IT3010

Network Design and Management

Special Topics in SNMP

Shashika Lokuliyana

Faculty of Computing

Department of CSE

SLIIT

Discover Your Future

Structure of management information (SMI)

Goal of SNMP

Or at least the intention of the inventors…

- A world where a person’s audio system, video system, HVAC system and toaster are all connected to the same network.
- To that end, computer scientists developed a protocol capable of managing any network device.
- The result was Simple Network Management Protocol (SNMP).

Problem 1

- Different software languages have slightly different sets of data types (integers, strings, bytes, characters etc.).
- But an SNMP manager sending a message full of Java data types may not be understood by an SNMP agent written in C.
- The solution for this problem is to use ASN.1 defined data types.
- Since ASN.1 is independent of any particular programming language, the SNMP agent/manager can be written in any programming language.

Problem 2

- When sending a particular data type over the wire, how should it be encoded?
- Should strings be null terminated as in the programming language C, or not? Should Boolean values be 8 bits as in C++ or 16 bits as in VB6?
- To address this problem ASN.1 includes Basic Encoding Rules (BER).
- To send a properly formatted message, the programmer must understand ASN.1 and BER encoding.

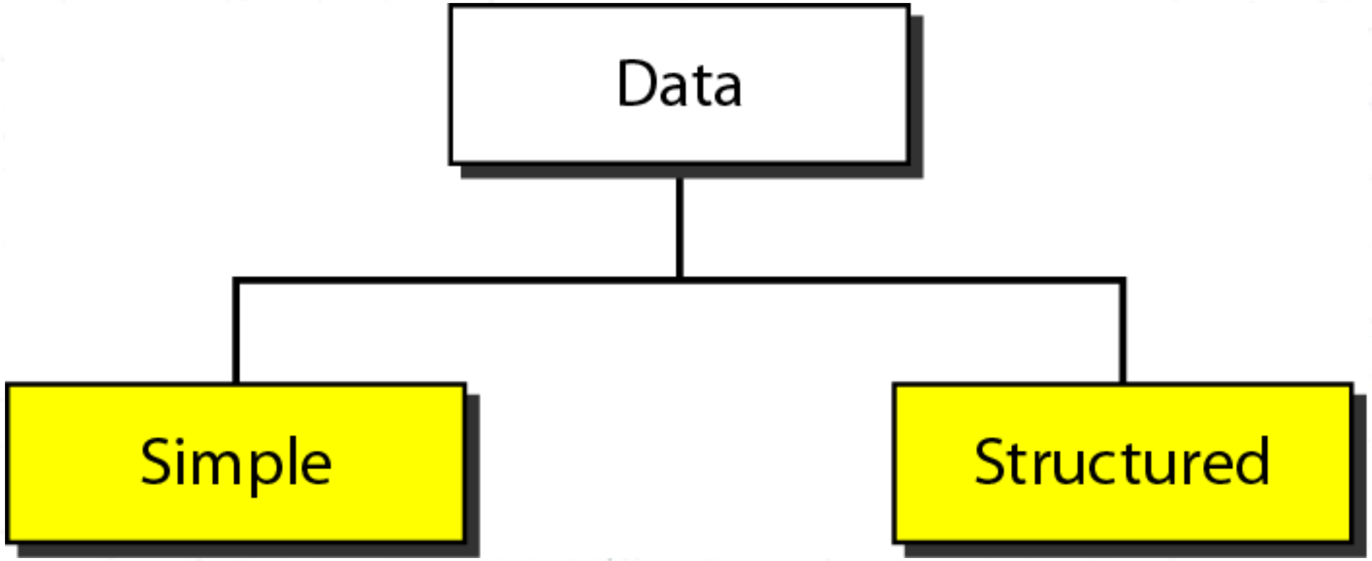
Structure of Management Information (SMI)

- Adapted subset of **ASN.1**
- Structure of Management Information (SMI) is used to define the objects in MIBs.
  - Module definitions
    - MODULE-IDENTITY
  - Object definitions
    - OBJECT-TYPE
  - Notification definitions
    - NOTIFICATION-TYPE

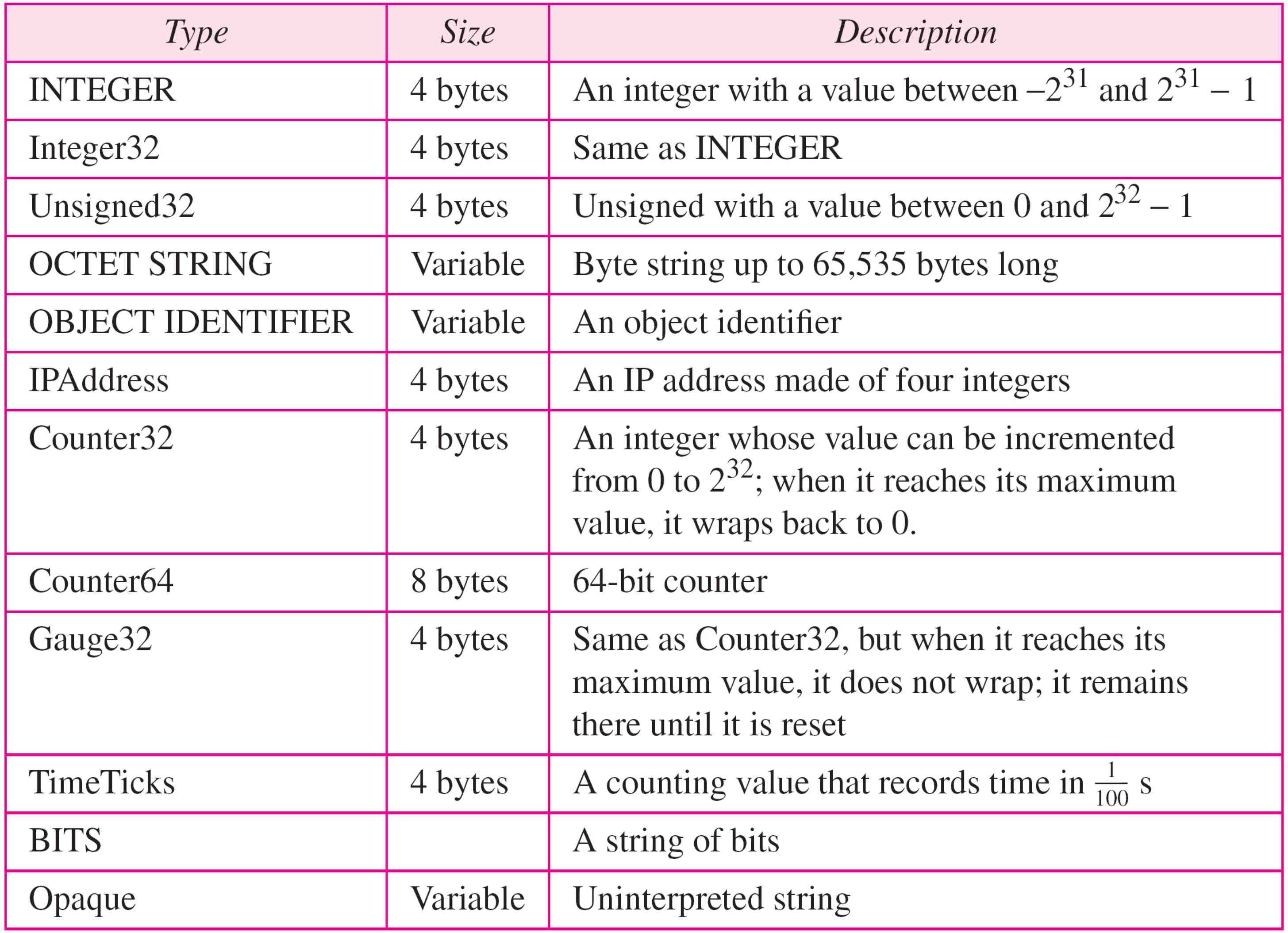
ASN.1

- Abstract syntax notation one
- Formal notation for describing data structures and message formats
- Type definitions, value definitions, combined
- Predefined basic types
  - BOOLEAN, INTEGER, OCTET STRING, BIT STRING, REAL, ENUMERATED, CHARACTER STRING, OBJECT IDENTIFIER
- Constructed types
  - SEQUENCE, SEQUENCE OF, CHOICE
  - Arbitrary nesting of types and sub-types
- Encoding

