiment description file shows all of them. You should replace 1) ENCD's IP address, 2) FTP server's IP address, username, password and filenames, before running.

```
#
# you should read pre.k before this file
#
# ./kuma pre.k this.k
#

user "starbed" "info@starbed.org"
project "starproj"
rbhfile "kuma.his"
encl ipaddr "172.16.220.118"
ipaddr range "192.168.3.0/24"

nodeclass svclass {
    method "HDD"
    disktype "IDE"
    partition 2
    ostype "FreeBSD"
    diskimage \
        "ftp://install:install@172.16.210.9/sintclb001-rad0s2-200407061104.gz"
    netif media fastethernet
    scenario {
        netifit "/tmp/ifscan"
        wakewait "/sbin/ifconfig" "/sbin/ifconfig" \
            self.netif[0].rname self.netif[0].ipaddr
        wakewait "/usr/bin/pkill" "pkill" "netserver"
        wake "/tmp/netserver" "/tmp/netserver"
        loop {
            recv x
            msgswitch x {
                "quit" {
                    wakewait "/usr/bin/pkill" "pkill" "netserver"
                    exit
                }
            }
        }
    }
}

nodeclass clclass {
    method "HDD"
    disktype "IDE"
    partition 2
    ostype "FreeBSD"
    diskimage \
        "ftp://install:install@172.16.210.9/sintclc001-rad0s2-200407051104.gz"
    netif media fastethernet
    scenario {
        netifit "/tmp/ifscan"
        wakewait "/sbin/ifconfig" "/sbin/ifconfig" \
            self.netif[0].rname self.netif[0].ipaddr

        sleep 3
        recv dst
        sleep 3
        wakewait "/bin/ping" "/bin/ping" "-c" "10" dst
        wakewait "/tmp/netperf" "/tmp/netperf" "-H" dst
        send "cdone"
        sleep 3
    }
}
```

```

    }
}

netclass ethclass {
    media fastethernet
}

nodeset client class clclass num 3
nodeset server class svclass num 3

netset ethnet class ethclass num 1
ethnet[0].ipaddr = "192.168.3.0/24"

attach server[0].netif[0] ethnet
attach client[0].netif[0] ethnet
attach server[1].netif[0] ethnet
attach client[1].netif[0] ethnet
attach server[2].netif[0] ethnet
attach client[2].netif[0] ethnet

scenario {
    sleep 60
    send client[0] haddr(server[0].netif[0].ipaddr)
    send client[1] haddr(server[1].netif[0].ipaddr)
    send client[2] haddr(server[2].netif[0].ipaddr)
    sync {
        msgmatch client[0] "cdone"
        msgmatch client[1] "cdone"
        msgmatch client[2] "cdone"
    }
    send server[0] "quit"
    send server[1] "quit"
    send server[2] "quit"
    sleep 10
    exit
}

```

4.2.1 The Number of Spares

In actually, the number of actor hosts in the experiment is greater than user specified. To avoid error, **kuma** allocate spare host(s). The number of actor hosts are solved two parameters. These parameter is the ratio for spare and the minimum number of spare. Below equation express this procedure.

$$S = \sum_{i=1}^m \text{MAX} \left(\left\lceil \frac{N_i \cdot r}{100} \right\rceil, N_i + k \right)$$

- S the total number of actor hosts
- m the number of nodesets
- N_i the number of defined node in i -th nodeset
- r the ratio for spare (default 105)
- k the minimum number of spare (default 1)

Therefore, eight actor hosts are allocated in last example. The number of spare actor hosts is two.

nodeset name	user specified	allocated	
		spare	all
server	3	1	4
client	3	1	4
total	6	2	8

Using **sparenoderatio** and **sparenodemini** in description file, you can change these parameters. When you set 100 and 0 to them, spare hosts are nothing. However, no spare means no error avoidance. be carefully.

```
sparenoderatio 100
sparenodemini 0
```

4.2.2 How to run

To apply facility definition file (here, I call that **pre.k**) and this listing (**netperf.k**) to **kuma** with password for **erm** and switches, the experiment will start.

```
% ./kuma -p rmpasswd pre.k netperf.k
```

A lot of debugging options is available. Use them if you worry the behavior of the program. The option **-d** enables debug messages in all modules. Furthermore, you can save messages by running of **tee**.

```
% ./kuma -t -d -p rmpasswd pre.k netperf.k |& tee m.log
```

Using option **-U**, you can make the masking of debug messages. Following options order to record all messages expects module **eval** and **engen**.

```
% ./kuma -t -d -U eval,engen -p rmpasswd pre.k netperf.k |& tee m.log
```

4.3 sns

sns sends an instruction to skip node setup to ENCD. Node setup, especially disk operation, often occupies the most of the experiment execution time. When you use node setup method **HDD**, ENCD starts to install OS diskimage into HDD. User can execute **kuma** without node setup in this case using **sns**. Figure 4.5 depicts where **sns** is applied in a process flow.

Some case user not required this step. For example, in middle of iteration loop with parameter sweep, user does not require to change diskimage.

Method **DIRECT** means that ENCD does not care OS diskimage and start the scenario execution immediately. **sns** have no effect in such case.

4.4 Phases Skipping of kuma

kuma running consists of 3 phases — “resource allocation/setup”, “scenario evaluation” and “resources purging”. User may want to skip one or many of them because he/she use same resources in many times. The program have a feature the skipping of phase. Especially the parameter sweeping of some test, this feature is useful. By option **-r**, the program re-employ resources of last running as nodes and VLANs. They are stored into the history file of resource allocation. The file is named RBH(resource binding history). After re-employing, the program allocates new resources for unresolved nodes and VLANs via ERM. Since new nodes and VLANs are not stored into RBH, they allocated in this step. This skip reduces the time of resource allocation and setup.

Option **-e** brings the skip of global scenario evaluation. This skip contributes a lots to the improvement of running time. However it means the skip of most of **kuma**. Option **-j**

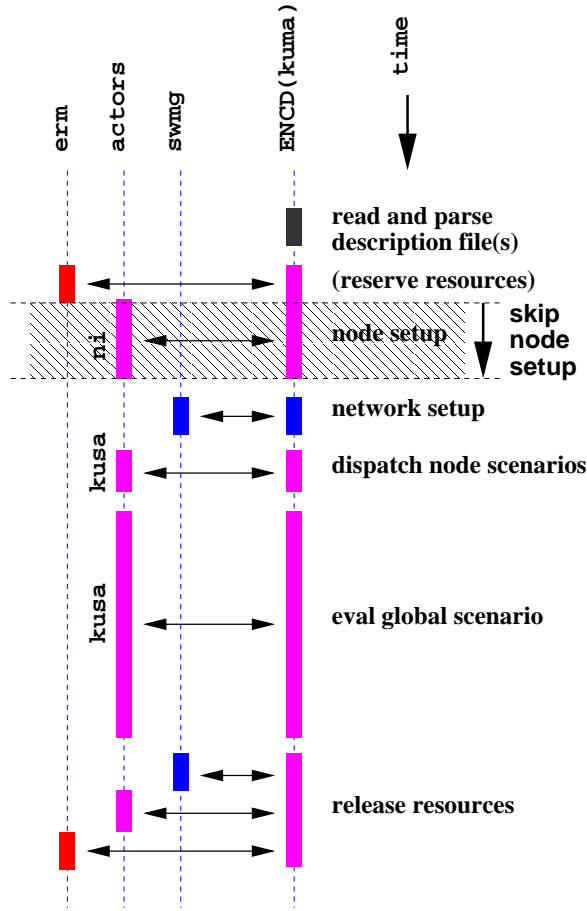


Figure 4.5: A Process Flow with skipnsetup

applies the skip of resource purging. It means that nobody can use your PC and VLANs after your running. Moreover, the network of last running lefts without destroy. If you want to apply manual operations for network, this option is useful. Figure 4.6 shows these phases and options for their skips.

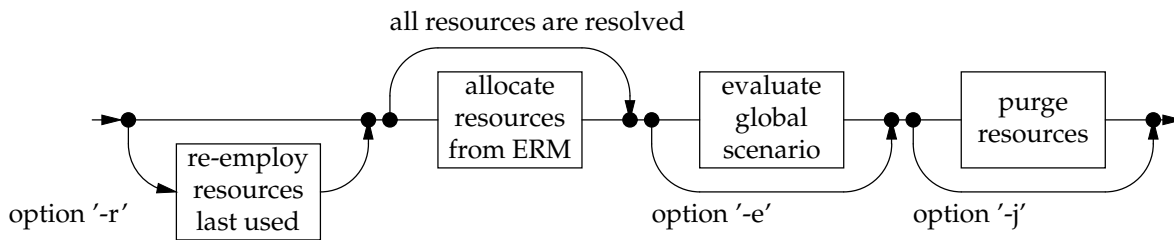


Figure 4.6: Phases of kuma and Its Skip Option

For example, if you want to evaluate global scenario 3 times with once resource allocation, run following:

```
% ./kuma ... -j ...
% ./kuma ... -r -j ...
% ./kuma ... -r ...
```

Following sequence is same above:

```
% ./kuma ... -e -j ...  
% ./kuma ... -r -j ...  
% ./kuma ... -r -j ...  
% ./kuma ... -r -j ...  
% ./kuma ... -r -e -j ...
```

4.5 Hints of File Writing

- Check your situation — server programs and their hosts
- Draw timing chart
- Image process flow parallel and/or serial
- Make diskimage of actor hosts if you need
- Consider the experiment by the role of nodes
- Care the process delay of swmg and others

Part II

Administrator's Manual

5

Boot-Loader Setup

SpringOS expects the using of meta bootloader. Meta bootloader is a boot loader of boot loader. This chapter describes 'coil', one of the meta bootloader.

5.1 coil

coil is a boot loader for multi-partition host. You can boot OS on the partition of actor hosts you want by this program.

Compile that using **make**. This compile requires **nasm** (netwide assembler.) You should install it if you do not have.

```
% cd prog
% make coil
```

After **make** you will get 6 boot loaders (**p1.bin** through **p6.bin**). These boot loaders kick the OS on the partition, respectively. You have to rename and copy them to **dman** expected (See Page II-25.)

```
% mv p1.bin boot_p1.bin
% mv p2.bin boot_p2.bin
% mv p3.bin boot_p3.bin
% mv p4.bin boot_p4.bin
% mv p5.bin boot_p5.bin
% mv p6.bin boot_p6.bin
% cp -p boot_p?.bin /tftpboot
```

When you want the boot loader for other partition (e.g., 10th partition), you can get that by rewriting one byte on these file. See **pch.p1** in this package.

NOTE *coil can boot many OS however it is not perfect. Some versions of FreeBSD and ESXi can not boot via coil.*

5.1.1 Backup MBR

Coil is not aim to be use as MBR (Master boot record; the first sector of the disk) on disk. However, It is necessary to take the backup of MBR sometimes. In some case, the system requires backup with a lot of sector to recover that. You should research your system.

The program **dd** is useful to backup MBR.

```
% dd bs=512 count=1 if=/dev/hda of=hda.mbr
```

To restore, run **dd**, too.

```
% dd bs=512 count=1 if=hda.mbr of=/dev/hda
```


6

Daemon Setup

You have to write several configuration files for daemons.

- `dhcpd.conf` for `dhcpd`.
- RDB, UDB, ACL and RULE for `erm` (or `pqerm`).
See (Section 6.8.3) and (Figure 6.2).
- `swmg.conf` for `swmg`.
- `pwmg.conf` for `pwmg`.

6.1 dhcpd.conf

`dhcpd` is not included SpringOS. You have to install that from other products. Some OSs includes that. This document expects TurboLinux built-in `dhcpd` (by ISC version 2.0pl5.)

Edit configuration file `dhcpd.conf`. It is located `/etc`, usually.

NOTE *This configuration makes the binding among MAC, IP address and boot loader filename.*

```
shared-network starbed-cde {
    server-identifier 172.16.3.101;

    option domain-name "si.star-bed.net";
    option domain-name-servers 172.16.3.101,172.16.1.101;
    option ip-forwarding off;
    option dhcp-server-identifier 172.16.3.101;
    use-host-decl-names on;

    # for Standerd Host
    group {
        next-server 172.16.3.101;
        option subnet-mask 255.255.254.0;
        option broadcast-address 172.16.3.255;
        option routers 172.16.3.254;

        subnet 172.16.2.0 netmask 255.255.254.0 {
        }

        host sintclc001 {
            hardware ethernet 00:00:4C:0F:77:C8;
```

```

        fixed-address 172.16.2.1;
        filename "sintclc001.pxe";
    }
    host sintclc002 {
        hardware ethernet 00:00:4C:0F:77:C6;
        fixed-address 172.16.2.2;
        filename "sintclc002.pxe";
    }
    host sintclc003 {
        hardware ethernet 00:00:4C:4F:A3:5E;
        fixed-address 172.16.2.3;
        filename "sintclc003.pxe";
    }
    host sintclc004 {
        hardware ethernet 00:00:4C:4F:A3:00;
        fixed-address 172.16.2.4;
        filename "sintclc004.pxe";
    }
    host sintclc005 {
        hardware ethernet 00:00:4C:4F:A3:26;
        fixed-address 172.16.2.5;
        filename "sintclc005.pxe";
    }
    ...

