TELE9752 Network Operations and Control

Lecture 4: Management Protocols
Outline

• Encoding rules
  BER types
  BER length and value
  Example of Basic Encoding

• Tasks for NM protocols

• SNMP
  History & context
  Messages

• syslog

Protocol: SNMP
SNMP history
  RFC 1052's Recommendation for NM
What's in a name: “Simple”
SNMP and UDP
SNMP fragmentation
SNMP message format
  Sample SNMP messages
Restricting access
  SNMP communities
  Communities restrict access
  MAX-ACCESS + Access Mode
SNMP PDU types
  Standard PDU header
    brief notes about fields
    Full list of errors
Get requests
  GetRequest
    e.g. Address Translation table
  GetNextRequest
    Lexicographic ordering
    GetNextRequest example
  GetBulkRequest
SetRequest
Trap notifications
  SNMPv1-Trap message format
    Trap information
    Generic traps
  SNMPv2-Trap
    Reliability of notifications
    GET vs TRAP
Encoding Rules

MIBs define what information is available, but in order to transfer info, it must be encoded. (recall `htons()` etc in Sockets API)

**Basic Encoding Rules (BER)** are defined in ITU X.690 (originally X.209)/ISO 8825

- Alternatives include Canonical, Distinguished and Packed Encoding Rules (CER, DER, PER)

BER gives some flexibility in encoding (in particular, in formatting length). DER ensures only one possible encoding - e.g. to facilitate comparison/hashing?

Each object is encoded in **Type-Length-Value** format

Usually 1B Type, 1B Length, variable-bytes Value

 Exceptions: NULL may have Length=0 => no Value
BER types

1 type byte has 3 parts:

2b: Class
- 00: universal, e.g. basic data types INTEGER
- 01: application, e.g. SNMP app: INTEGER32, INETADDRESS.
- 10: context-specific, e.g. SNMP GET request
- 11: private – like enterprises/private part of OID tree, but here for defining types rather than objects

1b: Primitive (0) or constructed (1)

5b: Type/tag, e.g.
- 02: INTEGER
- 04: OCTET STRING
- 05: NULL
- 06: OBJECT IDENTIFIER
- 10: ENUMERATED
- 16: SEQUENCE
BER length and value

**Length fields**: (of the *value*, not the TLV triplet)

**Short form**: 1B, MSb = 0

**Long form**: 2-127B†:
- 1stB: MSb=1, other bits = length of length
- other bytes: MSb=0

**Values**: INTEGERs are encoded in 2s-complement form:
- +ve: binary form, with MSb=0
- -ve: binary form, complemented, then add 1
  - e.g. -3 = 00000011 -> 11111100 -> 11111101

OCTET STRINGs are not NULL terminated

OIDs composed using complicated formula [Kaliski]
Example of Basic Encoding

e.g. SNMP header (see Tim's packet zoo b925588b.pcap)
version=0, community=“public”, GetRequest, request-ID=0x061f,
   error-status=0, error-index=0, VarBind OID=SNMPv2-SMI::mib-2.25.3.2.1.5.1, value=unspecified
encoded as (TT=type field, LL=length field):

```
30 4c 02 01 00 04 06 70 75 62 6c 69 63 a0 3f 02
```

```
TT LL TT LL 0
```

```
02 06 1f 02 01 00 02 01 00 30 33 30 0f 06 0b 2b
```

```
LL vv vv
```

```
06 01 02 01 19 03 02 01 05 01 05 00
```

[2 more varbinds not shown, each taking 17B]

```
type 30 = universal constructed sequence, 02 = integer,
04=OCTET STRING, a0=context-specific: get request, 06=OID,
05=NULL
```

Outline

• Tasks for NM protocols
• SNMP
Tasks for NM protocols

• **Carry content**
  – Identify what is to be accessed & its value
  – Encode values
  – Matching responses to requests
  – Introspection - What does the other end support? (protocols & objects)
  – Asynchronous notifications (Trap/Inform)
  – Fragmentation and reassembly

• **Error control:**
  – What if no response received within expected time?
  – Are atomic transactions <AN] supported?