data type

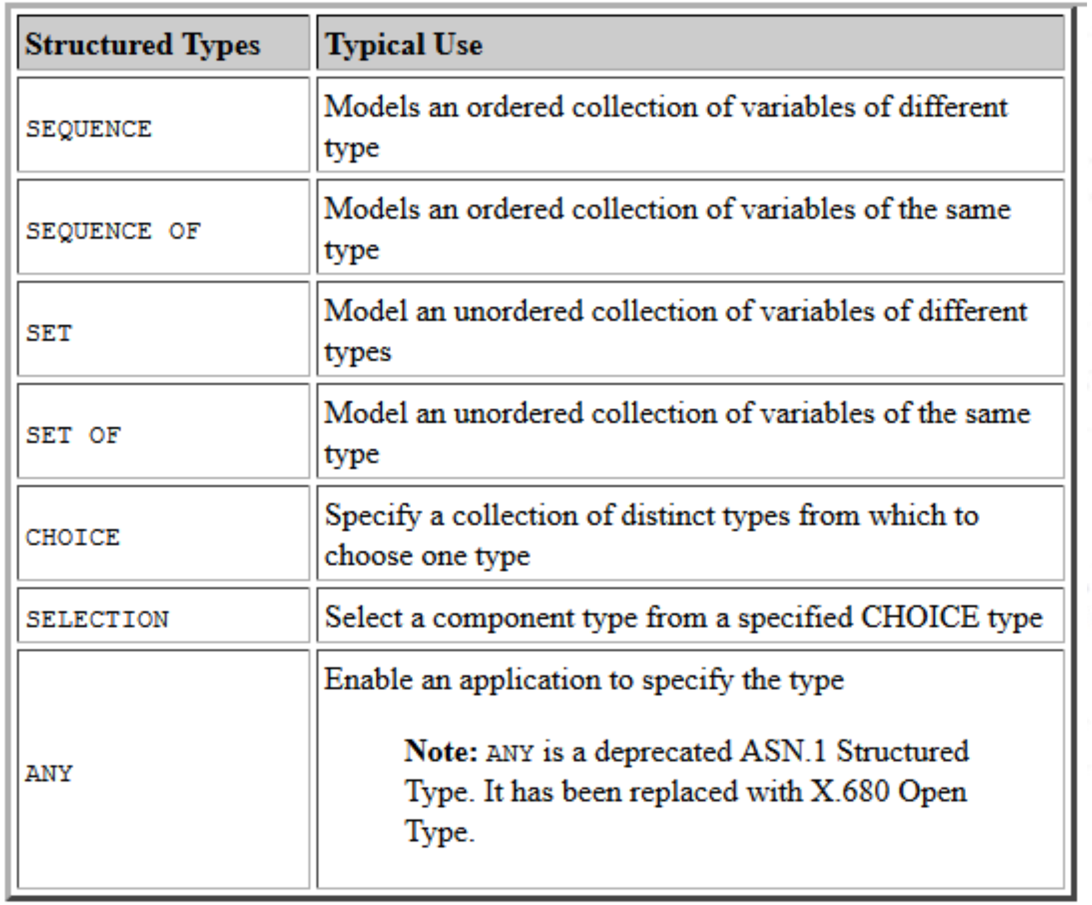


Simple type



Type	*Size*	*Description*
INTEGER	4 bytes	An integer with a value between -2^{31} and $2^{31} - 1$
Integer32	4 bytes	Same as INTEGER
Unsigned32	4 bytes	Unsigned with a value between 0 and $2^{32} - 1$
OCTET STRING	Variable	Byte string up to 65,535 bytes long
OBJECT IDENTIFIER	Variable	An object identifier
IPAddress	4 bytes	An IP address made of four integers
Counter32	4 bytes	An integer whose value can be incremented from 0 to 2^{32}; when it reaches its maximum value, it wraps back to 0.
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter32, but when it reaches its maximum value, it does not wrap; it remains there until it is reset
TimeTicks	4 bytes	A counting value that records time in $\frac{1}{100}$ s
BITS		A string of bits
Opaque	Variable	Uninterpreted string

Structured Types



Structured Types	Typical Use
SEQUENCE	Models an ordered collection of variables of different type
SEQUENCE OF	Models an ordered collection of variables of the same type
SET	Model an unordered collection of variables of different types
SET OF	Model an unordered collection of variables of the same type
CHOICE	Specify a collection of distinct types from which to choose one type
SELECTION	Select a component type from a specified CHOICE type
ANY	Enable an application to specify the type **Note:** ANY is a deprecated ASN.1 Structured Type. It has been replaced with X.680 Open Type.

ASN.1 types and values

- **Type definitions**
  - `NumberofStudents ::= INTEGER`
  - `PassorFail ::= BOOLEAN`
  - `GradeType ::= ENUMERATED {A, B, C}`
  - `PointsScored ::= REAL`
  - `Image ::= BIT STRING`
  - `Data ::= OCTET STRING`
- **Value definitions**
  - `studentsMonaySession NumberofStudents ::= 9`
  - `NDMCourse PassorFail ::= TRUE`
  - `NumberofStudents ::= 10`

- **Combine type value definitions**
  - StudentType

    ```
    StudentType ::= INTEGER {
      ugrad  (0)
      ms     (1)
      phd    (2)
    }
    ```
  - `NumberofStudents ::= 10`

ASN.1 structured types and values

-
  ```
  StudentRecord ::= SEQUENCE {
      regNo      INTEGER,
      numClasses      INTEGER OPTIONAL,
      fathersName     STRING
      }
  ```

  type definition

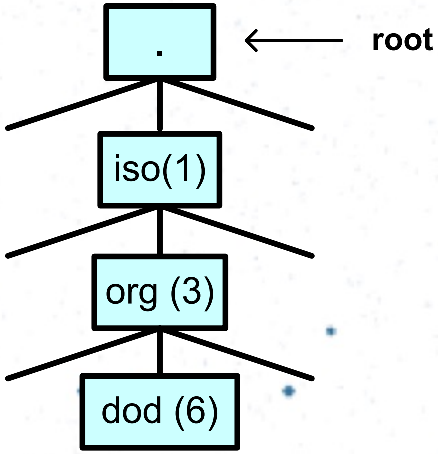
-
  ```
  John StudentRecord ::= {
      regNo      1234,
      numClasses      5,
      fathersName     Don
  }
  ```

  value definition

- `studentNo ::= SEQUENCE OF regNo    (type definition)`
- `studentNo ::= {1234, 5678, 9012}   (value definition)`

ASN.1 OBJECT IDENTIFIER (MIB)

- Define an information object that is managed at the international level
- `internet OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) }`



SLIIT FACULTY OF COMPUTING

ASN.1 MACRO (MIB)

provide the capability of defining types and values that are not included in the standard repertoire.

```
OBJECT-TYPE MACRO ::=
   BEGIN
      TYPE NOTATION ::= "SYNTAX" type (TYPE
  ObjectSyntax)
                     "ACCESS" Access
                     "STATUS" Status
      VALUE NOTATION ::= value (VALUE ObjectName)
```

```
Access ::= "read-only"
                | "read-write"
                | "write-only"
                | "not-
  accessible"
     Status ::= "mandatory“
                | “current"
                | "optional"
                | "obsolete"
   END
```

ASN.1 Encoding

- ASN.1 defines syntax and not how to encode them
- ASN.1 encoding rules
  - Basic encoding rules (BER) (will be discuss separately)
  - DER encoding rules (DER)
  - Canonical encoding rules (CER)
  - XML encoding rules (XER)
  - Packet encoding rules (PER)
  - Generic string encoding rules (GSER)

Management Information Base (MIB)

MIB

- A MIB specifies the managed objects
- MIB is a text file that describes managed objects using the syntax of ASN.1
- What is a managed object?
  - interface, TCP stack (RTO, congestion control alg.), ARP etc.
- In Linux, MIB files are in the directory */usr/share/snmp/mibs*
  - Multiple MIB files
  - MIB-II (defined in RFC 1213) defines the managed objects of TCP/IP networks

Managed Objects

- Each managed object is assigned an *object identifier (OID)*
- The OID is specified in a MIB file.
- An OID can be represented as a sequence of integers separated by decimal points or by a text string:

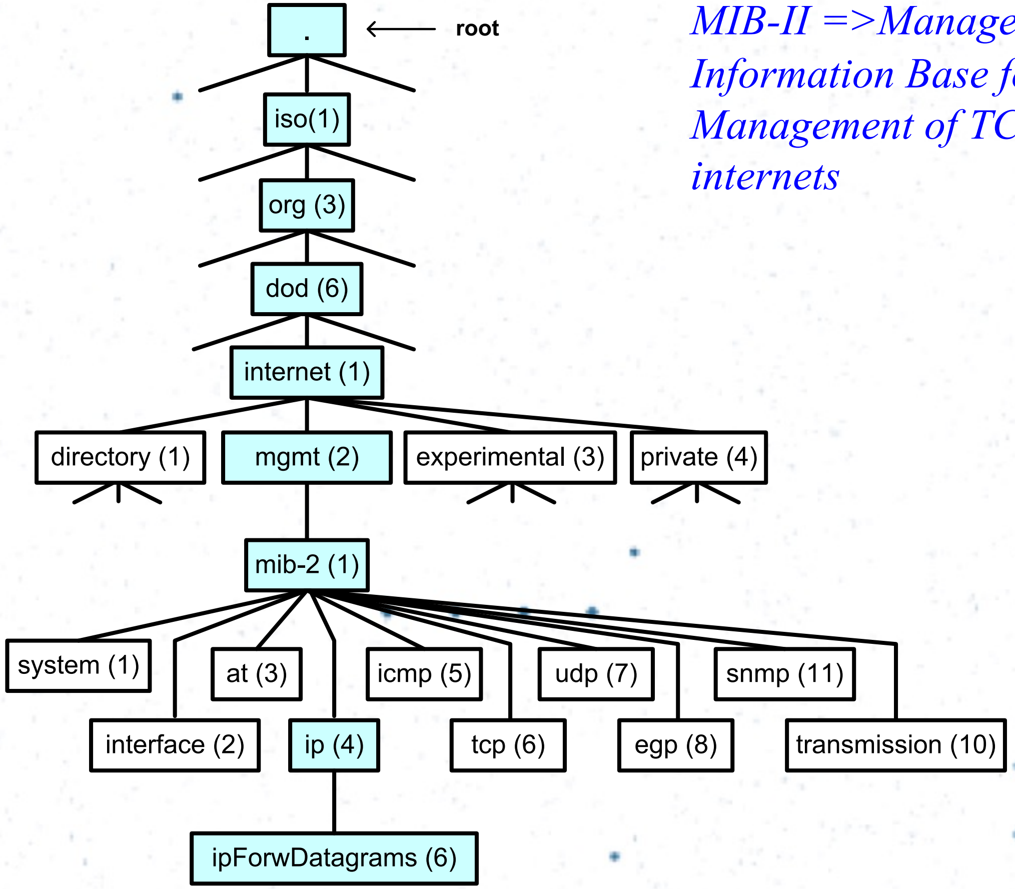
Example:

- *1.3.6.1.2.1.4.6.*
- *iso.org.dod.internet.mgmt.mib-2.ip.ipForwDatagrams*

- When an SNMP manager requests an object, it sends the OID to the SNMP agent.

