SpringOS Version 1.5+ Manual

StarBED Project
2014 Autumn

This document is compiled at July 7, 2015.

SpringOS Version 1.5+ Manual
by StarBED Project

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.

Readers

This book expects three type readers 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 V.

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-v16pre-20120715.tar.gz`**) by e-mail to following address:

`info@starbed.org`

Further information, see also our WWW site:

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

License

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

III Command Reference

IV K Language Reference

V Protocol Manuals

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)
  $\Rightarrow$ 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

What SpringOS is not

- Operating system likes UNIX, WINDOWS or Mac OS X
- Distributed operating system likes Amoeba and others
- Benchmark software for some products

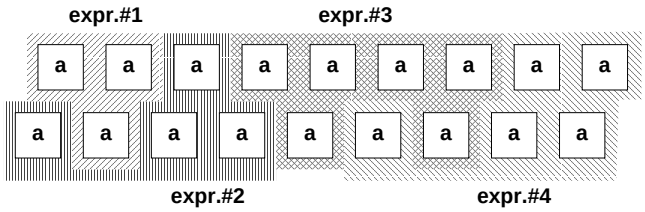


Figure 1.1: The test-bed is divided by multi experiments

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 20).

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

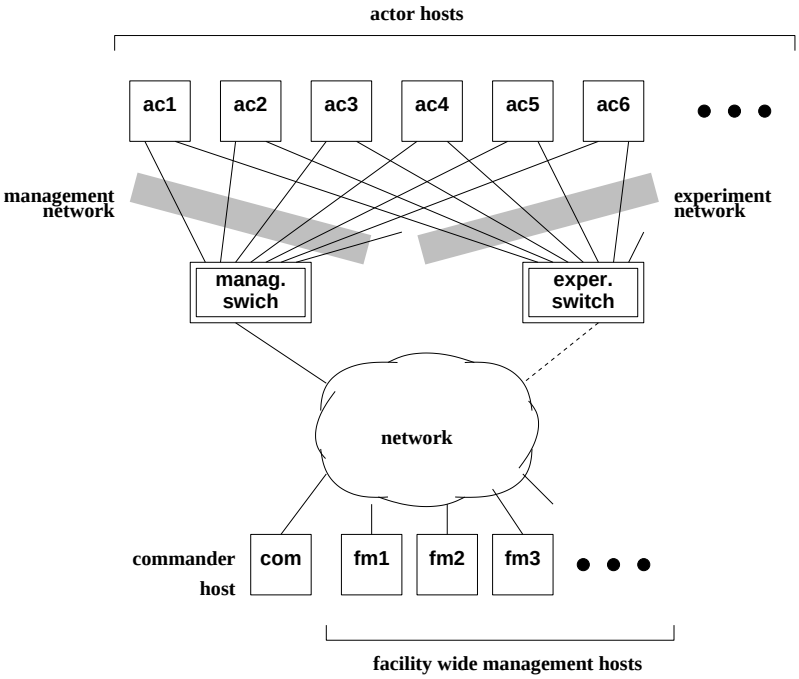


Figure 1.2: Conceptual Topology of Testbed

1.4.2 Program Categories

Programs are separated 2 region.

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

* **ni**
* **snmpmine**
* **coil**

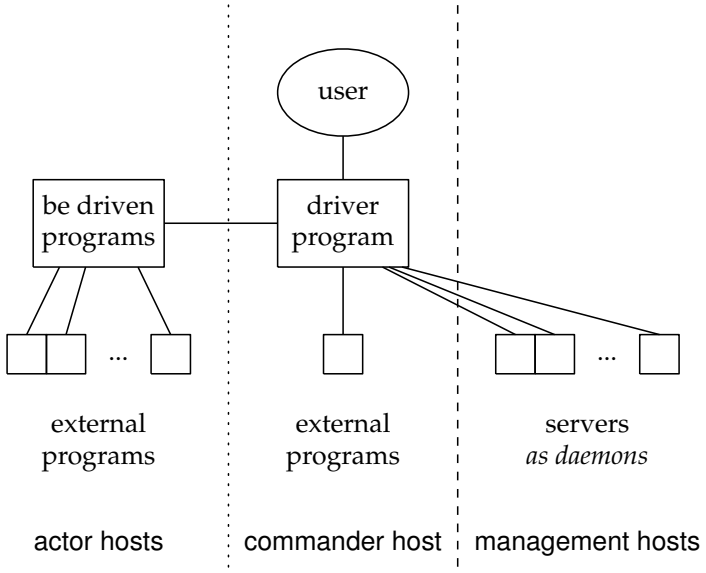


Figure 1.3: Program Rough Locations

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/simply 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 applies 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.

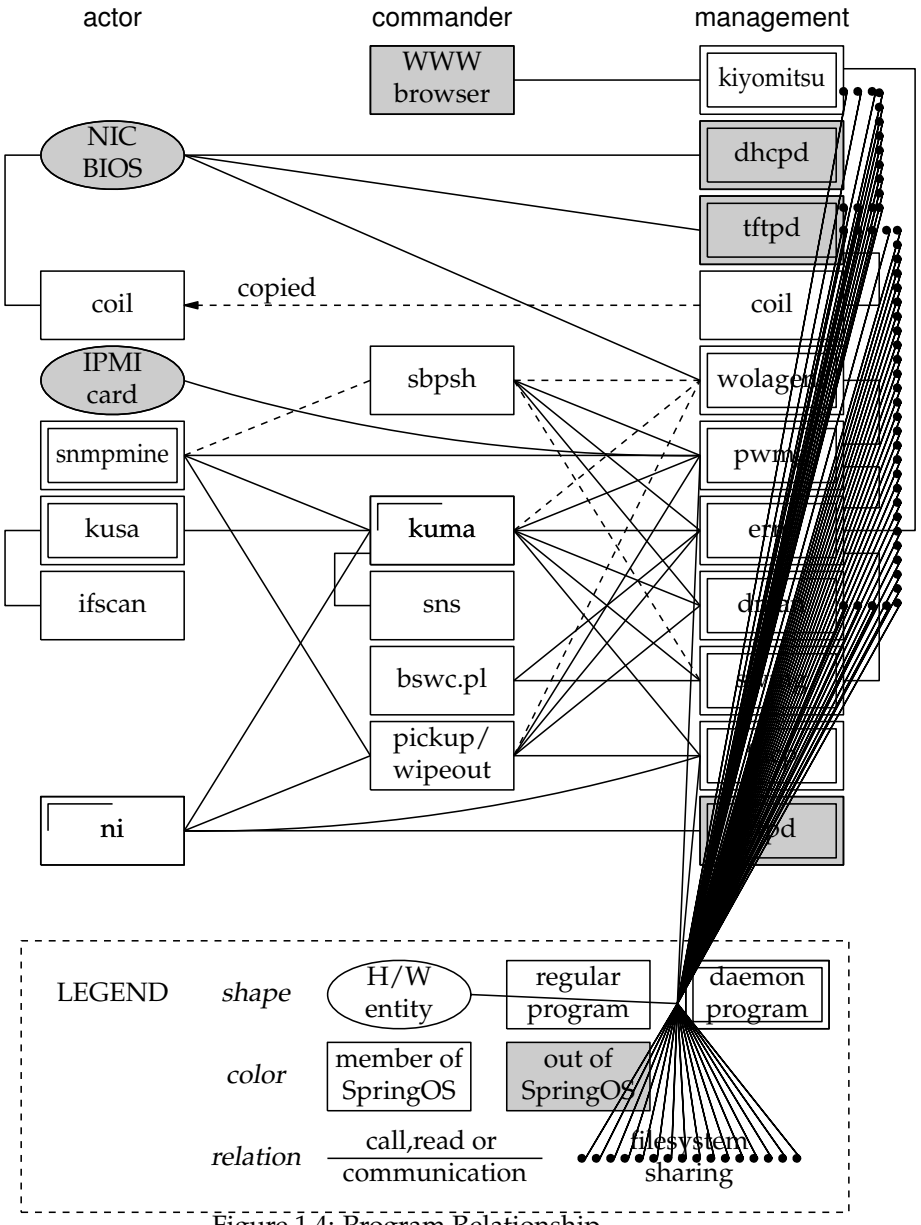


Figure 1.4: Program Relationship

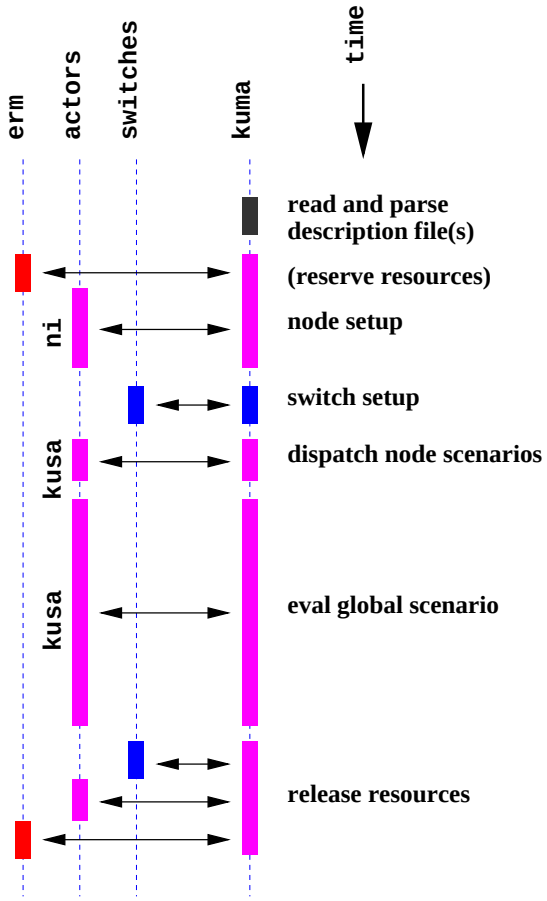


Figure 1.5: A Process flow of Automatic Driving

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**.

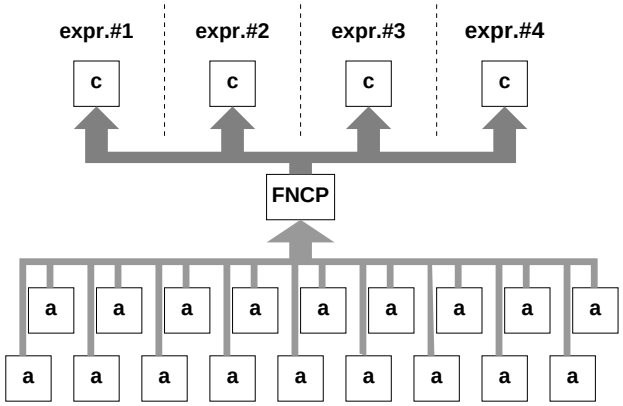


Figure 1.6: FNCP navigates actor hosts to commander host

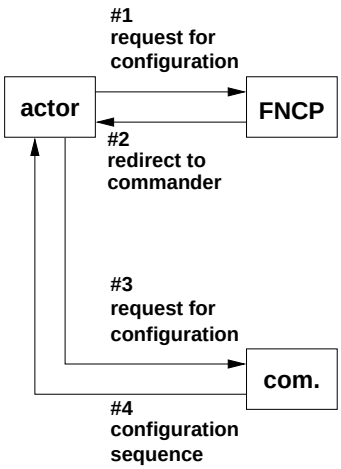


Figure 1.7: The navigation by HTTP redirection

When nodes and switches are ready, it is time to run scenarios. **kuma** sends node scenarios to **kusa**s 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.

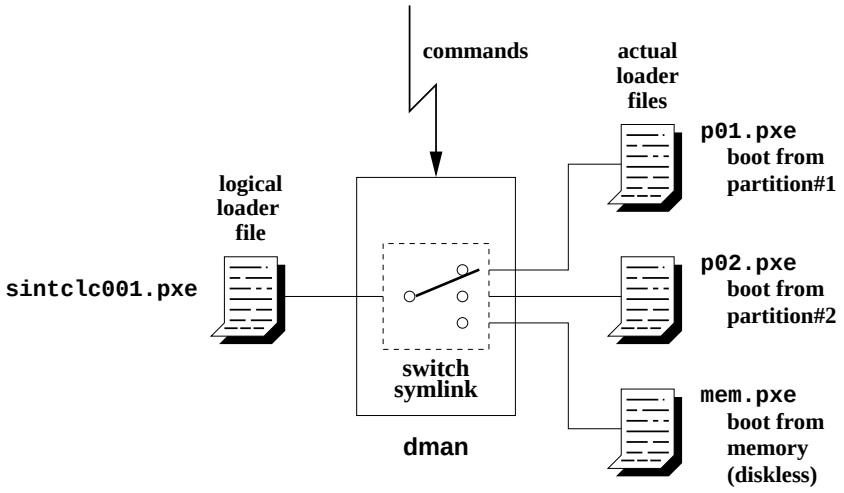


Figure 1.8: DMAN — symlink switcher

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 24), 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 25 about protocol about wolagent. PXE booting by DHCP and TFTP. After **ni** waking, OS installation is started. **ni** reboot OS itself.

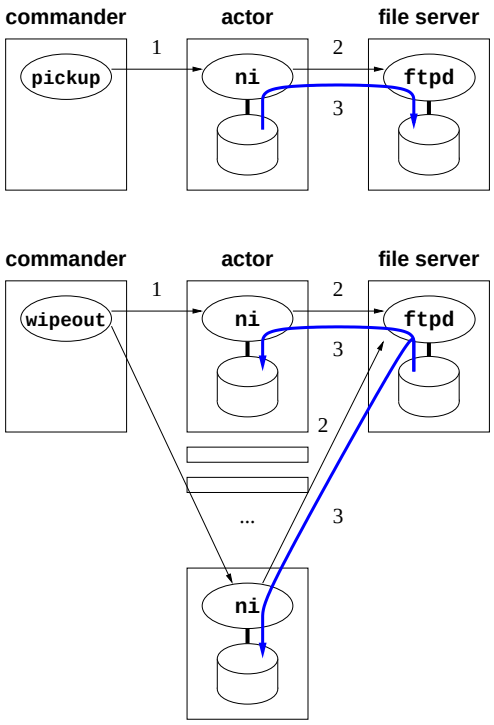


Figure 1.9: the operation flow of pickup/wipeout

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

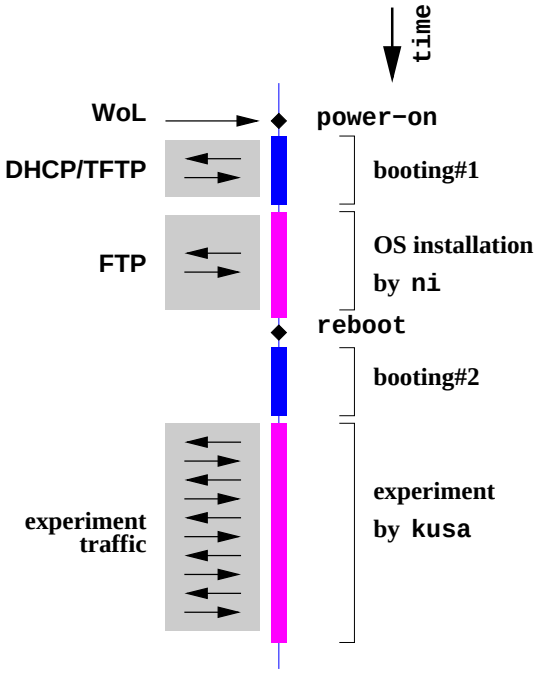


Figure 1.10: A Life of Actor Host

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 **`pqerm`**) PostgreSQL
- (for **`coil`**) nasm

In many case user may use followings in experiment. You should check their existance.

- ifconfig (or simular)

For check running of SpringOS, you may install followings.

- telnet
- tcpdump
- ping

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 it 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-16.

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:

- UNIX like OS
  Our primary developing platform is CentOS 6.3 . And we used CentOS 5.4 in several years.
- 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).

Commander host

Hardware:

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

Software:

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

> **NOTE** *Your can used 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 yourself.
- (**`dhcpd`**, **`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.
- (**`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/softwares/** 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)

Sometime, you could see one more style for other versions.