• **Security:** Authentication of manager (or even response/notification from agent)
Protocol: SNMP

Full name: Simple Network Management Protocol
Purpose: Convey management information
Layer: Application
Uses: UDP (or TCP)
Identified by: port number 161 (server) or 162 (traps)
Can also operate over other protocols, e.g. Ethertype 0x814C
[ RFC 4789]
Standards: STD15 (historic), STD62: SNMPv3
SNMP history

Simple Gateway Monitoring Protocol (SGMP) [RFC 1028]
1990: SNMPv1
1995: v2 (developed with SMIv2)
   – Aimed to improve security, but couldn't agree. Used with weak community strings => “SNMPv2c”
   – More efficient bulk reading (GetBulkRequest)
   – Confirmed traps (InformRequest)
   – Minor improvements (e.g. more error codes)
RFC 1052's Recommendation for NM

1. In the **short term**, the Internet community should adopt and adapt the Simple Network Management Protocol (**SNMP**) for use as the basis of common network management throughout the system.

   (Rationale: The software is available and in operation.)

2. In the **longer term**, the Internet research community and the vendors should develop, deploy and test a network management system based on the International Standards Organization (**ISO**) Common Management Information Services/Common Management Information Protocol (**CMIS/CMIP**).

   (Rationale: The Internet community can take the high ground in protocol development by virtue of the experimental environment in which it can operate. Recommendations to the ISO from this community, the IAB and the vendors will carry great weight if they are in the language of the ISO common network management system and if they are rooted in actual experience with implementation and use in the field.)
What's in a name: “Simple”

- **Simple relative to contemporaries** (CMIP)
  - Imperatives/actions effected through `SetRequests`
- **Simple agent**, but complicated manager.
  - Manager fully responsible for error control, e.g. retransmit get/set request; try to detect lost traps.
- **Simple protocol is just part** of the system
  - “how does the client indicate which piece of information it wants to retrieve ... The answer is that SNMP depends on a companion specification called the Management Information Base (MIB). (This explains why SNMP's first name is “simple” - it doesn't tell the whole story.)” - Peterson and Davie, p. 472
SNMP and UDP

SNMP usually used over UDP:
• no setup overhead
• leave error control to app: partial info now better than complete info later

=>
• NM traffic will affect stats for UDP/IP/ARP/link & may affect state of them (e.g. need to know IP route & have ARP entry)
• Unsuitable for bulk transfers because UDP has no flow or congestion control
  – Since ca 1999, SNMP can also operate over TCP [RFC 3430: “SNMP over TCP Transport Mapping”]
SNMP fragmentation

• Mgr/Agent must support datagrams of at least 484B
  – **SNMP avoids IP fragmentation** because it increases sensitivity to loss: any fragment loss causes message damage.
  – If Response too large to fit in datagram, Agent gives tooBig error & Mgr tries separating large request into several smaller requests.

• One SNMP message can carry multiple PDUs - e.g. GetRequest + SetRequest.

• Response to failure of some requests:
  – **Set**: All requests in a message succeed or none.
  – **Get**: SNMPv2 can return values for valid requests and point (with error-index) to erroneous request.
SNMP message format

[RFC1157]
Message ::= SEQUENCE {
  version -- version-1 for this RFC
    INTEGER {
      version-1(0)
    },
  community -- community name
    OCTET STRING,
  data -- e.g., PDUs if trivial
    ANY -- authentication is being used
}

Note that SNMP confusingly uses “PDU” to name the objects carried in its messages, whereas for most protocols Protocol Data Units are what the protocol exchanges with its peer.
SNMP communities

- **Communities define groups** of Managers & Agents that are permitted to communicate.
- **Identified by a text string.** Default community = “public”
  - (Until SNMPv3 only authentication of membership is through sending, in plaintext, of community text string => Insecure!)
- Both Managers & Agents can belong to multiple communities.

**e.g.** define EE&T, CSE & UNSW communities. (like local+“global” NOC)

- EE&T Manager belongs to EE&T community alone. Similarly for CSE.
- EE&T Elements give read access to EE&T community & read-write access to UNSW community.
  - EE&T & CSE Managers can get info from their elements, but not set nor get info from other's elements.
- UNSW institutional IT Support Manager belongs to UNSW community & can get&set all elements.
Communities restrict access

• Recall: SMI defines MAX-ACCESS for each object ("MIB view"): maximum access that makes "protocol sense" e.g. can't write to sysObjectID
• Each element defines, for each community, an "access mode" (READ-ONLY or READ-WRITE) for each object.
• Permissible access determined by MIB view x access mode (next slide [N0])
### MAX-ACCESS + Access Mode

<table>
<thead>
<tr>
<th>MIB view</th>
<th>Community Access Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX-ACCESS</td>
<td>READ-ONLY</td>
</tr>
<tr>
<td>read-only</td>
<td>get &amp; trap</td>
</tr>
<tr>
<td>read-write</td>
<td>get&amp;trap</td>
</tr>
<tr>
<td>read-create</td>
<td>get&amp;trap</td>
</tr>
<tr>
<td>accessible-for-notify</td>
<td>trap</td>
</tr>
<tr>
<td>not-accessible</td>
<td>can't set</td>
</tr>
</tbody>
</table>

