Table of Contents

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Chapter 4. Neo

4.1. Overview of Neo

In Chapter 2, you saw how a networked device can be managed with SNMP. With `snmpget`, `snmpset`, and `snmpwalk`, you can retrieve operational information and make configuration changes. As useful as these programs are, they are not well suited for debugging day-to-day network problems. A tool is needed that handles all of the SNMP requests behind the scenes while presenting a simple interface to the user.

At MIT, we wrote just such a tool called Neo. Using Neo's simple commands, we can check bandwidth usage or determine on which switch port a particular host resides. For us, it has become the bread and butter of network administration. Neo is a text-based client that is well suited to an environment in which an operator may need to use a remote login session to manage the network.

Of course, Neo is helpful only if your devices can be managed with SNMP. If your devices have only a telnet interface or have no management interface at all, Neo will not be a useful tool for interacting with them.

4.2. What Neo Can Help You Do

Neo can perform a number tasks, and as a relatively new tool, it is constantly having more functionality added. The core functionality is currently:

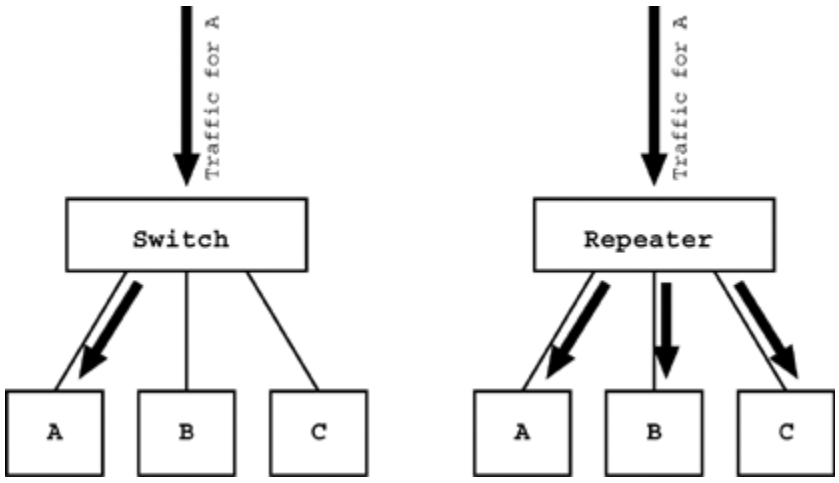
- Determining on which switch port a host resides
- Translating an IP address to a hardware address
- Obtaining traffic statistics
- Obtaining board or port information
- Disabling or enabling a port
- Obtaining general device information
- Obtaining information about device power and environmental conditions

Locating Hosts and the Forwarding Table

One of the benefits of network switches over repeaters is that switches send traffic destined for a particular host only to the port on which that host resides. A repeater, in contrast, sends all traffic to every port (see Figure 4.1). The switch must therefore know which host is on which port so that it can direct traffic appropriately. The switch figures this out using a process known as **station learning**. The idea is for the switch to build a **forwarding table** that maps each host to a port on the switch and to do so without any operator intervention. This allows a host to be plugged into a switch with no need to change the switch configuration manually.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Figure 4.1. Switched Traffic Versus Repeated Traffic.



The process of learning which host is on which port begins with an empty forwarding table. In this state, the switch acts just like a repeater: Because it doesn't know which port traffic should be sent to, it sends the traffic to every port. Eventually, a host on the switch is expected to send some data of its own, either spontaneously or in response to data sent to it. When this happens, the switch notes the hardware address of the sending host and the port the traffic came from. This information is then stored in the forwarding table, and the switch now knows to send traffic destined for that address to that port. In this way, the source address of traffic is used to determine where traffic destined for that address should be sent in the future.

The switch makes information in the forwarding table available via SNMP as described in Chapter 2. Neo allows an operator to query this information easily. You can find the port on which a particular hardware address is located:

```
neo: locate 00:03:BA:09:1F:36 @switch13.example.com
Found on 6@switch13.example.com
```

or ask the switch for all hardware addresses present on a particular port:

```
neo: port search 2/5@switch2.example.com
00:04:76:CB:B1:CD
00:05:02:4B:C6:50
00:06:5B:1D:68:43
00:06:5B:1D:68:46
00:50:8B:AD:FE:8E
00:E0:63:2B:D7:C4
00:E0:63:2B:D7:DC
```

Neo syntax also allows many different devices to be searched in a single query in case the operator does not know on which switch a host will be:

```
neo: locate 00:03:BA:09:1F:36 @k:northannex
Found on 6@switch13.example.com
```

Translating an IP Address to a Hardware Address

Neo can use the SNMP `ipNetToMediaTable` to ask a device for the hardware address associated with a particular IP address:

```
neo: arpfind host.example.com router.example.com
10.5.0.1 says 10.5.1.2 is 00:03:BA:09:1F:36
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

This is particularly useful when you need to locate a host and have only its IP address or host name. You can use Neo first to ask the nearest router for a host's hardware address and then use that hardware address in a `locate` command.

Obtaining Traffic Statistics

It's often very helpful when you are debugging network problems to be able to view the amount of traffic being sent to or from particular ports. Neo is capable of presenting this data:

