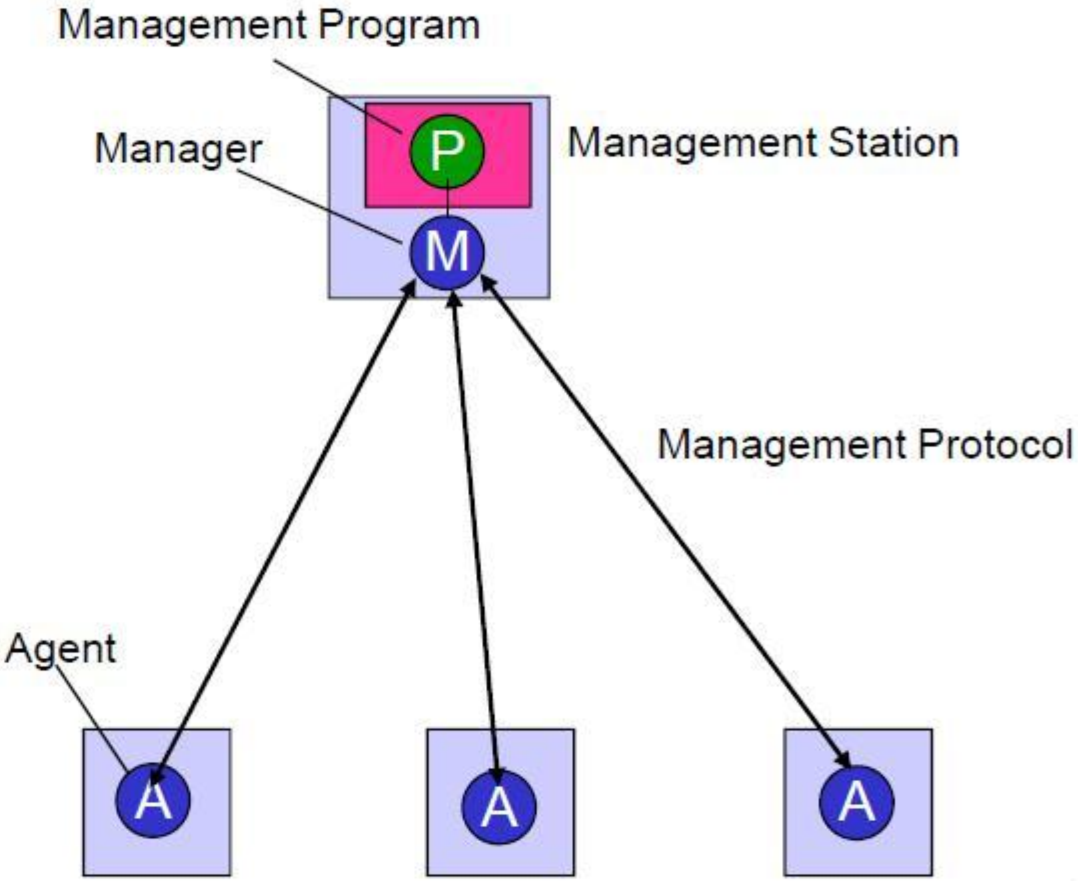
SNMP

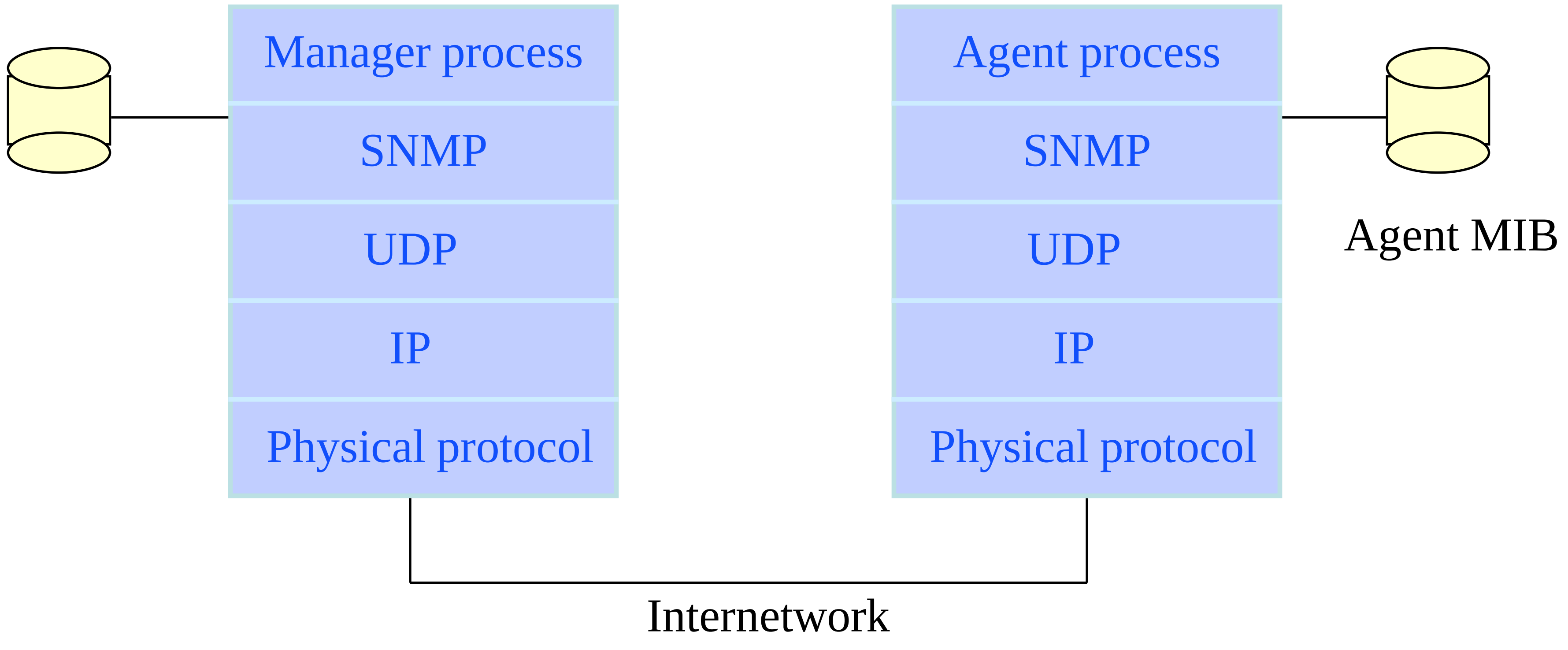
...Simple Network Management Protocol...