In terms of actual PDUs, listed later [KW>:  
“get” = GetRequest + GetNextRequest + GetBulkRequest  
“trap” = SNMPv2-Trap + InformRequest  
“set” = SetRequest
SNMP messages

Main types of “PDUs”:
• Requests: Get & Set
• Responses to requests
• Trap notifications
Full list on slide [KW>]

Use Wireshark to view packets in Tim's Packet Zoo
[http://uluru.ee.unsw.edu.au/~tim/zoo/index.html#SNMP]
• 2 examples on the following slides
Text in figure: SNMP get-request

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.000000</td>
<td>172.31.19.54</td>
<td>172.31.19.73</td>
<td>SNMP</td>
<td>get-request SNMPv2-MIB::sysObjectID.0</td>
</tr>
<tr>
<td>2</td>
<td>20.000067</td>
<td>172.31.19.73</td>
<td>172.31.19.54</td>
<td>SNMP</td>
<td>get-response SNMPv2-MIB::sysObjectID.0</td>
</tr>
<tr>
<td>3</td>
<td>30.003623</td>
<td>172.31.19.54</td>
<td>172.31.19.73</td>
<td>SNMP</td>
<td>get-request SNMPv2-MIB::sysName.0</td>
</tr>
<tr>
<td>4</td>
<td>40.009495</td>
<td>172.31.19.73</td>
<td>172.31.19.54</td>
<td>SNMP</td>
<td>get-response SNMPv2-MIB::sysName.0</td>
</tr>
</tbody>
</table>

Simple Network Management Protocol

version: version-1 (0)
  community: public
  data: get-request (0)
  get-request
    request-id: 38
    error-status: noError (0)
    error-index: 0
  variable-bindings: 1 item
    SNMPv2-MIB::sysObjectID.0 (1.3.6.1.2.1.1.2.0): unSpecified
    Object Name: 1.3.6.1.2.1.1.2.0 (SNMPv2-MIB::sysObjectID.0)
      UnSpecified

Frame 1 (82 bytes on wire, 82 bytes captured)

Ethernet II, Src: Dell_4a:33:d2 (00:12:3f:4a:33:d2), Dst: Fuji-Xer_15:e6:bc (08:00:37:15:e6:bc)
  Internet Protocol, Src: 172.31.19.54 (172.31.19.54), Dst: 172.31.19.73 (172.31.19.73)
  User Datagram Protocol, Src Port: 15916 (15916), Dst Port: snmp (161)
Text in figure: SNMP get-response
SNMP PDU types

[RFC3416]

**Bold green** = new for SNMPv2

**Italic blue** = modified names

\[
PDU s \ ::= \ CHOICE \ { \\
\text{get-request} \quad \text{GetRequest-PDU}, \\
\text{get-next-request} \quad \text{GetNextRequest-PDU}, \\
\text{get-bulk-request} \quad \text{GetBulkRequest-PDU}, \\
\text{response} \quad \text{Response-PDU}, \\
\text{set-request} \quad \text{SetRequest-PDU}, \\
\text{inform-request} \quad \text{InformRequest-PDU}, \\
\text{snmpV2-trap} \quad \text{SNMPv2-Trap-PDU}, \\
\text{report} \quad \text{Report-PDU} \ } \\
\]

(Report-PDU is not further defined.)

+ Name changes: get-response --> response, trap-->snmpv2trap
Outline

• standard PDU header
  – error fields & error control
  – VarBinds and multiple PDUs
• get
  – getnext
  – getbulk
• set
• notifications
  – traps
  – inform
  – (week 6: RMON alarms [Z5>])
Standard PDU header

PDU ::= SEQUENCE {
    request-id INTEGER (-214783648..214783647),
    error-status INTEGER {noError(0), tooBig(1), ... },
    error-index INTEGER (0..max-bindings),
    variable-bindings
        VarBindList
}

GetBulkRequest PDUs replace error-status and error-index with non-repeaters & max-repetitions

trap messages have different format (for no good reason)
brief notes about fields

request-id: allows Manager to match responses to requests. SNMP doesn't standardise error control: leaves recovery from loss to the implementation

Error fields are ignored on requests

error-status: Full list on next slide [1Q>.

error-index: Identifies which variable caused an error (if any) by indicating the position of the variable in the VarBindList