```
neo: stats switch.example.com
Probing devices ...
Getting first set of stats...
Getting second set of stats...
Port statistics:
p  type  u  lnk adm ap kbs ikbs okbs pps ipps opps ierps oerps
-------------------------------------------------------------
 1 100TX    100  On     20   0   20   26   0   26     0    0
 2 100T     100  On     19   0   19   26   0   26     0    0
 3 100TX     10  On     20   0   20   27   0   27     0    0
 4 100TX      -  On      0   0    0    0   0    0     0    0
 5 100TX      -  On      0   0    0    0   0    0     0    0
 6 100TX    100  On    455  42  413  157  51  106     0    0
 7 100TX     10  On     19   0   19   26   0   26     0    0
 8 100TX    100  On     19   0   19   26   0   26     0    0
 9 100TX    100  On     19   0   19   26   0   26     0    0
10 100TX     -  On      0   0    0    0   0    0     0    0
11 100TX    100  On     19   0   19   26   0   26     0    0
12 100TX    100  On     19   0   19   27   0   27     0    0
13 100?X *  100  On    382 368   14   84  71   13     0    0
14 100?X *   -  On      0   0    0    0   0    0     0    0
15 loop      10  On     59  28   31   80  40   40     0    0
```

Disabling or Enabling a Port

When a host or a network segment is causing an operational problem, the only solution may be to disable that part of the network in order to keep everything else up and running. Neo has commands for turning a port on or off:

```
neo: port disable 22@switch.example.com
22@switch.example.com disabled

neo: port enable 22@switch.example.com
22@switch.example.com enabled
```

Power, Environmental, and General Device Information

Finally, Neo can ask devices about the status of power, environmental conditions, and other information. On any device, Neo can print the system name, the address of the administrative contact, and other information from the SNMP `system` group. On devices that support it, Neo can print the temperature of the device or the status of the power supplies. On a managed UPS, Neo can retrieve detailed information about the battery lifetime and the operational status of the device.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Other Functionality

Aside from these core functions, Neo has support for a few additional features such as accessing information from printers, performing domain name server (DNS) lookups, and executing shell commands.

4.3. Installing Neo

The source for Neo is available at http://web.mit.edu/ktools/. Download the latest version and uncompress the package:

```
Solaris% gunzip -c neo-1.2.12.tar.gz | tar xvf -
```

The unpacked source can then be found in a directory with the same name as the version you have downloaded. Change to that directory:

```
Solaris% cd neo-1.2.12
```

And then configure and build the package:

```
Solaris% ./configure
Solaris% make
```

You may also then type:

```
Solaris% make install
```

which simply places the Neo binary in `/usr/local/bin/`.

Neo should build without trouble on most platforms and is regularly tested on Solaris and Linux in particular. Because Neo is a relatively new piece of software, the configuration and build process has not yet been tested at many sites. If you encounter a problem with either step, you are encouraged to send email to `bug-neo@mit.edu` with a detailed description of the failure.

If the configure script is not able to find the GNU readline library, Neo will successfully build without it, but command line editing will not be available. If your system does not have the readline library, you can download it at http://www.gnu.org/ and install it on your system. Solaris, for example, does not come with the readline library, though Linux typically does.

If you have readline installed in a nonstandard location, you may need to set the `CFLAGS` and `LDFLAGS` environment variables before running the configure script. For example, if your shell is csh or tcsh, you would use:

```
Solaris% setenv CFLAGS -I/usr/local/include
Solaris% setenv LDFLAGS -L/usr/local/lib
Solaris% ./configure
```

But if your shell is bash or the Bourne shell, it would instead be:

```
Solaris% CFLAGS=-I/usr/local/include export CFLAGS
Solaris% LDFLAGS=-L/usr/local/lib export LDFLAGS
Solaris% ./configure
```

Neo builds its own small SNMP implementation and therefore does not require any external SNMP packages.

Once the binary is built, try running it to make sure it works. You should see output like this:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
Solaris% ./neo

Welcome to neo version 1.2.12 (Atropine).
Consult 'help version' for information on features in this version.

neo:
```

If so, you have successfully built Neo and are ready to begin using it.

4.4. Using Neo

The following sections discuss Neo commands and syntax in detail.

4.4.1. The Command Prompt

Neo presents you with its own command prompt, which is the word “neo” followed by a colon. From this prompt, you can type commands for Neo to execute. If upon startup Neo did not print a warning about the readline library, you will have full command line editing available from the prompt. Figure 4.2 lists some of the common command line editing features; for a more detailed reference, consult http://www.gnu.org/directory/readline.html.

Figure 4.2. Basic Readline Editing Keys.

Key	Function
C-f / Right	Move cursor right
C-b / Left	Move cursor left
C-a	Move cursor to start of line
C-e	Move cursor to end of line
M-f	Move cursor to next word
M-b	Move cursor to previous word
C-p / Up	Retrieve the previous command
C-n / Down	Retrieve the next command

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Key	Function
C-k	Kill text to end of line
C-y	Yank killed text into buffer

At any time while Neo is executing a command, you can type Control-C to interrupt the running command and return to the Neo prompt.

The Neo command prompt also allows a limited amount of command abbreviation. For example, the `device` command can be abbreviated `dev`, and the `port enable` command can be abbreviated `port ena`.

