SNMP

Simple Network Management Protocol

A Standard Protocol for Systems and Network Management

Network Management — the problem: a scenario

- BAD:
  - User: the server has been down for an hour, and printing has stopped working, and the connection to the Internet is down.
  - System manager: Oh, really? Well, let's have a look and see what we can do.

Network Management — the problem: a better scenario

- BETTER:
  - User: the server has just gone down, and printing has stopped working, and the connection to the Internet is down.
  - System manager: Yes, we have been working on it; we know that this is a problem with our main switch, and the guys from Cisco are working with us to solve the problem.

Network Management — the aim

- BEST:
  - The user does not see any problem
  - The system managers could see from trends in the network traffic that there was a problem, e.g., number of bad packets
  - The problem was fixed before the users were aware of it.

Network Management — its aims

- Networks contain equipment and software from many vendors
- Many protocols
- One company's solution can manage their equipment, but not all the rest
- Need a standard way to communicate information about performance, configuration, accounting, faults and security.

Possible solutions to Network Management that do not use SNMP

- There are programs that check the availability of network services, e.g.:
  - mon: http://www.kernel.org/software/mon/
  - nagios: http://www.nagios.org/
- Log monitoring software such as logwatch
- Software to analyse network traffic by examining packets: http://www.ntop.org/
- There are other home-made programs and scripts possible, e.g., using cron or scheduler
- A good approach is to use many monitoring methods together

configuration management: cfengine

- cfengine is a sophisticated system for setting up and maintaining computer systems
- You set up a single central system configuration
  - this determines how every computer on your network is configured
  - interpreter runs on each host parses this file
    - any deviation from the required configuration is automatically fixed (if you choose)
- can manage large or huge networks
- Runs on Windows, Linux and Unix
- http://www.cfengine.org/

Automated installation

- systemimager: automates Linux installs: http://www.systemimager.org
- kickstart: automate Red Hat installation
- Symantec Ghost: use multicast to distribute system images

SNMP — how it was born

- In 1980's, networks grew, hard to manage
- Many vendors, many protocols
- Many saw a need for standard
- SNMP Proposed to IETF (Internet Engineering Task Force) as a Request for Comments (RFC)
- RFCs are the standards documents for the Internet

SNMP: An IETF standard

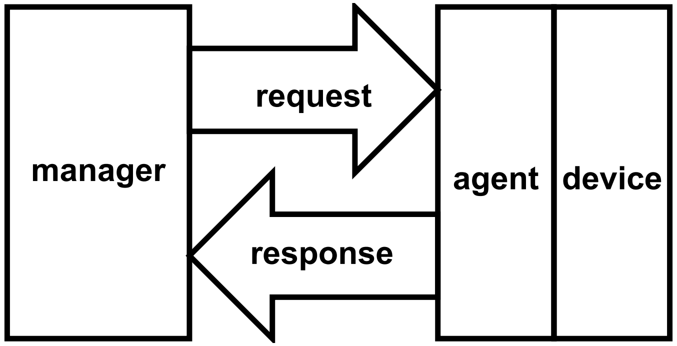
- There are three versions of SNMP
- SNMPv1: RFC 1157
  - Basic functionality, supported by all vendors
- SNMPv2: RFC 1905, 1906, 1907
  - Some useful additional features; supported by many vendors
- SNMPv3: RFC 1905, 1906, 1907, 2571, 2572, 2573, 2574, 2575.
  - Still a **proposed standard**
  - Adds strong authentication
  - Supported by Net SNMP and some Cisco products

Managers and Agents

- A network management system consists of two software components:
- Network **manager**
  - often called a **NMS** (Network Management Station)
- **Agent**
  - Software that runs on the **device** being monitored/managed

