

## Lecture 22

### Network Management (SNMP)

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.

SNMP uses the concept of **manager** and **agent**. That is a Manager, usually a host, controls and monitors a set of agents, usually routers. If Network grows, then there is one manager and multiple sub-divisional managers (which also act as agents), each sub-manager managing and containing information about their part of network resources and reporting to Manager as and when asked for or required.

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.

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. A network management system incorporates the following key elements

1. Management Station or manager
2. Agent
3. Management Information Base
4. Network Management Protocol (SNMP)
5. SMI (Structure Of Management Information described by ASN.1)

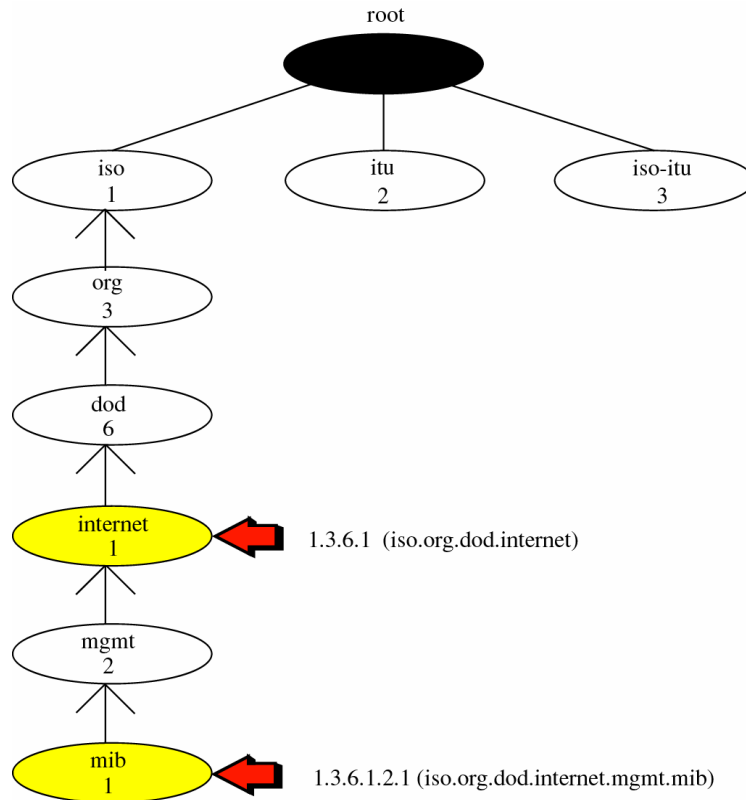
We have already discussed what is job of Manager and what the job of agent is? Now discuss about the rest of the components.

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.

Before going further I would like to **discuss about objects and their values**. These objects are like attributes of a resource, a resource such a router can be defined by multiple attributes with their values, instead of using the work attribute, we refer it as Object and its value. So Data part of packet, contains object-value pairs.

SMI specifies the rules for naming object. Remember one thing, 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, which we will discuss in detail.

Another thing to remember about SMI is that, it only defines the rules for naming object, but it doesn't tell anything about how many objects will be there for a particular resource and which object uses which type



MIB, is required to define what SMI was not doing, that is, SMI didn't define how many objects are required for each entity, and what the type of the object is. So SMI defines the number of objects for each entity and their respective type.

To understand in more detail, let us discuss an analogy from Forouzan,

Before we write a program, the syntax of the language must be predefined. The language also defines the structures of variables, naming convention, data types that can be used. In programming these rules are defined by the language, here by SMI.

In every program written in some language, we declare variables; associate them with some data type. This task is done by MIB in network management.

After declaration of variables, statements are written which stores the value in variables, reads the values from the variable or changes the values of the variable, this job is done by SNMP, which reads or writes or changes the value of objects.