- **springos-v16pre-YYYYMMDD.tar.gz** (YYYYMMDD is date)

2.3 Structure of Source Directories

[lib] the library for SpringOS

[prog] programs

[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 zxf 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 zxf 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
```

Some OSes cannot support **libvirt**, the library for virtualization. You can compile the package without **libvirt**.

```
% ./configure --without-libvirt
```

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

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

NOTE *To run programs, you should check service ports. Because most OS protect ports in security issues. See Section 6.9.*

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

COLUMN

In CentOS 5.5 we installed followings to compile SpringOS.

for principal programs

- gcc
- make
- binutils
- flex
- bison
- perl
- zlib-devel
- ncurses-devel
- readline-devel
- openssl-devel
- libvirt-devel
- libuuid-devel

for coil

- nasm

for pqerm

- postgresql
- postgresql-server
- postgresql-devel

for demo programs

- libX11-devel
- libXt-devel
- libXaw-devel
- libICE-devel
- libSM-devel
- libXau-devel
- libXdmcp-devel
- libXmu-devel
- libXpm-devel
- xorg-x11-proto-devel
- xorg-x11-util-macros

In CentOS 5.7, Fedora 15 and others you may need to install following.

- compat-flex

A user reports us that Debian requires followings. Please check other document to installation procedures.

- build-essential
- flex
- bison
- zlib1g-dev
- ncurses-dev
- libreadline-dev

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

Table 2.2: Supported and Tested Network Switches

Maker	H/W Model	S/W	swcmd	M.net	E.net
Cisco	Catalyst 6509	IOS	IOS	-	√
	Catalyst 2960S	IOS	IOS	-	√
	Nexsus 5000	NexsusOS	NexsusOS	√	-
Brocade (Foundry)	BigIron MG8	IronWare	BigIron	-	√
	BigIron RX4, 16 and 32	BigIron	IronWare	-	√
	BigIron FCX648	IronWare	FastIron	√	√
Juniper	QFX series	-	QFabric	√	√
Extreme	X450 and X650	ExtremeXOS	√	√	
D-Link	DGS-3426, 3427 and 3450	*noname*	DLDGS	√	-

H/W: hardware.
S/W: software.
swcmd: switch command type used in **`swmg`**.
M.net: management network.
E.net: experiment network.

COLUMN

OBSOLTED KNOWN BUGS

- In SpringOS Version 1.5, MD5 related code includes bug under 64bits (x86_64) CentOS. This code is used in IPMI operations on **`pwmg`** and **`sheepdog`**. You may meet that on other 64bit OSs.

  In CentOS, you can avoid that by compiling with **`-m32`** . You should install related packages for libraries and headers. We installed **`libgcc.i686`** and **`glibc-devel.i686`** .

  Recently version SpringOS (e.g., 1.5+ or 1.6pre) use another code for MD5. You can use SpringOS without this MD5 issues.

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 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
snmport 161
snmpinterval 100
tftpdman undef
nikernel ni-kernel
nidiskimage ni-diskimage
pxeloader 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 -e 172.16.3.101:1242 \
-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
-e 172.16.3.101:1242
-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**.

Table 3.1: Options of pickup

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

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.

type	device name
IDE(ATA)	/dev/rad*D*s*P*
SCSI	/dev/rda*D*s*P*

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 22). 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
```

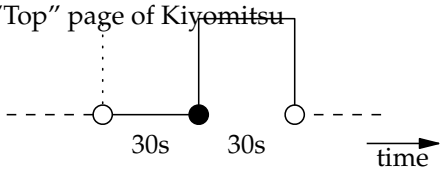
Switch ports are specified by the combination of node name and network interface number with a zero origin. "devpc2:1" means second interface of node-"devpc2". The program accepts a range expression in nodename. "devpc2-3:1" is recognized "devpc2:1" and "devpc3:1". However, the program does not accept 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) unreach within 30 seconds
2) reach within 30 seconds
3) unreach

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
```

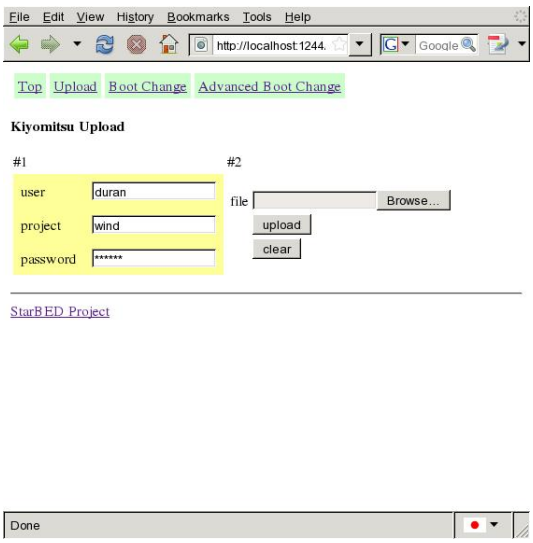


Figure 3.2: "Upload" page of Kiyomitsu

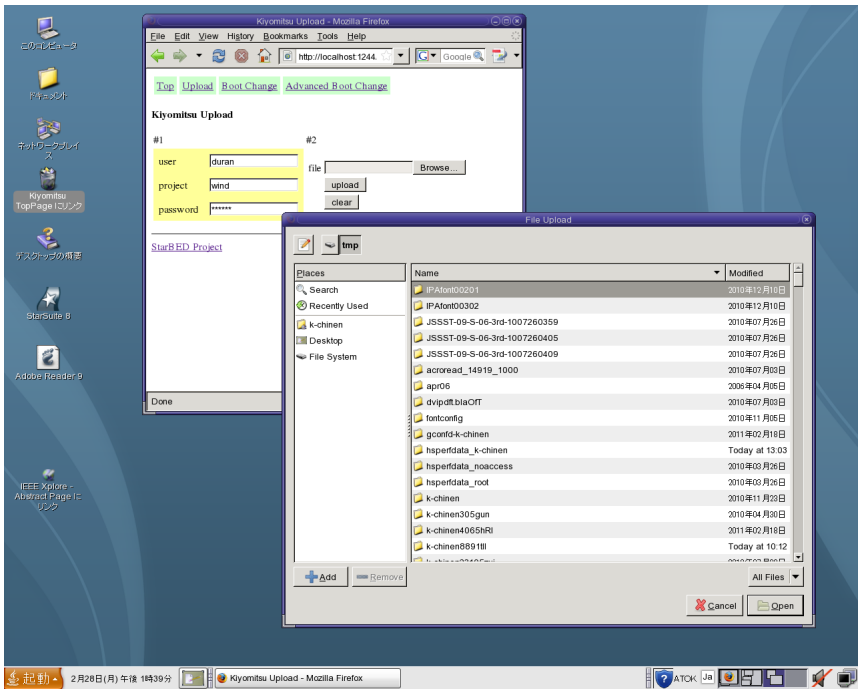


Figure 3.3: Selection of local file for uploading

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

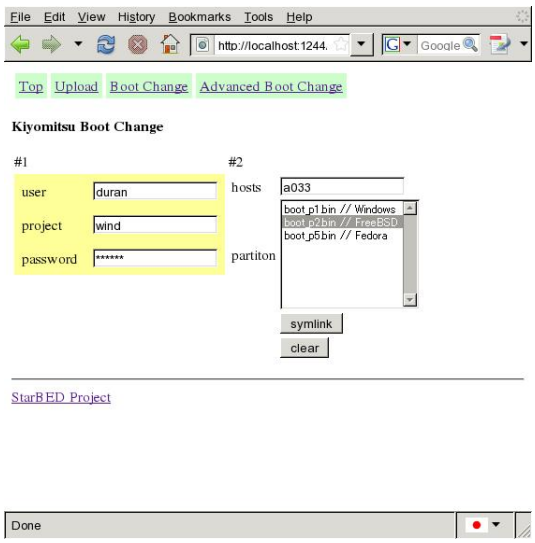


Figure 3.4: "Boot Change" page of Kiyomitsu

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.

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"
pxeloader "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
pxeloader	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.

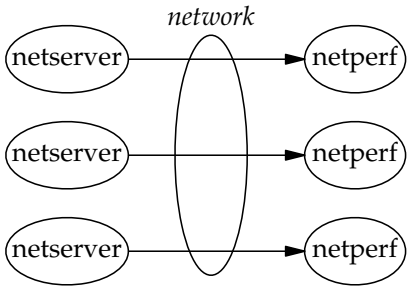


Figure 4.1: The Network Flow of Example Experiment

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 **pkill**. 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-**netiffit**. 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.

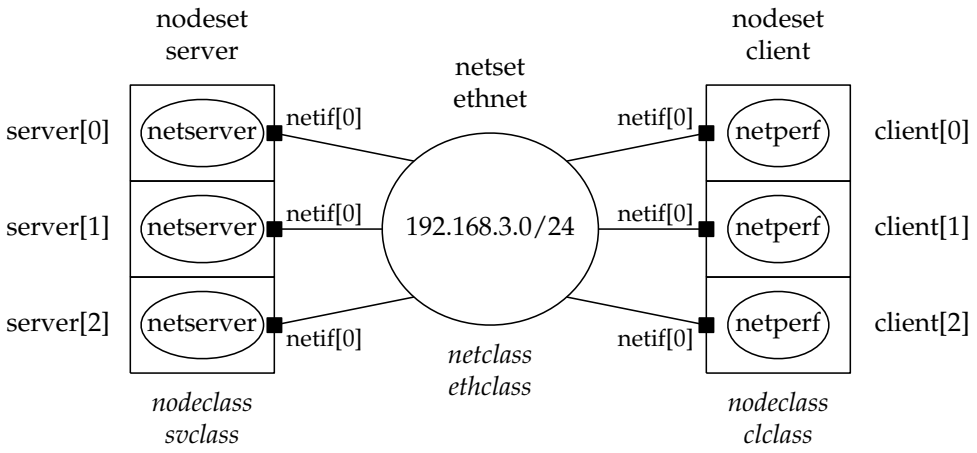


Figure 4.2: The Relationship of Example Experiment

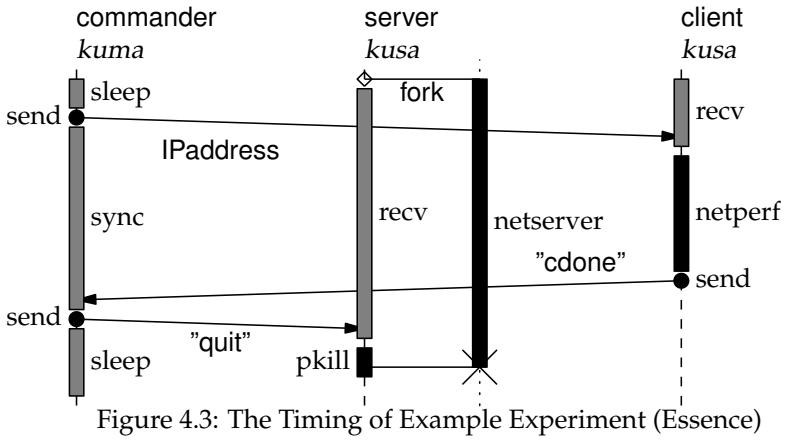


Figure 4.3: The Timing of Example Experiment (Essence)

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"
encd ipaddr "172.16.220.118"
ipaddrrange "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 {
    netiffit "/tmp/ifscan"
```

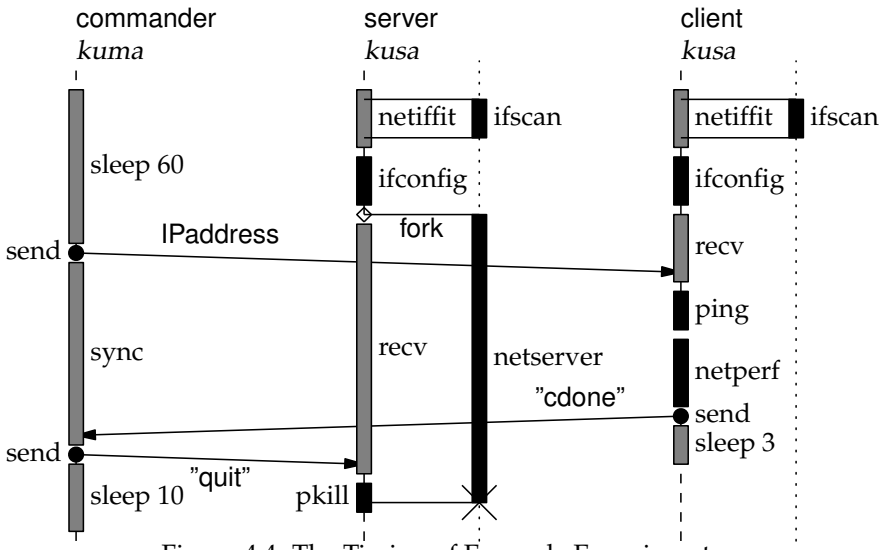


Figure 4.4: The Timing of Example Experiment

```
    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 {
    netiffit "/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].ipaddrrange = "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.

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

- 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	allocated all
server	3	1	4
client	3	1	4
total	6	2	8

Using **sparenoderatio** and **sparenodemin** 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
sparenodemin 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** applys 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.

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

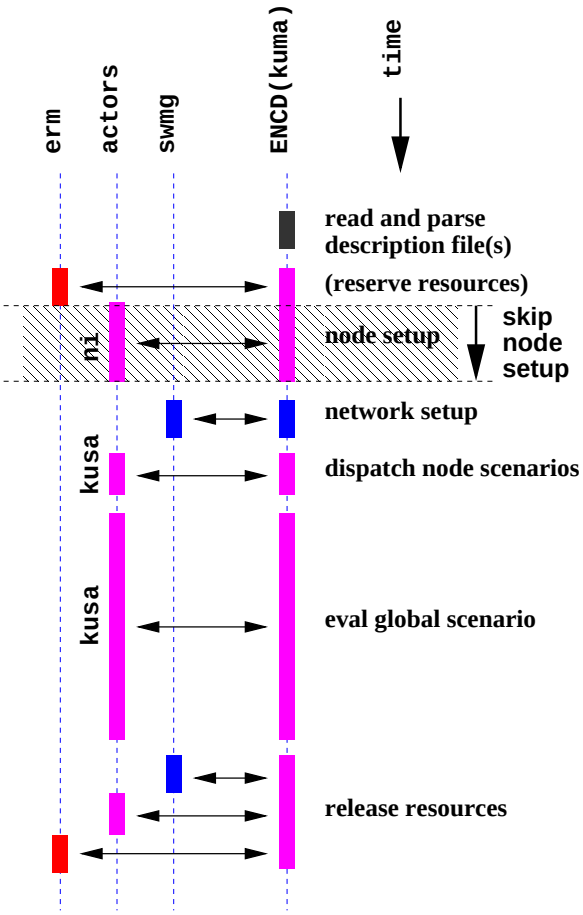


Figure 4.5: A Process Flow with skipnsetup

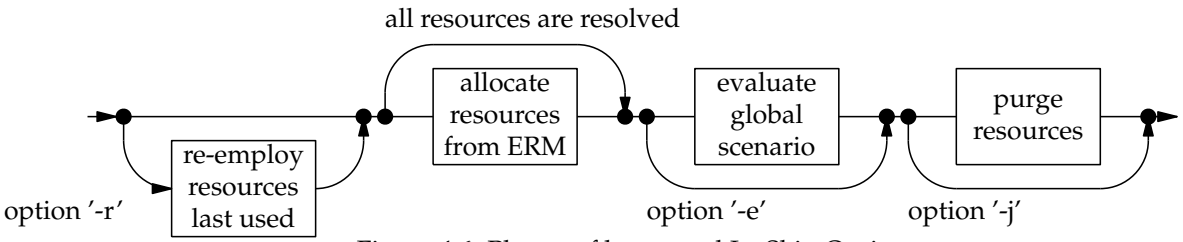


Figure 4.6: Phases of kuma and Its Skip Option

```
% ./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 Static Binding of Resources

Version 1.6 and later supports static binding of resources. By passing the binding table by file in arguments, you can use that.

