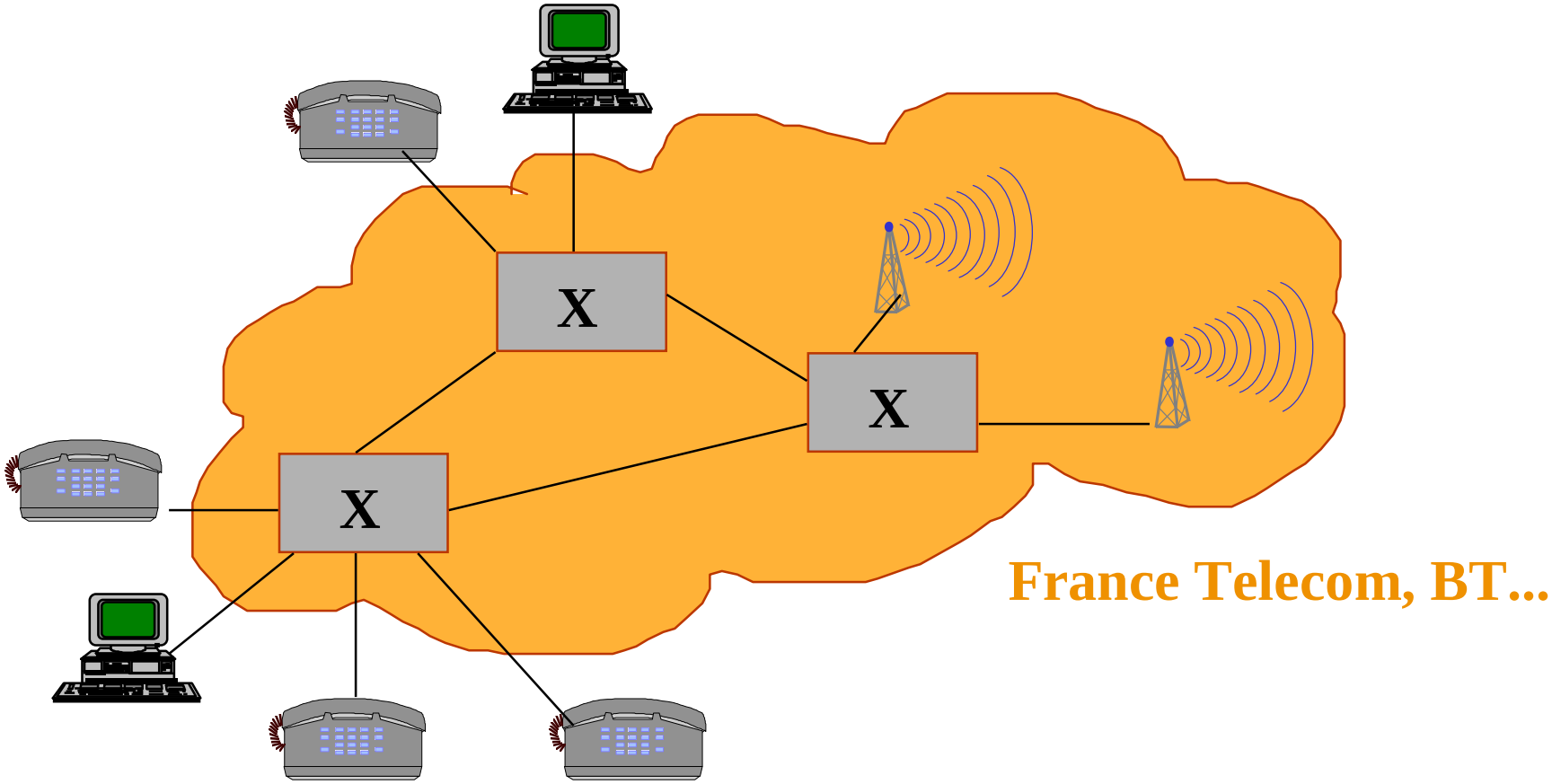
SNMP Overview

Jean-Luc Ernandez
http://dess.ernandez.com
Jean-Luc.Ernandez@AtosOrigin.com

Outline

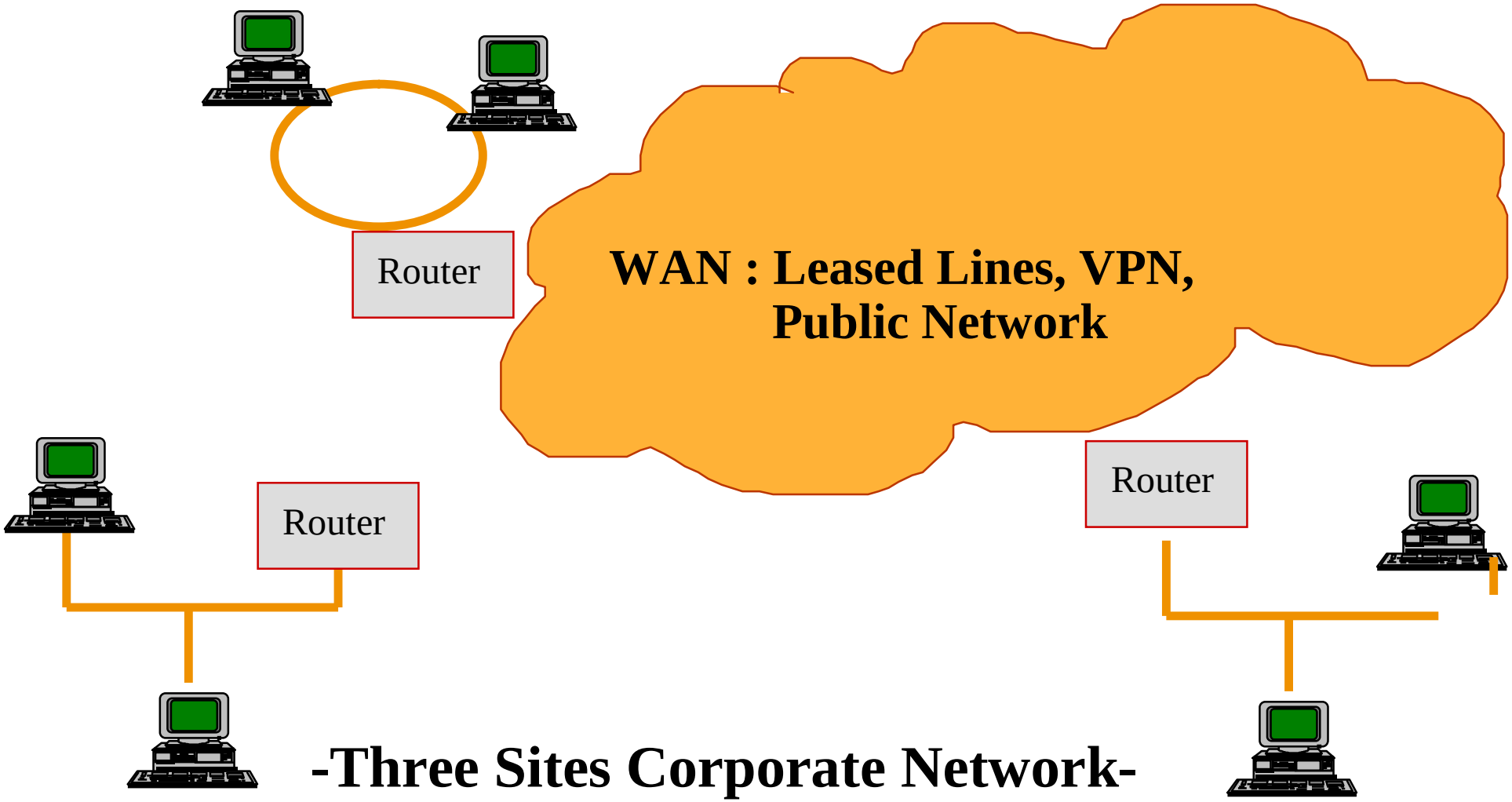
- **A Network Management Definition**
- The SNMP History
- Key Management Concepts
- SNMP Information Modeling
- SNMP Protocol
- Security Features

Networks (1/2)



-Typical Public Network Configuration-

Networks (2/2)



-Three Sites Corporate Network-

Need for Standardized Network Management

Users/Customers

+ End-to-end Availability
+ Flexibility
+ Quality of Service

Network Operators

+ Increasing Size of Networks
+ Technological Heterogeneity
+ Multivendor Environment
+ Evolutivity of Networks

There is a need for managing automatically the target networks thanks to recognized standards (i.e., planning, organizing, monitoring, accounting and controlling resources and activities).

Management Functional Areas

What – Which - When

- Fault Management : Detection, isolation, correction of abnormal operation in the target network
- Configuration Management : Initialization and further reconfiguration of networks and/or network elements
- Performance Management : Control effectiveness of communication activities at various levels of concerns
- Accounting Management : Enables to charge for the usage of the network resources
- Security Management : Protection of the target network integrity (including the management system itself)

What Can be Managed ?

What – Which - When

- Network Elements
- Network (seen as a whole logical entity)
- Services (as provided to the users/customers)
- Business Activities and Policies

TimeFrame of Management Activities

What – Which - When

- Short Term : Alarms management
- Mean Term : Monthly Billing
- Long Term : Planning of future network evolution based on statistics and simulation

Management Activities

	Fault	Config.	Performance	Accounting	Security
Business		• Planning • Ordering		• Pricing	
Service		• Inventory • Traffic Mgt.	• QoS Mgt.	• Billing	• Authenti-cation
Network			• Performance Monitoring and Analysis	• Charging	• Network Integrity
Network Element	• Alarm Mgt., • Trouble Tickets, • Tests	• Activation • Reconfi-guration			

Outline

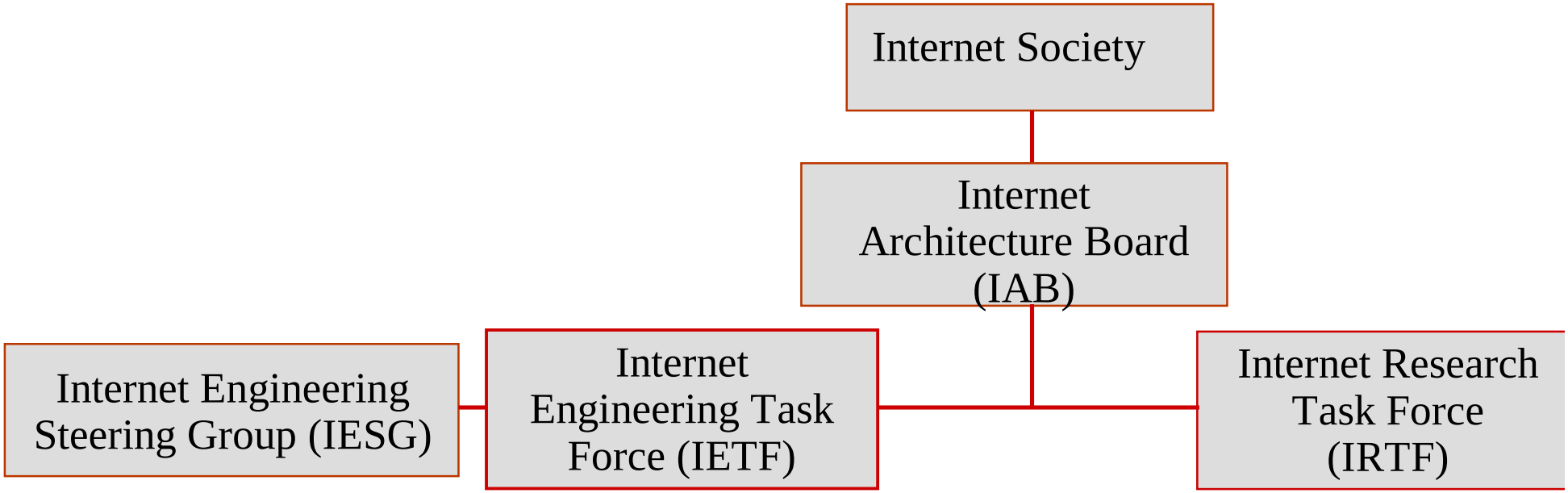
- A Network Management Definition
- **The SNMP History**
- Key Management Concepts
- SNMP Information Modeling
- SNMP Protocol
- Security Features