Variable-bindings (VarBinds): Pairs of variable identifiers (OID) and value. Both are encoded using BER (e.g. type of variable=OID, type of value = INTEGER)
Full list of errors

noError(0)    Hooray!
tooBig(1)      Couldn't fit response in datagram

Access errors: readOnly(4), noAccess(6),
noCreation(11), notWritable(17),

Encoding errors: wrongType(7), wrongLength(8),
wrongEncoding(9), wrongValue(10),

Other errors: genErr(5), noSuchName(2), badValue(3),
inconsistentValue(12), resourceUnavailable(13),
commitFailed(14), undoFailed(15),
authorizationError(16), inconsistentName(18)
GetRequest

Self-explanatory?!

Note **OIDs used to identify**

- **scalars**: OID.0, e.g. sysUpTime.0
- **cells of tables**: OID(table entry).col.indexvals, e.g. [QC>

Doesn't OID + value = **redundancy** for GetRe*?:

Request to agent: Value set to NULL

  Kept for consistency with Responses & SetRequests

Response to Manager: GetRequest => OID

  but OID useful for:
  - partial response to multiple GetRequests
  - GetNextRequest
(Get-)Response

SNMPv1 defines a "get-response" PDU, but uses it to respond to get or set requests. SNMPv2 rightly renames it to just "Response".

Response to get-requests provides object value

Response also provides acknowledgement [WA>
e.g. Address Translation table

AT is deprecated by MIB-II, but used in examples from Subramanian (forthcoming Figs) and Stevens pp. 375-6 1.3.6.1.2.1.at(3).atTable.atEntry(1).

MIB definition:

atEntry OBJECT-TYPE ...
   SYNTAX AtEntry ...
   INDEX {atIfIndex,atNetAddress}
   ::= {atTable 1}

AtEntry ::= SEQUENCE {
   atIfIndex INTEGER,
   atPhysAddress PhysAddress,
   atNetAddress NetworkAddress
}

<table>
<thead>
<tr>
<th>atIfIndex</th>
<th>atPhysAddress</th>
<th>atNetAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0xaa:00:04:00:f4:14</td>
<td>140.252.1.4</td>
</tr>
<tr>
<td>1</td>
<td>0x08:00:20:0f:2d:38</td>
<td>140.252.1.22</td>
</tr>
<tr>
<td>1</td>
<td>0x00:80:ad:03:6a:80</td>
<td>140.252.1.183</td>
</tr>
<tr>
<td>2</td>
<td>0x00:02:16:48</td>
<td>140.252.6.4</td>
</tr>
<tr>
<td>2</td>
<td>0x00:02:3c:48</td>
<td>140.252.6.6</td>
</tr>
</tbody>
</table>

These .1s exist because atNetAddress offers a choice of network addresses, with the .1 choice being an IPv4 address.
GetNextRequest

• GetNextRequest allows Mgr to get objects that it doesn't yet know the identities of (data from tables of variable size) - known as “walking a MIB”
  – GetNextRequest renders GetRequest redundant
  – If there is nothing next, value = “endOfMibView”†

• Need to specify OID that acts as reference for next. Not just for the 1st access: Also for successive accesses since SNMP operates over unreliable UDP.

† See RFC 3416 for details. The varbind includes the object ID and a choice of a value or error codes such as endOfMibView.
Lexicographic ordering

GetNextRequests require a linear ordering of objects in the tree. “Lexicographic ordering”:

- Access children before accessing siblings (objects with same parent). i.e. depth-first traversal of tree.
- Access siblings in order of numerically increasing OID.

ORIGIN from Greek lexikon biblion ‘book of words’, from lexis ‘word’.
GetNextRequest example

Puzzles for which solutions deserve bonus marks: How does agent order rows when:
1. INDEX value is not an integer, e.g. IP addresses MSB 1st or last?
2. they have the same INDEX value

Fig. 5.16 from Subramanian, mistakenly omits the .1 before the IP address <QC>
GetBulkRequest

GetNextRequests require many exchanges =>
• network load
• delay

GetBulkRequest allows repeated GetNext without resending Request

2 “new” fields (replacing error indicators of dubious value in requests):
• non-repeaters (replaces error-status): How many vars @ head of VarBind are to NOT be repeated (i.e. just GetNext for these).
• max-repetitions (replaces error-index): For remaining variables, how many repetitions are sought? (subject to object availability & space in message)
GetBulkRequest example

Fig. 6.42 from Subramanian, mistakenly omits the .1 before the IP address <QC>
Note that the last line of the 1st response should probably be 23.192.168.3.1
SetRequest

