CHAPTER

Simple Network Management Protocol

24

What You Will Learn

In this chapter, you will learn how SNMP is used to manage devices on a TCP/IP network. We'll explore the SNMP model with many servers (agents) and few clients (managers).

You will learn about MIBs and the SMI tree for designating management information. We also briefly discuss RMON (remote monitor) and private management information bases (MIBs).

Network management, like network security, is often treated like an adjunct to the true task of networking, which is to relentlessly shuttle bits about (i.e., until something goes wrong). Then everyone wonders why it couldn't be easier to figure out what went haywire. Without network management facilities, the network is like driving a car without fuel-level, water-temperature, or oil-pressure gauges. When the car slowly glides to a halt, there are few clues of even where to start looking.

The Internet outgrew the humble go-have-a-look-at-it school of network management by the late 1980s, when it seemed like colleges and universities were sticking routers in every other building around the campus and then finding someone who would not object to being placed in charge of the devices. Little did they realize that they would be expected to ensure that the out-of-the-way device was functional day and night, 365 days a year. They ran their portion of the Internet on a PING and a prayer.

It's not that management of network devices was unknown at the time, or deemed unnecessary. Vendors always had some sort of management functions tucked away in their software. The problem was that each vendor's interface was different (sometimes in the same product line), the client software expensive and proprietary, and the network operations centers (NOCs) that existed tended to consist of rooms full of equipment that no one knew how to operate equally well.

But knowing that network management was essential and creating a standard for network management on the Internet were two different things. The international

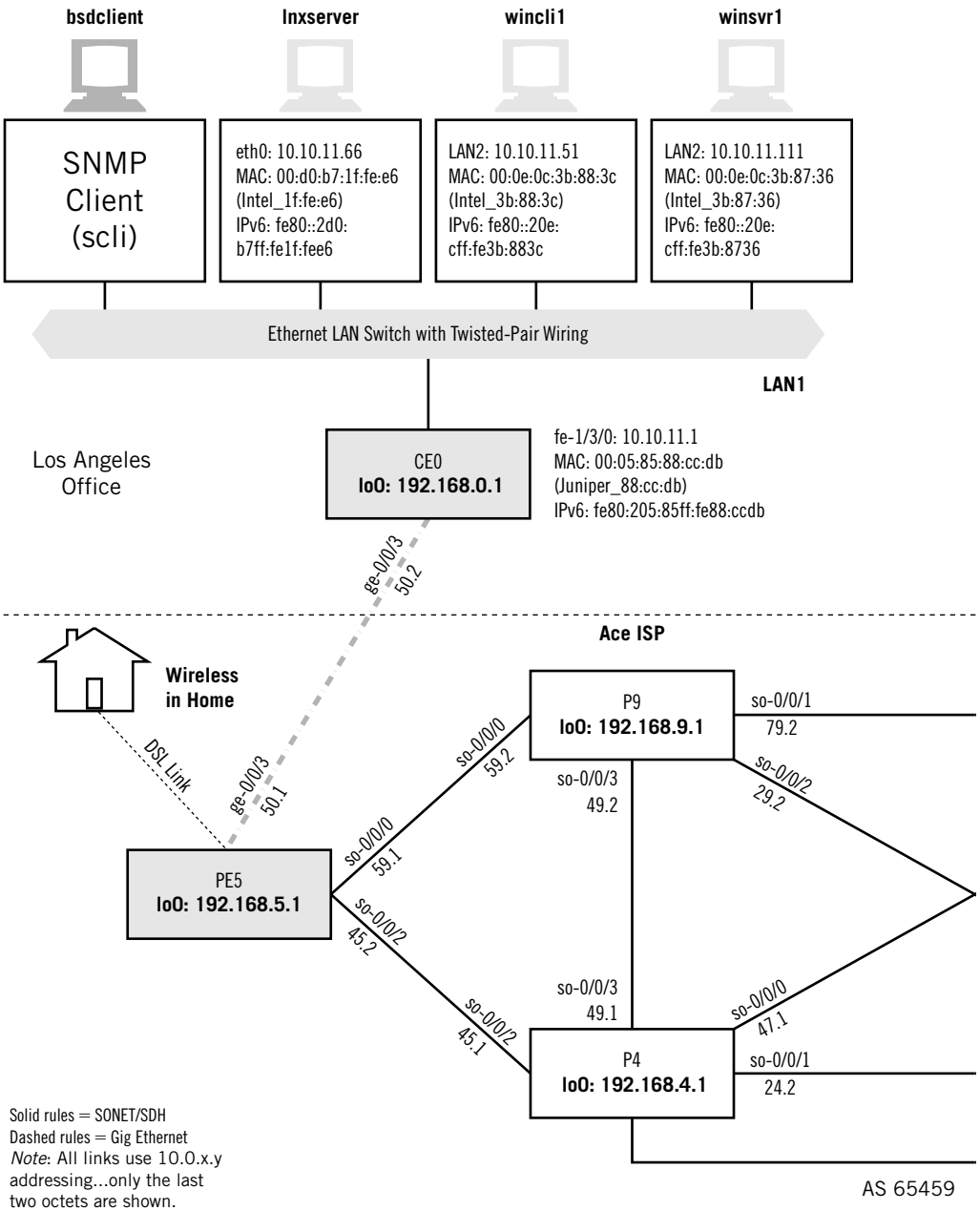
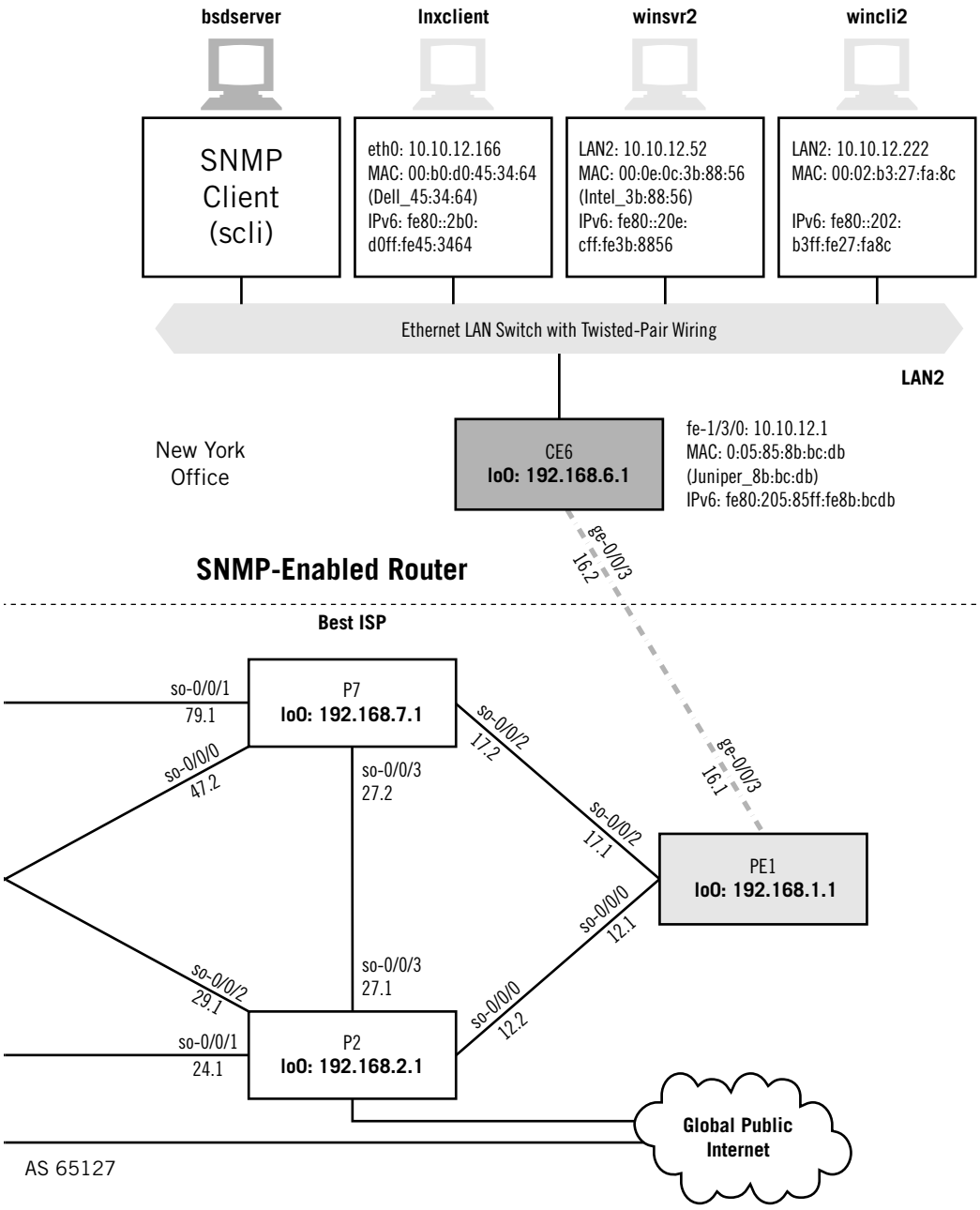