Approaches for Implementing Network Management

- **Proprietary :** - e.g., IBM Netview (early versions)
- **CMIP (OSI) :**
  - Manages any type of network
  - Functionally rich
  - Complex (==> Expensive)
- **SNMP (TCP/IP) :**
  - For TCP/IP based networks
  - Functionally limited
  - Simple, cheap and widespread
- **IEEE :** - For LAN and MAN management

Internet/SNMP Standardisation Process

- SNMP Standardised by the Internet Community



- Process : Fast, Open, Experimental

- Free Availability of Standards (RFCs)

SNMP “Components”

- **MIB (Management Information Base)**

  Database where ‘manageable’ objects are defined.

- **SMI (Structure of Management Information)**

  Information that explain “How to write/define a MIB”

- **Protocol**

  How to exchange information

SNMP Development History

SNMP v1
(RFC 1157)

SMI v1
(RFC 1155)

"MIB 1/I"
(RFC 1156)

"MIB 2/II"
(RFC 1213)

SNMP v2 Standards

(8 RFC : 1901 to 1908)

MIB for SNMP v2

SMI v2

Divergent SNMP v2 Standards

SNMP v3 Standards ?

1989 / 1990 | **1991 / 1992** | **1993** | **1996** | **1998** | **TODAY**

SNMP V1 RFC References

RFC 1155 : Structure of management information (SMI)

RFC 1157 : SNMP protocol

RFC 1212 : Concise MIB definitions

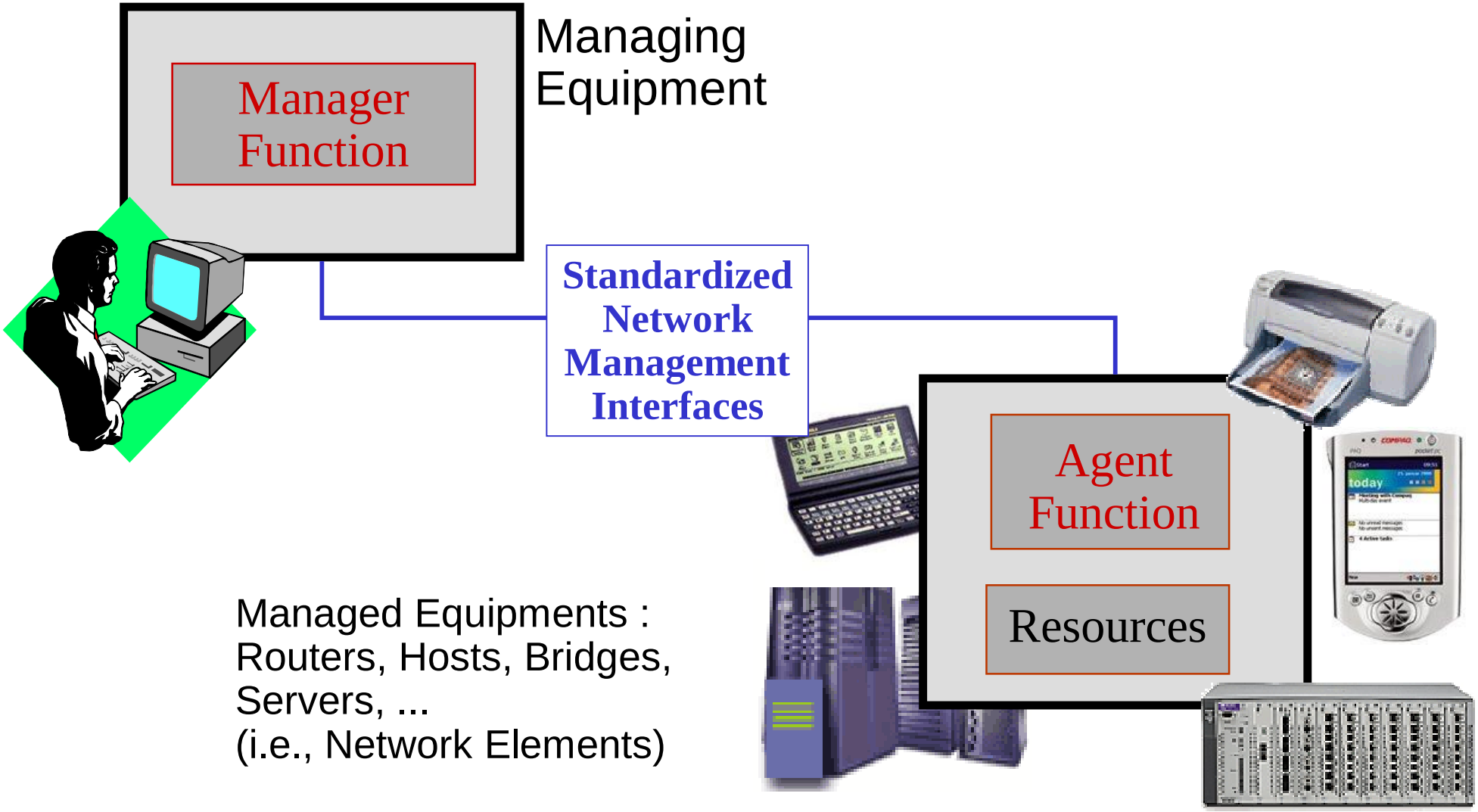
RFC 1213 : MIB-II

RFC 1227 : SMUX

Outline

- A Network Management Definition
- The SNMP History
- **Key Management Concepts**
- SNMP Information Modeling
- SNMP Protocol
- Security Features

Managers and Agents



Managed Equipments :
Routers, Hosts, Bridges,
Servers, ...
(i.e., Network Elements)

Resources, Managed Objects, MIB (1/5)

How do we Model the Management Information ?

Network Management World | **«Real» World**

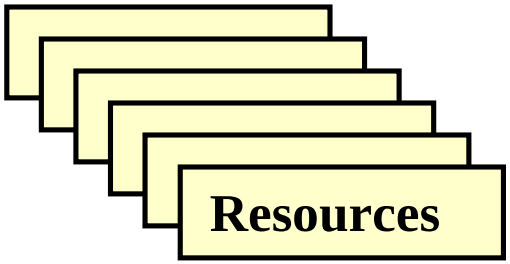
Manager | **Agent**

Resources, Managed Objects, MIB (2/5)

How do we Model the Management Information ?

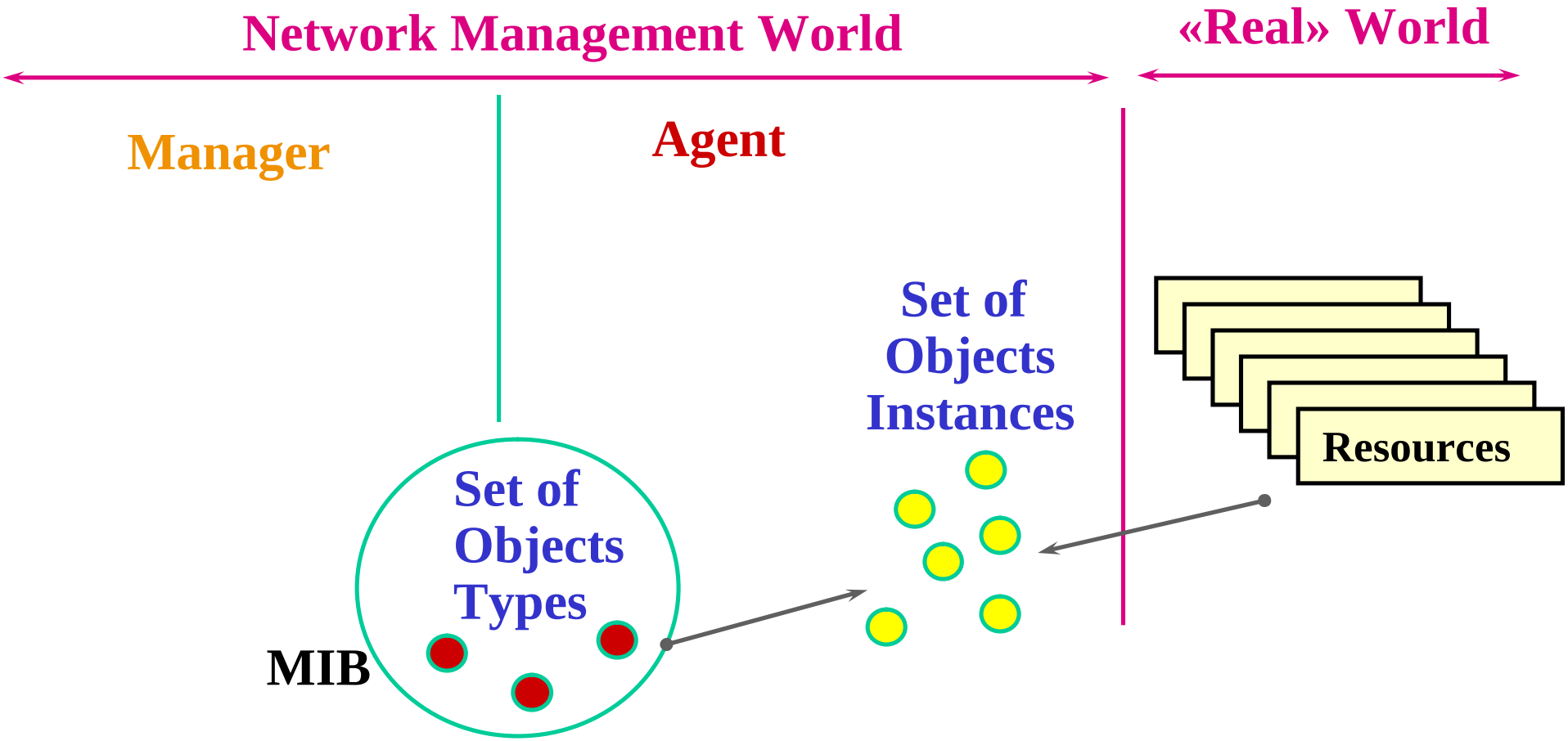
Network Management World | **«Real» World**