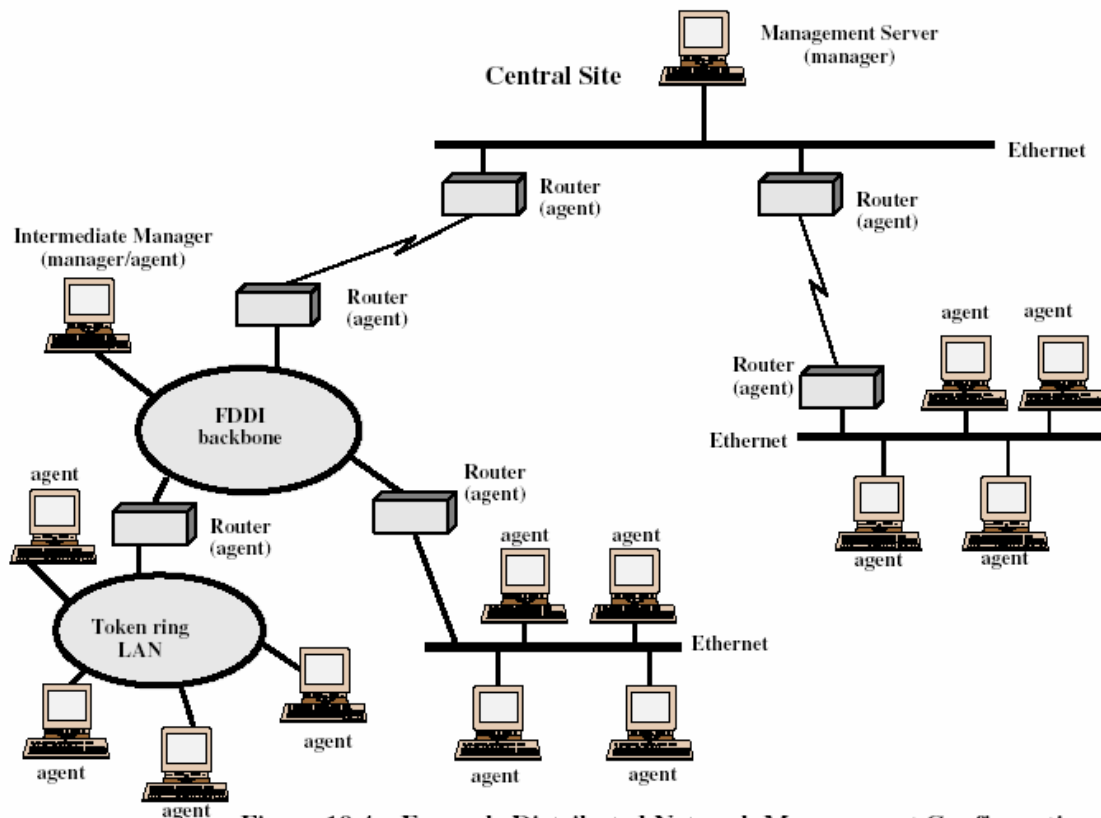


Figure 19.4 Example Distributed Network Management Configuration

### SMI (Structure of Management Information)

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.

To name objects and to define different types of data, ASN.1 (Abstract System Notation) is used. It emphasizes three attributes to handle an object.

1. Name
2. Type
3. Encoding Method

SMI requires that each managed object (being managed by an agent), such as router, a variable in a router has a **unique name**. To name objects globally (unique), SMI uses an object identifier, which is based on hierarchical tree structure as shown above on page 2. Each object can be defined using a sequence of integers separated by dots. This integer-dot representation is used in SNMP.

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

The Structured Type is made by combining simple type; same as structures in C. SMI defines two structured data types

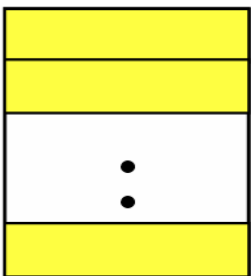
1. **Sequence:** A sequence data type is a combination of simple data type, not necessarily of the same type.
2. **Sequence of:** A sequence of data type is a combination of simple data type all of the same type or combination of sequence data types all of same type, similar to arrays of primitive data type in C or array of structures.