For example, make the file **today.rbl** with following contents. This binds node **cl[0]** and **cl[1]** to host **n010**, **n011** and **n013**. The number of value in **rname** part have to longer or equal than that of **name** part. The lack of **rname** part causes error. In overflow, the rest of **rname** part is ignored.

```
# this is comment.
node name "cl#0-1" rname "n010-11,13"
```

And run **kuma** with **-r** and **-c**.

```
% ./kuma ... -r -c static:today.rbl ... some.k
```

Of cause, you should verify appeared nodes in the script and the binding table.

4.6 Hints of Script 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 wanted 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.pl** 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 does not aim to be used 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 include 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 mentions all types).

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 recognizes 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.

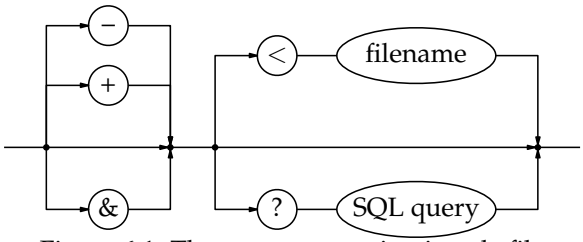


Figure 6.1: The operator parsing in rule file

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

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

Perphaps, 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 pctable 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
nodenumZIIId	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 pctable
        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 11 types. Do not use "obsoleted" and "reserved" types.

token	Software	Hardware
IOS	Cisco IOS	Catalyst 6500, 6509, 2960S and 3750
NexsusOS	Cisco NexsusOS	Nexsus 5000 and 300 series
EXXOS	Extreme XOS	Summit X650 and X450
IronWare*	Brocade(Foundry) IronWare	BigIron MG8, RX-16 and -32
BigIron	Brocade(Foundry) IronWare	BigIron MG8, RX-16 and -32
FastIron	Brocade(Foundry) IronWare	FastIron FCX628
DLDGS	D-Link *unknown*	DGS Series DGS-3426, -3427 and -3450
Qfabric	Juniper Junos	QFX
SeaMicro	SeaMicro *unknown*	SeaMicro
OSL2	Alaxala ; reserved	AX-2400
CATOS	Cisco Catalyst OS; obsoleted	Catalyst 6500

* *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.*

  - **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.

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 "sw3" requires username "ringo", you should write following.

```
...
swtype name "sw3" type "IOS" user "ringo"
...
```

If switch "sw4" requires username "ringo" and password "starr", you should write following.

```
...
swtype name "sw4" type "IOS" user "ringo" passwd "starr"
...
```

This configuration syntax allows enable password also. If switch "sw5" requires enable password "obladi", you should write following.

```
...
swtype name "sw5" 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 "swh001" 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
hostipmiuser a1-3,b12-13 john
hostipmipasswd a1-3,b12-13 elton
```

The syntax of the file is simular that of **swmg** (Section 6.6)

ipmiuser and **ipmipasswd** sets username and password for IPMI **pwmg** checks environment variables IPMI_USER and IPMI_PASSWORD for **ipmiuser** and **ipmipasswd**.

hosipmiuser and **hostipmipasswd** are same of then except they required hostname list. In above example 5 hosts (a001, a002, a003, b012 and b013) applied this configuration. So, **ipmiuser** and **ipmipasswd** are a common configuration for all hosts.

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 those 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 ownershipment 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
dhcpd	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). In CentOS, it is controlled by **iptables**. Other OS have simular programs. You can disable/enble ports by such programs. If you use closed network (no connectivity for Internet and other networks), you may can stop the programs in that rare case.

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

protocol	chapter	description
DMP	24	DMP
ERRP	20	ERRP
ESQP	21	ESQP
HTTP+	-	-
PWCP	23	PWCP
SWCP	22	SWCP
WOLAP	25	WOLAP
s-exp	-	-
s-exp-i	-	-
text	-	-

program order

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

protocol order

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

port order

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

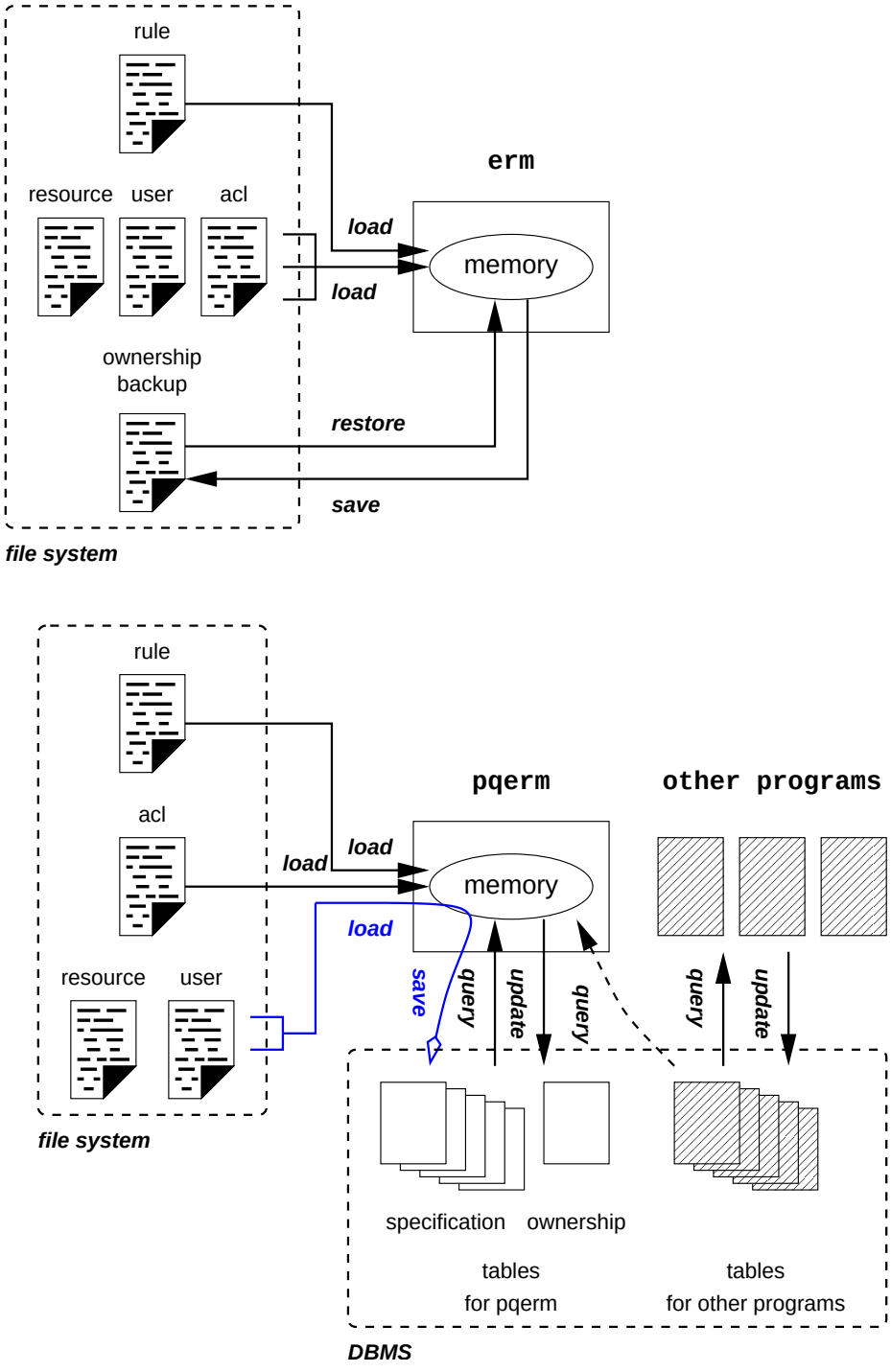


Figure 6.2: erm data operations

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 Nakagawa, Ken-ichi Chinen, Shinsuke Miwa and Yoichi Shinoda: StarBED and SpringOS Architectures and Their Performance, Tridentcom 2011.

[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 **`kusa`**. They stand "Kuroyuri master evaluator" and "Kuroyuri slave evaluator".

ifsetup

`ifsetup` is shell script. It is executor of **`ifconfig`**. Since **`netif`** statement of the language is available, **`ifsetup`** is obsolete.

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 referrded to as "alice.") However, it is not free. Then StarBED project decided to develope 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 manipurator

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. Futhermore, 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 *port_no*	set port number (default 1234.)
-R *file*	set resource database file.
-U *file*	set user database file.
-A *file*	set access control list file.
-S *file*	set rule file.
-B *file*	set ownership backup file.
-s *num*	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 offernode sweepvlan** and **offervlan**. sweep means gathering of available resource. Typical its defintion is RDB and its masking by ACL.

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

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

```
offernode [
&<ACL
]
```

pqerm(1) accept more complex rule using SQL.

SEE ALSO

pqerm(1), ERRP

NAME

fncp - faclity 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 tunning 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 ouput is used to call **ifconfig** and related. **netiffit**-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), netiffit(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* |-c *method*:*file*] [-ej] [-D masks] [-U masks] [-P *port_no*] [-Q *port_no*] [-L *num*] [-RWFTNS] [-iI] [-m] [-Z] [*var.def.s*] *file* [*file* ...]

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 *passwd*	set password for ERRP.
-r	reuses resources of last session. it is equal -c history:kuma.rbh .
-c *file*	set resource binding history (rbh) file. Default is **kuma.rbh**.
-c *method*:*file*	set resource binding method and file for that method. -c static:today.rbl -c history:last.rbh
-e	skips evaluation of global-scenario.
-j	skips resource releasing at program termination.
-d	debug mode
-D *masks*	set debug masks.
-U *masks*	unset masks.
-P *port_no*	set port number for ENCD.
-Q *port_no*	set port number for ESQP.
-L *num*	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 expriment description files as input, it recognize that the script has distribution processing and setup up them.
Without *scenario*-statement, the script file is evaluted as standalone program. It may useful for some purpose like following (address calculation).

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

RBH FILE

Resource binding history is recorded as following. The line started "#@" means recording time in epoch time.

```
#@time 1326344761
node name "client-0" rname "127.0.0.1"
node name "server-0" rname "127.0.0.1"
```

In static binding, time information is not required.

```
# this is comment
node name "a#0-1" rname "sbcla7-13"
node name "a7#0-3" rname "n3-4,f7-9"
node name "a7#1" rname "j33"
```

VARIABLE DEFINITION

User can set variable's value before the evaluation of specified description files. Setting are recognized by = (equal) in command arguments. In following example, variable x is 3 in default. If x=8 is applied in argument, its values is setted 8. Statement *assure* means "assigment if is not setted." Otherhand variable y is If y=9 is applied in argument, its values is overwrited y=7 in the script. So, without *assure* user cannot change the value of variables.

```
% cat bar.k
assure x=3
y=7
print x+y
% ./kuma bar.k
10
% ./kuma x=8 bar.k
15
% ./kuma y=9 bar.k
10
```

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 *port_no*	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 *file*	set the name of log-file.
-i	enter interactive mode.
-I *port_no*	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 *agentID*	set agent ID.
-S *sessID*	set session ID.
-p *port_no*	set the port number of reporting.
-r *times*	set the times of configuration retrieval retry.
-w *times*	set breath radix of configuration retrieval.
-W *sec*	set breath interval of configuration retrieval.
-R *times*	set the times of disk-image retrieval retry.
-i *sec*	set interval of monitoring — ticks of monitoring. *ni* checks transfer size and elap time to progress report in several seconds.
-I *sec*	set interval of progress report by elap time. the value of this parameter should large or equal than that of -i option.
-B *path*	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*] [-e *PM*] [-P *path*] [-K *NI*] [-U *path*] [-L *path*] [-V *dev-pattern*] [-X *prefix*] [-x *prefix*] [-m] [-zYQBRM] *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*] [-e *PM*] [-P *path*] [-K *NI*] [-U *path*] [-L *path*] [-V *dev-pattern*] [-X *path*] [-x *path*] [-m] [-zYQBRME] *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 *mask*	mask debug messages.
-u *user*	set user name.
-p *passwd*	set password.
-j *project*	set project. **erm** and related identify experiment by user name and project.
-F *file*	set option file. the program reads the file and applies options on the file.
-l *file*	set logging file.
-r *ERM*	set RM information. the syntax of information is *host*:*port* .
-f *FNCP*	set FNCP information. the syntax of information is *host*:*port* .
-s *ENCD*	set ENCD information. the syntax of information is *host*:*port* . 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 *DMAN* — set **dman** information. the syntax of information is *host*:*port* .

-e *PM* — set **pwmg** information. the syntax of information is *host*:*port* .

-P *path* — set the path of PXE boot-loader. You should indicates absolute path in **tftpd**.

-K *NI* — set **ni** information. the syntax of information is *diskimage*:*kernel* . You should indicates absolute path in **tftpd**.

-m — enable MBR operations.

-X *prefix* — 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 *prefix* — set prefix of MBR like -X.

-U *path* — 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 *path* — set device-name of target device for MBR like -U.

-V *device-pattern* — set device-pattern. the syntax is *disk-type1:num=device-path,num=deive-path;disk-type2:num=device-path* . pattern separator is semicolon. pattern label is separated by colon. each partition is separated by comma. assignment between partition and device-path is separated by equal.

pattern-list ::= *pattern* ";" *pattern* ...
pattern ::= *label* ":" *partition-list*
partition-list ::= *partition* "," *partition* ...
partition ::= *num* "=" *device-path*

-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**.

-E — do not connect **pwmg**.

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
-e localhost:1242
-f localhost:1238
-K FreeBSD/ni02.fs:recover_system/kernel_recover
-P recover_system/pxeboot-nohang
```

```
-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

pwmg - power manager

SYNOPSIS

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

DESCRIPTION

pwmg 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 **pwmg.conf**.)

LIMITATIONS

pwmg 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

pwmg.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
setpxelinuxcfg
set
quit

ENVIRONMENT VARIABLES

KUROPASSWD

SEE ALSO

erm(1), pwmg(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 *port_no*	set port number (default SNMP (161).)
-R *path_and_arg*	set path of reboot and its argument
-P *path_and_arg*	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 retrievals 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 *port_no*	set port number (default 1240.)
-f *file*	set the name of configuration-file.
-S	do not access switch. it is fake mode for debugging.
-T *ms*	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 enought.