Manager | **Agent**



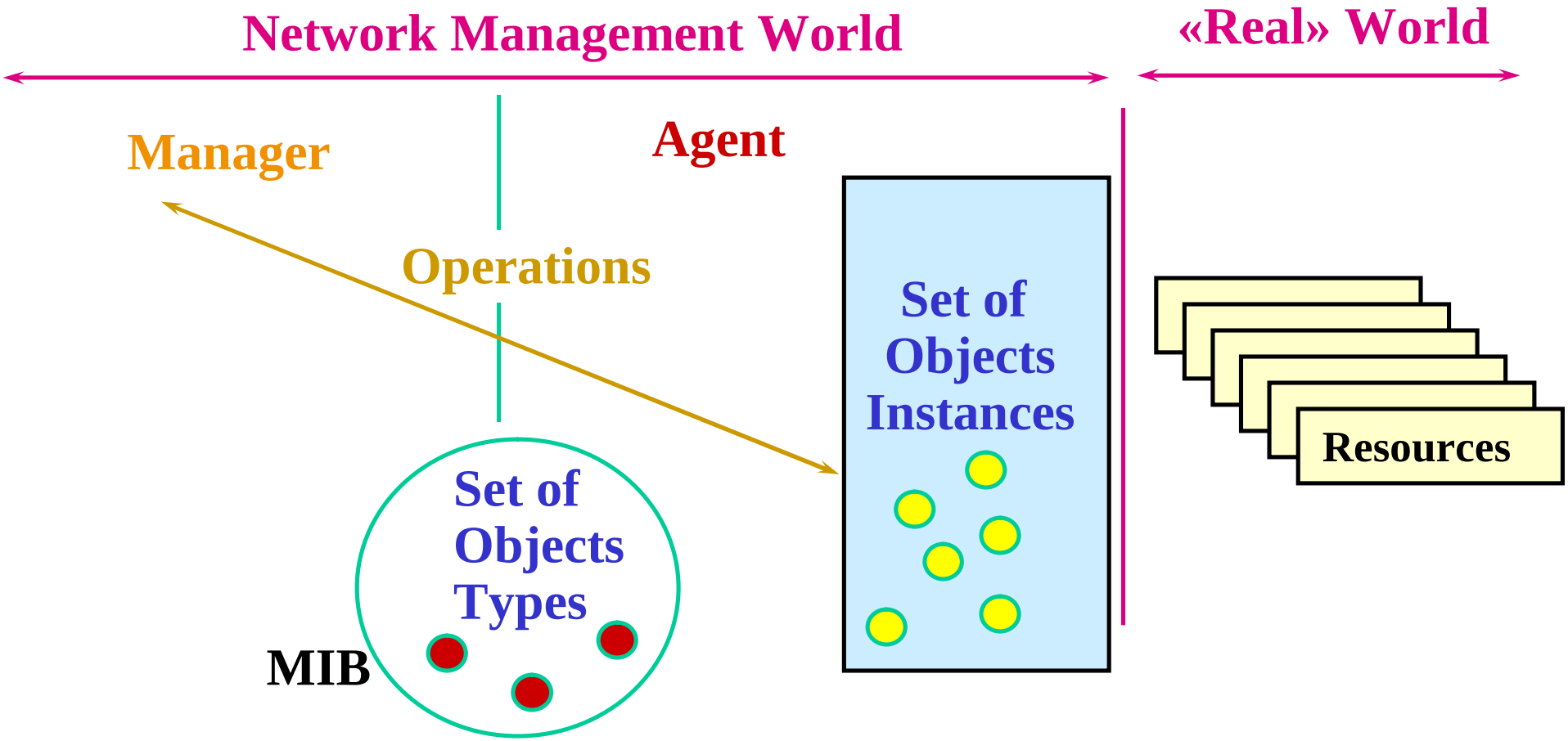
Resources, Managed Objects, MIB (3/5)

How do we Model the Management Information ?



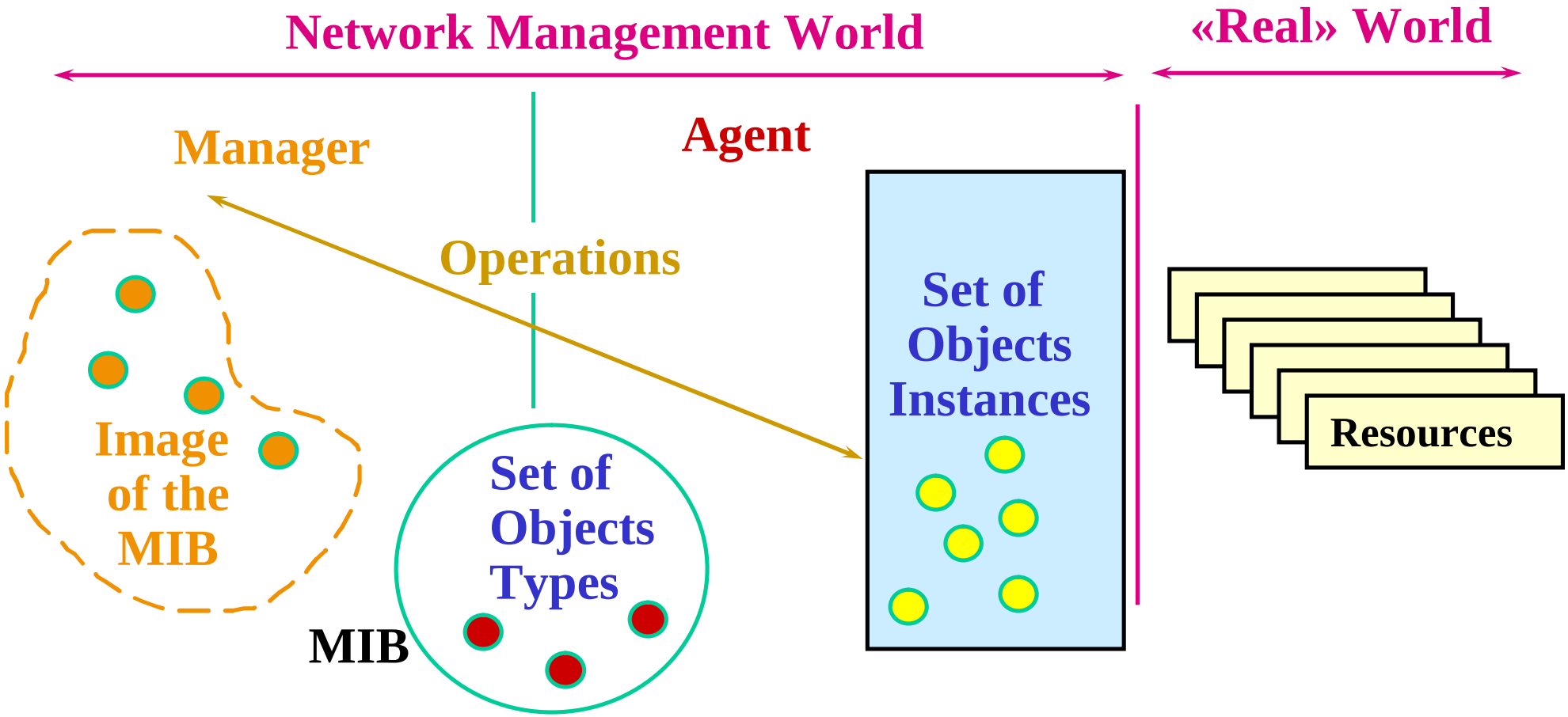
Resources, Managed Objects, MIB (4/5)

How do we Model the Management Information ?



Resources, Managed Objects, MIB (5/5)

How do we Model the Management Information ?



Outline

- A Network Management Definition
- The SNMP History
- Key Management Concepts
- **SNMP Information Modeling**
- SNMP Protocol
- Security Features

Structure of Management Information (1/2)

- **How do we Define the Objects Types ?**
  - Subset of the ASN.1 Notation
  - Specific ASN.1 Types Defined for Describing Objects Types
  - Simple or Tabular Object Types
  - Access Rights
- **How do we Identify Unambiguously Each Object Type ?**
  - International Registration Scheme

Structure of Management Information (2/2)

- **How Managers Name Each Object Instance they Want to Access ?**
  - Access to the Target Network Equipment Agent Thanks to its Network Address
  - Identification of the Type of the Required Object Instance (Simple Type)
  - Identification of the Type and the Instance Index for the Required Object Instance (Tabular Type)

Management Information Bases (1/3)

- MIB-II

  defines a minimal object subset that:

  - may be common to all equipments
  - adapted to routers administration
  - encourage the development of private MIBs

Management Information Bases (2/3)

Apprx. 170 Object Types / 10 Groups of Objects Types

- System
- Interfaces
- Address Translation
- IP
- ICMP
- TCP
- UDP
- EGP
- Transmission
- SNMP

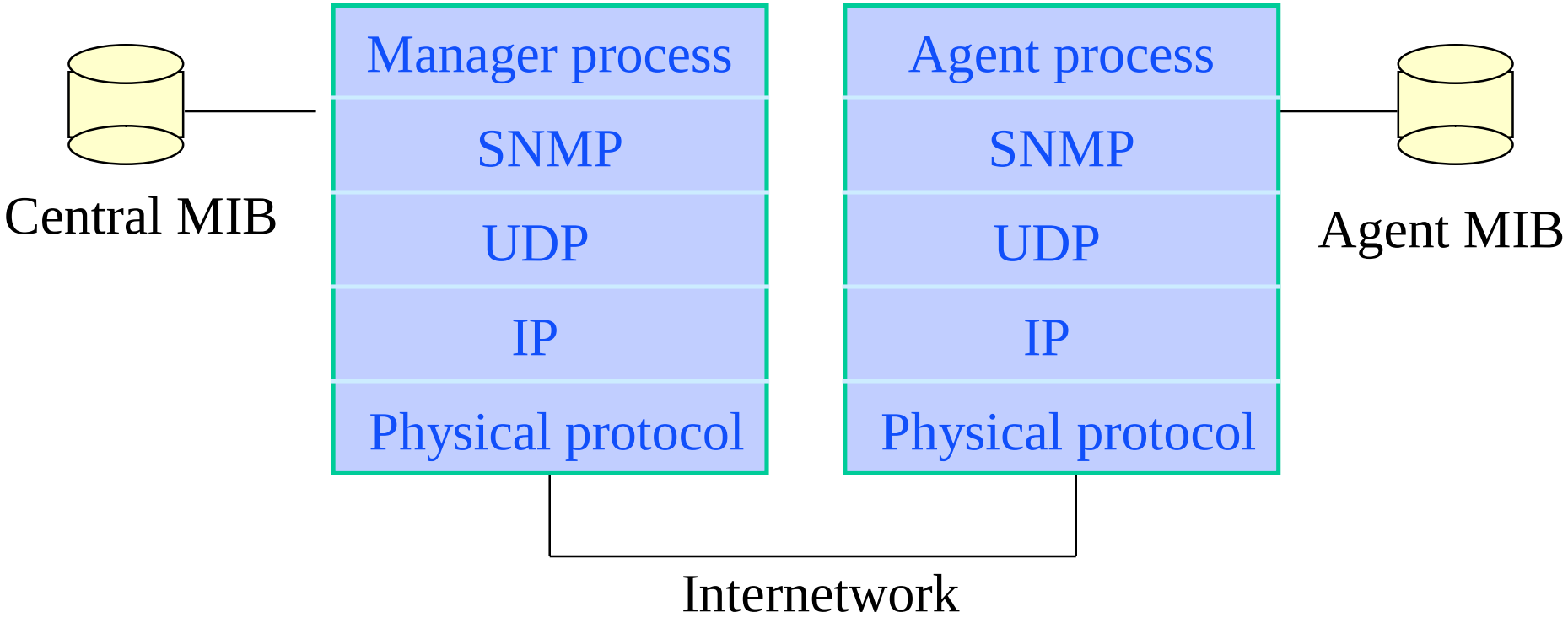
Management Information Bases (3/3)