Outline of the SNMP Framework

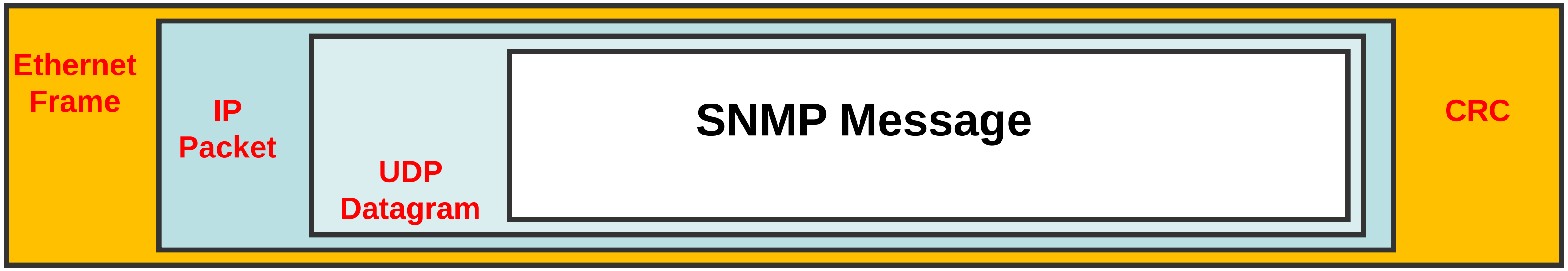


SNMP Transport Architecture

- UDP
  - „unreliable” transport layer



SNMP Encapsulation



- **UDP Port 161** - SNMP Messages
- **UDP Port 162** - SNMP Trap Messages

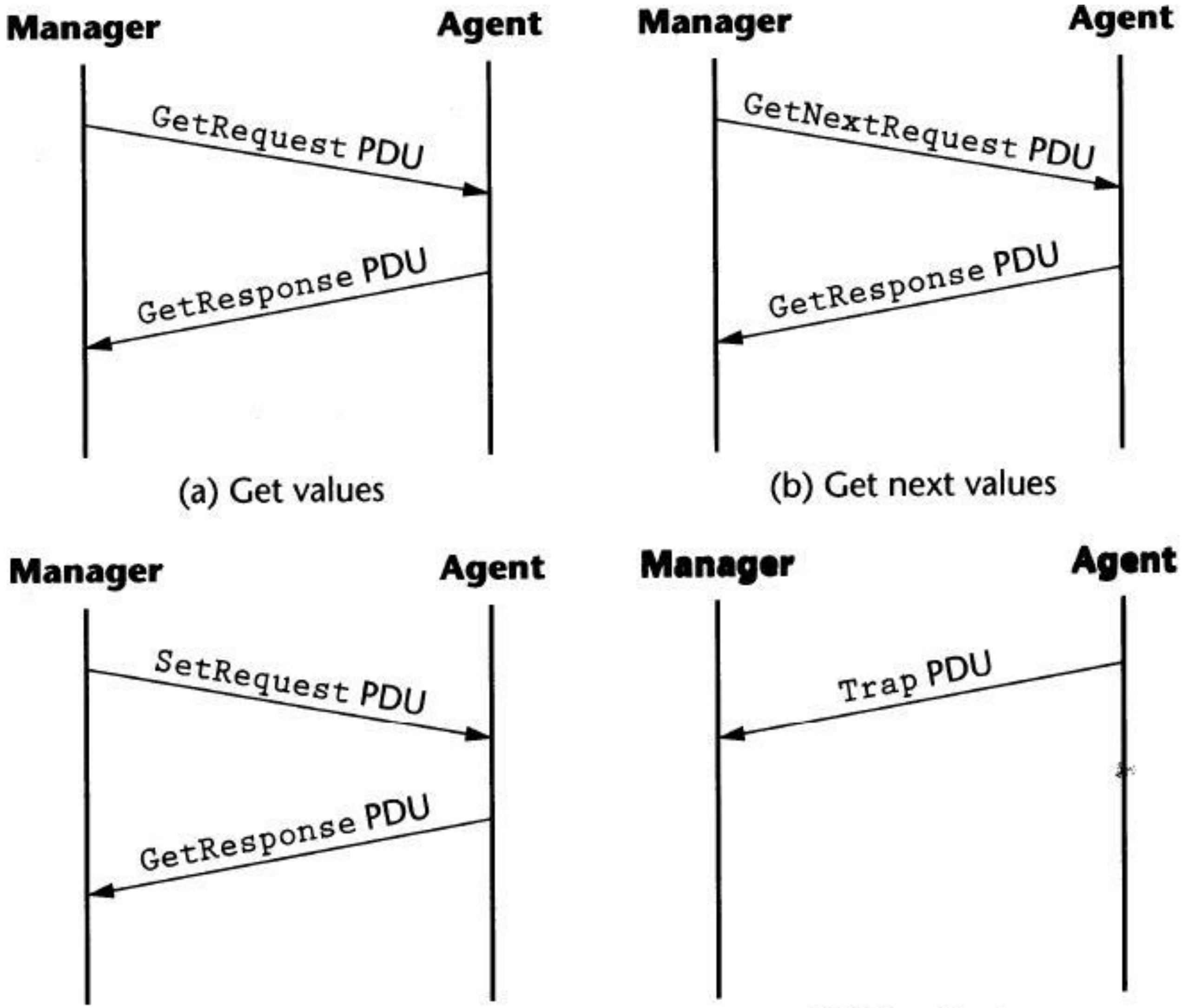
Connectionless Protocol

- Because of UDP
- There are no low level guarantees for receiving the management-traffic
- Pro’s
  - Smaller overhead
  - Simpler protocol
- Con’s
  - The Connection-oriented behavior (if needed) must be implemented by the application

SNMP Operations

- **Get-Request(0)** requests a value or set of values from a Management Agent MIB.
- **Get-Next-Request(1)** requests the value of the next lexicographically larger Object Identifier in a MIB tree given the present Object Identifier.