4.4.2. The Location Syntax

One of Neo's strengths is its flexible location syntax. The location syntax is the format Neo uses to describe a device, a number of devices, or particular ports on a device. Generically, the location syntax takes the form *port@device*. That is, a port designator and a device designator separated by an at sign. In a simple example, this may be `5@switch.example.com`, representing port five on switch.example.com. However, either the port designator or the device designator may use more exotic options.

The Device Designator

In the simplest case, a location may be just a single device with no associated ports. For this, use an at sign followed by the host name:

```
@switch.example.com
```

Or in just this example alone, you can abbreviate it to the host name with no at sign:

```
switch.example.com
```

Sometimes you will want a location to represent more than one device, which you can accomplish several different ways. First, you can list device names separated by commas:

```
@switch1.example.com,switch2.example.com,router.example.com
```

Make sure there are no spaces in the location or Neo will interpret it as multiple arguments to a command instead of as a single location.

Often you will want to refer to so many devices that separating them all by commas would be cumbersome. One option is to reference a file that contains a list of host names and use the syntax @f:*filename*. For example, if you have a file `/var/tmp/switches` with a list of hosts, one per line, use the location syntax:

```
@f:/var/tmp/switches
```

You can check yourself by asking Neo to describe which devices are present in a location using the `location print` command:

```
neo: location print @f:/var/tmp/switches
Devices (4) are:
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
SWITCH1.EXAMPLE.COM
SWITCH2.EXAMPLE.COM
SWITCH3.EXAMPLE.COM
SWITCH4.EXAMPLE.COM
```

On a large network, there may be so many devices that it's useful to divide them into smaller groups of devices. For example, you may wish to be able to refer to all the devices on one particular network or broadcast domain. Neo supports this through the use of a **keyfile**. The keyfile is a text file that looks something like this:

```
10.115|M5-201T-SWITCH-ENTRY.EXAMPLE.COM
10.115|M5-201T-SWITCH-1.EXAMPLE.COM
10.115|M5-201T-SWITCH-2.EXAMPLE.COM
10.115|M5-201T-SWITCH-3.EXAMPLE.COM
10.115|M5-201T-SWITCH-4.EXAMPLE.COM
10.115|M5-301T-SWITCH-1.EXAMPLE.COM
10.116|M6-332T-SWITCH-ENTRY.EXAMPLE.COM
10.116|M6-332T-SWITCH-1.EXAMPLE.COM
10.116|M6-332T-SWITCH-2.EXAMPLE.COM
10.116|M6-332T-SWITCH-3.EXAMPLE.COM
10.116|M6-432T-SWITCH-1.EXAMPLE.COM
10.116|M6-432T-SWITCH-2.EXAMPLE.COM
```

Each line contains a key string, followed a vertical bar, and then a value string. In Neo location syntax, we can refer to all of the devices whose key is 10.116 with the syntax:

```
@k:10.116
```

The `k` stands for key, of course. If you ask Neo for details on that location, you will find:

```
neo: location print @k:10.116
Devices (6) are:
 M6-332T-SWITCH-ENTRY.EXAMPLE.COM
 M6-332T-SWITCH-1.EXAMPLE.COM
 M6-332T-SWITCH-2.EXAMPLE.COM
 M6-332T-SWITCH-3.EXAMPLE.COM
 M6-432T-SWITCH-1.EXAMPLE.COM
 M6-432T-SWITCH-2.EXAMPLE.COM
```

At MIT we find it useful to organize the keyfile by network number because of the addressing scheme we use. But the key can really be any text string. You may choose to create a keyfile more like:

```
eastwing|SWITCH1.EXAMPLE.COM
eastwing|SWITCH2.EXAMPLE.COM
eastwing|SWITCH3.EXAMPLE.COM
northannex|SWITCH4.EXAMPLE.COM
northannex|SWITCH5.EXAMPLE.COM
northannex|SWITCH6.EXAMPLE.COM
core|SWITCH7.EXAMPLE.COM
core|SWITCH8.EXAMPLE.COM
core|SWITCH9.EXAMPLE.COM
```

and then refer to groups of devices as `@k:core` or `@k:northannex`. There is also no rule that says the same device can't belong to multiple key groups if that suits your needs.

By default, Neo expects the keyfile to be named `/usr/local/etc/neo.keyfile`. You can modify this by changing one of Neo's built-in variables, described in Section 4.4.3, or by using the `-k` command line option, described in Section 4.4.11.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

The Port Syntax

A port designator, if present in a location, always precedes the at sign. If the location is to refer to an entire device, the port is left off entirely. If we do wish to specify a port, the format will depend on the design of the device.

Many devices have a simple notion of numbered ports; the first port on the device is port one, the next one is port two, and so on. The physical ports on the device may be labeled with this number. From a management point of view, there will typically be one or two more ports available via SNMP than are physically present on the device. These are loopback ports or ports otherwise intended for internal use. For a straightforward device like this, the port syntax is simply the number of the port. So port twelve on switch1.example.com would be:

```
12@switch1.example.com
```

Other devices, particularly devices with removable modules, have a notion of a port being numbered by a board number and port number on that board. For example, a Cisco Catalyst switch may have six slots, each one capable of holding a board with 24 ports. Instead of numbering each port on the device with a unique integer, we'd like to refer to a port as the board number the port is on and then the port number on that board. The notation for this is *board*/*port*. So port five on board three on switch5.example.com would be:

```
3/5@switch5.example.com
```

Note that this means the appropriate syntax depends on the type of device Neo is talking to. Most of the time, you will know the layout of the device before you wish to specify a particular port, but if you do use the wrong port syntax for a device, Neo will print a warning about the mistake. In Section 4.4.7, there is information on using the Neo `device` command to ask a device for its layout so that you will definitively know the correct syntax.