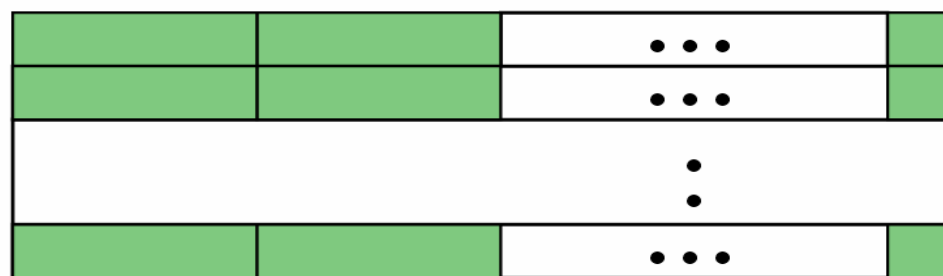
a. Simple variable



c. Sequence

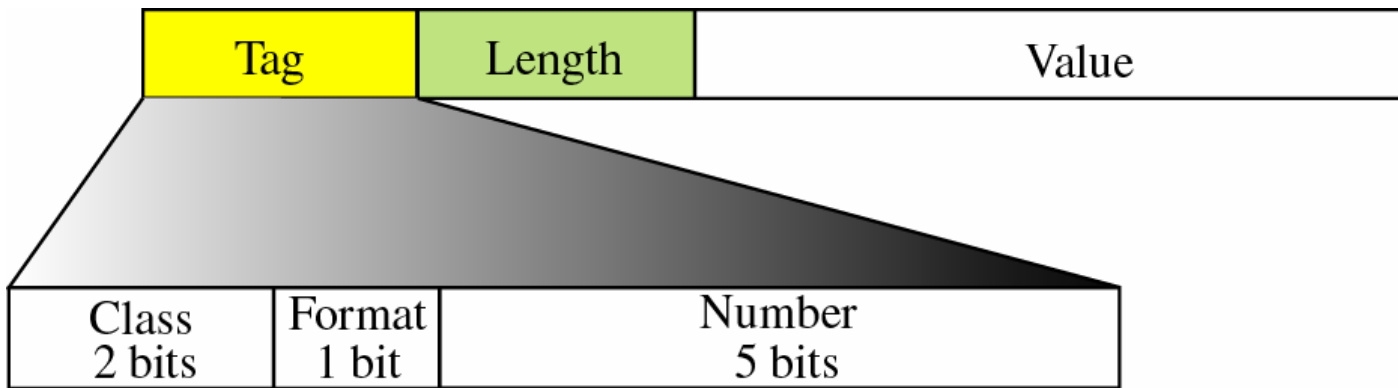


b. Sequence of (simple variables)



d. Sequence of (sequences)

SMI uses another standard called as **Basic Encoding Rules (BER)** to encode data to be transmitted over the network as shown below.



**Tag:** 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)

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 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

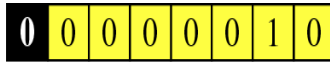
The **format** subfield indicates, whether the data is simple (0) or structured (1).

The number subfield further divides simple or structured data into sub-groups.

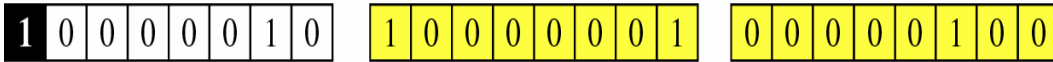
**Length:** The length field is 1 or more bytes.

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.