- Interface Specific MIBs (Under Transmission)
  - Ethernet
  - Token-Ring
  - FDDI
  - Modem…
- RMON MIB
- Private MIBs
  - To be User Defined

Outline

- A Network Management Definition
- The SNMP History
- Key Management Concepts
- SNMP Information Modeling
- **SNMP Protocol**
- Security Features

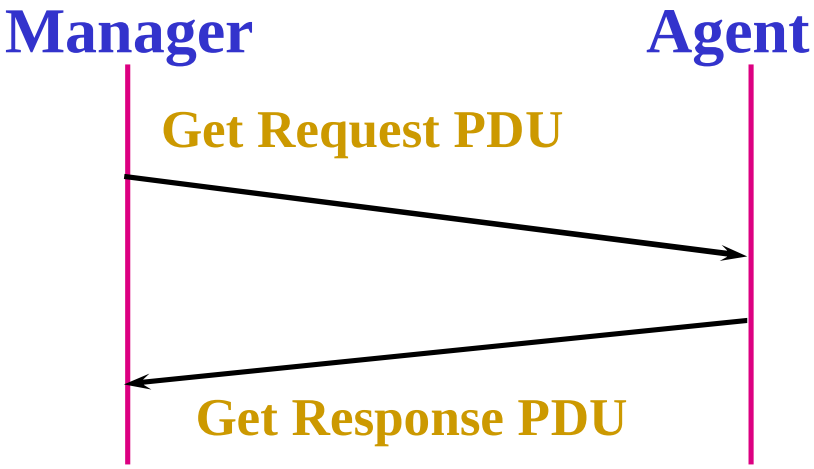
SNMP and IP

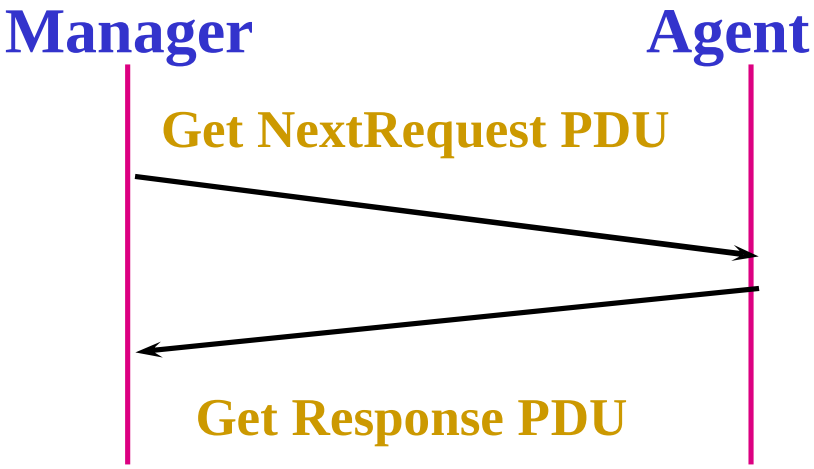

Central MIB
Manager process
SNMP
UDP
IP
Physical protocol
Agent process
SNMP
UDP
IP
Physical protocol
Agent MIB
Internetwork

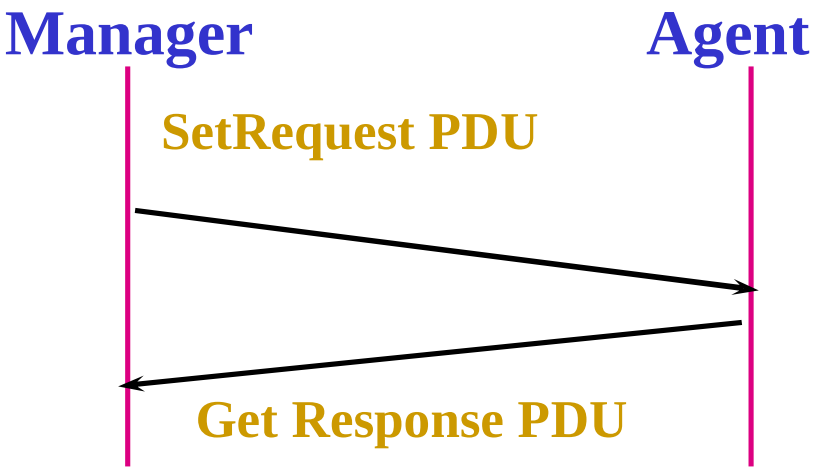
SNMP Protocol

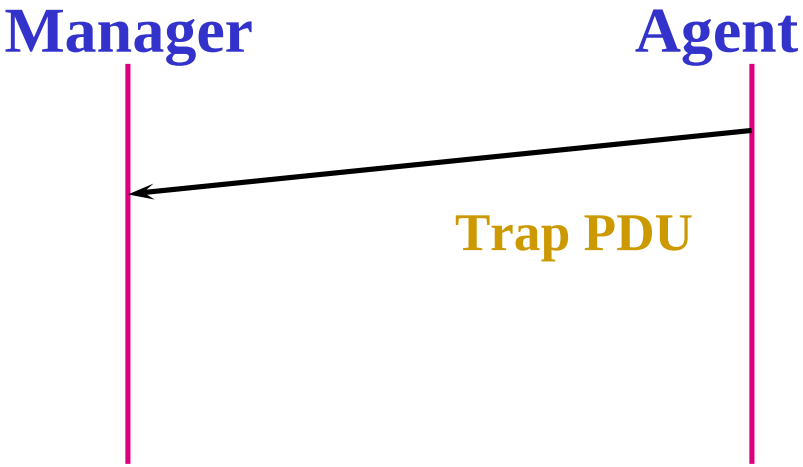
- Objective : Support the Manager-Agent Asymetric Dialog About the Status of Object Instances in the MIB.

SNMP v1 Protocol





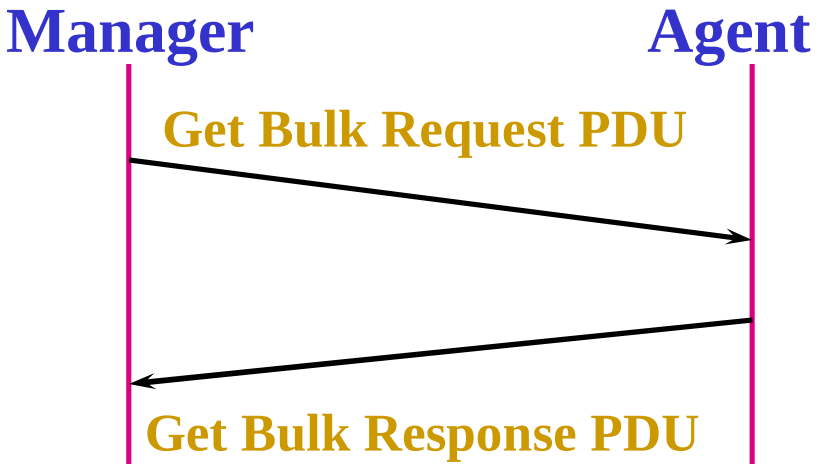


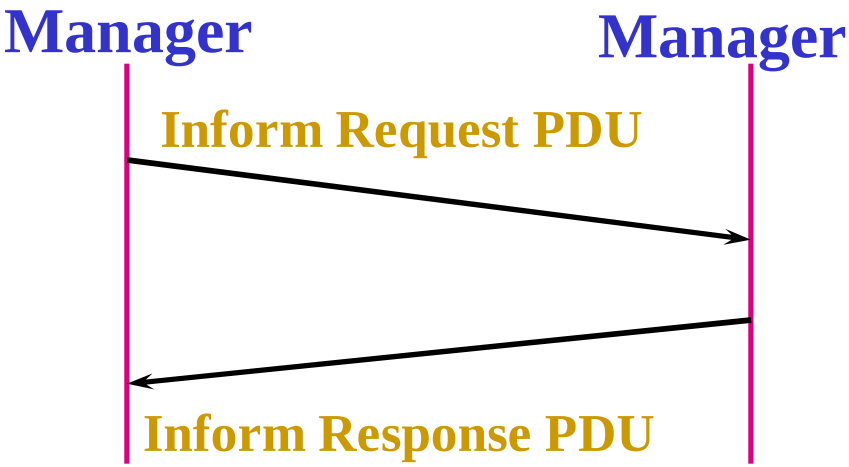


SNMP v2 Protocol

SNMP v2 = SNMP v1 +

- **New Services/PDUs**
- **Security**
- **Manager to Manager Communication**
- **Synchronisation of Managers**





Outline

- A Network Management Definition
- The SNMP History
- Key Management Concepts
- SNMP Information Modeling
- SNMP Protocol
- **Security Features**

Security Aspects of SNMP

- Communities
  - Defined locally by each Agent as :
    (Community Name, Access Rights on local
    MIB Object Instances)
  - Provide Basic Authentication Scheme
  - Access Right Control to MIB objects

- Data Encryption Mechanisms (SNMP v2)

SNMP v1
Structure of Management Information

Outline

- **Definition and Goals of the Structure of Management Information (SMI)**
- MIB Structure
  - The Internet Naming Hierarchy
  - Objects Types
  - Simple/Tabular Objects
  - Instances Identification
- MIB Syntax
  - The Abstract Syntax Notation One (ASN.1)
  - Objects Definition
  - Tables Definition
  - Traps Definition

Definition and Goals (1/2)

- The SMI provides a standardised way for defining a MIB
  - defining the structure of a particular MIB
  - defining the managed objects (syntax and value)
  - encoding object values
- The SMI avoids complex data types:
  - to simplify the task of implementation
  - to enhance interoperability

the MIB can store only scalars and two-dimensional arrays of scalars

Definition and Goals (2/2)