Managers and Agents 2

- simple request -> response protocol



manager

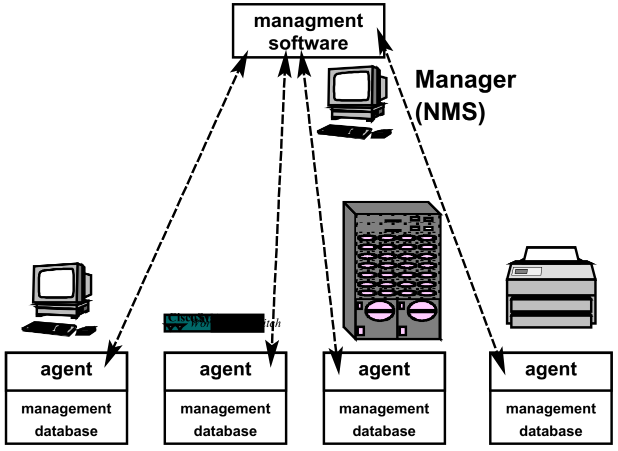
request

response

agent

device

Managers and Agents 3



managment software

Manager (NMS)

agent

management database

agent

management database

agent

management database

agent

management database

Managed Devices

SNMP runs on UDP

- UDP = User Datagram Protocol
- Unreliable (no acknowlegment in UDP protocol)
- Low overhead
- Won't flood a failing network with retransmissions
- UDP port 161 for sending, receiving requests
- UDP port 162 for receiving traps

SNMP Communities

- SNMPv1, v2 use a "community" as a way of establishing trust between manager and agent
- This is simply a plain text password
- There are three:
  - Read-only (often defaults to "public")
  - Read-write (often defaults to "private")
  - Trap
- Change from default for production!!!!!!!!!!!

Authentication in SNMPv3

- Sophisticated authentication system
- User based
- Supports encryption
- Overcomes the biggest weakness of SNMPv1, v2 community strings

What is a **managed object**?

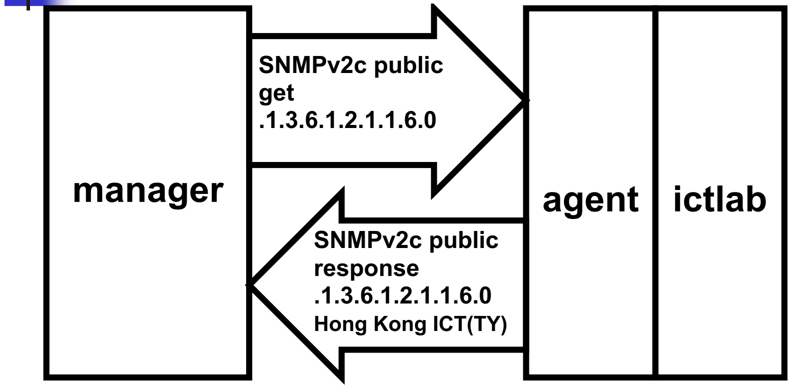
- A better name is **variable**, but called **managed object** more often
- You have looked at the managed object system.sysUpTime.0 in the lab
  - Gives time since agent was started
- Is (generally) located on the **agent**
- A managed object has one object identifier (**OID**)
- Carries one scalar value, or a **table** of related information
- Management involves monitoring and setting values in these managed objects
- Agent software changes SNMP requests to action to read or set the requested value(s)

Example: getting location

- The Net-SNMP tools provide a tool **`snmpget`** that directly implements the get request from a manager
- Here we request location of **`ictlab`** from its agent:

```
$ snmpget -v 2c -c public ictlab
  SNMPv2-MIB::sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING:
  "Hong Kong, IVE(TY)/ICT"
```