FIGURE 24.1

SNMP on the Illustrated Network, showing the hosts used as SNMP clients and the router with SNMP enabled.


bsdserver
lnxclient
winsvr2
wincli2
SNMP Client (scli)
eth0: 10.10.12.166
MAC: 00:b0:d0:45:34:64
(Dell_45:34:64)
IPv6: fe80::2b0:
d0ff:fe45:3464
LAN2: 10.10.12.52
MAC: 00:0e:0c:3b:88:56
(Intel_3b:88:56)
IPv6: fe80::20e:
cff:fe3b:8856
LAN2: 10.10.12.222
MAC: 00:02:b3:27:fa:8c
IPv6: fe80::202:
b3ff:fe27:fa8c
Ethernet LAN Switch with Twisted-Pair Wiring
LAN2
New York Office
CE6
lo0: 192.168.6.1
fe-1/3/0: 10.10.12.1
MAC: 0:05:85:8b:bc:db
(Juniper_8b:bc:db)
IPv6: fe80:205:85ff:fe8b:bcdb
ge-0/0/3
16.2
SNMP-Enabled Router
Best ISP
so-0/0/1
79.1
P7
lo0: 192.168.7.1
so-0/0/2
17.2
so-0/0/0
47.2
so-0/0/3
27.2
ge-0/0/3
16.1
so-0/0/2
17.1
PE1
lo0: 192.168.1.1
so-0/0/0
12.1
so-0/0/2
29.1
so-0/0/3
27.1
so-0/0/0
12.2
so-0/0/1
24.1
P2
lo0: 192.168.2.1
Global Public Internet
AS 65127

standard for network management, itself a new creation at the time, was the Common Management Information Services/Common Management Information Protocol (CMIS/CMIP). However, this standard (geared to the needs of public telephony carriers) was loaded with features unnecessary to the Internet at the time. So, Internet administrators took what they could from the ISO specifications and created SNMP fairly independently.

SNMP CAPABILITIES

The need for network management information has to be weighed against the need for security. Yet many organizations routinely run SNMPv1 on their network nodes, hubs, or routers, and seldom take advantage of the heightened security available in many SNMPv1 implementations or consider SNMPv2. Organizations routinely block Telnet access to their routers, yet allow SNMP access without too much worry.

Just how much information can be gathered from a router running SNMPv1when no steps have been taken to protect information? Quite a bit, actually.

Let's enable SNMP on one of our routers, CE6, attached to LAN2, and use `bsdclient` on LAN1 and `bsdserver` on LAN2 to see what we can do with SNMP. There are many nifty GUIs available for SNMP, but we'll use FreeBSD's `scli` application to maximize information and minimize clutter on the screen. We won't be interested in traffic histograms or historical data anyway. The equipment used in this chapter is shown in Figure 24.1.

Enabling SNMP on a Juniper router is very straightforward (just setting values to the proper variables) and need not be shown. The following is the result of our initial configuration.

```
admin@CE6# show snmp
name Router_CE6;
description M71-Router;
contact WalterG;
```

There is much more we could have configured, and in fact this is really more than we need. But it will allow us to ensure that it's the right router. Now we can run a Unix command-line management application on `bsdclient` called `scli` to router CE6. (We haven't put the routers in DNS, and many organizations don't for security purposes, so we'll access the router by an interface IP address instead of by name.)

```
bsdclient# scli 10.10.12.1
100-scli version 0.2.12 (c) 2001-2002 Juergen Schoenwaelder
100-scli trying SNMPv2c ... good
(10.10.12.1) scli >
```

We are now running SNMPv2 to the router. Note that `scli` is an interactive application with its own > prompt, like `nslookup`, so we can execute all types of commands

(known through `help`) at this point until an `exit` takes us out to the shell again. Let's ensure that we have the right router and examine the system information.

```
(10.10.12.1) scli > show system info
Name:              Router_CE6
Address:           10.10.12.1:161
Description:       M7i-router
Contact:            WalterG
Location:
Vendor:            unknown (enterprises.2636)
Services:          network
Current Time:      2008-02-28 20:11:36 -07:00
Agent Boot Time:   2008-02-21 20:44:12 -08:00
System Boot Time:  2008-02-21 20:43:27 -08:00
System Boot Args:  /kernel
Users:             3
Processes:         61 (532 maximum)
Memory:            256M
Interfaces:        50
Interface Swap:    2008-02-21 20:45:31 -08:00
(10.10.12.1) scli >
```

That's the router all right. Note that we get a lot more information than we entered. And some people would be very nervous about the system details that SNMP has gathered from this router. But let's look at SNMP in action first. Figure 24.2 shows the SNMP messages and details. One response is of particular interest—the one that has the information we entered on the router. Most of the information displayed at the start of the `show` command can be picked out of the lower pane in the figure.