- **Get-Response(2)** is a response from the management agent to the management station supplying the requested values.
- **Set-Request(3)** sets a value (or an action) in the management agent MIB.
- **Trap(4)** is an unsolicited message from a management agent to management station that is initiated by an alarm/event pair on the management agent.

SNMP Message Sequences



(a) Get values

(b) Get next values

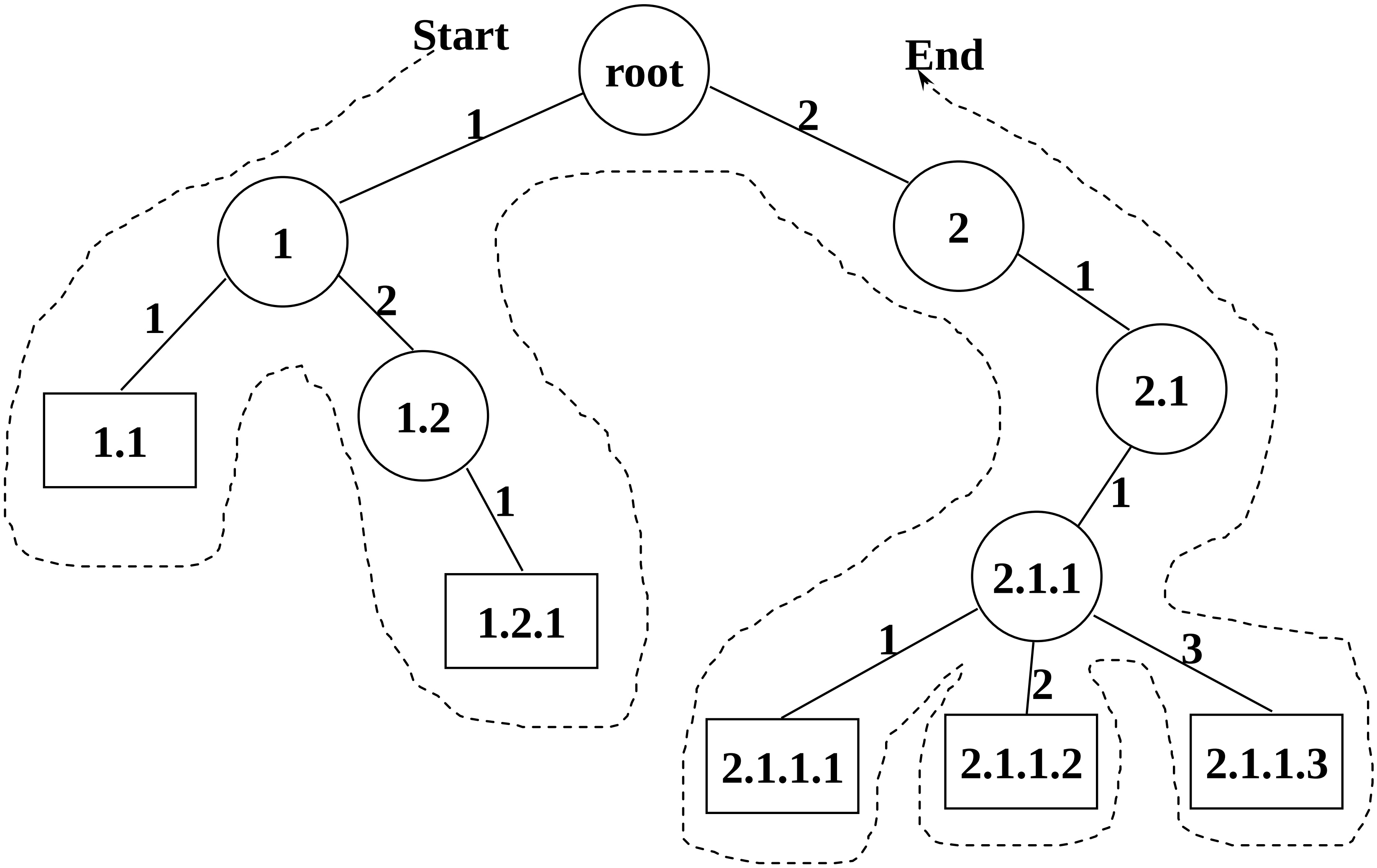
(c) Set values

(d) Send trap

Lexicographical Ordering

- is used for **accessing MIB objects serially**
- given the tree structure of a MIB, the OID for a particular object may be derived by tracing a path from the root to the object
- **lexicographical ordering is also referred to as:**
  - preorder traversal (root, left, right) of a tree
  - depth-first search
- **useful for examining MIBs whose structure is not known to NMS**