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

pwmg(1)

Part IV

K Language Reference

8 導入

この文章はK言語の文法マニュアルである。ネットワーク実験遂行システムKuroyuriで用いられることを念頭に設計されている。この文章では文法にのみ焦点を当てている。Kuroyuriや関連プログラムの集合体であるSpringOSに関しては、別の文献を参照して欲しい。

8.1 記述の基本的規則

K言語は行指向の言語である。

行頭が#(シャープ; shape)の行はコメントとして扱われる。以下の例では最後の行のみ有効となる。

```
# this is comment, please ignore.
print "helo"
```

数字やシンボルはそのまま記述することができる。文字列は″(ダブル・クォート; double quote)で囲む必要がある。

```
1323
abc
"broadway"
```

行が長くなった場合は\(バック・スラッシュ; back-slash)で次の行と連結することを指示する。

```
jugemu jugemu goko no surikire kaijari suigyo no suigyoumatsu \
unraimatsu furaimatsu kuneru tokoro ni sumu tokoro yaburakouji \
no burakouji
```

変数は＝(イコール; equal)で新たに作ることができる。

```
a = 123
b = "www server"
```

詳細は付録19(IV-51ページ)に譲るが、いくつか予約語があるので関数や変数の名前として使わないよう注意が必要である。特に、初心者が変数にtを使う場合が多いようなので注意してほしい。

9 状況および環境

この章では実験の状況を記述する構文を紹介する。

9.1 実験実施者とプロジェクト名

user <*"name"*> <*"mail-address"*> `user`

担当者の名前および連絡先を宣言する。

project <*"name"*> `project`

プロジェクト名を宣言する。

encd ipaddr <*"address"*> port <*"number"*> `encd`

実験全体のノード設定案内プログラム (experiment node configuration driver;ENCD) に関する設定。これは kuroyuri のマスターを指す。したがって、kuroyuri のマスターを実行するノードの IP アドレスを指定する。一般的には管理ネットワークの IP アドレスを用いる。ノードから複数のネットワークで到達できる場合は、それ以外のネットワークを用いることも可能。

9.2 ネットワーク

ipaddrrange <*"ipaddr[/mask-length]"*> `ipaddrrange`

この実験で利用する IP アドレスの範囲を指定する。実験記述中にこの範囲に含まれない IP アドレスが登場した場合に警告が出力される。

9.3 グローバル・シナリオ

`scenario`

scenario <*block*>

シナリオの記述。13 参照。

9.4 施設関連サーバ

`rmanager`

rmanager ipaddr <*"address"*> port <*"number"*>

リソースマネージャをアドレスとポート番号で指定する。ボート番号が文字列であることに注意。

`wolagent`

wolagent ipaddr <*"address"*> port <*"number"*> ipaddrrange <*"address/len"*>

Wake On LAN エージェントとその担当アドレス範囲を指定する。この構文は実験記述中に複数個記述できる。

```
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`

fncp ipaddr <*"address"*> port <*"number"*>

施設全体のノード設定案内プログラム (facility node configuration pilot;FNCP) に関する設定。

`tftpdman`

tftpdman ipaddr <*"address"*> port <*"number"*>

TFTPD のディレクトリを担当している、ディレクトリ制御プログラム (direcotry manipulator; DMAN) に関する設定。

`tftpdir`

tftpdir <*"path"*>

Kuroyuri は TFTP を用いてディスクイメージをインストールしている。そのディスクイメージを格納しているディレクトリを指示する。

`nidiskimage`

nidiskimage <*"path"*>

nikernel <*"path"*>	`nikernel`

nipxeloader <*"path"*>	`nipxeloader`

ディスクイメージをインストールするプログラム NI を起動するために、NI に用いる OS のディスクイメージ、カーネル、PXE ローダを指定する [1] 。

[1] 従来の pxeloader を変更

`swtype`

swtype name <*"name"*> type <*"type"*>

スイッチ名と種類を指定する。現在サポートしているのは IOS,CATOS そして IronWare である。

たとえば、ある施設では以下のような設定が記述される。

```
swtype name "silaswa001" type "IOS"
swtype name "silaswb001" type "CATOS"
swtype name "silaswb002" type "CATOS"
swtype name "silaswb003" type "IronWare"
```

以下のようなスイッチ種類が用意されている。

スイッチ種類	摘要
IOS	Cisco IOS 向け
CATOS	Cisco Catalyst 向け
IronWare	Foundry IronWare 向け
Thru	自動的な操作をしない。swcmd で操作するため

現行では swmg を介して通信するため、Thru の効果はない。

10 ノード

ノードとはネットワーク実験上のエンティティを指す。ここでは、PC と考えて良い。

```
node c {
    method "HDD"
    disktype "IDE"
    partition 2
    ostype "FreeBSD"
    diskimage "ftp://172.16.210.9/ifcheck-e.gz"
    netif media fastethernet
    scenario {
        recv x
        msgswitch x {
        "start" {
            wakewait "/sim/netperf" "/sim/netperf" "-H" "192.168.3.1"
            send "finishperf"
          }
        }
    }
}
```

node <*name*> <*block*>	node

<block> には以下のような属性を記述する。

10.1 ノード属性

method "HDD"|"DIRECT"	method

ノードの起動方法を指定する。現状ではハードディスクを指定できる [1] 。

[1] 従来の Memory は使用停止

`disktype`

disktype ″IDE″|″SCSI″

ディスクのタイプ。現状では IDE と SCSI を指定できる。

`ostype`

ostype <*″ostype″*>

利用する OS の種類

`partition`

partition <*number*>

利用するパーティション番号。利用できるパーティションは施設の管理者に問い合わせると良いだろう。
ちなみに、StarBED のシミュレーションクライアント装置では以下のようになっている。

番号	摘要
1	WINDOWS 2000
2	FreeBSD
3	なし (FAT32 でフォーマットしてある)
4	拡張パーティション
5	TurboLinux

`diskimage`

diskimage <*″URL″*>

利用するディスクイメージを URL で指定する。

`netif`

netif media fastethernet|gigabitethernet

ネットワークインターフェースを宣言。メディアはファスト・イーサネットとギガビット・イーサネットに対応している。

`netif`

netif media fastethernet|gigabitethernet emulation <*″type program″*>

ネットワーク／エミュレーション (11 章) 利用時は emulation を指定する。

種別	摘要
internal	内部設置
external	外部設置

プログラム	摘要
netem	Linux Qdisc のサブモジュール
dummynet	FreeBSD のモジュール

scenario <*block*> `scenario`

ノード・シナリオの記述。後述 (第 13 章)。

10.2 ノード・クラス

実験では属性の同じノードを多数利用する場合が多いので、同じ属性のノード集合を作る方法を用意してある。

nodeclass <*name*> <*block*> `nodeclass`

クラスの構文は node と同様であるが、実験記述では直接操作する実体ではない。

nodeset <*name*> class <*class-name*> num <*number*> [*spare* <*number*>] `nodeset`

class-name で宣言したノード・クラスと同じ属性のノード集合を宣言する。このノード集合は num で指定した個数のノードで構成される。なお予備機は spare で指定した個数となる。これは sparenodemin, sparenoderatio より高い優先度で適用される。

ノード群を構成するノードの属性の変更は以下のように行う。

```
nodeset frontend class serverC num 5
...
frontend[4].netif[4].ipaddr = "192.168.9.41"
```

10.3 セットアップ・パラメータ

sparenodemin <*integer*> `sparenodemin`

予備ノードの最小値を指定する。標準値は 1 である。

sparenoderatio <*integer*> `sparenoderatio`

確保するノードの比率を指定する。標準値は 105 である。

最終的には、n 台のノードが必要な際には、MIN(n+sarenodemin,n*sparenoderatio/100) の結果が確保するノードとなる。

したがって予備機の確保を抑制する場合は以下のような記述が必要となる。

```
sparenodemin 0
sparenoderatio 100
```

`setuptimeout`

setuptimeout total <*number*> warm <*number*>

ノードの起動処理を打ち切る時間を指定する。単位は秒。total はノード起動処理全体を打ち切る。warm は必要最小限のノードがそろった後に待つ時間。

```
setuptimeout total 1800 warm 6
```

11 ネットワーク／エミュレーション

11.1 エミュレーション／ノード／シナリオ

emunodescenario *<"type program">* media *<"media-type">* block — `emunodescenario`

emunodescenario *<"type program">* — `emunodescenario`

エミュレーション種別とプログラム毎にノード／シナリオを記述する。media を指定した場合は、特定のメディア種別だけの指定となる。

```
emunodescenario "internal netem" {
    callw "/usr/bin/ping" "172.16.3.8"
}
```

emunetiprangerule *<"mask">* *<offset>* — `emunetiprangerule`

ネットワーク／エミュレーションのための IP アドレス範囲の生成規則を指定する

```
emunetiprangerule "10.0.7.0/24" 33
```

12 ネットワーク

ノードと似た形態でネットワークを宣言する。ノード同様にクラス定義とクラスを用いた集合宣言も可能である。

```
netclass e {
    media fastethernet
    ipaddrrange "192.168.3.0/24"
}
```

net <*name*> <*block*>	net

netclass <*name*> <*block*>	netclass

netset <*name*> class <*classname*> num <*num*>	netset

ネットワークの属性は以下で説明する。

12.1 ネットワーク属性

media fastethernet|gigabitethernet	media

ネットワークのメディア。

ipaddrrange <*"ipaddr/*[*mask-length*]*"*>	ipaddrrange

このネットワークに割当てる IP アドレスの範囲。

12.2 接続と開放

ネットワークへのノードの論理的な接続と開放を記述する構文が用意されている。

`attach`

attach <*netif*> <*net*>

このネットワークにノードのインターフェースの接続を示す。この構文で、評価器はそのノードの参加するネットワークを認識する。IP アドレスは参加したネットワークの属性から自動的に割り当てられる。そして、ネットワークスイッチの設定も自動的に施される。

```
attach cl.netif[2] leaf
```

`detach`

detach <*netif*> <*net*>

detach は attach と反対にインターフェースをネットワークから切り放す。

```
detach cl.netif[2] leaf
```

`routeq`

routeq <*node*> add net <*dst-net-addr*> gw <*gw-addr*>

ある特定ノードの経路を指定する。

```
net A {
    ipaddrrange "192.168.3.0/25"
}
routeq foo add net A.ipaddrrange gw bar.netif[0].ipaddr
```

13 シナリオ

シナリオはノードの挙動の集まりで、各ノードの属性として定義すると、そのノードの挙動 (ノードシナリオ; node scenario) となり、それ以外は、実験全体の挙動 (グローバルシナリオ; global scenario) となる。
双方とも記述された順に実行される。

scenario <*block*>	`scenario`

ブロックでは以下の文が利用できる。

13.1 外部プログラム起動

シナリオの大部分は実験対象となるソフトウェアの設定と起動に費される。したがって、対象プログラムや設定プログラムの起動は重要な構文である。

wake <*"program-path"*> <*"program"*> <*"args"*>...	`wake`

wakewait <*"program-path"*> <*"program"*> <*"args"*>...	`wakewait`

wake と wakewait はプログラムを起動する。wakewait は起動したプログラムの終了を待つ点が wait と異なる。program-path は起動対象のプログラムの絶対パスである。program はプログラムに与えるプログラムの名称 (名称で挙動の変わるプログラムがあるため必要) である。args は引数となる。
コンテキストの面で見ると、wake は並行実行、wakewait は直列実行となる。
wakewait を用いる場合は終了条件を考える必要がある。たとえば、多くの OS の ping は特別な引数を与えない場合には自動的には停止しない。そのような ping には -c を試みると良いだろう。

```
wakewait "/usr/bin/ping" "/usr/bin/ping" "-c" "10" "www.starbed.org"
```

自動停止しないプロセスを wake で起動した場合は、kill や killall を使って停止することになる。

```
wake "/usr/bin/ping" "/usr/bin/ping" "anywhere.onthe.net"
sleep 10
wakewait "/bin/killall" "/bin/killall" "ping"
```

簡単に停止しない場合は、数秒後に強制的に停止させることも必要だろう。

```
wakewait "/bin/killall" "/bin/killall" "foo"
sleep 3
wakewait "/bin/killall" "/bin/killall" "-KILL" "foo"
```

名称で挙動の変わるプログラムは少ないため、wake と wakewait をもうすこし簡略化した構文も用意した。それが、call と callw である。

`call` call *<"program-path"> <"args">...*

`callw` callw *<"program-path"> <"args">...*

可変長の引数に対応するためリストによる実行の構文 lcall を用意した。

`lcall` lcall *<list>*

`lcallw` lcallw *<list>*

```
p1=list("/bin/echo","helo")
lcall p1
p2b=list("world")
p2=append(p1,p2b)
lcall p2
```

13.1.1 リダイレクト

wake や wakewait で起動したプログラムの出力をリダイレクトする文法も用意してある。

記号	意味
`>`	標準出力 (stdout) をリダイレクト
`>>`	標準出力をリダイレクト (追記)
`2>&1`	標準エラー (stderr) を標準出力へ混ぜる

たとえば、以下のような文で ps と ls の実行結果を集めることができる。

```
wakewait "/bin/ps" "/bin/ps" "aux" > "/tmp/kusa.out"
wakewait "/bin/ls" "/bin/ls" "-l" "/" >> "/tmp/kusa.out"
```

13.2 プロセス停止

kill *<pid>* `kill`

指定したプロセスに指定したシグナルを発行する。シグナルは伝統にしたがって SIGTERM を用いる。

```
cid = wake "/usr/bin/ping" "/usr/bin/ping" "anywhere.onthe.net"
sleep 10
kill cid
```

killval *<signal>* *<pid>* `killval`

指定したプロセスに指定したシグナルを発行する。signal が数字の場合は、その数字をシグナル番号としてシグナルを発行する。signal が文字列の場合は、シグナル番号に変換する。シグナル番号は OS によって異なるので、文字列を使った方が汎用性が高い。

```
cid = wake "/usr/bin/ping" "/usr/bin/ping" "anywhere.onthe.net"
sleep 10
killval "TERM" cid
```

番号を指定する場合は以下のようになる (詳細はマニュアル参照[1])。

```
killval 9 cid
```

現状で考慮している文字列は ”TERM”, ”KILL”, ”QUIT”, ”HUP”, ”INT” の 5 つである。

[1] kill(2), signal(3) 等

13.3 表示

print は数字や文字列、変数内容を表示する構文である。

print

print <*expr*>

```
print 34
print 4+9
print "helo"
print a
print theserver.netif[1].ipaddr
```

13.4 実行状態変更

exit

exit

exit

exit <*val*>

評価器の評価が終了する。値 val を与えた場合は、その値をプロセスの終了値とする。

abort

abort

評価器が異常終了する。

atexit

atexit <*block*>

終了時の処理をあらかじめ登録する。
以下のような記述では、32 が表示された後に文字列 bye が表示される。

```
atexit {
        print "bye"
}
print 32
```

sleep

sleep <*second*>

msleep

msleep <*milsecond*>

指定された時間だけスリープ (何もしない) する。明示的な待機やタイミング調整で用いられる。sleep は秒単位、msleep はミリ秒単位である。ただし、厳密な精度は期待できない。

13.5 ディレクトリ

作業ディレクトリを移動する。

chdir <*"path"*>	`chdir`

13.6 メッセージ交換

実験全体を管理するマスタ、ここのノードで実験を駆動するスレーブ間でメッセージを交換することができる。このメッセージ交換により実験進行の信号や個別処理終了の応答の記述が可能となる。

13.6.1 メッセージ送信

send [*node-name*] <*"message"*>	`send`

ノードへメッセージを送信する。<node-name> が省略された場合は、node が省略された場合は、既定値のノードへ送る。スレーブではマスターが既定値となっている。

multisend <*nodeset-name*> <*"message"*>	`multisend`

ノード集合のメンバへメッセージを送信する。多数ノードへのメッセージ送信を容易にするために設けられている。

13.6.2 メッセージ受信

メッセージを受信し、受信内容を変数に格納する。

recv [*node-name*] <*var*>	`recv`

ノードへメッセージを受信する。node-name が省略された場合は、既定値のノードから読み込む。スレーブではマスターが既定値となっている。

recv では受信内容を一つの要素 (数字や文字列) として扱う。複数の要素を受信する場合は、以下の recva を用いる。recva は受信内容を配列として変数に格納する。

recva [*node-name*] <*var*>	`recva`

13.6.3 メッセージ待機

sync

sync <*block*>

sync は処理を同期するための構文で、ブロック内に処理を再開する条件を列挙する。ブロック内の条件が満たされるまで休止する。ブロック内の条件には msgmatch および timeout を記述できる。

```
sync {
    msgmatch foo "done"
    timeout 900
}
```

msgmatch

msgmatch [*node-name*] <*"message"*>

node-name から指定メッセージを受信を待つ。

multimsgmatch

multimsgmatch <*nodeset-name*> <*"message"*>

nodeset-name で指定されたノード集合からの受信を待つ。ノード集合メンバ全員からの受信内容が揃うまで待つ。

timeout

timeout <*seconds*>

タイムアウト時間の指定。

13.7 制御構造

if

if (<*cond*>) <*block*>

if

if (<*cond*>) <*block*> else <*block*>

条件を満たした場合に block に記述された処理を行う。

```
if(i%2==0) {
    print "even"
}
else {
    print "odd"
}
```

repeat <*num*> <*block*> `repeat`

指定回数分処理を繰り返す。

```
repeat 10 {
    print "a"
}
```

while(<*cond*>) <*block*> `while`

条件を満たしている間、処理を繰り返す。

```
i=0
while(i<10) {
    print "a"
    i++
}
```

for(<*init*>;<*limit*>;<*incr*>) <*block*> `for`

初期処理のあと、条件を満たしている間、処理を繰り返す。同時に繰り返す前に行う処理を指定する。