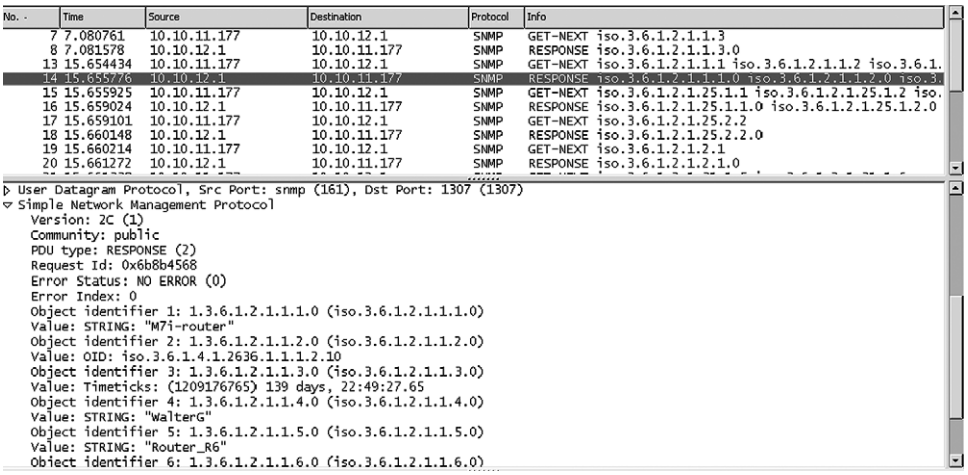


FIGURE 24.2

SNMP session to router CE6.

Let's see what harm we can cause with SNMP by changing something.

```
(10.10.12.1) scli > set system contact NotMe
500 noResponse 1.00 vpm
(10.10.12.1) scli >
```

The `noResponse` tells us that our request was ignored by CE6. Most devices will enable SNMP with read-only access unless told otherwise. Still, there's a lot of information available about good old router CE6, such as the following:

```
(10.10.12.1) scli > show interface
# show interface info [10.10.12.1] [2008-02-28 20:43:38 -07:00]

INTERFACE STATUS  MTU TYPE              SPEED NAME       DESCRIPTION
        1  UUCN  1514 ethernetCsmacd     100m fxp0       fxp0
        2  UUCN  1514 ethernetCsmacd     100m fxp1       fxp1
        4  UUNN  1496 mplsTunnel            0 lsi        lsi
        5  UUNN 2147483647 other                0 dsc        dsc
        6  UUNN 2147483647 softwareLoopback     0 lo0        lo0
        7  UUNN 2147483647 other                0 tap        tap
        8  UUNN 2147483647 tunnel               0 gre        gre
        9  UUNN 2147483647 tunnel               0 ipip       ipip
       10  UUNN 2147483647 tunnel               0 pime       pime
       11  UUNN 2147483647 tunnel               0 pimd       pimd
       12  UUNN 2147483647 tunnel               0 mtun       mtun
       13  UUNN  1500 propVirtual        100m fxp0.0     fxp0.0
       14  UUNN  1514 propVirtual        100m fxp1.0     fxp1.0
       16  UUNN 2147483647 softwareLoopback     0 lo0.0      lo0.0
       21  UUCN  4474 sonet              155m so-0/0/0   so-0/0/0
       22  UUNN  4470 ppp                155m so-0/0/0.0 so-0/0/0.0
       23  UUCN  4474 sonet              155m so-0/0/1   so-0/0/1
       24  UUNN  4470 ppp                155m so-0/0/1.0 so-0/0/1.0
       25  UUCN  4474 sonet              155m so-0/0/2   so-0/0/2
       26  UUNN  4470 ppp                155m so-0/0/2.0 so-0/0/2.0
       27  UUCN  4474 sonet              155m so-0/0/3   so-0/0/3
       28  UUNN  4470 ppp                155m so-0/0/3.0 so-0/0/3.0
       29  UUNN 2147483647 softwareLoopback     0 lo0.16385  lo0.16385
       30  UUNN 2147483647 tunnel        800m pd-1/2/0   pd-1/2/0
       31  UUNN 2147483647 tunnel        800m pe-1/2/0   pe-1/2/0
       32  UUNN 2147483647 tunnel        800m gr-1/2/0   gr-1/2/0
       33  UUNN 2147483647 tunnel        800m ip-1/2/0   ip-1/2/0
       34  UUNN 2147483647 tunnel        800m vt-1/2/0   vt-1/2/0
       35  UUNN 2147483647 tunnel        800m mt-1/2/0   mt-1/2/0
       36  UUNN     0 tunnel             800m lt-1/2/0   lt-1/2/0
       37  UUCN  1514 ethernetCsmacd     100m fe-1/3/0   fe-1/3/0
       38  UDCN  1514 ethernetCsmacd     100m fe-1/3/1   fe-1/3/1
       39  UUNN 2147483647 tunnel        800m pd-0/3/0   pd-0/3/0
       40  UUNN 2147483647 tunnel        800m pe-0/3/0   pe-0/3/0
       41  UUNN 2147483647 tunnel        800m gr-0/3/0   gr-0/3/0
```

```
       42  UUNN 2147483647 tunnel        800m ip-0/3/0   ip-0/3/0
       43  UUNN 2147483647 tunnel        800m vt-0/3/0   vt-0/3/0
       44  UUNN 2147483647 tunnel        800m mt-0/3/0   mt-0/3/0
       45  UUNN      0 tunnel            800m lt-0/3/0   lt-0/3/0
       46  UDCN  1504 e1                   2m e1-0/2/0   e1-0/2/0
       47  UDCN  1504 e1                   2m e1-0/2/1   e1-0/2/1
       48  UDCN  1504 e1                   2m e1-0/2/2   e1-0/2/2
Byte 2969
```

And this is only *part* of it. Just imagine if someone managed to break in and . . . but wait: All we did is use a router interface's IP address. No breaking in was needed.