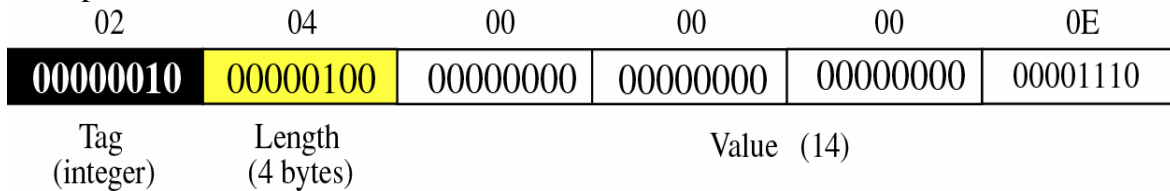
a. The shaded part defines the length (2)



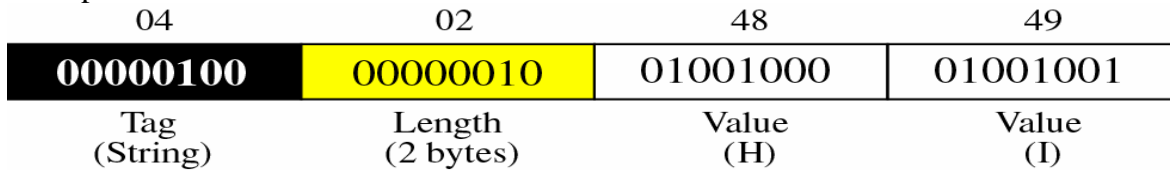
b. The shaded two bytes define the length (260)

**Value:** The value field codes the value of the data according to the rules defined in BER.

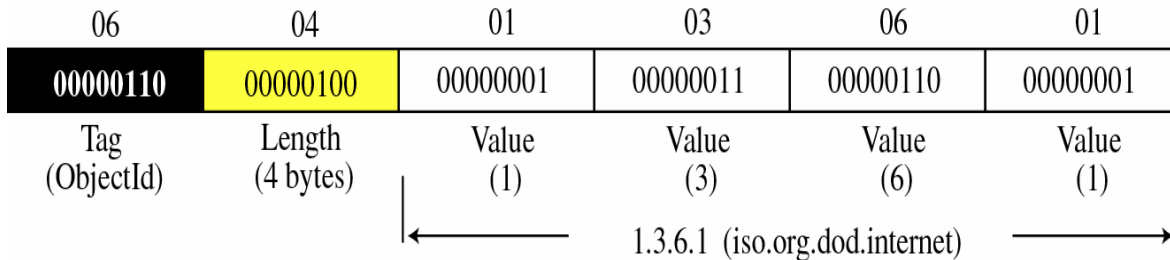
Example 1: To define INTEGER 14



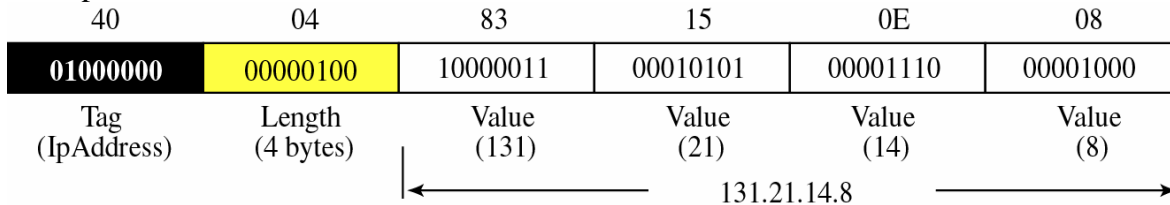
Example 2: To define the OCTET STRING "HI"



Example 3: To define Object Identifier 1.3.6.1 (iso.org.dod.internet)



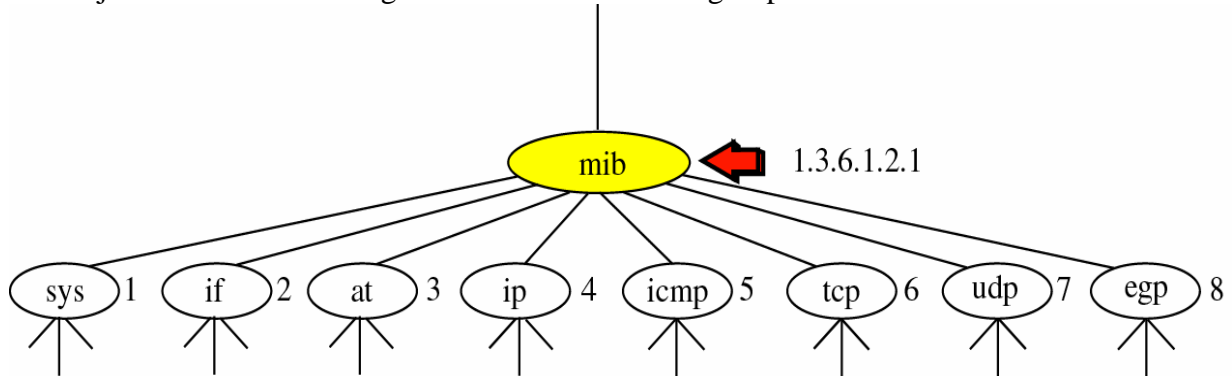
Example 4: To define the IP Address 131.21.14.8