Organization of managed objects

- Organized in a tree-like hierarchy
- OIDs reflect the structure of the hierarchy.
- Each OID represents a node in the tree.
- The OID 1.3.6.1.2.1 (*iso.org.dod.internet.mgmt.mib-2)* is at the top of the hierarchy for all managed objects of the MIB-II.
- Manufacturers of networking equipment can add product specific objects to the hierarchy.

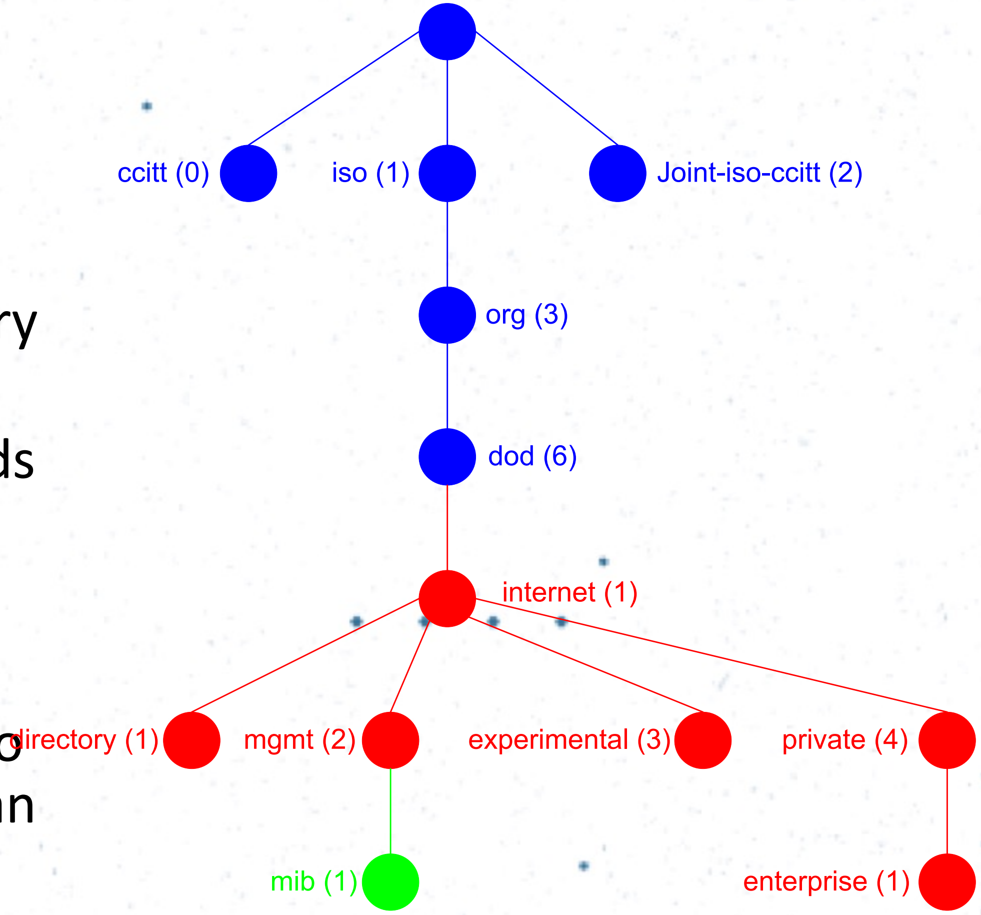


MIB-II =>Management Information Base for Network Management of TCP/IP-based internets

Organization of managed objects

RFC 1155 defines top of the administrative domain **managed by the IETF**:

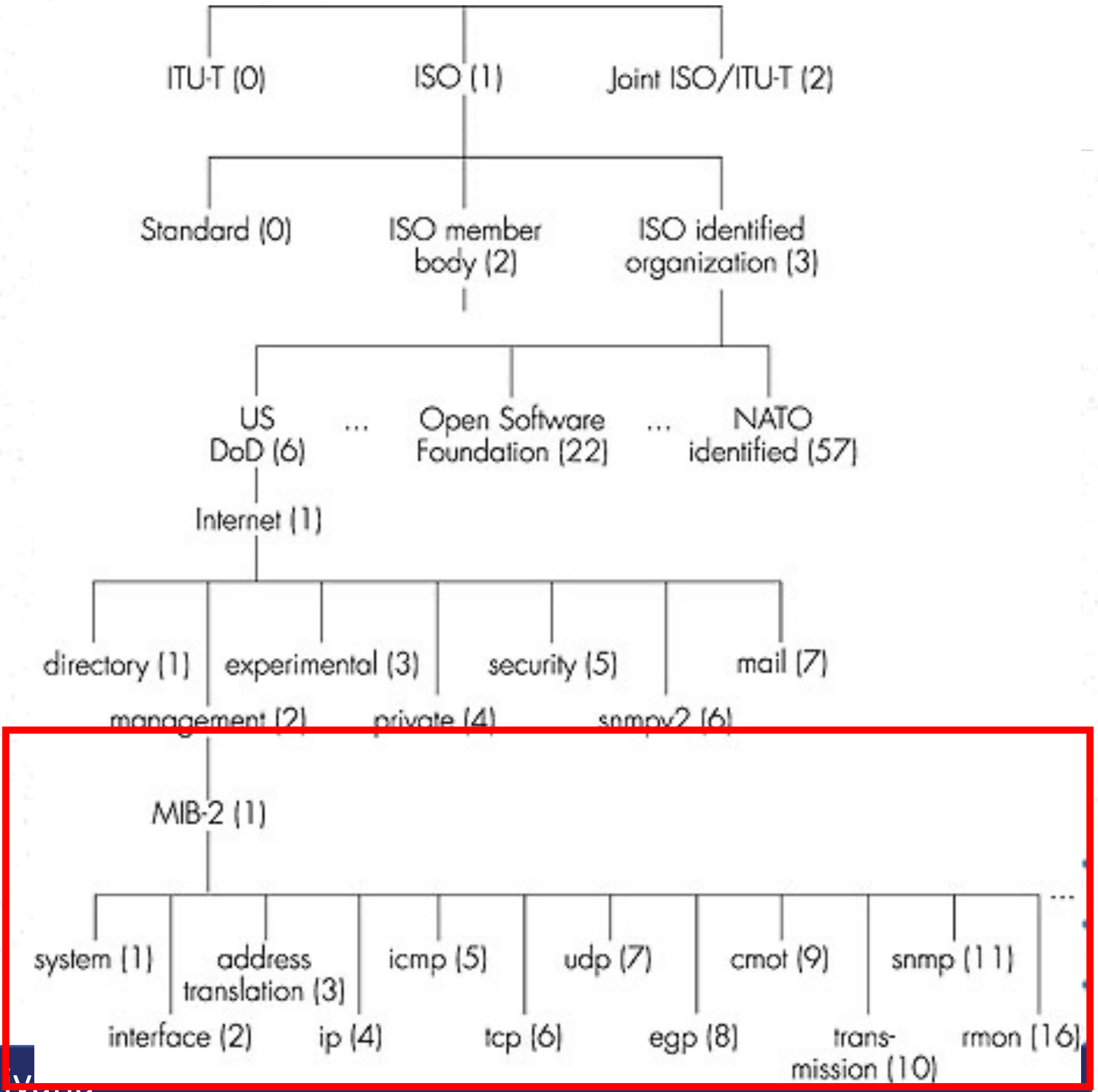
- 1.3.6.1 - Internet
  - ❑ Directory – reserved for use with a future memo that discusses how the OSI directory may be used in the Internet.
  - ❑ Mgmt – area for items defined in standards track documents.
  - ❑ Experimental – area for IETF experimental items.
  - ❑ Private – area for delegation of subtrees to enterprises, that is, anyone who asks for an enterprise number.



MIB-II (RFC1213)

MIB-II defines **11 separate groups** :

- ❑ system (1)
- ❑ interfaces (2)
- ❑ address translation (3)
- ❑ ip (4)
- ❑ icmp (5)
- ❑ tcp (6)
- ❑ udp (7)
- ❑ egp (8)
- ❑ cmot (9) CMIS over TCP (for historic reasons only)
- ❑ transmission (10)
- ❑ snmp (11)

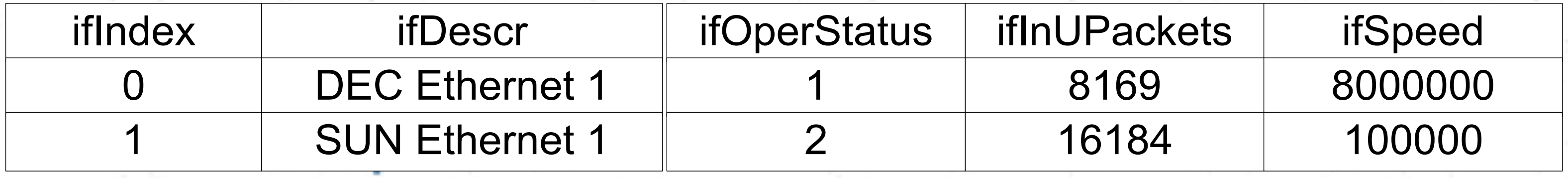


System Group

- Contains data pertaining to the system where the agent is residing in.
- Fault management objects:
  - SysObjectID – System manufacturer.
  - sysServices – Protocol layers that device services, using formula $2^{(L-1)}$.
    - e.g. host that runs transport + application layer services.
    - $2^{(4-1)} + 2^{(7-1)} = 72$.
  - sysUptime – Amount of time system has been operational.
- Configuration management objects:
  - sysDescr – Description of the system.
  - sysLocation – System's physical location.
  - sysContact – System's name.