What can we do to tighten things up? Let's limit SNMP access to a single interface on the router, and a single host reachable through the interface. The interface will be LAN2, on `fe-1/3/0`, not surprisingly. We'll use the LAN2 host `bsdserver` so that we can still use `scli`. We'll also let an administrator with `root` privileges on `bsdserver` make changes with the `set` request in the SNMP *community* (a sort of SNMP "password," but it's really not) called `locallan`. Almost all of this is configured on the router, not the host. The `scli` limitation to execute a remote `set` command is a function of the application. The following presents the new router configuration.

```
set snmp name Router_CE6;
set snmp description M7i-router;
set snmp contact WalterG;
set snmp interface fe-1/3/0.0; # restrict SNMP to the LAN2 interface
set snmp view syscontact oid sysContact include; # let the manager change
   the sysContact
set snmp community locallan view sysContact; # establish new community
   string and add sysContact to view. . .
set snmp community locallan authorization read-write; # . . .and let it be
   read and write access. . .
set snmp community locallan clients 10.10.12.77/32; # . . .but only from
   bsdserver for the locallan community string
```

We have to explicitly add the `sysContact` object ID to a "view" for the community string `locallan` if we are going to allow the network manager on `bsdserver` to change the value of that object. Back on `bsdclient`, the effects of these changes are immediate.

```
(10.10.12.1) scli >  show ip
500 noResponse
500 noResponse
500 noResponse
500 noResponse
500 noResponse
(10.10.12.1) scli >
```

But things are different once we switch to `bsdclient` (and remember to use the community string `locallan`).

```
> bsdserver# scli
100-scli version 0.2.12 (c) 2001-2002 Juergen Schoenwaelder
scli > open 10.10.12.1 localIan
100-scli trying SNMPv2c ... good
(10.10.12.1) scli > set system contact NotMe
(10.10.12.1) scli > show system
# show system info [10.10.12.1] [2008-02-28 21:02:07 -07:00]

Address:          10.10.12.1:161
Contact:          NotMe
(10.10.12.1) scli >
```

If we forget to add the object explicitly to the community on the router, `bsdserver` still has access but will not be able to write to the object.

```
(10.10.12.1) scli > set system contact NotMe
500 noAccess @ varbind 1
(10.10.12.1) scli >
```

By now it should be obvious that SNMP can be a powerful network management tool, independent of remote-access or vendor-specific management techniques. However, all of this talk about objects, community strings, SNMPv1, and v2 can be confusing. SNMP introduces a lot of terms and concepts. Let's start at the beginning and see just what SNMP can do and how it does it.

THE SNMP MODEL

This section takes a more detailed look at how SNMP, versions 1 and 2, works. This chapter identifies the shortcomings of SNMPv1 that led to the creation of SNMPv2, and then shows what SNMPv3 will add to SNMP. SNMP remains the most popular and most viable method of managing networks today, let alone the Internet.

All network management standards, not just SNMP, work by means of what is known as the *agent/manager model*. This is not really a new term or concept. The term "agent/manager model" is essentially the client/server model idea extended to network management. A manager is just a management console in the NOC running the network management software, not an actual human being. An *agent* is software that runs on all manageable devices on the network. As in the client/server model, managers "talk" and the agents "listen." So, managers are clients for network management purposes and agents are servers for network management purposes. Obviously, a major difference in the agent/manager model from traditional client/server is that in a network management situation, there are many servers (agents) and generally only a few clients (management consoles).

The manager running in the network management station (or any host setup to run it) sends commands to the agent software on the managed device using a network management protocol that both the manager and agent understand. The agent responds and then waits (or "listens") for a further command, and so on. The command may be generated by the manager software periodically, without human intervention, and the results

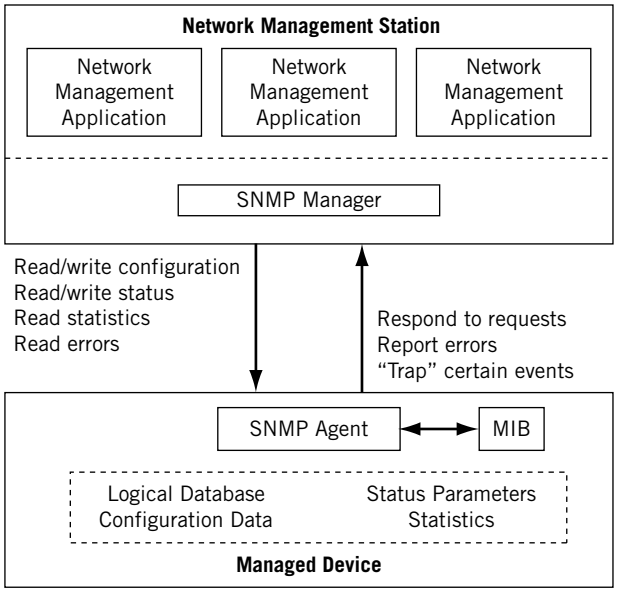


FIGURE 24.3

SNMP model, showing that an agent has access to a MIB in the managed devices.

stored in a manager console database for future reports or reference. Alternatively, the commands may be generated by NOC personnel using the manager console to solve outstanding network problems, perform routine testing, and so forth. In the case of a serious event, such as major link failure, an alarm (called a *trap* in SNMP) is generated without anyone asking. Most servers, hubs, routers, and even end-user devices sold today have built-in SNMP agent software that does not usually have to be purchased separately. The SNMP model of network management is shown in Figure 24.3.

Note that network managers can both monitor the status of the device and actually change the configuration (a dangerous capability that requires careful considerations if it is to be allowed at all). The network management station typically keeps the historical information about the network device (devices have better things to do), and has a number of applications whose main goal is to provide detailed reports about the network's performance, often in a graphical format designed for visual impact.

In addition, all network management standards provide for a special type of agent (known as the *proxy agent*) to provide the manager console with management information about network devices that do not understand the network management protocol. Of course, the network devices must understand *some* type of network management protocol or they would not be manageable at all. But the proxy agent performs a type of gateway function to translate back and forth between the network manager console protocol and the different network management protocol, often proprietary, understood by the network devices accessed by the proxy agent.

The MIB and SMI

The agent software has access to the current value of various *objects* in the managed device. The exact function and meaning of an object, and the relationship of one object to another, is described in the MIB for the managed device. The MIB is a crucial concept in all network management standards, not only in SNMP, although there are many MIBs for devices used on the Internet.

The MIB is a *database* description of all fields (objects) that make up the totality of information an agent can furnish to a manager console when requested. So, a MIB is most often just a piece of paper (RFC) that says things such as "the first field is alphanumeric, 20 characters long, and contains the name of the vendor" and "the fifth field is an integer and contains the number of bad packets received." Not that this is rendered in plain English. A special ISO "language" called ASN.1 (Abstract Syntax Notation version 1) is used to represent all fields of the MIB database in very terse and cryptic language that all MIB implementers understand.

The SMI

The problem with trying to manage all possible network device agents with a single management protocol is that there are so many different types of network devices. Some deal with packets (routers), and some with frames (bridges). Some are quite simple (hubs), and some are very complex (switches). The challenge is to find a way to sort out all of the possible MIB variables in a standard fashion so that any implementation of the network manager console protocol will be able to request the value of any particular object accessible by any agent. Fortunately, standards organizations have all agreed on and defined a standard structure for network management information.

