

Network Management

- As we all know, networks are growing larger and larger every day, it becomes difficult to manage them manually through human.
- SNMP provides the automated management of all the networks in an organization.

Network Management

Network management is defined as:

monitoring

testing

configuring

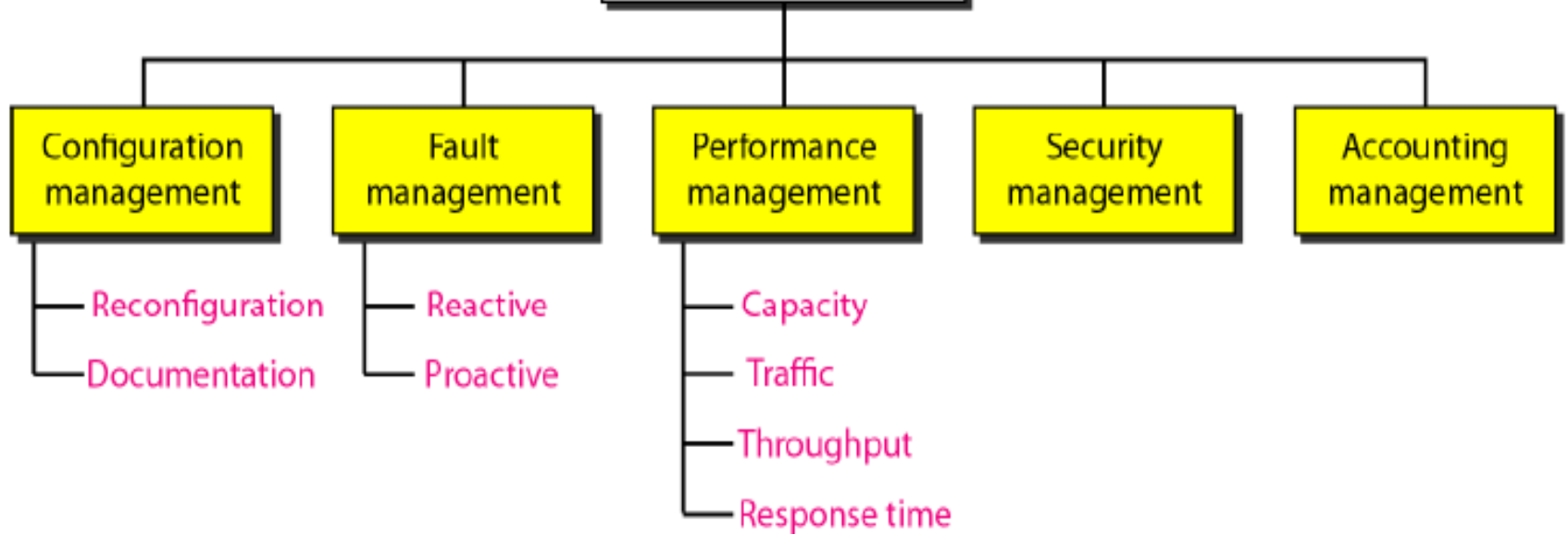
troubleshooting

network components to meet a set of requirements defined by an organization.

The functions of network management system

- **configuration management**
- **fault management**
- **Performance management**
- ***security management***
- ***accounting management***

Functions of a network management system



configuration management

- ***The configuration management system updates information about the status of each entity and its relation to other entities.***
- **Configuration management can be divided into two subsystems:**
 - ***Reconfiguration***
 - ***Hardware reconfiguration***
 - ***Software reconfiguration***
 - ***User account reconfiguration***
 - **documentation.**

Fault management

- ***Fault management supervises the operation of the network, which depends on the proper operation of each individual component and its relation to other components.***
- **A fault management system has two subsystems:**
 - ***reactive fault management***
 - ***proactive fault management.***

Performance management

- ***Performance management monitors and controls the network to ensure that it is***
- **running as efficiently as possible.**
- four measurable quantities in performance management are:
 - ❑ ***capacity***
 - ❑ ***traffic***
 - ❑ ***throughput***
 - ❑ ***response time.***

Security management

- ***Security management is responsible for controlling access to the network based***
- **on the predefined policy.**

Accounting management

- ***Accounting management is the control of users' access to network resources through charges.***

Under accounting management, individual users, departments, divisions, or even projects are charged for the services they receive from the network.

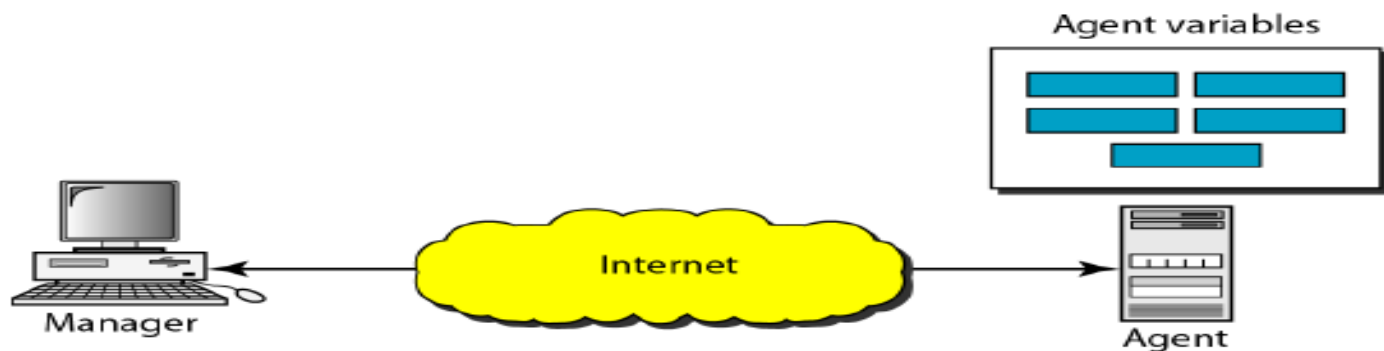
- SNMP uses the concept of
- **Manager:**
- Manager is usually a host, controls and monitors a set of agents
- **Agent**
- Agents are usually routers.

SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

- *The Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using the TCP/IP protocol suite.*
- *It provides a set of fundamental operations for monitoring and maintaining an internet.*

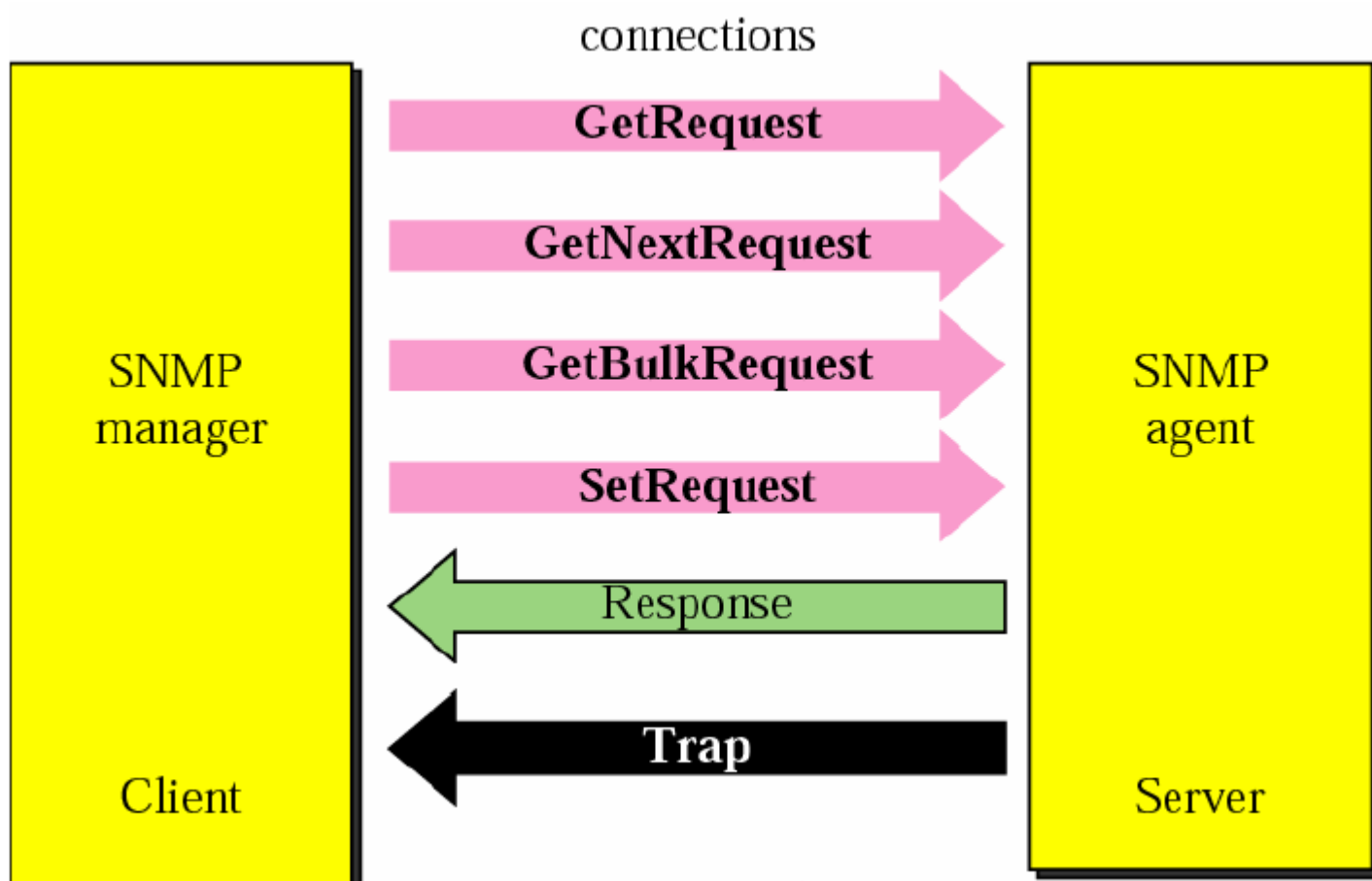
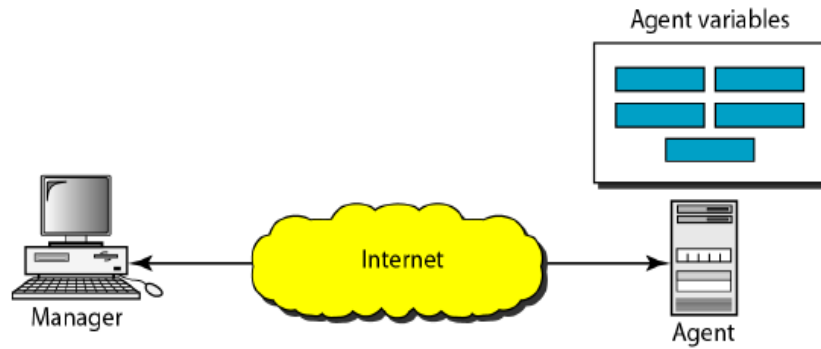
SNMP concept

- SNMP uses the concept of manager and agent
- Manager is usually a host controls and monitors a set of agents
- Agent: is usually a router
- SNMP is a application level layer.