Also be aware that some devices that do not physically have boards do have an internal notion of boards. The Cisco Catalyst 2948 has 50 static, numbered ports with no modular components, but internally, the Cisco software considers these ports to be on board two, while a phantom board one is the management board.

Occasionally, you may wish to refer to a board itself with no particular port. You might want to ask a device about the status of board three, for example. The syntax for this is *board*/@ *device*, as in:

```
3/@switch5.example.com
```

Just as you may wish to specify many devices in a single location, it is also useful to be able to specify many ports in a single location. You can use an asterisk in the port designator to represent multiple ports. By example:

```
*@switch1.example.com
```

refers to *all* ports on switch1.example.com, and this switch must be a switch without boards. If you wish to refer to all ports on a device that does have boards, use:

```
*/*@switch5.example.com
```

Similarly, you could refer to all ports on a particular board with:

```
4/*@switch5.example.com
```

Neo also allows you to use a period in place of an asterisk, for convenience in using wildcards on a shell command line. The following two examples are then identical:

```
*@switch1.example.com
```

```
.@switch1.example.com
```

Note that wildcards like these can be used on only the left side of the at sign.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Figure 4.3 summarizes the different forms of location syntax.

Figure 4.3. Summary of Neo Location Syntax.

Location	Meaning
`@switch1`	The device switch1
`switch1`	The device switch1
`@switch1,switch5`	The devices switch1 and switch5
`@f:/var/tmp/devicelist`	The devices listed in /var/tmp/devicelist
`@k:east`	Devices associated with “east” in the keyfile
`3/2@switch5`	Board 3, port 2 on switch5
`10@switch1`	Port 10 on switch1
`3/@switch5`	Board 3 on switch5
`*@switch1`	All ports on switch1
`.@switch1`	All ports on switch1
`3/*@switch5`	All ports on board 3 on switch5
`*/*@switch5`	All ports on switch5
`*/@switch5`	All boards on switch5

4.4.3. Variables

Neo has a number of built-in variables it uses for controlling user configuration options. Figure 4.4 lists the variables currently in use. You can list the value of all variables by typing the `print` command with no arguments:

```
neo: print
writecom = 'public'
readcom = 'public'
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
keyfile = '/usr/local/etc/neo.keyfile'
statsdelay = 5
timeout = 8
macmode = 'standard'
version = '1.2.12 (Atropine)'
burst = 1
```

Figure 4.4. All Neo Variables.

Variable	Function
readcom	SNMP read community
writecom	SNMP write community
keyfile	Filename of the keyfile
statsdelay	Delay for statics gathering
timeout	SNMP timeout
macmode	Format for MAC addresses
burst	The SNMP burst rate

You can change the value of a variable with the `set` command, and you can view the value with the `print` command:

```
neo: set writecom mysecret
neo: print writecom
writecom = 'mysecret'
```

The above example sets the SNMP community used for SNMP writes to `mysecret`. When setting the `writecom` and `readcom` variables only, you may leave off the final argument and Neo will prompt you to enter the community name without echoing the keystrokes you type to the terminal. This allows an operator to enter a secret string without others being able to see it.

The `timeout` variable controls how long Neo should wait for an SNMP response from a device. The value is in seconds and the default is eight seconds. If for some reason this is not a long enough timeout, you can change the value.

The `burst` variable can be used to force SNMP to send multiple copies of packets in the event that network conditions are degraded and packets are being dropped. This is discussed later in the chapter, as are the `statsdelay` and `macmode` variables.

4.4.4. The Arpfind Command

The `arpfind` command queries a device's ARP cache in order to translate an IP address to a hardware address. The syntax is simply: `arpfind` *hostname devicelocation*. For example:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
neo: arpfind myhost.example.com router.example.com
router.example.com says 10.5.1.2 is 00:03:BA:09:1F:36
```

Or if you wish to search multiple devices, you can use a more interesting location syntax:

```
neo: arpfind otherhost.example.com @k:routers
router-east.example.com says 10.6.0.12 is 08:00:20:9A:D3:5F
```

4.4.5. The Locate Command

As described earlier, the `locate` command searches a switch's forwarding table for a particular hardware address. The syntax for the locate command is `locate [-v] [-u]` *address devicelocation*. Typical use would be:

```
neo: locate 00:03:BA:09:1F:36 @k:northannex
Found on 6@switch13.example.com
```

There is an important subtlety to notice here. If we run the locate command again with the `-v` option, it tells us about all the devices being searched:

```
neo: locate -v 00:03:BA:09:1F:36 @k:northannex
Probing devices ...
Searching switch11 ...
Searching switch12 ...
Searching switch13 ...
Found on 6@switch13 ...
Searching switch14 ...
Searching switch15 ...
Searching switch16 ...
1 locations found
```

In this particular example, all the switches are on the same physical network. But then why doesn't this MAC address appear on the other switches as well? The answer is that Neo employs some trickery to try to ignore ports that are the uplink to the device. If you use the `-u` option to the `locate` command, Neo will print every location found, without filtering uplink ports:

```
neo: locate -u 00:03:BA:09:1F:36 @k:northannex
Found on 25@switch10
Found on 25@switch11
Found on 25@switch12
Found on 6@switch13
Found on 25@switch14
Found on 25@switch15
Found on 25@switch16
```

Here we see that port 25 is the uplink to these switches and that Neo filtered those answers for us.

4.4.6. The Port Command

The port command has four different subcommands used to perform functions on a port:

- `port enable`: Turn the port on
- `port disable`: Turn the port off
- `port status`: Report the administrative status of the port

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

- `port search`: Print the address of all devices on the port

These are all self-explanatory and examples follow. Do note that the `port enable` and `port disable` commands use an SNMP write. Assuming you do not have the write community for your devices set to "public," you will need to set the Neo SNMP write community first with the `set writecom` command.