Interfaces Group

- The interfaces group provides information pertaining to each specific network interface (ifTable).
- Useful for configuration, performance, fault and accounting management.
- ifNumber - number of interfaces.
- ifTable example:



ifIndex	ifDescr	ifOperStatus	ifInUPackets	ifSpeed
0	DEC Ethernet 1	1	8169	8000000
1	SUN Ethernet 1	2	16184	100000

Interfaces Group cont.

Example – Determining Utilization

Total bytes = $(\text{ifInOctects}_y - \text{ifInOctects}_x) +$
$(\text{ifOutOctects}_y - \text{ifOutOctects}_x)$

Total bytes per sec = Total bytes / (y-x)

Utilization = (Total bytes per sec * 8) / ifSpeed

IP Group

- Provides information about the IP layer in a systems network protocol stack.
  - ❑ Information pertaining to errors and types of packets seen.
  - ❑ Routing table (i.e. ipRouteTable).
- Configuration/Fault management objects:
  - ❑ ipForwarding – If device is set up to route IP packets.
  - ❑ ipAddrTable – Addresses on the device.
  - ❑ ipRouteTable – Routing table.
- Performance management objects:
  - ❑ ipInDiscards – Rate of input datagrams discarded.
  - ❑ ipInHdrErrors – Rate of input header errors.
  - ❑ ipInAddrErrors – Rate of input address errors.
- Accounting management objects:
  - ❑ ipOutRequests – Number of IP datagrams sent.
  - ❑ ipInDelivers – Number of IP datagrams received.

ICMP Group

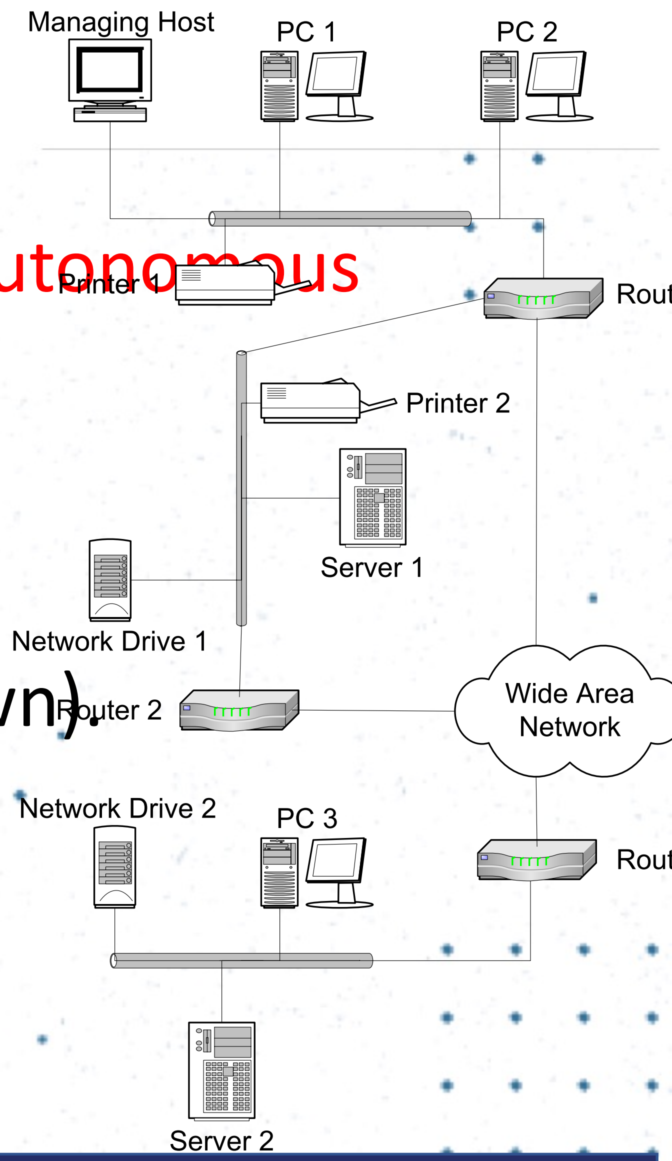
- Provides information pertaining to systems ICMP entity.
  - ❑ icmpInRedirects – rate of redirect messages received.
  - ❑ icmpOutDestUnreachs – rate of destination unreachable errors sent.
  - ❑ icmpInSrcQuenchs – input rate of source quench messages (source quench msgs – requests that the sender decrease the rate of messages sent to a router or host).
  - ❑ icmpOutEchos – rate of output echo messages.
- Mainly useful for performance management.

TCP Group

- TCP – The Transmission Control Protocol (TCP) provides reliable transport services between applications (UDP is the unreliable transport service).
- Configuration management objects:
  - ❑ tcpRtoAlgorithm – Retransmission algorithm.
  - ❑ tcpConnTable – Connection table (i.e. netstat).
- Performance management objects:
  - ❑ tcpAttemptFails – Number of failed attempts to make a connection.
  - ❑ tcpEstabResets – Number of resets in established connections.
- Accounting management objects:
  - ❑ tcpActiveOpens – Number of times this system has opened a connection.
  - ❑ tcpInSegs – Number of TCP segments received.
- Security management objects:
  - ❑ tcpConnTable – Connection table (i.e. netstat).
- The User Datagram Protocol (UDP) group provides similar information. (e.g. udpTable)

EGP Group

- EGP (RFC 904) is a protocol that tests for the reachability of IP networks.
  - ❑An IP network can be divided into networks of autonomous systems.
- egpNeighTable – information about this entity's EGP neighbors.
- Fault management objects:
  - ❑egpNeighState – state of EGP neighbour (up,down).
- Configuration management objects:
  - ❑egpIntervalHello – hello message interval.
  - ❑egpAs – local EGP autonomous system.



Transmission Group

- Reserved for information pertaining to specific media underlying the interfaces of a system.

- Various RFCs:
  - ❑RFC 1512 FDDI.
  - ❑RFC 1493 Bridge.
  - ❑RFC 1743 Token Ring.

SNMP Group

- **Management protocols also need to be managed…!!!**
  - ❑Useful to all 5 areas of network management.
- Fault management objects:
  - ❑snmpInASNParseErrrors – Number of malformed SNMP messages.
  - ❑snmpInNoSuchNames – Number of requests to invalid objects.
- Configuration management objects:
  - ❑EnableAuthenTraps – Enables entity to send traps when authentication errors occur.
- Performance/Accounting management objects:
  - ❑snmpInPkts – Rate of SNMP packets input.
  - ❑snmpInTraps – Rate of traps input.
- Security management objects:
  - ❑snmpInBadCommunityNames – Number of authentication failures.
  - ❑snmpInBadCommunityUses – Number of requests without sufficient privileges.

Definition of managed objects in a MIB

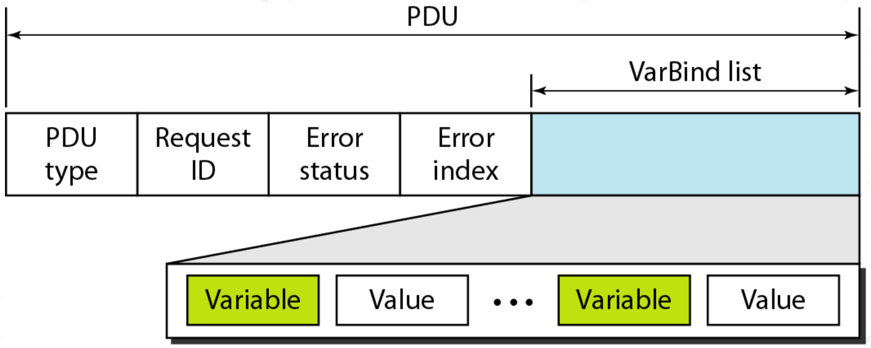
- Specification of ipForwDatagrams in MIB-II.

```
ipForwDatagrams OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The number of input datagrams for which this
            entity was not their final IP destination, as a
            result of which an attempt was made to find a
            route to forward them to that final destination.
            In entities which do not act as IP Gateways, this
            counter will include only those packets which were
            Source-Routed via this entity, and the Source-
            Route option processing was successful."
    ::= { ip 6 }
```

BER encoding

Format of a BER encoded message

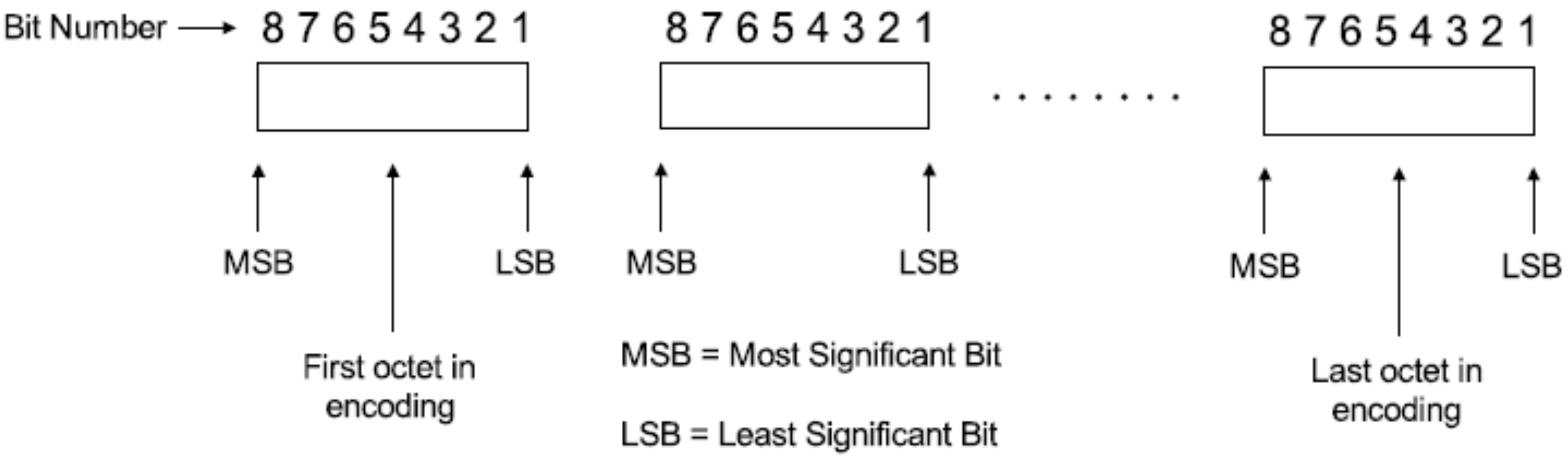
- In BER encoding, the most fundamental rule states that each field is encoded in three parts:
  - ❑ Type (or Tag)
  - ❑ Length
  - ❑ Value
- Hence, this is also known as TLV encoding.



