SNMP and Network Management

Simple Network Management Protocol

A Standard Protocol for Systems and Network Management
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
Network Management - Basic paradigm

Manager

Client

Req

Server

Res

Agent

MIB

Events/Alarm

Managed Node

Management Station
SNMP runs on UDP

- UDP = User Datagram Protocol
- Unreliable (no acknowledgment 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
Management Models in SNMP

- Organizational Model
  - Two-tier architecture
  - Three-tier architecture (RMON)
  - Proxy Servers
  - SNMP network management architecture
  - Containment model (Rack->Shelf->Slot->Card->Port->Timeslots)
What is a *managed object*?

- A Managed Object composed of object type and an object instance.
- A better name is *variable*, but called *managed object* more often.
- Ex. the managed object system.sysUpTime.0
  - 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)
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)**
More on managed object?

- SMI is concerned only with the object type and not object instance
  - E.g. Object type with different instances of hub card would be the identical object ID (iso.org.dod.internet.private.enterprise.43.1.8.5) but different ip addresses for each instance
Naming managed objects

- Objects are organised into a tree
- Object type (i.e. name) is uniquely identified by a DESCRIPTOR and an associated OBJECT IDENTIFIER
- DESCRIPTOR defining the name is mnemonic and is in all lower case
- Object ID is unique name and number in the MIT
- 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
Naming managed objects

- internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }  
- internet OBJECT IDENTIFIER ::= {1 3 6 1}  
- directory OBJECT IDENTIFIER ::= { internet 1 }  
- mgmt OBJECT IDENTIFIER ::= { internet 2 }  
- experimental OBJECT IDENTIFIER ::= { internet 3 }  
- private OBJECT IDENTIFIER ::= { internet 4 }
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 or Primitive Date 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 Defined data types

- **Counter**: 32-bit unsigned value that wraps after $2^{32} - 1$. It can only increase.
- **IpAddress**: 32-bit IPv4 address
- **NetworkAddress**: can hold other types of protocol family
- **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
**SEQUENCE (ipAddrEntry)**

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Object ID</th>
<th>Obj Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ipAdEntAddr</td>
<td>{ipAdEntry 1}</td>
<td>ipAddress</td>
</tr>
<tr>
<td>2 ipAdEntIfIndex</td>
<td>{ipAdEntry 2}</td>
<td>INTEGER</td>
</tr>
<tr>
<td>3 ipAdEntNetMask</td>
<td>{ipAdEntry 3}</td>
<td>ipAddress</td>
</tr>
<tr>
<td>4 ipAdEntBcdAddr</td>
<td>{ipAdEntry 4}</td>
<td>INTEGER</td>
</tr>
<tr>
<td>5 ipAdEntReaMaxSize</td>
<td>{ipAdEntry 5}</td>
<td>INTEGER</td>
</tr>
<tr>
<td>6 ipAddrEntry</td>
<td>{ipAddrTable 1}</td>
<td>SEQUENCE</td>
</tr>
</tbody>
</table>
Definition of ipAddrEntry

List: IpAddrEntry ::= SEQUENCE {
  ipAdEntAddr             IpAddress
  ipAdEntIfIndex           INTEGER

}
SEQUENCE OF (Table)

Table: IpAddrTable

\[ \text{ipAddrTable} \]

\[ \text{SEQUENCE OF} \]

IpAddrEntry

<table>
<thead>
<tr>
<th>7</th>
<th>IpAddrTable</th>
<th>{ip 20}</th>
<th>SEQUENCE OF</th>
</tr>
</thead>
</table>
MIB – Management Information Base

- **Object IDentifier (OID)**
  - Example .1.3.6.1.2.1.1
  - iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1)

**Note:**
- .1.3.6.1 ~100% present.
- mgmt and private most common.
- MIB-2 successor to original MIB.
- STATUS ‘mandatory’. All or nothing in group
- Contains objects that describe some basic information on an entity.
- An entity can be the agent itself or the network object that the agent is on.