The SNMP developers defined a Structure of Management Information (SMI) tree in RFC 1155. The same SMI is defined in ISO 10165, where it is called the Management Information Model (MIM), and in ITU-T X.720, X.721, and X.722.

MIB information is structured through the use of a *naming tree* known as the SMI conceptual tree. Figure 24.4 shows the SMI conceptual tree with the emphasis on SNMP MIB definitions.

The root of the tree is unlabeled. All branches of the tree from the root have both labels and numbers associated with them. All SNMP MIB objects are under the branch that leads from ISO (1) to Identified Organizations (3) to the Department of Defense (DoD) (6) to the Internet (1). At the lowest branches of the tree are the MIB objects themselves. These are organized into MIB-I (the original SNMP definitions) and MIB-II (extended SNMP definitions).

The system group of MIB-II is probably the most commonly used and easily understood of all MIB objects in SNMP. The System(1) group contains seven objects that provide a general description of the network device. The seven objects are:

- sysDescr(1)—A description of the network device ("router," "hub," etc.)
- sysObjectID(2)—The identifier of the device's private MIB location, if any (discussed more fully in material following)

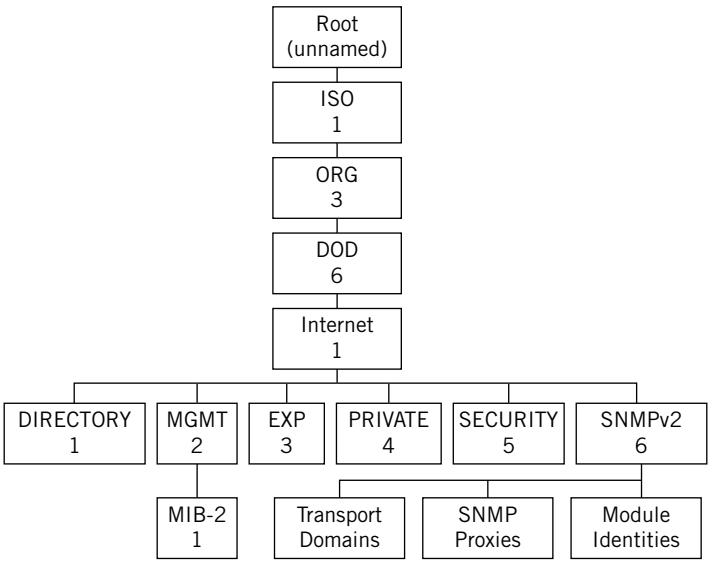


FIGURE 24.4

SMI tree, showing how the names are organized.

- sysUpTime(3)—The time, measured in 100ths of a second, since the network management software (not necessarily the device!) was reinitialized
- sysContact(4)—The name of the local contact person responsible for the network device
- sysName(5)—The name of the manufacturer of the network device
- sysLocation(6)—The physical location of the network device
- sysServices(7)—The services the network device is capable of rendering

The importance of MIBs in network management should not be overlooked. From a single console, a network manager can merely point a mouse at an icon and with a click determine that the device is a router located at 1194 North Mathilda Avenue in Sunnyvale, California; that the person responsible for the device is Walter Goralski; and so on. All of this information is provided over the network, on the fly, from the device itself (as long as it is entered and maintained on the device, of course).

The numbers and labels referred to previously are technically called object identifiers and object descriptors in SMI. The SMI tree is used by the network management protocol to designate objects in the MIB. Object identifiers are numeric, and all SNMP manageable devices commonly found on a network begin with `1.3.6.1...` (shown in Figure 24.4). Identifiers are used by the network management software. Object descriptors, on the other hand, are labels, and all SNMP manageable devices also begin with `ISO.ORG.DOD.INTERNET...`, which is the exact equivalent of the numeric string. This view of the MIB tree is shown in Figure 24.5.

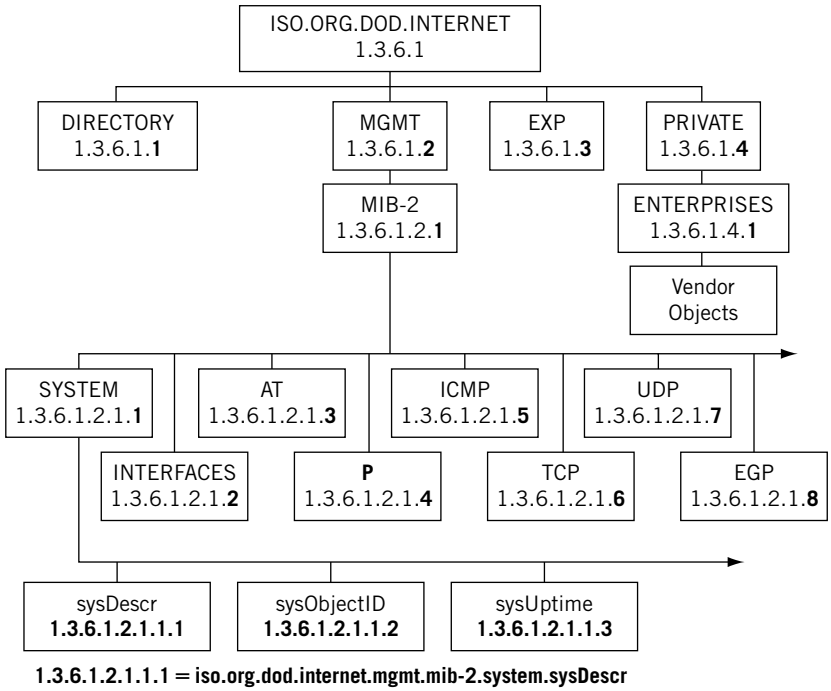


FIGURE 24.5

MIB tree by number and name. The numeric strings can quickly become very long.

As an example of the use of object identifiers, consider the case in which a network manager may need to change the system contact for a particular network device. An SNMP command, in this case a `get` request, is used to retrieve the current value of the sysDescr object. The SNMP message requests the current value of the object `1.3.6.1.2.1.1.1`, which is the object identifier equivalent of the object descriptor `iso.org.dod.internet.mgmt.mib-2.system.sysDescr`. The device knows to reply with the current value of the sysDescr object and no other. If permitted, the network manager can even use the SNMP `set` command to replace to current value of the sysDescr object with the name of the new local contact for the network device (if there is a reason to change it, perhaps to reflect an upgrade).

The MIB

All of the MIB objects in SNMP are defined in ISO ASN.1, a presentation layer (OSI-RM Layer 6) standard syntax. The definition of a managed object in a network device's agent MIB consists of the following seven fields.

- *Syntax*—An ASN.1 data type such as integer, time ticks (hundredths of a second), string, and so on.

