The Laptop-net Manual

Edition 1.3
23 February 2002

by Chris Hanson

Introduction

The laptop-net package is a configuration mechanism for laptop computers that have built-in network interfaces. It extends the standard GNU/Linux networking administration tools to facilite moving the laptop between several different networking environments. In addition, for computers with supported network interfaces, it also automatically detects network cable insertion and removal.

The package is built around two basic concepts, which will be described below in greater detail. First, there is configuration information that describes how the network should be initialized for a particular location. This information is organized as a set of named *schemes*, each of which describes the network configuration for a particular location. For example, a typical network configuration would specify an IP address, an associated netmask, a gateway, etc. Second, there is configuration information that specifies higher-level details, such as a copy of '`/etc/printcap`' describing the printers available at that location. This higher-level information is organized as a set of named *profiles*, which are indexed by IP-address patterns. Together, these two mechanisms provide a flexible means for configuring the network environment.

The profile mechanism is modeled on a similar mechanism in the `boot-profiles` package, written by Al Stone. The mechanism here is my attempt to integrate this functionality directly into the network-management code. The schemes mechanism is modeled on the schemes mechanism of `pcmcia-cs`, written by David Hinds.

This package has been designed and tested on Hewlett-Packard OmniBook model 6000 and 500 computers, which were provided by Hewlett-Packard as part of their support for the development of this software. Although it has not yet been tried on other computers, the package is designed to be general and should port easily.

The package has been tested *only* on systems running Debian GNU/Linux. And unfortunately one key piece of the code is specific to Debian, which means that porting this code to other GNU/Linux systems requires a little thought. However, the Debian-specific code has been isolated as well as possible from the rest of the package.

Configuration File

The configuration file defines some global configuration information for the program, and is named '`/etc/default/laptop-net`'. It is a Bourne shell script that is expected to set some shell variables defining the network configuration.

`MODULE_NAME`
: should be set to the name of the kernel module that contains the driver for the built-in network interface. If this is set to anything other than an empty string, the laptop-net package will load and unload the module as needed.

  Note that you *must* build the driver as a module, and set this variable to its name, in versions of the kernel prior to 2.4. You must also do this if the driver in the 2.4 kernel doesn't do power management properly. It is always safe to do this, but if the driver and chipset support power management, it is better to leave `MODULE_NAME` unset.

At present this has only been tested with '`MODULE_NAME=3c59x`'. This is always necessary on the HP OmniBook 500 and 6000 computers because the power management code in the 2.4 kernel driver isn't fully functional for the chipset used in these computers.

`MII_SUPPORTED`
: should be set to '`"yes"`' if the network-interface driver supports MII, and '`"no"`' otherwise. Link-beat detection is only available if `MII_SUPPORTED` is '`"yes"`'.

Schemes

The laptop-net package organizes network-configuration information as a set of named *schemes*. Each scheme normally defines the network configuration for a particular location. But a scheme can also specify that *Dynamic Host Configuration Protocol* (DHCP) is to be used to obtain the network configuration; such a scheme can be used to configure the network at any location that provides DHCP service.

At any time, there is a *selected scheme*. On Debian systems, the selected scheme can be printed by running the command

```
/etc/init.d/laptop-net scheme
```

The initial scheme is named '`default`'. A new scheme *name* may be selected by running the command

```
/etc/init.d/laptop-net scheme name
```

Alternatively, the scheme can be automatically selected by specifying one or more IP addresses that are associated with a particular scheme. Then, when the laptop is connected to a network, it probes the local subnet for those addresses, and if one of them is found, it selects the associated scheme.

Schemes File

Schemes are defined by editing a configuration file named '`/etc/laptop-net/schemes`'. This file is a Bourne shell script that is expected to set some shell variables defining the network configuration. Some shell variables are passed in to the configuration file when it is executed. The script should use these variables to determine what to do:

`SCHEME`
: is set to the name of the selected scheme. This might be something like '`default`', '`home`', or '`office`'. The specific names used here are defined by you when you edit the configuration file.

`INTERFACE`
: is set to the name of the built-in network interface. This is almost always the string '`eth0`'.