## MIB (Management Information Base)

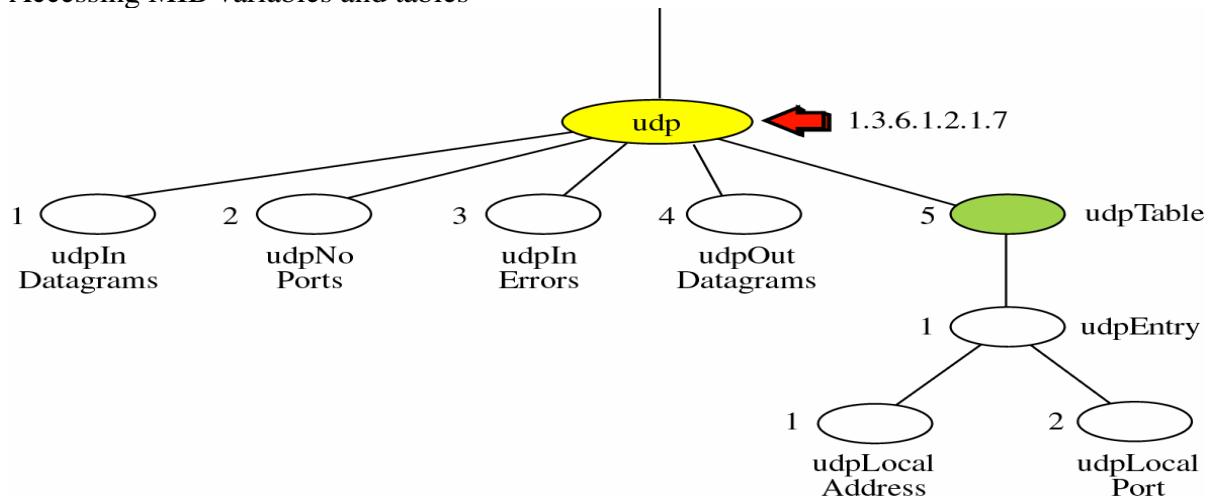
The management Information base is the second component of Network Management. **Each agent has its own MIB, which is a collection of all the objects that the manager can manage. These objects correspond to attributes of the resources under his control.**

The objects in MIB2 are categorized into ten different groups.



Each group has defined variables and tables.

Accessing MIB variables and tables




To access simple variables like


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 there values, 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