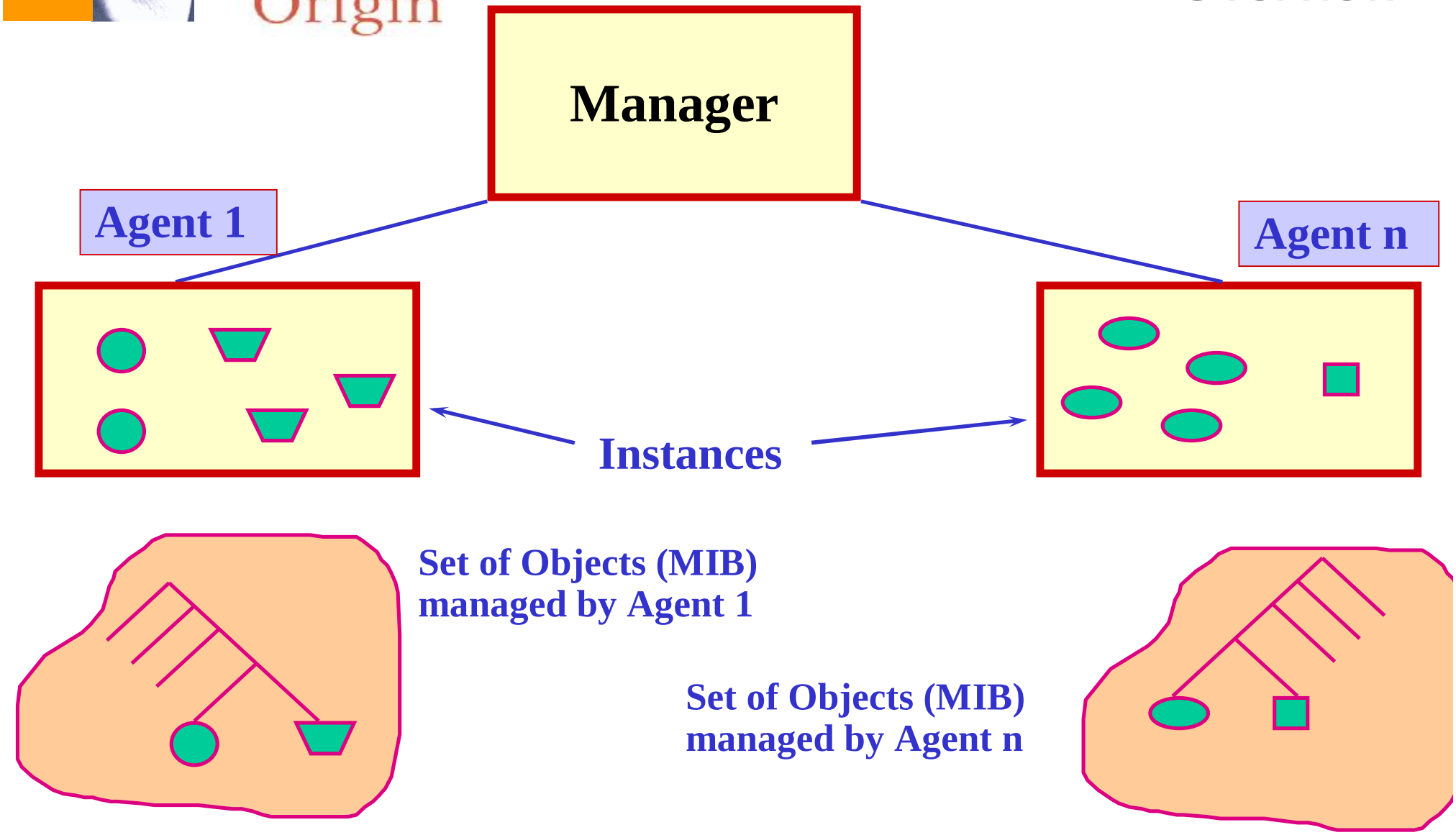
- A subset of the ASN.1 notation is used to describe the managed objects as well as the entire MIB structure

- The SMI is specified in RFC 1155

Outline

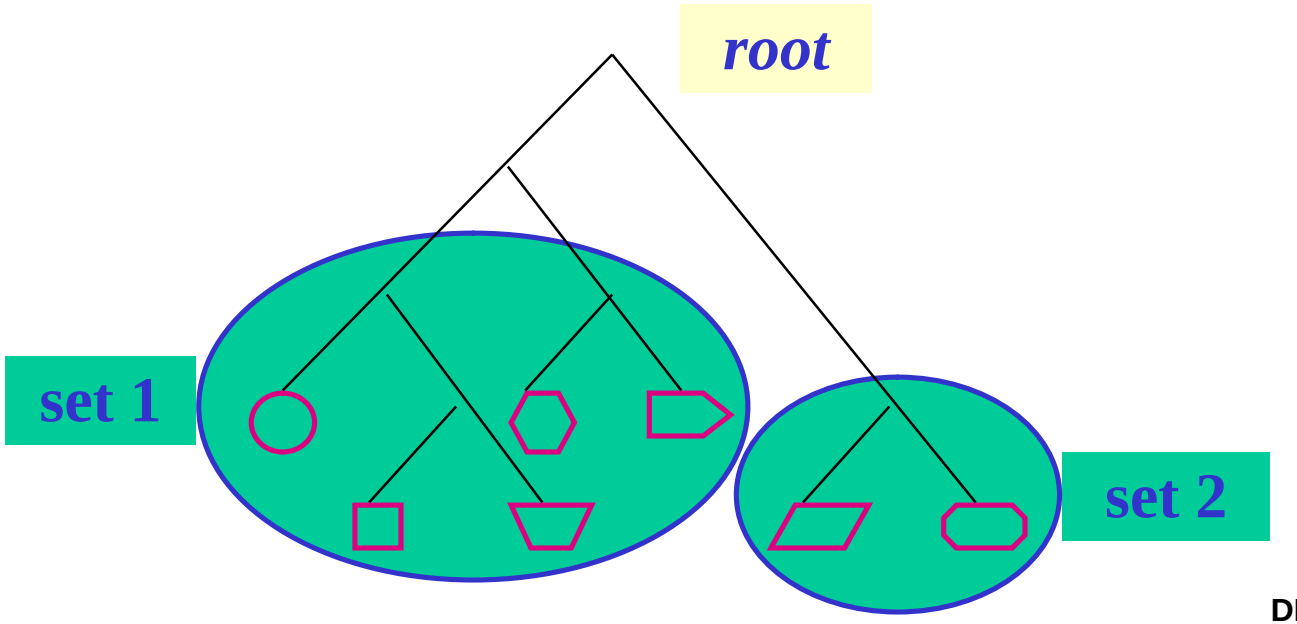
- Definition and Goals of the Structure of Management Information (SMI)
- **MIB Structure**
  - **The Internet Naming Hierarchy**
  - Objects Types
  - Simple/Tabular Objects
  - Instances Identification
- MIB Syntax
  - The Abstract Syntax Notation One (ASN.1)
  - Objects Definition
  - Tables Definition
  - Traps Definition

Overview


Manager
Agent 1
Agent n
Instances
Set of Objects (MIB)
managed by Agent 1
Set of Objects (MIB)
managed by Agent n

The Internet Naming Hierarchy

- Naming of the managed objects is based on a tree structure
- The leaves represent the managed objects
- The intermediate nodes allow to group the objects into logical sets

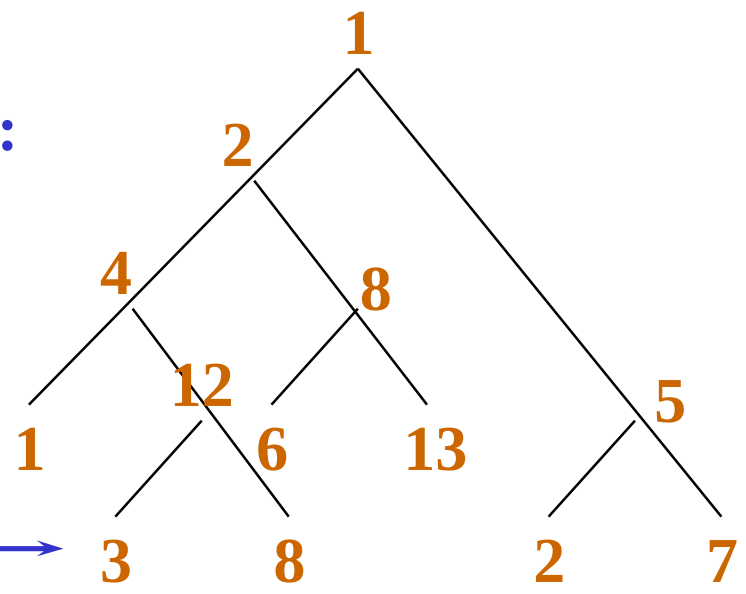


Objects Identification

- Each node is identified by a numerical identifier
- Each object is named by the sequence of the identifiers from the root to the object

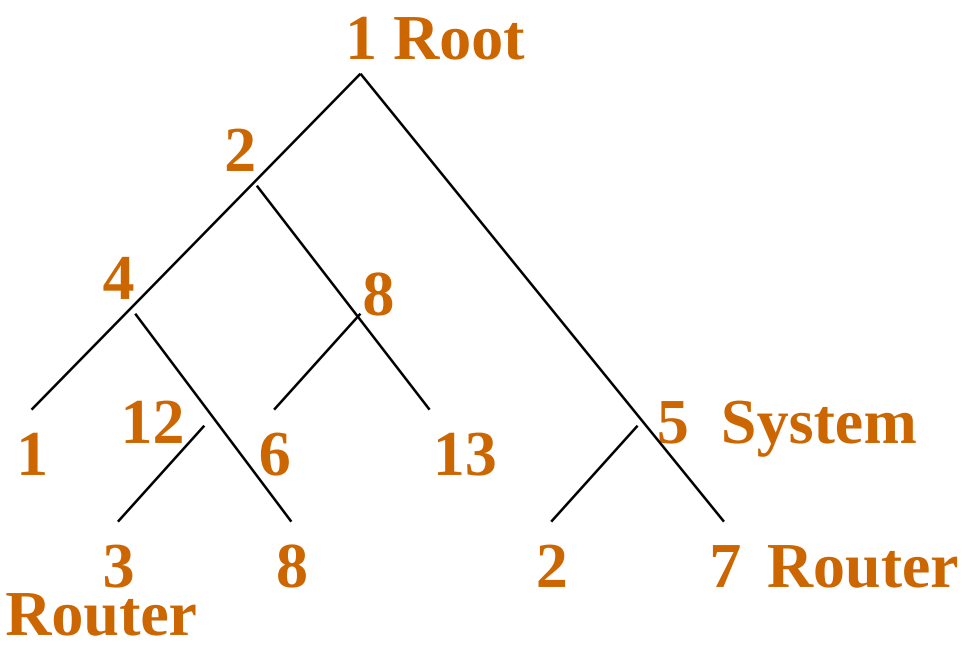
The object identifier is :

1.2.4.12.3



Object Identification (Textual Form)

- A name (string) can be associated to each node
- A name is unique in the context of its "parents"