```

6.2 RDB: Resource Database File

You have to make data files for **erm**. **erm** required data files. This section introduces RDB (resource data base). The file consists definition of nodes (actor hosts, here), VLANs and switches with their attributes like following:

```
vlan 800-831 JAIST-team
```

```

node sintcla001 [
    bootdisk,IDE
    power,WOL,SNMP-NECMIB
    health,SSH,ICMP
    net,manage,FastEthernet,00:00:4C:0F:76:74,,172.16.0.1,
    net,empty,FastEthernet,00:00:4C:0F:76:75,,,
    net,experiment,ATM,, "siatswa001:10",,
]
node sintclf023 [
    bootdisk,IDE
    power,WOL,SNMP-NECMIB,IPMI
    health,SSH,ICMP,IPMI
    managecard,IPMI,Splitted,00:00:FF:FF:FF:FF,,172.16.99.32,
    net,manage,FastEthernet,00:00:4C:4F:A9:F8,,172.16.1.23,
    net,experiment,FastEthernet,00:00:4C:4F:A9:F9,"silaswa001:7/29",,
    net,experiment,ATM,, "siatswa004:132",,
]

switch sw01 [
    ipaddr,10.0.1.32
    cmdtype,IOS
]

```

6.2.1 VLAN Entry

Define VLAN with ID and description. Range is acceptable. **800-831** means a range from 800 to 831. Some switches have a restriction for VLAN, reserved some numbers and/or narrow range (e.g., 1024). See switch's documents.

```
vlan VLAN-ID short-description
```

6.2.2 Node Entry

Node definition consists with entity's name, disktype and NICs

```
node node-name [  
    bootdisk, disk-type  
    power, power-type  
    health, healthcheck-type  
    managecard, mc-type, mc-conn, MAC-addr, switch-port, IP-addr,  
    net, network-type, media-type, MAC-addr, switch-port, IP-addr,  
]
```

node-name:

The name of node. If you have many hosts, 3digits numbering with one-origin (e.x., a023) is strong recommended because some tools can complete to enter multiple nodes. You should fill 0, if the number less than 100. It does not support zero-origin, 2digits and 4digits.

a1	ignore
a23	ignore
a000	ignore
a03	ignore
a2314	ignore
a023	accept
a003	accept
a131	accept
sintclb017	accept

disk-type:

The type of boot disk.

literal	description
IDE	IDE; ATA
SATA	Serial ATA
SCSI	SCSI
SAS	Serial attached SCSI

power-type:

SpringOS supports some mechanism to power control of nodes. This properties are used by **pwmg**.

literal	description
WOL	Wake on LAN
SNMP-NECMIB	SNMP with NEC MIB
IPMI	IPMI

healthcheck-type:

SpringOS includes some health check programs (e.g., **sheepdog**). **erm** keeps the type of them.

literal	description
ICMP	ICMP like ping
SSH	TCP with SSH-port (22)
IPMI	IPMI

mc-type:

the type of management card.

literal	description
IPMI	IPMI
iLO	iLO

mc-conn:

the connectivity of management card.

literal	description
Splitted	Splitted from other NICs
Override	Override on other NICs

network-type:

The type for network.

literal	description
manage	management network
experiment	experiment network (historical reason)
empty	not used

media-type:

The type of media.

literal	description
ATM	Asynchronous Transfer Mode; reserved, not used
Ethernet	Ethernet (10BASE); reserved, not used
FastEthernet	Fast Ethernet (100BASE)
GigabitEthernet	Gigabit Ethernet (1000BASE)
TenGigabitEthernet	10 Gigabit Ethernet (10GBASE); reserved, not used

MAC-addr:

The MAC address of network :(colon) separated. Set empty in ATM, because it is undefined.

switch-port:

The pair of switch name and its port (with board if necessary) actor host connected. It is used to configure switch. You should describe under the manner of your switch (e.g., **3** or **7/29** .) Switch name is used for DNS resolving. Care the spell of that.

IP-addr:

The IP address of network .(dot) separated. Set this for management network. You should manage consistency between it and that of DHCP's. Otherwise, don't set this IP address. Because SpringOS allocates IP addresses automatic. When you set this, it will be conflict and/or confuse.

6.2.3 Switch Entry

Define switches with IP address and command type. The syntax is following.

```
switch switch-name [
    ipaddr, IP-address
    cmdtype, command-type
]
```

Following table shows command type of SpringOS supported switches (Section 6.6).

Command-type	Software	Hardware
IOS	Cisco IOS	Catalyst 6500 and 6509
IronWare	Brocade(Foundry) IronWare	BigIron MG8, RX-16 and -32
DLDGS	D-Link DGS	DGS-3426, -3427 and -3450

Next example defines switch "sw01" and "sw02". First one is IOS switch. Second one is IronWare switch.

```
switch sw01 [
    ipaddr,10.0.1.32
    cmdtype,IOS
]
switch sw02 [
    ipaddr,10.0.1.64
    cmdtype,IronWare
]
```

6.3 UDB: User Database File

User datafile consists sets of tuple of user, password and project. Following example means user 'john', password 'ajapa' and project 'diskbench'. When you executes **kuma**, you have to give password as its argument. User and project name should be written into the experiment description file.

```
john:ajapa:diskbench
```

6.4 ACL: Access Control List File

Following sample configure that user "george" can use all nodes and all VLANs.

```
permitrange node george all
permitrange vlan george all
```

When you want to separate 4 nodes and 3 VLANs for two person "george" and "mike".

```
permitrange node george pc1-4
permitrange node mike pc5,pc6,pc7,pc8
permitrange vlan george 1-3
permitrange vlan mike 4-6
```

Do not write duplicated or overlapping definitions like following.

```
permitrange node george pc1-4
permitrange node mike pc4-8
permitrange vlan george 1-3
permitrange vlan mike 4-6
permitrange vlan george 1-2
```

pc4 is overlapped. george's VLAN is duplicated. This configuration is ignored.

6.5 RULE: Rule File

There are many case to run **erm**. Some case want to switch its behavior by file. Other case want to that by contents of some database. Thus, we design that **erm** works according to file which describes the behavior rule of **erm**. Rule file consists of 4 rules. It is a matrix of methods (sweep and offer) and targets (node and VLAN). Simple rule file is following.

```
sweepnode [
+<RDB
&<ACL
]
offernode [
<ACL
]
sweepvlan [
+<RDB
&<ACL
]
offervlan [
<ACL
]

```

"sweep" means scanning of available targets. This is set operation. User can get available targets with the result of this operation. If the result is empty, user can use no resource. "offer" means probe that the resource is available or not. It is boolean operation. If the result of this operation is true, user can use the resource. If the result is false, **erm** does not allow the user to use the resource.

Empty line is ignored as comment. Line which starts #(sharp) is comment. Line which starts whites(space and tab) is jointed to previous line. Rule is defined with block which lines from [to]. In the block, lines have 1 or 2 operators. First operator is one of set operators -(SUB), +(ADD) and &(AND). The program recognize that first operator is ADD when it is not specified. Second one is an < (import) operators RDB and ACL file or ?(SQL). Figure 6.1 shows the flow of operator parsing. Previous example means that program imports from RDB-file and masks by ACL-file.

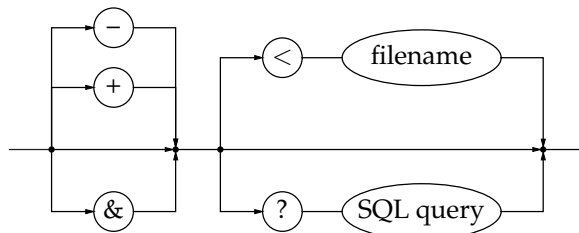


Figure 6.1: The operator parsing in rule file

When you use **pqerm** (**erm** with PostgreSQL), you can insert SQL queries in these rules. Following example shows the query for name list of paul's PC.

```
sweepnode [
+?SELECT name FROM ptable WHERE owner='paul'
&<ACL
]

```

Perhaps, you want to use username of **erm** client's. In that case, user \$user in the query. **pqerm** replace \$user to individual username.

```
sweepnode [
+?SELECT name FROM ptable WHERE owner='$user'
&<ACL
]
```

To escape \$, type \$ twice. To escape variable name in alphabetical literal, use { (left curly bracket; left brace) and } (right curly bracket; right brace). In curly bracket, the program supports effects like \${name:6}. It means first 6 letters in \$name. When \$name holds "SkyWalker", following sample shows such escapes and effects.

input	output
\$name-0	SkyWalker-0
\${name}abc	SkyWalkerabc
\${name:6}	SkyWal
\$\$def	\$def

Following table shows the name of variables in "sweep".

name	description
user	name of user
project	name of project
ENV	environment variables \$ENV(LANG) is replaced language definitions.
LOCALTIME	date in localtime with format Format is according to C language function strftime(). For example, date is %Y%m%d or %y%m%d .
GMTIME	date in GMT (probably UTC) Format is same of LOCALTIME

In "offer", you can use more variables in following table.

name	description
nodename	name of node
nodepre	prefix of nodename
nodenum	number of node
nodenumZIIIId	number of node by 3 digits with zero padding
vlannum	number of VLAN

SQL query often become long. You should use concatenate them with white-head line.

```
sweepnode [
+?SELECT name
      FROM ptable
      WHERE owner='$user'
&<ACL
]
```

Following is an example of complex rule file.

```

sweepnode [
+?SELECT sc_clgrp||sc_bgiclno||'-'||sc_endclno
  FROM sc_schedule
  WHERE sc_proid='$user'
  and '$LOCALTIME(%Y%m%d)'>=sc_bgiday
  and '$LOCALTIME(%Y%m%d)'<=sc_endday
]

offernode [
?SELECT * FROM sc_schedule
  WHERE sc_proid='$user'
  and '$LOCALTIME(%Y%m%d)'>=sc_bgiday
  and '$LOCALTIME(%Y%m%d)'<=sc_endday
  and '${nodepre:6}'=sc_clgrp
  and '$nodenumZIIId'>=sc_bgiclno
  and '$nodenumZIIId'<=sc_endclno
]

sweepvlan [
+?SELECT vl_vlanfr||'-'||vl_vlanto FROM vl_vlan
  WHERE vl_proid='$user'
  and '$LOCALTIME(%Y%m%d)'>=vl_bgiday
  and '$LOCALTIME(%Y%m%d)'<=vl_endday
]

offervlan [
?SELECT * FROM vl_vlan
  WHERE vl_proid='$user'
  and '$LOCALTIME(%Y%m%d)'>=vl_bgiday
  and '$LOCALTIME(%Y%m%d)'<=vl_endday
  and '$vlannum'>=vl_vlanfr and '$vlannum'<=vl_vlanto
]

```

6.6 swmg.conf

swmg stands "switch manager." At first, you should make new file for configuration of **swmg**. Typically name of the file is **swmg.conf**.

```
rmanager ipaddr "10.9.7.31" port "4989"
user "john"
passwd "lennon"
project "imagine"
swpasswd "shone"
swtype name "swa001" type "IOS"
swtype name "swa002" type "IronWare"
```

- **port** sets service port(TCP) of swmg
Usually, it is not required because it is set default value(1240).
- **rmanager** sets resource manager parameters
 - **name** sets hostname of erm
 - **ipaddr** sets IP address of erm
 - **port** sets service port(TCP) of erm
- **user** sets user name of swmg as erm user
- **passwd** sets password of swmg as erm user
- **project** sets project name of swmg as erm user
- **swpasswd** sets default password for switches
- **swtype** sets attributes of individual switch
 - **name** sets name of the switch
 - **type** sets command type of the switch

Currently supported switch are following 3 types. Do not use "obsoleted" and "reserved" types.

token	Software	Hardware
IOS	Cisco IOS	Catalyst 6500 and 6509
IronWare*	Brocade(Foundry) IronWare	BigIron MG8, RX-16 and -32
DLDS	D-Link DGS Series	DGS-3426, -3427 and -3450
EXXOS	Extreme XOS ; reserved	Summit X650 and X450
OSL2	Alaxala ; reserved	AX-2400
CATOS	Cisco Catalyst OS; obsoleted	Catalyst 6500

- **user** sets user name for the switch
- **passwd** sets login password for the switch
- **epasswd** sets enable password for the switch
- **opt** sets some options
It is reserved for future features.

* In near future, **IronWare** may be obsoleted. Because we just planning to support FastIron series. It is different from BigIron although its software is named IronWare.

When you use switch with no username and common password "penyrain", you should write following.

```
...
swpasswd "penyrain"
swtype name "sw1" type "IOS"
swtype name "sw2" type "DLDGS"
...
```

If switch "sw2" requires username "ringo", you should write following.

```
...
swtype name "sw1" type "IOS" user "ringo"
...
```

If switch "sw1" requires username "ringo" and password "starr", you should write following.

```
...
swtype name "sw1" type "IOS" user "ringo" passwd "starr"
...
```

This configuration syntax allows enable password also. If switch "sw1" requires enable password "obladi", you should write following.

```
...
swtype name "sw1" type "IOS" user "ringo" passwd "starr" epasswd "obladi"
...
```

Some switch requires user and password to login, but no password to enable. You should write following in such case. Password "-" is special for indication of common password.

```
...
swtype name "sw001" type "OSL2" user "admin" passwd "-" epasswd ""
...
```

6.7 pwmg.conf

pwmg stands "power manager." The program requires configuration file which is often named **pwmg.conf**.

```
rmanager ipaddr "10.9.7.31" port "4989"
user john
passwd lennon
project imagine
ipmiuser jack
ipmipasswd betty
```

The syntax of the file is same that of **swmg** (Subsection 6.6) except **swpasswd**, **swtype**, **ipmiuser** and **ipmipasswd**. **swpasswd** and **swtype** are removed.

- **ipmiuser** sets username for IPMI
- **ipmipasswd** sets password for IPMI

pwmg uses environment variables **IPMI_USER** and **IPMI_PASSWORD** for **ipmiuser** and **ipmipasswd**.

Using **ipmitool** of OpenIPMI or others tools, you can test them.

```
% ipmitool -H 10.0.0.1 -u john chassis status
```

6.8 Start Daemons

Before starting server programs, you have to decide where does program run. See required and recommended conditions in previous Section 2.1.

Program dependency is following (make like syntax):

```
kuma: erm kusa fncp dman dhcpd tftpd ftpd wolagent
kusa: kuma ifscan fncp dman dhcpd tftpd ftpd
ni: kuma fncp dman dhcpd tftpd ftpd
ifscan:
dman: tftpd
swmg: erm
pwmg: erm
erm:
fncp:
wolagent:
ftpd:
tftpd:
dhcpd:
```

Experiment programs (**kuma**, **kusa** and **ni**) are depend servers deeply. Since **ifscan** is not used standalone, don't care that in this time.

Server programs are independent each other except **dman**. You can start them without order. Also you can start **dman** without order. However, its results are shown through **tftpd**. You should start **dman** after **tftpd** starting.

Furthermore, **dhcpd**, **tftp** and **ftpd** are configured as automatically starting at OS booting or on-demand starting by **inetd**(or **xinetd**) in many OSs. You should check the conflict between these server programs of SpringOS and those of OS built-ins.

The recommended order of SpringOS server programs booting is **erm**, **fncp**, **dman** and **wolagent**.

6.8.1 tftpd

tftpd is not included SpringOS. following document is described for tftp-hpa version 0.28 .

6.8.2 ftpd

ftpd is not include SpringOS. Nothing condition is required to **ftpd**, mention above. You should care the direction of connection. SpringOS expects that the program supports PASS command (passive data connection.)

Check access permission of the program. Because experiment users have to write login account and password into the experiment description file, special user for SpringOS are recommended. The below example of description file expects special user 'install'. You should change that suits your situation.

