CS 556 - Networks II
Internet Teaching Lab (MCS B-24)
Simple Network Mgmt Protocol (SNMP)

Simple Network Management Protocol

What you will learn in this lab:

- Details of the SNMP protocol.
- Contents of a MIB (Management Information Base).
- How to use an SNMP manager to query an SNMP agent.
- How to use SNMP to learn about the network configuration.

Prelab & Extra References

NET-SNMP tools: Read about the NET-SNMP software tools at http://net-snmp.sourceforge.net/. Specifically, read about the following commands, which are used throughout this lab, in the tutorial section:

- snmpd
- snmpget
- snmpgetnext
- snmpwalk

Make sure to read the following documents (SNMP overview and SNMP tutorial):
http://www2.rad.com/networks/1995/snmp/snmp.htm
http://www.dpstele.com/layers/l2/snmp_l2_tut_part1.html

SNMP

The Simple Network Management Protocol (SNMP) framework provides facilities for managing and monitoring network resources on the Internet. The SNMP framework consists of SNMP agents, SNMP managers, Management Information Bases (MIBs), and the SNMP protocol itself. An SNMP agent is software that runs on a piece of network equipment (host, router, printer, or others) and that maintains information about its configuration and current state in a database. An SNMP manager is an application program that con tacts an SNMP agent to query or modify the database at the agent. The organization of information in the database is described by so-called Management Information Bases (MIBs). Finally, the SNMP protocol is the application layer protocol used by SNMP agents and managers to send and receive data.

1 Managment Information Base (MIB)

The managed objects maintained by an agent are specified in a Management Information Base (MIB). A MIB file is a text file that describes managed objects using the syntax of ASN.1 (Abstract Syntax Notation 1), a formal language for describing data and its properties. A network device may have multiple MIB files. All current Internet devices support MIB-II, which specifies about 170.

In a MIB, each object is assigned an object identifier (OID) , which is used to name the object. An OID is a unique sequence of integers separated by decimal points. An example of an OID is 1.3.6.1.2.1.4.6. Each OID has a textual description. For example, the textual description of OID 1.3.6.1.2.1.4.6 is iso.org.dod.internet.mgmt.mib-2.ip.ipForwDatagrams, or simply ipForwDatagrams . In MIB-II, the ipForwDatagrams object is described as an integer counter that stores the number of forwarded datagrams at a router. When an SNMP manager requests an object, it sends the OID for an instantiation of the object to the SNMP agent.

Managed objects are organized in a tree-like hierarchy and the OIDs reflect the structure of the hierarchy. Each OID represents a node in the tree. For example, the OID 1.3.6.1.2.1.4.6 is a child node of OID 1.3.6.1.2.1.4. The OID 1.3.6.1.2.1 (iso.org.dod.internet.mgmt.mib-2) is at the top of the hierarchy for all managed objects of the MIB-II. Manufacturers of networking equipment can add product specific objects to the hierarchy. The object hierarchy of the MIB tree is illustrated in Figure 1.