```
neo: port disable 22@switch.example.com
22@switch.example.com disabled

neo: port enable 22@switch.example.com
22@switch.example.com enabled

neo: port status 22@switch.example.com
22@switch.example.com enabled

neo: port search 2/5@switch5.example.com
00:04:76:CB:B1:CD
00:05:02:4B:C6:50
00:06:5B:1D:68:43
00:06:5B:1D:68:46
00:50:8B:AD:FE:8E
00:E0:63:2B:D7:C4
00:E0:63:2B:D7:DC
```

When you use the `port search` command, the `macmode` variable controls the format of the hardware addresses returned. Figure 4.5 provides examples of the different settings for the `macmode` variable and the hardware address format that each setting corresponds to. You may wish to change the format to "cisco" if you will be feeding the results to a Cisco command prompt, thereby saving you the effort of converting each address format yourself.

Figure 4.5. Settings for the `macmode` Variable.

macmode	Example
standard	`00:04:76:CB:B1:CD`
cisco	`0004.76CB.B1CD`
dash	`00-04-76-CB-B1-CD`

4.4.7. The Device Summary Command

The `device` command has three subcommands:

- `device summary`
- `device info`
- `device type`

The first command, `device summary`, provides a useful summary of the layout of a device. For a simple device without boards, this prints a summary of all ports:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
neo: dev sum switch1.example.com
Port summary:
 p type     u  lnk   adm ap
------------------------------
 1 100TX       100   On
 2 100TX       100   On
 3 100TX        10   On
 4 100TX         -   On
 5 100TX         -   On
 6 100TX       100   On
 7 100TX        10   On
 8 100TX       100   On
 9 100TX       100   On
10 100TX         -   On
11 100TX       100   On
12 100TX       100   On
13 100TX       100   On
14 100TX       100   On
15 100TX       100   On
16 100TX       100   On
17 100TX       100   On
18 100TX         -   On
19 100TX       100   On
20 100TX         -   On
21 100TX         -   On
22 100TX       100   On
23 100TX         -   On
24 100TX         -   On
25 100?X     * 100   On
26 100?X     *   -   On
27 loop         10   On
```

The first column indicates the port number. The "type" column denotes the media type of the port. The "u" column contains an asterisk if Neo believes this port is an uplink. The "lnk" column is the link speed of the port, or a hyphen if no link is present. Finally, the "adm" column tells us whether the port is administratively on or off. The column marked "ap" is not often used, but it can indicate that a port has been automatically disabled by the device for some particular reason.

On a device with boards, the `device summary` command is particularly useful because it shows us the board layout of the device:

```
neo: dev sum switch5.example.com
Board summary:
b type              sg  ul ultype
-----------------------------------------
1 Mgmt2 2 1000X      0
2 24 10/100/1000T    0
3 24 10/100/1000T    0
4 Empty
5 Empty
6 24 100FX (mm)      0
```

The first column is the board number and the second is a description of the board. On this device, board three has 24 ports, each 10/100/1000T. The remaining columns are used occasionally for certain kinds of repeaters and indicate what segment the board is on and whether it has a modular uplink.

The most important aspect of the device summary command is that it can be used on a device without your knowing what the layout of the device is ahead of time. If you're starting cold with just a host name for a device, the device summary command is a good way to get your bearings before proceeding with commands that expect you to know its layout.

Once you do know that a device has boards, you can also use the `device summary` command to generate a port summary, just as for a device without boards. Simply use the location syntax for accessing all ports on a board:

```
neo: dev sum 3/*@switch5.example.com
Port summary:
 p type     u lnk adm ap
------------------------------
 1 1000T       -  On
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
 2 1000T        -  On
 3 1000T        -  On
 4 1000T        -  On
 5 1000T        -  On
 6 1000T        -  On
 7 1000T        -  On
 8 1000T        -  On
 9 1000T        -  On
10 1000T        -  On
11 1000T        -  On
12 1000T        -  On
13 1000T        -  On
14 1000T        -  On
15 1000T        -  On
16 1000T        -  On
17 1000T        -  On
18 1000T     1000  On
19 1000T     1000  On
20 1000T     1000  On
21 1000T     1000  On
22 1000T      100  On
23 1000T     1000  On
24 1000T     1000  On
```

4.4.8. The Device Info Command

In its simplest form, the `device info` command prints the system information from a device:

```
neo: dev info switch.example.com
switch.example.com
 Device type: C2200
 Contact    : admin@example.com
 Name       : switch.example.com
 Location   : 5-142T
 Uptime     : 92 days 22:07:20
 ObjectID   : .1.3.6.1.4.1.52.3.9.3.4.84
 Descr      : Cabletron Systems, Inc. 2H253-25R Rev 04.00....
```