6.8.3 erm

erm is the heart of SpringOS. This subsection describes compiling of that and its configuration.

If your test-bed holds larger or equal 2048. You should change the maximum number of nodes. Edit definition of **ND_MAX_NODE** in **Makefile**. Do not change **nd.h**. Following is an example to setup that as 8192.

```
DEFS=-DND_MAX_NODE=8192
```

Run **erm** with **-R**, **-U**, **-A** and **-S** options.

```
% ./erm -R resourcefile -U projectfile -A aclfile -S rulefile
```

erm records ownership into a file, in default it is named **backup**. You can change the name of the file with **-B**-option. The program read this at start time. So, it is "hot start". With **-C** option, you can "cold start" if necessary.

```
% ./erm -C -R datafile -U userfile -A aclfile -S rulefile
```

6.8.4 Compile and Execution of **pqerm**

pqerm is DBMS-base **erm**. It uses PostgreSQL engine and its libraries. When your platform has PostgreSQL and its libraries

If you want to compile **pqerm**, you should 1) install PostgreSQL engine and its libraries 2) re-run **configure** and **make**. **configure** probes PostgreSQL related files and changes **Makefile** and **config.h**.

```
% yum -y install postgresql postgresql-server postgresql-devel
% ./configure
% make
```

You can compile **pqerm** manually in **prog**-directory. However, you may edit **Makefile** and **config.h**.

```
% vi config.h
% cd prog
% vi Makefile
% make pqerm
```

To "hot start", you type following.

```
% ./pqerm -R resourcefile -U projectfile -A aclfile -S rulefile
```

pqerm supports "cold start", you type following. This running takes several minutes to drop and re-create tables.

```
% ./pqerm -C -R datafile -U userfile -A aclfile -S rulefile
```

6.8.5 **wolagent**

wolagent is a Perl script. You should check running of **perl** before starting **wolagent**.

```
% perl -v
```

When you got the version of **perl**, you can use **perl**, certainly.

```
% cd prog
% perl ./wolagent
```

6.8.6 **fncp**

fncp is a kind of HTTP server. Its features are so poor. The possibility of compiling error is low.

Run **fncp** with no argument.

```
% cd prog
% ./fncp
```

Table 6.1: Service Ports of SpringOS

program	port#	protocol	description
dhcpcd	UDP 67	DHCP	server
	UDP 68	DHCP	client
tftpd	UDP 69	TFTP	control
ftpd	20	-	data (temporary)
	21	FTP	control
erm,pqerm	1234	ERRP	resource information/reservation
dman	1236	DMP	directory manipulation (symlink and information)
fncp	1238	HTTP	node configuration (redirect; FNCP)
swmg	1240	SWCP	switch management
pwmg	1242	PWCP	power management
kiyomitsu	1244	HTTP	file uploading
wolagent	5959	WOLAP	issue WoL magic packet
kuma	3456	HTTP	node configuration (ENCD)
	3458	ESQP	status reporting
kusa	2345	*noname*	node scenario input
ni	80	HTTP	status reporting

6.8.7 dman

dman is the symlink switcher. See also Section 1.7.

Wake **dman** with **tftpd**'s directory (e.g., **/tftpboot**.)

```
% cd prog
% ./dman -D /tftpboot
```

6.8.8 Kiyomitsu

Kiyomitsu is HTTP server for user's file uploading.

```
% cd prog
% ./kiyomitsu
```

6.9 Check Service Ports

The number of listening ports is 14. Check these ports if you feel strange things or want to hack them. You may want them to other configuration (e.g., firewall).

Table 6.1 shows these ports. You can check them via **netstat**, **lsof** and other programs.

protocol	chapter	description
DMP	12	DMP
ERRP	8	ERRP
ESQP	9	ESQP
HTTP+	-	-
PWCP	11	PWCP
SWCP	10	SWCP
WOLAP	13	WOLAP
s-exp	-	-
s-exp-i	-	-
text	-	-

program order			protocol order			port order		
program*	protocol	port	program	protocol*	port	program	protocol	port*
dhcpcd†	DHCP-S	67 u	dhcpcd†	DHCP-C	68 u	ftpd†	FTP-D	20
dhcpcd†	DHCP-C	68 u	dhcpcd†	DHCP-S	67 u	ftpd†	FTP	21
dman	DMP	1236	dman	DMP	1236	dhcpcd†	DHCP-S	67 u
erm	ERRP	1234	erm	ERRP	1234	dhcpcd†	DHCP-C	68 u
fncp	HTTP+	1238	pqerm	ERRP	1234	tftpd†	TFTP	69 u
ftpd†	FTP	21	kuma	ESPQ	3458	snmpd†	SNMP	161 u
ftpd†	FTP-D	20	ftpd†	FTP	21	erm	ERRP	1234
kiyomitsu	HTTP	1244	ftpd†	FTP-D	20	pqerm	ERRP	1234
kuma	HTTP+	3456	sheepdog	HTTP	2468	dman	DMP	1236
kuma	ESPQ	3458	kiyomitsu	HTTP	1244	fncp	HTTP+	1238
kuma	s-exp-i	4569	kuma	HTTP+	3456	swmg	SWCP	1240
kusa	s-exp-i	4567	pickup	HTTP+	3456	pwmg	PWCP	1242
kusa	s-exp	2345	wipeout	HTTP+	3456	kiyomitsu	HTTP	1244
mlog	text	3460	fncp	HTTP+	1238	kusa	s-exp	2345
ni	HTTP+	2347	ni	HTTP+	2347	ni	HTTP+	2347
pickup	HTTP+	3456	pwmg	PWCP	1242	sheepdog	HTTP	2468
pqerm	ERRP	1234	snmpd†	SNMP	161 u	kuma	HTTP+	3456
pwmg	PWCP	1242	swmg	SWCP	1240	pickup	HTTP+	3456
sheepdog	HTTP	2468	tftpd†	TFTP	69 u	wipeout	HTTP+	3456
snmpd†	SNMP	161 u	wolagent	WOLAP	5959	kuma	ESPQ	3458
swmg	SWCP	1240	kusa	s-exp	2345	mlog	text	3460
tftpd†	TFTP	69 u	kuma	s-exp-i	4569	kusa	s-exp-i	4567
wipeout	HTTP+	3456	kusa	s-exp-i	4567	kuma	s-exp-i	4569
wolagent	WOLAP	5959	mlog	text	3460	wolagent	WOLAP	5959

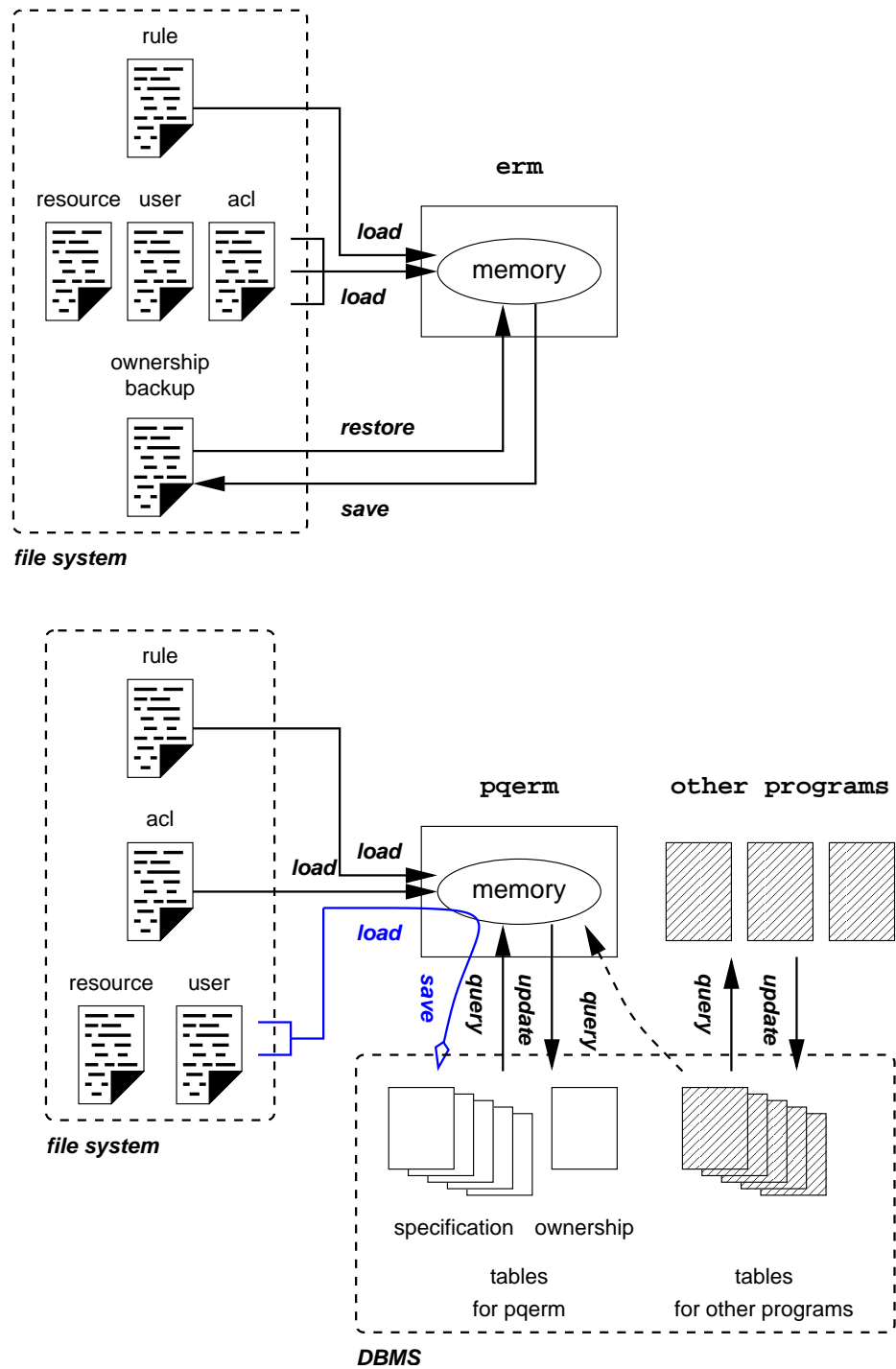


Figure 6.2: erm data operations

7

Diskimage Making for Disk Operations

Disk handling consists the program-**ni** and the diskimage to wake **ni**. This chapter shows you how to install them. See also Section 3.2 for user-side operation by **pickup/wipeout**.

Because **ni** writes disks, OS for **ni** should not access a target disk. A way to satisfy the requirement is diskless. You may employ other ways, using other disk (e.g, floppy, NFS). This document does not cover them.

7.1 **ni**

ni is the node initiator. The program reads and writes diskimage through network. So, it is means remote backup and restore.

Try running of **ni** following command, manually. To run **ni**, you have to know FNCP's service URL. Replace IP address and port number to your **fncp**'s (default port number is 1238).

```
% ./ni http://1.2.3.4:1238/
```

7.2 **SNMPmine**

SNMPmine(**snmpmine**) is the SNMP responder. According to request, the program applies **shutdown** or **reboot** on actor host.

The program aims at NEC MIB. Unfortunately, the MIB is private.

7.3 **Setup Diskless OS via PXE**

The memo for making of diskless PicoBSD (a kind of FreeBSD) using PXE is published at StarBED WWW Server (<http://www.starbed.org/tips/netboot/index-j.html>.) Unfortunately, it is written by Japanese. This subsection is translated from the WWW page.

Steps in the subsection is required the PC which installed FreeBSD with kernel source. You have to pickup or install the PC. The author tested them on FreeBSD-4.6 .

7.3.1 **pxeboot**

pxeboot is bootloader for PXE. You can get that by making with **LOADER_TFTP_SUPPORT**.

```
# cd /usr/src/sys/boot/i386
# make -D LOADER_TFTP_SUPPORT
```

7.3.2 loader.rc

The file `loader.rc` is configuration file of PXE booting.

```
load /FreeBSD/kernel_${boot.netif.ip}.gz
load -t mfs_root /FreeBSD/diskimage_${boot.netif.ip}
autoboot 10
```

The variable `${boot.netif.ip}` is replaced to the IP address of each hosts. First line specifies kernel file (e.g., `kernel_172.16.1.5.gz`.) Second line specifies kernel file (e.g., `diskimage_172.16.1.5.gz`) also.

7.3.3 kernel

A MFS and MD_ROOT enabled kernel is required. You can use GENERIC kernel. After to apply `kgzip`, copy the kernel to the TFTP server.

7.3.4 diskimage

1) Login the host with root.

2) Download a kit file set. The URL of the file is following.

```
http://www.starbed.org/tips/netboot/mkdiskimage-20050803.tgz
```

3) Extract files from the file.

```
# tar xvfz mkdiskimage-20050803.tgz
# cd mkdiskimage-20050803
```

4) Edit `crunch.conf.local` if you want change contents of `crunch`.

```
# vi crunch.conf.local
```

5) Change the name and size of diskimage.

```
# vi mkdiskimage.sh
```

These parameters in following table are defined in the file `mkdiskimage.sh`. Without edit, the capacity of the diskimage is approx. 7MB.

name	description	default
IMAGE_SIZE	the size of diskimage	30000KB
IMAGE_NAME	the name of diskimage	diskimage

6) The directory `mfs_tree/etc` will become `/etc` on booted OS. Modify them if you want.

7) Make password database. Generate MD5 hashed string by OpenSSL or its compatibles. Enter your password twice. The string of last line in following example is the hashed string.

```
# openssl passwd -1
Password: your password
Verifying - Password: your password
$1$zYB0U39b$kdL/yM3O7FaPNCvCsQYd3/
```

Using `vipw`, add the hushed string into password file.

```
# cd mfs_tree/etc
# vipw -d ./
```

The password of root is empty in original.

```
root:0:0:0:0:root:/root:/bin/sh
```

Set hashed string as your password into the file.

```
root:$1$zYB0U39b$kdL/yM3O7FaPNCvCsQYd3/:0:0:0:0:root:/root:/bin/sh
```

8) Make the diskimage.

```
# cd ..
# ./mkdiskimage.sh all
```

9) Check existence the diskimage. If you find the diskimage (default name **diskimage**) in current directory, you got the diskimage.

7.4 Diskimage Customize for ni

For SpringOS, this section describes the customization of the diskimage.