Type	Length	Value

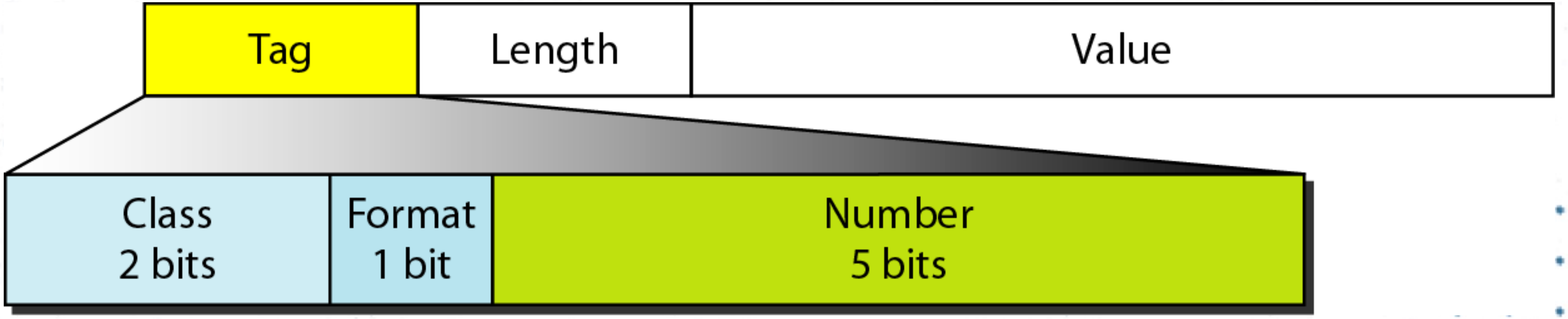
Format of a BER encoded message cont..

- Like in most protocols, in BER also the Most Significant Bit (MSB)/Octet is encoded on the left.



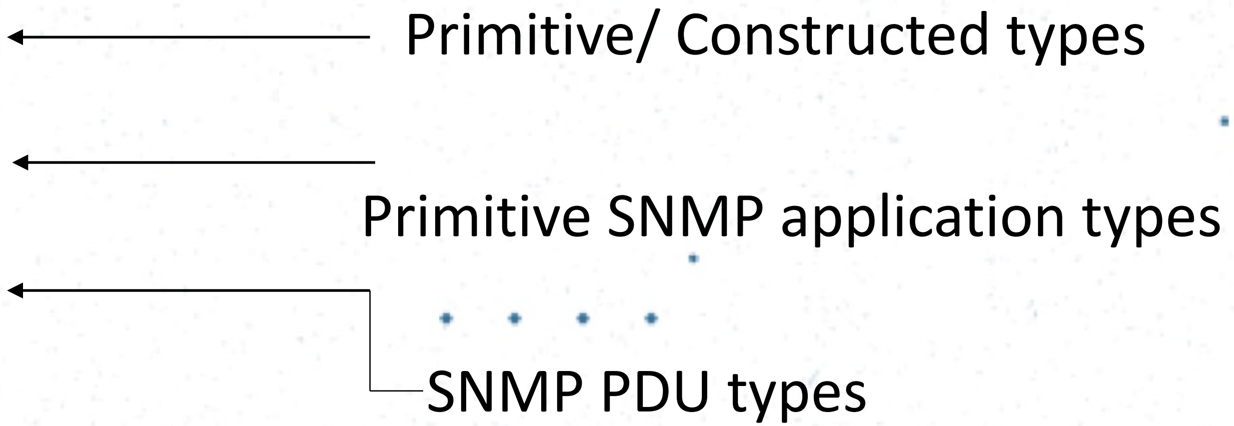
Type/Tag format

- Every BER encoded ASN.1 data type has a type field.
- This type can be one of four classes which is indicated by the first two bits of the type octet.



Data Type : class

Class	**Bit 8**	**Bit 7**
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1



Primitive/ Constructed types

Primitive SNMP application types

SNMP PDU types

Class 2 bits	Format 1 bit	Number 5 bits

ASN.1 Primitive Types

Data Type	Class	Format	Number	Type/Tag (Binary)	Type/Tag (Hex)
BOOLEAN	00	0	00001	00000001	01
INTEGER	00	0	00010	00000010	02
BIT STRING	00	0	00011	00000011	03
OCTET STRING	00	0	00100	00000100	04
NULL	00	0	00101	00000101	05
OBJECT IDENTIFIER	00	0	00110	00000110	06

Class	Format	Number
2 bits	1 bit	5 bits

ASN.1 Constructed Types

Data Type	Class	Format	Number	Type/Tag (Binary)	Type/Tag (Hex)
SEQUENCE and SEQUENCE OF	00	1	10000	00110000	30

Class 2 bits	Format 1 bit	Number 5 bits

Primitive SNMP Application Types

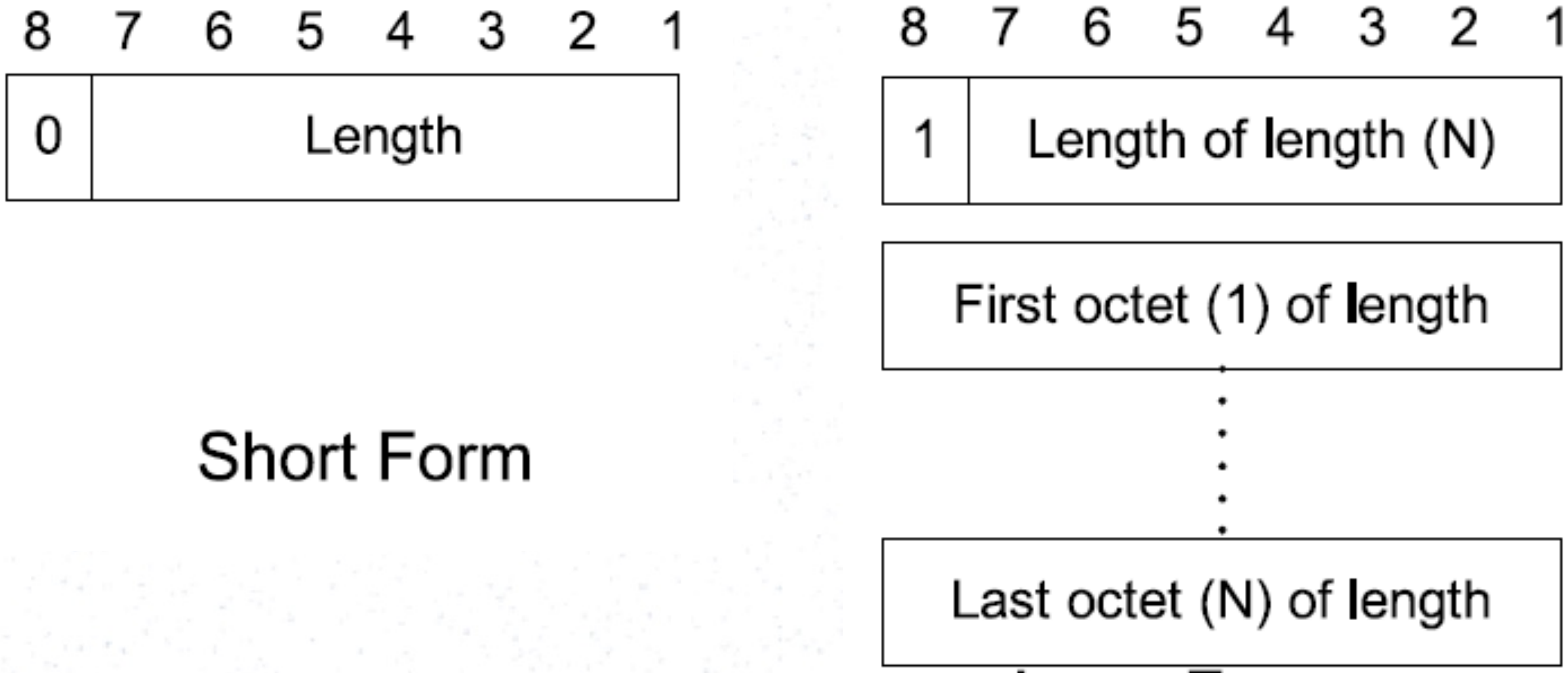
Data Type	Class	Format	Number	Type/Tag (Binary)	Type/Tag (Hex)
IpAddress	01	0	00000	01000000	40
Counter (Counter32)	01	0	00001	01000001	41
Gauge (Gauge32)	01	0	00010	01000010	42
TimerTicks	01	0	00011	01000011	43
Opaque	01	0	00100	01000100	44
NsapAddress	01	0	00101	01000101	45
Counter64	01	0	00110	01000110	46
Uinteger32	01	0	00111	01000111	47