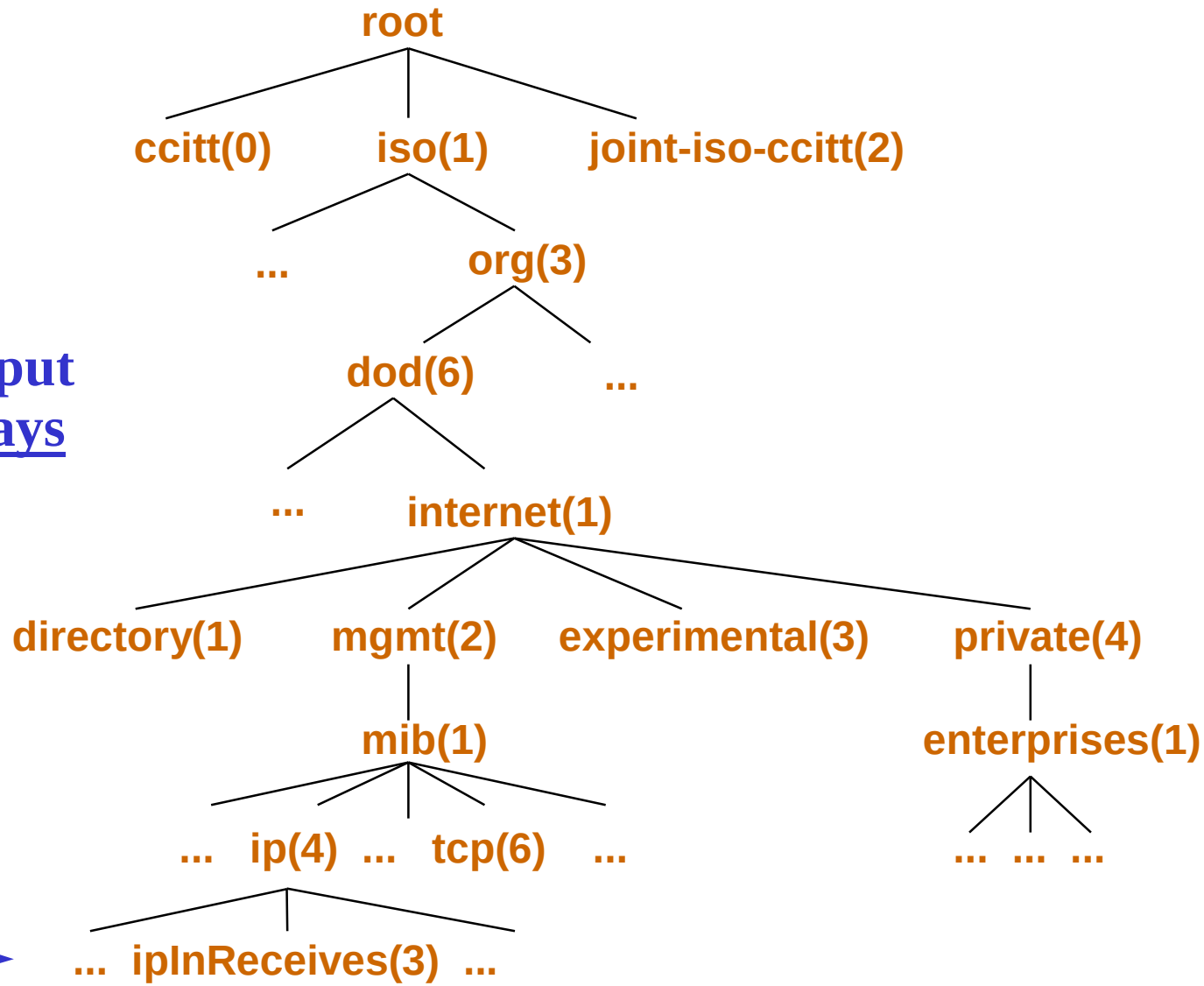


Two ways to named the object :

1.5.7 or Root.System.Router

Internet Registration Hierarchy Example

The number of input datagrams is <u>always</u> identified as 1.3.6.1.2.1.4.3



Outline

- Definition and Goals of the Structure of Management Information (SMI)
- **MIB Structure**
  - The Internet Naming Hierarchy
  - **Objects Types**
  - Simple/Tabular Objects
  - Instances Identification
- MIB Syntax
  - The Abstract Syntax Notation One (ASN.1)
  - Objects Definition
  - Tables Definition
  - Traps Definition

Objects Types

- A restricted subset of ASN.1 is used to describe objects types
- Two ASN.1 classes are used :
  - Universal Types (Application Independent)
  - Application-Wide Types :
    - Defined in the context of a particular application
    - Each application, including SNMP, is responsible for defining its own application-wide data types

Universal Types

- **The following data types are permitted :**
  - **<u>Integer</u>** (ex. : 5, -10)
  - **<u>Octet string</u>** (ex. : protocol)
  - **<u>Null</u>** (object with no value associated)
  - **<u>Object identifier</u>** (ex. : 1.3.6.1.2)
- **And the constructor type (used to build tables) :**
  - Sequence, Sequence-of

Application-Wide Types

- **RFC 1155 defines the following application-wide data types :**

  - **<u>Network address</u>, <u>IP address</u> :**

    Internet 32-bit address

  - **<u>Counter</u> :**

    Non-negative integer (can be incremented but not decremented)

Application-Wide Types

- **Gauge** :

  Non-negative integer that may increase or decrease

- **Timeticks** :

  Non-negative integer counting the time in hundredths of second

- **Opaque** :

  Arbitrary data transmitted in the form of an octet string

Outline

- Definition and Goals of the Structure of Management Information (SMI)
- **MIB Structure**
  - The Internet Naming Hierarchy
  - Objects Types
  - **Simple/Tabular Objects**
  - Instances Identification
- MIB Syntax
  - The Abstract Syntax Notation One (ASN.1)
  - Objects Definition
  - Tables Definition
  - Traps Definition

Simple/Tabular Objects (1/2)

The SMI supports two forms of objects : Simple or Tabular

- **Simple Objects** :
  - Object with a unique instance within the agent.
  - Its type is one of the following : integer, octet string, null, object identifier, network address, IP address, counter, gauge, time ticks or opaque.

Simple Object Example

...

mib**(1)**

ip**(4)**

ipInReceives**(3)**

453201

The ipInreceives object has one instance

Simple/Tabular Objects (2/2)

- **Tabular Objects :**
  - Two-dimensional table containing zero or more rows.
  - Each row is made of one or more simple objects (components).
  - One or more components are used as indexes to unambiguously identifying the rows
  - The definition of tables is based on ASN.1 types "Sequence" and "Sequence-of "ASN.1 type.

Tabular Object Example

mib2**(1.3.6.1.2.1)**

interfaces**(2)**

ifTable**(2)**

ifEntry**(1)**

- **The table is indexed by ifIndex.**
- **Each row is an instance of the ifIndex, ifPhysAddress and ifAdminStatus objects**

ifIndex**(1)**	ifPhysAddress**(6)**	ifAdminStatus**(7)**	
1	**00:00:39:20:04**	**1 (up)**	row **1**
2	**08:00:56:16:11**	**3 (testing)**	row **2**
3	**00:00:b4:02:33**	**2 (down)**	row **3**

Outline

- Definition and Goals of the Structure of Management Information (SMI)
- **MIB Structure**
  - The Internet Naming Hierarchy
  - Objects Types
  - Simple/Tabular Objects
  - **Instances Identification**
- MIB Syntax
  - The Abstract Syntax Notation One (ASN.1)
  - Objects Definition
  - Tables Definition
  - Traps Definition

Instance Identification of Simple Objects

Instance identifier = Object identifier + 0

...

mib**(1)**

ip**(4)**

ipInReceives**(3)**

Object	Instance identifier
ipInReceives	mib.4.3.0

Instance Identification of Table Objects

Instance identifier =

Object identifier.index1value. ... .indexn value

mib2**(1.3.6.1.2.1)**

interfaces**(2)**

ifTable**(2)**

ifEntry**(1)**

ifIndex**(1)** ifPhysAddress**(6)** ifAdminStatus**(7)**

1	00:00:39:20:04	1 (up)
2	08:00:56:16:11	3 (testing)
8	00:00:b4:02:33	2 (down)

Col	Object	Instance identifier
1	ifIndex	
2	ifPhysAddress	if.2.1.6.1 if.2.1.6.2 if.2.1.6.8
3	ifAdminStatus	if.2.1.7.1 if.2.1.7.2 if.2.1.7.8

Outline

- Definition and Goals of the Structure of Management Information (SMI)
- MIB Structure
  - The Internet Naming Hierarchy
  - Objects Types
  - Simple/Tabular Objects
  - Instances Identification
- **MIB Syntax**
  - **The Abstract Syntax Notation One (ASN.1)**
  - Objects Definition
  - Tables Definition
  - Traps Definition

How to Define MIB Objects

How can we define objects to include them in the MIB ?

Abstract Syntax Notation 1 (<u>ASN.1</u>)

What is ASN.1 ?

- **ASN.1 has been standardized by CCITT (X.208) and ISO (ISO 8824)**
- **ASN.1 is a formal language used to define e.g., upper layer protocols**
- **It is used to define :**
  - the abstract syntaxes of application data
  - the structure of application and presentation PDUs
  - the MIBs for both SNMP and OSI system management

ASN.1 Data Types (for SNMP)

SNMP uses two categories of types :

- Simple types : these are atomic types, with no component
- Structured types : a structured type has components

Simple Types

- **Simple types are defined by specifying the set of its values:**

Tag	Type name	Set of values
1	*BOOLEAN*	*true/false*
2	INTEGER	integers
3	BIT STRING	sequence of 0 or more bits
4	OCTET STRING	sequence of 0 or more octets
	...	

Structured Types (Sequence)

- **Sequences are used to define an ordered list of data types :**

```
atTable ::= SEQUENCE OF AtEntry

AtEntry ::= SEQUENCE {
        atIndex          INTEGER,
        atPhysAddress    OCTET STRING,
        atNetAddress     NetworkAddress
}
```

ordered, variable number of elements, all from the same type

ordered list of data types

Outline

- Definition and Goals of the Structure of Management Information (SMI)
- MIB Structure
  - The Internet Naming Hierarchy
  - Objects Types
  - Simple/Tabular Objects
  - Instances Identification
- **MIB Syntax**
  - The Abstract Syntax Notation One (ASN.1)
  - **Objects Definition**
  - Tables Definition
  - Traps Definition

ASN.1 Macro Definitions

- The ASN.1 macro notation allows the user to extend the syntax of ASN.1 to define new types and their values
- The OBJECT-TYPE macro defines the model of SNMP MIB objects
- The MIB objects are instances of this type
- The OBJECT-TYPE macro was initially defined in RFC 1155 (MIB-I) and later expanded in RFC 1212 (MIB-II)

The OBJECT-TYPE Macro

***OBJECT-TYPE* MACRO ::= BEGIN**

TYPE NOTATION ::= «SYNTAX» type (*ObjectSyntax*)
«ACCESS» Access
«STATUS» Status
DescrPart ReferPart
IndexPart DefValPart

VALUE NOTATION ::= value (*ObjectName*)

Access ::= «read-only» | «read-write» | «write-only» | «not-accessible»

Status ::= «mandatory» | «optional» | «obsolete» | «deprecated»

DescrPart ::= «DESCRIPTION» value (*DisplayString*) | empty

ReferPart ::= «REFERENCE» value (*DisplayString*) | empty

IndexPart ::= «INDEX» «{« value (*ObjectName*), ... «}» | empty

DefValPart ::= «DEFVAL» «{« value (*ObjectSyntax*) «}» | empty

END

Key Components (1/4)