Example: getting location 2



manager

SNMPv2c public get .1.3.6.1.2.1.1.6.0

SNMPv2c public response .1.3.6.1.2.1.1.6.0 Hong Kong ICT(TY)

agent

ictlab

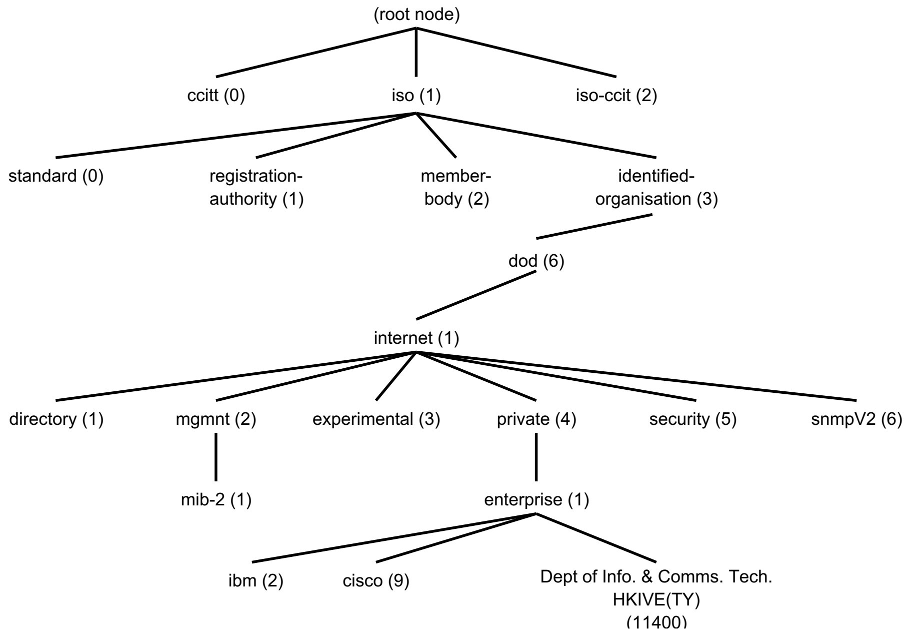
.1.3.6.1.2.1.1.6.0 is SNMPv2-MIB::sysLocation.0

Structure of Management Information (SMI)

- Defines how managed objects are named, and specifies their datatypes (called **syntax**).
- Definition has three attributes:
  - Name (also called object identifier). Two forms (both very long):
    - Numeric
    - “Human readable”
  - Type and syntax: defined using a subset of **ASN.1** (**Abstract Syntax Notation One**)
    - ASN.1 is machine independent
  - Encoding:
    - how an instance of a managed object is encoded as a string of bytes using the **Basic Encoding Rules** (BER)

Naming managed objects

- Objects are organised into a tree
- Object ID is series of numbers separated by dots
- “human readable” name substitutes a name for each number
  - But the names are very long and hard for a human to remember
- NMS makes it easier to find variables (objects) in a more human friendly way



ASN.1

- MIBs defined with a **`SYNTAX`** attribute
- The **`SYNTAX`** specifies a **datatype**, as in a programming language
- Exact specification, so works on any platform
- Will see examples of MIB definitions later

ASN.1 Basic data types

- **`INTEGER`**: length can be specified
- **`OCTET STRING`**: byte string
- **`OBJECT IDENTIFIER`**: 1.3.6.1.4.1.11400 is ICT private enterprise OID.

SNMPv1 data types

- **`Counter`**: 32-bit unsigned value that wraps
- **`IpAddress`**: 32-bit IPv4 address
- **`NetworkAddress`**: can hold other types of addresses
- **`Gauge`**: 32-bit unsigned value that can increase or decrease but not wrap
- **`TimeTicks`**: 32-bit count in hundredths of a second
- **`Opaque`**: allow any kind of data

SNMPv2 data types

- **`Integer32`**: a 32-bit signed integer
- **`Counter32`**: same as **`Counter`**
- **`Gauge32`**: Same as **`Gauge`**
- **`Unsigned32`**: 32-bit unsigned value
- **`Counter64`**: Same as **`Counter32`**, except uses 64 bits, a useful extension to cope with high-speed networks which can wrap a 32-bit counter in a short time
- **`BITS`**: a set of named bits

Protocol Data Unit (PDU)

- The **PDU** is the message format that carries SNMP operations.
- There is a standard PDU for each of the SNMP operations.