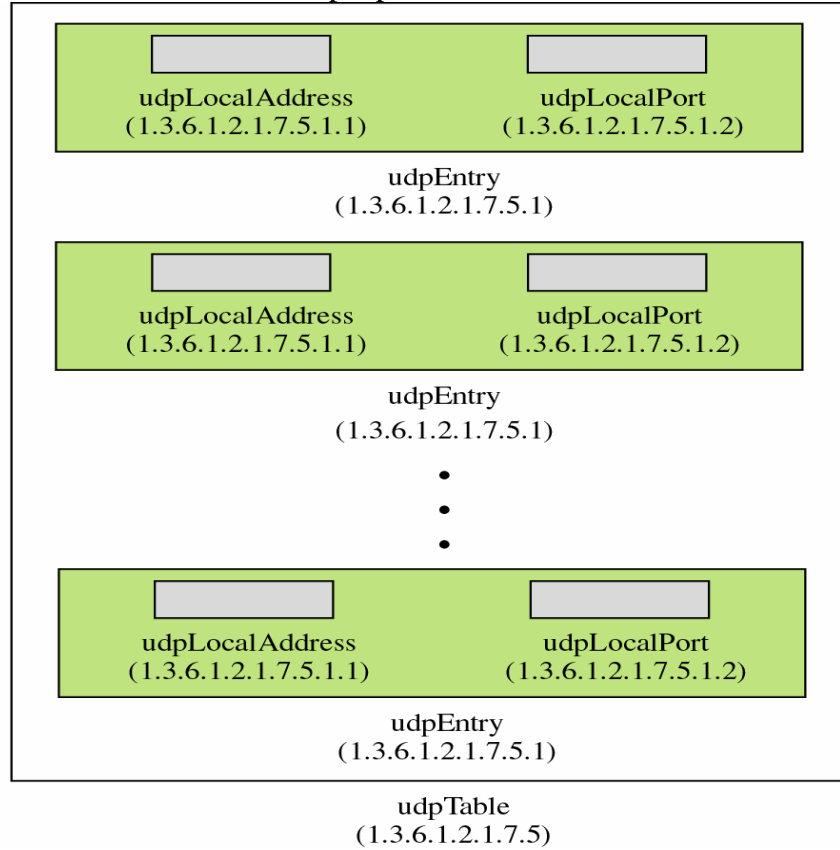
To access a table, for example for udpLocalAddress there must be many udpLocalPort, because corresponding to one address there are multiple port, so that form a table.

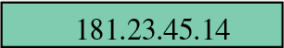
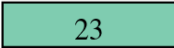
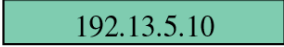
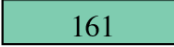
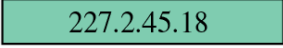

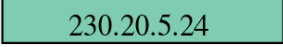
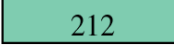
  
 udpInDatagrams  
 (1.3.6.1.2.1.7.1)

  
 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)