Class 2 bits	Format 1 bit	Number 5 bits

Context Specific SNMP PDU Types

Data Type	Class	Format	Number	Type/Tag (Binary)	Type/Tag (Hex)
GetRequest	10	1	00000	10100000	A0
GetNextRequest	10	1	00001	10100001	A1
Get/Response	10	1	00010	10100010	A2
SetRequest	10	1	00011	10100011	A3
Trap	10	1	00100	10100100	A4
GetBulkRequest	10	1	00101	10100101	A5
InformRequest	10	1	00110	10100110	A6
SNMPv2 Trap	10	1	00111	10100111	A7

Length format



8 7 6 5 4 3 2 1

0 | Length

Short Form

8 7 6 5 4 3 2 1

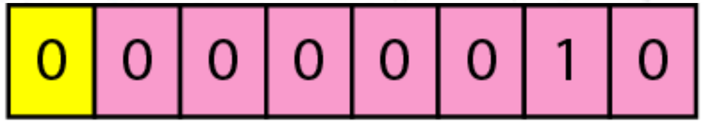
1 | Length of length (N)

First octet (1) of length

Last octet (N) of length

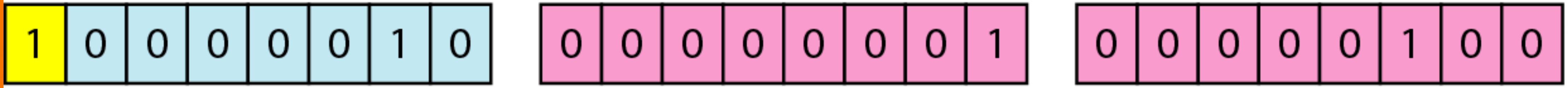
Long Form

Length format



0 0 0 0 0 0 1 0

a. The colored part defines the length (2).

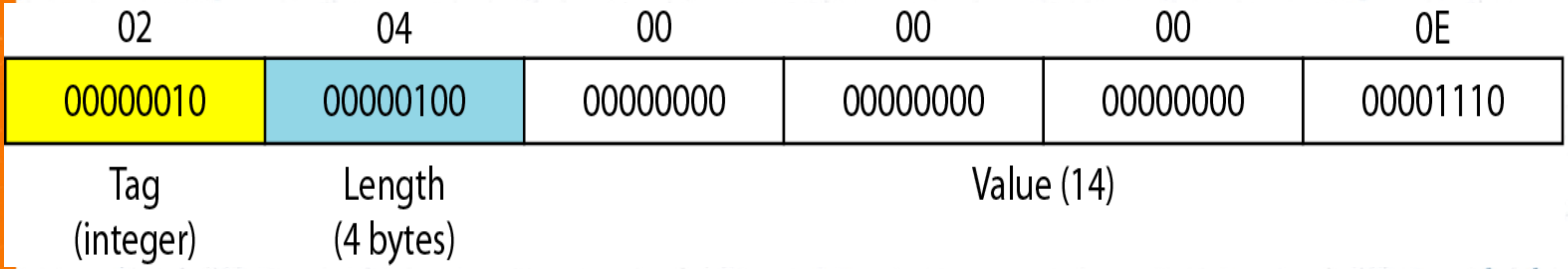


1 0 0 0 0 0 1 0   0 0 0 0 0 0 0 1   0 0 0 0 0 1 0 0

b. The shaded part defines the length of the length (2 bytes);
the colored bytes define the length (260 bytes).

Value Format

- How to define an integer value;
- *integer INTEGER ::= 14*



02	04	00	00	00	0E
00000010	00000100	00000000	00000000	00000000	00001110
Tag (integer)	Length (4 bytes)	Value (14)			

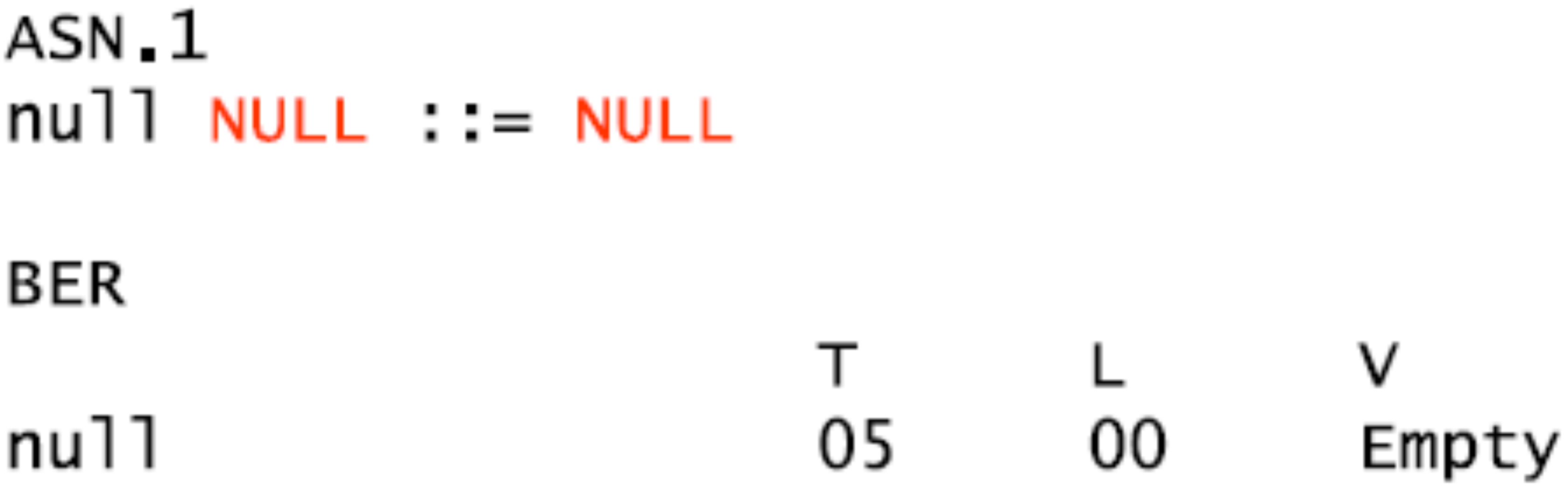
Value Format

- How to define a string value;
- *Octetstring OCTECT STRING ::= ‘HI’*

04	02	48	49
00000100	00000010	01001000	01001001
Tag (String)	Length (2 bytes)	Value (H)	Value (I)

Value Format

- How to define a null value;

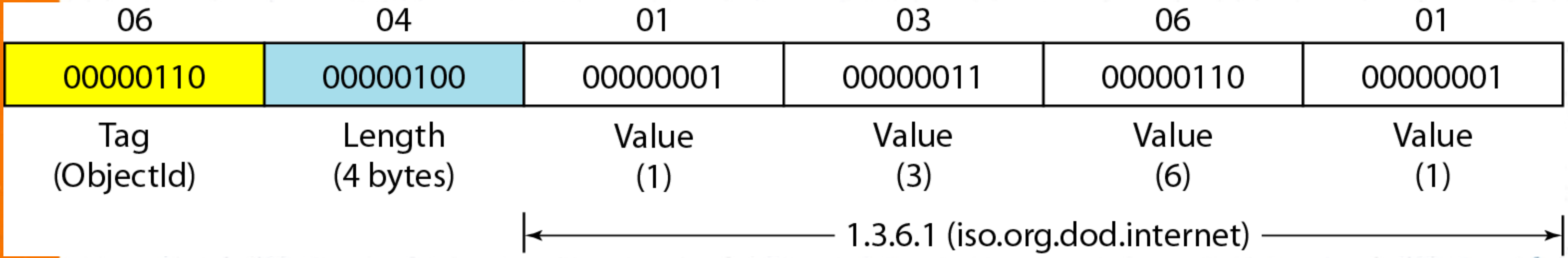


```
ASN.1
null NULL ::= NULL

BER
                  T       L       V
null              05      00      Empty
```

Value Format

- *How to define ObjectIdentifier 1.3.6.1 (iso.org.dod.internet).*



06	04	01	03	06	01
00000110	00000100	00000001	00000011	00000110	00000001
Tag (ObjectId)	Length (4 bytes)	Value (1)	Value (3)	Value (6)	Value (1)

1.3.6.1 (iso.org.dod.internet)

Encoding OBJECT IDENTIFIER

- Two rules apply when encoding OIDs using BER.
- The first rule states that, the first two numbers ‘x.y’ of the OID are encoded as a single value using the formula (40*x)+y.
- The first two numbers of any SNMP related OID is always 1.3. Therefore the first two numbers of an SNMP related OID is always encoded as 43 or 0x2B, because (40*1)+3 = 43.

Encoding OBJECT IDENTIFIER

- Second rule applies when encoding large numbers in OIDs that cannot be represented using one octet (i.e. one byte or 8 bits).
- For example, the OID 1.3.6.1.4.1.2680.1.2.7.3.2 contains 2680 which cannot be encoded using a single octet (since 8 bits can only represent 0-255).
- The rule indicates that, when encoding large numbers in OIDs, only the lower 7 bits of the octet are used for holding the actual value (0-127). The highest order bit is used as a flag to indicate that this number spans more than one byte.