7.4.1 Copy ni into Diskimage

Since the diskimage does not include **ni**, you should copy in to the file. The file is named 'ni02.fs' in next example. You can see that in Section 4.1.

```
# vnconfig -c -s labels vn0 $PWD/ni02.fs && mount /dev/vn0 /mnt
# cp -p /usr/home/ni/src/ni /mnt/usr/bin/ni
# cp -p /usr/home/mine/src/snmpmine /mnt/usr/bin/snmpmine
```

7.4.2 Automatic Waking of ni

You must configuration to wake **ni** in booting. Because actor hosts are rebooted automatically, SpringOS related programs expect automatic waking of **ni**.

To wake **ni**, you should set it in booting procedures. Edit **rc.local**.

```
# vi /mnt/etc/rc.local
```

Typically command sequence for the waking is following:

```
/usr/bin/ni http://1.2.3.4:1238/
```

In actually, the **rc.local** in our **ni02.fs** have following commands:

```
ulimit -m 200000
ulimit -d 200000
echo -n "SNMPmine "
/usr/sbin/snmpmine > /dev/null 2>&1 &
echo -n "NI "
/usr/sbin/ni http://172.16.3.101:1238/nodeconfig.txt
```

ulimit, a shell built-in command, is a parameter modifier for user process. **snmpmine** is a self-destruction program to apply **reboot** and **shutdown**.

To generate and/or edit some script in **/etc** directory may makes similar behavior, also. This document does not cover that.

7.4.3 Deploy the Diskimage

Copy the diskimage file to `tftpd`'s data directory via `scp`, `ftp` and others.

```
# umount /mnt && vnconfig -u vn0  
# scp -p $PWD/ni02.fs root@172.1.3.101:/tftpboot/FreeBSD
```

References

SpringOS is designed for StarBED. You may find reasons for design of SpringOS.

- [1] Toshiyuki Miyachi, Ken-ichi Chinen and Yoichi Shinoda: Automatic Configuration and Execution of Internet Experiments on an Actual Node-based Testbed, Tridentcom 2005, Trento, Italy, ISBN 0-7695-2219-X, pp.274–282, Feb, 2005.
- [2] Ken-ichi Chinen, Toshiyuki Miyachi, Yoichi Shinoda: A Rendezvous in Network Experiment — Case study of Kuroyuri, Tridentcom 2006, Barcelona, Spain, ISBN 1-4244-0106-2, March 1-3, 2006.
- [3] Toshiyuki Miyachi, Takeshi Nakgawa, Ken-ichi Chinen, Shinsuke Miwa and Yoichi Shinoda: StarBED and SpringOS Architectures and Their Performance, Tridentcom 2011. printing
- [4] Homepage of StarBED Project. <http://www.starbed.org/>
- [5] Tips of netboot (Japanese). <http://www.starbed.org/tips/netboot/index-j.html>
- [6] Hokuriku IT Open Laboratory (Official Name of the Organization which holds StarBED) . <http://www.hokuriku-it.nict.go.jp/english/>

Historical Issues

SpringOS

The name 'SpringOS' originates from a spring in a bed. Spring supports the weight of object on a bed. Similarly, SpringOS supports a network test-bed.

sbrm

erm is called **sbrm** until early 2005. **sbrm** means "StarBED Resource Manager." The model of **sbrm** is useful in other test-bed. Then, we rename it as **erm**.

master/slave

K language evaluators are called **master** and **slave** in early because they work under master-slave model. Someday, we recognize that these name are too generic. We change them **kuma** and **slave**. They stand "Kuroyuri master evaluator" and "Kuroyuri slave evaluator".

ifsetup

ifsetup is shell script. It is executor of **ifconfig**. After **netif** statement of the language is available, **ifsetup** is no longer.

wol_agent

wolagent was made Sep 2005 as an instead of **wol_agent** . Because rights of **wol_agent** are not clear.

Boot loaders

Initially, StarBED has the special boot loader for multi-partition (hereinafter also referred to as "alice.") However, it is not free. Then StarBED project decided to develop free boot loader. It is named "coil." So, coil is substitute of alice.

Actually, alice is a set of boot loaders. Each boot loader boots the OS in the partition of which it takes charge. You have to know following 3 boot loaders.

name	partition	OS
boot_p1.bin	#1	WINDOWS Server 2000
boot_p2.bin	#2	FreeBSD 4.2
boot_p5.bin	#5 (first sub-partition on partition #4)	TurboLinux 7

'simulation' field in resource datafile of erm

Since StarBED is called "Internet Simulator" in early, experiment networks in the test-bed is called "simulation network". The term lefts in that file.

NEC MIB

NEC's management programs are installed in StarBED. To introduce the feature, SpringOS supports that. **SNMPmine** and **sbpsh** run under its manner.

SpringOS/VM

SpringOS/VM is a kind of extended SpringOS. It supports VMware. Users can increase the number of nodes for nextwork experiment. SpringOS/VM may be released if requested.

Part III

Command Reference

NAME

ENCD - experiment node configuration driver, a function of SpringOS

DESCRIPTION

ENCD is a function of SpringOS. It send driving command for nodes like diskimage backup/restore and reboot. Currently, SpringOS have **kuma**, **pickup** and **wipeout** as ENCD.

In most case, nodes reaches ENCD via **fncp(1)**.

SEE ALSO

kuma(1), pickup(1), wipeout(1), fncp(1)

NAME

bswc.pl - batch switch configuration program

SYNOPSIS

```
bswc.pl -h
bswc.pl [-d] [-S] [conf-file]
```

DESCRIPTION

bswc is a client of **swmg**. Via **swmg**, the program changes network configuration. User have to write configuration steps with node-ports. The program converts node-ports to switch-ports. The program use env.-var. BSWCPASSWD as password for RM if present.

OPTIONS

The following options are supported:

```
-h          print help messages.
-d          enter debug mode
-S          disable access for swmg.
```

NODEPORT

Nodeport consists nodename and netif number. Interfaces are numbered with zero-origin according to the order of appearance in node information of erm. Letter "m" before netif number means management interface. Letter "e" before netif number and no letter means experiment interface.

*node:[em]*netif#* — untagged VLAN port

*!node:[em]*netif#* — tagged VLAN port

For example, you can write nodeports like following:

```
b23:2 — b023's third experiment interface with untagged
a33:e3 — a033's fourth experiment interface with untagged
f133:m — f133's first management interface with untagged
!d7:3 — d007's fourth experiment interface with tagged
```

CONF-FILE

The syntax of *conf-file* is following:

```
rm IP-addr port — resource manager
rmuser user
rmpasswd pass
rmproject project — username, password and project in rm
sm IP-addr port — switch manager
joinvlan vlan-num nodeport1 [nodeport2 ...] — join nodeports to VLAN
leavelan vlan-num nodeport1 [nodeport2 ...] — leave nodeports from VLAN
rmvlan vlan-num — remove VLAN
activate nodeport — activate (a.k.a. no shutdown) port
deactivate nodeport — deactivate (a.k.a. shutdown) port
sleep sec
exit
```

Following example shows a adding 2 nodes to VLAN 800.

```
$ cat nettopo
rm localhost 1234
rmuser starbed
rmpasswd foobar
rmproject starproj
sm localhost 1240
activate devpc2:1
activate devpc3:1
joinvlan 800 devpc2:1 devpc3:1
```

```
exit
$ perl bswc.pl nettopo
```

SEE ALSO

swmg(1), erm(1), SWCP

NAME

dman - directory manipulator

SYNOPSIS

dman -h

dman [-p *port_no*] [-P] [-D *dir*] [-d *num*]

DESCRIPTION

dman makes and removes symbolic links on the directory.

OPTIONS

The following options are supported:

-h print help messages.

-p *port_no* set port number (default 1236.)

-D *dir* set the dictory of working. In most case, it is that of TFTPd like /tftpboot.

-d *num* set debug level.

-P disable path checking. **dman** checks paths in the request. If the path is ignore or out-side of working directory, the proragm rejects the request.

EXAMPLE

\$ dman -D /tftpboot

SEE ALSO

ENCD(1), DMP

NAME

erm - experiment resource manager

SYNOPSIS

```
erm -h
erm -v
erm [-td] [-V] [-C] [-p port_no] [-R file] [-U file] [-B file] [-s sec] -S rule -A acl
```

DESCRIPTION

erm is a resource manager. The program holds the specification of resources and the ownership of them. Furthermore, it dispatch resource to user according to his/her client request.

OPTIONS

The following options are supported:

-h	print help messages.
-v	print version.
-t	trace mode.
-d	debug mode.
-V	verbose contents. The program listing contents. Because of the list is too long, this option is used only debugging.
-p <i>port_no</i>	set port number (default 1234.)
-R <i>file</i>	set resource database file.
-U <i>file</i>	set user database file.
-A <i>file</i>	set access control list file.
-S <i>file</i>	set rule file.
-B <i>file</i>	set ownership backup file.
-s <i>num</i>	set the interval of ownership backup.
-C	work as cold start (default warm start.)

DATA LIFECYCLE

erm expects that resource specification does not change. When they are changed you should restart the program.

The program knows that user and project change sometime. By a signal SIGHUP, you can command it re-reading of UDB file.

Continuity of resource ownership information must be guaranteed. Then the program stores that ownership information into a file with a interval. Usually the program read the file before accepting of requests. You can skip that step by -C option. In such case, the program does not care ownership on last running.

RULE OF RESOURCE AVAILABILITY

erm decides that a user can use the resource by rules. There are 4 rules **sweepnode** **offerode** **sweepvlan** and **offervlan**. sweep means gathering of available resource. Typical its definition is RDB and its masking by ACL.

```
sweepnode [
+<RDB
&<ACL
]
```

offer means query by client request. Typical its definition is masking by ACL.

```
offerode [
&<ACL
]
```

pqerm(1) accept more complex rule using SQL.

SEE ALSO

pqerm(1), ERRP

NAME

fncp - facility node configuration pilot

SYNOPSIS

fncp -h
fncp [-p *port_no*] [-H *num*] [-Q *num*]

DESCRIPTION

fncp leads node agent like **ni** to ENCD (**pickup**, **wipeout** and **kuma**.) Using HTTP redirect, the program navigates actor hosts to their driver (ENCDS.)

OPTIONS

The following options are supported:

-h print help messages.
-p *port_no* set port number (default 1238.)
-H *num* set the number of HTTP handlers
-Q *num* set the length of request queue

Options -H and -Q are tuning parameters. Usually user have no need to change them.

SEE ALSO

ENCD(1)

NAME

ifscan - network interface scanner

SYNOPSIS

ifscan [-v]

DESCRIPTION

ifscan scans network interfaces and prints them into stdout. Its output is used to call **ifconfig** and related. **netifft**-statement of Kuroyuri expects the output of **ifscan**.

OPTIONS

-v verbose mode. **ifscan** prints messages into stderr.

EXAMPLE

Following output means that the machine has 7 network interfaces. This program does not care interfaces type (e.g., logical or physical.)

```
$ ./ifscan > ifscan.out
$ cat ifscan.out
# 7 interfaces; lo eth0 eth1 eth2 eth3 eth4 eth5
nic lo 00:00:00:00:00:00
nic eth0 00:30:48:56:5f:78
nic eth1 00:30:48:56:5f:79
nic eth2 00:04:23:c8:87:0c
nic eth3 00:04:23:c8:87:0d
nic eth4 00:04:23:c8:87:0e
nic eth5 00:04:23:c8:87:0f
```

BUGS

ifscan work well on Linux and BSDs except SunOS 5.x.

SEE ALSO

ifconfig(1), netifft(K)

NAME

kiyomitsu - file upload server

SYNOPSIS

kiyomitsu -h

kiyomitsu [-P *port_no*] [-D *dir*] [-r *host*]

DESCRIPTION

kiyomitsu aims to achive user file uploading without user account on management hosts. When your management hosts of your facility have your account to file create and modify, you do not require to use the program.

kiyomitsu is a kind of HTTP server. You should access the program using WWW browsers.

OPTIONS

The following options are supported:

-h print help message.

-t trace mode.

-d debug mode.

-P *port_no* set port number (default 1244.)

-D *dir* set working directory. You should set that of **tftpd** and **dman**.

-r *host* set the host of **erm**.

SEE ALSO

erm(1)

NAME

kuma - Kuroyuri master

SYNOPSIS

kuma -h

kuma -v

kuma [-t] [-d] [-p *passwd*] [-r] [-c *file*] [-ej] [-D *masks*] [-U *masks*] [-P *port_no*] [-Q *port_no*] [-L *num*] [-RWFTNS] [-iI] [-m] [-Z] [*var.def.s*] *files*

DESCRIPTION

kuma is an evaluator over the commander host. *var.def.s* means variable definitions. **files** are experiment description file(s).

OPTIONS

The following options are supported:

-h	print help message.
-v	print version.
-p <i>passwd</i>	set password for ERRP.
-r	reuses resources of last session
-c <i>file</i>	set resource binding history (rbh) file.
-e	skips evaluation of global-scenario.
-j	skips resource releasing at program termination.
-d	debug mode
-D <i>masks</i>	set debug masks.
-U <i>masks</i>	unset masks.
-P <i>port_no</i>	set port number for ENCD.
-Q <i>port_no</i>	set port number for ESQP.
-L <i>num</i>	set the length of parsing buffer.
-R -W -F -T -N -S	set fake of connections.
-i	enter passive interactive mode
-I	enter active interactive (shell) mode
-m	enter active interactive (monitor) mode
-Z	wakes slaves on same hosts.

SCENARIO EVALUATION

When the program find *scenario*-statement in experiment description files as input, it recognize that the script has distribution processing and setup up them.

Without *scenario*-statement, the script file is evaluated as standalone program. It may useful for some purpose like following (address calculation).

```
% cat a.k
for(i=253;i<257;i++) {
    print addradd("10.0.0.1/16", i)
}
% ./kuma a.k
10.0.0.254/16
10.0.0.255/16
10.0.1.0/16
10.0.1.1/16
```

ENVIRONMENT VARIABLES

KUROPASSWD

SEE ALSO

kusa(1), kush(1), K Language Reference Manual

NAME

kusa - Kuroyuri slave

SYNOPSIS

```
kusa -h
kusa [-t] [-1] [-q] [-p port_no] [-i] [-I port_no] [-z file]
```

DESCRIPTION

kusa is an evaluator over actor hosts.

OPTIONS

The following options are supported:

-h	print help messages.
-p <i>port_no</i>	set port number for kuma (default 2345.)
-1	enter one-shot mode (default multi-shot mode.) The program exits when node-scenario is over.
-q	quiet mode.
-z <i>file</i>	set the name of log-file.
-i	enter interactive mode.
-I <i>port_no</i>	set port number for kush (default 4567.)