```
for(i=0;i<10;i++) {
    print "a"
}
```

loop <*block*> `loop`

単純に処理を繰り返す。

```
i=0
loop {
    if(i>=10) {
        exit
    }
    print "a"
    i++
}
```

foreacharray <*var*> <*array-name*> <*block*> `foreacharray`

配列の要素分だけ処理を繰り返す。

```
nodeclass Ccl {
}
nodeset cl class Ccl num 3
foreach i cl {
    print i.name
}
```

上の例ではノード集合分だけ繰り返すので、以下のような実行結果を得るだろう。

```
cl-0
cl-1
cl-2
```

`foreachlist`

foreachlist *<var>* *<list-name>* *<block>*

リストの要素分だけ処理を繰り返す。

```
fruit=list("apple","orange")
foreachlist x fruit {
    print x
}
```

msgswitch *<"string"> <block>* msgswitch

msgswitch は値にしたがって処理を分岐する。msgswitch のブロック中にはさらに以下のようなブロックが記述できる。値が指定した文字列と一致した場合そのブロック中の処理が行われる。

```
"<string>" {
  ...
}
```

そこで、msgswitch を使って以下のような処理を記述できる。

```
loop {
    recv cmd
    msgswitch cmd {
      "start" {
        wakewait "/bin/thetarget" "/bin/thetarget" "start"
        send "start-ack"
      }
      "stop" {
        wakewait "/bin/thetarget" "/bin/thetarget" "stop"
        send "stop-ack"
      }
      "done" {
        sleep 30
        exit
      }
    }
}
```

13.8 スイッチ操作

`swcmd`

swcmd *<"sw"> <string>*

swcmd はスイッチへ直接命令を発行する構文である。swtype 構文で宣言したスイッチのみ対象とする。swtype 構文によるスイッチ種類の指定を Thru にしておくと、パスワード等の様々な暗黙な命令が発行されず、swcmd による明示的な命令だけ発行される。実験施設から外部へ接続する境界スイッチ等、一時的に操作する場合に有用である。

現行では swmg を介して通信するため、Thru の効果はない。

```
swcmd "sw717" "abrakatabura"
swcmd "sw717" "set terminal length 0"
swcmd "sw717" "show vlan"
```

13.9 インターフェース走査

netiffit <*"path"*>	`netiffit`

指定されたパスのプログラムを使ってネットワークインターフェースを走査する。現在のところ SpringOS に含まれている ifscan を想定している。

netiffit 実行後はデバイス名が決まるため、ifconfig 等のネットワーク関連の設定が容易となる。

```
netiffit "/sim/ifscan"
wakewait "/sbin/ifconfig" "/sbin/ifconfig" self.netif[0].rname \
        self.netif[0].ipaddr
```

savenodeinfo <*node*> <*"path"*>	`savenodeinfo`

指定されたノードの情報をファイルに保存する。

13.10 デバッグ向け

`settrace`

settrace *<"target">*

`unsettrace`

unsettrace *<"target">*

指定した評価器のモジュールの追跡出力を設定/解除する。現在の所、制御できるモジュールは以下のようになっている。

モジュール識別子	摘要
addr	IP アドレス
conn	コネクション
ejump	エラージャンプ
enbc	バッファリング入出力
encd	ENCD 処理
engen	汎用的要素処理
errp	ERRP 処理
esqp	ESQP 処理
eval	評価
mst	各種構造体
ncdb	ノード候補データベース
net	ネットワーク
nio	ネットワーク I/O
node	ノード
nsetup	ノードセットアップ
parse	K 言語解析
sw	スイッチ制御
symtbl	シンボル表
syntax	文法
tftpdman	TFTP 向けディレクトリ制御
thand	時刻処理
wol	WoL (wake on LAN)

評価に関する追跡出力の制御は以下のように行う。

```
chdir "/"
settrace "eval"
wakewait "/bin/ls" "/bin/ls"
unsettrace "eval"
```

13.11 関数定義

func <*name*> <*block*>	`func`

func <*name*> (<*arg*> [, ...]) <*block*>	`func`

引数がなくても良い。手続きを記述するために用いても良い。

```
func alpha {
    32+432
}
func beta(a,b) {
    a+b
}
func gamma(a) {
    print a
}

print alpha()
print beta(7,23)
gamma(42)
```

13.12 スレッド分岐

thread <*func*>	`thread`

```
func gonbei {
    ...
    exit
}

thread gonbei
```

14 変数

多くの言語と同様に K 言語では変数が扱える。データ型については後述 (16 章) する。

14.1 宣言・初期化

変数名はアルファベットか _ (アンダー・スコア; under score) で始まるアルファベットかアンダー・スコアか数字の文字の並びが使える。文字列や数字等の変数の型は代入時に動的に定まる。

特に宣言の必要はなく、通常は以下のように = で値を代入すれば初期化となり、宣言も兼ねる。

```
binpath = "/usr/local/bin"
iteral = 23
```

初出の変数には undef という特殊な値が入っている。使う前に必ず初期化すること。

array <*var*>[<*num*>]	array

配列は宣言が必要で、array 構文を用いる。

```
array m[3]
m[0] = "a"
m[1] = 43
m[2] = m[1]*2
foreach i m {
    print i
}
```

配列の長さは関数 alength で得られる。上の例は以下のように書くこともできる。

```
array m[3]
m[0] = "a"
m[1] = 43
m[2] = m[1]*2
for(i=0;i<alength(m);i++) {
    print m[i]
}
```

`assure`

assure <*var*> = <*expr*>

assure は初期化していない変数に代入を施す構文で、指定された変数が初期化されていない場合は、与えられた値 (右辺の評価結果) で初期化する。
例えば、以下のような記述ファイルを評価器に与えると結果は 56 が得られる。

```
assure foo = 43+13
print foo
```

評価器がこの記述を評価する前に変数 foo に値が代入されていれば、評価器は 56 の代入を行わない。

`export`

export <*var*>

実験記述を解析するマスター評価器とノードシナリオを評価するスレーブ評価器では異なるシンボル表を持つため、それぞれで用いられている変数は共通ではない。ただし、export で指定した変数はノードシナリオ評価開始時にマスターからスレーブに譲渡される。

14.2 変換

listoarray <*list-name*> <*array-name*>	`listoarray`

stringtolist <*string-name*> <*list-name*>	`stringtolist`

stringtoarray <*string-name*> <*array-name*>	`stringtoarray`

以下のようなスクリプトでは文字列 second が表示される。

```
orderlist=list("first","second","third")
listtoarray orderlist sar
print sar[1]
```

リストは型にとらわれないので、数字でも良い。以下のスクリプトでは 73 が表示される。

```
orderlist=list(22,73,91)
listtoarray orderlist sar
print sar[1]
```

文字列にたいしても同様の処理が可能である。

```
cmd="echo helo world"
stringtoarray cmd sar
print sar[1]
```

14.3 特殊変数 self

self はシナリオを実行するノードを指す変数である。グローバルシナリオではマスターを、ノードシナリオではスレーブを指す。文脈に依存する点に注意。

以下のようにノードの持つネットワークインターフェイスの IP アドレスを使う際に重宝する。

```
wakewait "/sbin/ping" "/sbin/ping" "-i" self.netif[0].ipaddr target
```

14.4 スコープ

ノードシナリオではノードシナリオ内で宣言した変数しか扱えない。グローバルシナリオではグローバルシナリオに至るまでに宣言した変数が扱える。

たとえば、以下のような実験記述では、ノードクラス foo のシナリオは変数 y は扱えないのでエラーとなる。一方、グローバルシナリオでは変数 x, y, z が扱える。

```
x = "ishikawa"
y = "toyama"

nodeclass foo {
    scenario {
       wake "/bin/echo" "/bin/echo" y
    }
}

z = "fukui"

scenario {
    wake "/bin/echo" "/bin/echo" x y z
}
```

14.5 名前の衝突

K 言語にとって変数と関数の違いはない。また、変数は要素一つの配列である。したがって、変数、配列、関数に同じ名前を割り当てることはできない。

15 内蔵関数

15.1 数学演算

rand(<*num*>) `rand`

整数の乱数を得る。

```
rand(8) ⇒ 6
rand(8) ⇒ 4
```

srand(<*num*>) `srand`

乱数の種を与える。毎回異なる乱数が欲しい場合は、時刻を種にすると良いだろう。

```
srand(time())
print rand(10)
```

15.2 アドレス演算

haddr(<*"IPaddress"*>) `haddr`

IP アドレスの文字列から、ホストアドレスのみを取り出す。

```
haddr("12.34.56.78/24") ⇒ "12.34.56.78"
```

naddr(<*"IPaddress"*>) `naddr`

IP アドレスの文字列から、ネットワークアドレスを取り出す。

```
naddr("12.34.56.78/24") ⇒ "12.34.56.0/24"
naddr("172.16.3.44/23") ⇒ "172.16.2.0/23"
```

`addradd`

addradd(<*"base"*>,<*delta*>)

ある IP アドレスに数を加えて新たな IP アドレスを合成する。

```
addradd("172.16.2.0/23", 300) ⇒ "172.16.3.44/23"
```

15.3 型変換

tostring(<*num*>) `tostring`

数字から文字列を生成する。

```
tostring(10) ⇒ "10"
k = 37
tostring(k) ⇒ "37"
```

なお、文字列は + で連結できる。

```
"haku"+"san" ⇒ "hakusan"
```

toint(<*"string"*>) `toint`

文字列を数字に変換する。

```
toint("13") ⇒ 13
d = 54
toint(d) ⇒ 54
```

15.4 その他

alength(<*array-name*>) `alength`

配列の長さを得る。

isinlist(<*x*>, <*list*>) `isinlist`

ある値がリストに含まれるかを判定する。含まれる場合は t を返す。

isinarray(<*x*>, <*array*>) `isinarray`

ある値が配列に含まれるかを判定する。含まれる場合は t を返す。

time() `time`

時刻を得る。一般的な UNIX 向けの C 言語の関数から得ていて、1970 年 1 月 1 日からの秒数。閏秒に関しては考慮されていないようだ。

`getpid`

getpid()

評価器のプロセス識別子。

15.5 実験的関数

以下は深層にある内部関数を使った実験的な実装である。仕様変更の可能性があるので、積極的には使わないこと。

list(<*expr*>[, <*expr*>]) `list`

リストを生成する。

```
list(a,b) ⇒ (a b)
```

append(<*expr*>,<*expr*>) `append`

リストへリストを追加する

```
append(a,b) ⇒ (a b)
append(list(1,2),list(3,4)) ⇒ (1 2 3 4)
```

length(<*expr*>) `length`

リストの長さを得る。

```
length(list(1,2,3,4)) ⇒ 4
```

quote(<*expr*>) `quote`

指定されたい引数を評価しない。

eval(<*expr*>) `eval`

指定された引数を評価する。
以下のサンプルを実行すると 7 と 3 の階乗として、5040 と 6 を得る。

```
func frac(n) {
    if(n<2) {
        1
    }
    else {
        n*frac(n-1)
    }
}

a = list(quote(frac),3)

print frac(7)
print eval(a)
```

16 データ型

K 言語は、単純型は INT, STR の 2 種類、複雑型は NODE, NET, NEIIF, AGENT, ADDFILE の 5 種類を持つ。そして、複数の NEITF と ADDFILE を扱うための型として、MNETIF, MADDFILE がある。たとえば、a.netif の型は MNETIF で a.netif[3] の型は NETIF となる。

名称	摘要
INT	整数
STR	文字列
NODE	ノード
NET	ネットワーク
NEIIF	ネットワークインターフェイス
AGENT	エージェント、通信相手のスレーブを指す
ADDFILE	追加ファイル
MNETIF	ネットワークインターフェイス配列
MADDFILE	追加ファイル配列

複雑な型は以下のような属性を持つ。

type	attr	R/W	get	put	desc.
NODE	name	RW	STR	STR	名称
	rname	RW	-	-	資源名
	description	R	STR	-	摘要
	method	RW	STR	STR	起動方法 (HDD/memory)
	ostype	RW	-	-	OS 種
	disktype	RW	STR	STR	ディスク種 (IDE/SCSI)
	partition	RW	STR	STR	パーティション (1,2 or 5)
	diskimage	RW	STR	STR	ディスクイメージ
	kernel	RW	STR	STR	カーネル
	n-netif	-	-	-	ネットワークインターフェイス数
	netif	RW	MNETIF	?	ネットワークインターフェイス
	n-addfile	-	-	-	追加ファイル数
	addfile	RW	MADDFILE	?	追加ファイル
	scenario	RW	any	any	シナリオ
	agent	RW	AGENT	?	スレーブ情報
	conn	P	-	-	スレーブコネクション
	lastmsg	P	-	-	最終メッセージ
	issued	-	-	-	発行時間
NET	name	RW	STR	STR	名称
	description	R	STR	-	摘要
	media	RW	STR	STR	メディア種
	ipaddrrange	RW	STR	STR	IP アドレス範囲
	bandwith	RW	STR	STR	帯域
	agent	-	-	-	通信エージェント
	conn	-	-	-	コネクション
NETIF	name	RW	STR	STR	名称
	type	?	-	-	種別
	phyport	?	-	-	スイッチの物理ポート
	media	RW	STR	STR	メディア種
	ipaddr	RW	STR	STR	IP アドレス
	macaddr	RW	STR	STR	MAC アドレス
	description	R	STR	-	摘要
AGENT	hostname	RW	STR	STR	ホスト名
	ipaddr	RW	STR	STR	IP アドレス
	portname	RW	STR	STR	ポート名
	pgname	RW	STR	STR	プログラム名
	msg	R	-	-	メッセージ
	lastmsg	R	-	-	最終メッセージ
ADDFILE	name	RW	STR	STR	名称
	description	R	STR	-	摘要
	file	RW	STR	STR	ファイル
	dir	RW	STR	STR	作業ディレクトリ

17 ユーザ定義型およびクラス

複雑な構造を扱うためにユーザ定義型を用意してある。これはC言語の構造体 (structure) に相当する。

struct <*struct-name*> <*block*>	`struct`

型の定義。

struct <*struct-name*> <*var-name*>	`struct`

定義された型を使った変数の宣言。

```
nodeclass serverC {
        netif media fastethernet
        netif media gigabitethernet
}
nodeclass firewallC {
        netif media gigabitethernet
        netif media fastethernet
}
struct site {
        node server class serverC num 2
        node firewall class firewallC num 2
        integer asnum
}
struct garaxy {
        struct site front num 2
        struct site rear num 2
        integer mnum
}

struct garaxy andromeda
print andromeda.mnum
print andromeda.front[0].asnum
print andromeda.front[0].firewall[0].name
print andromeda.front[0].firewall[0].netif[0].media
```

17.1 ユーザ型要素

ユーザ型には整数、文字列、ノードを含めることができる。

`integer`

integer <*name*> [*num* <*num*>]

`string`

string <*name*> [*num* <*num*>]

`node`

node <*name*> [*num* <*num*>]

`struct`

struct <*name*> [*num* <*num*>]

`method`

method <*name*> [*args*] block

17.2 クラス

class <*class-name*> <*block*>	`class`

class <*class-name*> based <*baseclass-name*> <*block*>	`class`

18 実践

この章では実験を行う際の作業や注意点について述べる。エディタなどファイルを記述するプログラムについては他の文献を参照のこと。

18.1 作業手順

エディタなどファイル編集プログラムについては他の文献を参照のこと。この節章では実験操作の手順例を示す。

1) 施設に依存する項目を調査する
   - リソース・マネージャ (RM) の稼働場所 (IP アドレス、ポート番号)。そして、そのバージョンと用いるプロトコル。
   - Wake On LAN agent の稼働場所 (IP アドレス、ポート番号)。
   - PXE ブートのための TFTP サーバ の稼働場所 (IP アドレスのみ、ポート番号は固定)。
   - NI を内蔵している OS のディスクイメージとカーネルの名称と場所。
   - 実験に使うことができる IP アドレス領域。
   - 実験に使うことができる VLAN 番号領域。
   - FNCP (NI 向け HTTP リダイレクトサーバ) の稼働場所 (IP アドレス、ポート番号)。
2) 実験に登場するノード集合を役割毎に分割する。
3) 役割毎にノード・シナリオを決める。
4) 各ノードで使うディスク・イメージを作成する
   - OS や各種ソフトウェアをインストールする。
   - ディスク・イメージを FTP サーバに保存する。

   Kuroyuri のパッケージを含めて、スレーブが自動的に起動するように設定すること。
5) 実験記述ファイルを作成する。
   - 施設
     - rmanager
     - wolagent
     - tftpdir
     - nidiskimage, nikernel, nipxeloader

  - fncp
- 実験
  - user
  - project
  - ipaddrrange
  - nodeclass
  - node/nodeset
  - netclass
  - net/netset
- シナリオ
  - scenario

6) NI のためのファイル群やノードにインストールする実験ノードのディスクイメージが、それぞれ TFTP サーバと FTP サーバに格納されている事を確認する。

7) 実行

多くの場合、役割によってシナリオが決まり、必要なネットワーク・インターフェイスが定まる。したがって、実験を役割の視点で考えて、ノード・クラスや集合を記述すると良いだろう。

18.2 ポータビリティ

施設に依存する記述を集めて別ファイルにすると、実験記述のポータビリティが高まる。rmanager, wolagent 等が施設に依存する構文である。

たとえば、二つの施設で稼働させる場合は、施設に依存しない記述を expr.sc に、施設 A, B に向けにはそれぞれ facilityA.sc, facilityB.sc を用意しておく。施設 A では以下のように実行する。