There are many variables that may be set by the configuration file, but they are divided into groups according to the basic configuration type. There are three different configuration types, as follows.

1. If no other variables are set, then the network interface is disabled. In the default configuration file supplied with this package, the '`offline`' scheme is configured this way.
2. If the `DHCP` variable is set, then the network interface is to be configured using the DHCP protocol. This configuration type is referred to as *dynamic configuration*. This is the preferred type, because it trivializes configuration and considerably enhances the system's portability.
3. If the `ADDRESS` and `NETMASK` variables are set, then the network interface is to be configured manually. This configuration type is referred to as *static configuration*.

Here are the variables specifically for dynamic configuration:

`DHCP` should be set to '`yes`' if you want the network interface to be configured using DHCP. Set this to '`no`', or leave it unset, if you don't want to use DHCP.

Here are the variables specifically for static configuration:

`ADDRESS` should be set to an IP address. Example: '`ADDRESS=192.168.1.17`'.

`NETMASK` should be set to the netmask for the subnet that corresponds to the IP address in `ADDRESS`. Example: '`NETMASK=255.255.255.0`'.

`GATEWAY` should be set to the gateway address for the subnet that corresponds to the IP address in `ADDRESS`. Example: '`GATEWAY=192.168.1.1`'.

`BROADCAST`
: should be set to the broadcast address for the subnet that corresponds to the IP address in `ADDRESS`. Normally this isn't needed, because the broadcast address can be computed from `ADDRESS` and `NETMASK`. Example: '`BROADCAST=192.168.1.255`'.

`NETWORK` should be set to the network address for the subnet that corresponds to the IP address in `ADDRESS`. Normally this isn't needed, because the network address can be computed from `ADDRESS` and `NETMASK`. Example: '`BROADCAST=192.168.1.0`'.

The following variables are required for static configuration. (Only one of `DOMAIN` and `SEARCH` need be set.) For dynamic configuration, they are optional and usually not needed.

`DOMAIN` should be set to the local domain name. Example: '`DOMAIN=ai.mit.edu`'.

`SEARCH` should be set to a space-separated list of domain names. These names are searched, in order, when looking up a DNS name that doesn't contain any dots. Note that you must put double-quotes around the right-hand side of the variable assignment; otherwise the space characters will confuse the shell. Example: '`SEARCH="ai.mit.edu mit.edu"`'.

`NAMESERVERS`
: should be set to a space-separated list of IP addresses for DNS servers. Note that you must put double-quotes around the right-hand side of the variable assignment; otherwise the space characters will confuse the shell. Example: '`NAMESERVERS="192.168.1.57 192.168.4.71"`'.

IP Map File

Laptop-net has the ability to automatically choose the scheme by probing the network. This is accomplished by associating one or more IP addresses with some or all of your schemes. When the computer is connected to a network, it probes the network to see which of the IP addresses are located on the local subnet. If one of the IP addresses is found, the associated scheme is selected.

The association between scheme names and IP addresses is stored in a configuration file named '`/etc/laptop-net/ip-map`'; this file is normally called the *IP map*. The IP map consists of a number of one-line entries. Each line starts with the name of a scheme in the first column, and is followed by one or more IP addresses, separated by space or tab characters. Blank lines are allowed, as are comments that begin with the '`#`' character and extend to the end of the line.

If the IP map file contains one or more entries, then the network is probed by sending out an *Address Resolution Protocol* (ARP) request packet for each IP address appearing in the map. If an ARP reply packet is received, its source IP address is looked up in the map, and the associated scheme is selected. (Only the first reply is used; subsequent replies are ignored.) If no replies are received, then one of two things happens. If the selected scheme was manually selected by the user, then it is not changed. However, if the selected scheme was automatically selected by a previous network probe, then the '`default`' scheme is selected.

If the IP map file contains no entries, or if it doesn't exist, then the network is not probed and the selected scheme is not automatically changed.