PROCESS LIFECYCLE

In multi-shot(default) mode, **kusa** launches child processes to handle node scenario. In that case, logs of these proccces are append into one file. To save disk space and/or check detail you may want to rorate. However, in this mode, rotation may cause a suddenly termination of process.

```
% kusa >> kusa.log 2>&1
```

One shot mode **kusa** terminates after each evaulation of node scenario. You should make contineous running using some technique. Popular way to this is a loop by shell.

```
#!/bin/sh
while /bin/true
do
    DATE=`date +%Y%m%d%H%M`
    kusa -1 >> kusa.$DATE.log 2>&1
    sleep 1
done
```

Using this way, log files are separated per node scenario running. You can check file contents easily. You can delete old log files by **find(1)**, if necessary.

SEE ALSO

kuma(1), kush(1), mlog(1)

NAME

mlog - monitering logger

SYNOPSIS

mlog -h

mlog [-p *port_no*] [-o *file*]

DESCRIPTION

mlog receive log and store them into a file.

OPTIONS

The following options are supported:

-h print help message.

-p *port_no* set port number (default 3460.)

-o *file* set the name of log-file.

SEE ALSO

kuma(1), kusa(1), ev(1)

NAME

ni - node initiator

SYNOPSIS

```
ni -h
ni [-q] [-A agentID] [-S sessID] [-p port_no] [-r times] [-w times] [-W sec] [-R times] [-i
sec] [-I sec] [-B path] [-s] URL-config
```

DESCRIPTION

ni is a disk operation program over actor host. Usually, it is called at end of boot sequence in disk setup OS image. It means diskless PXE booting or USB booting. *URL-config* is the path of FNCP or ENCD.

OPTIONS

The following options are supported:

-A <i>agentID</i>	set agent ID.
-S <i>sessID</i>	set session ID.
-p <i>port_no</i>	set the port number of reporting.
-r <i>times</i>	set the times of configuration retrieval retry.
-w <i>times</i>	set breath radix of configuration retrieval.
-W <i>sec</i>	set breath interval of configuration retrieval.
-R <i>times</i>	set the times of disk-image retrieval retry.
-i <i>sec</i>	set interval of monitoring — ticks of monitoring. <i>ni</i> checks transfer size and elap time to progress report in several seconds.
-I <i>sec</i>	set interval of progress report by elap time. the value of this parameter should large or equal than that of -i option.
-B <i>path</i>	set path for reboot .
-s	disable syslog reporing in initially. However, ENCD may change it. It is some difficult that this option is work or not.

REPORTING

ni reports progress by transfer size and elap time. Size-base reporting aims to reduce the number of report. *ni* reports when it finds 1M, 1G and some good limit size. Moreover *ni* reports when the size is some good limit ratio (e.g., 10, 90, 99

EXAMPLE

```
% ./ni http://1.2.3.4:1238/
```

SEE ALSO

ENCD(1), reboot(1)

NAME

pickup/wipeout - pickup and distribute diskimage of experiment nodes

SYNOPSIS

pickup -h

pickup [-td] [-D *masks*] [-u *user*] [-p *passwd*] [-j *project*] [-F *file*] [-l *file*] [-r *ERM*] [-s *ENCD*] [-f *FNCP*] [-k *DMAN*] [-w *WolAgent*] [-I *msec*] [-P *path*] [-K *NI*] [-U *path*] [-L *path*] [-V *dev-pattern*] [-X *prefix*] [-x *prefix*] [-m] [-zYQBRMW] *host:partition*

wipeout -h

wipeout [-td] [-D *masks*] [-u *user*] [-p *passwd*] [-j *project*] [-F *file*] [-l *file*] [-r *ERM*] [-s *ENCD*] [-f *FNCP*] [-k *DMAN*] [-w *WolAgent*] [-I *msec*] [-P *path*] [-K *NI*] [-U *path*] [-L *path*] [-V *dev-pattern*] [-X *path*] [-x *path*] [-m] [-zYQBRMW] *host:partition* [*host:partition*]

DESCRIPTION

pickup uploads the diskimage of actor host to file server. **wipeout** download the diskimage to experiment nodes. Mention above, the program have a lot of options. You can apply those options via a file by -F option instead of command-line.

DISKIMAGE PATH RULE

The path of diskimage consists -X option and/or target hosts.

Following operation makes the diskimage from 3rd partition of node1 as "ftp://john:helo@ftp.dummy.com/img/node1-da0s3-20100101.gz".

```
% pickup ... -X ftp://john:helo@ftp.dummy.com/img ...
... node1:3
```

You can make replica on 3rd partion of node7 and node9 using the diskimage like following.

```
% wipeout ...
-X ftp://john:helo@ftp.dummy.com/img/node1-da0s3-20100101.gz ...
... node7:3 node9:3
```

OPTIONS

The following options are supported:

-h	print help messages.
-v	print version.
-t	enter trace mode. enable trace message printings.
-d	enter debug mode. enable debug message printings.
-D <i>mask</i>	mask debug messages.
-u <i>user</i>	set user name.
-p <i>passwd</i>	set password.
-j <i>project</i>	set project. erm and related identify experiment by user name and project.
-F <i>file</i>	set option file. the program reads the file and applies options on the file.
-l <i>file</i>	set logging file.
-r <i>ERM</i>	set RM information. the syntax of information is <i>host:port</i> .
-f <i>FNCP</i>	set FNCP information. the syntax of information is <i>host:port</i> .
-s <i>ENCD</i>	set ENCD information. the syntax of information is <i>host:port</i> . The host is who is running pickup/wipeout . However, do

	not use "localhost". Because ENCD clients like ni connect to this host. When you use "localhost", ni try to connect the ENCD port of the node who is running ni .
-k <i>DMAN</i>	set dman information. the syntax of information is <i>host:port</i> .
-w <i>WolAgent</i>	set wolagent information. the syntax of information is <i>host : port : ipaddrange</i> . <i>ipaddrange</i> requires the netmask length like 172.16.3.0/24 .
-I <i>msec</i>	set interval between WoL operations. You may worry that the electric power of hosts is turned on at the same time. In that case, you should set the interval.
-P <i>path</i>	set the path of PXE boot-loader. You should indicates absolute path in tftpd .
-K <i>NI</i>	set ni information. the syntax of information is <i>diskimage:kernel</i> . You should indicates absolute path in tftpd .
-m	enable MBR operations.
-X <i>prefix</i>	set prefix of diskimage. In pickup , it is the destination directory of uploaded file. In wipeout , it is the source path (directory and file) of downloaded file.
-x <i>prefix</i>	set prefix of MBR like -X.
-U <i>path</i>	set device-name of target device for diskimage. In default, the program generates it by device-pattern and partition-number. However this approach is weak in strange OSes and hosts. Then, the program is ready to set device name user want by -U.
-L <i>path</i>	set device-name of target device for MBR like -U.
-V <i>device-pattern</i>	set device-pattern. the syntax is <i>disk-type1:num=device-path,num=device-path;disk-type2:num=device-path</i> . pattern separator is semi-colon. pattern label is separated by colon. each partition is separated by comma. assignment between partition and device-path is separated by equal. <pre> pattern-list ::= pattern ";" pattern ... pattern ::= label ":" partition-list partition-list ::= partition "," partition ... partition ::= num "=" device-path </pre>
-z	disable compression.
-Y	probe disk size.
-Q	disable autoreboot.
-B	disable boot parameter restoring.
-R	do not connect RM.
-M	do not connect tftpd .
-W	do not connect wolagent .

MBR OPERATIONS

If you set -m, these programs apply MBR operations. **pickup** save partition and MBR to -X and -x option prefix, respectively.

EXAMPLE OF OPTION FILE

The program requires many options. You can set options by file.

```
-r localhost:1234
-k localhost:1236
-s 172.16.220.118
-f localhost:1238
-K FreeBSD/ni02.fs:recover_system/kernel_recover
-P recover_system/pxeboot-nohang
-w 172.16.1.101:5959:172.16.1.0/23
-w 172.16.3.101:5959:172.16.3.0/23
-V IDE:1=/dev/rad0s1,2=/dev/rad0s2,4=/dev/rad0s4,5=/dev/rad0s5;
-V SATA:1=/dev/rda0s1,2=/dev/rda0s2,4=/dev/rda0s4,5=/dev/rda0s5;
-V SCSI:1=/dev/rda0s1,2=/dev/rda0s2,4=/dev/rda0s4,5=/dev/rda0s5;
-V SAS:1=/dev/rda0s1,2=/dev/rda0s2,4=/dev/rda0s4,5=/dev/rda0s5;
```

SEE ALSO

dman(1), erm(1), fncp(1), kuma(1), tftpd(1)

NAME

pqerm - an implementation of experiment resource manager using PostgreSQL

SYNOPSIS

pqerm [-td] [-V] [-rC] [-p *port_no*] [-R *file*] [-U *file*] [-B *file*] [-s *sec*] -S *rule* -A *acl*

DESCRIPTION

pqerm is a resource manager. **pqerm** and *erm* are same except method of data storage and few options. **pqerm** employs PostgreSQL to store data.

OPTIONS

Most of options are same of **erm**. Here, following shows different from **erm**:

- C work as cold start (default warm start.) In **pqerm**, this option commands to recreate ownership table. It takes several ten seconds.
- r regenerate all tables from data files. You have to be carefully. All data in tables are lost. It includes dynamic updated data also. This procedure may takes several minutes.

DATA LIFECYCLE

pqerm care the table for user. If you modify the table, that modification effects **pqerm** behavior immediately.

SQL IN RULE

pqerm supports SQL query in RULE file.

SEE ALSO

erm(1)

NAME

pwmng - power manager

SYNOPSIS

pwmng -h
pwmng [-p *port_no*] [-f *file*]

DESCRIPTION

pwmng controls the power of actor hosts. It supports SNMP(NEC-MIB), WoL and IPMI.

OPTIONS

The following options are supported:

-h print help messages.
-p *port_no* set port number (default 1242.)
-f *file* set the name of configuration-file (default **pwmng.conf**.)

LIMITATIONS

pwmng does not support iLO, which is employed group-H of StarBED.

ACRONYMS

IPMI: Intelligent Platform Management Interface
WoL: Wake on LAN
iLO: integrated Lights-Out
SNMP: Simple Network Management Protocol

ENVIRONMENT VARIABLES

IPMI_USER — username for IPMI
IPMI_PASSWORD — password for IPMI

SEE ALSO

pwmng.conf(5), erm(1), snmpmine(1), wolagent(1)

NAME

sbpsh - StarBED power shell

SYNOPSIS

sbpsh -h

sbpsh [-t] [-r *file*] [-n]

DESCRIPTION

sbpsh is shell for manual operaions.

OPTIONS

The following options are supported:

-h help

-t enter trace mode.

-r *file* set run-command file. **sbpsh** reads the file before the printing of prompt.

-n disable reading of run-command file.

COMMANDS

help

trace

poweron

poweroff

reboot

setbootpxe

setdiskboot

setniboot

setpxelinux

setpxelinucfg

set

quit

ENVIRONMENT VARIABLES

KUROPASSWD

SEE ALSO

erm(1), pwm(1), swmg(1), dman(1)

NAME

snmpmine - mine for SNMP

SYNOPSIS

snmpmine -h

snmpmine [-p *port_no*] [-R *path_and_arg*] [-P *path_and_arg*]

DESCRIPTION

snmpmine is daemon for system power-off and reboot.

OPTIONS

The following options are supported:

-h	print help messages.
-p <i>port_no</i>	set port number (default SNMP (161).)
-R <i>path_and_arg</i>	set path of reboot and its argument
-P <i>path_and_arg</i>	set path of shutdown and its argument

SEE ALSO

pwmg(1), reboot(1), shutdown(1)

NAME

sns - skip node setup

SYNOPSIS

sns -h

sns [-s *server_name*]

DESCRIPTION

sns sends dummy disk operation complete message to ENCD via ERRP. The program retrieves formation of experiment from ENCD via ESQP.

Usually, that ENCD is **kuma**.

OPTIONS

The following options are supported:

-h print help messages.

-t trace mode.

-d debug mode.

-s *server_name* set ENCD and ESQP server name (default localhost.)

SEE ALSO

kuma(1)

NAME

swmg - switch manager

SYNOPSIS

```
swmg -h
swmg [-p port_no] [-f file] [-S] [-T ms]
```

DESCRIPTION

swmg controll switches according to user request via SWCP.

OPTIONS

The following options are supported:

-h	print help messages.
-p <i>port_no</i>	set port number (default 1240.)
-f <i>file</i>	set the name of configuration-file.
-S	do not access switch. it is fake mode for debugging.
-T <i>ms</i>	set the tick of switch checking.

EXAMPLE

```
rmanager ipaddr "10.9.7.31" port "4989"
user "john"
passwd "lennon"
project "imagine"
swpasswd "shone"
swtype name "swa001" type "IOS"
swtype name "swa002" type "IronWare"
```

SWTYPE LINE IN CONF-FILE

Normalize sytanx of **swtype**-line is following.

```
swtype name name type type [user user] [passwd passwd] [epasswd epasswd]
```

User and password are used in default definitions. If you need username and password to login the switch, set **user**-part and **passwd**-part. If you need password to enable the switch, set **epasswd**-part.

TYPE OF SWITCHES

The program supports Cisco 6509(**IOS**), Brocade(Foundry) BigIron (**IronWare**) and D-Link DGS-3400 (**DLDGS**) serises. Futhermore, it includes many switches supportments however they are reserved and obsoluted here because they does not verified enough.

SEE ALSO

bswc(1), swmg.conf(5), erm(1)

NAME

wolagent - WoL agent

SYNOPSIS

wolagent -h

wolagent [-p *port*] [-s *addr*] [-d *addr*]

DESCRIPTION

wolagent is a magic packet sender of WoL(Wake on LAN). In many case the network equipments can power-on by some network procedures. WoL is one of them. You should put the program per L2 network because WoL is L2 technology.

OPTIONS

The following options are supported:

- h print help messages.
- p *port* set the port of service. Sometime, several **wolagent** runs on single machine. You should setup and identify them by service port.
- s *addr* set the source address of magic packet. It is a key of interface selection, in multi-interface machine.
- d *addr* set the destination address of magic packet. default is 255.255.255.255 .

LIMITATIONS

WoL have no acknowledgment. The program cannot recognize success or not of the operation. User should check that by other apporach. User may confirm it by watching lighting the power indication light(or LED).