- **SYNTAX (*INTEGER, OCTET STRING, OBJECT IDENTIFIER ...*) :**

  the type of an instance of the object

- **ACCESS (*read-only, read-write, write-only, not-accessible*) :**

  the way in which an instance of the object must be accessed via SNMP

Key Components (2/4)

- **STATUS :**

  indicates if the implementation is required for this object

  - ***mandatory*** **:** The agents must implement the object
  - ***optional*** **:** The implementation by the agents is optional
  - ***obsolete*** **:**The agents need no longer implement the object
  - ***deprecated*** **:** The object must be supported, but it will most likely be removed from the next version of the MIB

Key Components (3/4)

- **DESCRIPTION :**

  a textual description of the object

- **REFERENCE :**

  a textual cross-reference to an object defined in some other MIB module

Key Components (4/4)

- **INDEX (used in defining table definition):**

  the INDEX clause determines which object value(s) will unambiguously distinguish one row in the table

- **DEFVAL :**

  defines the default value that may be used when an object instance is created

OBJECT-TYPE Instance Example

rs232InSigName **OBJECT-TYPE**
SYNTAX INTEGER { rts(1), cts(2), dsr(3) }
ACCESS read-only
STATUS mandatory
DESCRIPTION «Identification of a hardware signal»
REFERENCE «EIA Standard RS-232»
::= { rs232InSigEntry 2 }

Outline

- Definition and Goals of the Structure of Management Information (SMI)
- MIB Structure
  - The Internet Naming Hierarchy
  - Objects Types
  - Simple/Tabular Objects
  - Instances Identification
- **MIB Syntax**
  - The Abstract Syntax Notation One (ASN.1)
  - Objects Definition
  - **Tables Definition**
  - Traps Definition

Tables Definition

- **A table is defined using the SEQUENCE OF clause :**

  Table OBJECT-TYPE

  SYNTAX SEQUENCE OF <*Entry*>

  ACCESS ...

- **A row is defined using the SEQUENCE clause :**

  Entry ::= SEQUENCE { <Column1_Descriptor> <Type1>,

  < Column2_Descriptor> <Type2>, ...}

  <ColumnN_Descriptor> is the name of the Nth columnar object of the table

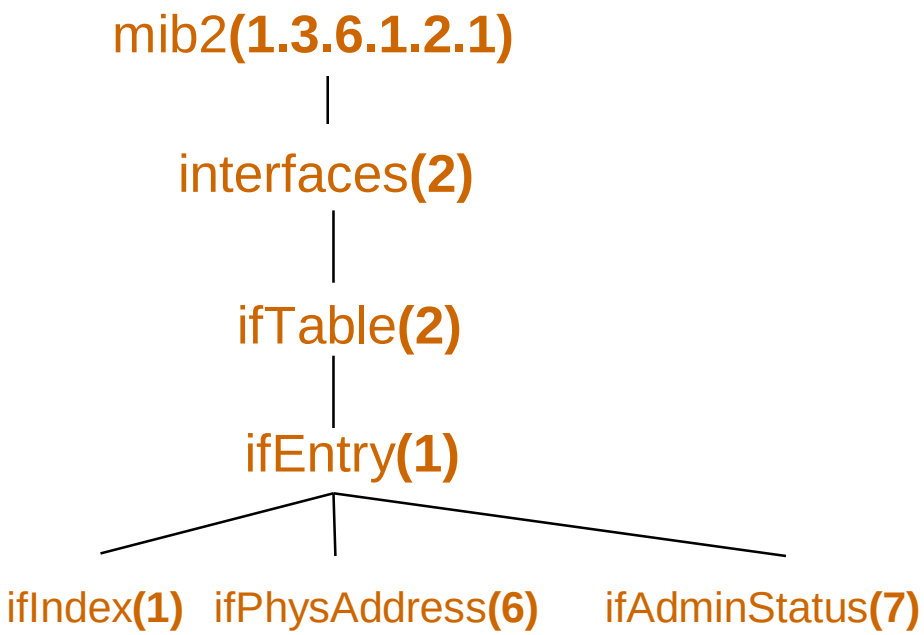
  <TypeN> is the type of the columnar object

Tables Definition Example (1/2)

```
ifTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF IfEntry
        ACCESS  not-accessible
        STATUS   mandatory
        ::= { interfaces 2 }

ifEntry OBJECT-TYPE
        SYNTAX  IfEntry
        ACCESS  not-accessible
        STATUS   mandatory
        INDEX  {ifIndex}
        ::= { ifTable 1 }

IfEntry ::= SEQUENCE {
        ifIndex              INTEGER,
        …
        ifPhysAddress        PhysAddress,
        ifAdminStatus        INTEGER
        …
        }
```



1	00:00:39:20:04	1 (up)
2	08:00:56:16:11	3 (testing)
8	00:00:b4:02:33	2 (down)

Tables Definition Example (2/2)

ifIndex **OBJECT-TYPE**
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
::= { *ifEntry* 1 }

ifPhysAddress **OBJECT-TYPE**
SYNTAX PhysAddress
ACCESS read-only
STATUS mandatory
::= { *ifEntry* 6 }

ifAdminStatus **OBJECT-TYPE**
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
::= { *ifEntry* 7 }

mib2**(1.3.6.1.2.1)**
interfaces**(2)**
ifTable**(2)**
ifEntry**(1)**

ifIndex**(1)**	ifPhysAddress**(6)**	ifAdminStatus**(7)**
1	**00:00:39:20:04**	**1 (up)**
2	**08:00:56:16:11**	**3 (testing)**
8	**00:00:b4:02:33**	**2 (down)**

Outline

- Definition and Goals of the Structure of Management Information (SMI)
- MIB Structure
  - The Internet Naming Hierarchy
  - Objects Types
  - Simple/Tabular Objects
  - Instances Identification
- **MIB Syntax**
  - The Abstract Syntax Notation One (ASN.1)
  - Objects Definition
  - Tables Definition
  - **Traps Definition**

Traps Definition

- **Traps are unacknowledged messages used by agents to notify events to managers**

- **The TRAP-TYPE macro defines the model of SNMP traps (RFC 1215)**

The TRAP-TYPE Macro

***ObjectName* ::= OBJECT IDENTIFIER**

***DisplayString* ::= OCTET STRING**

***TRAP-TYPE* MACRO ::= BEGIN**

TYPE NOTATION ::= «ENTERPRISE» value (OBJECT IDENTIFIER)

VarPart DescrPart ReferPart

VALUE NOTATION ::= value (INTEGER)

VarPart ::= «VARIABLES» «{» VarType, VarType, ... «}» | empty

VarType ::= value (*ObjectName*)

DescrPart ::= «DESCRIPTION» value (*DisplayString*) | empty

Status ::= «REFERENCE» value (*DisplayString*) | empty

END

TRAP-TYPE Key Components (1/2)

- **ENTERPRISE : identification of the management enterprise that generates the trap**

- **VARIABLES : ordered sequence of MIB objects identifiers contained within every trap message**

TRAP-TYPE Key Components (2/2)

- **DESCRIPTION : a textual description of the trap**

- **REFERENCE : a textual cross-reference to an object or trap defined in some other MIB module**

TRAP-TYPE Value

- **The value required in TRAP-TYPE macro is the *Specific* code**

- **It indicates more specifically the nature of the problem and is defined by the management enterprise**

- **Some traps are predefined in RFC 1215 :**

  coldStart*, *warmStart,

  ***linkDown*, *linkUp*,**

  ***authenticationFailure*,**

  egpNeighborLoss

TRAP-TYPE Instance Example

***atos* OBJECT IDENTIFIER ::= { *enterprises* 3629 }**

***myLinkDown* TRAP-TYPE**
ENTERPRISE *atos*
VARIABLES { *ifIndex* }
DESCRIPTION «Failure of a communication link»
::= 2

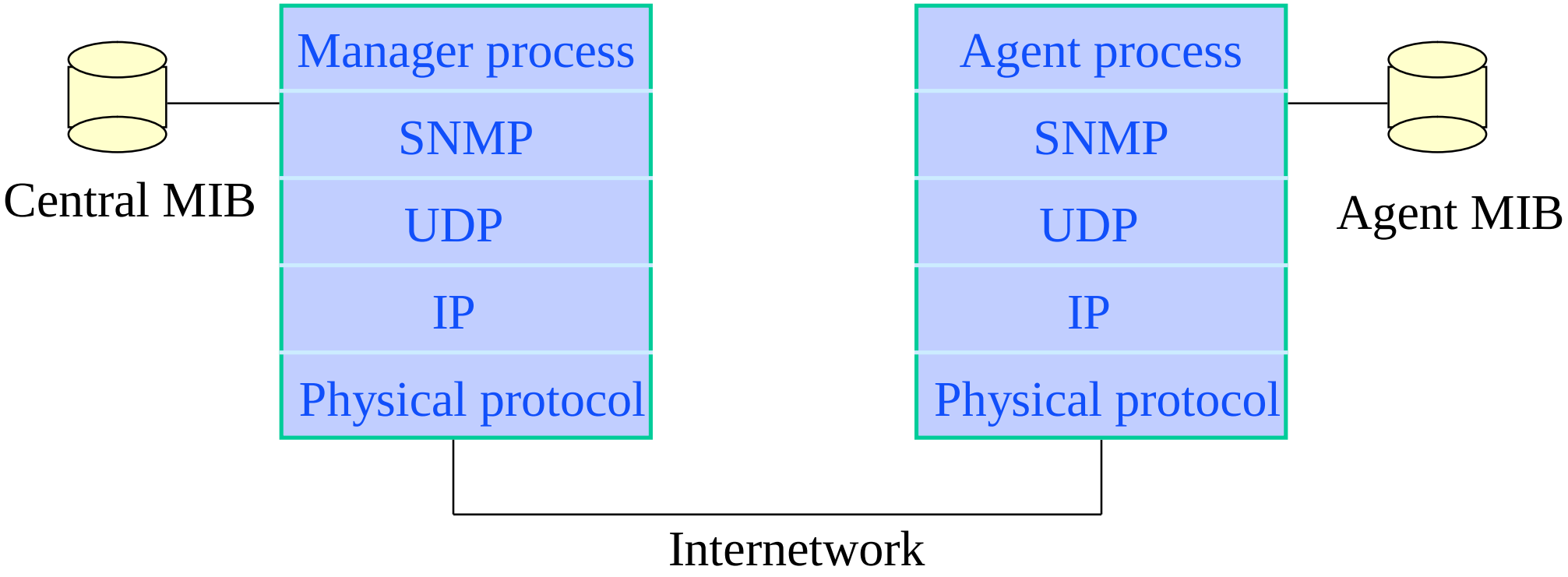
SNMP V1 :
Protocol Description

Outline

- **SNMP Architecture**
- **SNMP Protocol**
  - SNMP Operations
  - SNMP Protocol Data Units
  - SNMP PDUs Format
  - SNMP PDUs Avanced Concepts
  - SNMP PDUs Encoding
- **SNMP Security Mechanisms**