A Schemes Example

Here is a complete example to give you an idea of the overall structure of the schemes file:

```
case $SCHEME in
home)
    # At home we use static configuration.
    ADDRESS=192.168.1.6
    NETMASK=255.255.255.0
    GATEWAY=192.168.1.254
    SEARCH="ai.mit.edu mit.edu"
    NAMESERVERS="24.128.44.6 24.128.52.6 24.128.1.80"
    ;;
offline)
    # The offline scheme turns the network off.
    ;;
*)
    # All other schemes use dynamic configuration.
    DHCP=yes
    ;;
esac
```

And here is an associated IP map file:

```
home 192.168.1.1 192.168.1.2
office 192.168.2.1 192.168.2.2 192.168.2.3
```

Profiles

The second layer of configuration is the *profile* layer, which adapts the software configuration of the computer to match the network environment. For example, if you have profiles for the home and for the office, you will usually need to have a different copy of the '`/etc/printcap`' file for each location, because each has different printers.

A profile consists of some files that are copied into the file system, and some shell scripts that are run when the profile is selected or deselected. Each profile also has a set of patterns that specify the IP addresses that this profile is associated with. Whenever the network is reconfigured, either to turn it on, turn it off, or change the network address, laptop-net searches the available profiles to find one that matches. If the new profile is different from the current one, then the current one is deselected and the new one is selected.

The behavior of profiles could have been integrated into the schemes layer. But profiles are implemented as a separate layer because it makes them more flexible. This is because a named scheme that uses dynamic configuration can have different IP addresses, and each of those addresses might map to a different profile. Or there might be two different schemes that map to the same profile, for example when you have a statically-configured scheme and a dynamically-configured scheme that can each be used at the same network location.

The Profiles Directory

Profiles are defined by creating directories in the profiles directory, which is named '`/etc/laptop-net/profiles`'. A profile is a subdirectory of the profiles directory. The first character of the name of the subdirectory must be either a lower-case letter or a digit; other subdirectories are ignored. Some examples of profile subdirectories are

```
/etc/laptop-net/profiles/home
/etc/laptop-net/profiles/office
/etc/laptop-net/profiles/default
```

A profile subdirectory itself contains other files and subdirectories, each with a specific purpose, as follows.

'`patterns`'
: This file contains patterns that match IP addresses, with one pattern on each line. These patterns are ordinary shell *filename-globbing* patterns, where the '`*`' character matches any substring and the '`?`' character matches any character. So, for example, '`18.*.*.*`' will match any IP address starting with '`18.`', while '`*.*.*.*`' will match any IP address at all. See below for further information on how the patterns are used to select a profile.

'`files.d/`'
: This subdirectory contains files that are to be copied into the file system when the profile is selected. The files in this directory will be copied directly into the root directory of the machine. So, for example, the file '`files.d/etc/printcap`' will be copied to '`/etc/printcap`'.

The files are copied using the command

```
(cd files.d; /bin/cp -pr * /)
```

which means that all file permissions are preserved, and that the contents of subdirectories are also copied. This also means that files starting with the character '`.`' will not be copied.

'`before-select`'
: This file is an arbitrary program to be run before the profile is selected. It is called with two arguments. The first argument is the file name of the profiles directory, and the second is the name of the profile being selected. For example,

```
before-select /etc/laptop-net/profiles home
```

'`after-select`'
: This file is an arbitrary program to be run after the profile is selected. Like '`before-select`', it is called with two arguments, the file name of the profiles directory and the name of the profile being selected.

'`deselect`'
: This file is an arbitrary program to be run after the profile is deselected. Like '`before-select`', it is called with two arguments, the file name of the profiles directory and the name of the profile being deselected.

'`rc.d/`'
: This subdirectory contains symbolic links to *init scripts* that start and stop operating-system services. These links provide a simple way to manage these services, although of course you could also do this using the '`after-select`' and '`deselect`' programs. See below for more discussion of the '`rc.d/`' directory.

Profile Management

The management of profiles consists of two processes. First is the *profile matching* process, in which the set of available profiles is searched to find one that matches the current IP address. Second is the *profile selection* process, in which the current profile is deselected and the new one selected.