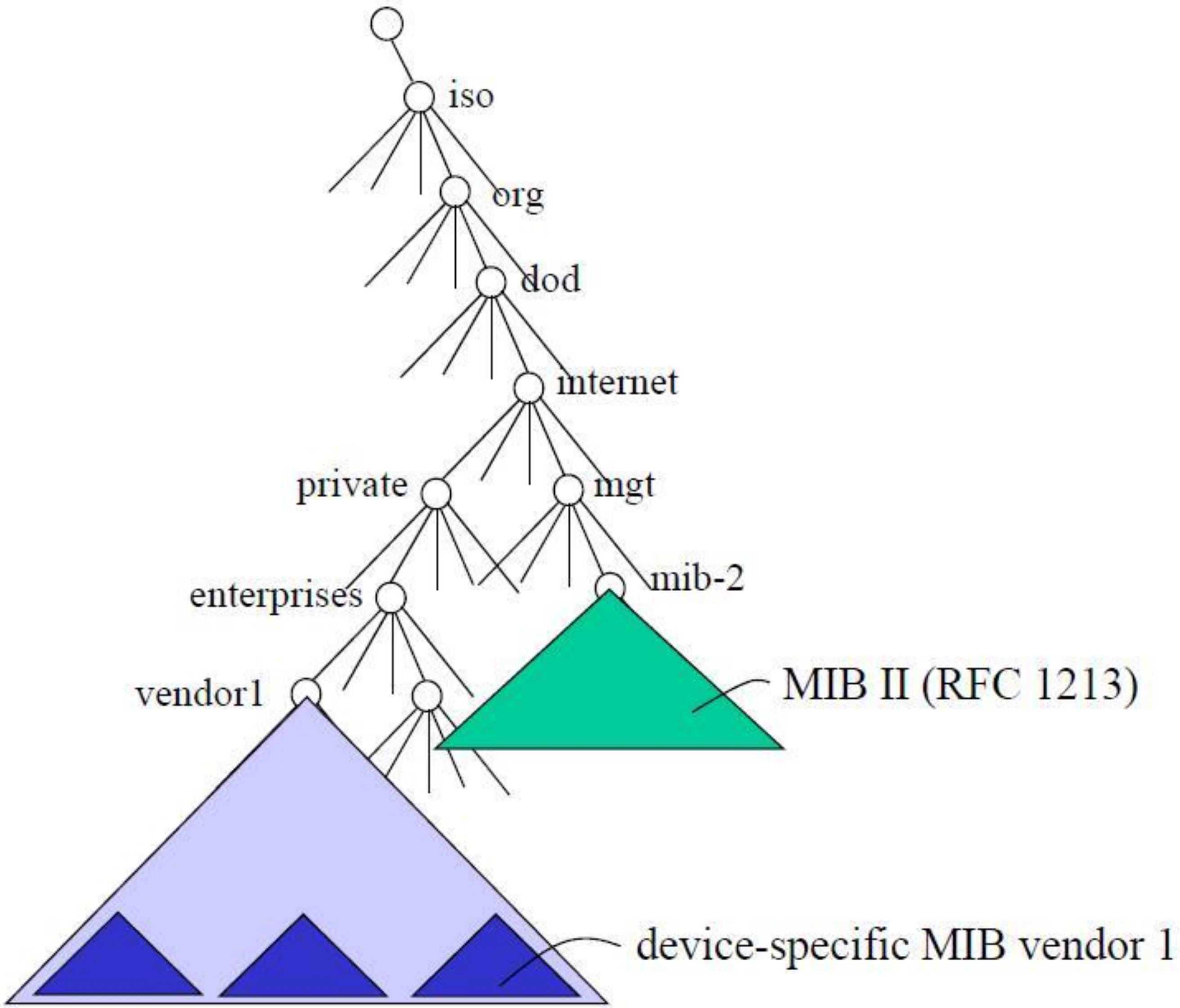
Lexicographical Ordering Example



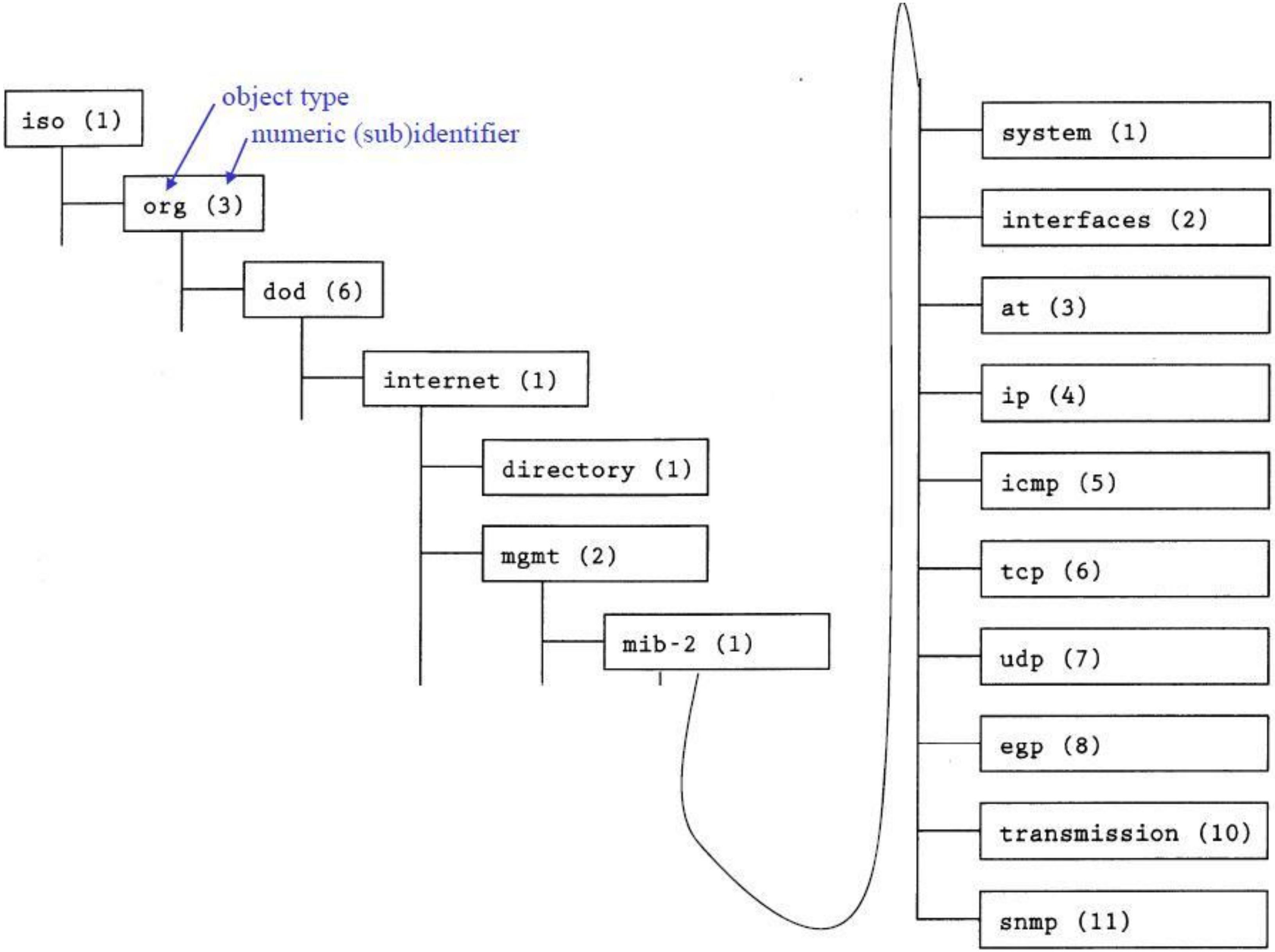
SNMP Management Information

- Management Information is modeled as *(managed) objects* and relationships among them.
- A ***MIB (Management Information Bases)*** *is a collection of* objects, grouped for a specific management purpose.
- All objects are organized in the ***global MIB tree****.*
- Each MIB represents a sub tree of this global MIB tree.
- The *leaf objects of the tree contain object instances with* the state and control variables of the managed system.