Message Format: message header

message header	**PDU**

- SNMPv1, v2c message has a header and PDU
- header contains:
  - version number (version of SNMP)
  - Community name (i.e., the shared password)

Message Format: the PDU

PDU type	Request ID	Error Status	Error Index	Object 1 Value 1	Object 2 Value 2	. . .	Object n Value n

variable bindings

- **get**, **get-next**, **response**, **set** PDUs all contain same fields
- PDU type indicated operation (i.e., **get**, or **set**)
- request ID associates request with response
- Error status, index: show an error condition
  - used in response only
- Variable Bindings: object ID and value.
  - SNMP allows more than one OID/value pair to be sent together for efficiency

SNMP Operations

SNMPv1

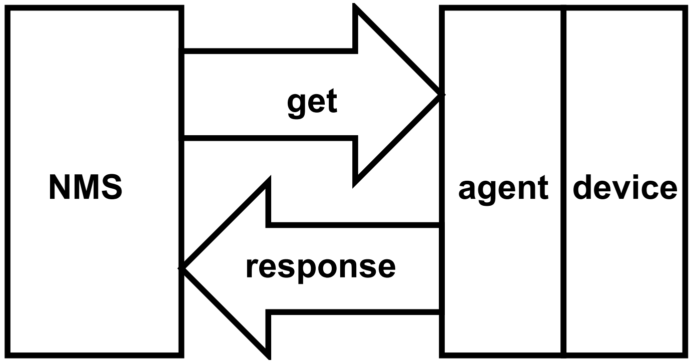
- **get-request**
- **get-next-request**
- **set-request**
- **get-response**
- **trap**

SNMPv2, v3

- **get-bulk-request**
- **Notification** (actually just a macro for **trap** or **inform-request**)
- **inform-request**
- **report**

get-request operation

- Net SNMP tool: **snmpget**



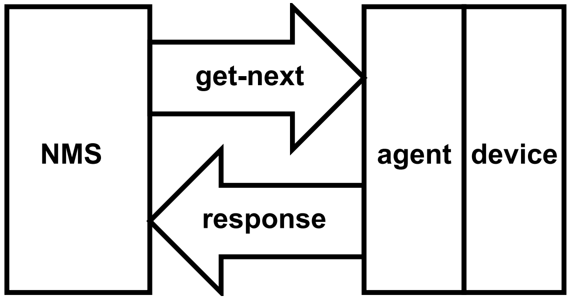
get-request

- NMS sends a **get-request** for, say, the system load of **ictlab**
- The agent on **ictlab** sends a **response** PDU containing the system load.

```
snmpget -v 2c -c public ictlab UCD-
  SNMP-MIB::laLoad.1
UCD-SNMP-MIB::laLoad.1 = STRING:
  0.39
```

`get-next-request` operation

- Net-SNMP tools:
- `snmpgetnext`
- `snmpwalk`



`get-next-request`

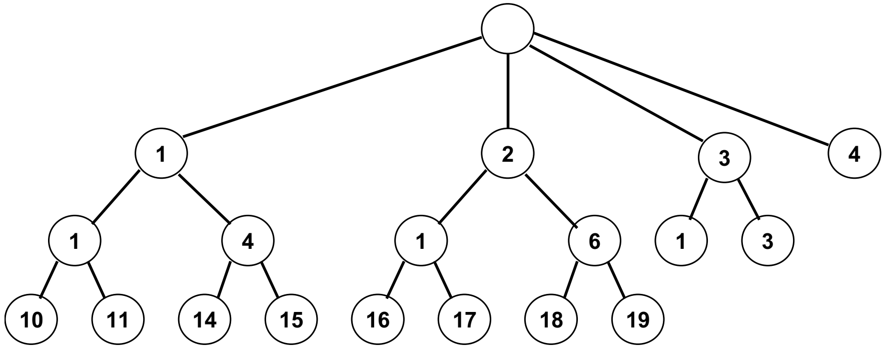
- NMS sends a **`get-next-request`**
- Agent sends a **`response`** PDU containing the value for the **next** variable:

```
$ snmpgetnext -v 2c -c public ictlab laLoad
UCD-SNMP-MIB::laLoad.1 = STRING: 0.74
```