**system(1) group objects**
- `sysDescr(1)` → Description of the entity.
- `sysObjectID(2)` → Vendor defined OID string.
- `sysUpTime(3)` → Time since net-mgt was last re-initialised.
- `sysContact(4)` → Name of person responsible for the entity.
sysUpTime OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
“The time (in hundredths of a second) since the network management portion of the system was last re-initialized.”

::= {system 3}
MIB – Management Information Base

MIB - tree view

MIB - syntax view

sysDescr OBJECT-TYPE
SYNTAX
OCTETSTRING (SIZE (0…255))
ACCESS read-only
STATUS mandatory
DESCRIPTION
“This value include the full name and version id of the system hardware, OS and NOS.”
 ::= {system 1}
OBJECT-TYPE MACRO

OBJECT-TYPE MACRO ::= 

BEGIN

TYPE NOTATION ::= “SYNTAX” type (TYPE ObjectSyntax)
  “ACCESS” Access
  “STATUS” Status

VALUE NOTATION ::= value (VALUE ObjectName)

Access ::= “read-only”|“write-only”|“not-accessible”

Status ::= “mandatory”|“optional”|“obsolete”

END
Aggregate Objects

- Group of related objects e.g. ifTable & ipAddrTable
- Objects which make up Aggregate Objects are called subordinate object type e.g. ipAddrEntry
- Columnar Objects
Columnar ipAddrTable

- ipAddrTable {1.3.6.1.2.1.4.20}
  - ipAddrEntry {1}
    - ipAdEntAddr {1}
    - ipAdEntIfIndex {2}
    - ------
  - Columnar object ID of ipAdEntIfIndex would be \{1. 3.6.1.2.1.4.20.1.2\}
  - Iso.org.dod.internet.mgmt.mib-2.ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex
### ipAddrTable

<table>
<thead>
<tr>
<th>Row</th>
<th>ipAdEnt Addr</th>
<th>ipAdEnt IfIndex</th>
<th>ipAdEnt NetMask</th>
<th>ipEntBroadcastAddr</th>
<th>ipAdEnt DaGram MaxSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123.45.2.1</td>
<td>1</td>
<td>255.255.255.0</td>
<td>0</td>
<td>12000</td>
</tr>
<tr>
<td>2</td>
<td>123.45.2.4</td>
<td>3</td>
<td>255.255.0.0</td>
<td>1</td>
<td>12000</td>
</tr>
<tr>
<td>3</td>
<td>165.8.9.25</td>
<td>4</td>
<td>255.255.255.0</td>
<td>0</td>
<td>12000</td>
</tr>
<tr>
<td>4</td>
<td>9.96.8.138</td>
<td>2</td>
<td>255.255.255.0</td>
<td>0</td>
<td>12000</td>
</tr>
</tbody>
</table>
SNMP Protocol Specification

- Protocol Entities
- Application Layer -> Physical Layer
  - Get-Request [0]
  - Get-Next-Request [1]
  - Set-Request [2]
  - Get-Response [3]
  - Trap [4]
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

- 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

- **get**, **get-next**, **response**, **set** PDUs all contain the 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, zero otherwise
- **Variable Bindings**: object ID and value.
  - SNMP allows more than one OID/value pair to be sent together for efficiency
**get-bulk-request PDU**

<table>
<thead>
<tr>
<th>PDU type</th>
<th>Request ID</th>
<th>Non-repeaters</th>
<th>Max-repetitions</th>
<th>Object 1 Value 1</th>
<th>Object 2 Value 2</th>
<th>...</th>
<th>Object n Value n</th>
</tr>
</thead>
</table>

- All fields same as other SNMP PDUs in v1, v2c, except *Nonrepeaters* and *Max-repetitions*

- **Nonrepeaters**: Specifies the number of object instances in the variable bindings field that should be retrieved no more than once from the beginning of the request.
  - used when some of the instances are scalar objects with only one variable.