- Device manufacturers often define their own device specific MIBs.

The Global MIB Tree



MIB Tree Example



SNMP Operations Cont’d

- **Not possible to change the structure of a MIB**
  - cannot add or delete object instances
- **No explicit action is supported**
- **Access is provided only to leaf objects in the MIB tree**
  - not possible to access an entire table or a row of a table with a single atomic action
- **These simplify the implementation of SNMP but limit the capability of the NMS**

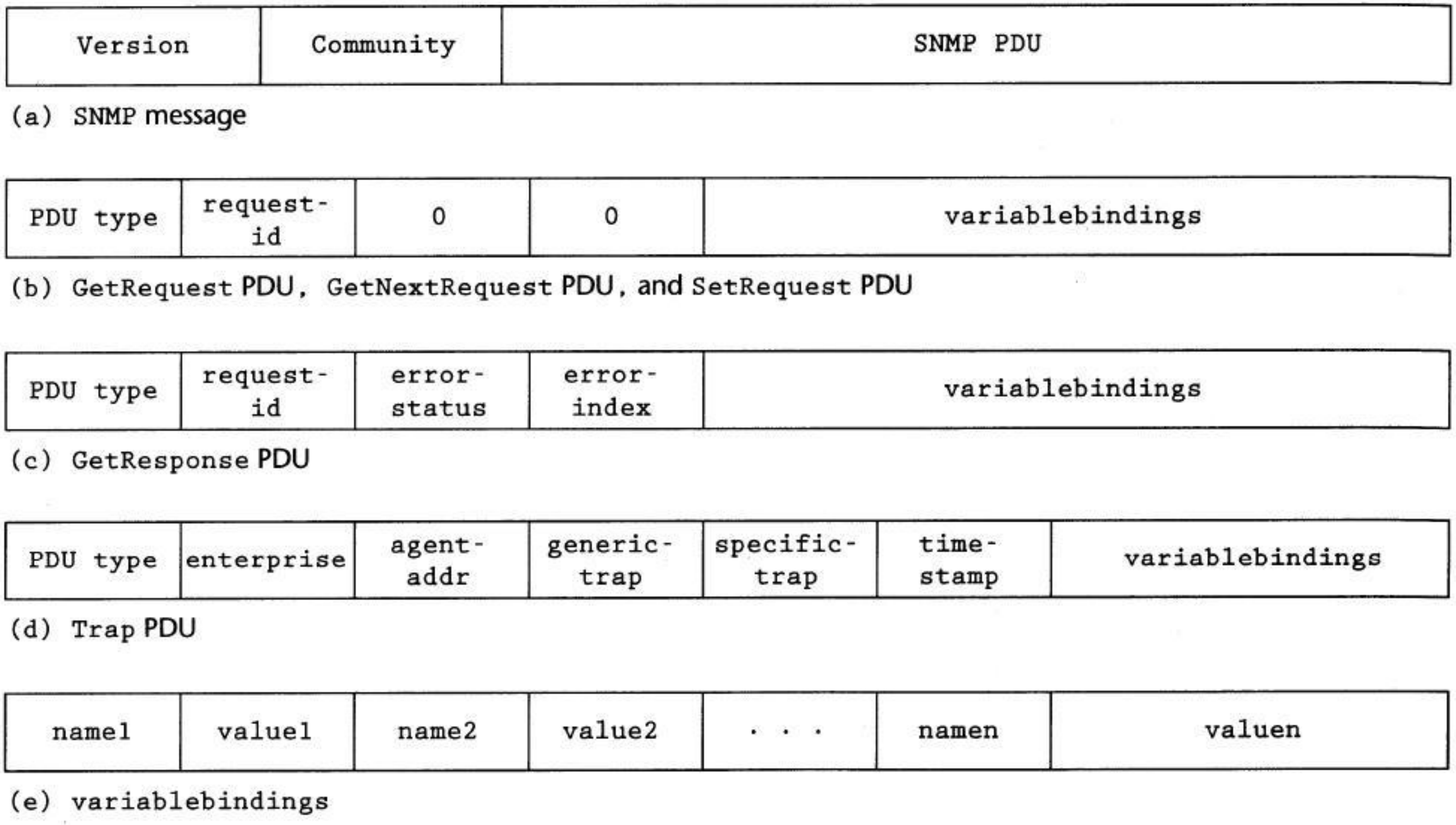
The Structure of SNMP Management Information (SMI)