Ordering of OIDs: the next value

- The ordering of the variables is "lexical"
  - visit the node, then visit each of its children in order
  - this applies recursively
- The example MIB tree on the next slide…

An example MIB tree



This example MIB tree is listed in this order:

- 1
- 1.1
- 1.1.10
- 1.1.11
- 1.4
- 1.4.14
- 1.4.15
- 2
- 2.1
- 2.1.16
- 2.1.17
- 2.6
- 2.6.18
- 2.6.19
- 3
- 3.1
- 3.3
- 4

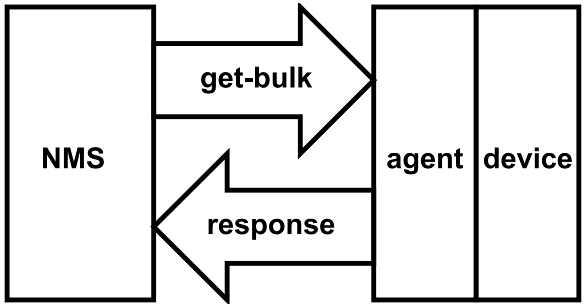
get-next-request: snmpwalk

- **snmpwalk** provides a convenient way to request a number of entries at once:

```
$ snmpwalk -v 2c -c public ictlab laLoad
UCD-SNMP-MIB::laLoad.1 = STRING: 0.74
UCD-SNMP-MIB::laLoad.2 = STRING: 0.53
UCD-SNMP-MIB::laLoad.3 = STRING: 0.48
```

get-bulk-request (v2, v3)

- Net-SNMP tools: **snmpbulkget, snmpbulkwalk**



get-bulk-request

- NMS sends a **get-bulk-request** for a number of variables
- Agent replies with a **response** PDU with as many answers as are requested, or will fit in the PDU
- Much more **efficient**
  - fewer requests and responses required to fetch data

`get-bulk-request` and `snmpbulkget`: example

```
$ snmpbulkget -v 2c -c public ictlab laLoad
UCD-SNMP-MIB::laLoad.1 = STRING: 0.62
UCD-SNMP-MIB::laLoad.2 = STRING: 0.66
UCD-SNMP-MIB::laLoad.3 = STRING: 0.59
UCD-SNMP-MIB::laConfig.1 = STRING: 2.00
UCD-SNMP-MIB::laConfig.2 = STRING: 4.00
UCD-SNMP-MIB::laConfig.3 = STRING: 4.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 61
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 66
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 58
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.620000
```

`get-bulk-request`

- Get can request more than one MIB object
  - But if agent cannot send it **all** back, sends error message and no data
- **`get-bulk-request`** tells agent to send as much of the response back as it can
- Possible to send incomplete data
- Requires two parameters:
  - Nonrepeaters
  - Max-repetitions

`get-bulk-request`: nonrepeaters, max-repetitions: 1

- Nonrepeaters:
  - A number, **N**
  - Indicates first **N** objects can be retrieved with simple get-next operation
- Max-repetitions:
  - A number, R
  - Can attempt up to R get-next operations to retrieve remaining objects

`get-bulk-request`: nonrepeaters, max-repetitions: 2

```
$ snmpbulkget -v 2c -C n2r3 -c public ictlab laLoad ifInOctets
   ifOutOctets
UCD-SNMP-MIB::laLoad.1 = STRING: 0.63
IF-MIB::ifInOctets.1 = Counter32: 35352440
IF-MIB::ifOutOctets.1 = Counter32: 35352440
IF-MIB::ifOutOctets.2 = Counter32: 297960502
IF-MIB::ifOutOctets.3 = Counter32: 0
```