- *Access*—If the object is read-write, read-only, not-accessible, and so on.
- *Status*—Objects may be mandatory, optional, obsolete, or deprecated (replaced by newer).
- *Description*—An optional text string describing the object type.
- *Reference*—An optional cross reference to another MIB definition (e.g., a CMIP branch).
- *Index*—If the object is a table, this defines how SNMP access a unique logical row.
- *Defval*—An optional default value assigned to the object.

In the following are two sample MIB object definitions in ASN.1, `ifMTU` and `sysUpTime`.

```
OBJECT:  ifMtu { ifEntry 4 }
Syntax:  INTEGER
Definition:  The size of the largest IP datagram that can be sent/received
   on the interface, specified in octets.
Access:  read-only.
Status:  mandatory.

OBJECT:  sysUpTime { system 3 }
Syntax:  TimeTicks
Definition:  The time (in hundredths of a second) since the network
   management portion of the system was last reinitialized.
Access:  read-only.
Status:  mandatory.
```

The `ifMtu` object is from the interface (`ifEntry`) group, and gives the maximum transmission unit size, a key TCP/IP parameter. The object is the fourth entry in the group (an integer); may only be read by the network manager software, not changed; must be in all SNMP compliant equipment that uses TCP/IP; and gives the size in bytes of the largest IP datagram that can be sent or received by this network device on this particular interface (port).

The `sysUpTime` object is the third in the `system` group, and gives the time the network management agent software has been running. The units are a special type of integer called time ticks. The object is read-only, and must be present.

MIBs are technically just pieces of paper, like a customer database data field description. MIBs must be coded and implemented in the agent software and installed in the network device before the network device can be managed by a manager console. Typically, a MIB is coded by the programmers of the network device's software in a C-language module and compiled into an object-code module with a special compiler known (not surprisingly) as a *MIB compiler*. The MIB object-code module is then linked with the SNMP protocol model to yield the entire executable module, which can be installed in the memory of the network device. All of this is usually done before the network device is sold, of course.

There are exceptions to this rule, however. MIBs exist for a variety of purposes and network types. For instance, a router may have both an Ethernet MIB and a SONET/SDH MIB if the router supports both types of network connections, and even a frame-relay

MIB on the SONET/SDH port of the router. Sometimes, though, a network device may be sold with only an Ethernet port (for example) and then upgraded to provide SONET/SDH connectivity as well, usually through the addition of a new interface card. In this case, the router may have included only the Ethernet MIB because no SONET/SDH MIB was needed. When the new SONET/SDH card is added, the SONET/SDH MIB must be added as well.

Not all modifications to network devices involve hardware. In some cases, a new MIB may have to be installed when a new software feature is activated on the network device. In many SNMP implementations, the *extensible MIB* may be activated or installed over the network without even being present at the network device site.

RMON

One additional aspect of SNMP MIBs should be discussed, in that this concept is extremely helpful in managing large networks. There is a potential problem with managing SNMP devices on a network over the network itself (security is another matter). The problem is simply this: What if the link to the network device is down? How is the status of the network device to be determined under these conditions? The answer is provided by means of a special optional MIB: the *RMON MIB*. RMON stands for "remote monitor," and this MIB provides for a dial-in port to the network device that may be used by the manager console to communicate with the network device regardless of other network link availability.

RMON may also be used with leased lines to provide another benefit for large IP networks. The larger the enterprise network, the more network devices there are that need managing. Network managers will try to monitor network device performance and workload to prevent congestion on the network. The problem is that all of these SNMP messages flowing over the network back and forth to all of the network devices can add a considerable load to a network at the worst possible time, when things are going suspiciously wrong. If RMON is configured to run on separate leased lines to critical network devices, the SNMP messages add no load at all to the enterprise network itself.

Unfortunately, not many organizations can afford the additional expense of the necessary leased lines to many of these important network devices (usually the routers). Still, RMON remains a useful option for heavily loaded or delay-sensitive IP networks.

The Private MIB

Standard MIB objects are designed for a wide variety of technologies and network devices. These MIB objects cover a large range of possibilities, but there are always situations and conditions that a network manager should be aware of that are not covered by a standard MIB object. These are usually very low-level, device-specific hardware functions, such as whether a network device's cooling fan has failed, whether the device has battery backup or a redundant power supply, or any of a number of other vendor hardware-implementation choices and options.

To cover all of these vendor-specific situations, the SMI conceptual tree includes a branch for private MIB extensions. The SMI path to the private MIB is `1.3.6.1.4.1`. This leads to the *enterprise* branch of the SMI tree, where each vendor may obtain a branch number (identifier) and label (descriptor) from the Internet Assigned Number Authority (IANA) for the vendor's private MIB. For example, all IBM private MIB objects reside at `1.3.6.1.4.1.2...` on the SMI tree because "2" is IBM's enterprise number. Cisco routers use `1.3.6.1.4.1.9...`, Hewlett-Packard has `1.3.6.1.4.1.11...`, and so forth. More than 700 enterprise code numbers have been assigned by the IANA, showing the wide availability of SNMP-compliant products.

This system of private MIBs makes sense because only the manufacturer of the network device could possibly know whether the device even has a cooling fan, battery backup, or other hardware feature. Obviously, a network manager would like to know if a device's fan has failed, especially if the device is in a closet where it may overheat and fail after a few hours. The private MIB offers a way of allowing this information to be accessed by the network manager.

SNMP manager software will generally have no concept of just where the private MIB objects are and what these objects represent. Some vendors would actually "hide" their private MIB descriptions by limiting their availability, and just what the number `2` in a private MIB field might mean (Status code? Error code? Two minutes to failure?) often remained a mystery. In most cases, this means that this vendor's network device could only be completely manageable using that vendor's network manager software, which would have a built-in description of this private MIB. Private MIBs are an effective way to "lock in" a company to using only a specific vendor's SNMP software as a network manager.

Few companies go to that extent anymore. But the problem of how any particular manager console software could know just where any vendor's private MIB is located and what the vendor's private MIB means still exists. This is where the system group `sysObjectID` object can be helpful. Accessing the object `1.3.6.1.2.1.1.2` (the second object in the system group: `sysObjectID`) from the management console will return a string such as `1.3.6.1.4.1.999.1.1...`. This is, of course, the location of the private MIB objects for the vendor of the particular device. Further requests to that SMI tree location might yield the private MIB description implemented by that vendor (`1` means fan failure, `2` means fan normal).

Manufacturers may extend private MIBs with as many objects in whatever structure they desire. Many vendors publish (on the Internet) their private MIB descriptions so that makers of SNMP management console software can easily build in private MIB support without having to follow `sysObjectID` links.