- SMI, the SNMP management information model, provides guidelines for defining MIBs, object types and object identifiers.

- These definitions are written in the language ASN.1 (Abstract Syntax Notation 1).

- ASN.1 includes also rules on how the management information is encoded, i.e., mapped into octet strings.

SNMP PDU Structure



Version	Community	SNMP PDU

(a) SNMP message

PDU type	request-id	0	0	variablebindings

(b) GetRequest PDU, GetNextRequest PDU, and SetRequest PDU

PDU type	request-id	error-status	error-index	variablebindings

(c) GetResponse PDU

PDU type	enterprise	agent-addr	generic-trap	specific-trap	time-stamp	variablebindings

(d) Trap PDU

name1	value1	name2	value2	. . .	namen	valuen

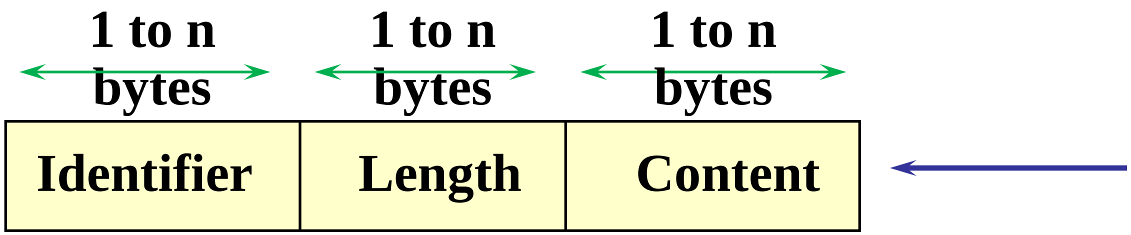
(e) variablebindings

SNMP Encoding using ASN.1

- CCITT (X.209) and ISO (ISO 8825)

- Basic Encoding Rules (BER)
  - Type-Length-Value (TLV))
  - Recursive structure,
    - «V» can contain another TLV

