What is network management?

- autonomous systems (aka “network”): 100s or 1000s of interacting hw/sw components
- other complex systems requiring monitoring, control:
  - jet airplane
  - nuclear power plant
  - others?

"Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

Infrastructure for network management

definitions:

managed devices contain managed objects whose data is gathered into a Management Information Base (MIB)
Network Management standards

- **OSI CMIP**
  - Common Management Information Protocol
  - Designed 1980’s: the unifying net management standard
  - Too slowly standardized

- **SNMP: Simple Network Management Protocol**
  - Internet roots
  - Started simple
  - Deployed, adopted rapidly
  - Growth: size, complexity
  - Currently: SNMP V3
  - De facto network management standard

SNMP overview: 4 key parts

- **Management information base (MIB):**
  - Distributed information store of network management data

- **Structure of Management Information (SMI):**
  - Data definition language for MIB objects

- **SNMP protocol**
  - Convey manager<->managed object info, commands

- **Security, administration capabilities**
  - Major addition in SNMPv3
SMI: data definition language

**Purpose**: syntax, semantics of management data well-defined, unambiguous
- base data types:
  - straightforward, boring
- **OBJECT-TYPE**
  - data type, status, semantics of managed object
- **MODULE-IDENTITY**
  - groups related objects into MIB module

**Basic Data Types**

- INTEGER
- Integer32
- Unsigned32
- OCTET STRING
- OBJECT IDENTIFIED
  - IPaddress
  - Counter32
  - Counter64
  - Gauge32
  - TimeTicks
  - Opaque

SNMP MIB

MIB module specified via SMI

**MODULE-IDENTITY**

(100 standardized MIBs, more vendor-specific)

objects specified via SMI

**OBJECT-TYPE** construct
**SMI: Object, module examples**

**OBJECT-TYPE:** ipInDelivers

- **ipInDelivers** OBJECT TYPE
  - **SYNTAX** Counter32
  - **MAX-ACCESS** read-only
  - **STATUS** current
  - **DESCRIPTION**
    - "The total number of input datagrams successfully delivered to IP user-protocols (including ICMP)"

```plaintext
::= {ip 9}
```

**MODULE-IDENTITY:** ipMIB

- **ipMIB** MODULE-IDENTITY
  - **LAST-UPDATED** "941101000Z"
  - **ORGANIZATION** "IETF SNPv2 Working Group"
  - **CONTACT-INFO** "Keith McCloghrie"
  - **DESCRIPTION**
    - "The MIB module for managing IP and ICMP implementations, but excluding their management of IP routes."
  - **REVISION** "019331000Z"

```plaintext
::= {mib-2 48}
```

---

**MIB example: UDP module**

<table>
<thead>
<tr>
<th>Object ID</th>
<th>Name</th>
<th>Type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.6.1.2.1.7.1</td>
<td>UDPInDatagrams</td>
<td>Counter32</td>
<td>total # datagrams delivered at this node</td>
</tr>
<tr>
<td>1.3.6.1.2.1.7.2</td>
<td>UDPNoPorts</td>
<td>Counter32</td>
<td># undeliverable datagrams no app at portl</td>
</tr>
<tr>
<td>1.3.6.1.2.1.7.3</td>
<td>UDInErrors</td>
<td>Counter32</td>
<td># undeliverable datagrams all other reasons</td>
</tr>
<tr>
<td>1.3.6.1.2.1.7.4</td>
<td>UDPOutDatagrams</td>
<td>Counter32</td>
<td># datagrams sent</td>
</tr>
<tr>
<td>1.3.6.1.2.1.7.5</td>
<td>udpTable</td>
<td>SEQUENCE</td>
<td>one entry for each port in use by app, gives port # and IP address</td>
</tr>
</tbody>
</table>
SNMP Naming

**question:** how to name every possible standard object (protocol, data, more..) in every possible network standard??

**answer:** ISO Object Identifier tree:
- hierarchical naming of all objects
- each branchpoint has name, number

```
1.3.6.1.2.1.7.1
```