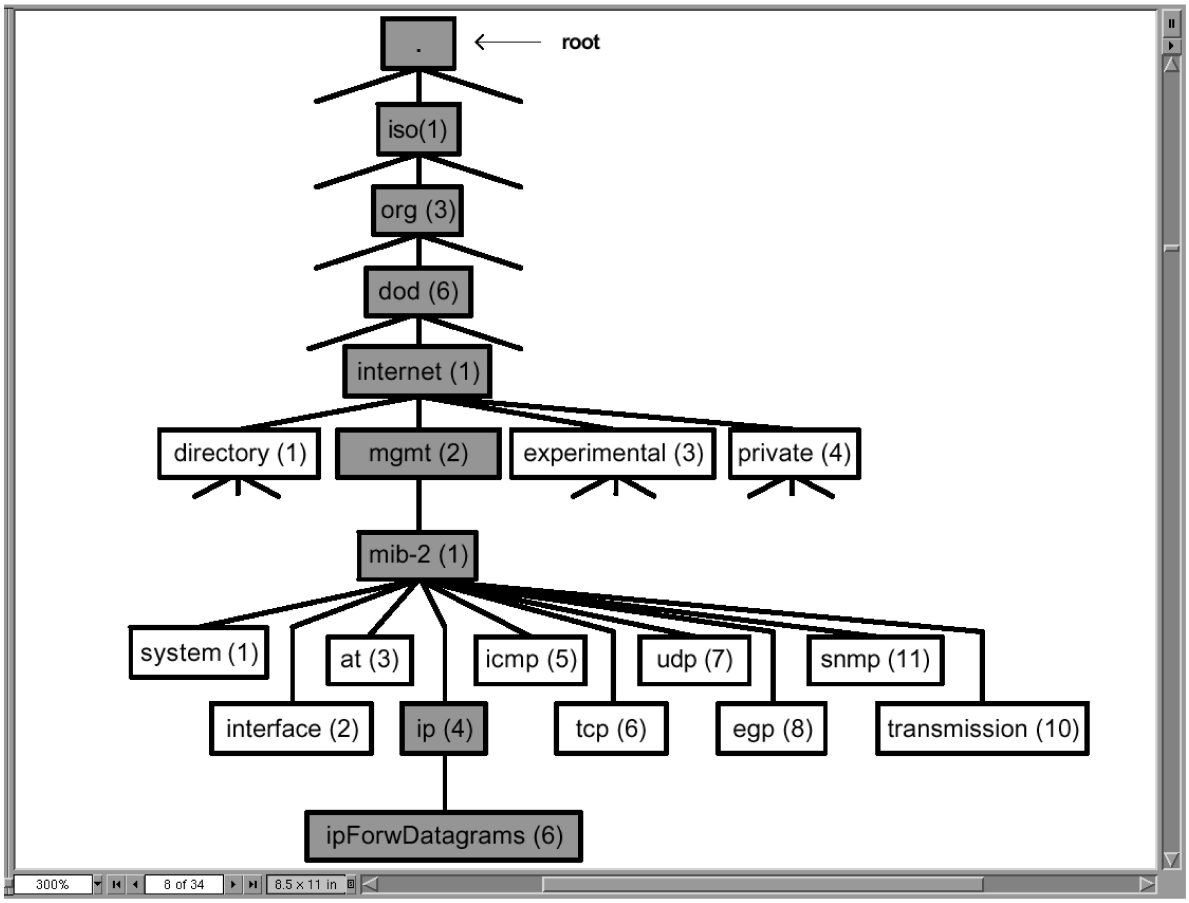

Figure 1: MIB Tree

Exercise 1-a: Definition of managed objects in a MIB

In this exercise, you explore how managed objects are defined in MIB files, and how MIB files and objects are organized. On the Linux PCs, the MIB files are stored in the directory /usr/share/snmp/mibs. When an SNMP agent starts, the MIB files are compiled and loaded into a database.

Each MIB file specifies a set of managed objects. The objects are specified following the SMI format (Structure of Management Information), which defines, for each managed object, the OID, the textual descriptor for the OID, and the data type ASN.1 syntax.

1. On localhost, go to directory /usr/share/snmp/mibs/ and explore the MIB files that are relevant to this lab.
   - RFC1155 -SMI.txt
   - RFC1213 -MIB.txt
   - IF-MIB.txt
   - INET-ADDRESS -MIB.txt
   - IP-FORWARD -MIB.txt
   - IP-MIB.txt
   - TCP -MIB.txt
   - UDP -MIB.txt
2. Find and save the definition of the following managed objects (hint - `grep` and `less` are useful tools):
   - ipForwarding
   - sysName
   - icmp
3. **REPORT**. For each object above, answer the following questions individually:
   (a) What is the numerical OID?
   (b) What is the full textual descriptor?
   (c) Does this object have children in the MIB hierarchy?
   (d) Characterize the data type of the object.

Exercise 1-b: View MIB structure using snmptranslate

You can use the command snmptranslate to explore the structure of a MIB tree. The command snmptranslate can translate between numerical and textual representation of objects. In addition, the command can display the tree hierarchy in textual form.

1. On localhost, run the following commands and save the output to a file.

   % snmptranslate -On .1.3.6.1.2.1.7.4

   % snmptranslate icmp

   % snmptranslate -Tp -OS tcp

   NOTE: The first command displays the textual representation of an OID (Note the leading '.' (period)) The second command performs the opposite translation. By default, snmptranslate adds the prefix .iso.org.dod.internet.mgmt.mib-2 (or .1.3.6.1.2.1) to the argument. The third command prints a diagram of the portion of the MIB hierarchy rooted at node tcp.

2. Display and save the MIB hierarchy rooted at nodes interfaces and ip.

3. **REPORT**. Answer the following questions individually:

   (a) What are the routing protocols recognized by SNMP?
   (b) List the states of a TCP connection?
   (c) How might one view the whole tree (ie. rooted at .iso) using snmptranslate?

2 SNMP agents and mgrs: Retrieving MIB objects

In this part of the lab, you explore the interactions between an SNMP agent and an SNMP manager. You start SNMP agents on the Linux PCs and Cisco routers, and you run an SNMP manager on a Linux PC. An SNMP manager and an SNMP agent communicate using, of course, the SNMP protocol.

The SNMP message types used in this lab are supported in all three versions of SNMP. For the most part of this lab, we work with SNMPv1, which is still the default version on most systems. SNMPv2c and SNMPv3 have additional message formats and richer functionality.

With the exception of trap messages, the exchange in SNMP is initiated by an SNMP manager by sending a request to an SNMP agent. The SNMP agent replies in a response message. The mess age types are as follows:

Get-request. Requests the values of one or more objects from an SNMP agent.