Encoding a value



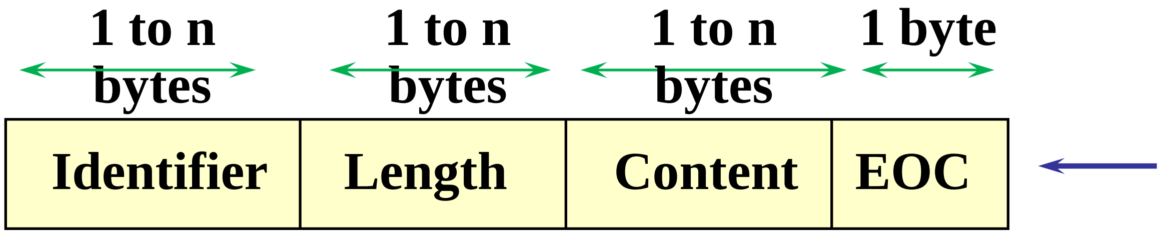
1 to n bytes

1 to n bytes

1 to n bytes

Identifier	Length	Content

Value length is known



1 to n bytes

1 to n bytes

1 to n bytes

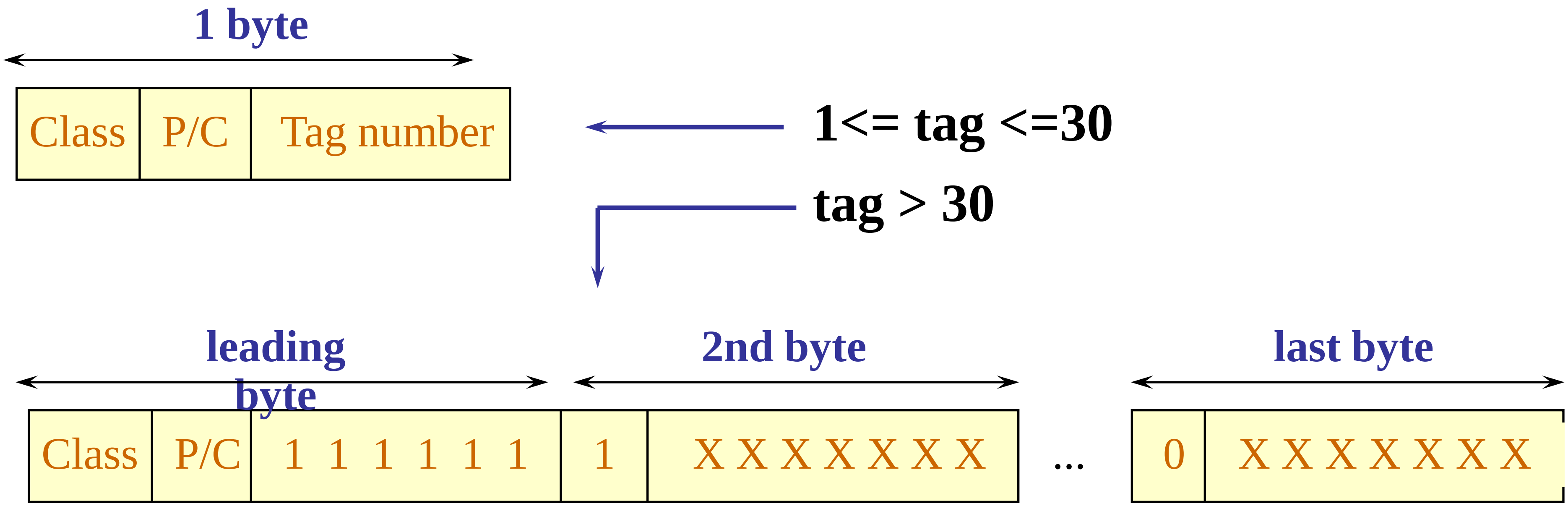
1 byte

Identifier	Length	Content	EOC

Value length is not known at the moment

EOC = 00000000

Identifiers



1 byte

Class | P/C | Tag number

1<= tag <=30

tag > 30

leading byte

2nd byte

last byte

Class | P/C | 1 1 1 1 1 1 | 1 | X X X X X X X ... 0 | X X X X X X X

<u>Class</u> :

00 = Universal

01 = Application

10 = Context specific

11 = Private

<u>P/C</u> :

0 = Primitive type

1 = Constructed type

<u>Tag number</u> :

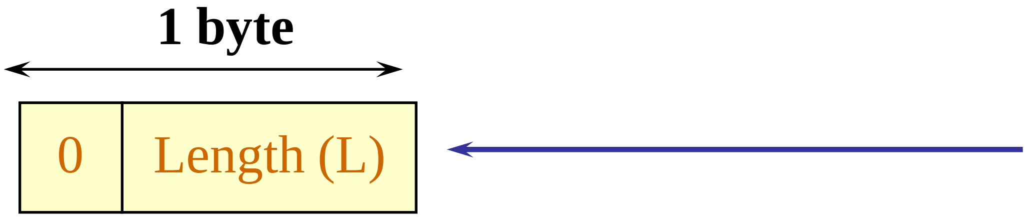
1 = Boolean type

2 = Integer type

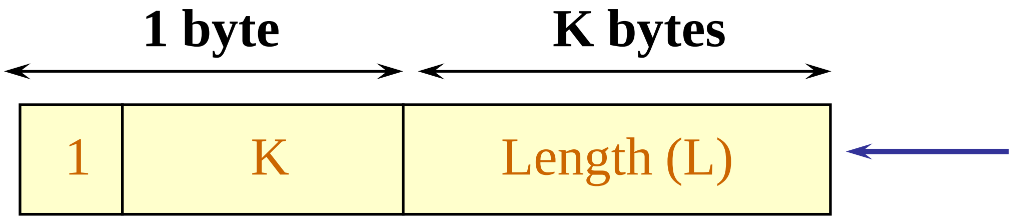
...

> 30 : X...X = tag number

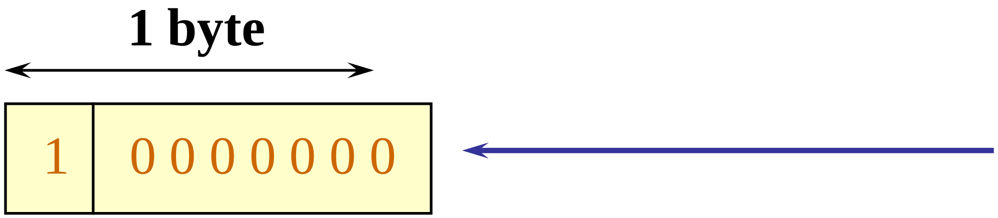
Length



Short, known length:
1<= L <= 127



Long, known length:
128 <= L <= 2^{128}



Unknown length:
ending with EOC

ASN.1 Encoding Example

TYPE	VALUE	ENCODING
INTEGER	-129	02 02 FF 7F
OCTET STRING	«John»	04 04 4A 6F 68 6E

SEQUENCE
(INTEGER,
INTEGER)
	(3, 8)	30 06
		02 01 03
		02 01 08

Example: Encoding **Get Request**

GET 1.3.6.1.2.1.1.1.0 (sysDescr)

```
30 27                         SEQUENCE (0x30) 39 bytes
  02 01 00                      INTEGER VERSION (0x2) 1 byte: 0
  04 06 70 75 62 6c 69 63       OCTET STRING COMMUNITY (0x4) 6 bytes:
                                                       «public»
  a0 1a                           GET-REQUEST-PDU (0xa0) 26 bytes
    02 02 73 00                     INTEGER REQUEST-ID (0x2) 2 bytes: 29440
      02 01 00                      INTEGER ERROR-STATUS (0x2) 1 byte:
                                                       noError
      02 01 00                      INTEGER ERROR-INDEX (0x2) 1 byte: 0
      30 0e                         SEQUENCE (0x30) 14 bytes
        30 0c                         SEQUENCE (0x30) 12 bytes
          06 08 2b 06 01 02 01 01 01 00     OBJECT ID (0x6) 8 bytes:
                                                 1.3.6.1.2.1.1.1.0
          05 00                             NULL VALUE (0x5) 0 byte
```