There are also subcommands that provide additional information. When possible, the `device weather` command will print information about the environmental conditions of the device:

```
neo: dev info weather router.example.com
Intake temp: 25C / 77F
Hotpoint   : 35C / 95F
Exhaust    : 32C / 89F
```

The power subcommand will print information about the power status of the device. This may be the status of the power supplies on a piece of network hardware or detailed information about the condition of a UPS:

```
neo: dev info power ups.example.com
 Type             : Symmetra
 Name             :
 # Alarms Present : <unsupported>
 Status           : Normal

 Battery Capacity : 100%
 Battery Run Time : 0 d 02:44:00
 Time On Battery  : 0 d 00:00:00
 Battery Voltage  : <unsupported>
 Battery Current  : <unsupported>

 Input 1 Voltage  : 199 V
 Input 1 Current  : <unsupported>
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
  Input 1 Freq.    : 60 Hz

  Output 1 Voltage : 213 V
  Output 1 Current : 13 A
  Output 1 Freq.   : 60 Hz
  Output 1 Load    : 20%

   (device specific section)

  Battery Status  : All batteries OK
  Last Fail Cause : Self Test
```

If power or environmental information is not available for the device, Neo will print a warning message:

```
neo: dev info power switch3.example.com
Power info is not yet supported on this device
```

4.4.9. The Stats Command

The stats command is another extremely useful Neo command. It prints per-port traffic statistics:

```
neo: stats *@switch1.example.com
Probing devices ...
Getting first set of stats...
Getting second set of stats...
Port statistics:
p type   u lnk adm ap kbs ikbs okbs pps ipps opps ierps oerps
-------------------------------------------------------------
1 100TX    100 On     20    0   20   26    0   26     0    0
2 100TX    100 On     19    0   19   26    0   26     0    0
3 100TX     10 On     20    0   20   27    0   27     0    0
4 100TX      - On      0    0    0    0    0    0     0    0
5 100TX      - On      0    0    0    0    0    0     0    0
6 100TX    100 On    455   42  413  157   51  106     0    0
7 100TX     10 On     19    0   19   26    0   26     0    0
8 100TX    100 On     19    0   19   26    0   26     0    0
9 100TX    100 On     19    0   19   26    0   26     0    0
10 100TX     - On      0    0    0    0    0    0     0    0
11 100TX   100 On     19    0   19   26    0   26     0    0
12 100TX   100 On     19    0   19   27    0   27     0    0
13 100?X * 100 On    382  368   14   84   71   13     0    0
14 100?X *   - On      0    0    0    0    0    0     0    0
15 loop     10 On     59   28   31   80   40   40     0    0
```

Note that this is a device without boards. If we wanted port statistics on a device with boards, we would use:

```
neo: stats 2/*@switch5.example.com
Probing devices ...
Getting device summary ...
Getting first set of stats...
Getting second set of stats...
Port statistics:
 p type  u  lnk adm ap  kbs ikbs okbs pps ipps opps ierps oerps
-----------------------------------------------------------------
  1 1000T   1000  On      11    0   11  13    0   13     0    0
  2 1000T   1000  On     235   78  157 191   90  101     0    0
  3 1000T   1000  On      13    1   12  14    2   12     0    0
  4 1000T   1000  On      36   12   24  43   15   28     0    0
  5 1000T    100  On     287  253   34  54   27   27     0    0
  6 1000T   1000  On      12    0   12  11    0   11     0    0
  7 1000T   1000  On      11    0   11  10    0   10     0    0
  8 1000T   1000  On      11    0   11  10    0   10     0    0
  9 1000T   1000  On      11    0   11  10    0   10     0    0
 10 1000T   1000  On    2200 2126   74 288  176  112     0    0
```


Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
 11 1000T   1000  On    2276   68 2208 306  106  200     0    0
 12 1000T   1000  On      24    5   19  21    4   17     0    0
 13 1000T   1000  On      14    0   14  14    0   14     0    0
 14 1000T   1000  On      13    0   13  15    1   14     0    0
 15 1000T   1000  On      13    1   12  13    2   11     0    0
 16 1000T      -  On       0    0    0   0    0    0     0    0
 17 1000T      -  On       0    0    0   0    0    0     0    0
 18 1000T      -  On       0    0    0   0    0    0     0    0
 19 1000T      -  On       0    0    0   0    0    0     0    0
 20 1000T      -  On       0    0    0   0    0    0     0    0
 21 1000T   1000  On      13    1   12  12    1   11     0    0
 22 1000T   1000  On      12    1   11  10    0   10     0    0
 23 1000T   1000  On      13    0   13  11    0   11     0    0
 24 1000T   1000  On      12    0   12  10    0   10     0    0
```

which would give us port statistics for all ports on board two. Some devices support board statistics; that is, statistics on how much traffic is being handled by a particular board. If a device is capable of this, you can retrieve the information with `stats */@`*device* or just `stats` *device*.

In the above examples, the first few columns are labeled the same as for the `device summary` command. The latter columns denote, in order:

- **kbs:** Total kilobits per second of traffic through the port
- **ikbs:** Kilobits per second of traffic into the port
- **okbs:** Kilobits per second of traffic out of the port
- **pps:** Packets per second through the port
- **ipps:** Packets per second into the port
- **opps:** Packets per second out of the port
- **ierps:** Error packets per second into the port
- **oerps:** Error packets per second out of the port

In the last example, we can see 2.2Mb/s of traffic on board two, ports 10 and 11. Looking more closely, we can see that 2.1Mb/s of traffic is coming from port 10 (into the port) and 2.2Mb/s of traffic is being sent to port 11 (out of the port). Our intuition tells us this is probably data being sent from a machine on port 10 to a machine on port 11, though it is entirely possible that the two machines are independently speaking to other machines on a different board.

When running the `stats` command, you will usually notice a short delay between the text “Getting first set of stats...” and “Getting second set of stats...”. This delay is present so that Neo can gather statistics over a reasonably average period of time. The default is five seconds, but you can change this by setting the `statsdelay` variable to another value. Note that Neo will allow there to be *more* time between data runs than the value specified in `statsdelay`, but it will always allow at least that much time. If for some reason Neo requires a long time to gather the first set of data, the second set may not begin until after the `statsdelay` time has elapsed.

4.4.10. Online Help

Neo has a complete online help system, which you can access by typing the `help` command with no arguments. Neo will print a list of help topics and the name of each command. If you give the topic or command name as an argument to the help command, Neo will print detailed information on the subject. For example, `help locate` will print information on using the locate command, and `help syntax` will print help on Neo’s location syntax.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

4.4.11. Command Line Arguments

You can use Neo in scripts by providing the commands you wish to execute as arguments to the program. For example:

```
Solaris% neo arpfind host.example.com router.example.com
10.5.0.1 says 10.5.1.2 is 00:03:BA:09:1F:36
```

For convenience, Neo has four command line options. From `help args`:

```
-k <keyfile>
      Set the kefyile to <keyfile>.

-w <community>
      Set the write community to <community>.

-r <community>
      Set the read community to <community>.

-c <community>
      Set both the read and write communities to <community>
```

This allows you to set private community names without requiring an extra command:

```
Solaris% neo -w mysecret port dis 10@switch1.example.com
10@switch1.example.com disabled
```

Be aware that this is a security risk. If you run Neo on a system that others can log into, and you place a secret community string on the command line, other users will be able to read it by looking at a list of running processes.

4.4.12. Other Commands

Neo has a few other commands that you may find useful. The `hostinfo` command performs DNS lookups, using built-in code:

```
neo: hostinfo host.example.com
Official name: HOST.EXAMPLE.COM
Host address:  10.5.1.2
Host CPU:    SUN/ULTRA-10
Host OS:    SOLARIS
```

The `location print` command, as illustrated earlier, can be used to print information about the devices present in a Neo location. The `exec` command can be used to exec a shell command from the Neo prompt.

4.4.13. Using Neo in Degraded Network Conditions

One of the times we use Neo most is when a network is having an operational problem. Unfortunately, when a network is in trouble, it may be hard to get SNMP packets to the necessary devices on the network. Imagine a host on your network that has been broken into by a malicious cracker. The cracker runs a denial of service attack program that sends as much network traffic as possible to some distant host on another network. The traffic flooding the connection to your devices may make it difficult for your SNMP management packets to get through.

In an effort to remedy this situation Neo has a variable called `burst`, whose default value is one. In this state Neo will send out one SNMP packet when it needs to make a request. If no response is received and the SNMP timeout is close to expiring, Neo will try sending a second

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

packet. If the burst value is set to two, however, Neo will send out two packets each time instead of one. If the burst value is 10 Neo will send 10 identical SNMP requests each time. The hope is that one or more of the packets will make it through the noise and reach the device and that one of the responses will make it back to your management station.

Use some care, even in very adverse network conditions, that you do not set the burst value too high. There is a tradeoff between ensuring reliability and adding to the congestion by sending multiple packets. Additionally, with a high burst rate, Neo has to work harder to process all the extra information sent and received. Generally, it's a good idea to try a burst setting between two and four before using anything higher. Larger values may be used, but they work better for queries that require only a few packets, such as disabling a port. Trying to gather statistics with a high burst rate can be difficult, for example. In general, there is probably never a reason to use a burst rate higher than 10 or 20, and those rates would be used only in the most extreme circumstances.

4.5. Examples of Use

How does it all come together in the day-to-day operation of a network? Here are a few examples that demonstrate common scenarios.

Finding and Disabling a Host

Say you have the name of a host, `broken.example.com`, that you know has been compromised by a malicious cracker. You want to find the host and disable its network access.[1] First you ping the host in case it has been inactive and is no longer in the forwarding table of the relevant switches. When the device responds to your ping, any switches in the switching path that do not have the host in their forwarding table will add it:

[1] This is one reason it is highly recommended you have only one machine attached to each managed port on the leaf switches in your network. If several machines were connected behind an unmanaged switch, you might be forced to disable network access to all of them.