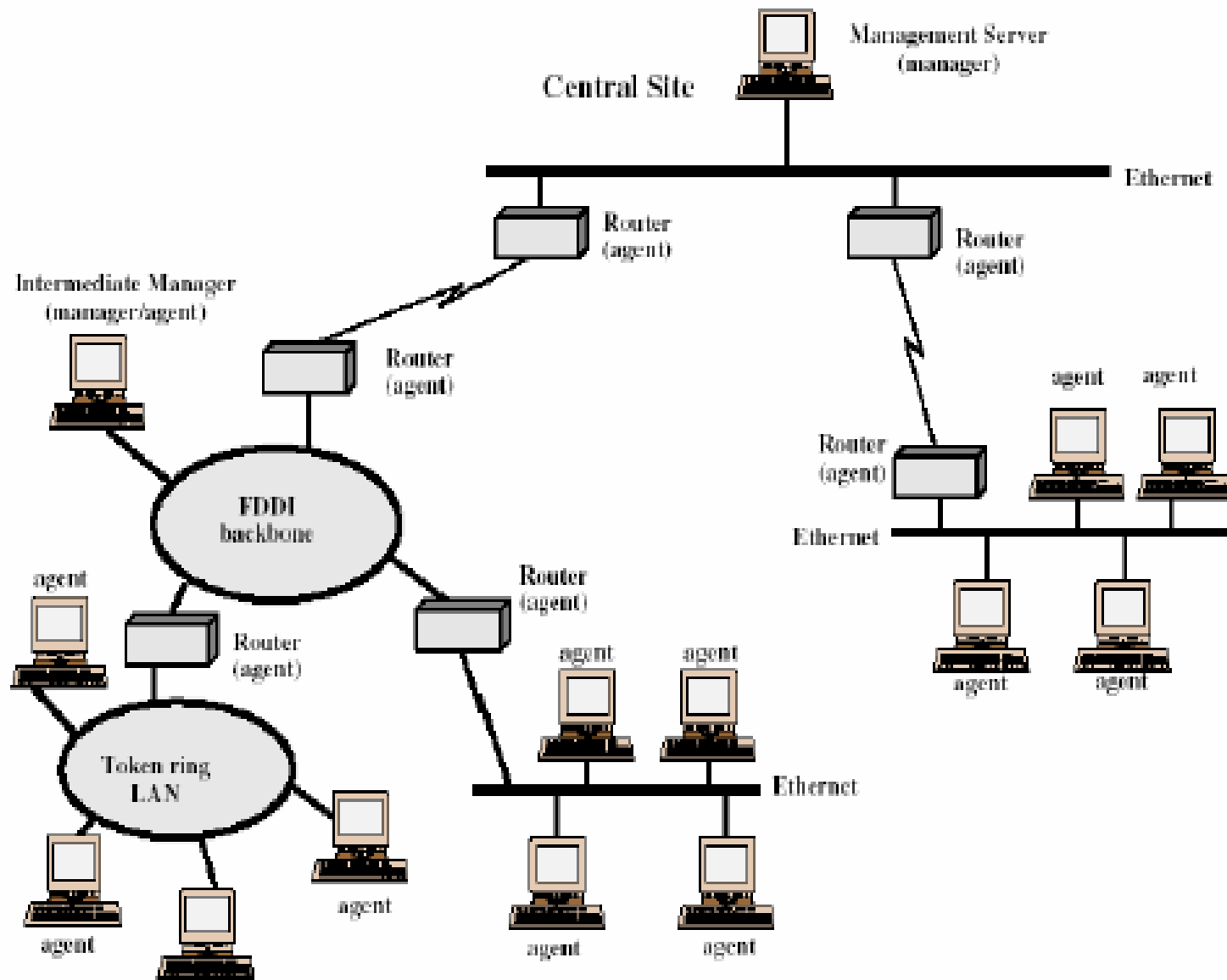
Manager and Agents

- A management station, called a manager, is a host that runs the SNMP client program
- A managed station, called an agent, is a router or host that runs the SNMP server program
- Management is achieved through simple interaction between manager and an agent



SNMP

- SNMP is an application level protocol and can be used in a heterogeneous internet made of different LANs and WANs connected by routers made by different manufacturers.

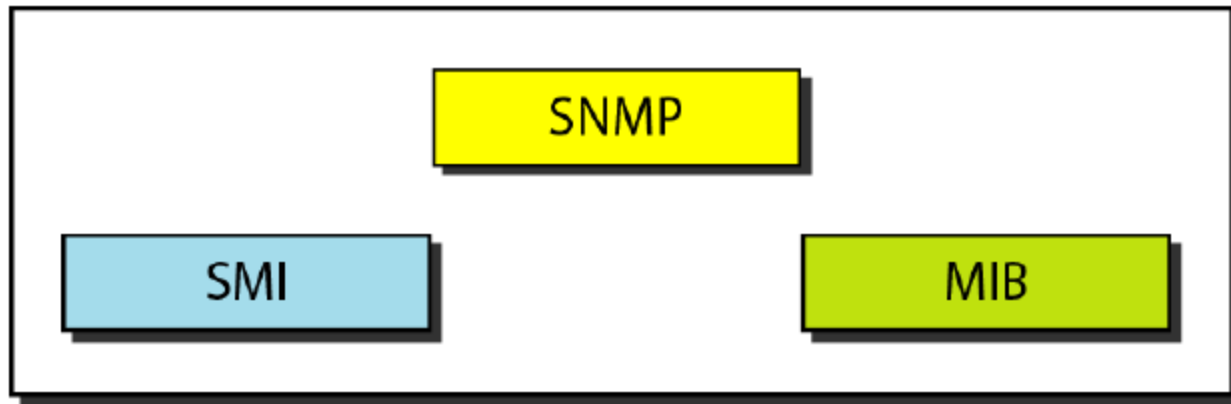


- The agent responds to requests for information from a management station and may asynchronously provide the management station with important information, which generally happens if some exceptional event occurs, called as traps.

management system elements

- A network management system incorporates the following key elements
- 1. Management Station or manager
- 2. Agent
- 3. Management Information Base (MIB)
- 4. Network Management Protocol (SNMP)
- 5. SMI (Structure Of Management Information

Management



- SNMP has a very specific and very important role in Network Management.
- It defines the format of the packet to be sent from Manager to agent and from agent to Manager.
- The packets exchanged contain the control information and data, where data consists of variables (objects) and their values.

Note

SNMP defines the format of packets exchanged between a manager and an agent. It reads and changes the status (values) of objects (variables) in SNMP packets.

- SMI defines the general rules for naming
- objects, defining object types (including
- range and length), and showing how to
- encode objects and values.

Note

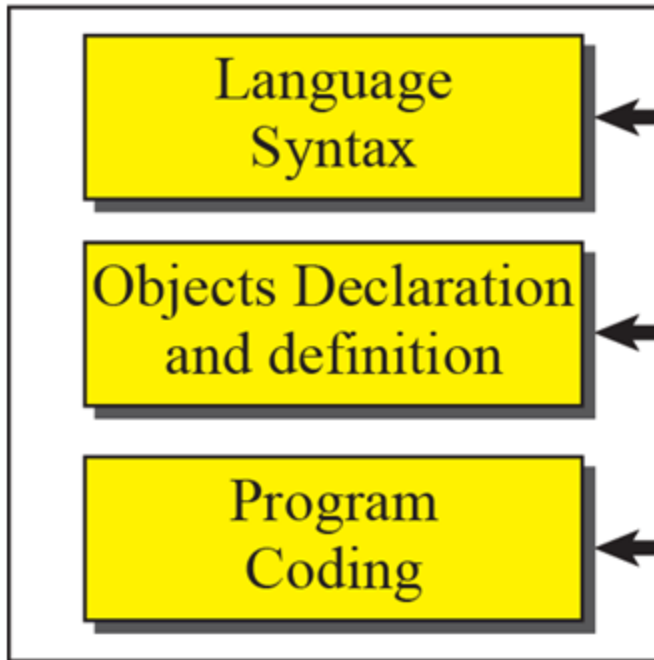
MIB creates a collection of named objects, their types, and their relationships to each other in an entity to be managed.

Note

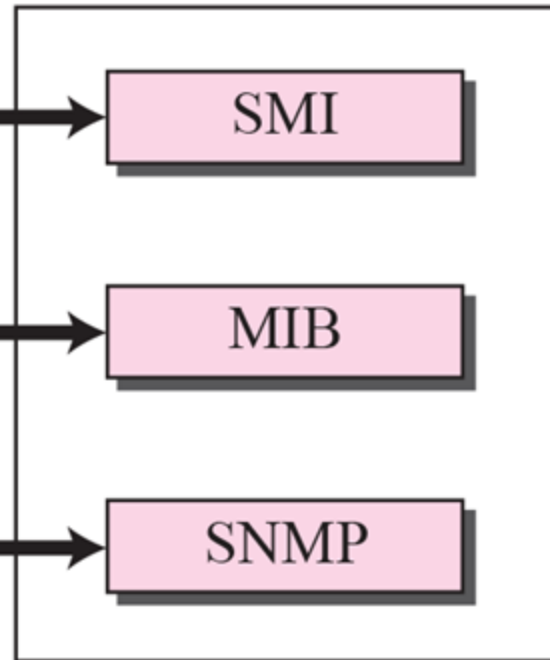
We can compare the task of network management to the task of writing a program.

- ❑ Both tasks need rules. In network management this is handled by SMI.
- ❑ Both tasks need variable declarations. In network management this is handled by MIB.
- ❑ Both tasks have actions performed by statements. In network management this is handled by SNMP.