Example: Encoding Get Response

GET RESPONSE 1.3.6.1.2.1.1.1.0 (sysDescr = «alphaB...»)

```
30 81 84                  SEQUENCE (0x30) 132 bytes
  02 01 00                    INTEGER VERSION (0x2) 1 byte: 0
  04 06 70 75 62 6c 69 63         OCTET STRING COMMUNITY (0x4) 6 bytes:
                                                   «public»
  a2 77                       GET-RESPONSE-PDU (0xa2) 119 bytes
    02 02 73 00                  INTEGER REQUEST-ID (0x2) 2 bytes: 29440
    02 01 00                     INTEGER ERROR-STATUS (0x2) 1 byte: noError
    02 01 00                     INTEGER ERROR-INDEX (0x2) 1 byte: 0
    30 6b                        SEQUENCE (0x30) 107 bytes
      30 69                          SEQUENCE (0x30) 105 bytes
        06 08 2b 06 01 02 01 01 01 00    OBJECT ID (0x6) 8 bytes:
                                              1.3.6.1.2.1.1.1.0
        04 5d 61 6c 70 68 61 42 ...      OCTET STRING (0x4) 93 bytes:
                                              «alphaB...»
```

SNMP Security Concepts

- **Authentication service**
  - agent may wish to limit access to the MIB to authorized managers
- **Access policy**
  - agent may wish to give different access privileges to different managers
- **Proxy service**
  - agent may act as a proxy to other managed devices
  - this may require authentication service and access policy for other managed devices on the proxy
- **SNMP provides only a primitive and limited security capability via the concept of *community***

SNMP Community

- **is a relationship between an agent and a set of managers that defines authentication, access control & proxy characteristics**
- **a community is locally defined by the agent**
  - each community is given a unique community name
  - an agent may establish a number of communities
  - the community name is needed for all get and set operations
  - the same community name may be used by different agents
- **SNMP authentication service**
  - every SNMP message from a manager includes a community name (used as a password) --- very primitive
  - most agents only allow GET operations

SNMP Community Cont’d

- **SNMP Access Policy**
  - an agent can provide different categories of MIB access using the following concepts: **SNMP MIB View** & **Access Mode**
- **SNMP MIB View**
  - a subset of objects within a MIB
  - different MIB views may be defined for each community
  - the set of objects in a view need not belong to a single subtree
- **SNMP Access Mode**
  - an access mode {READ-ONLY, READ-WRITE} is defined for each community
  - the access mode is applied uniformly to all objects in the MIB view
- **SNMP Community Profile**
  - a combination of a MIB view and an access mode

MIB ACCESS Category *vs.* SNMP Access Mode

MIB ACCESS Category	SNMP Access Mode	
	READ-ONLY	READ-WRITE
read-only	Available for get and trap operations	
read-write	Available for get and trap operations	Available for get, set, and trap operations
write-only	Available for get and trap operations, but the value is implementation-specific	Available for get, set, and trap operations, but the value is implementation-specific for get and trap operations.
not accessible	Unavailable	

SNMP RFC’s

RFC	Description	Published	Current Status
1065	SMIv1	Aug-88	Obsoleted by 1155
1066	SNMPv1 MIB	Aug-88	Obsoleted by 1156
1067	SNMPv1	Aug-88	Obsoleted by 1098
1098	SNMPv1	Apr-89	Obsoleted by 1157
1155	SMIv1	May-90	Standard
1156	SNMPv1 MIB	May-90	Historic
1157	SNMPv1	May-90	Standard
1158	SNMPv1 MIB-II	May-90	Obsoleted by 1213
1212	SNMPv1 MIB definitions	Mar-91	Standard
1213	SNMPv1 MIB-II	Mar-91	Standard
1215	SNMPv1 traps	Mar-91	Informational
1351	Secure SNMP administrative model	Jul-92	Proposed Standard
1352	Secure SNMP managed objects	Jul-92	Proposed Standard
1353	Secure SNMP security protocols	Jul-92	Proposed Standard
1441	Introduction to SNMPv2	Apr-93	Proposed Standard
1442	SMIv2	Apr-93	Obsoleted by 1902
1443	Textual conventions for SNMPv2	Apr-93	Obsoleted by 1903
1444	Conformance statements for SNMPv2	Apr-93	Obsoleted by 1904
1445	SNMPv2 administrative model	Apr-93	Historic
1446	SNMPv2 security protocols	Apr-93	Historic
1447	SNMPv2 party MIB	Apr-93	Historic
1448	SNMPv2 protocol operations	Apr-93	Obsoleted by 1905
1449	SNMPv2 transport mapping	Apr-93	Obsoleted by 1906
1450	SNMPv2 MIB	Apr-93	Obsoleted by 1907
1451	Manger-to-manger MIB	Apr-93	Historic
1452	Coexistence of SNMPv1 and SNMPv2	Apr-93	Obsoleted by 1908
1901	Community-Based SNMPv2	Jan-96	Experimental
1902	SMIv2	Jan-96	Draft Standard
1903	Textual conventions for SNMPv2	Jan-96	Draft Standard
1904	Conformance statements for SNMPv2	Jan-96	Draft Standard
1905	Protocol operations for SNMPv2	Jan-96	Draft Standard
1906	Transport mapping for SNMPv2	Jan-96	Draft Standard
1907	SNMPv2 MIB	Jan-96	Draft Standard
1908	Coexistence of SNMPv1 and SNMPv2	Jan-96	Draft Standard
1909	Administrative infrastructure for SNMPv2	Feb-96	Experimental
1910	User-based security for SNMPv2	Feb-96	Experimental