- Notice that we have one entry only for **`laLoad`**, and for **ifInOctets**
  - the first two variables are "non-repeaters", i.e., we just fetch one value for each
- We get **three** values for **`ifOutOctets`**
  - we ask for three values for all remaining variables after the first two

get-bulk-request: nonrepeaters, max-repetitions: 3

```
$ snmpbulkget -v 2c -C n1r3 -c public ictlab laLoad
    ifInOctets ifOutOctets
UCD-SNMP-MIB::laLoad.1 = STRING: 0.77
IF-MIB::ifInOctets.1 = Counter32: 5356045
IF-MIB::ifOutOctets.1 = Counter32: 5356045
IF-MIB::ifInOctets.2 = Counter32: 1881446668
IF-MIB::ifOutOctets.2 = Counter32: 3664336845
IF-MIB::ifInOctets.3 = Counter32: 0
IF-MIB::ifOutOctets.3 = Counter32: 0
```

- We have one value for the **first** variable `laLoad` (non-repeaters = 1)
- We have 3 values for all the remaining variables we ask for

get-bulk-request: nonrepeaters, max-repetitions: 4

```
$ snmpbulkget -v 2c -C n3r3 -c public ictlab laLoad
    ifInOctets ifOutOctets
UCD-SNMP-MIB::laLoad.1 = STRING: 0.71
IF-MIB::ifInOctets.1 = Counter32: 35370916
IF-MIB::ifOutOctets.1 = Counter32: 35370916
```

- Notice we only have **one** entry for all **three** OIDs we specified on the command line.
- Same result, regardless of value of **R**, I.e., `snmpbulkget -v 2c -C n3r0 ...` gives the same result.

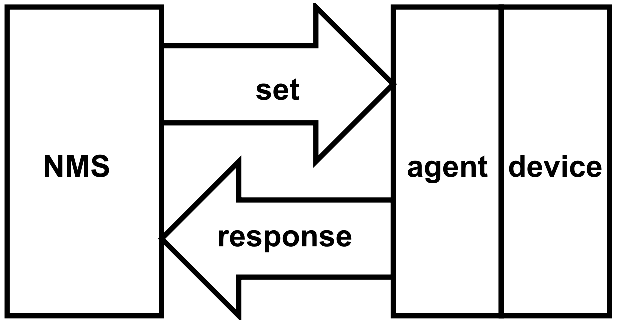
get-bulk-request: snmpbulkwalk

- **snmpbulkwalk** is convenient for **efficiently** browsing large tables in the MIB tree

```
$ snmpbulkwalk -v 2c -c public ictlab laLoad
UCD-SNMP-MIB::laLoad.1 = STRING: 0.52
UCD-SNMP-MIB::laLoad.2 = STRING: 0.58
UCD-SNMP-MIB::laLoad.3 = STRING: 0.56
```

set-request operation

- Net-SNMP tool: **snmpset**

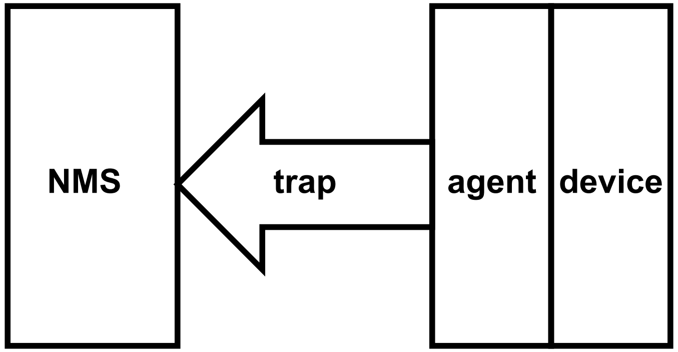


`set`

- NMS sends a **`set-request`** to set **`sysLocation`** to ICT Laboratory, Hong Kong
- Agent replies with either an error response, or a noError response in a **`request`** PDU