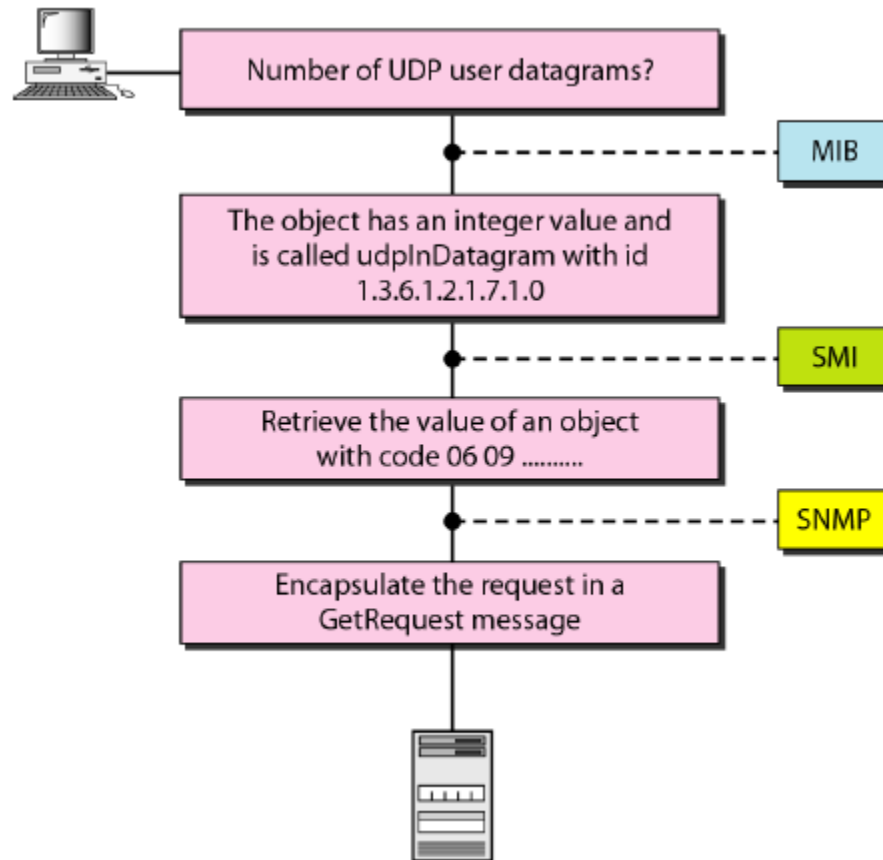
Computer Programming



Network Management



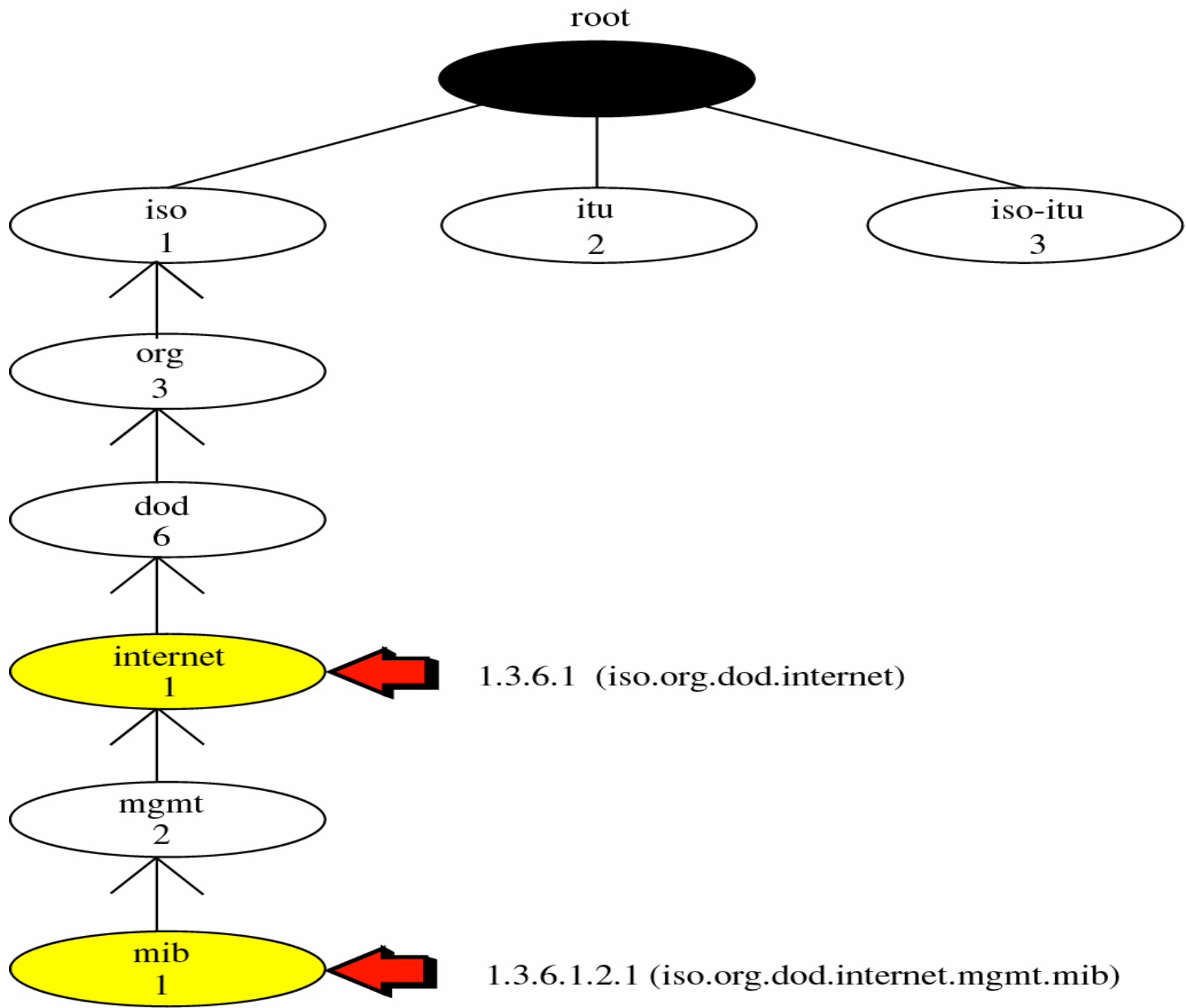
Management overview



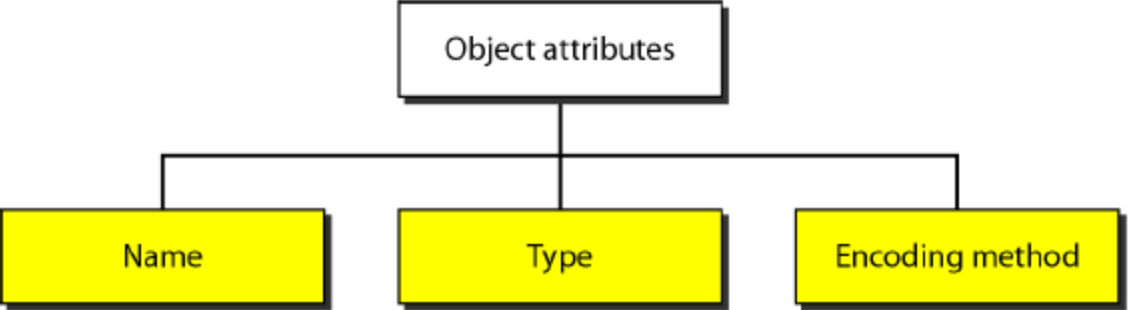
SMI

(Structure of Management Information)

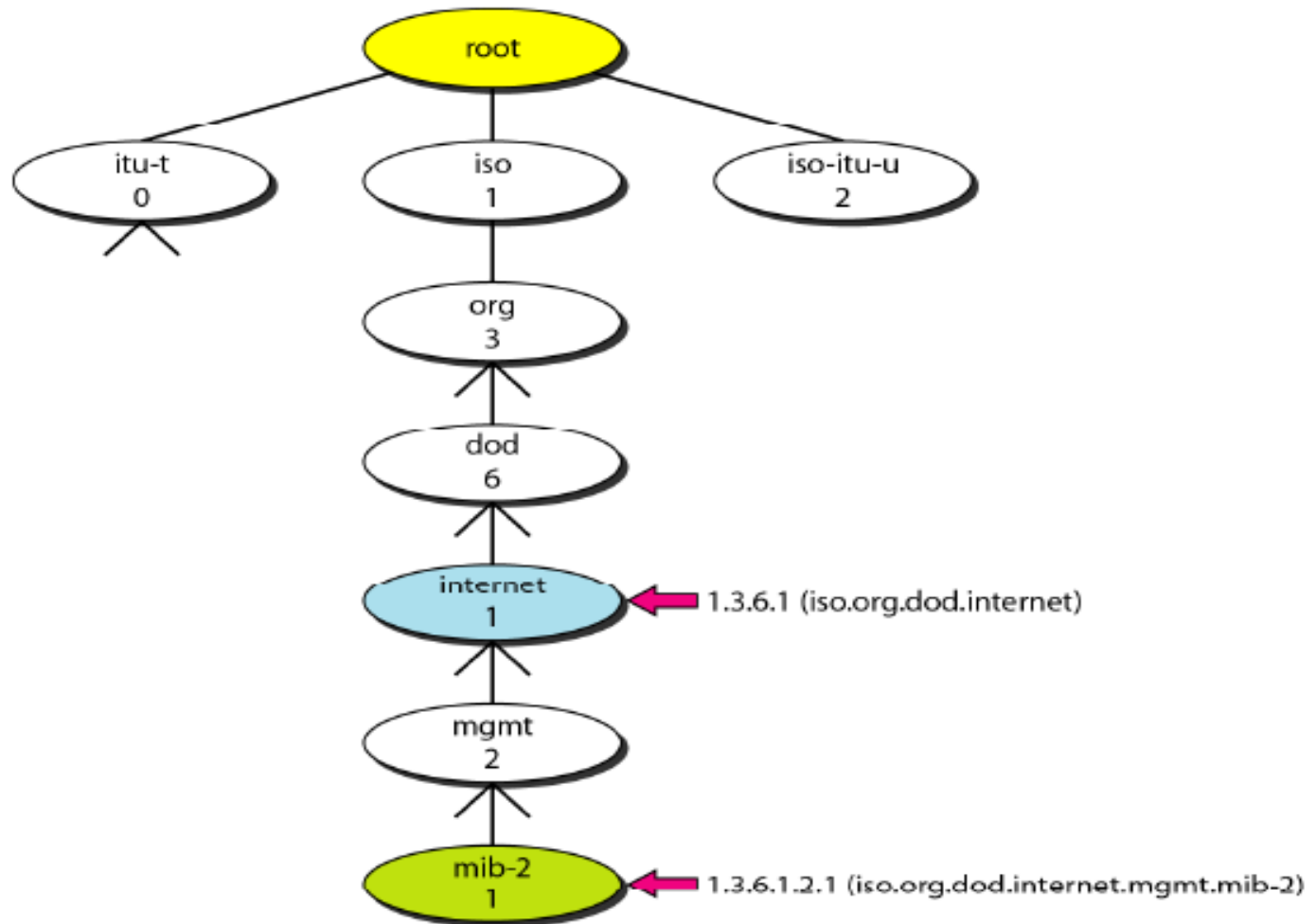
- SMI specifies the rules for naming object.
- Objects in SNMP form a hierarchical structure as shown below.
- Besides naming the object, data type of the object has to be defined.
- All these issues are covered in SMI



- SMIv2 functions are as follows
- 1. To name objects
- 2. To define the different types of data
- 3. To show how to encode data for transmission over the network.