SNMP OPERATION

All of the foregoing discussion on SMI, MIBs, and private MIBs applies equally to any standard network management package that may be used on a network. Granted, there are a few differences between SNMP network management terminology and the

others. Specifically, the SMI objects in network management protocols other than SNMP may not all necessarily start with `1.3.6.1...` because these are by definition TCP/IP Internet objects and the MIB in CMIP is referred to as MIM (Management Information Model). There are other minor differences as well, but the point is that all of the previous material and concepts apply to network management in general.

However, this section will deal entirely with the specifics of SNMP as the most widespread, cost-efficient, and viable network management standard for IP networks in use today. For the remainder of this section, SNMP without qualification means SNMPv1. SNMPv2 and SNMPv3 will always be qualified with the version number.

SNMP was invented to manage routers on the Internet, and early versions of SNMP had few MIB objects suitable for managing other network devices. The latest SNMP MIB definitions have been extended to include objects defined for most LAN and WAN technologies, even ATM and frame relay. SNMP was initially intended as an interim solution until ISO's CMIP network management standard was completed, at which time SNMP was supposed to merge with CMIP. But SNMP has had such success independently of CMIP that this is unlikely to happen.

SNMP is part of the TCP/IP protocol stack and is considered a standard TCP/IP application like FTP or Telnet. Of course, SNMP is a very special type of application, one that is seldom bundled with TCP/IP software as FTP and Telnet are. Due to its TCP/IP origins, the original SNMP did suffer from one annoying limitation that severely hampers the use of SNMP for managing mission-critical networks that should not fail.

The limitation is bound up with the fact that SNMP is defined as a request–response protocol, similar to DNS. Each message sent was expected to generate a reply before the next request was sent. This made perfect sense for SNMP: Why send a stream of messages to a device that has failed? And like any request–response protocol, SNMP used speedy and connectionless UDP for its messages.

But there is a price to be paid for connectionless speed. What if an SNMP message is sent and no reply received? There can be at least three causes. First, the data may have been lost by the network on the way to the destination (due to network faults or congestion). Second, the destination network device itself may be down or powered off. Third, the data may have been lost by the network on the way back from the destination (for the same reasons as the first two causes).

On the other hand, connection-oriented networks and applications that first establish a connection across the network with a remote device have a better chance of figuring out just what is wrong if a reply to a particular message is not received. If a device accepts a connection request, it means the device is turned on and ready to communicate and the network between the two devices linked by the connection is up and running. It is important to realize that this knowledge is established *even before any messages have been sent from a source to a destination*.

Obviously, toward obtaining a more robust and effective network management protocol network, managers would rather that SNMP be connection oriented, as is clear from the previous discussion. A lot could be found out just from establishing a connection between a manager console and a network device's agent. However, SNMPv1 was

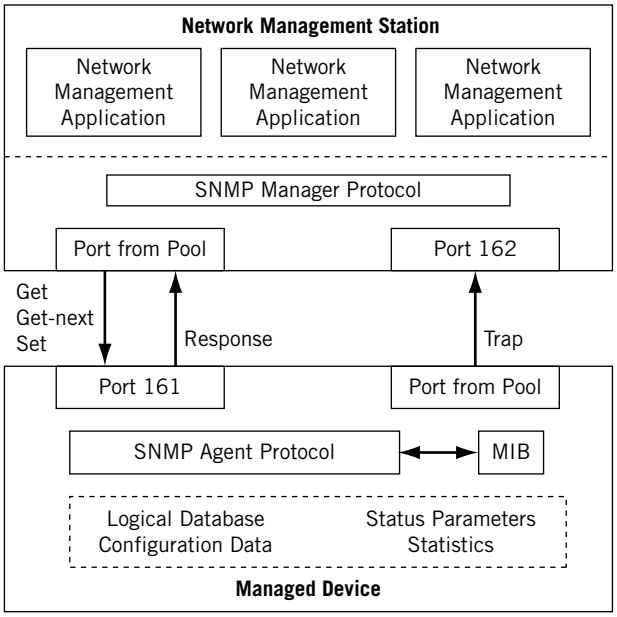


FIGURE 24.6

SNMPv1 protocol operation, showing ports for the five SNMP message types.

a connectionless TCP/IP application, which limited its effectiveness on many enterprise networks. The operation of the SNMPv1 protocol is shown in Figure 24.6.

SNMP is an extremely simple protocol. There are only five types of messages defined: `GetRequest` (or `Get`) to ask an agent to return the current value of an object (based on the SMI tree), `GetNextRequest` (or `GetNext`) to ask an agent to return the current value of the very next object, `GetResponse` (or `Response`) to return the current value of an object to the manager, `SetRequest` (or `Set`) to tell an agent to replace the current value of an object with a new value, and `Trap` to allow an agent to send a message to a manager without being asked.

The agent device accepts SNMP requests on port 161 and replies using that port. The manager chooses a source port from a pool, often restricted to SNMP only. Traps are sent via port 162 on the manager, also using a source port chosen from a pool.

Traps are used to address another quirk of SNMP. Generally, agents tell the manager console absolutely nothing without being asked. In view of this, it is normal for the SNMP manager software to periodically generate `GetRequest` messages to every manageable device's agent on the network just to ensure that everything is all right. This process is known as *SNMP polling,* and not only adds traffic to the network, but means that long periods of time may elapse between successive polls on a complex SNMP enterprise network.

Traps help to remedy this situation. These are messages sent from the agent to the manager without waiting for a poll. There are seven generic trap types that include such events as link failures and the fact that the agent network device is being reinitialized, and so on. An *enterprise-specific* trap type is included to allow vendors to extend traps to include other events (such as fan failure, battery backup activated, etc.).

All SNMPv1 messages consist of a message header and the actual SNMP protocol data unit (PDU). The header only contains the version number (1) and the community string (default is `public`).

The PDUs contain the command specifics and their operands. The fields are variable in length, and end with strings of *variable bindings,* which are the pairs of objects and their current values the network management system has asked to see. On the way to the managed device, these bindings are typically filled in with the zero or blanks, and naturally they come back with the current values filled in. The structure of the SNMPv1 PDU is shown in Figure 24.7.

- *PDU Type*—Specifies the PDU Type: `GetRequest`, `GetNextRequest`, `GetResponse`, and `Setrequest`.
- *Request ID*—A field used to associate SNMP requests with the proper response.
- *Error Status*—Only a `GetResponse` sets a numeric error code in this field. Otherwise, the field is zero.
- *Error Index*—Associates the error code with a particular object in the bindings. Only a `GetResponse` sets a numeric index in this field. Otherwise, the field is zero.
- *Variable Bindings*—The data field of the Simple Network Managment Protocol PDU. Each pair associates the object with its current value, except of course in the `GetRequest` and `GetNextRequest`.

`Traps` are not included in the figure because in SNMPv1 they have a distinctive (and annoying) structure all their own. In the previous discussion, at least two limitations of SNMPv1 have been identified. First, SNMPv1 is connectionless, which means that SNMP is much less effective than it could be. Second, SNMP must poll devices in most cases for effective network management because the traps are few and not very helpful.