Trap

- A trap has no response:



SNMP traps

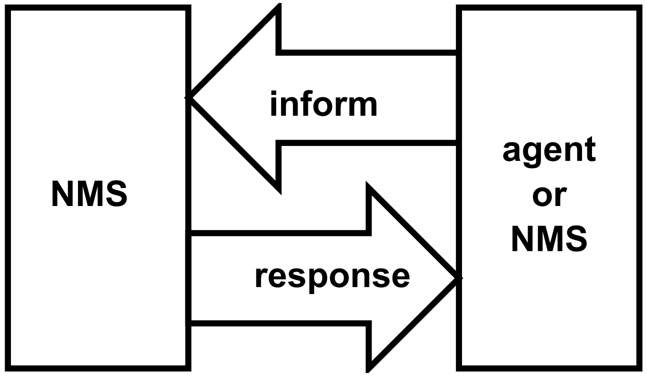
- Lets the **agent tell the manager** something happened, e.g.,
  - A network interface is done on the device where the agent is installed
  - The network interface came back up
  - A call came in to the modem rack, but could not connect to any modem
  - A fan has failed

SNMP **`inform-request`** (v2, v3)

- A kind of trap with an acknowledgment
- Can be sent by a manager or by an agent
- There is an acknowledgement: a **`response`** PDU
- The agent can resend the **`inform-request`** if no response is received in a reasonable time.

`inform-request`

- An `inform-request` has a confirmation response:



SNMP `Notification` (v2, v3)

- This is a macro that sends either a trap or an inform-request

Traps and Inform: port 162

- Other SNMP operations are on UDP port 161
- trap and inform-request operations are on UDP port 162.

SNMP v3

Authentication and Encryption
Some security at last!

SNMPv1 now officially "historic"

- Recently, SNMPv3 has moved futher to becoming an official standard
- SNMPv1 RFCs are being changed from the status of **standard** to being **historic**
- for details:
  - see news link from Net-SNMP web site
  - or go directly to http://sourceforge.net/forum/forum.php?forum_id=203052

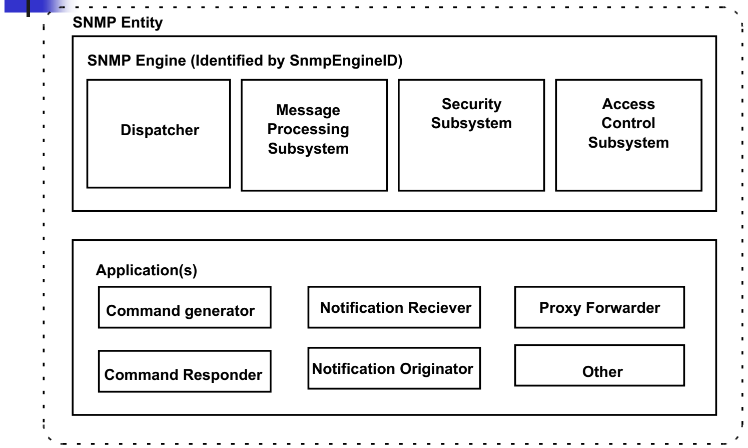
Main RFCs for SNMP v3

- RFC 2571: an **architecture** for describing SNMP Management Frameworks
- RFC 2572: Message Processing and Dispatch for SNMP
- RFC 2573: SNMPv3 Applications MIBs
- RFC 2574: User-based Security Model (**USM**) for SNMPv3
- RFC 2575: View-based Access Control Model (**VACM**) for SNMP

Changes in SNMPv3

- Aim: provide cryptographic security
- Make backwardly compatible with SNMPv1, SNMPv2c
- Many new terms
- Most importantly:
  - now abandon notion of managers and agents
  - both managers and agents now called SNMP **entities**
- SNMPv3 defines an **architecture**
  - not just a set of messages