Name_Object identifier

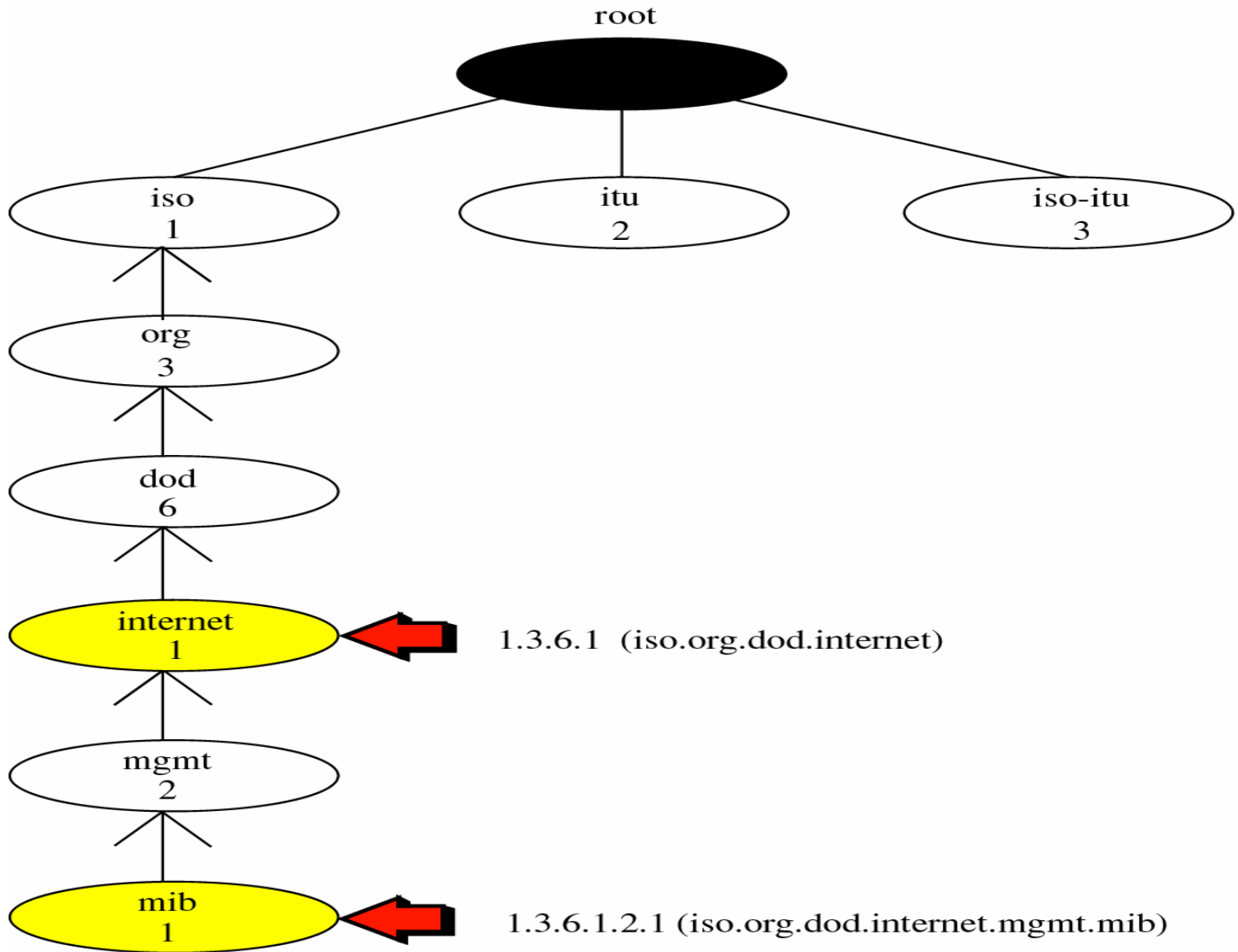


Object identifier

Note

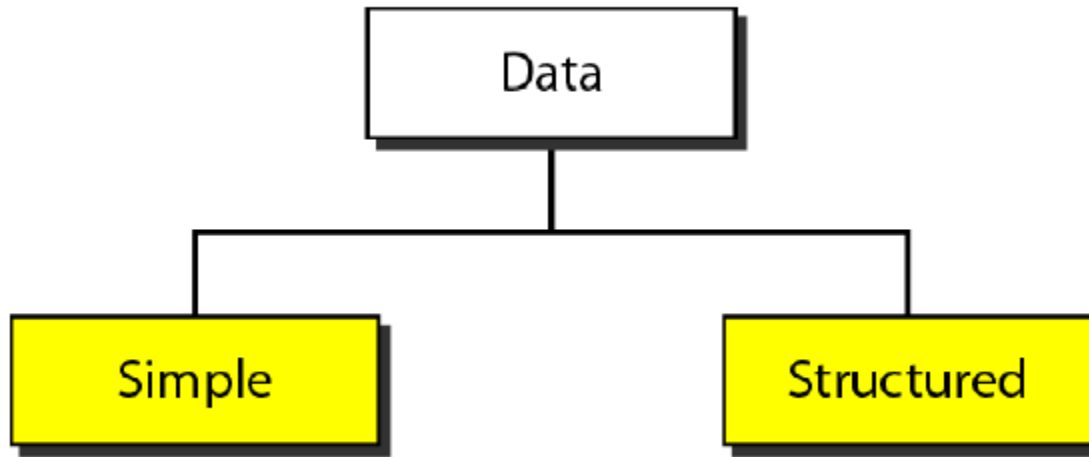
All objects managed by SNMP are given an object identifier.

The object identifier always starts with 1.3.6.1.2.1.



- Each object can be defined using a sequence of integers separated by dots.
- This representation is used in SNMP.

Data type

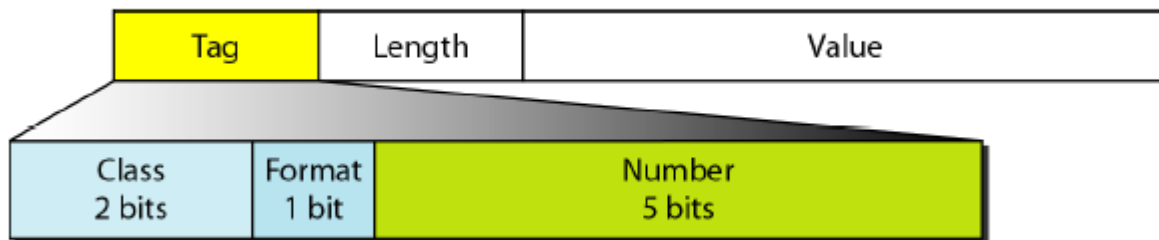


- **SMI defines two types of TYPE,**
- **one is simple type and other one is structured type.**
- **The simple Types are atomic types.**
- **Some of them are directly taken from ASN.1 and some are added to SMI.**
- **The first five are from ASN.1, and the next seven are defined by SMI.**

Type	Size	Description
Integer	4 bytes	An integer with a value between $-2^{31}-1$ and $2^{31}-1$
Integer32	4 bytes	Same as integer
Unsigned 32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string upto 65,535 bytes long
OBJECT IDENTIFIER	Variable	An Object Identifier
IP Address	4 bytes	An IP address made up of 4 integers
Counter32	4 bytes	An integer whose value can be incremented from 0 to 2^{32}
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter 32, but doesn't wrap
Time Ticks	4 bytes	A counting value that records time in $1/100^{\text{th}}$ of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted String

Encoding format

- Tag is a 1-byte field that defines the type of data.
- It is composed of three subfields:
 - (i) Class (2 bits)
 - (ii) Format (1 bit)
 - (iii) Number (5 bits).



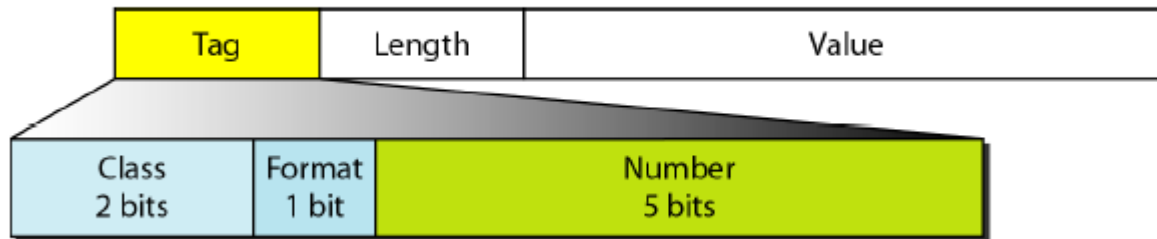
- The **class field specifies the scope of the data.**
Four classes are defined
- 1. universal (00) data types taken from ASN.1
(first five in data type table)
- 2. application wide (01) data types taken from
SMI (next seven)
- 3. Context Specific (10)
- 4. Private (11)

Codes for data types

<i>Data Type</i>	<i>Class</i>	<i>Format</i>	<i>Number</i>	<i>Tag (Binary)</i>	<i>Tag (Hex)</i>
INTEGER	00	0	00010	00000010	02
OCTET STRING	00	0	00100	00000100	04
OBJECT IDENTIFIER	00	0	00110	00000110	06
NULL	00	0	00101	00000101	05
Sequence, sequence of	00	1	10000	00110000	30
IPAddress	01	0	00000	01000000	40
Counter	01	0	00001	01000001	41
Gauge	01	0	00010	01000010	42
TimeTicks	01	0	00011	01000011	43
Opaque	01	0	00100	01000100	44

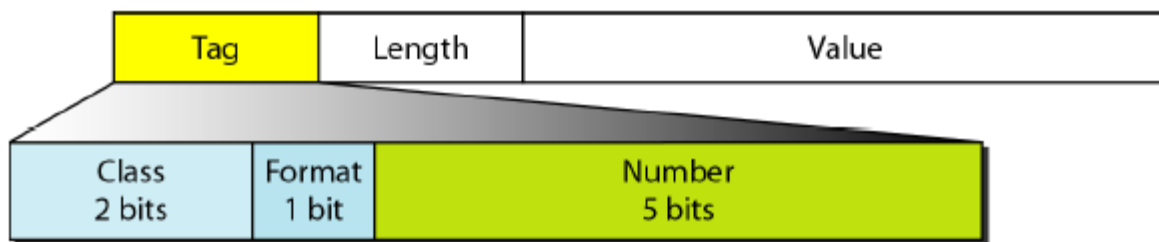
Data	Class	Format	Number	Tag(Binary)	Tag(Hex)
Integer	00	0	00010	00000010	02
String	00	0	00100	00000100	04
ObjectIdentifier	00	0	00110	00000110	06
Sequence <i>sequence of</i>	00	1	10000	00110000	30
IPAddress	01	0	00000	01000000	40
Counter	01	0	00001	01000001	41
Gauge	01	0	00010	01000010	42
TimeTicks	01	0	00011	01000011	43

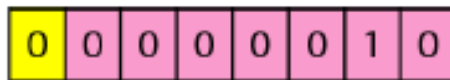
- The **format subfield indicates, whether the data is simple (0) or structured (1).**
- The number subfield further divides simple or structured data into sub-groups.



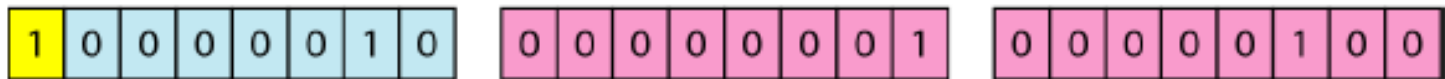
Length format

- **Length: The length field is 1 or more bytes.**
- If it is 1 byte, the most significant bit must be 0. The other 7 bits define the length of the data.
- If it is more than 1 byte, the most significant bit must be 1.
- The other 7 bits define the number of bytes needed to define the length.