- **Max-repetitions**: Defines the maximum number of times that other variables beyond those specified by the non-repeaters field should be retrieved.
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
Trap

- A trap has no response:
SNMP traps

- Lets the *agent tell the manager* something happened, e.g.,
  - A network interface is down 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
Trap Format: the PDU(V1)

- **Generic Trap Types**
  - `coldStart(0)`
  - `warmStart(1)`
  - `linkDown(2)`
  - `linkUp(3)`
  - `authenticationFailure(4)`
  - `egpNeighborLoss(5)`
  - `enterpriseSpecific(6)`
Trap Format: the PDU(V2)

- **linkUp NOTIFICATION-TYPE**
  OBJECT {ifIndex}
  STATUS current
  DESCRIPTION “...”

<table>
<thead>
<tr>
<th>PDU TYPE</th>
<th>REQID</th>
<th>Error Status</th>
<th>Error Index</th>
<th>SystemUpTime</th>
<th>SNMPTrapoid</th>
<th>VARBIND 1</th>
<th>VARBIND N</th>
</tr>
</thead>
</table>

Systems and Network Management

SNMP

39
Traps and Inform: port 162

- Other SNMP operations are on UDP port 161
- Trap and inform-request operations are on UDP port 162.
SNMP notification (v2, v3)

- This is a macro that sends either a trap or an inform-request
- Bilingual Manager
- SNMP Proxy Server
Table Augmentation (SNMPV2)

- Aggregate Objects – Single to Mutliple
- Expansion by adding more columnar objects – When (1)
  - Number of conceptual rows not affected
  - One-to-One correspondence b/w the rows of the two tables
  - INDEX of second table – same as first.
Table Augmentation – ASN.1 construct

- **table1Entry** OBJECT-TYPE  
  - SYNTAX TableT1Entry  
  - MAX-ACCESS not-accessible  
  - STATUS current  
  - DESCRIPTION “An entry in table T1”  
  - INDEX {T1.E1.C1}  
    - ::= (table 1)

- **table2Entry** OBJECT-TYPE  
  - SYNTAX TableT2Entry  
  - MAX-ACCESS not-accessible  
  - STATUS current  
  - DESCRIPTION “An entry in table T2”  
  - AUGMENTS {tableEntry}  
    - ::= (table2 1)
Table Augmentation (SNMPv2)

- Dense Table
  - Large no of rows than the base table
  - Combined indices of both
  - INDEX CLAUSE – combined columnar objects as the index clause for the aggregate object.
- Base Table & Dependent Table
Table Augmentation Dense Table – ASN.1 construct

- table1Entry OBJECT-TYPE Base Table
  - SYNTAX TableT1Entry
  - MAX-ACCESS not-accessible
  - STATUS current
  - DESCRIPTION “An entry in table T1”
  - INDEX {T1.E1.C1}
    - ::= (table 1)

- table2Entry OBJECT-TYPE Augmented Table
  - SYNTAX TableT2Entry
  - MAX-ACCESS not-accessible
  - STATUS current
  - DESCRIPTION “An entry in table T2”
  - INDEX {T1.E1.C1, T2.E2.C4}
    - ::= (table2 1)
Table Augmentation (SNMPv2)

- Sparse Table
  - Less no of rows than the base table.
  - INDEX of second table – same as first.
  - AUGMENT CLAUSE – Substitute with INDEX CLAUSE
Creation/Deletion of rows (SNMPV2)

- Significant features of V2
- Creation of rows
  - Create a row & make it active
  - Create a row & make it active later.
  - A new column – status column added to the tables.
## RowStatus

<table>
<thead>
<tr>
<th>State</th>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>1</td>
<td>Row exist &amp; operational</td>
</tr>
<tr>
<td>notInService</td>
<td>2</td>
<td>Operation on the row is suspended</td>
</tr>
<tr>
<td>notReady</td>
<td>3</td>
<td>Row doesn’t have all the columnar objects</td>
</tr>
<tr>
<td>createAndGo</td>
<td>4</td>
<td>One-step creation and become active</td>
</tr>
<tr>
<td>createAndWait</td>
<td>5</td>
<td>Row under creation; not to be commissioned</td>
</tr>
<tr>
<td>Destroy</td>
<td>6</td>
<td>Row deleted.</td>
</tr>
</tbody>
</table>