Profile matching follows a simple search pattern. First, the current IP address is determined. This can have one of three values:

1. A valid IP address consisting of four decimal numbers separated by periods, e.g. '`192.168.1.17`'.
2. The word '`down`', meaning that the network interface is down and doesn't have a valid IP address.
3. The word '`unknown`', meaning it was impossible to determine the state of the network interface. (For example, the driver for that interface might not be loaded.)

Next, the profiles are searched in alphabetical order by name. Each profile is tested by examining the contents of its '`patterns`' file. (If the file doesn't exist or can't be read, the profile is ignored.) Each pattern in the file is tested against the IP address, and if one of the patterns matches, then this profile is considered to match the address.

This search pattern can be exploited to create a *default profile*, which is used when nothing else matches. To do this, create a profile with a name that will always sort after all other profile names (e.g. '`zzz-default`'), and give it a single pattern '`*`'.

Once the new profile has been determined by the matching process, it must be selected. (Note, however, that nothing is done if the new profile is the same as the current one.) The selection process has two phases. First, if there is a current profile, we perform the following steps to *deselect* it:

1. If the '`rc.d/`' directory exists, any services specified in it are stopped. See below for further details of this process.
2. If the '`deselect`' program exists, it is executed.

Next, if there is a new profile, we perform the following steps to *select* it:

1. If the '`before-select`' program exists, it is executed.
2. If the '`files.d/`' directory exists, any files it contains are copied to the root directory.
3. If the '`rc.d/`' directory exists, any services specified in it are started. See below for further details of this process.
4. If the '`after-select`' program exists, it is executed.

It is worth mentioning that the selection process provides for situations in which there is no profile. Thus, when there is no current profile, the new profile is still correctly selected. Likewise, when there is a current profile but no new one, the current one will still be deselected.

Managing System Services

The one remaining complexity is the '`rc.d/`' directory, which requires a little background. Most GNU/Linux systems employ a mechanism called *SysV init scripts*, in which each system service has a corresponding shell script that is used to turn it on and off. The full details of how this works are unimportant, but in a nutshell, the script is called with the argument '`start`' to start the service, and '`stop`' to stop it. On Debian systems, these scripts are normally stored in the directory '`/etc/init.d/`', but on some other systems (e.g. Red Hat Linux) they are stored in '`/etc/rc.d/init.d/`'.

The '`rc.d/`' directory contains symbolic links to these scripts, with special names that control how they are started and stopped, and in what order. Each name consists of a letter, either '`S`' or '`K`', followed by two decimal digits, followed by the name of the script. Examples of these names for the '`lpd`' script would be '`S20lpd`' and '`K20lpd`'.

The names are interpreted as follows. When selecting a profile, we start all the services in the '`rc.d/`' directory. This means we find all of the links starting with '`S`', and in standard dictionary order, call each one with the argument '`start`'. When deselecting a profile, we find all the links starting with '`K`' and call them in order with the argument '`stop`'.

Note that the operating system has similar links stored in directories called '`/etc/rcN.d/`', where N is a digit. (The directories are called '`/etc/rc.d/rcN.d/`' on Red Hat systems.) Normally you should use the same link names as those that appear in the operating system's directories, since the system's maintainers have already determined appropriate orders for starting and stopping services. Furthermore, for best operation you should delete the operating system's links, since those services will now be controlled by laptop-net instead. (On Debian systems, the links are managed with the program `update-rc.d`; see that program's man page for instructions on how to remove the links.)

Using Profiles with PCMCIA

The profile mechanism also can be used with PCMCIA devices. However, in order for this to work, it's necessary to edit the `pcmcia-cs` configuration files. Specifically, the file '`/etc/pcmcia/network.opts`' must be edited, and the following functions defined there.

```
start_fn ()
{
    /usr/share/laptop-net/profile-change $DEVICE
}

stop_fn ()
{
    /usr/share/laptop-net/profile-change $DEVICE down
}
```

If these functions already exist, the calls to '`profile-change`' must be added into them. Note that it is important that '`profile-change`' be called *every* time that this script is invoked: otherwise the profile mechanism might get out of sync with the network.

Another consideration occurs because there might be an internal network interface and a PCMCIA network interface that are both active at the same time. If there is a possibility of such a situation, you must make sure that both devices are using the same profile! Otherwise the profile mechanism can get confused.