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



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

- SMI uses:

Basic Encoding Rules (BER)

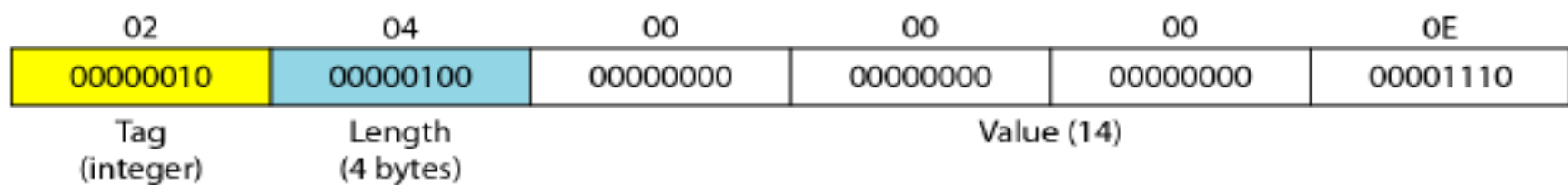
- to encode data to be transmitted over the network as shown below

Example 1

- *shows how to define INTEGER 14*

Type	Size	Description
Integer	4 bytes	An integer with a value between $-2^{31}-1$ and $2^{31}-1$
Integer32	4 bytes	Same as integer
Unsigned 32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string upto 65,535 bytes long
OBJECT IDENTIFIER	Variable	An Object Identifier
IP Address	4 bytes	An IP address made up of 4 integers
Counter32	4 bytes	An integer whose value can be incremented from 0 to 2^{32}
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter 32, but doesn't wrap
Time Ticks	4 bytes	A counting value that records time in $1/100^{\text{th}}$ of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted String

Data	Class	Format	Number	Tag(Binary)	Tag(Hex)
Integer	00	0	00010	00000010	02
String	00	0	00100	00000100	04
ObjectIdentifier	00	0	00110	00000110	06
Sequence <i>sequence of</i>	00	1	10000	00110000	30
IPAddress	01	0	00000	01000000	40
Counter	01	0	00001	01000001	41
Gauge	01	0	00010	01000010	42
TimeTicks	01	0	00011	01000011	43



- Message code:
- 02 04 00 00 00 0E

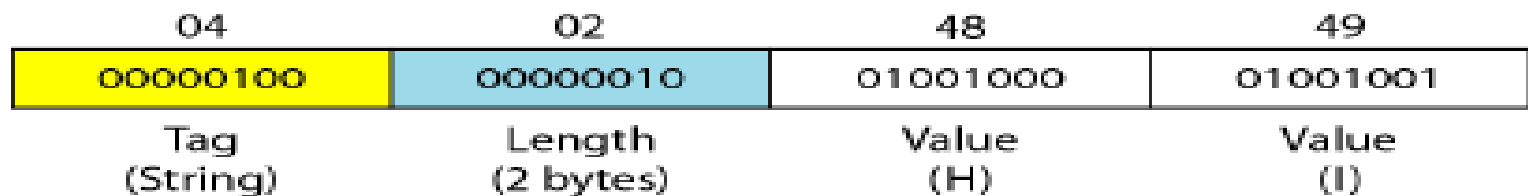
Example 2

- *shows how to define the OCTET STRING*
- *“HI”.*

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	'	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DEL

Type	Size	Description
Integer	4 bytes	An integer with a value between $-2^{31}-1$ and $2^{31}-1$
Integer32	4 bytes	Same as integer
Unsigned 32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string upto 65,535 bytes long
OBJECT IDENTIFIER	Variable	An Object Identifier
IP Address	4 bytes	An IP address made up of 4 integers
Counter32	4 bytes	An integer whose value can be incremented from 0 to 2^{32}
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter 32, but doesn't wrap
Time Ticks	4 bytes	A counting value that records time in $1/100^{\text{th}}$ of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted String

Data	Class	Format	Number	Tag(Binary)	Tag(Hex)
Integer	00	0	00010	00000010	02
String	00	0	00100	00000100	04
ObjectIdentifier	00	0	00110	00000110	06
Sequence <i>sequence of</i>	00	1	10000	00110000	30
IPAddress	01	0	00000	01000000	40
Counter	01	0	00001	01000001	41
Gauge	01	0	00010	01000010	42
TimeTicks	01	0	00011	01000011	43

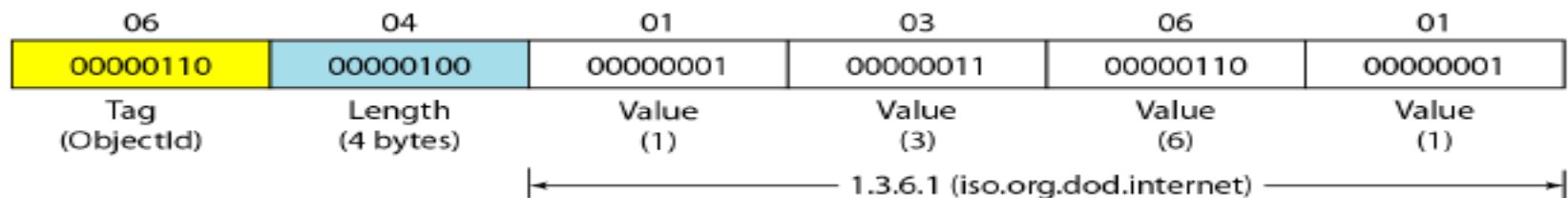


Example 3

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

Type	Size	Description
Integer	4 bytes	An integer with a value between $-2^{31}-1$ and $2^{31}-1$
Integer32	4 bytes	Same as integer
Unsigned 32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string upto 65,535 bytes long
OBJECT IDENTIFIER	Variable	An Object Identifier
IP Address	4 bytes	An IP address made up of 4 integers
Counter32	4 bytes	An integer whose value can be incremented from 0 to 2^{32}
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter 32, but doesn't wrap
Time Ticks	4 bytes	A counting value that records time in $1/100^{\text{th}}$ of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted String

Data	Class	Format	Number	Tag(Binary)	Tag(Hex)
Integer	00	0	00010	00000010	02
String	00	0	00100	00000100	04
ObjectIdentifier	00	0	00110	00000110	06
Sequence <i>sequence of</i>	00	1	10000	00110000	30
IPAddress	01	0	00000	01000000	40
Counter	01	0	00001	01000001	41
Gauge	01	0	00010	01000010	42
TimeTicks	01	0	00011	01000011	43

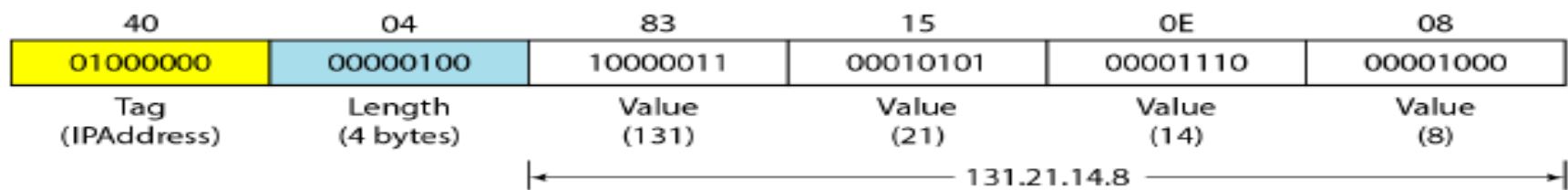


Example 4

- *shows how to define IPAddress 131.21.14.8*

Type	Size	Description
Integer	4 bytes	An integer with a value between $-2^{31}-1$ and $2^{31}-1$
Integer32	4 bytes	Same as integer
Unsigned 32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string upto 65,535 bytes long
OBJECT IDENTIFIER	Variable	An Object Identifier
IP Address	4 bytes	An IP address made up of 4 integers
Counter32	4 bytes	An integer whose value can be incremented from 0 to 2^{32}
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter 32, but doesn't wrap
Time Ticks	4 bytes	A counting value that records time in $1/100^{\text{th}}$ of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted String

Data	Class	Format	Number	Tag(Binary)	Tag(Hex)
Integer	00	0	00010	00000010	02
String	00	0	00100	00000100	04
ObjectIdentifier	00	0	00110	00000110	06
Sequence <i>sequence of</i>	00	1	10000	00110000	30
IPAddress	01	0	00000	01000000	40
Counter	01	0	00001	01000001	41
Gauge	01	0	00010	01000010	42
TimeTicks	01	0	00011	01000011	43



Management Information Base

- The Management Information Base, version 2 (MIB_2) is the second component used in network management.
- Each agent has its own MIB_2, which is a collection of all the objects that the manager can manage.

- The objects in MIB_2 are categorized under 10 different groups:
- system,
- interface,
- address translation,
- ip,
- icmp,
- tcp,
- udp,
- egp,
- transmission,
- snmp.
-

sys

- This object (system) defines general information about the system (node):
- His name
- Location
- Lifetime

if

- This object (interface) defines information about all the interfaces of the node including:
- Interface number
- Physical address
- IP address

at

- This object (address translation) defines the information about the ARP table.

ip

- This object defines information related to IP, such as the routing table and the IP address

icmp

- This object defines information related to ICMP, such as the number of packets sent and received and total error