```
% ./kuma facilityA.sc expr.sc
```

一方、施設 B では以下のように実行する。

```
% ./kuma facilityB.sc expr.sc
```

例えば、StarBED では以下に近い設定になるだろう。

```
#
# facility oriented information
#
# for StarBED 2003, 2004.
#
rmanager ipaddr "172.16.3.101" port "1234"

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"

tftpdir "/osbank"
nidiskimage "fs.PICOBSD.old"
nikernel "picokernel"
nipxeloader "pexboot"
```

```
swtype name "silaswa001" type "IOS"
swtype name "silaswb001" type "CATOS"
swtype name "silaswb002" type "CATOS"
swtype name "silaswb003" type "IronWare"
```

18.3 タイミング

分散プログラミングの常として、ネットワーク実験ではタイミングが重要となる。K 言語ではメッセージを交換しつつ実験を進行させることが可能なので、実行時間の分からない実験にも対応できたり、対話的なシナリオが書けたり、他の方法に比べいくぶん便利になっている。しかし、便利になっただけでタイミングの困難さを完全に取り除いたわけではない。タイミングには十分に注意して欲しい。

K 言語で処理がブロックするは受信である。受信は recv や recva、そして sync である。特に sync は複数の入力の状態を検査してビジー・ループに近い状態になるため、注意が必要である。処理が止まったように見える場合は recv や recva、そして sync に注目すると良い。

ノードの起動は OS や管理サーバの状態など多くの要素に依存するため、ノード・シナリオ開始時刻は正確には揃わない。同一仕様の PC を十台程使う場合は、sleep で数秒待機する程度で回避できる。さまざまな仕様の PC を用いたり、数百台規模の場合には、開始時刻が広がると予想されるため、ノード・シナリオで準備完了のメッセージを送信し、グローバル・シナリオで sync を使って同期すると良いだろう。

```
nodeclass fooC {
    ...
    scenario {
        sleep 30
        send "ready"
    }
}
nodeclass barC {
    ...
    scenario {
        sleep 30
        send "ready"
    }
}
nodeset foo class fooC num 32
nodeset bar class barC num 64

scenario {
    sync {
        multimsgmatch foo "ready"
        multimsgmatch bar "ready"
    }
    ...
}
```

シナリオ終了時刻も原則として揃わない。多くの場合は問題は起きないが、最後のメッセージ送信が相手に届く前にシナリオが終了してしまい、コネクションが開放されてメッセージが届かないという現象が起きやすい。ノード・シナリオ、グローバル・シナリオともに最後に sleep を書いておくと、トラブルが減る。

kuroyuri は自動的にスイッチを制御するため、K 言語では明示的なスイッチ制御の記述は不要である。ただし、スイッチ内部で設定内容が反映されるまでのタイムラグは検出できない。これは実際の通信をしてのみ確認できることで、自動的なスイッチ制御が原因ではない。シナリオ開始時に即座に実験対象プログラムを起動すると、このタイムラグで実験対象プログラムが不安定になったり、異常終了することがある。スイッチ制御が反映されるまで sleep でしばらく待つ (30 秒から 60 秒程度が目安) か、ping 等で疎通確認後に実験を開始すると良いだろう。

複雑な実験を記述すると、一方の送信回数が他方の受信回数より多かったり、あるいは、その逆のシナリオを書いてしまうことが起こる。送受信の回数が少ないシナリオを書くことをお勧めする。

タイミングチャートを描くと、すぐにミスが見付かることも多い。

19 言語仕様

K 言語が扱えるのは ASCII のみ、マルチバイトコードや日本語は扱えない。

19.1 文法

```
program
    : lines
block
    : '{' '\n' lines '}' '\n'
    | '{' '\n' '}' '\n'
lines
    : lines '\n'
    | '\n' lines
    | lines line
    | line
    | error '\n'
line
    : NOP wargs '\n'
    | LOOP block
    | REPEAT PINT block
    | WHILE '(' expr ')' block
    | FOR '(' expr ';' expr ';' expr ')' block
    | FOREACH PSYM PSYM block
    | FOREACHARRAY PSYM PSYM block
    | FOREACHLIST PSYM PSYM block
    | IF '(' expr ')' block
    | IF '(' expr ')' block ELSE block
    | FUNC PSYM fargwrap block
    | THREAD PSYM '\n'
    | ARRAY PSYM '[' expr ']' '\n'
    | LISTTOARRAY PSYM PSYM '\n'
    | STRINGTOLIST PSYM PSYM '\n'
    | STRINGTOARRAY PSYM PSYM '\n'
    | CLASS PSYM structblock
    | CLASS PSYM BASED PTYPE structblock
    | knowntype PSYM
    | knowntype PSYM '[' expr ']'
    | STRUCT PSYM structblock
    | STRUCT PSYM PSYM nodesetdecos
    | NODE PSYM nodesetdecos '\n'
    | NODESET PSYM nodesetdecos '\n'
    | NODE PSYM nodeblock
```

```
| NODECLASS PSYM nodeblock
| SWEEPNODESET PSYM VNTYPE PSYM '\n'
| NETSET PSYM netsetdecos '\n'
| BIND primary nodeblock
| BIND primary nodeline
| NET PSYM netblock
| NETCLASS PSYM netblock
| SWTYPE swattributes '\n'
| SWCMD expr expr '\n'
| ROUTE expr routecmd routenethost expr routegw expr '\n'
| ROUTEQ expr routecmd routenethost expr routegw expr '\n'
| RMANAGER agentattributes '\n'
| PMANAGER agentattributes '\n'
| SMANAGER agentattributes '\n'
| FNCP agentattributes '\n'
| ENCD agentattributes '\n'
| MONPORT agentattributes '\n'
| TFTPDMAN agentattributes '\n'
| TFTPDIR PSTR '\n'
| NIDISKIMAGE PSTR '\n'
| NIKERNEL PSTR '\n'
| NIPXELOADER PSTR '\n'
| PXELOADER PSTR '\n'
| WOLAGENT wolagentattributes '\n'
| SETUPTIMEOUT setuptimeoutattributes '\n'
| SPARENODEMIN PINT '\n'
| SPARENODERATIO PINT '\n'
| WOLINTERVAL expr '\n'
| SNMPINTERVAL expr '\n'
| RBHFILE PSTR '\n'
| EMUNODESCENARIO expr MEDIA expr block
| EMUNODESCENARIO expr block
| EMUNETIPRANGERULE expr expr '\n'
| SCENARIO block
| ATEXIT block
| NOATEXIT '\n'
| RECV expr '\n'
| RECVA expr '\n'
| RECV expr expr '\n'
| RECVA expr expr '\n'
| SEND expr '\n'
| SEND expr expr '\n'
| MULTISEND expr '\n'
| MULTISEND expr expr '\n'
| MSGSWITCH expr msgswblock
| SYNC syncblock
| USER PSTR PSTR PSTR '\n'
| USER PSTR PSTR '\n'
| USER PSTR '\n'
| PROJECT PSTR '\n'
| EXPRID expr '\n'
| CHECKEXPRID expr '\n'
| IPADDRRANGE PSTR '\n'
| _IPADDRMASK PSTR '\n'
| SAVENODEINFO pchains PSTR '\n'
| PRINT expr '\n'
| MONMSG expr '\n'
| SLEEP expr '\n'
| MSLEEP expr '\n'
```

```
    | EXIT expr '\n'
    | EXIT '\n'
    | ABORT '\n'
    | ATTACH expr expr '\n'
    | DETACH expr expr '\n'
    | SWCONF '\n'
    | NETIFFIT expr '\n'
    | SETTRACE PSTR '\n'
    | UNSETTRACE PSTR '\n'
    | NETGET expr expr '\n'
    | NETPUT expr expr '\n'
    | NETPAUSE PSYM expr expr '\n'
    | NETPAUSE PSYM expr '\n'
    | _DEBUGPRINT PSTR '\n'
    | CD expr '\n'
    | CHDIR expr '\n'
    | pchains '=' wakeline '\n'
    | wakeline '\n'
    | pchains '=' callline '\n'
    | callline '\n'
    | lcallline '\n'
    | KILL expr '\n'
    | KILLVAL expr expr '\n'
    | expr '\n'
    | EXPORT PSYM '\n'
    | ASSURE PSYM '=' expr
knowntype
    : PTYPE
    | STRING
    | INTEGER
structmemberdecos
    : /* empty */
    | structmemberdecos structmemberdeco
    | structmemberdeco
structmemberdeco
    : aclass
    | anum
nodesetdecos
    : /* empty */
    | nodesetdecos nodesetdeco
    | nodesetdeco
nodesetdeco
    : aclass
    | anum
    | aspare
netsetdecos
    : /* empty */
    | netsetdecos netsetdeco
    | netsetdeco
netsetdeco
    : aclass
    | anum
aclass
    : CLASS PSYM
anum
    : NUM expr
aspare
    : SPARE expr
routecmd
```

```
    : ADD
    | DEL
routenethost
    : NET
    | NODE
    | HOST
routegw
    : GW
msgswblock
    : '{' '\n' msgcaselines '}' '\n'
    | '{' '\n' '}' '\n'
msgcaselines
    : msgcaselines msgcaseline
    | msgcaseline
msgcaseline
    : expr block
    | '\n'
syncblock
    : '{' '\n' synccaselines '}' '\n'
    | '{' '\n' '}' '\n'
synccaselines
    : synccaselines synccaseline
    | synccaseline
synccaseline
    : MSGMATCH expr expr '\n'
    | MULTIMSGMATCH expr expr '\n'
    | MULTIMSGMATCH word expr expr '\n'
    | TIMEOUT PINT '\n'
    | TIMEOUT expr '\n'
    | '\n'
structblock
    : '{' '\n' structlines '}' '\n'
    | '{' '\n' '}' '\n'
structlines
    : structlines '\n'
    | '\n' structlines
    | structlines structline
    | structline
    | error '\n'
structline
    : INTEGER PSYM structmemberdecos
    | REAL PSYM structmemberdecos
    | STRING PSYM structmemberdecos
    | STRUCT PSYM PSYM nodesetdecos
    | NODE PSYM nodesetdecos '\n'
    | NODESET PSYM nodesetdecos '\n'
    | METHOD PSYM fargwrap block
nodeblock
    : '{' '\n' nodelines '}' '\n'
    | '{' '\n' '}' '\n'
nodelines
    : nodelines nodeline
    | nodeline
nodeline
    : NETIF PSTR netifattributes '\n'
    | NETIF netifattributes '\n'
    | AGENT agentattributes '\n'
    | EXAGENT agentattributes '\n'
    | DISKIMAGE PSTR '\n'
```

```
    | SCENARIO block
    | MAXVNODES PINT '\n'
    | VNTYPE PSYM '\n'
    | OSTYPE PSTR '\n'
    | METHOD PSTR '\n'
    | DISKTYPE PSTR '\n'
    | KERNEL PSTR '\n'
    | PARTITION PINT '\n'
    | ADDFILE PSTR addfileattributes '\n'
    | ADDFILE addfileattributes '\n'
    | '\n'
netblock
    : '{' '\n' netlines '}' '\n'
    | '{' '\n' '}' '\n'
netlines
    : netlines netline
    | netline
netline
    : amedia '\n'
    | aipaddrrange '\n'
    | '\n'
agentattributes
    : agentattributes agentattr
    | agentattr
agentattr
    : aipaddr
    | aport
    | atype
    | aprogram
    | ahost
    | aname
swattributes
    : swattributes swattr
    | swattr
swattr
    : aipaddr
    | aname
    | aport
    | atype
    | auser
    | apasswd
    | aepasswd
wolagentattributes
    : wolagentattributes wolagentattr
    | wolagentattr
wolagentattr
    : aipaddr
    | aport
    | aipaddrrange
setuptimeoutattributes
    : setuptimeoutattributes setuptimeoutattr
    | setuptimeoutattr
setuptimeoutattr
    : atotal
    | awarm
addfileattributes
    : addfileattributes addfileattr
    | addfileattr
addfileattr
```

```
    : adir
    | afile
netifattributes
    : netifattributes netifattr
    | netifattr
netifattr
    : amedia
    | aswporthint
    | aipaddr
    | amacaddr
    | anet
    | avia
    | aemulation
    | aemuparam
aprogram
    : PROGRAM PSTR
aname
    : NAME PSTR
    | NAME '\n'
atype
    : TYPE PSTR
    | TYPE '\n'
ahost
    : HOST PSTR
    | HOST '\n'
aipaddr
    : IPADDR expr
    | IPADDR '\n'
amacaddr
    : MACADDR expr
    | MACADDR '\n'
aport
    : PORT expr
    | PORT '\n'
aswporthint
    : SWPORTHINT expr
amedia
    : MEDIA PSYM
    | MEDIA '\n'
anet
    : NET PSYM
    | NET expr
    | NET '\n'
avia
    : VIA PINT
    | VIA '\n'
aemulation
    : EMULATION expr
aemuparam
    : EMUPARAM expr
aipaddrrange
    : IPADDRRANGE expr
    | IPADDRRANGE '\n'
afile
    : FILE PSTR
    | FILE '\n'
adir
    : DIR PSTR
    | DIR '\n'
```

```
atotal
    : TOTAL PINT
    | TOTAL '\n'
awarm
    : WARM PINT
    | WARM '\n'
auser
    : USER PSTR
    | USER '\n'
apasswd
    : PASSWD PSTR
    | PASSWD '\n'
aepasswd
    : EPASSWD PSTR
    | EPASSWD '\n'
qexpr
    : pchains
    | '(' expr ')'
fargwrap
    :
    | '(' fargs ')'
fargs
    : fargs ',' expr
    | expr
wargs
    :
    | wargs qexpr
rdirs
    :
    | rdirs PLT qexpr
    | rdirs PGT qexpr
    | rdirs PGG qexpr
    | rdirs P2G qexpr
    | rdirs P2GG qexpr
    | rdirs P2J1
pexpr
    : qexpr
wakeline
    : wakes pexpr pexpr wargs rdirs
    | wakes pexpr pexpr
    | wakes pexpr pexpr wargs
wakes
    : WAKEWAIT
    | WAKE
lcallline
    : lcalls PSYM rdirs
    | lcalls PSYM
lcalls
    : LCALL
    | LCALLW
callline
    : calls pexpr wargs rdirs
    | calls pexpr
    | calls pexpr wargs
calls
    : CALL
    | CALLW
expr
    : pchains '=' expr
```