SEE ALSO

pwmng(1)

Part IV

Protocol Manuals

8

Experiment Resource Reservation Protocol Version 0.6 (ERRP/0.6)

This article describes ERRP (Experiment Resource Reservation Protocol) version 0.6. The protocol is used between RM (Resource Manager) and its client.

8.1 Overview

Experiment resource reservation protocol (ERRP) aims reservation and maintenance of resources for network experiments — hardware equipments (PC, switch, and others) and logical items (VLAN number, various network service and others). The protocol is a connection oriented protocol on TCP/IP, and it is designed for client/server style. Client and server communicate over the connection. Server is a program for management of resources, Resource Manager (RM). Client reserves resources of manager and controls those resources. So, client is a driving program of network experiments. We call it 'driver'.

Driver(client) and manager(server) communicate over the connection. Communication consists of pairs of request by driver and its response by manager. Driver requests retrievals of resource information, appending new resources, reservation of those resources.

By this protocol, you can do exclusive locks for resources. The driver use favorite resources for its experiment with search.

8.2 Terminology

Resource: It is most important thing both logical and physical.

Category: A category of resources. We expect 'node', 'vlan', 'switch', 'switch-port' and 'service'.

(RM) Resource Manager: A server program of ERRP.

(ED) Experiment Driver: A client program of ERRP.

(ENCD) Experiment Node Configuration Driver: same as ED.

8.3 Connections

ERRP is a connection oriented protocol. Driver(client) and manager(server) communicate over the connection. Following figure depicts client, server and connection.

```

(client) ERRP (server)
+-----+ +-----+
|driver|-----|manager|
+-----+ +-----+

```

In case of using multi-user experiment facility or experiments with multi-facilities, the experiment requires to use several managers. Managers are connected serially. Closer manager to driver is called 'lower'. the other one is called 'upper'.

```

              (lower)          (upper)
+-----+ ERRP +-----+ ERRP +-----+
|driver|-----|manager|-----|manager|
+-----+ +-----+ +-----+

```

Manager can handle multiple client. Chain of ERRP often forms a tree structure. However, each client runs independently. Client does not care other clients.

```

+-----+ ERRP +-----+
|driver1|-----|manager|
+-----+ +-----+
|ERRP
+-----+ ERRP +-----+ |
|driver2|-----|manager|---+
+-----+ +-----+

```

8.4 Name Space

Since resources are stored in multiple managers, the way to identify resource is required. According to aceant manner of e-mail and netnews, "!" separates resource and manager.

```

foo!A      resource 'A' on upper manager 'foo'.
C          if the manager has private resource 'C', it express this.
           otherwise, it express the resource of upper manager.

```

Manager applies resource name resolving from closer to further. At first, the manager scan the resource into the resource database itselfs. When the resources are found, the manager handle it. Otherwise (not found), the manager bypasses such command tu upper manager.

For example, from the view point of client 'apple' in below figure, left IDs in following samples mean right resources.

```

x      => bar!x
!x     => bar!x
y      => bar!y

```

From that of 'orange', "!" effects to identify managers.

```

x      => foo!x
!x     => foo!x
y      => bar!y
bar!x => bar!x

```

```

driver          manager
+-----+ ERRP +-----+
|apple|-----|bar|
+-----+ ERRP+-----+
|
driver          manager |
+-----+ ERRP +-----+ |
|orange|-----|foo|---+ |
+-----+ +-----+ |
|
x

```

8.5 Resource Category

Resource manager have to handle various type resources. Then, the program handles using category.

category	description
cat	categories
conn	connection for upper managers
node	node; PC, PDA and others
switch	network switch (or router)
swport	network switch port
vlan	VLAN-ID; the number for VLAN
service	services; WoL and others

8.6 Property of Resource

Except following several literals, ERRP related program can use any literals as property of resources. Following literals are reserved properties:

name	name of the resource; stores string
owner	ownership of the resource; stores user and project
user	user of the resource; stores user and project
state	state of resource; depend on category

8.6.1 Node Properties

Currently, following properties are discussed and appeared. Most of them are hardware specification.

required	owner name state	ownership; stores user and project name; stores string state. see Section 8.7
recommended	if-N n_if	N-th network interface the number of network interfaces
optional	helth power n_mc mc-N bootdisk n_cpu cpu-N memsize	methods for helth check; IPMI, ICMP, SNMP and others methods for power control; IPMI, WoL, SNMP and others the number of management card N-th management card; IPMI, iLO and others bootdisk; IDE, SCSI and others the number of CPU N-th CPU the size of memory

For example, some manager holds following properties for node.

```
info node sintclf001
201 Okay
name: sintclf001
diskhint: IDE
bootdisk: IDE
Helth: ICMP,SSH,IPMI
Power: SNMP-NECMIB,IPMI
n_if: 6
n_experiment_if: 4
if-0: type=manage media=GigabitEthernet MAC='00:14:85:38:A2:66' \
phy-port='silaswc001,1/1' IP-addr=172.16.4.1
if-1: type=empty media=GigabitEthernet MAC='00:14:85:38:A2:67' \
phy-port='' IP-addr=0.0.0.0
if-2: type=experiment media=GigabitEthernet MAC='00:0E:0C:A7:81:0E' \
phy-port='silaswc002,1/1' IP-addr=0.0.0.0
if-3: type=experiment media=GigabitEthernet MAC='00:0E:0C:A7:81:0F' \
phy-port='silaswc002,2/1' IP-addr=0.0.0.0
if-4: type=experiment media=GigabitEthernet MAC='00:0E:0C:85:BE:00' \
phy-port='silaswc002,3/1' IP-addr=0.0.0.0
if-5: type=experiment media=GigabitEthernet MAC='00:0E:0C:85:BE:01' \
phy-port='silaswc002,4/1' IP-addr=0.0.0.0
n_mc: 1
mc-0: type=IPMI conn=Override MAC='' phy-port='' IP-addr=172.16.4.1
use: 1
owner: undef
user: undef
state: pooled
.
```

Moreover, location, country, price and other things are expected as properties. This framework allows adding of such properties when programs want to use them.

8.7 Node State

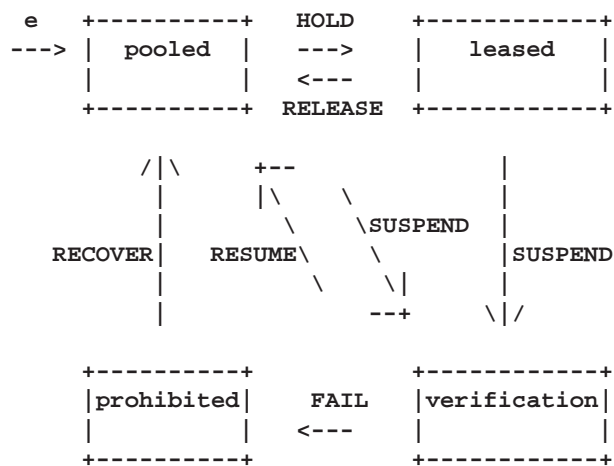
RM sets node's state according to client's request. Following 4 states are defined.

- pooled (idle, free, health)
- leased (reserved)
- verification (maybe sick)
- prohibited (dead)

'leased' and 'pooled' means user occupied or not. Since node often broken, administrator have to verify its health. 'verification' means 'wait to administrator check' and/or unreliable. When administrator authorize the node to broken, the state of the node changes 'prohibited'. Otherwise, administrator shifts it to 'pooled'.

Following figure depicts transition of these states with client commands as input.

Note `RESUME`, `FAIL` and `RECOVER` commands are available for only administrator. See Section 8.11.



8.8 Communication Syntax

8.8.1 Generic Rule

```

<string>    ::= [_A-Za-z0-9]+
<abstring>  ::= [_A-Za-z][_A-Za-z0-9]*
<number>    ::= [0-9]+
<white>     ::= (" " | "\t")+
<nl>        ::= "\n" | "\r\n"

<name>      ::= <abstring>
<id>        ::= <name> | <number> | <name> "!" <name>
<id-list>   ::= <id-list> "," <id> | <id>
<host>      ::= <FQDN>|<IP-addr>
<FQDN>      ::= [A-Za-z0-9\.] +
<IP-addr>   ::= [1-2]*[0-9]*[0-9]+\.[1-2]*[0-9]*[0-9]+\.[1-2]*[0-9]*[0-9]+\.[1-2]*[0-9]*[0-9]+
<port>      ::= <number>
<cat>       ::= "service" | "node" | "vlan"
              | "switch" | "swport" | "conn" | "cat"

```

8.8.2 Request

```

<request>   ::= <command> <arg-list>
<arg-list>  ::= <arg-list> <white> <argument> | <argument>
<argument>  ::= <abstring> "=" <string>
              | <abstring> | <number>

```

8.8.3 Response

Response consists of response-code by 3 digits and response-body.

```

<response> ::= <even-response-code> <arguments> <nl>
              | <odd-response-code> <string> <nl> <prop-chunk> "." <nl>

```

When response-code is even number, response forms multi lines. And it terminates a line which start by period (.). Otherwise, odd number, the response is single line. Detail of responses appeared other sections.

8.9 Commands

Most commands require login procedure with USER and PASSWD. The procedure aims avoiding conflicts and collision of resource usage. Moreover, to identify experiment, PROJECT command is ready. Client must issue PROJECT after login (USER and PASSWD).

The number of commands that not required login are few. HELP is prepared for usefulness in manual operation. VER and SYST is useful for checking manager ability.

Adminitrator can issue RESUME, FAIL, RECOVER and PERMITRANGE .

Following table shows a summary of them.

command	p	a	description
VER			version of protocol
SYST			system version
HELP			print commands
NOP			no operation
USER			user
PASSWD			password for user
PROJECT	X		project
QUIT			leave
CONNECT	X		connect upper manager
LIST	X		list of entities
INFO	X		information, list properties of entity
DISLIKE	X		mark as dislike resource
LIKE	X		mark as like resource; unmark as dislike resource
FIND	X		find entities
HOLD	X		hold entities (be leased)
FINDHOLD	X		find and hold entities (for atomic)
RELEASE	X		release entities (be pooled)
SUSPEND	X		enter verification from pooled or leased
RESUME	X	X	return pooled from verification
FAIL	X	X	enter prohibited from verification
RECOVER	X	X	return pooled from prohibited
REGIST	X		register new resource
RANGE	X		choice range
PERMITRANGE	X	X	assign range v.s. users
PERMITCHECK	X	X	check permission

8.9.1 Generic Information

```

VER
  100 ERRP/0.6

SYST
  100 ERM/0.7

HELP
  201 OK
  <description>
  .

NOP
  100 NOP

  <description> ::= <description> <desc-line> | <desc-line>
  <desc-line>   ::= [^\.].* <nl>

```

Description is a chunk of line which consist of any charactors expect period starts. Because period indicates the end of response.

8.9.2 Login

```

USER <user>
  200 OK
  210 please send password

PASSWD <passwd>
  200 OK
  440 auth error

PROJECT <project>
  200 OK
  440 ignore project

  <user>      ::= <abstring>
  <passwd>    ::= <string>
  <project>   ::= <abstring>

```

8.9.3 Leave

```

QUIT
  100 OK

```

Manager closes the connection without response. After response manager may disconnect immediately. Some client program could not receive response.

Note QUIT does not means resource release. Because resource reservation is indepent from connection. Thus, drivers have to release those resources by RELEASE command.

8.9.4 Connect Upper Manager

```

CONNECT <manager-name> <host:port>

  <manager-name> ::= <abstring>

```

8.9.5 Retrieval of Resource Information

LIST <cat> <state>

LIST project

201 OK

node1

...

nodeN

.

400 ignore

INFO <cat> <id>

201 Ok

<prop-chunk>

.

400 not found/ignore

```
<prop-chunk> ::= <prop-chunk> <prop-line>
<prop-line>  ::= <abstring> ":" <white> <prop-v-list> <nl>
<prop-v-list> ::= <prop-v-list> <white> <prop-v>
                | <prop-v-list> "," <prop-v>
                | <prop-v>
<prop-v> ::= <string>
```

8.9.6 Find Resources

FIND <cat> <conds>

201 Ok

node1

...

nodeN

.

400 ignore request

410 no idle node

414 not enough idle nodes

420 no matched node

'conds' means conditions. The number of require node was specified like 'num=13'.

It means single node that you not specified the number.

The condition about network interface is little complex. If you require fastethernet, you have to use 'if[media=fastethernet]'. If you don't care media type, only said 'if'.

```
<conds> ::= <conds> "," <cond> | <cond>
<cond>  ::= "num" "=" <number>
                | "bootdisk" "=" ("IDE" | "SCSI")
                | "if"
                | "if" "[" <ifqlist> "]"

<ifqlist> ::= <ifqlist> "," <ifq> | <ifq>
<ifq>     ::= "media" "=" <string>
                | "type" "=" <string>
```

Example of FIND

```
find node num=4
```

— want any 4 nodes

```
find node num=9,if[media=fastethernet,type=experiment]
```

— want 9 nodes with 1 interface for experiment network

```
find node num=7,if[media=atm],if[media=fastethernet],if
```

— want 7 nodes with 3 interface(atm, fastethernet and any).

```
find vlan num=4
```

— want any4 VLANs

Dislike/like

Sometime, clients have dislike resources in some reason. The client can regist dislike resources by DISLIKE. The server should skip these resources when it replys in FIND or FINDHOLD.

```
DISLIKE <cat> <id>
200 Ok
210 Sure
```

```
LIKE <cat> <id>
200 Ok
210 Sure
```

Using LIKE, the client can cancel the effects of DISLIKE.

Algorithm

FIND decides response according to following.

```
clear(cand-list)
foreach r <permitted resources> {
    if(<r is dislike>) {
        continue
    }
    if(<r is leased>) {
        continue
    }
    if(<r match conditions>) {
        add(cand-list, r)
    }
}
reply(cand-list)
```

8.9.7 State Transition

HOLD <cat> <id-list>

200 Ok

210 Ok some items are holded already

400 ignore

401 ignore

node1

...

nodeN

.

410 busy

420 reserved

FINDHOLD <cat> <conds>

201 Ok

node1

...

nodeN

.

RELEASE <cat> <id-list>

200 Ok

210 Ok some items are released already

400 ignore

401 ignore

node1

...

nodeN

.

410 free entity

420 not permitted, ownership mismatch

Release all resources for this project.

RELEASE project

200 OK

400 ignore

Release all resources for this user.