tcp

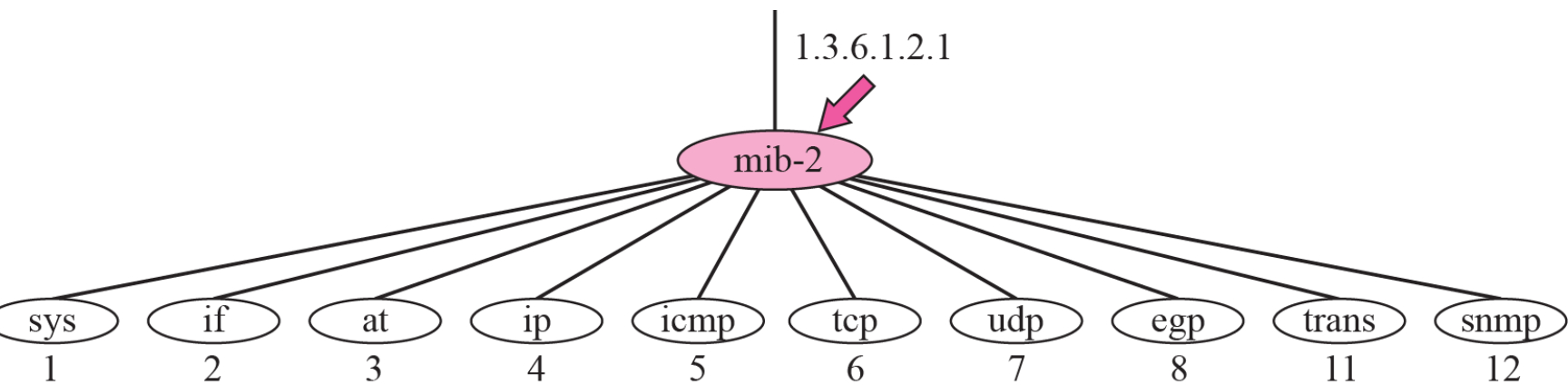
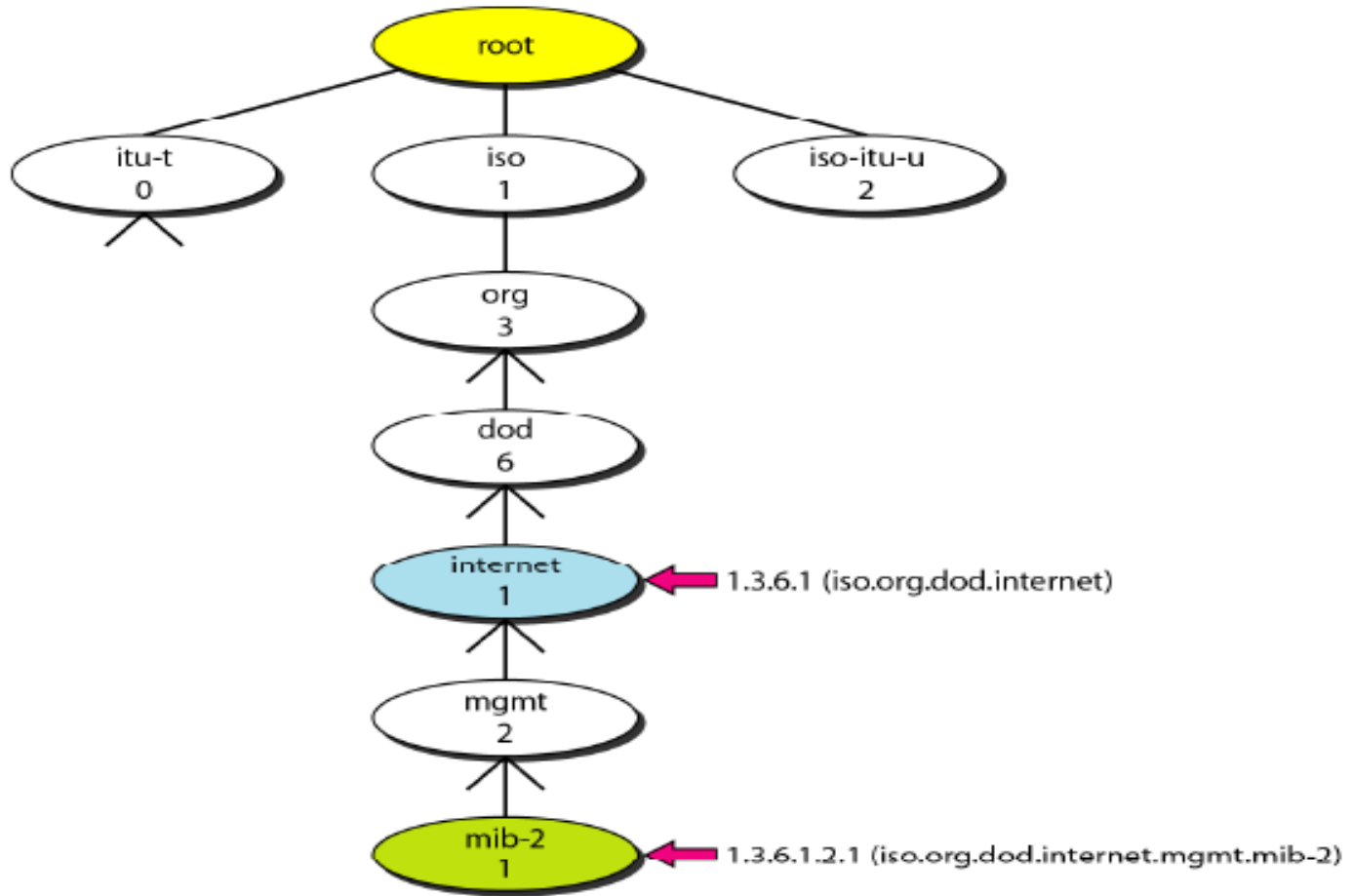
- This object defines general information related to TCP, such as the connection table, number of ports, number of packets sent and received

udp

- This object defines general information related to UDP, such as the number of ports, number of packets sent and received.

snmp

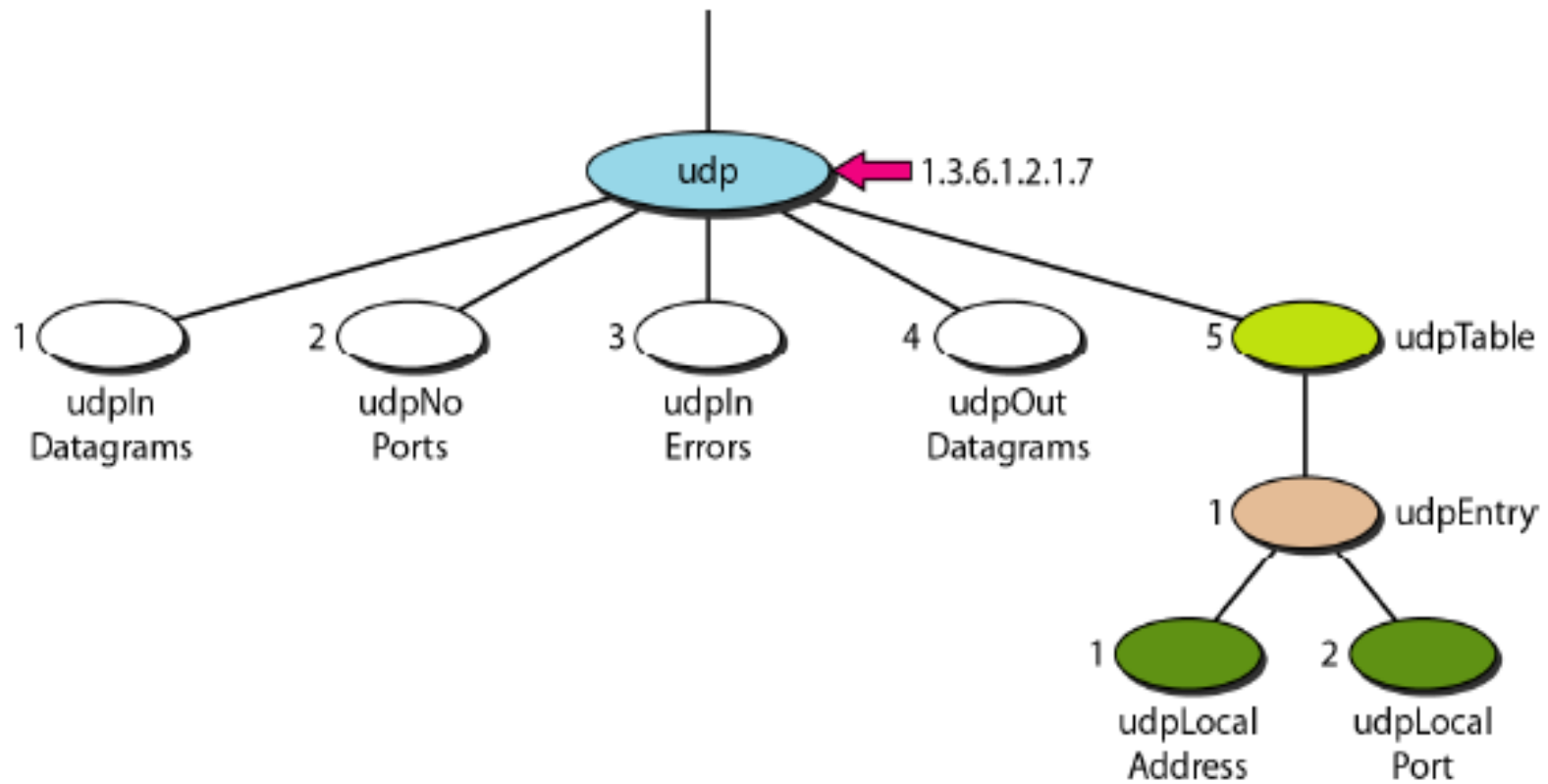
- This object defines general information related to SNMP itself



Accessing MIB variables




- For example, we use the “ udp “ group
- There are 4 simple variables in the udp group
- And 1 sequence of (table of) record

UDP Groups




Simple variable


- To access any of the simple variables, we use the “ id “ of the group (1.3.6.2.1.7) followed by the id of the variable


- udpInDatagrams 1.3.6.1.2.1.7.1
- udpNoPorts 1.3.6.1.2.1.7.2
- udpOutDatagrams 1.3.6.1.2.1.7.4
- These are the variables, to access their values,
- So we have to suffix them with .0 i.e.
- udpInDatagrams.0  1.3.6.1.2.1.7.1.0
- udpNoPorts.0  1.3.6.1.2.1.7.2.0
- udpOutDatagrams.0  1.3.6.1.2.1.7.4.0


Accessing Tables

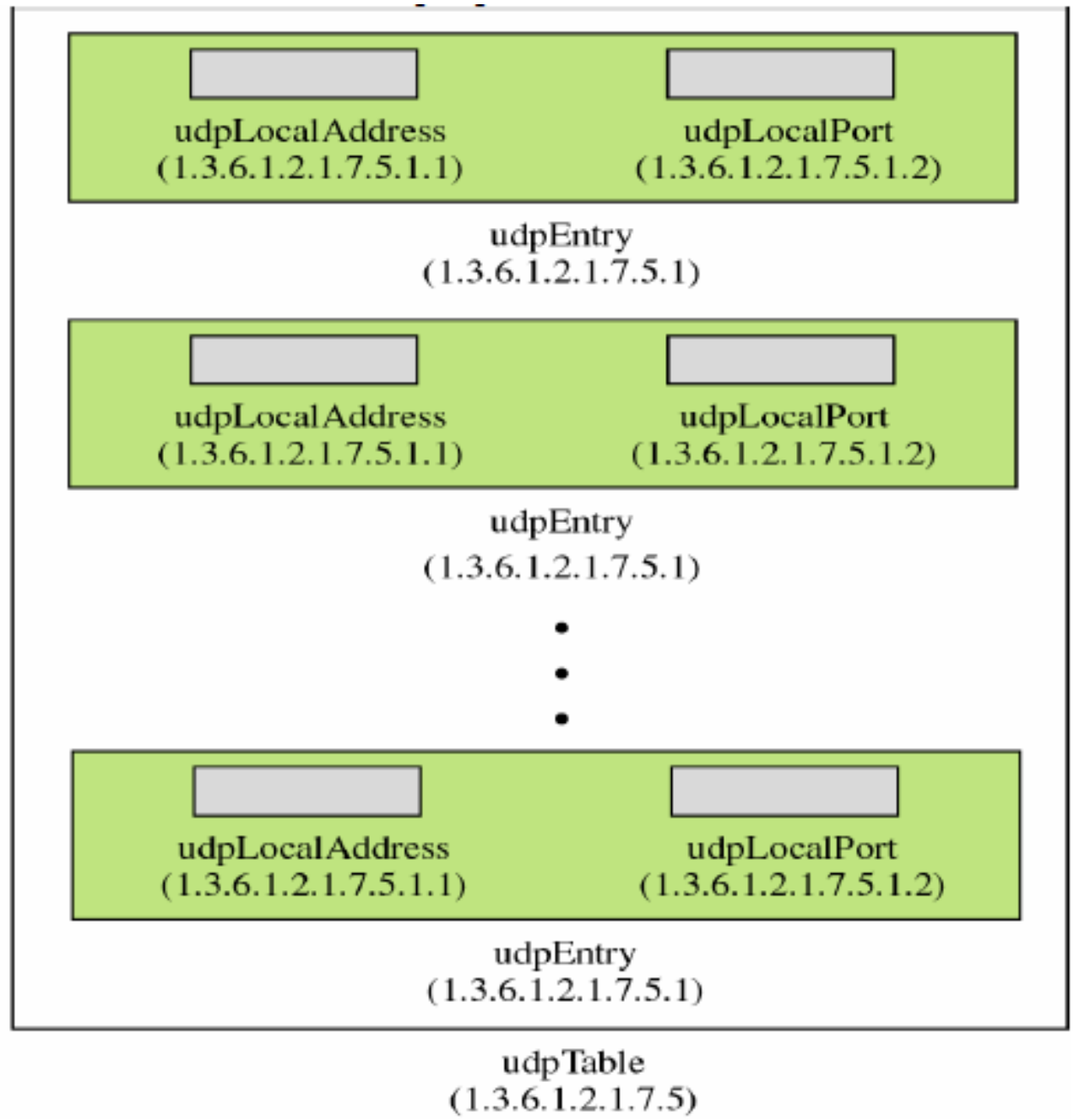
- for example for `udpLocalAddress` :
- there must be many `udpLocalPort`,
because corresponding to one address there
are multiple port, so that form a table.