```
Solaris% ping broken.example.com
broken.example.com is alive
```

You use the arpfind command to determine the hardware address of the host:

```
neo: arpfind host.example.com router.example.com
10.5.0.1 says 10.5.1.2 is 00:03:BA:09:1F:36
```

Having that, you then locate the host on a collection of switches:

```
neo: locate 00:03:BA:09:1F:36 @k:northannex
Found on 6@switch13.example.com
```

Next, you may wish to see how much traffic is being generated by the host. Though this is optional, it may be of use:

```
neo: stats 6@switch13.example.com
Probing devices ...
Getting first set of stats...
Getting second set of stats...
Port statistics:
p type  u lnk adm ap kbs ikbs okbs pps ipps opps ierps oerps
-------------------------------------------------------------
6 100TX   100  On    486   92  394 111   37   74     0     0
```

Before you turn it off, you also may wish to check how many devices are located on that port:

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

```
neo: port search 6@switch13.example.com
00:03:BA:09:1F:36
```

In this case, it is just the one machine, so you disable it:

```
neo: set writecom mysecret
neo: port dis 6@switch13.example.com
6@switch13.example.com disabled
```

Locating a Problem and Disabling It

Now imagine a portion of your network is undergoing an active denial of service attack, sourced from a host within your own network. How can we find the problematic host? Start with the switch connected to the router:

```
neo: stats entry-switch.example.com
Probing devices ...
Getting first set of stats...
Getting second set of stats...
Port statistics:
 p  type  u lnk adm ap kbs  ikbs  okbs   pps ipps opps ierps oerps
-------------------------------------------------------------------
 1 100TX    100  On     20     0    20   26    0    26     0    0
 2 100TX    100  On     19     0    19   26    0    26     0    0
 3 100TX     10  On     20     0    20   27    0    27     0    0
 4 100TX      -  On      0     0     0    0    0     0     0    0
 5 100TX      -  On      0     0     0    0    0     0     0    0
 6 100TX    100  On  50455 50042   413 7157 7051   106     0    0
 7 100TX     10  On     19     0    19   26    0    26     0    0
 8 100TX    100  On     19     0    19   26    0    26     0    0
 9 100TX    100  On     19     0    19   26    0    26     0    0
10 100TX     -  On      0     0     0    0    0     0     0    0
11 100TX    100  On     19     0    19   26    0    26     0    0
12 100TX    100  On     19     0    19   27    0    27     0    0
13 100?X  * 100  On    382   368    14   84   71    13     0    0
14 100?X  *   -  On      0     0     0    0    0     0     0    0
15 loop      10  On     59    28    31   80   40    40     0    0
```

You can see that a suspiciously large amount of traffic is coming from port six, so you would then like to know what other network devices are connected to this port. There are several ways to do this. One method is to search the port for hardware addresses, then pick one of those hardware addresses and use the keyfile syntax to search for that address on the relevant network:

```
neo: port search 6@entry-switch.example.com
00:04:76:31:E5:78
00:06:5B:48:35:09
00:60:97:4D:FE:39
00:E0:29:05:85:66
00:E0:29:86:3D:0D
00:E0:63:C7:23:CB
00:E0:63:C7:23:E3
08:00:69:0E:AF:DD

neo: locate 00:04:76:31:E5:78 @k:northannex
Found on 6@entry-switch.example.com
Found on 5@switch4.example.com
```

You then know that switch4.example.com is a device behind port six. Another method would be to login to the router and ask it to translate each MAC address to an IP address, one of which will be that of the device in question.

Once you gather statistics on switch4.example.com, you again find one source of a significant amount of traffic and turn off the culprit just as in the previous example.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Using Neo in a Script

Imagine you have a UPS and you would like to regularly track the battery capacity and input load. You might use Neo in a script like this one:

```
#!/bin/sh
ups=ups.example.com
log=/home/admin/upslog
while [ 1 ]; do
   date >> $log
   neo dev info power $ups | egrep "Capacity|Load" >> $log
   echo "" >> $log
   sleep 1800
done
```

This text is stored in a file and then the program is executed. More detail on writing and using scripts is presented in Chapter 10. This script runs Neo once every 30 minutes, saving the relevant information from the UPS into a logfile. Note that Neo commands are simply presented to Neo as arguments on the command line.

4.6. Maintaining Neo

Neo requires two maintenance tasks. The first is regularly updating the installed version of Neo simply because, as with any new piece of software, features and bug fixes are constantly being added. The latest version is always available on the Web site.

The second maintenance task is updating the keyfile, if you use one. Neo is an effective tool only if it knows which devices to contact. Make sure your keyfile reflects the current state of your network either by updating it manually or by using an automated process.

Neo does not require you to maintain SNMP MIBs; any necessary variable data is coded into the program itself. The upshot is that there is less complexity for the maintainer, though it also means that the software must be updated in order to include new functionality.

4.7. References and Further Study

Chapter 2 describes the SNMP protocol that Neo uses to perform its tasks and includes details on some of the relevant SNMP variables as well. The GNU readline library and other GNU tools are available from http://www.gnu.org/. Neo is available from http://web.mit.edu/ktools/, and bugs, problems, or suggestions for improvements should be sent to bug-neo@mit.edu.

There are many other SNMP-based management programs available. HP's OpenView (http://www.openview.hp.com/) is a widely used commercial example, and Scotty/Tkined (http://www.home.cs.utwente.nl/~schoenw/scotty/) is an open source example. This kind of program most often has a graphical user interface, and these two programs are no exception. This is a major disadvantage for accomplishing many administration tasks. One tool that does not have a graphical user interface is ND, written at Texas A&M University. A paper describing this system is available at http://www.usenix.org/events/lisa00/mitchell.html.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003