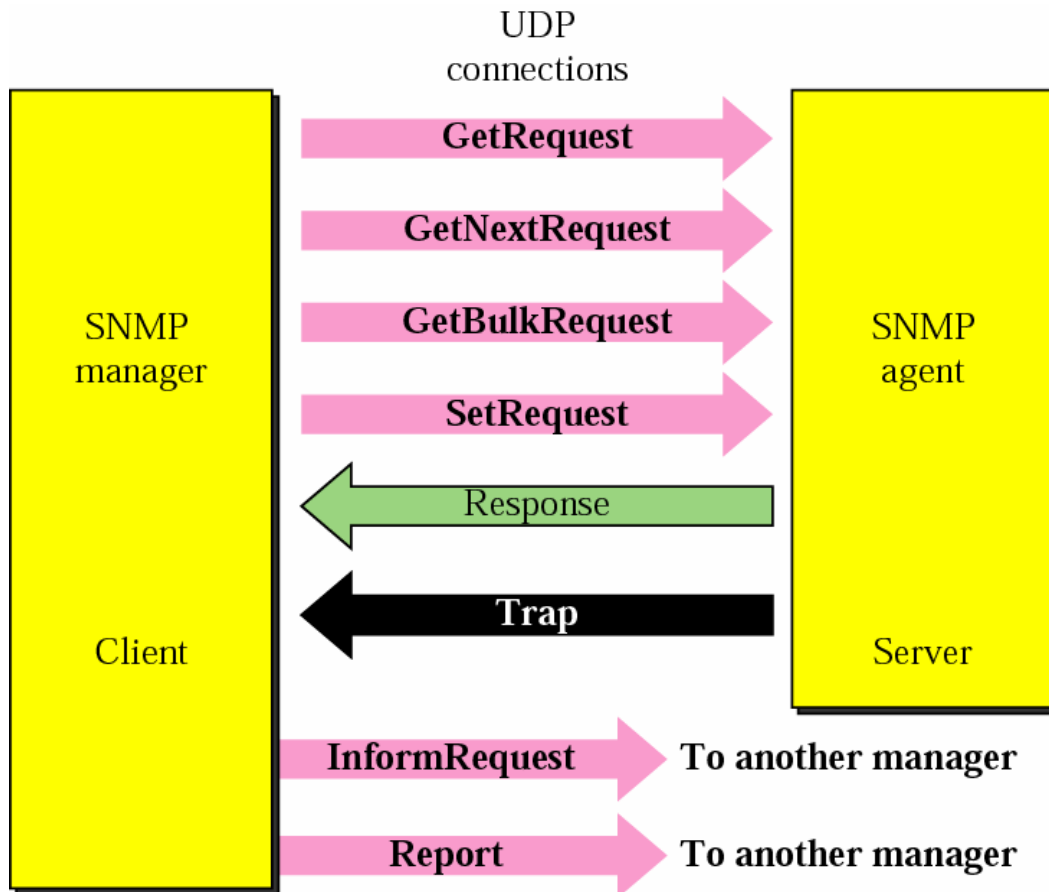


 181.23.45.14	 23
1.3.6.1.2.1.7.5.1.1.181.23.45.14.23	1.3.6.1.2.1.7.5.1.2.181.23.45.14.23
 192.13.5.10	 161
1.3.6.1.2.1.7.5.1.1.192.13.5.10.161	1.3.6.1.2.1.7.5.1.2.192.13.5.10.161
 227.2.45.18	 180
1.3.6.1.2.1.7.5.1.1.227.2.45.18.180	1.3.6.1.2.1.7.5.1.2.227.2.45.18.180
 230.20.5.24	 212
1.3.6.1.2.1.7.5.1.1.230.20.5.24.212	1.3.6.1.2.1.7.5.1.2.230.20.5.24.212



## SNMP (Simple Network Management Protocol)

SNMP uses both SMI and MIB in network management. As told earlier, SNMP define the rules or protocol for exchange of information between the SNMP manager and SNMP Client as shown below.



**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:** The GetBuldRequest 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.

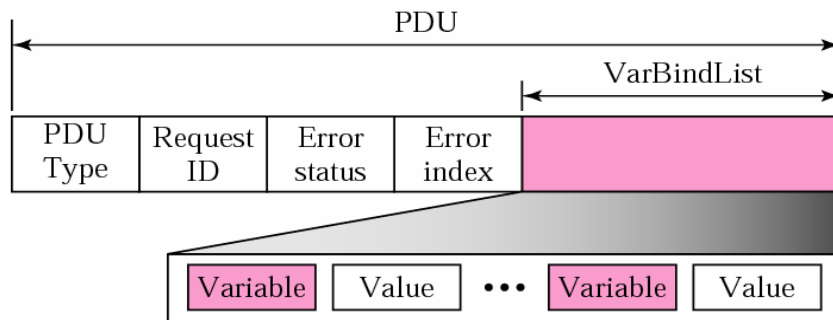
**Response:** The response PDU is sent from an agent to the manager in response to GetRequest or GetNextRequest. It contains the values of variables requested by manager.

**Trap:** This PDU is sent from an agent to the manager to report an unexpected event.

**Report:** The report PDU is designed to report some types of errors.

**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.

## SNMP PDU



### 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.

1. PDU type: This field defines the type of PDU as shown below.

Data	Class	Format	Number	Whole Tag (Binary)	Whole Tag (Hex)
GetRequest	10	1	0000	10100000	A0
GetNextRequest	10	1	00001	10100001	A1
GetResponse	10	1	00010	10100010	A2
SetRequest	10	1	00011	10100011	A3
Trap	10	1	00100	10100100	A4

2. 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.
3. 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.
4. Error Index: The error index is an offset that tells the manager which variable caused the error
5. 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.



This document was created with Win2PDF available at <http://www.daneprairie.com>.  
The unregistered version of Win2PDF is for evaluation or non-commercial use only.