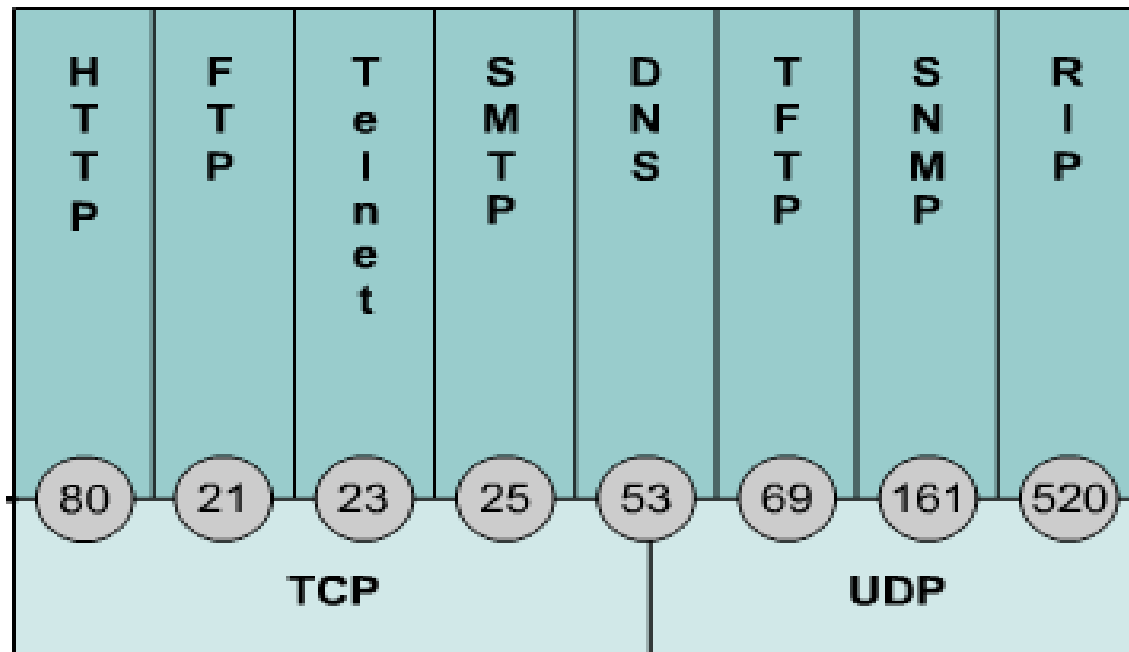

udpInDatagrams
(1.3.6.1.2.1.7.1)


udpNoPorts
(1.3.6.1.2.1.7.2)


udpInErrors
(1.3.6.1.2.1.7.3)


udpOutDatagrams
(1.3.6.1.2.1.7.4)



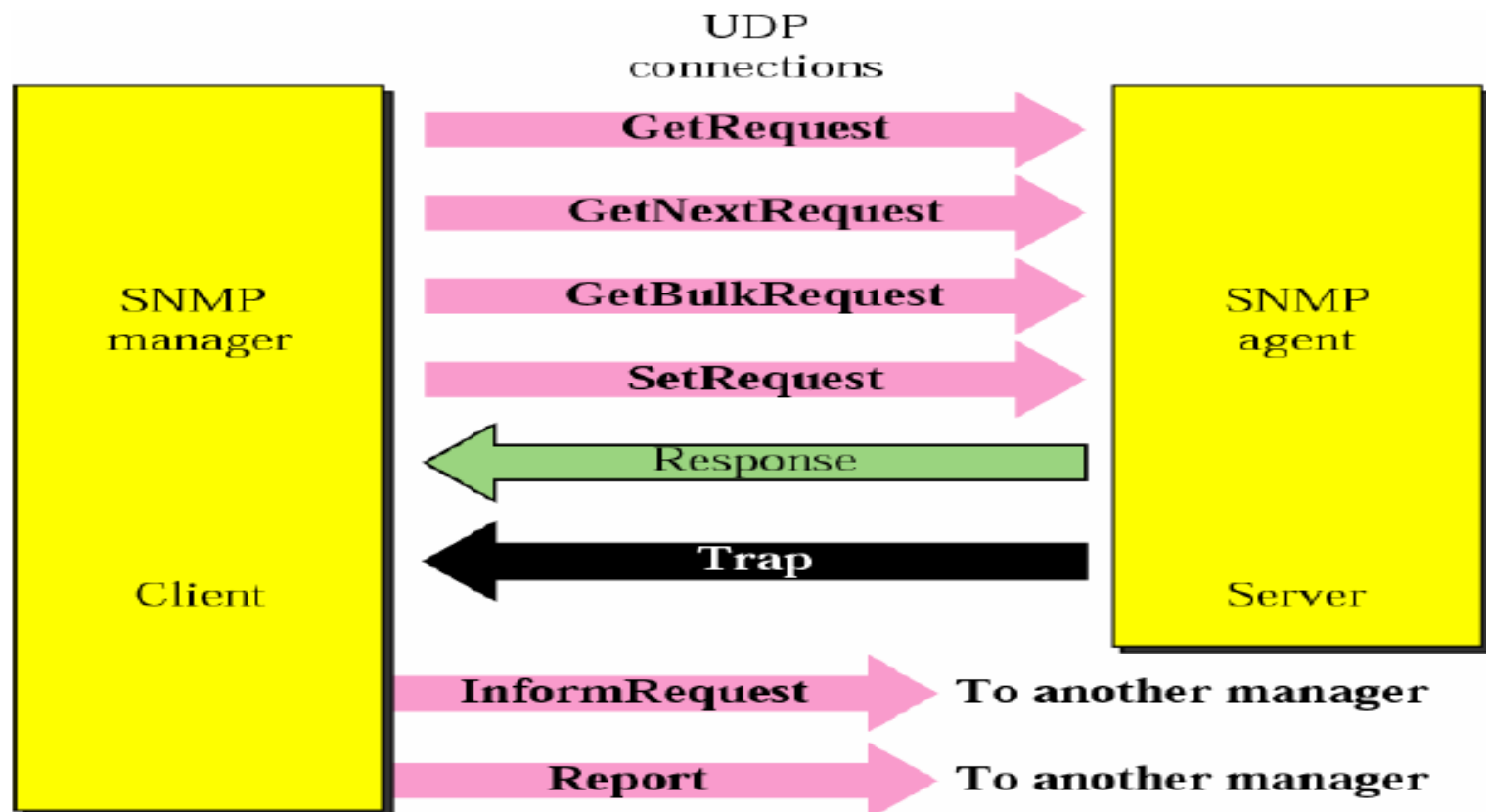


SNMP (Simple Network Management Protocol)

- SNMP uses both SMI and MIB in network management.
- SNMP define the rules or protocol for exchange of information between the SNMP manager and SNMP Client

Packet datagram unit

- SNMPv3 defines 8 types of packets as shown:



- **GetRequest:**
- The Get Request PDU is sent from the manager to the agent to retrieve the value of a variable or set of variables.
- **GetNextRequest:**
- The GetNextRequest is sent from the manager to agent to retrieve the value of variable.
- The retrieved value is the value of the object following the defined ObjectId in the PDU.
- **GetBulkRequest: T**
- The GetBulkRequest PDU is sent from the manager to agent to retrieve a large amount of data, generally a table.
- **SetRequest:**
- The SetRequest PDU is sent from the manager to agent to set (store) a value in a variable.

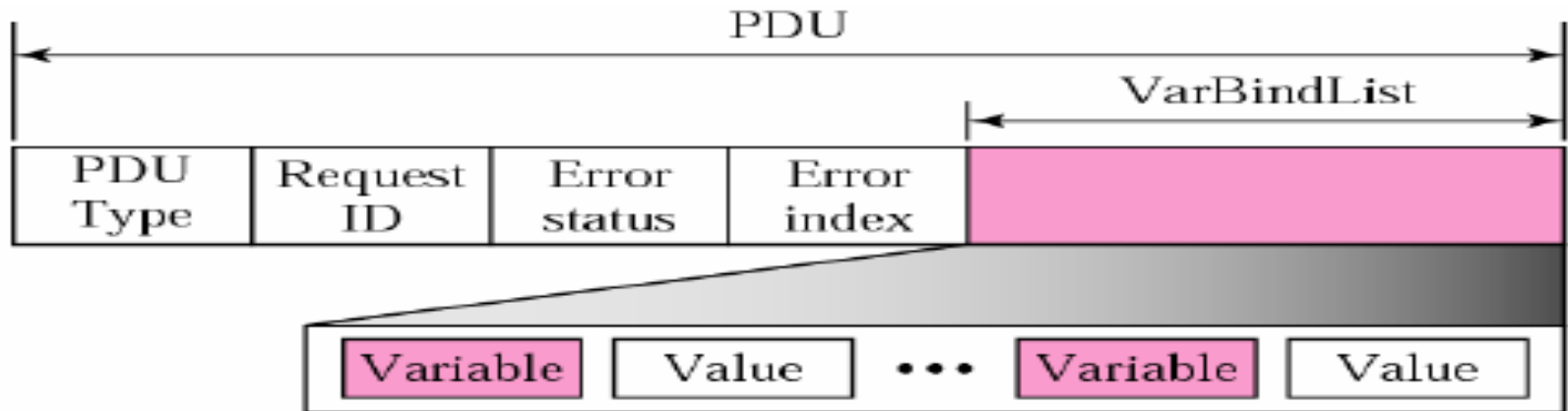
- **Response:**
- The response PDU is sent from an agent to the manager in response to GetRequest or GetNextRequest.
- It contains the values of variables requested by manager.
- **Trap:**
- This PDU is sent from an agent to the manager to report an event. For example, if the agent is rebooted, it informs the manager and reports the time of rebooting

- **Report:**
- The report PDU is designed to report some types of errors between managers.
- **InformRequest:**
- The InformRequest is sent from one manager to another remote manager to get the value of some variables from agents under the control of remote manager.