Get-next-request. Requests the value of the next object, according to a lexicographical order ing of OIDs .

Set-request. A request to modify the value of one or more objects at the agent.

Get-response. Sent by SNMP agent in response to a get-request, get-next-request, or set-request message.

Trap. An SNMP trap is a notification sent by an SNMP agent to an SNMP manager, which is triggered by certain events at the agent .

In this part of the lab, you enable and access SNMP agents on a Linux PC and on a Cisco router, and learn to interpret SNMP messages exchanged between the SNMP manager and the SNMP agent.

The general syntax of the SNMP manager commands of the NET-SNMP software is as follows (details in man page):

```
snmpcmd [-v version] agent -c community objectid
```

Exercise 2-a: Send SNMP requests to a local SNMP agent

In this exercise, the SNMP agent and SNMP manager are run on the same system. You run the SNMP manager commands snmpget, snmpgetnext, snmptranslate and snmpwalk to retrieve and display the contents of MIB objects managed by the local SNMP agent.

1. Start an SNMP agent on localhost by invoking the command: `snmpd`. This command starts an SNMP agent. The command loads the management database with the MIB files in directory /usr/share/snmp/mibs and configures the agent with file /usr/share/snmp/snmpd.conf.

2. Execute the following commands to access scalar variables and record the output:

   % `snmpget localhost -c public system.sysDescr.0`

   % `snmpget localhost -c public udp.udpOutDatagrams.0`

   % `snmpget localhost -c public udp.udpInDatagrams.0`

   Note the ".0" suffix at the end of the OID. This suffix is a reference to an instantiation of the referred object which stores the value. We will refer to the instantiations as variables. For example, 1.3.6.1.2.1.7.4.0 refers to the variable that contains the value for object udp.udpOutDatagrams. Variables that store a single value are called scalar variables . If an object consists of a sequence of values, then the suffix indicates the index in the sequence. For example, interfaces.ifTable.ifEntry.ifDescr.2, refers to the second entry of the sequence of values in interfaces.ifTable.ifEntry.ifDescr. - The community string of the agent is set to public . This is often the default community string for the read-only community.

3. Execute the following commands to access non -scalar variables:

   ```
   % snmpget localhost -c public interfaces.ifTable.ifEntry.ifDescr.1
   % snmpget localhost -c public interfaces.ifTable.ifEntry.ifDescr.2
   ```

   Continue to increase the index, until you get an error message. Note what happens if you query the value as a scalar value, that is, using the ".0" suffix.

4. Use the command snmpget to request multiple object values with the same command:
   ```
   % snmpget localhost -c public system.sysUpTime.0 system.sysName.0
   ```

5. The command snmpgetnext retrieves the value of the next variable according to the lexicographical ordering of OIDs. In the object hierarchy tree, the command retrieves the next object in the tree.

   Use the command snmpgetnext to retrieve all objects under the node *at* ("address translation") in the MIB tree. The object at defines a group of objects that store the ARP table of the local system. Start with the command:

   ```
   % snmpgetnext localhost -c public at
   ```

   This value retrieves the variable name of the first entry of the ARP table, as well as its contents.

   Issue an snmgetnext c ommand that retrieves the second entry of the ARP table. (The output of the command above displayed the identifier that stores the first ARP table entry. Provide this identifier as the argument of the snmgetnext command). Repeat until the entire table is displayed.

6. **REPORT**. Answer the following question individually:

   (a) How can you know that you have reached the end of the group at ?

7. The command snmpwalk can be used to retrieve the values of all objects under a particular location in the MIB object hierarchy tree. The retrieval of a complete subtree is referred to as "walking the MIB." Use snmpwalk to obtain the values of all the objects under the nodes system and interfaces.

   ```
   % snmpwalk localhost -c public system
   % snmpwalk localhost -c public interfaces
   ```

8. Now try retrieving information from remote hosts. For example,

   ```
   % snmpget 10.0.C.D -c public system.sysDescr.0
   % snmpget 10.0.C.D -c public udp.udpOutDatagrams.0
   ```

   where C.D are network and host of other workstations in the lab.

2.1 Exercise 2-b: Send SNMP request to a Cisco router

All Cisco routers support SNMP. In most scenarios, routers play the role of an SNMP agent, but it is also possible to run SNMP manager commands on a router. The set of IOS commands to configure SNMP is quite large, and this lab uses only the basic commands.