RELEASE user

200 OK

400 ignore

```

SUSPEND <cat> <id>
    200 OK, to be verification
    410 unknown node
    420 that node is not 'pooled'
    430 you are not permitted

RESUME <cat> <id>
    200 OK, to be pooled
    410 unknown node
    420 that node is not 'verificationed'
    430 you are not permitted

FAIL <cat> <id>
    200 OK, to be prohibited
    410 unknown node
    420 that node is not 'verificationed'
    430 you are not permitted

RECOVER <cat> <id>
    200 OK, to be pooled
    410 unknown node
    420 that node is not 'prohibited'
    430 you are not permitted

```

8.9.8 Regist New Resource

```

REGIST <cat> <id>
<prop-chunk>
.
    200 Ok
    400 ignore properties
    402 broken request
        timeout or does not terminated
    410 ignore id

```

8.9.9 Access Control

```

RANGE <cat> <id-list>
    200 OK
    400 ignore
    401 unknown node
    node1
    ...
    nodeN
.

PERMITRANGE <cat> <user> <id-list>
    200 OK
    400 ignore
    410 ignore user
    401 unknown node
    node1
    ...
    nodeN
.

```

8.10 Sample of Command Sequence

8.10.1 Using Several Nodes

```

SYST
VER
USER foo
PASSED bar
PROJECT hawaii
LIST node pooled
INFO node n13
HOLD node n7,n43,n12,n3
FIND node num=4,if[media=fastethernet],if
{do experiment(s)}
RELEASE node n7,n43,n12,n3
QUIT

```

Use node-‘n54’ according to manager’s oracle.

```

{login}
FINDHOLD node num=4,if[media=fastethernet],if
{use n54 as any role}
RELEASE node n54
QUIT

```

8.10.2 Releasing When You Meet Unexpected ENCD Trouble

RM is daemon program. So, the program must run anytime. If the contents of the program is broken, the program will restart. However, clients (almost ENCD) disconnect suddenly in rare case. In such case, RM is not wrong but the consistency of contents is broken. Since the client is down, user do not know what nodes are reserved to last project. Then, a releasing method without nodename is required.

```

{connnect}
USER foo
PASSWD bar
PROJECT hawaii
HOLD node n3
{disonnnect peer}

{re-connect}
USER foo
PASSWD bar
PROJECT hawaii
RELEASE project
{run experiment, again}
HOLD node n3

```

8.11 Administrator

User ‘admin’ is a special user. He/she can issues special commands, like following:

```

RESUME
FAIL
RECOVER

PERMITRANGE

```

8.12 Literals

8.12.1 Interface Media

Following terms are defined for query.

<code>ATM</code>	<code>ATM; asynchronous transfer mode</code>
<code>Ethernet</code>	<code>Ethernet (10Mbps)</code>
<code>FastEthernet</code>	<code>Fast Ethernet (100Mbps)</code>
<code>GigabitEthernet</code>	<code>Gigabit Ethernet (1Gbps)</code>
<code>10GigabitEthernet</code>	<code>10 Gigabit Ethernet (10Gbps)</code>

FastEthernet means 100Mbps Ethernet. It does not care detail (e.g., 100BASE-TX, 100VG-ANYLAN and others).

8.12.2 Interface Type

Following terms are defined for query.

<code>manage</code>	<code>management network (probably 1 per host)</code>
<code>experiment</code>	<code>experiment network</code>
<code>empty</code>	<code>open circuit</code>

History

ERRP was called SBRP (StarBED Resource Protocol) previously. Design and implementation of SBRP starts October 2003 by K. Chinen.

SBRP/0.1	early of Oct 2003
SBRP/0.2	Oct 14, 2003
SBRP/0.3	unknown
SBRP/0.4	Dec 2003
SBRP/0.5	Oct 2004

We change name SBRP to ERRP because the protocol has no restriction for StarBED. Most network testbeds can use that.

ERRP/0.6	just working
----------	--------------

Changing from ERRP/0.5

- Registration of resource is available.
- Introduce the chain of managers and its name space.
- Introduce the concept of resource categories.
 - adding new category 'service', 'switch' and 'swport'.
- Merge the name of node states to recent papers.
 - 'verification' and 'prohibited'.
- Add properties 'n_cpu' and 'cpu-N'
- Add response-code 210 as positive ack for 'HOLD' and 'RELEASE'
- Add command 'DISLIKE' and 'LIKE'
- Add response-code 210 as positive ack for 'DISLIKE' and 'LIKE'
- Arrange response codes of REGIST

Changing from SBRP/0.4

added command:

```

USER
PASSWD
PROJECT
SUSPEND
RESUME
FAIL
RECOVER
LIST PROJECT
RELEASE PROJECT
RELEASE USER
RANGE
PERMITERANGE

```

removed command:

```

JOIN

```

People

Current ERRP is a result of corrabolation of following people:

Ken-ichi Chinen*
Toshiyuki Miyachi
Makoto Misumi
Naoki Isozaki*
Shinsuke Miwa

Persons listed with star, joined the making of the reference implementation, **sbrm** and **erm**.

9

Experiment Structure Query Protocol Version 0.5 (ESQP/0.5)

9.1 Overview

ESQP is text and line-oriented protocol with TCP. This protocol is used Experiment Node Configuration Driver (ENCD). The client of this protocol can recognize entities of experiment on ENCD. Some client implementation draw topology of the experiment.

Line of protocol is terminated CR and LF. Response have 3 digit as response code. Port number is 3458.

Furthermore, early versions of ESQP are called SSQP.

9.2 Commands

To do health-check, ESQP have commands for resource list retrieval.

Command	description
SYST	print system version
VER	print protocol version
HELP	print help message
QUIT	quit
NODESETLIST	print list of node-set
NODELIST	print list of node
NODEINFO	print information of node
NETSETLIST	print list of net-set
NETLIST	print list of net
NETINFO	print information of network

9.2.1 SYST

request

```
SYST
```

response

```
100 kusa/1.0
```

9.2.2 VER

request

```
VER
```

response

```
100 ESQP/0.4
```

9.2.3 HELP

request

```
HELP
```

response

```
101 OK
  human readable message
  ...
.
```

9.2.4 QUIT

request

```
QUIT
```

no response.

9.2.5 NODESETLIST

request

```
NODESETLIST [simnode]
```

Prints all nodes when sim-node is not specified.

response(success)

```
201 <N> OK
<simnode1> <stat1> <w1> <m1> <physnode1-1> <physnode1-2> ... <physnode1-m>
<simnode2> <stat2> <w2> <m2> <physnode2-1> <physnode2-2> ... <physnode2-m>
...
<simnodeN> <statN> <wN> <mN> <physnodeN-1> <physnodeN-2> ... <physnodeN-m>
.
```

<stat> indicate stat of node. S indicates that it is stable. U indicates un-stable.

When no resource, print empty list with response code 201.

example:

```
201 2 OK
sv U 1 2 sintclc001 sintclc002
cl S 1 1 sintclc003
.
```

response(empty)

```
201 0 OK
.
```

9.2.6 NODELIST

request

```
NODELIST [simnode]
```

response(success)

```

201 <N> OK
<simnode1> <stat1> <physnode1-1>
<simnode2> <stat2> <physnode2-1>
...
<simnodeN> <statN> <physnodeN-1>
.

```

example:

```

201 4 OK
sv-0 U
sv-1 S sintcla003
cl-0 S sintclb003
cl-1 S sintclb004
.

```

response(empty)

```

201 0 OK
.

```

9.2.7 NODEINFO

print information of nodes

request

```
NODEINFO <simnode>
```

response(success)

```

201 OK
name <name>
resourcename <resourcename>
netif <attributes-1>
netif <attributes-2>
...
netif <attributes-N>
.

```

response(error; not found)

```
400 <simnode> NG not found
```

example:

```
nodeinfo client-0
```

response

```

200 OK
name client-0
state SECONDBOOT
resourcename sintclb049
agent host=sintclb049 ipaddr=172.16.1.49 portnum=2345
netif type=2 media=fastethernet ipaddr=192.168.3.1/24 macaddr=00:00:4C:4F:A1:D5
.

```

response

```
400 simnode[0] NG not found
```

response(no bounding)

```
201 OK
name client-0
resourcename
state INIT
agent host= ipaddr= portnum=
netif type=0 media=fastethernet ipaddr= macaddr=
netif type=0 media=fastethernet ipaddr= macaddr=
netif type=0 media=fastethernet ipaddr= macaddr=
netif type=0 media=fastethernet ipaddr= macaddr=
.
```

9.2.8 NETSETLIST

request

```
NETSETLIST
```

response

```
200 <N> Okay
<name1> <status1> <m1>
<name2> <status2> <m2>
...
<nameN> <statusN> <mN>
.
```

example

```
200 1 Okay
ethnet S 2
.
```

9.2.9 NETLIST

request

```
NETLIST
```

response

```
200 <N> Okay
<name1> <status1> <vlannum2>
<name2> <status2> <vlannum2>
...
<nameN> <statusN> <vlannumN>
.
```

example:

```
200 2 Okay
ethnet-0 U 0
ethnet-1 S 133
.
```

9.2.10 NETINFO

print information of network.
request

```
NETINFO <name>
```

response

```
201 Okay
name <name>
resourcename <resourcename>
media <media>
ipaddrange <ipaddrange>
members <mem1> <mem2> ...
.
```

example

```
201 OK
name ethnet-0
resourcename vlan0804
media fastethernet
ipaddrange 192.168.3.0/24
members server-0-0 client-0-0
.
```

9.2.11 Others

The server of this protocol replies 500 when command is unknown or un-implemented.

```
500 not implemented yet
```

or

```
500 ignore command
```

9.3 Name of Entities

Basically, names used ESQP are logical.

9.4 See Also

[1] ERRP/0.6

10

Switch Configuration Protocol Version 1.2 (SWCP/1.2)

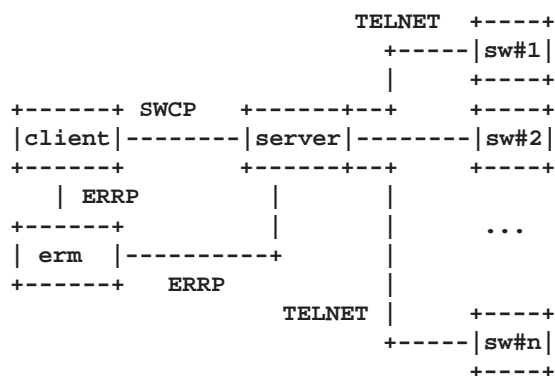
10.1 Overview

SWCP is a protocol for switch manager (SWMG). SWMG aims to configure network switches. So, the client of SWCP expects to setup network via SWMG.

SWCP have some characteristics following:

- TCP with port#1240
- text-base and line oriented protocol
- line is terminated by CR(0x0d) LF(0x0a)

SWMG identifies sessions by user, password and project. This identification according to resource manager (ERM) via Experiment Resource Reservation Protocol (ERRP).



10.2 Commands

command	description
HELP	print help message
SYST	print system version
VER	print protocol version
NOP	no operation
QUIT	quit
USER	set user
PASSWD	set password
PROJ	set project
MKVLAN	make VLAN
RMVLAN	remove VLAN
JOINVLAN	join switch ports to VLAN
LEAVEVLAN	leave switch ports from VLAN
ACTIVATEPORT	activate switch ports
DEACTIVATEPORT	deactivate switch ports
BINDPORT	bind switch port to VLAN
UNBINDPORT	unbind switch port to VLAN
ISLCARE	change the mode of ISL handling

When command is not found, server replies 400.

10.2.1 SYST

SYST

100 SWMG/1.0

10.2.2 VER

VER

100 SWCP/1.0

10.2.3 QUIT

QUIT

no response.

10.2.4 USER

USER <user>

200 Okay

10.2.5 PASSWD

PASSWD <passwd>

200 Okay

400 Ignore password

10.2.6 PROJ

PROJ <proj>

200 Okay

10.2.7 MKVLAN

MKVLAN <VLAN-ID>

```
200 Okay
400 empty user/project
420 ignore VLAN
422 you are not permitted to the VLAN
```

10.2.8 RMVLAN

RMVLAN <VLAN-ID>

```
200 Okay
400 empty user/project
420 ignore VLAN
422 you are not permitted to the VLAN
```

10.2.9 JOINVLAN

JOINVLAN <VLAN-ID> [<swport-1>[,<swport-2> ...]]

```
200 done
400 empty user/project
420 ignore VLAN
422 you are not permitted to the VLAN
412 you are not permitted to these swports
410 not found switch
414 not found swports
```

If target switch port already joined some VLAN. SWCP server have to care their combinations. If the switch port joined tagged VLAN, server process this join. If the switch port joined untagged VLAN, server have to care one more condition. If the new VLAN is untagged VLAN, server must process leaving the port from old VLAN before joint of new VLAN. Following table shows its rule. T means tagged VLAN. U means untagged VLAN.

old	new	after
T1	T2	T1,T2
T1	U1	T1,U1
U1	U2	U2
U1	T1	U1,T1

Case of 3 and more VLAN are similar, if the switch port was joined to untagged VLAN and requested new untagged VLAN, the old untagged VLAN have to leave before new untagged VLAN joint. SWCP allows only 1 untagged VLAN for each switch port.

old	new	after
T1,T2,...	Tx	T1,T2,...,Tx
T1,T2,...	U	T1,T2,...,U
T1,T2,...,U	Tx	T1,T2,...,U,Tx
T1,T2,...,U1	U2	T1,T2,...,U2

10.2.10 LEAVEVLAN

LEAVEVLAN <VLAN-ID> [<swport-1>[,<swport-2> ...]]

```

200 done
400 empty user/project
420 ignore VLAN
422 you are not permitted to the VLAN
412 you are not permitted to these swports
410 not found switch
414 not found swports

```

10.2.11 ACTIVATEPORT

ACTIVATEPORT <swport-1>[,<swport-2> ...]

```

200 done
400 empty user/project
412 you are not permitted to these swports

```

10.2.12 DEACTIVATEPORT

DEACTIVATEPORT <swport-1>[,<swport-2> ...]

```

200 done
400 empty user/project
412 you are not permitted to these swports

```

10.2.13 BINDPORT

BINDPORT <swport> <vlan1>[,<vlan2>...]

```

200 okay
400 empty
412 you are not permitted to these swports
410 not found switch
414 not found swports

```

10.2.14 UNBINDPORT

UNBINDPORT <swport> <vlan1>[,<vlan2>...]