Codes for SNMP messages

<i>Data</i>	<i>Class</i>	<i>Format</i>	<i>Number</i>	<i>Whole Tag (Binary)</i>	<i>Whole Tag (Hex)</i>
GetRequest	10	1	00000	10100000	A0
GetNextRequest	10	1	00001	10100001	A1
Response	10	1	00010	10100010	A2
SetRequest	10	1	00011	10100011	A3
GetBulkRequest	10	1	00101	10100101	A5
InformRequest	10	1	00110	10100110	A6
Trap (SNMPv2)	10	1	00111	10100111	A7
Report	10	1	01000	10101000	A8

SNMP PDU format

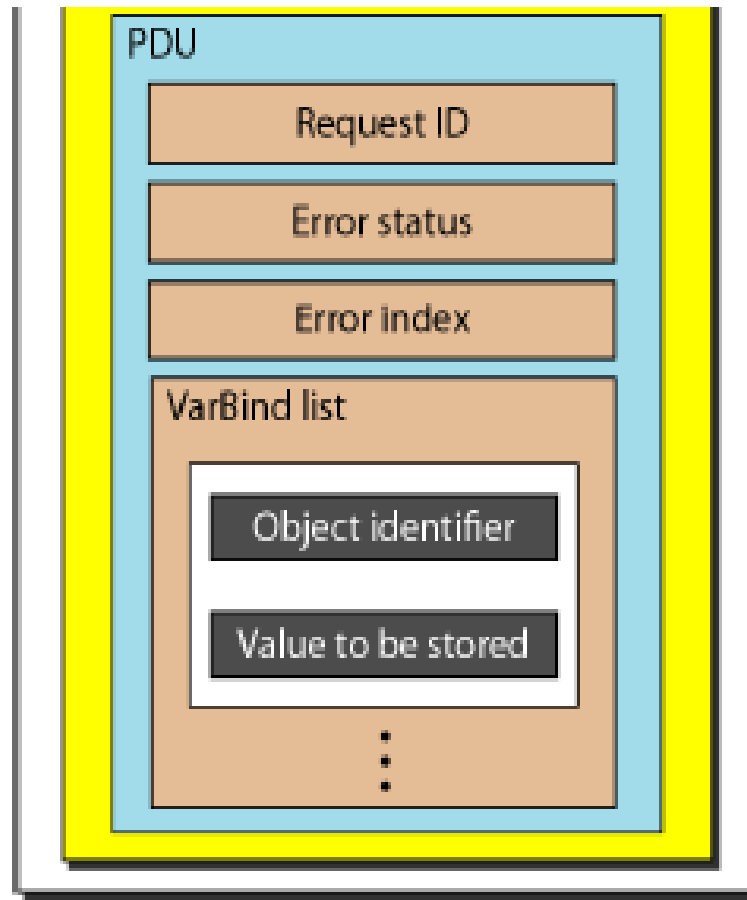


Differences:

1. Error status and error index values are zeros for all request messages except GetBulkRequest.
2. Error status field is replaced by non-repeater field and error index field is replaced by max-repetitions field in GetBulkRequest.

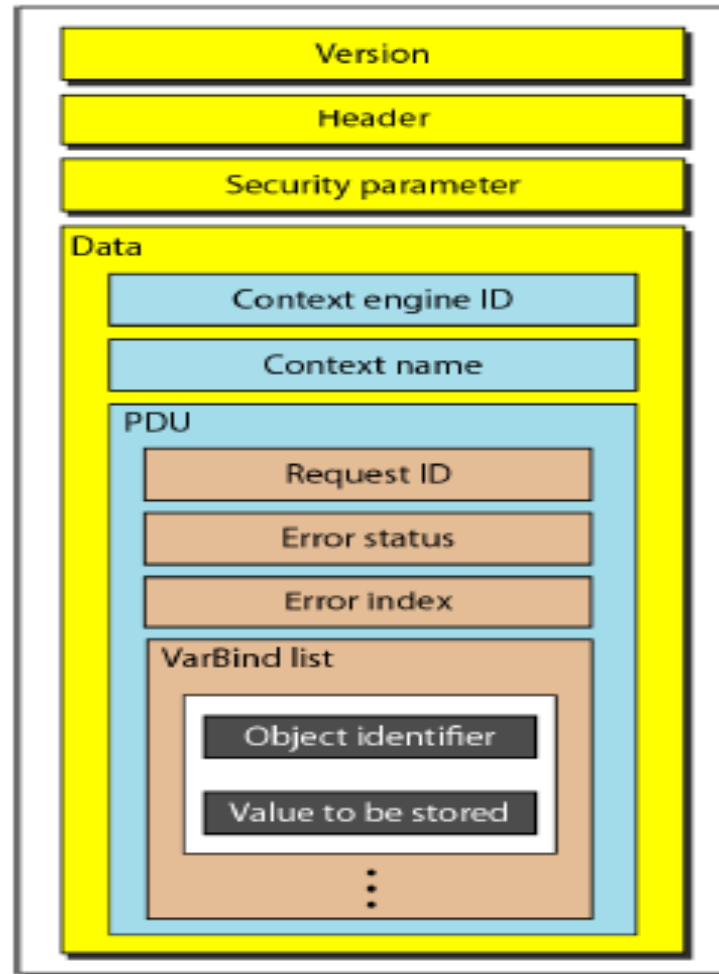
- **Request Id:**
- This field is a sequence number used by the manager in a request PDU and repeated by agent in a response.
- It is used to match a request with a response.
- **Error Status:** This is an integer that is used only in response PDU to show the
- types of error reported by agent. Its value is 0 in request PDU.
- **Error Index:** The error index is an offset that tells the manager which variable
- caused the error
- **Variable List:** This is a set of variables with the corresponding values the manager want to retrieve or set.
- The values are null in request and GetNextRequest.

SNMP PDU format



SNMP message

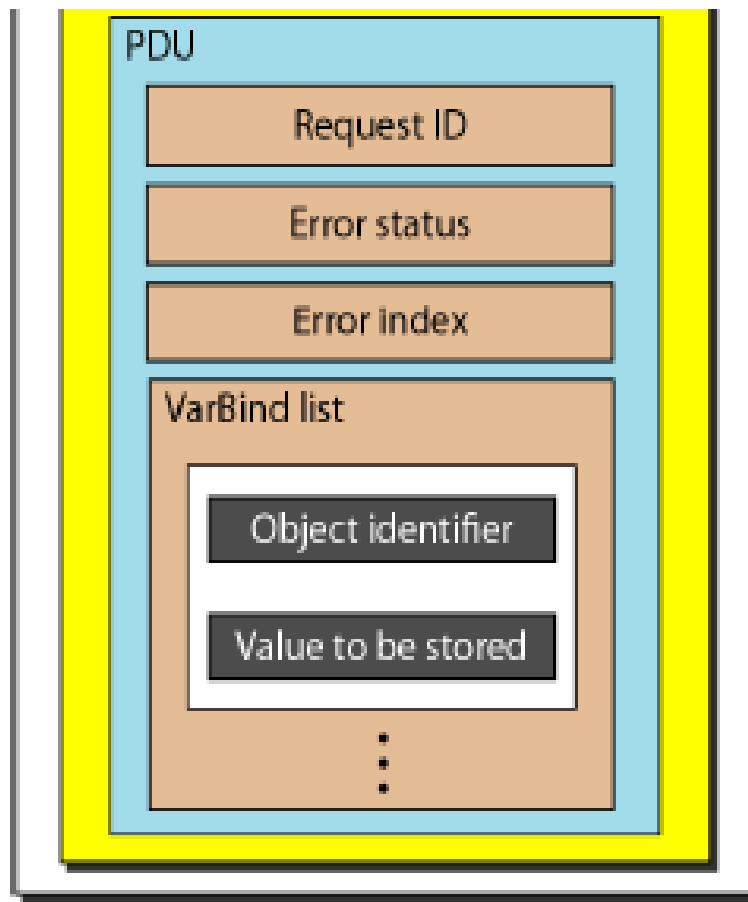
Message



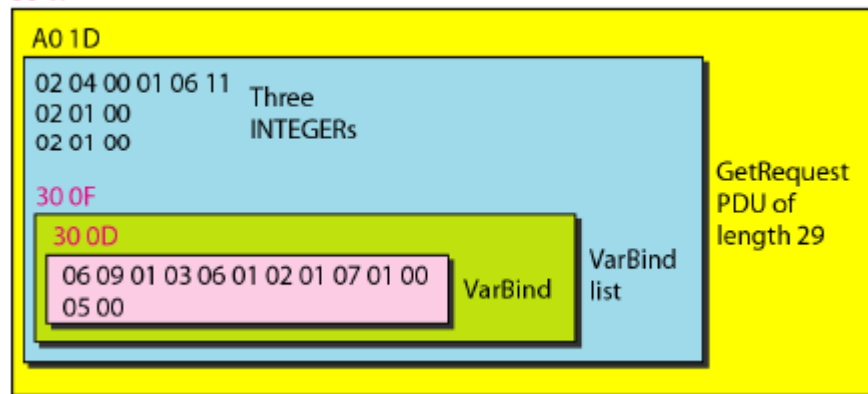
Example

- ***In this example, a manager station (SNMP client) uses***
- ***the GetRequest message to retrieve the number of UDP***
- ***datagrams that a router has received.***
- ***There is only one VarBind entity.***
- ***The corresponding MIB variable related to this information is udplnDatagrams with the object identifier 1.3.6.1.2.1.7.1.0. The manager wants to retrieve a value (not to store a value), so the value defines a null entity.***
- ***Figure shows the conceptual view of the packet and the hierarchical nature of sequences.***
- ***We have***
- ***used white and colored boxes for the sequences and a***
- ***gray one for the PDU.***
- ***The VarBind list has only one VarBind***

- *The variable is of type 06 and length 09.*
- *The value is of type 05 and length 00.*
- *The whole VarBind is a sequence of length 0D (13).*
- *The VarBind list is also a sequence of length 0F (15).*
- *The GetRequest PDU is of length 1D (29).*
- *Now we have three OCTET STRINGS related to the*
- *security parameter, security model, and flags.*
- *Then we have two integers defining maximum size (1024) and message ID (64). The header is a sequence of length 12, which we left blank for simplicity. There is one integer, version (version 3). The whole message is a sequence of 52 bytes.*



30 1F



Data, a sequence of 31 bytes

- ***Figure shows the actual message sent by manager station (client) to agent (server)***

02 01 03 INTEGER, version

30 0C

— — . . . Header, sequence
of length 12, not shown

02 01 40] Two
02 02 04 00] INTEGERS

04 01 00] Three
04 00] OCTET STRINGs
04 00

30 1F

A0 1D

02 04 00 01 06 11 Three
02 01 00 INTEGERs
02 01 00

30 0F

30 0D

06 09 01 03 06 01 02 01 07 01 00
05 00 VarBind

VarBind
list

GetRequest
PDU of
length 29

Data, a
sequence
of 31
bytes

Whole message
a sequence of
52 bytes

- *All thanks to Behrouz A Forouzan*
- *For his lectures which help to cover the topics of this course*