SNMPv3 architecture (RFC 2571)



SNMP Entity

SNMP Engine (Identified by SnmpEngineID)

Dispatcher

Message Processing Subsystem

Security Subsystem

Access Control Subsystem

Application(s)

Command generator

Notification Reciever

Proxy Forwarder

Command Responder

Notification Originator

Other

SNMP Engine: 5 components

- Dispatcher
  - send and receive messages.
  - determines version of each received message (v1, v2, v3)
  - if can handle received message, hands to Message Processing Subsystem
- Message Processing Subsystem
  - prepares messages to be sent
  - extracts data from received messages
  - can have modules for each of SNMP v1, v2 and v3 (or any other future type of message)
- Security Subsystem
  - provides authentication and encryption ("privacy")
  - Uses MD5 or SHA algorithms to authenticate users
  - passwords not sent in clear text
- Access Control Subsystem
  - controls access to MIB objects
  - which objects, and level of access
- Applications module (discussed next)

SNMPv3 Applications Module

- Each SNMPv3 entity has one or more **applications**
- Really are elements used to build applications:
- command generator (NMS)
- notification receiver (NMS)
- proxy forwarder (NMS)
- command responder (agent)
- notification originator (agent)

Command Generator: manager role

- This application is found on managers
- used to send
  - **`get-request`**
  - **`set-next-request`**
  - **`set-request`**
  - **`get-bulk-request`**

Command Responder: agent role

- processes commands sent by Command Generator
- performs the action required
- sends a **`response`** message

Notification Originator: agent role

- Generates a **`trap`** or **`inform-request`** message
- generally implemented on agents

Notification Receiver: manager

- receives **`traps`** and **`inform-requests`**, and
- acts on them

Proxy Forwarder: manager role

- A front end to manager for older SNMP agents
- e.g., convert **`get-bulk-request`** to **`get-next-requests`**
- handles requests from:
  - command generator
  - command responder
  - notification generator.

SNMPv3 names: Engine ID

- A manager or agent has an **engineID**
- The engineID is usually based on **IP address** of the device or manager

SNMPv3 names: context

- An entity can be responsible for more than one managed device.
  - e.g., for some interfaces, a switch may have a different **context**; they are managed separately, although there is one agent (entity) on the switch
- Each managed device has a **contextEngineID** and a **contextName**
- normally contextEngineID = snmpEngineID

SNMPv3 MIBs

- New MIBs for SNMPv3 support
  - management architecture
  - authentication and encryption
- Location: under snmpv2 (.1.3.6.1.6) in snmpModules (.1.3.6.1.6.3)

SNMPv3 User-based Security Model (USM)

- Supports authentication using
  - MD5 (Message Digest 5) or
  - SHA1 (Secure Hash Algorithm)
- Supports encryption using DES (Data Encryption Standard)
- Supports individual user accounts

SNMPv3 Access Control: VACM

- Uses the **View-based Access Control Model** (**VACM**)
- Has 5 elements:
  - groups
  - security level
  - contexts
  - MIB views and view families
  - access policy

VACM: MIB views and view families

- A MIB view is part of the MIB tree
- can be a **subtree** (i.e., SNMPv2-MIB::system and below)
- Can be a set of trees
- Can be a **family of view subtrees**:
  - e.g., monitor a set of columns from a table, but not all the columns
  - useful for ISPs to allow customers to monitor input, output traffic

VACM: groups

- Basically, a set of one or more users
- All elements belonging to a group have equal access rights
- Can make a group that is compatible with SNMPv1 community, so the name of the group is the community string.

VACM: security level

- There are three levels:
  - no authentication, no privacy
  - authentication, no privacy
  - authentication, privacy
- **privacy** means encryption using DES
- **authentication** requires a password hashed with MD5 or SHA1.

VACM: Access Policy

- Four levels:
  - not accessible
  - read view
  - write view
  - notify view