```
    | expr PGT expr
    | expr PGE expr
    | expr PLT expr
    | expr PLE expr
    | expr PEQ expr
    | expr PNE expr
    | expr '+' '+'
    | expr '-' '-'
    | expr '+' expr
    | expr '-' expr
    | expr '*' expr
    | expr '/' expr
    | expr '%' expr
    | '(' expr ')'
    | '-' expr %prec UMINUS
    | pchains
list
    : '(' listv ')'
listv
    : expr
    | listv ',' expr
pchains
    : pchains '.' primary
    | pchains '.' primary '[' expr ']'
    | pchains '[' expr ']'
    | primary
bfunc
    : TOSTRING
    | TOINT
    | RAND
    | SRAND
    | HADDR
    | NADDR
    | ADDRADD
    | GETPID
    | TIME
    | LASTMSG
    | GETVNODES
    | INSTANCE
    | LENGTH
    | ALENGTH
    | APPEND
    | XPACK
    | LIST
    | QUOTE
    | EVAL
primary
    : bfunc list
    | bfunc '(' ')'
    | pchains '(' fargs ')'
    | pchains '(' ')'
    | word
    | const
word
    : NETIF
    | ANETIF
    | NODE
    | MEDIA
    | SCENARIO
```

```
    | ADDFILE
    | NUM
    | CPU
    | DISK
    | DISKIMAGE
    | MEMORY
    | ANY
    | ALL
    | AGENT
    | IAGENT
    | OSTYPE
    | PORT
    | HOST
    | IPADDR
    | IPADDRRANGE
    | MACADDR
    | METHOD
    | RNAME
    | NAME
    | FILE
    | DIR
const
    : PSYM
    | PSTR
    | PINT
    | DEFAULT
    | T
    | NIL
    | UNDEF
    | PERR
```

19.2 予約語

以下のような単語が予約されている。変数や関数の名前には使えないので注意して欲しい。

ABORT, ADD, ADDFILE, ADDRADD, AGENT, ALENGTH, ALL, ALQ, ANETIF, ANY, APPEND, APPLY, AQ, ARRAY, ASLISTQ, ASSURE, ATEXIT, ATOM, ATTACH, BASED, BEGIN, BIND, BOUNDP, CALL, CALLW, CAR, CD, CDR, CHDIR, CLASS, COND, CONS, CPU, DEC, DEFAULT, DEFCLASS, DEFSTRUCT, DEL, DESCRIPTION, DETACH, DIE, DIR, DISK, DISKIMAGE, DISKTYPE, DIV, ELSE, EMULATION, EMUNETIPRANGERULE, EMUNODESCENARIO, EMUPARAM, ENCD, END, EQ, EQUAL, EVAL, EXIT, EXPORT, FILE, FNCP, FOR, FOREACH, FOREACHARRAY, FOREACHLIST, FUNC, GE, GETAEQ, GETMAEQ, GETMEQ, GETPID, GETVNODES, GT, GW, HADDR, HOST, IF, INC, INSTANCE, INTEGER, IPADDR, IPADDRRANGE, KERNEL, KILL, KILLA, KILLVAL, KILLVALA, LAMBDA, LASTMSG, LCALL, LCALLW, LE, LENGTH, LET, LIST, LISTTOARRAY, LOOP, LT, MACADDR, MAKEAQ, MAXVNODES, MEDIA, MEMORY, METHOD, METHOD, METHOD, MKTHREAD, MOD, MSGMATCH, MSGSWITCH, MSLEEP, MUL, MULTIMSGMATCH, MULTISEND, NADDR, NAME, NCONC, NE, NET, NETCLASS, NETGET, NETIF, NETIFFIT, NETMASK, NETPUT, NETSET, NIDISKIMAGE, NIKERNEL, NIL, NIPXELOADER, NOATEXIT, NODE, NODECLASS, NODESET, NOP, NOT, NUM, NUM, OPTION, OSTYPE, PARTITION, PORT, PRINT, PRINTNET, PRINTNODE, PROGRAM, PROJECT, PUTAEQ, PUTMAEQ, PUTMEQ, PXELOADER, QUOTE, RAND, RBHFILE, REAL, RECV, RECVA, REPEAT, RMANAGER, RNAME, ROUTE, ROUTEQ, SAVENODEINFO, SCENARIO, SEND, SET, SETALQ, SETAQ, SETQ, SETTRACE, SETUPTIMEOUT, SLEEP, SPARE, SPARENODEMIN, SPARENODERATIO, SRAND, STDERR, STDERRAPP, STDERRJOINSTDOUT, STDIN, STDOUT, STDOUTAPP, STRING, STRINGTOARRAY, STRINGTOLIST, STRUCT, SUB, SWCMD, SWCONF, SWEEPNODESET, SWTYPE, SYNC, T, TFTPDIR, TFTPDMAN, THREAD, THREAD, TIME, TIMEOUT, TOINT, TOSTRING, TOTAL, TYPE, UNDEF, UNSETTRACE, USER, VIA, VNTYPE, WAIT, WAKE, WAKEWAIT, WARM, WHILE, WOLAGENT, XPACK, _DEBUGPRINT

Part V

Protocol Manuals

20 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.

20.1 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 favarit resources for its experiment with search.

20.2 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) Expeirment Node Configuration Driver: same as ED.

20.3 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 expriment 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|---+
+-------+      +-------+
```

20.4 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+---+-------+
                         |    x
driver        manager    |    y
+------+ ERRP +-------+   |
|orange|------|  foo  |---+
+------+      +-------+
              x
```

20.5 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

20.6 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

20.6.1 Node Properties

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

required	**owner**	ownership; stores user and project
	name	name;stores string
	state	state. see Section 27.4
recommended	**if-N**	N-th network interface
	n_if	the number of network interfaces
optional	**helth**	methods for helth check; IPMI, ICMP, SNMP and others
	power	methods for power control; IPMI, WoL, SNMP and others
	n_mc	the number of management card
	mc-N	N-th management card; IPMI, iLO and others
	bootdisk	bootdisk; IDE, SCSI and others
	n_cpu	the number of CPU
	cpu-N	N-th CPU
	memsize	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.

20.7 Node State

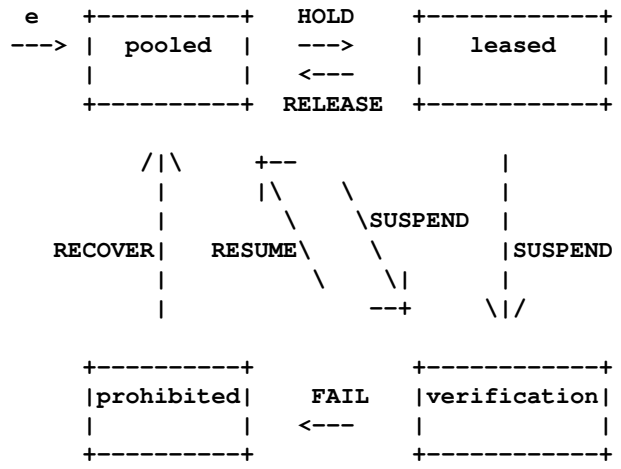
RM sets node's state according to client's request. Following 4 states are defined.

- pooled (idel, free, helth)
- leased (reserved)
- verification (meybe 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 20.11.



20.8 Communication Syntax

20.8.1 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"
```

20.8.2 Request

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

20.8.3 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.

20.9 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
UPDATE	X		update property of resource
RANGE	X		choice range
PERMITRANGE	X	X	assign range v.s. users
PERMITCHECK	X	X	check permition

20.9.1 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.

20.9.2 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>
```

20.9.3 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.

20.9.4 Connect Upper Manager

```
CONNECT <manager-name> <host:port>

        <manager-name> ::= <abstring>
```

20.9.5 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>
```

20.9.6 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 is 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)
```

20.9.7 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 missmatch
```

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
```

20.9.8 Regist New Resource

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

20.9.9 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
  .
```

20.10 Sample of Command Sequence

20.10.1 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
```

20.10.2 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.

```
{conncect}
USER foo
PASSWD bar
PROJECT hawaii
HOLD node n3
{disonncect peer}

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

20.11 Administrator

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

```
RESUME
FAIL
RECOVER

PERMITRANGE
```

20.12 Literals

20.12.1 Interface Media

Following terms are defined for query.

ATM	**ATM; asynchronous transfer mode**
Ethernet	**Ethernet (10Mbps)**
FastEthernet	**Fast Ethernet (100Mbps)**
GigabitEthernet	**Gigabit Ethernet (1Gbps)**
10GigabitEthernet	**10 Gigabit Ethernet (10Gbps)**

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

20.12.2 Interface Type

Following terms are defined for query.

manage	**management network (probably 1 per host)**
experiment	**experiment network**
empty	**open circuit**

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**.

21 Experiment Structure Query Protocol Version 0.5 (ESQP/0.5)

21.1 Overview

ESQP is text and line-oritented 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.

Futhermode, early versions of ESQP are called SSQP.

21.2 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

21.2.1 SYST

request

```
SYST
```

response

```
100 kusa/1.0
```

21.2.2 VER

request

```
VER
```

response

```
100 ESQP/0.4
```

21.2.3 HELP

request

```
HELP
```

response

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

21.2.4 QUIT

request

```
QUIT
```

no response.

21.2.5 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
.
```

21.2.6 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 sintclc003
cl-1 S sintclc004
.
```

response(empty)

```
201 0 OK
.
```

21.2.7 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=
.
```

21.2.8 NETSETLIST

request

```
NETSETLIST
```

response

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

example

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

21.2.9 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
.
```

21.2.10 NETINFO

print information of network.

request

```
NETINFO <name>
```

response

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

example

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

21.2.11 Others

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

```
500 not implemented yet
```

or

```
500 ignore commmand
```

21.3 Name of Entities

Basically, names used ESQP are logical.

21.4 See Also

[1] ERRP/0.6

22 Switch Configuration Protocol Version 1.2 (SWCP/1.2+)

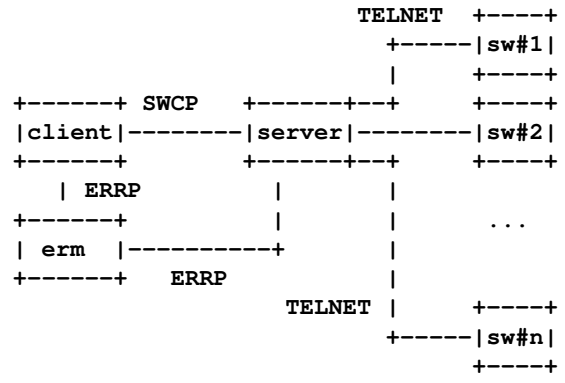
22.1 Overview

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 oritented 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).



22.2 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
XPORTTAGTABLE	print a table of port vs VLAN tags
ISLCARE	change the mode of ISL handling

When command is not found, server replys 400.

22.2.1 SYST

SYST

100 SWMG/1.0

22.2.2 VER

VER

100 SWCP/1.0

22.2.3 QUIT

QUIT

no response.

22.2.4 USER

USER <user>

```
200 Okay
```

22.2.5 PASSWD

PASSWD <passwd>

```
200 Okay
400 Ignore password
```

22.2.6 PROJ

PROJ <proj>

```
200 Okay
```

22.2.7 MKVLAN

MKVLAN <VLAN-ID>

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

22.2.8 RMVLAN

RMVLAN <VLAN-ID>

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

22.2.9 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 joind 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 unttaged 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 simular, if the switch port was joined to untagged VLAN and requested new untagged VLAN, the old untagged VLAN heve to leave before new unttaged 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

22.2.10 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
```

22.2.11 ACTIVATEPORT

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

```
200 done
400 empty user/project
412 you are not permitted to these swports
430 this method is unsupported in the port
```

22.2.12 DEACTIVATEPORT

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

```
200 done
400 empty user/project
412 you are not permitted to these swports
430 this method is unsupported in the port
```

22.2.13 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
```

22.2.14 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.

22.2.15 XPORTTAGTABLE

XPORTTAGTABLE <swport>

```
202 Okay
Last-Fetched: Tue Jul  7 17:13:59 2015

0/3 u:3950|
0/11 u:3950|
0/4 t:3950,4000-4002|
0/20 u:3950|
0/5 u:4003|
0/13 u:3950|
0/8 u:4000|
0/14 u:3950|
0/9 u:3950|
0/15 u:3950|
0/16 t:3950,4000,4002-4003|
0/17 u:3950|
0/26 t:3950|u:4003|
0/18 t:3950,4000-4003|
0/1 t:3950|
0/2 u:4000|
0/10 u:3950|
.
```

410 empty switch name

XPORTTAGTABLE shows the table of port v.s. VLAN tags. Server only knows the status at last communication to the switch. Then this table is added the date/time of last fetched . The format of that is according to HTTP. This command may be official command in near future.

t means tagged VLAN. **u** means untagged VLAN.

22.2.16 NOP

NOP

```
100 okay
```

22.2.17 HELP

HELP

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

22.2.18 ISLCARE

ISLCARE changes mode of ISL.

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

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

22.3 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.

22.4 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)
```

22.5 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(swc:9/7)
S: 200 Okay
C: UNBINDPORT sw:9/10 890
S: 200 Okay
C: QUIT
S: <disconnect client>
```

22.6 See Also

[1] SWMG

[2] ERRP/0.6

[3] ERM

[4] K Language Reference Manual

22.7 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

[2015/07/07] add new command XPORTTAGTABLE

23 Power Configuration Protocol Version 0.1 (PWCP/0.1)

23.1 Overview

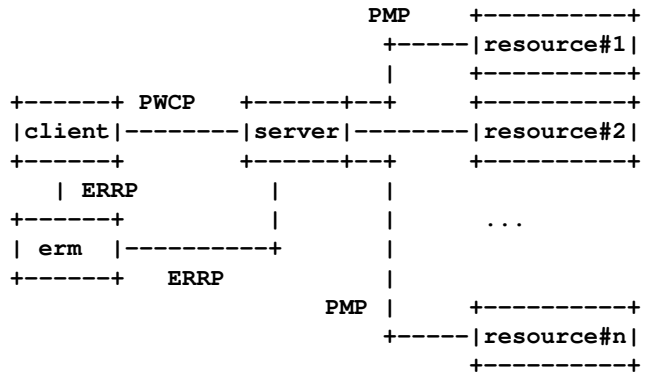
PWCP is a protocol for power manager (PWMG). PWMG aims to configure power of resources. So, the client of PWCP expects to power-on/off or shutdown resources via PWMG.

PWCP have some characteristics following:

- TCP with port#1242
- text-base protocol
- line oritented 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 menagement(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.

23.2 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 forcely
REBOOT	reboot

23.2.1 SYST

SYST

100 PWMG/1.0

23.2.2 VER

VER

100 PWCP/1.0

23.2.3 QUIT

QUIT

no response.

23.2.4 USER

USER <user>

200 Okay

23.2.5 PASSWD

PASSWD <passwd>

200 Okay 400 Ignore password

23.2.6 PROJ

PROJ <proj>

200 Okay

23.2.7 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

23.2.8 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

23.2.9 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

23.2.10 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

23.2.11 NOP

NOP

100 okay

23.2.12 HELP

HELP

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

23.3 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>
```

23.4 See Also

[1] PWMG

[2] ERRP/0.6

[3] ERM

[4] K Language Reference Manual

History

[2009 June] first draft (version 0.1)

Changes

24 Directory Manipulation Protocol Version 0.6 (DMP/0.6)

24.1 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 oritented protocol
- line is terminated by CR(0x0d) LF(0x0a)

24.2 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
STAT	state of file
SYMLINK	make symlink
RMSYMLINK	remove symlink
MKDIR	make directory
REMOVE	remove file
POST	make file

24.2.1 SYST

SYST

100 DMAN/0.4

24.2.2 VER

VER

100 DMP/0.6

24.2.3 QUIT

QUIT

no response.

24.2.4 USER

USER <username>

200 Okay

24.2.5 PASSWD

PASSWD <password>

200 Okay

24.2.6 PROJ

PROJ <projectname>

200 Okay 502 inner error 402 no USER, PASSWD and PROJ 400 ignore password

24.2.7 STAT

```
STAT <file>
        201 found
        X-Mode: 0xa1ff
        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
```

24.2.8 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

24.2.9 RMSYMLINK

RMSYMLINK <file>

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

24.2.10 MKDIR

MKDIR <dir>

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

24.2.11 REMOVE

REMOVE <file>

200 success 400 fail 406 ignore path 410 not found

24.2.12 POST

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

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

24.2.13 NOP

NOP [[message] ...]

100 okay

24.2.14 HELP

HELP

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

24.3 Response

code	description
100	okay
101	okay
200	success
201	found
400	fail
402	no USER, PASSWD and PROJ
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
502	inner error

24.4 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/0.4
C: ver
S: 100 DMP/0.6
C: USER john
S: 200 Okay
C: PASSWD abbey-road
S: 200 Okay
C: PROJ Imagine
S: 200 Okay
C: stat h001.pxe
S: 201 found
S: X-Mode: 0xa1ff
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
```

24.5 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

[2015/07/07] version 0.6; supports USER, PASSWD and PROJ

Changes

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

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

25.1 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 oritented protocol
- line is terminated by CR(0x0d) LF(0x0a)

25.2 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

25.2.1 SYST

SYST

100 wolagent/2.0

25.2.2 VER

VER

100 WOLAP/1.0

25.2.3 QUIT

QUIT

no response.