A Profiles Example

The profile mechanism is somewhat complicated, so let's look at a complete example to see how it might be used. First, here is a listing of the files in our example profiles directory.

```
/etc/laptop-net/profiles:
total 20
drwxr-xr-x    2 root  4096 Jun 11 11:34 SHARED
drwxr-xr-x    5 root  4096 Jun  8 23:33 home
drwxr-xr-x    2 root  4096 Jun  8 23:34 offline
drwxr-xr-x    5 root  4096 Jun 11 11:36 work
drwxr-xr-x    3 root  4096 Jun  8 23:34 zzz-default
```

This shows four profile subdirectories, named '`home`', '`offline`', '`work`', and '`zzz-default`'. The subdirectory named '`SHARED`' isn't a profile, because it begins with an upper-case letter. We'll see later how this is used.

The simplest of these profiles is '`offline`':

```
/etc/laptop-net/profiles/offline:
total 4
-rw-r--r--    1 root    13 Jun  7 23:43 patterns
```

Here is the contents of the file '`offline/patterns`':

```
down
unknown
```

This profile catches the case where the network is not running. It doesn't do anything in that case. Note that we could have omitted this profile and the end result would be exactly the same.

Next, let's look at one of the two primary profiles, '`home`':

```
/etc/laptop-net/profiles/home:
total 24
drwxr-xr-x    2 root  4096 Jun 11 11:36 LOCAL
-rwxr-xr-x    2 root    80 Jun  8 15:25 after-select
-rwxr-xr-x    2 root   378 Jun  8 15:09 before-select
drwxr-xr-x    3 root  4096 Jun  8 23:30 files.d
-rw-r--r--    1 root    12 Jun  7 22:59 patterns
drwxr-xr-x    2 root  4096 Jun 11 11:39 rc.d
```

This profile uses most of the features of the profile mechanism: the '`before-select`' and '`after-select`' scripts, the '`files.d/`' directory, and the '`rc.d/`' directory. It additionally has a '`LOCAL/`' directory that is not directly used by the profile mechanism, but is referred to by one of the scripts. This profile matches the following '`patterns`':

```
192.168.1.*
```

When this profile is selected, the '`before-select`' script is run. This script uses information from the '`SHARED`' and '`LOCAL`' directories to create a file '`/etc/fstab`'. This is saved in the '`files.d/`' directory, where it will subsequently be copied into the root file system.

```
#!/bin/sh
PROFILE_DIR="$1"
PROFILE="$2"
FSTAB="$PROFILE_DIR/$PROFILE/files.d/etc/fstab"
FSTAB_SHARED="$PROFILE_DIR/SHARED/fstab"
FSTAB_LOCAL="$PROFILE_DIR/$PROFILE/LOCAL/fstab"
if [ ! -f "$FSTAB" ] \
        || [ "$FSTAB_SHARED" -nt "$FSTAB" ] \
        || [ "$FSTAB_LOCAL" -nt "$FSTAB" ]; then
    cat "$FSTAB_SHARED" "$FSTAB_LOCAL" > "$FSTAB"
fi
```

Other files are also stored in the '`files.d/`' directory:

```
/etc/laptop-net/profiles/home/files.d:
total 4
drwxr-xr-x    2 root  4096 Jun 11 11:35 etc

/etc/laptop-net/profiles/home/files.d/etc:
total 4
-rw-r--r--    1 root   951 Jun 22  1999 printcap
```

Some system services are started and stopped from the '`rc.d/`' directory:

```
/etc/laptop-net/profiles/home/rc.d:
total 0
lrwxrwxrwx    1 root    15 Jun 11 11:39 K20lpd -> /etc/init.d/lpd
lrwxrwxrwx    1 root    19 Jun 11 11:39 K22ntpdate -> /etc/init.d/ntpdate
lrwxrwxrwx    1 root    15 Jun 11 11:39 K23ntp -> /etc/init.d/ntp
lrwxrwxrwx    1 root    15 Jun 11 11:39 S20lpd -> /etc/init.d/lpd
lrwxrwxrwx    1 root    19 Jun 11 11:39 S22ntpdate -> /etc/init.d/ntpdate
lrwxrwxrwx    1 root    15 Jun 11 11:39 S23ntp -> /etc/init.d/ntp
```

