Simple Network Management Protocol (SNMP)

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Overview

- Network Management
- SNMP
- Management information base (MIB)
- ASN.1 Notation
- RMON
- Ref: Chap 25, Stallings: “SNMP, SNMPv2 and RMON”, Addison Wesley
Network Management

- Management = Init, Monitoring, Control
  - Today: automated, reliable diagnosis, and automatic control are still in a primitive stage
- Architecture: Manager, Agents, and Management Information Base (MIB)

SNMP history

- Early: based upon ICMP messages (eg: ping, source routing, record routing)
- A lot of informal network debugging is done using tcpdump, netstat, ifconfig etc
- When the internet grew, Simple Gateway Management Protocol (SGMP) was developed (1987)
- Build single protocol to manage OSI and IP
  - CMIP (an OSI protocol) over TCP/IP {called CMOT}
  - Goal: Keep object level same for both OSI and IP
  - CMOT progressed very sluggishly
  - SNMP: parallel effort. Very simple => grabbed the market.
SNMP

- Based on SGMP
- Simple: only five commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>get-request</td>
<td>Fetch a value</td>
</tr>
<tr>
<td>get-next-request</td>
<td>Fetch the next value</td>
</tr>
<tr>
<td>get-response</td>
<td>Reply to a fetch operation</td>
</tr>
<tr>
<td>set-request</td>
<td>Set (store) a value</td>
</tr>
<tr>
<td>trap</td>
<td>Agent notifies manager</td>
</tr>
</tbody>
</table>

Simple: handles only scalars. “get-next-request” used successively to get array values etc.

SNMP contd

- Simple: one management station can handle hundreds of agents
- Simple: Works as an application protocol running over UDP
- Agent and manager apps work on top of SNMP
- Proxy-SNMP can be used to manage a variety of devices (serial lines, bridges, modems etc).
  - Proxy (similar to bridge) is needed because these devices may not run UDP/IP
  - For each new device define a new MIB.
Management Information Base (MIB)

- Specifies what variables the agents maintain
- Only a limited number of data types are used to define these variables
- MIBs follow a fixed naming and structuring convention called “Structure of Management Information” (SMI). See next slide.
- Variables are identified by “object identifiers”
  - Hierarchical naming scheme (a long string of numbers like 1.3.6.1.2.1.4.3 which is assigned by a standards authority)
  - Eg: iso.org.dod.internet.mgmt.mib.ip.ipInReceives 1.3.6.1.2.1.4.3

Global Naming Hierarchy

- ccitt(0)
- iso (1)
- joint-iso-ccitt (2)
- standard (0)
- org (3)
- dod (6)
- internet (1)
- directory (1)
- mgmt(2)
- experimental (3)
- private (4)
- system (1)
- interfaces (2)
- transmission(10)
- fddimib (73)
- fddi (8)
- ISO9314 (9314)
- fddiMIB (1)
- IS0 (3)
- joint-ISO-CCITT (2)
- standard (0)
- private (4)
- experimental (3)
- mgmt (2)
- internet (1)
MIB (contd)

- All names are specified using a subset of Abstract Syntax Notation (ASN.1)
- **Types**: INTEGER, OCTET STRING, OBJECT IDENTIFIER, NULL
- **Constructors**: SEQUENCE (like struct in C), SEQUENCE OF (table i.e. vector of structs), CHOICE (one of many choices)
- ASN.1 provides more types and constructors, but they are not used to define MIBs.

Standard MIBs

- New device => write MIB for it and include it as a branch of MIB-II
- MIB-II (RFC 1213) a superset of MIB-I (RFC 1156)
- Contains only essential objects
- Only “weak” objects. Tampering => limited damage
- No limit on number of objects (unlike MIB-I)
- Avoid redundant objects, and implementation-specific objects.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Category</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysUpTime</td>
<td>system</td>
<td>Time since last reboot</td>
</tr>
<tr>
<td>ifNumber</td>
<td>interfaces</td>
<td># of Interfaces</td>
</tr>
<tr>
<td>ifMTU</td>
<td>interfaces</td>
<td>MTU</td>
</tr>
<tr>
<td>ipDefaultTTL</td>
<td>ip</td>
<td>Default TTL</td>
</tr>
<tr>
<td>ipInReceives</td>
<td>ip</td>
<td># of datagrams received</td>
</tr>
<tr>
<td>ipForwDatagrams</td>
<td>ip</td>
<td># of datagrams forwarded</td>
</tr>
<tr>
<td>icmpInEchos</td>
<td>icmp</td>
<td># of Echo requests received</td>
</tr>
<tr>
<td>tcpRtoMin</td>
<td>tcp</td>
<td>Min retrans time</td>
</tr>
<tr>
<td>tcpMaxConn</td>
<td>tcp</td>
<td>Max connections allowed</td>
</tr>
</tbody>
</table>

**Instance Identification**

- How does the manager refer to a variable?
  - **Simple variables**: append “.0” to variable’s object identifier
    - Eg: udpInDatagrams.0 = 1.3.6.1.2.1.7.1.0
    - Only leaf nodes can be referred (since SNMP can only transfer scalars)
  - **Table elements**:  
    - Each element in a table needs to be fetched separately.  
    - Traverse MIB based upon lexicographic ordering of object identifiers using get-next  
    - Column-by-column: Elements of each column first.
RMON

- Remote Network Monitoring
- Defines remote monitoring MIB that supplements MIB-II and is a step towards internetwork management
- It extends SNMP functionality though it is simply a specification of a MIB
- Problem w/ MIB-II
  - Can obtain info that is purely local to individual devices
  - Cannot easily learn about LAN traffic as a whole (eg like LANanalyzers or “remote monitors”)

RMON (contd)

- Functionality added: Promiscuously count, filter and store packets
- System that implements RMON MIB is called an RMON probe (or less frequently, an RMON agent).
  - No changes to SNMP protocol.
  - Enhance the manager and agents only.
- RMON MIB organization:
  - Control table: read-write. Configures what parameters should be logged and how often.
  - Data table: read-only (statistics etc logged)
- Other issues: shared probes, ownership of tables, concurrent table access...
Summary

- Management = Initialization, Monitoring, and Control
- SNMP = Only 5 commands
- Standard MIBs defined for each object
- Uses ASN.1 encoding
- RMON extends SNMP functionality through definition of a new MIB