1. Telnet to your router.
2. Execute the following commands to enable an SNMP agent.
   - RouterX¿ **enable**
   - Password: **¡enable secret¿**
   - RouterX# **configure terminal**
   - RouterX(config)# **snmp -server community public**
   - RouterX(config)# **end**

   These commands start an SNMP agent, and set the community string to public . By default, access to the MIB of the router is read-only. With this command, the SNMP agent responds to SNMP manager that provides the community string public .
3. Use the command 'show running-config' to display the communities that are configured at the router. You may need to save the running state first by typing 'write mem'.
4. From your localhost, query the SNMP agent on your router using snmpget to obtain some of the values of the following object identifiers:
   - system.sysDescr.0
   - icmp.icmpInRedirects.0
   - icmp.icmpInSrcQuenchs.0
   - udp.udpInDatagrams.0
   - udp.udpNoPorts.0

3 SNMP Traps

A trap is a notification mechanism where the occurrence of certain events at an SNMP agent triggers the transmission of an SNMP trap message to an SNMP manager. SNMP has several predefined traps that include events when the system is rebooted, an SNMP agent is started, when a link goes up or down, and when an unauthorized access has been attempted.

In this part of the lab, we use the SNMP manager command snmptrapd to handle traps. The manager listens on UDP port 162 for SNMP trap messages and logs SNMP trap messages that are received.

Exercise 3-a: Configure SNMP traps on a Cisco router

Next, set enable traps on your Cisco router.

1. On your router, enable SNMP traps and have all SNMP traps sent to your workstation by typing the following commands:
   - Router3¿ enable
   - Password: ¡enable secret¿
   - Router3# configure terminal
   - Router3(config)# snmp-server host 10.0.c.d public
   - Router3(config)# snmp-server enable traps
   - Router3(config)# end
2. Use the command 'show running-config' to display the communities that are configured at the router. You may need to save the running state first by typing 'write mem'.

Exercise 3-b: Capture and handle SNMP trap messages

Here, you set up PC1 as the SNMP manager that collects and displays trap messages. **Please stop and notify the TF when you have reached this point.**

1. On localhost, start the snmptrapd SNMP manager program with the option '-P'. This option specifies that messages are printed in a terminal window:

   ```
   % snmptrapd -P
   ```

   Once you have seen two (2) events, save the output of the command to a file.
2. **REPORT**. Answer the following questions individually:
   (a) Include the snmptrapd output in your report.
   (b) Provide an explanation for the snmptrapd output.
   (c) Which trap types do you observe in the experiment? What are they telling you?
   (d) How many trap mess ages are sent each time an event is triggered?
   (e) What is the *absolute* OID of keepalive messages, that is, the OID rooted at .iso?

4 Applications of SNMP

In this part of the lab, you perform two network management tasks with SNMP. The first task is to query the status of network connections on a remote PC, the second task determines the return path between two remote hosts.

Exercise 4-a: Observe TCP connections on a remote Linux PC

A network administrator can use the SNMP manager to observe the status of TCP connections on remote hosts where an SNMP agent is running. The collected information can be used to monitor the activities on remote systems.

1. Set up an ssh session to a workstation in group (x+1)%3, by typing the following:
   `% ssh 10.0.`$(x+1)\%3$`.d`
2. From localhost, query the TCP portion of the MIB at the host to which you ssh'ed, with the command:
   `% snmpwalk 10.0.`$(x+1)\%3$`.d -c public tcp`
3. Save the output into a file.
4. Close the ssh session(s).
5. **REPORT**. Answer the following questions individually:
   (a) In the saved output, find any records that relate to the Telnet sessions, and include them in the lab report. What are they telling you?
   (b) How can this information be used by a network administrator?

Exercise 4-b: Determine the path between two hosts

The command traceroute determines the forwarding path taken by a packet from a source to a destination. With SNMP it is possible to determine the path between any pair of hosts. In this exercise, you use SNMP to find the forwarding path from host A (in group $(x+1)\%3$) to host B (in group $(x+2)\%3$). For example, if in group (3) then you will determine the return path from group $A = 1$ to group $B = 2$.

1. On localhost, execute the following command to find the next hop of Host A on the path from Host A to Host B:

`% snmpwalk host_`A `-c public \`

`ip.ipRouteTable.ipRouteEntry.ipRouteNextHop`

The output of the command consists of entries of the form:

ip.ipRouteTable.ipRouteEntry.ipRouteINextHop.NetIPAddress = IpAddress: NextHopIPAddress

There is one line for each routing table entry. NetIPAddress is the destination address and NextHopIPAddress is the IP address of the next hop router to destination NetIPAddress. Determine the matching routing table for destination Host B from the output, and obtain the corresponding NextHopIPAddress.

2. Repeat the process described above, but replace IP address of host A in the snmpwalk command with the IP address of the next hop, NextHopIPAddress , you obtained above. Repeat these steps, until a router is reached that is directly connected to Host B.

3. **REPORT**. Answer the following questions individually:

   (a) How does this process to determine the path from one host to another?

   (b) What is the return path you discovered? (Please include source and destination IP in your path)

This lab is based on the SNMP lab from the ITLab at University of Virginia.