UNBINDPORT <swport> all

```

200 okay
400 empty
412 you are not permitted to these swports
410 not found switch
414 not found swports

```

Literal *all* means every VLANs which the switch port are joined. It is useful in clean-up.

10.2.15 NOP

NOP

```

100 okay

```

10.2.16 HELP

HELP

```

101 okay
SYST
VER
<messages>
.

```

10.2.17 ISLCARE

ISLCARE changes mode of ISL.

ISLCARE <NONE>|<FIRST>|<SHORTEST>|<LONGEST>

```
200 okay
210 already setted
430 ignore mode
```

10.3 Identifier of Switch Ports

Identifier of switch ports consists switch-name, slot-port number and media. Media is optional. This is normal format.

`<name>:<slot>/<port>[:media]`

Some switches and routers used :(colon) as slot-port separator. In such case, swmg have to convert normal format to local format.

10.4 Tagged VLAN Expression

Version 1.2 and later must handle tagged VLAN. For JOINVLAN and LEAVEVLAN, tagged VLANs are expressed with switch ports like following:

`tagged(silaswa01:3/4)`

For BINDPORT, tagged VLANs are expressed with VLAN ID.

`tagged(34)`

10.5 Example

Following sequence depicts a communication of client and server of SWCP. "C" means client. "S" means server.

```
C: <connect server>
C: VER
S: 100 SWCP/1.2
C: SYST
S: 100 SWMG/1.2
C: USER jack
S: 200 Okay
C: PASSWD abrakatabura
S: 200 Okay
C: PROJ trial
S: 200 Okay
C: ACTIVATEPORT swa:3/4,swb:4/3,swc:7/48
S: 200 Okay
C: JOINVLAN 123 swa:3/4,swb:4/3,swc:7/48
S: 200 Okay
C: JOINVLAN 890 tagged(sw:9/10),tagged(sw:9/7)
S: 200 Okay
C: UNBINDPORT sw:9/10 890
S: 200 Okay
C: QUIT
S: <disconnect client>
```

10.6 See Also

- [1] SWMG
- [2] ERRP/0.6
- [3] ERM
- [4] K Language Reference Manual

10.7 Hisotory

[2008/04] specification of 1.0 is written

[2009/01/28] specification of 1.1 is written

- unify combinations between message and response-code
- define LEAVEVLAN's response
- define RMVLAN-command
- define RMVLAN's response

[2009/10/01] specification of 1.2 is written

- define BINDPORT
- define UNBINDPORT

[2010/01] add new command ISLCARE

11

Power Configuration Protocol Version 0.1 (PWCP/0.1)

11.1 Overview

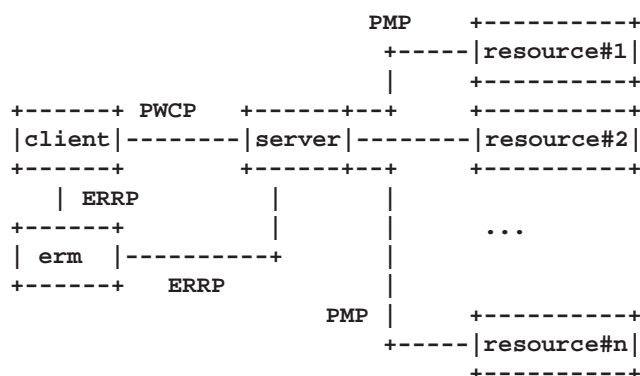
PWCP is a protocol for power manager (PWMG). PWMG aims to configure power of resources. So, the client of PWCP expects to power-on/off or shutdown resources via PWMG.

PWCP have some characteristics following:

- TCP with port#1242
- text-base protocol
- line oriented protocol
- line is terminated by CR(0x0d) LF(0x0a)

PWMG identifies sessions by user, password and project. This identification according to resource manager (ERM) via Experiment Resource Reservation Protocol (ERRP).

Most resources have protocol(s) to power management(PMP.) PWMG configure power cycle with these protocols. Currently, IPMI, SMASH and SNMP are expected.



If server is specified unknown resource (basically, server recognize resources ERM provided.) the server replys error message.

11.2 Command

command	description
HELP	print help message
SYST	print system version
VER	print protocol version
NOP	no operation
QUIT	quit
USER	set user
PASSWD	set password
PROJ	set project
PWON	power on
PWOFF	power off
PWFORCEOFF	power off forcibly
REBOOT	reboot

11.2.1 SYST

SYST

100 PWMG/1.0

11.2.2 VER

VER

100 PWCP/1.0

11.2.3 QUIT

QUIT

no response.

11.2.4 USER

USER <user>

200 Okay

11.2.5 PASSWD

PASSWD <passwd>

200 Okay 400 Ignore password

11.2.6 PROJ

PROJ <proj>

200 Okay

11.2.7 PWON

PWON <resource-name>

200 Okay 400 empty user/project 410 unknown resource 422 you are not permitted 500 unknown error 520 back-end server not defined 522 back-end server not found 530 back-end server busy or down

11.2.8 PWOFF

PWOFF <resource-name>

200 Okay 400 empty user/project 410 unknown resource 422 you are not permitted 500 unknown error 520 back-end server not defined 522 back-end server not found 530 back-end server busy or down

11.2.9 PWFORCEOFF

PWFORCEOFF <resource-name>

200 Okay 400 empty user/project 410 unknown resource 422 you are not permitted 500 unknown error 520 back-end server not defined 522 back-end server not found 530 back-end server busy or down

11.2.10 REBOOT

REBOOT <resource-name>

200 Okay 400 empty user/project 410 unknown resource 422 you are not permitted 500 unknown error 520 back-end server not defined 522 back-end server not found 530 back-end server busy or down

11.2.11 NOP

NOP

100 okay

11.2.12 HELP

HELP

101 okay
SYST
VER
<messages>
.

11.3 Example

Following sequence depicts a communication of client and server of PWCP. "C" means client. "S" means server.

```
C: <connect server>
C: VER
S: 100 PWCP/0.1
C: SYST
S: 100 PWMG/1.0
C: USER jack
S: 200 Okay
C: PASSWD abrakatabura
S: 200 Okay
C: PROJ trial
S: 200 Okay
C: PWON a101
S: 200 done
```

```
C: PWOFF b022
S: 200 done
C: QUIT
S: <disconnect client>
```

11.4 See Also

- [1] PWMG
- [2] ERRP/0.6
- [3] ERM
- [4] K Language Reference Manual

History

[2009 June] first draft (version 0.1)

Changes

12

Directory Manipulation Protocol Version 0.4 (DMP/0.4)

12.1 Overview

DMP is a protocol for directory manipulator (DMAN.) DMP aims generation and removal of symbolic link.

DMP have some characteristics following:

- TCP with port#1236
- text-base protocol
- line oriented protocol
- line is terminated by CR(0x0d) LF(0x0a)

12.2 Command

command	description
HELP	print help message
SYST	print system version
VER	print protocol version
NOP	no operation
QUIT	quit
STAT	state of file
SYMLINK	make symlink
RMSYMLINK	remove symlink
MKDIR	make directory
REMOVE	remove file
POST	make file

12.2.1 SYST

SYST

100 DMAN/0.3

12.2.2 VER

VER

100 DMP/0.4

12.2.3 QUIT

QUIT

no response.

12.2.4 STAT

```
STAT <file>
    201 found
    X-Mode: 0x1fff
    Type: SYMLINK
    Actual-Path: pxelinux.cfg/pxelinux.0
    Size: 23
    ATime: Mon, 24 Aug 2009 01:39:28 GMT
    MTime: Mon, 24 Aug 2009 01:39:25 GMT
    CTime: Mon, 24 Aug 2009 01:39:25 GMT
    .

    410 not found
```

12.2.5 SYMLINK

SYMLINK <FROM-file> <TO-file>

200 success 400 fail 406 ignore path 408 unknown error 414 TO file is not found
422 FROM file is not symlink 434 TO file is not regular file

12.2.6 RMSYMLINK

RMSYMLINK <file>

200 success 400 fail 406 ignore path 408 unknown error 410 file is not found 420
file is not symlink

12.2.7 MKDIR

MKDIR <dir>

200 success 400 fail mkdir 406 ignore path 440 dir already exist

12.2.8 REMOVE

REMOVE <file>

200 success 400 fail 406 ignore path 410 not found

12.2.9 POST

```
POST <file>
<contents>
.
```

200 success 406 ignore path 440 file already exist 450 too long 492 fail open 494 fail
write

12.2.10 NOP

NOP [[message] ...]

100 okay

12.2.11 HELP

HELP

```

101 okay
SYST
VER
<messages>
.
```

12.3 Response

code	description
100	okay
101	okay
200	success
201	found
400	fail
406	ignore path
408	unknown error
410	file not found
412	FROM file is not found
414	TO file is not found
420	file is not symlink
422	FROM file is not symlink
424	TO file is not symlink
434	TO file is not regular file
440	file is already exist
440	directory is already exist
450	too long content
492	fail open
494	fail write

12.4 Example

Following sequence depicts a communication of client and server of DMP. "C" means client. "S" means server.

```

C: <connect server>
C: syst
S: 100 DMAN/0.3
C: ver
S: 100 DMP/0.4
C: stat h001.pxe
S: 201 found
S: X-Mode: 0xalff
S: Type: SYMLINK
S: Actual-Path: pxelinux.cfg/pxelinux.0
S: Size: 23
S: ATime: Mon, 24 Aug 2009 01:39:34 GMT
```

```
S: MTime: Mon, 24 Aug 2009 01:39:25 GMT
S: CTime: Mon, 24 Aug 2009 01:39:25 GMT
S: .
C: mkdir foo
S: 200 success
C: mkdir foo
S: 440 dir already exist
C: post junk
C: helo
C: bye
C: .
S: 200 okay
C: symlink h002.pxe pxelinux.cfg/pxelinux.0
S: 200 success
C: rmsymlink h002.pxe
S: 200 success
C: rmsymlink h003.pxe
S: 410 file 'h003.pxe' is not found
C: quit
```

12.5 See Also

- [1] ERM
- [2] ERRP/0.6
- [3] K Language Reference Manual

History

[2004 or 2005] DMP was built. It is known as version 0.1

[2009/08/24] document version 0.3 was made

[2009/08/24] version 0.4 was built

Changes

- to be uniq response code
- to be normalize response code for even/odd rule of ERRP

13

Wake on LAN agent Protocol Version 1.0 (WOLAP/1.0)

13.1 Overview

WOLAP is a protocol for wolagent. wolagent generates WoL magic packets which kicks PCs.

WOLAP have some characteristics following:

- TCP with port#5959
- text-base protocol
- line oriented protocol
- line is terminated by CR(0x0d) LF(0x0a)

13.2 Command

command	description
HELP	print help message
SYST	print system version
VER	print protocol version
SEND	send WoL magic packet
mac-addr	send WoL magic packet
QUIT	quit

13.2.1 SYST

SYST

100 wolagent/2.0

13.2.2 VER

VER

100 WOLAP/1.0

13.2.3 QUIT

QUIT

no response.

13.2.4 SEND

```
SEND <mac-addr>
      200 send
      400 ignore MAC
```

```
<mac-addr>
      200 send
      400 ignore MAC
```

13.2.5 HELP

```
HELP
      101 okay
      HELP
      SYST
      VER
      <messages>
      .
```

13.3 Example

Following sequence depicts a communication of client and server of WOLAP. "C" means client. "S" means server.

```
C: <connect server>
C: VER
S: 100 wolagent/2.0
C: SYST
S: 100 WOLP/1.0
C: SEND 01:23:45:67:9a:bc
S: 200 send
C: QUIT
S: <disconnect client>
```

13.4 See Also

[1] ERRP/0.6

[2] ERM

History

[2010 Jan] first draft (version 0.1)

Changes

14

HTTP Extentions for ENCD

ENCD stands Experiment Node Config Driver. It is a category of programs. **kuma**, **pickup** and **wipeout** are included ENCD. These programs communicate via HTTP with exten-tions. Moreover, partner programs like **fncp** and **ni** use the extention also. This document describes the extention.

14.1 Overview

In the beginning, node agent (NA) does not know what configuration is required to this node. NA contacts ENCD to know that. Typically, NA and ENCD are **ni** and **pickup**. Fundamental communication is establish between NA and ENCD.

Figure 14.1 depicts these communication.

14.2 X-NSS

You can find fields "X-NSS" started in HTTP traffic. You can find also these fields in source files.

```
% grep ^X- encd.c
```

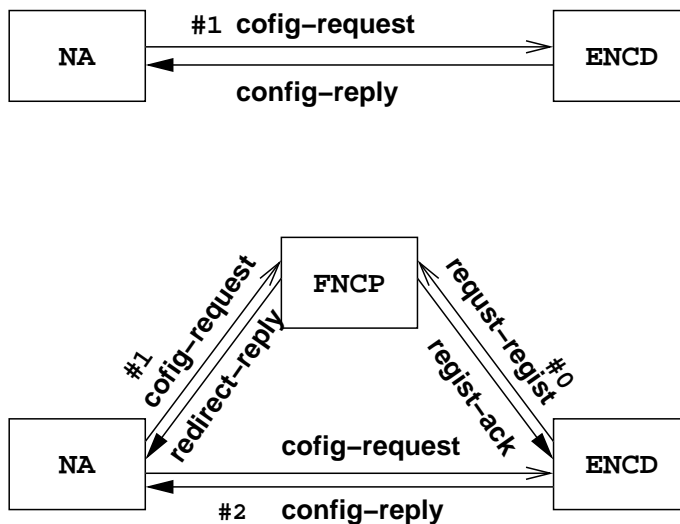


Figure 14.1: ni-ENCD communication flow

```

X-NSS-Version: $Revision: 2854 $\r\n\
X-NSS-Version: $Revision: 2854 $\r\n\
X-NSS-Driver: Kuroyuri/1.5alpha\r\n\
X-NSS-Driver-ID: %s\r\n\
X-NSS-Mission-ID: %s\r\n\
X-NSS-Node-IPAddr: %s\r\n\
X-NSS-Redirect-Dest: %s\r\n\

```

field	sender	description
X-NSS-Driver	D	driver name and version
X-NSS-Driver-ID	D	(reserved) driver ID
X-NSS-Mission-ID	D	mission ID
X-NSS-Node-IPAddr	D	node IP address
X-NSS-Redirect-Dest	D	redirect destination (ENCD)
X-NSS-Agent-ID	A	agent ID
X-NSS-Session-ID	A	session ID
X-NSS-State	A	state
X-NSS-Version	D/A	version of NSS communication protocol