- Response to *SetRequest* echoes new value back to manager
- *SetRequest* rarely enabled prior to SNMPv3 due to insecurity.
- There is no *SetNextRequest*. Instead use *GetNextRequests* to identify objects, and then set
- (This should have appeared last week near slide <AN>:)

For mutual exclusion
- between Managers: Use SpinLocks
- between Manager and Element: No problem:
  - Element can block Manager requests while modifying.
  - Managers don't usually modify objects modified by Element. e.g. performance statistics are read-only
  Counters: Manager will not get then set & lose events between those requests.
SNMPv1-Trap message format

IMPLICIT SEQUENCE {
  enterprise -- type of object† generating trap
    OBJECT IDENTIFIER,
  agent-addr -- address of object† generating
    NetworkAddress, -- trap
  generic-trap -- generic trap type
    INTEGER {coldStart(0), warmStart(1), ... enterpriseSpecificSpecific(6)},
  specific-trap -- specific code
    INTEGER,
  time-stamp -- time since element rebooted
    TimeTicks,
  variable-bindings
    VarBindList
} [RFC1157] † errors: “object” should be “network element”
Trap information

- Trap messages provide asynchronous notifications: element tells manager, without being requested by manager
- **agent-addr**: Element may not run its own manager & may not have a unique IP address (e.g. may have multiple interfaces; may use private IP addresses) => agent-addr to identify source of info
- **VarBindList** - e.g. identifying interface that went up/down

Exercise: Consider how trap could be expressed in same format as **Response**.
Generic traps

**Startup traps**: Element is reinitializing => Manager can:
- Investigate the cause
- Correctly interpret objects of TimeTicks type
- Consider repairing configuration (coldStart)

\texttt{coldStart(0)}: startup + config may have been altered.
\texttt{warmStart(1)}: startup + config unaltered.

**Change in link (interface) status**:
\texttt{linkDown(2), linkUp(3)}

\texttt{authenticationFailure(4)}: Message having invalid community field received.
\texttt{egpNeighborLoss(5), enterpriseSpecific(6)}
SNMPv2-Trap

• 1st 2 variables are always sysUpTime and snmpTrapOID
• Identify type of trap with OID instead of integer, e.g. internet.snmpv2.snmpModules.snmpMIB.snmpMIBObjects.snmpTraps.linkDown & defined using SMI NOTIFICATION-TYPE macro, e.g.

   linkDown NOTIFICATION-TYPE
   OBJECTS { ifIndex }
   STATUS current
   DESCRIPTION "A linkDown trap signifies that the SNMPv2 entity, acting in an agent role, recognizes a failure in one of the communication links represented in its configuration."

   ::= { snmpTraps 3 }
Reliability of notifications

- Trap messages may be lost => Manager unaware of event.
- Partially to simplify agent: no need to buffer and retransmit.
- Undesirable in a hierarchy of management agents: May want to forward traps up hierarchy and lower agent has resources to retransmit.
- SNMPv2 introduces **InformRequest**: Same content as a trap but recipient confirms receipt by sending a **Response**. Retransmission policy implementation-dependent.

+ A manager might later detect a missing coldStart/warmStart trap when it notices that the sysUptime is not what it expects, given its knowledge of previous startup times.
GET vs TRAP

(Like polling vs interrupts for computer I/O):

**TRAPs better than GET:**
- # of messages exchanged: Often GET no change.
- Timeliness: Time between event and next GET.
  - except element failures: may prevent notification with trap => detect from GET time out

**GET better than TRAP:**
- Timeliness of detecting *element* failure. e.g. time to next GET + timeout vs detect never or after element restarts.
- Robustness? (InformRequest intended to rectify this)
Outline

• syslog
syslog

- Many NMs prefer CLIs to SNMP get/set (particularly before SNMPv3 security).
- But CLI provides no asynchronous notifications.

=> Often augment with syslog

- syslog originated from Unix for logging system events
- IETF extensions:
  - network event notifications
  - protocol for carrying syslog data
Protocol: syslog

Full name: syslog / system log
Purpose: Convey event notification messages
Layer: Application
Uses: UDP (or TLS over TCP)
Identified by: port number: UDP 514 or TCP 6514
Standards: Proposed Standard
• MAC addr & message suggest this is from a Nokia IPSO firewall

The end for this week

- SNMPv3 covered in security lecture
- Mid-session test 6:30pm next week, covering material up to this week
- Consultation time: Tuesdays 5:15-6pm in room 341