ISO
US DoD
Internet
udpInDatagrams
UDP
MIB2
management

Check out www.alvestrand.no/harald/objectid/top.html
**SNMP protocol**

Two ways to convey MIB info, commands:

- **request/response mode**
- **trap mode**

**SNMP protocol: message types**

<table>
<thead>
<tr>
<th>Message type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRequest</td>
<td>Mgr-to-agent: “get me data”</td>
</tr>
<tr>
<td>GetNextRequest</td>
<td>(instance, next in list, block)</td>
</tr>
<tr>
<td>GetBulkRequest</td>
<td></td>
</tr>
<tr>
<td>InformRequest</td>
<td>Mgr-to-Mgr: here’s MIB value</td>
</tr>
<tr>
<td>SetRequest</td>
<td>Mgr-to-agent: set MIB value</td>
</tr>
<tr>
<td>Response</td>
<td>Agent-to-mgr: value, response to Request</td>
</tr>
<tr>
<td>Trap</td>
<td>Agent-to-mgr: inform manager of exceptional event</td>
</tr>
</tbody>
</table>
**SNMP protocol: message formats**

<table>
<thead>
<tr>
<th>PDU Type (0–3)</th>
<th>Request ID</th>
<th>Error Status (0–5)</th>
<th>Error Index</th>
<th>Name</th>
<th>Value</th>
<th>Name</th>
<th>Value</th>
<th>...</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PDU Type (4)</th>
<th>Enterprise</th>
<th>Agent Addr</th>
<th>Trap Type (0–7)</th>
<th>Specific Code</th>
<th>Time Stamp</th>
<th>Name</th>
<th>Value</th>
<th>...</th>
</tr>
</thead>
</table>

**SNMP security and administration**

- **encryption**: DES-encrypt SNMP message
- **authentication**: compute, send MIC(m,k): compute hash (MIC) over message (m), secret shared key (k)
- **protection against playback**: use nonce
- **view-based access control**
  - SNMP entity maintains database of access rights, policies for various users
  - database itself accessible as managed object!
Exercise

- Try the command on icu0
  - $snmptranslate .1.3.6.1.2.1.7.1
  - $snmpwalk -v 1 -c public dafinn.cs.mtu.edu .1.3.6.1.2.1.7.1
  - $snmpwalk -v 1 -c public dafinn.cs.mtu.edu system
  - $snmpwalk -v 1 -c public dafinn.cs.mtu.edu interface
  - $snmpwalk -v 1 -c public dafinn.cs.mtu.edu ip
  - $snmpwalk -Tp -IR
- Visit http://dafinn.cs.mtu.edu/MRTG
- Start snmpd on your box server
- Install net-snmp from www.net-snmp.org

Firewalls

- **firewall**
  - isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others.

Two firewall types:
- packet filter
- application gateways

To prevent denial of service attacks:
- SYN flooding: attacker establishes many bogus TCP connections. Attacked host alloc's TCP buffers for bogus connections, none left for "real" connections.

To prevent illegal modification of internal data:
- e.g., attacker replaces CIA's homepage with something else

To prevent intruders from obtaining secret info.
Packet Filtering

- Internal network is connected to Internet through a router.
- Router manufacturer provides options for filtering packets, based on:
  - source IP address
  - destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

- Example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.
  - All incoming and outgoing UDP flows and telnet connections are blocked.

- Example 2: Block inbound TCP segments with ACK=0.
  - Prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Application gateways

- Filters packets on application data as well as on IP/TCP/UDP fields.
- Example: allow select internal users to telnet outside.
  1. Require all telnet users to telnet through gateway.
  2. For authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
  3. Router filter blocks all telnet connections not originating from gateway.
Limitations of firewalls and gateways

- **IP spoofing**: router can't know if data "really" comes from claimed source
- If multiple app's. need special treatment, each has own app. gateway.
- Client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- Filters often use all or nothing policy for UDP.
- Tradeoff: degree of communication with outside world, level of security
- Many highly protected sites still suffer from attacks.