Then we run the '`after-select`' script, which changes the `pcmcia-cs` "scheme" to match the current profile:

```
#!/bin/sh
PROFILE_DIR="$1"
PROFILE="$2"
/sbin/cardctl scheme "$PROFILE"
```

And that's it for the '`home`' profile. The '`work`' profile is nearly identical, except for some of the contents of the files in '`files.d/`', and of course the '`patterns`' file:

```
18.*.*.*
128.52.*.*
```

Finally, we have the '`zzz-default`' profile, which is a kind of catch-all profile for locations that don't have specific profiles defined for them:

```
/etc/laptop-net/profiles/zzz-default:
total 8
-rw-r--r--    1 root     8 Jun  7 23:46 patterns
drwxr-xr-x    2 root  4096 Jun 11 11:39 rc.d
```

'`zzz-default`' has a '`patterns`' file that matches any IP address. The profile's name starts with '`zzz`' so that this pattern isn't tested until all other profiles have been tried.

```
*.*.*.*
```

This profile isn't specialized in any way, except to turn the '`ntp`' service on when it is selected:

```
/etc/laptop-net/profiles/zzz-default/rc.d:
total 0
lrwxrwxrwx    1 root    19 Jun 11 11:39 K22ntpdate -> /etc/init.d/ntpdate
lrwxrwxrwx    1 root    15 Jun 11 11:39 K23ntp -> /etc/init.d/ntp
lrwxrwxrwx    1 root    19 Jun 11 11:39 S22ntpdate -> /etc/init.d/ntpdate
lrwxrwxrwx    1 root    15 Jun 11 11:39 S23ntp -> /etc/init.d/ntp
```

Link-beat Detection

Another major feature of this package is automatic detection of the ethernet *link beat*. The link beat, when active, indicates that the physical ethernet connection is operating correctly; in other words, that the network cable is plugged into both the computer and the hub, and that both network interfaces are functioning. Unplugging the cable or turning off the network hub will stop the link beat.

On network interfaces that support link-beat detection, laptop-net monitors the link beat. When it detects that the link beat has stopped, it brings the network interface down. If the link beat later starts again, laptop-net brings the network interface back up. In either case, when the network interface goes up or down, laptop-net reinitializes the network and potentially switches to another profile.

In practice, the reason why this is useful is that it allows you to move the computer around by unplugging the network cable, moving the computer to another location, then plugging it back in. The software will detect these events and manage the interface as needed. Even better, if the machine uses dynamic configuration (and both of the networks support it), when the machine is plugged into the second network, it will get a new IP address and automatically select a new profile. So, for example, you can unplug the computer at

the office, go home, plug it in, and continue working without any manual intervention to tell the computer where it is.

The one caveat for this support is that you should be fairly certain that any services currently using the network are stopped *before* you unplug the network cable. This includes any open network connections or mounted NFS volumes. Otherwise, you'll be left with dangling connections; and once the cable is unplugged it is difficult to shut these connections down cleanly. (One simple way to detect open TCP connections is to run the command '`netstat -t`' and look for connections whose '`State`' is '`ESTABLISHED`'.)

The other potential problem with link-beat detection is that it is not supported by all network drivers; it depends on a feature called the *Media Independent Interface* (MII). However, MII is supported by most recent network cards, and most importantly, it is supported by the cards currently in use by most laptops. (The following Linux 2.4.5 drivers support MII, and should provide link-beat detection capability: `3c59x`, `8139too`, `eepro100`, `epic100`, `fealnx`, `hamachi`, `ioc3-eth`, `natsemi`, `pcnet32`, `pegasus`, `sis900`, `starfire`, `sundance`, `tlan`, `tulip`, `via-rhine`, `winbond-840`, and `yellowfin`.)

For network drivers that don't support MII, set the configuration variable `MII_SUPPORTED` to '`"no"`' and link-beat detection will be disabled.

Index

Table of Contents