25.2.4 SEND

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

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

25.2.5 HELP

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

25.3 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>
```

25.4 See Also

[1] ERRP/0.6

[2] ERM

History

[2010 Jan] first draft (version 0.1)

Changes

26 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 extentions. Moreover, partner programs like **fncp** and **ni** use the extention also. This document describes the extention.

26.1 ENCD Reachability

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**. Fundimental communication is established between NA and ENCD by HTTP with special extentions. Figure 26.1 depicts the relationship of them.

Moreover, NA does not know what host is ENCD when it starts. FNCP is designed to solve this problem. By the query to FNCP, NA knows their ENCD. Here NA have 2 communications, first is to ENCD, second is to FNCP. To simplify, we designed anti-ENCP communication as redirection. Then 2 communications has no different. NA can reach ENCD by retry of same type retrieval. ENCDs have to regist the relationship of theyself and their target (NAs) into FNCP before NA starting. Figure 26.2 and Figure 26.3 depicts the relationship of 3 entities and the timing of these communication.

26.2 X-NSS Fields

Because NA–ENCD communication is based on NSS, this disk image distribution mechanism use HTTP header which starts "X-NSS". NSS stands "Node Supervisor System" [1] . It is previous disk image distribution program for StarBED. Table 26.1 shows X-NSS headers.

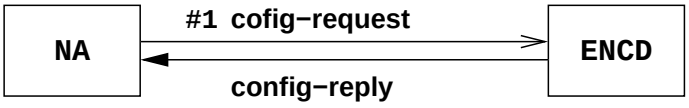


Figure 26.1: NA-ENCD relationship

[1] Misumi and etc:"NSS:Node Supervisor System on StarBED", (Japanese)IPSJ SIG Technical Report 2004-DPS-116, pp.95–100, Jan 2004

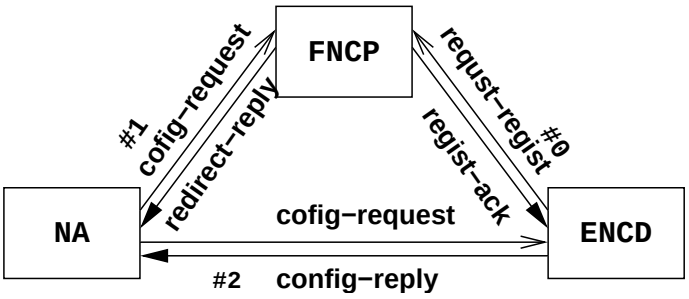


Figure 26.2: NA-ENCD-FNCP relationship

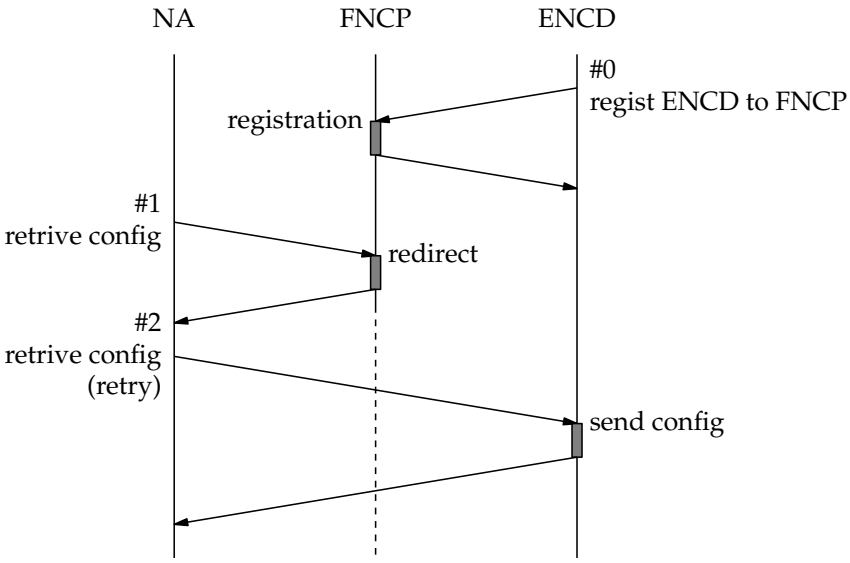


Figure 26.3: Timing Chart of NA-FNCP-ENCD Communication

Table 26.1: X-NSS Headers

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

Table 26.2: Node Configuration Commands

commands	description
no-reboot	disable auto reboot
sleep <second>	sleep
report <URL>	set URL for reporting
upload <local-device> <network-location> [offset] [length]	upload to remote
download <network-location> <local-device>	download to local

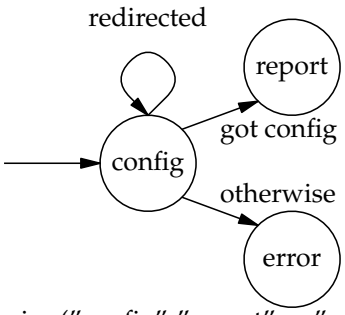
26.3 Node Configuration — Commands in HTTP-body

Node cofiguration is a kind of script. It is stored into body of HTTP communication. NA evaluates configuration step by step Table 26.2 shows a list of commands. After evaluation, NA excutes **`reboot`** automatically. Exception of reboot is ”no-reboot” command. In that case, NA terminates normally.

Since the configuration is sequence of commands, you can download both MBR and some partition in single configuration.

26.4 NA's State

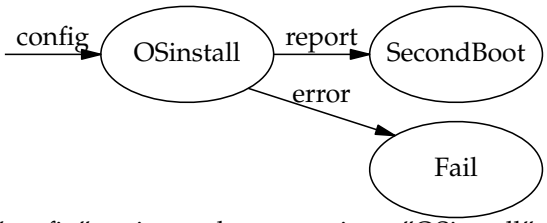
Field 'X-NSS-State' have 3 string ”config”, ”report” and ”error.” Transit is following. When configuration retrieval is redirected, tansit ”config” to ”config.” Otherwise succes of configration shifts it to ”report.” Fail of that shifts it into ”error.”



Then, ENCD got request twice (”config”-”report” or ”cofing”-”error”) per node.

26.5 Node Phase in ENCD

ENCD stores many phases per node. Phase is simular state. Here we introduce only 3 phases of them. See Section 1.8 and Section 1.9 for OS intallation. According to NA's request, phase is transit.



When state-”config” arrives, phase transit to ”OSinstall”, ENCD recognize that the node downloads disk image. Usually the node phase transits to ”SecondBoot” by state-”report”. If error is happend, state-”error” arrives and phase transits to ”Fail”. You may look these strings into ENCD trace/debug messages.

26.6 Example

NA trys to retrive a node configuration.

```
GET /somepath HTTP/1.0\r\n
User-Agent: NI/0.1\r\n
X-NSS-Version: $Revision: 2896 $\r\n
X-NSS-Mission-ID: some-mid\r\n
X-NSS-Agent-ID: some-aid\r\n
X-NSS-Session-ID: some-sid\r\n
X-NSS-State: config\r\n
\r\n
```

ENCD send a node configuration.

```
HTTP/1.0 200\r\n
Server: wipeout/1.5alpha\r\n
X-NSS-Version: $Revision: 2921 $\r\n
Content-Type: text/plain\r\n
\r\n
# disk download configuration as wipeout\r\n
use-syslog\r\n
download http://usr:pass@server/path/disk.img /dev/sd1a\r\n
```

You can download parition and MBR togather.

```
HTTP/1.0 200\r\n
Server: wipeout/1.5alpha\r\n
X-NSS-Version: $Revision: 2921 $\r\n
Content-Type: text/plain\r\n
\r\n
# disk download configuration as wipeout\r\n
download http://usr:pass@server/path/mbr.img /dev/sd1\r\n
download http://usr:pass@server/path/disk.img /dev/sd1a\r\n
```

FNCP may send a redirection instead of ENCD.

```
HTTP/1.0 302 Go Driver\r\n
Server: FCNP/0.1\r\n"
X-NSS-Version: $Revision: 2862 $\r\n
Location: http://somehost:8192/\r\n
\r\n
```

27 DHCPD Backend Protocol Version 0.1 (BP/0.1)

This article describes BP (Backend Protocol) version 0.1. The protocol is used between our DHCP (Dynamic Host Configuration Protocol) server and its controller.

27.1 Overview

27.2 Pattern of Command/Response in This Document

syntax of command

command <*argument*> [<*optional*>]

syntax of response

200 <number> Okay

201 <number> List
<host1>
<host2>
.

eample of response

```
201 3 hosts
exppc001
exppc002
exppc003
.
```

27.3 Connections

27.4 Node State

27.5 Communication Syntax

27.5.1 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"
```

27.5.2 Request

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

27.5.3 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.

27.6 Commands

BP have commands in following table. Its server replys with response code. If commands are unknown, the server reply 502. Reserved but not implemented command cause 504.

- 502 Unknown Commmand
- 504 Not Implemented

command	description
misc	
SYST	print system version
VER	print porocol version e.g., DHCCP/0.7
NOP	no operation
HELP	print help
USER	set user
PASS	set password
PROJ	set project
host	
HL	print all (list) host (name)
HLA	print all (list) active host (name)
host activation	
HAP	print host activation
HA[M]S	set host activation with all interface
HA[M]U	unset host activation with all interface
HAI(A/M/E)[M]S	set host activation with interface for all/management/experiment
HAI(A/M/E)[M]U	unset host activation with interface for all/management/experiment
HAIXS	set host activation with interface MAC specified
HAIXU	unset host activation with interface MAC specified
host configuration	
HCP	print host configuration
HCCP	print host common configuration
host information	
HIP	print host information
HISWEEP	sweep host information from specified RM
resource manager	
RML	print resource managers
MAC-base file mapping	
MFL	print all MAC address-base file mapping
MF(M)P	print MAC address-base file mapping
MF(M)S	set MAC address-base file mapping
MF(M)U	unset MAC address-base file mapping
MFHI(A/M/E)(M)S	set HOST interface specified for all/man./exp. MAC address-base file mapping
MFHI(A/M/E)(M)U	unset HOST interface specified for all/man./exp. MAC address-base file mapping
IP-base file mapping	
IFL	print all IP address-base file mapping
IF(M)P	print IP address-base file mapping
IF(M)S	set IP address-base file mapping
IF(M)U	unset IP address-base file mapping
IFHI(A/M/E)(M)S	set HOST interface specified for all/man./exp. IP address-base file mapping
IFHI(A/M/E)(M)U	unset HOST interface specified for all/man./.exp IP address-base file mapping

command	description ($1 \leq i \leq n$, $1 \leq j \leq m$, $m \geq n$)
@RT2(host)	
HL	print all (list) host (name) ; @Hall
HLA	print all (list) active host (name) ; @Hi $\|state = active$
@RT2(host activation)	
HAP	print host activation ; @Mj @Hi
HA[M]S	set host activation with all interface
HA[M]U	unset host activation with all interface
HAI(A/M/E)[M]S	set host activation with interface for all/management/experiment
HAI(A/M/E)[M]U	unset host activation with interface for all/management/experiment
HAIXS	set host activation with interface MAC specified
HAIXU	unset host activation with interface MAC specified
@RT2(host configuration)	
HCP	print host configuration ; @Hi @Ci
/HC[M]CLEAR	clear host configuration
/HCALLCLEAR	clear all host configuration
/HCUPDATE	update all host configuration
/HCP[M]ADD	add part of host configuration
/HCP[M]REPLACE	replace part of host configuration
HCCP	print host common configuration; @Cc
@RT2(host information)	
HIP	print host information ; @Si
/HIUPDATE	update host information in this connection
/HI[M]FIND	update HOST information from RM
HISWEEP	sweep host information from specified RM
@RT2(resource manager)	
RML	print resource managers
/RMADD	add resource manager
/RMDEL	del resource manager
@RT2(host binding)	
//HBL	print all host bindings; MAC–HOST
//HB[M]P	print host binding
//HB[M]S	set host binding; MAC–HOST
//HB[M]U	unset host biding

44 commands. Slash(/) started commands are reserved.

Misc

command	description
@RT2(misc)	
SYST	print system version
VER	print porocol version e.g., DHCCP/0.7
NOP	no operation
HELP	print help
USER	set user
PASS	set password
PROJ	set project

7 commands

HELP

101 HELP

NOP

100 NOP

SYST

100 <system>/<version>

```
100 NeoDHCPD/0.1
```

VER

100 <protocol>/<version> [<protocol>/<version>]

```
100 BP/0.1 DMP/0.4
```

USER <*user*>

200 Okay

210 Send Password

PASS <*password*>

200 Okay

440 Auth Error

PROJ <*project*>

200 Okay

440 Auth Error

Host List

HL

201 Host List
<host1>
<host2>
.

```
201 Host List
devpc001
devpc002
.
```

HLA

201 Active Host List
<host1>
<host2>
.

Host Activation

HAL

201 Host Activation List
.

500 Lock Error

HAIAS
HAIMS
HAIES

402 Not Found Host

404 Not Found NIC

200 <num> set/unset

406 Not Found MAC address

HAIXS

402 Not Found Host

200 <num> set/unset

406 Not Found MAC address

HAIAU
HAIMU
HAIEU

402 Not Found Host

404 Not Found NIC

200 <num> set/unset

406 Not Found MAC address

HAIXU

402 Not Found Host

200 <num> set/unset

406 Not Found MAC address

Host Configuration

Resource Manager

RML

201 <num> RMs
.

Host Information

HIP <*name*>

201 host information list
.

402 Not Found Host

500 Lock Error

File Mapping

MFP

MFS

MFU

MFIAS

MFIMS

MFIES

command	description
@RT2(MAC-base file mapping)	
MFL	print all MAC address-base file mapping
MF(M)P	print MAC address-base file mapping
MF(M)S	set MAC address-base file mapping; MAC–file
MF(M)U	unset MAC address-base file mapping
MFHI(A/M/E)(M)S	set HOST interface specified for all/man./exp. MAC address-base file mapping; H
MFHI(A/M/E)(M)U	unset HOST interface specified for all/man./exp. MAC address-base file mapping
@RT2(IP-base file mapping)	
IFL	print all IP address-base file mapping
IF(M)P	print IP address-base file mapping
IF(M)S	set IP address-base file mapping; IP–file
IF(M)U	unset IP address-base file mapping
IFHI(A/M/E)(M)S	set HOST interface specified for all/man./exp. IP address-base file mapping; HOS
IFHI(A/M/E)(M)U	unset HOST interface specified for all/man./.exp IP address-base file mapping
38 commands	

27.7 Response Code Summary

code	description
100	*system/version*
100	NOP
100	*protocol/version*
101	HELP
200	*number* RMs
200	*number* set/unset
200	Okey
201	Active host list
201	Host Activation List
201	Host Information List
201	Host List
210	Send Password
400	Ignore RM Name
402	Not Found Host
404	Not Found NIC
406	Not Found MAC Address
440	Auth Error
500	Lock Error
502	Unknown Command
504	Not Implemented