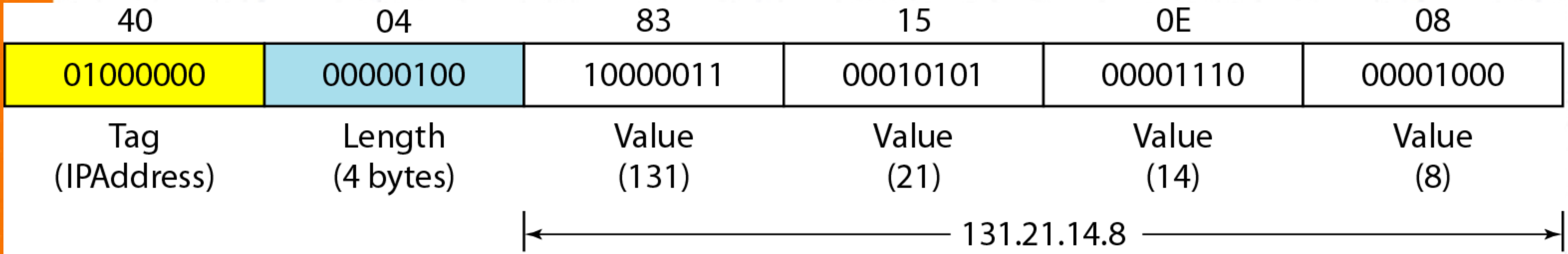
Encoding OBJECT IDENTIFIER

- According to the rule discussed for large numbers in OIDs, value of 2680 will be encoded as 0x8A 0x78.
- And the fully BER encoded value for OID 1.3.6.1.4.1.2680.1.2.7.3.2 will be,

T	L	V
06	0C	2B 06 01 04 01 8A 78 01 02 07 03 02

Value Format

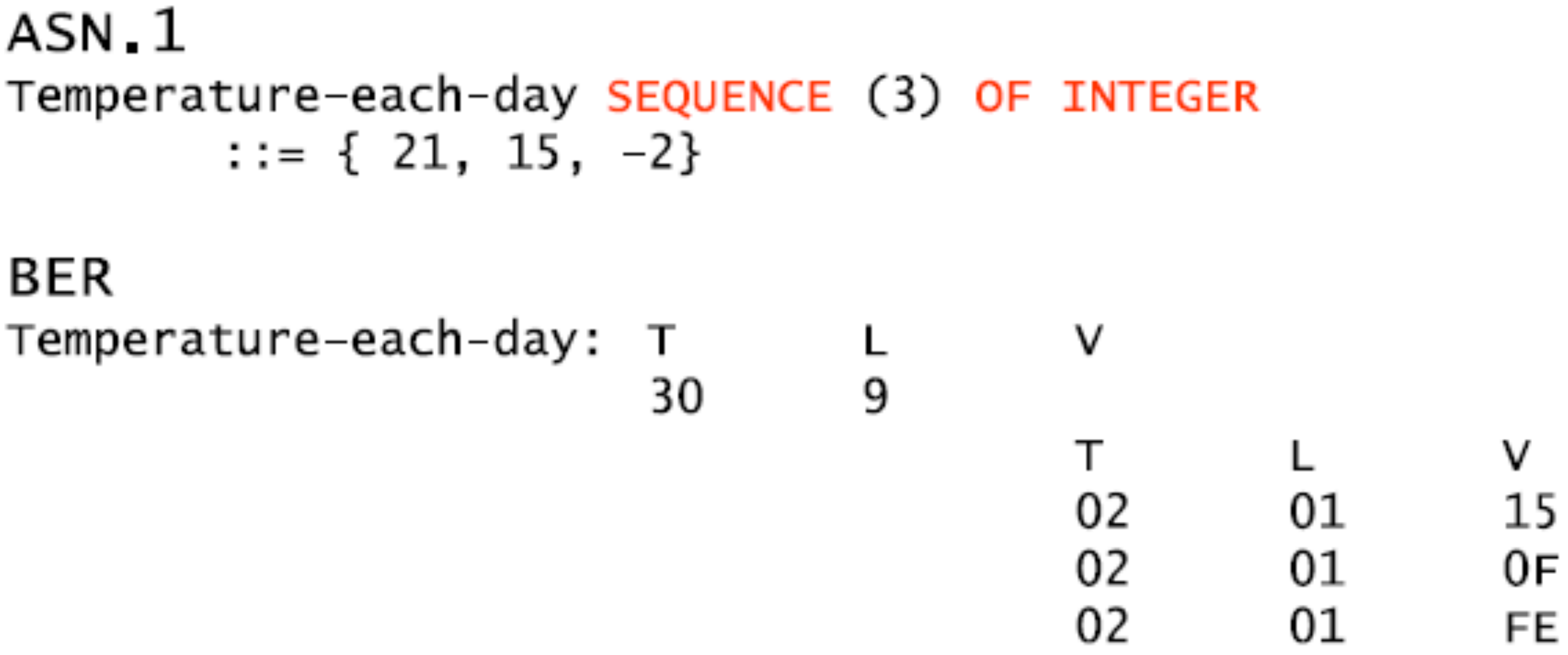
- *How to define IPAddress 131.21.14.8*



40	04	83	15	0E	08
01000000	00000100	10000011	00010101	00001110	00001000
Tag (IPAddress)	Length (4 bytes)	Value (131)	Value (21)	Value (14)	Value (8)
		131.21.14.8			

Value Format

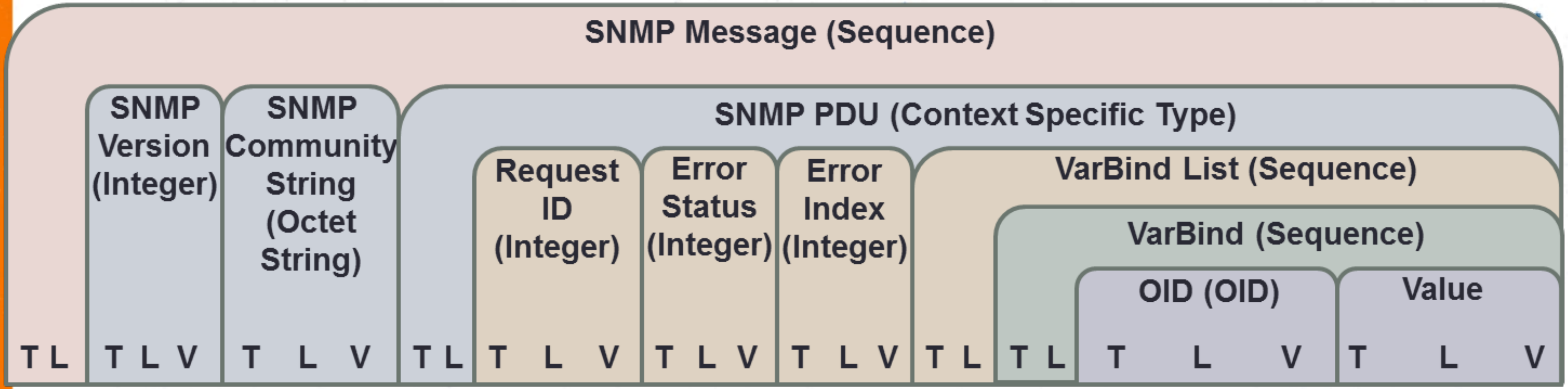
- *How to define SEQUENCE or SEQUENCE OF*



```
ASN.1
Temperature-each-day SEQUENCE (3) OF INTEGER
          ::= { 21, 15, -2}

BER
Temperature-each-day:  T        L        V
                       30       9
                                         T         L         V
                                         02        01        15
                                         02        01        0F
                                         02        01        FE
```

BER encoded format of a SNMP message



You must remember this format…!!!

How to organize the decoded values

- Tag (SNMP Message), Length,
  - Tag, Length, Version
  - Tag, Length, Community
  - Tag (PDU Type), Length
    - Tag, Length, Request ID
    - Tag, Length, Error Status
    - Tag, Length, Error Index
    - Tag (VarBind List), Length
      - Tag (VarBind), Length
        - Tag, Length, OID
        - Tag, Length, Value

Let’s try an example

```
30 2B 02 01 00 04 08 53 65 63 75 72 69 74 79 A1 1C 02 04 3B 0B 16 36 02 01 00 02
01 00 30 0E 30 0C 06 08 2B 06 01 02 01 01 02 00 05 00

30 2B                                  => SEQUENCE
   02 01 00                            => INTEGER 0
   04 08 53 65 63 75 72 69 74 79       => OCTET STRING Security
   A1 1C                               => Context Specific 1 Constructor
      02 04 3B 0B 16 36                => INTEGER 3B0B1636
      02 01 00                         => INTEGER 0
      02 01 00                         => INTEGER 0
      30 0E                            => SEQUENCE
         30 0C                         => SEQUENCE
            06 08 2B 06 01 02 01 01 02 00 => OBJECT IDENTIFIER
            05 00                              => NULL
```

Example cont..

SEQUENCE => SNMP Message

INTEGER 0 => Version 0 (SNMPV1)

OCTET STRING Security => Community String

Context Specific 1 Constructor => GetNextRequest

INTEGER 3B0B1636 => Request ID (aka Sequence #)

INTEGER 0 => Error Status

INTEGER 0 => Error Index

SEQUENCE => VarBind List

SEQUENCE => VarBind

OBJECT IDENTIFIER => 1.3.6.1.2.1.1.2.0

NULL => Empty value

Try this by yourself…

- 30 2F 02 01 00 04 08 53 65 63 75 72 69 74 79 A2 20 02 04 3B 0B 16 36 02 01 00 02 01 00 30 12 30 10 06 08 2B 06 01 02 01 01 03 00 43 04 1B E1 55 80

~ THE END ~

SNMP Operations

SET Request

Initializes or changes the value of a network element.

GET Request

Sent by manager requesting data from agent.