There is a third aspect of SNMP that makes the protocol less effective than it could be for managing large IP networks, especially portions of the Internet. This is the fact

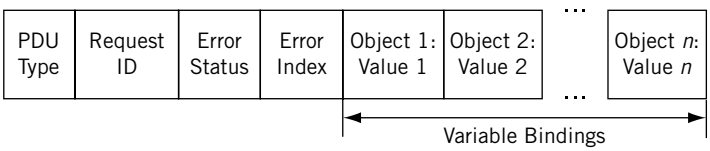


FIGURE 24.7

SNMPv1 PDU. Variable bindings allow the response to deliver a lot of information in one message.

that SNMPv1 had only rudimentary password and authentication features and even lacked a good encryption technique.

The greatest threat that network management poses to a network, ironically, comes from exploiting remote configuration capabilities, one of the most useful things in network management. Activating additional ports on hubs and routers, changing IP addresses, and modifying other operational functions over the network rather than by actually having a technician present at the network device location is a much sought-after feature of network management. But the routine practice of remote configuration is tied up with the establishment on the network of secure network management protocols to prevent hackers and other unauthorized persons from making such changes to these devices.

SNMPv1 has only rudimentary features that can be used to try to prevent this from happening. The SNMP protocol does include the use of a simple password scheme, known as the *community string*. All SNMP messages from a management console to an agent must include a community string field that is compared by the agent with the community string configured at installation in the network device. If the community strings do not match, the agent presumes that the message is not from the legitimate network management console software and discards the message.

The problem with expecting SNMP community strings to provide adequate password protection against unauthorized agent access is twofold. First, many agents are simply configured to respond to the community string `public`, which is essentially the SNMP default and might not be changed. Of course, hackers will quickly determine this fact and make immediate use of this. Second, even if the community string is altered to a more enterprise-specific string such as `Example Inc.`, the SNMP messages exchanged constantly on the enterprise network due to the SNMP polling process will make no effort to hide this fact: The community strings are not encrypted in SNMP but sent in plain text.

The problem of authentication is related to the use of passwords for network management. All SNMPv1 agents accept any SNMP messages and commands if the community string is correct. With an authentication scheme for network management, more should be needed for an agent to accept messages as proper commands sent from a valid network management console. Matching passwords is not enough: The message must come from the IP address of the network management console or consoles.

SNMPv2 Enhancements

SNMPv2 was widely anticipated in the network management community since its initial proposals. SNMPv1 also suffered from an annoying problem with the request–response system of polling. If one variable was not in the agent's database, the entire operation failed. In addition, as MIB grew and grew, SNMPv1 responses often exceeded the maximum size of a message (UDP doesn't fragment) and the operation failed.

To address these issues, SNMPv2 added a `GetBulk` message to the SNMP repertoire, which allowed the device to supply as much information as it could in response to the request. There was also a greatly expanded list of error codes used when an SNMP request failed.

`Inform` allows one network management system to trap information sent by another network management system and then get a response. In addition, the format of the `Trap` was changed to make it more like the other PDU type.

SNMPv2 can still run as a connectionless UDP application on IP networks. But implementers have the option of making SNMPv2 a connection-oriented TCP application. In addition, SNMPv2 includes very robust and standardized methods for true passwords, authentication, and encryption.

Yet the use of SNMPv1 remains common on the Internet. The problem with SNMPv2 is exactly the opposite of the simplicity of SNMPv1: SNMPv2 is very complex. This complexity translates to implementation expense, not only in the management console software but in the agent software installed by every vendor of SNMP-manageable network equipment. For very simple networks, SNMPv2 is overkill.

In addition, SNMPv2 is incompatible with SNMPv1. The message formats are different, and there are two new message types (`GetBulk` and `Inform`). RFC 1908 recommends the use of proxy agents, or simply running both when this incompatibility becomes an issue. Many Internet devices, such as routers, make use of SNMPv1 or SNMPv2 (or both) as a configuration option.

SNMPv3

A few words should be said about SNMPv3. SNMPv1 had little or no security to speak of, and SNMPv2 adds security to the basic operation of SNMP. However, SNMPv3 will essentially make network management and SNMP part of the overall *security framework* for a network. SNMP will have very strict requirements for authentication, encryption, and privacy of information. Discussions of SNMPv3 are best handled by texts devoted to the topic of security.

QUESTIONS FOR READERS

Figure 24.8 shows some of the concepts discussed in this chapter and can be used to answer the following questions.

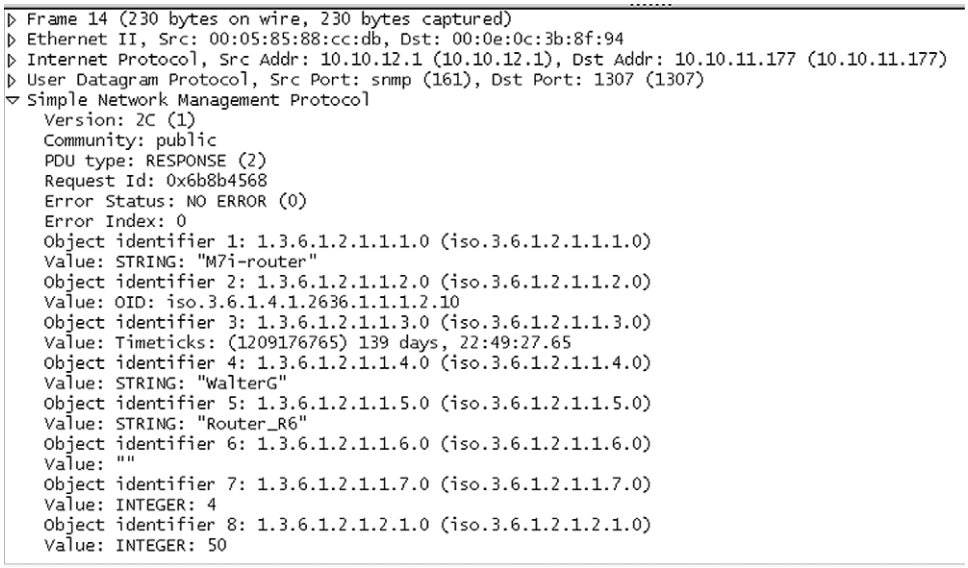


FIGURE 24.8

Ethereal capture of an SNMP response message. Note the object identifiers.

1. Which version of SNMP is used here?
2. Which router IP address and port are responding?
3. Express the SMI tree to the sysDescr group in English instead of numbers. It starts with "iso.org..."
4. The actual time ticks value of 1209176765 is interpreted. What does this value represent?
5. Where is the response telling the management application to go for more device-specific information?