GETNEXT Request

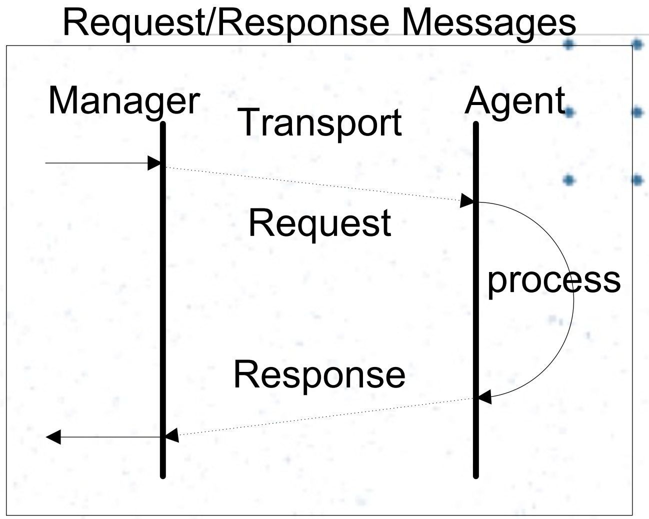
Sent by manager requesting data on the next managed object to the one specified.

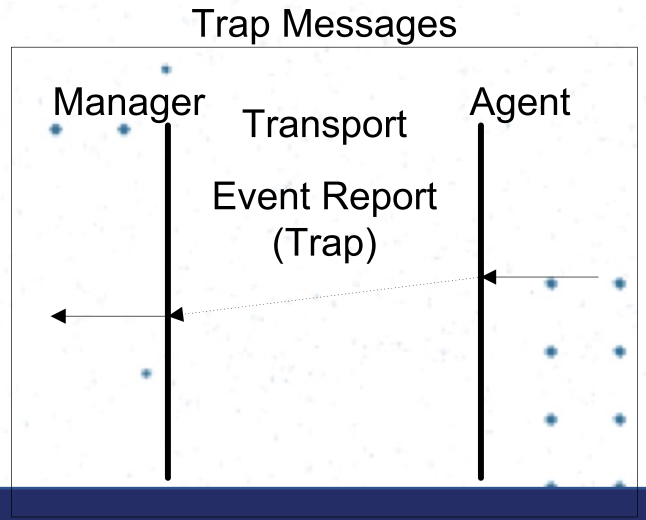
GET Response

Used by the agent to respond with data to get and set requests from the manager.

Trap

Alarm generated by agent.

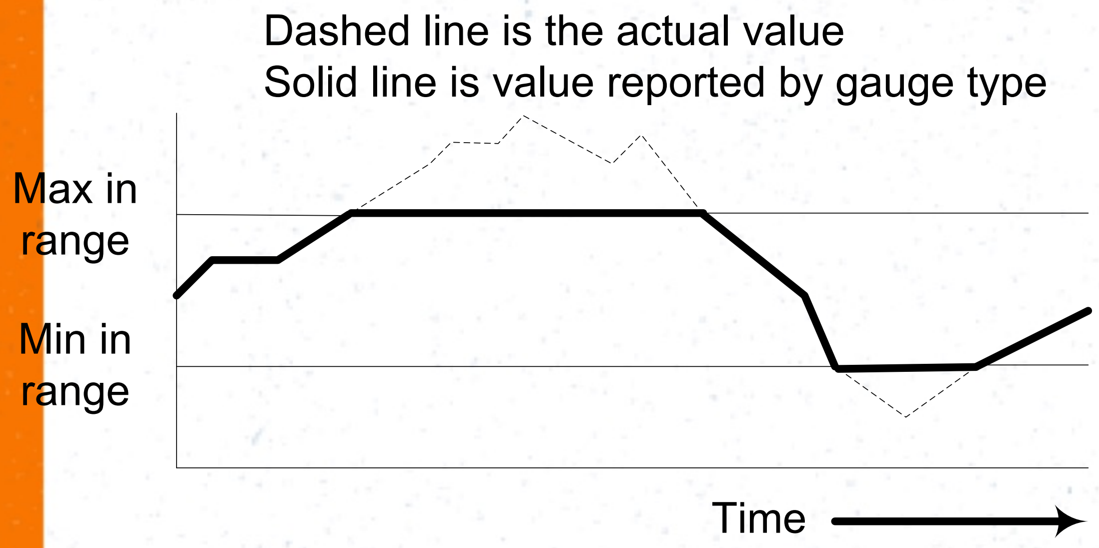




SMI (RFC1155) Defined Data Types

- Integer – Signed 32-bit integer.
  - ❑ Enumerated Integer.
- Octet String – String of bytes.
- Object Identifier.
- NetworkAddress – An address from one of possibly several protocol families.
- IpAddress – 32 bit IP address.
- Gauge – Non-negative integer from 0 to $2^{32} – 1$, which may increase or decrease.
- Counter – Non-negative monotonically increasing integer from 0 to $2^{32} – 1$.
- Timeticks – Non-negative integer which counts time in hundredths of a second.
- Opaque – Arbitrary syntax.

Gauge

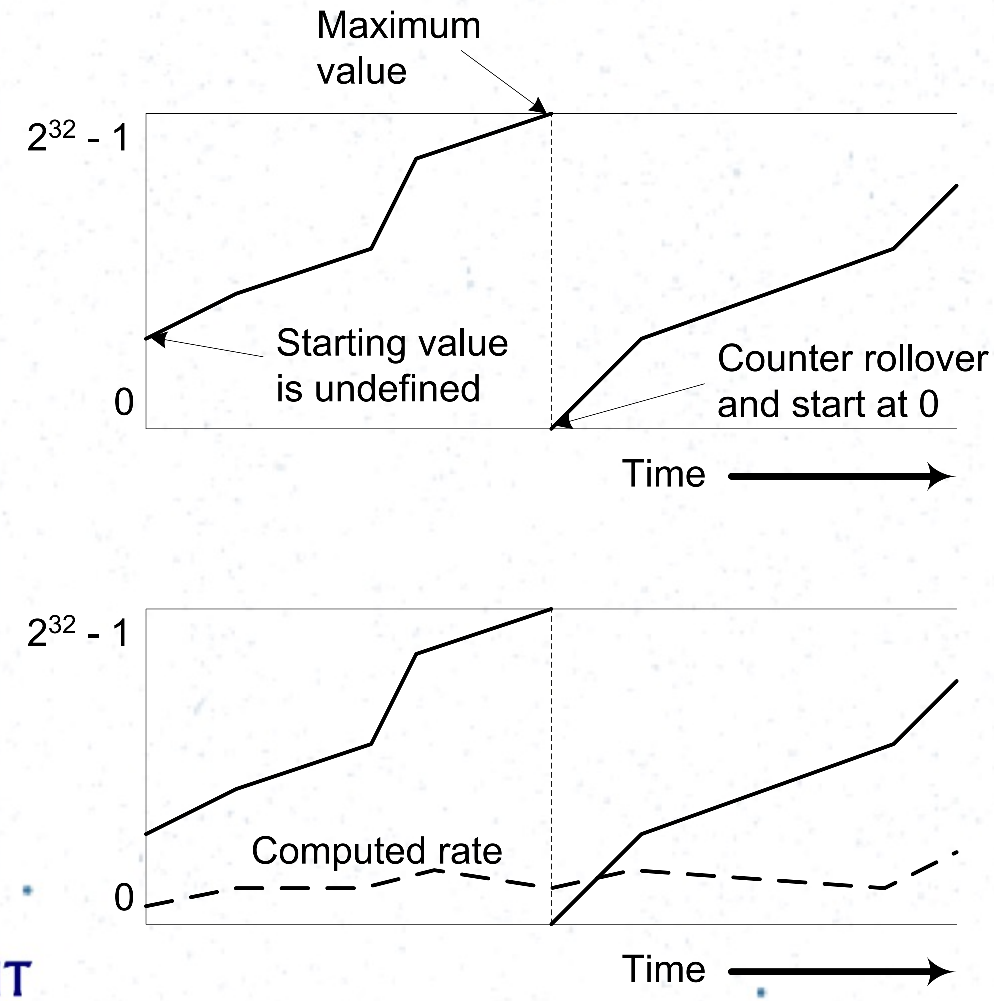


Dashed line is the actual value
Solid line is value reported by gauge type
Max in range
Min in range
Time

- Used to specify a value whose range includes only non-negative 32 bit integers.

- RFC 1155 – *this application-wide type represents a non-negative integer, which may increase or decrease, but which latches at a maximum value.*

Counter



- Use to specify a non-negative value whose range includes only positive 32-bit integers.
- Values reported by counters are not absolute, since the count is not required to start at 0 and the count may roll over.
- Counters are used by obtaining a value v_0 at t_o and then later obtaining a value v_1 at t_1.
  - □Difference between v_0 and v_1 is the count over the time period.
  - □Counter rollover can be detected iff $v_0 > v_1$.
  - □A periodic sampling of

Other Types

- Timeticks – used to specify a non-negative value whose range includes only non-negative integers.
  - ❑ Units are in hundredths of seconds.
  - ❑ Length of time between rollovers is 497 days.
- Network Address – used to specify a string of 4 octets.
  - ❑ Currently used to store IPv4 addresses.
  - ❑ Was designed to allow a network address of any type to be specified.
  - ❑ Obsolete – use **IpAddress.**
- Opaque – used to specify octets of binary information.
  - ❑ Generic type.

ASN.1 Examples (MIB)

- RFC 1155
- internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
  - 1.3.6.1.
- Counter ::= [APPLICATION 1] IMPLICIT INTEGER (0..4294967295)
- TimeTicks ::= [APPLICATION 3] IMPLICIT INTEGER (0..4294967295)
- IpAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING (SIZE (4))
- NetworkAddress ::= CHOICE { internet IpAddress }