SNMP Architecture

- **SNMP is designed to run on the top of the User Datagram Protocol**



Connectionless Protocol

- Because it uses UDP, SNMP is a connectionless protocol
- No guarantee that the management traffic is received at the other entity
- Advantages :
  - reduced overhead
  - protocol simplicity
- Drawbacks :
  - connection-oriented operations must be built into upper-layer applications, if reliability and accountability are needed

Outline

- **SNMP Architecture**
- **SNMP Protocol**
  - **SNMP Operations**
  - SNMP Protocol Data Units
  - SNMP PDUs Format
  - SNMP PDUs Avanced Concepts
  - SNMP PDUs Encoding
- **SNMP Security Mechanisms**

SNMP Operations

- SNMP provides three simple operations :
  - *GET* :

    Enables the management station to retrieve object values from a managed station
  - *SET* :

    Enables the management station to set object values in a managed station
  - *TRAP* :

    Enables a managed station to notify the management station of significant events
- SNMP allows multiple accesses with a single operation
- Adding and deleting object instances (e.g. in tables) is not normalized by RFC : it is an agent-specific implementation

Outline

- **SNMP Architecture**
- **SNMP Protocol**
  - SNMP Operations
  - **SNMP Protocol Data Units**
  - SNMP PDUs Format
  - SNMP PDUs Avanced Concepts
  - SNMP PDUs Encoding
- **SNMP Security Mechanisms**

SNMP Protocol Data Units

- ***Get Request*** **:**

  Used to obtain object values from an agent

- ***Get-Next Request*** **:**

  Similar to the Get Request, except it permits the retrieving of the next object instance (in lexicographical order) in the MIB tree

- ***Set Request*** **:**

  Used to change object values at an agent

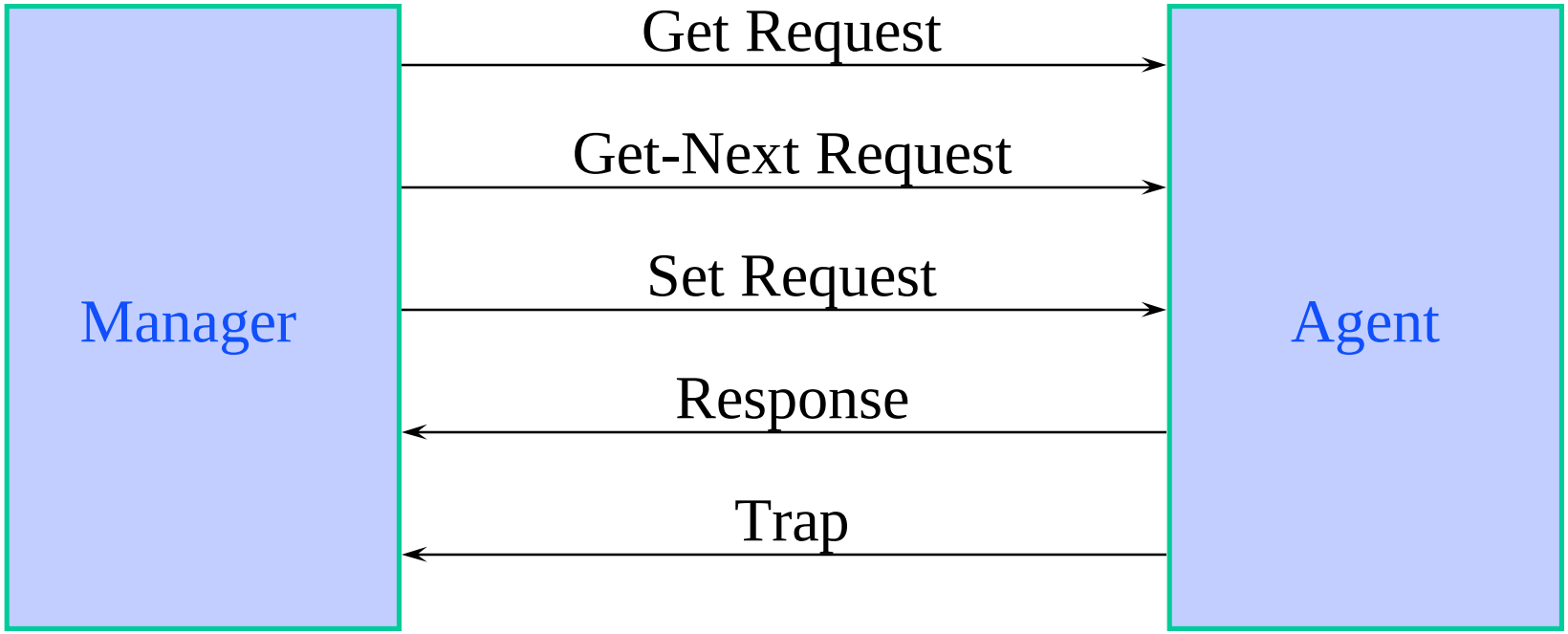
- ***Response*** **:**

  Responds to the Get Request, Get-Next Request and Set Request PDUs

- ***Trap*** **:**

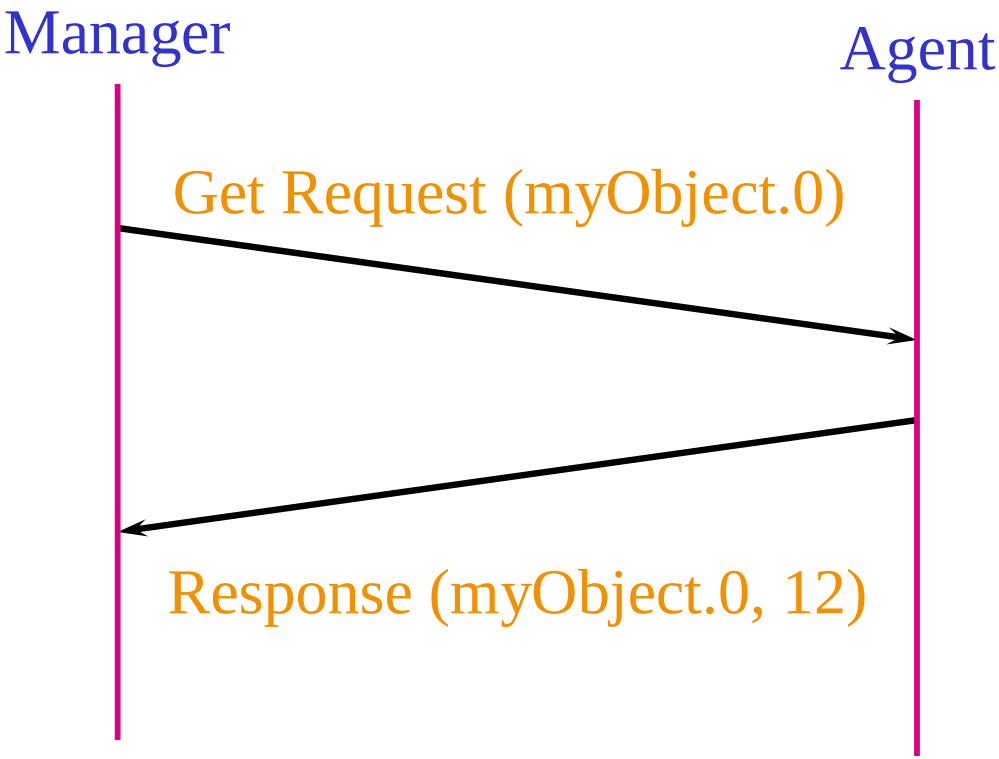
  Enables an agent to report an event to the management station (no response from the manager entity)

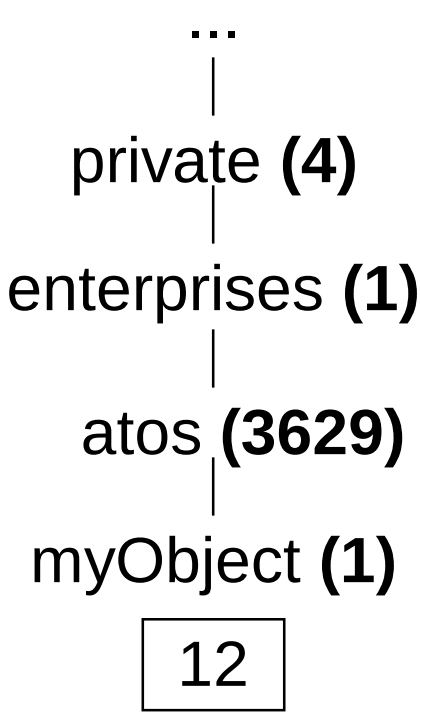
SNMP PDUs Direction



The Get Request

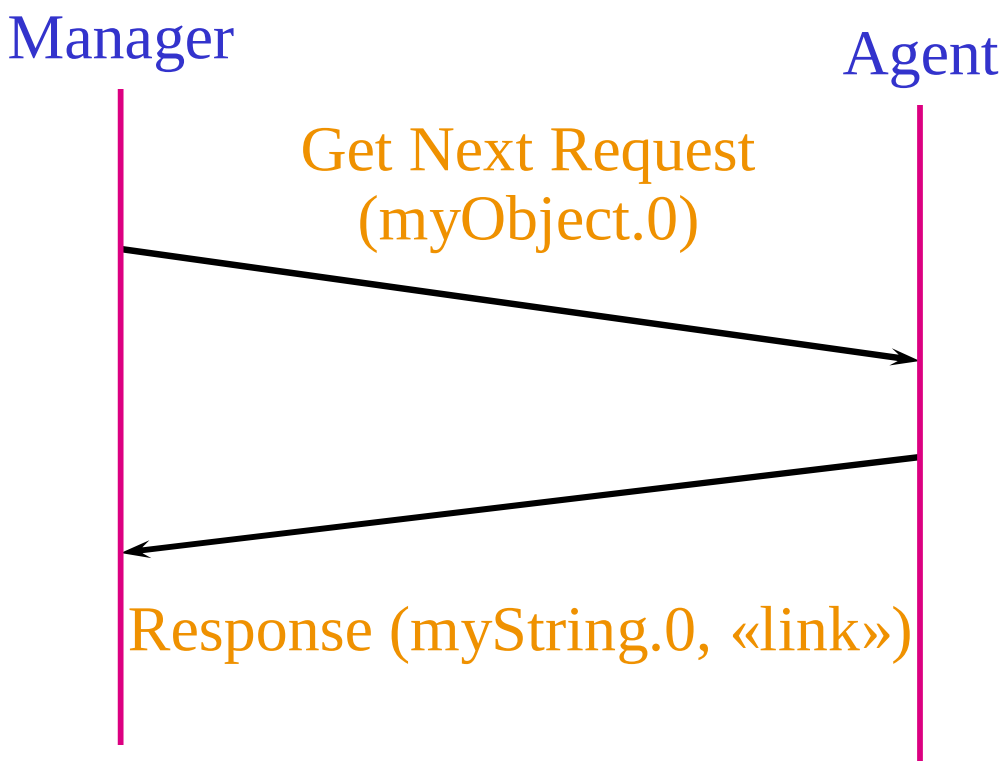
Used to obtain object instance values from an agent

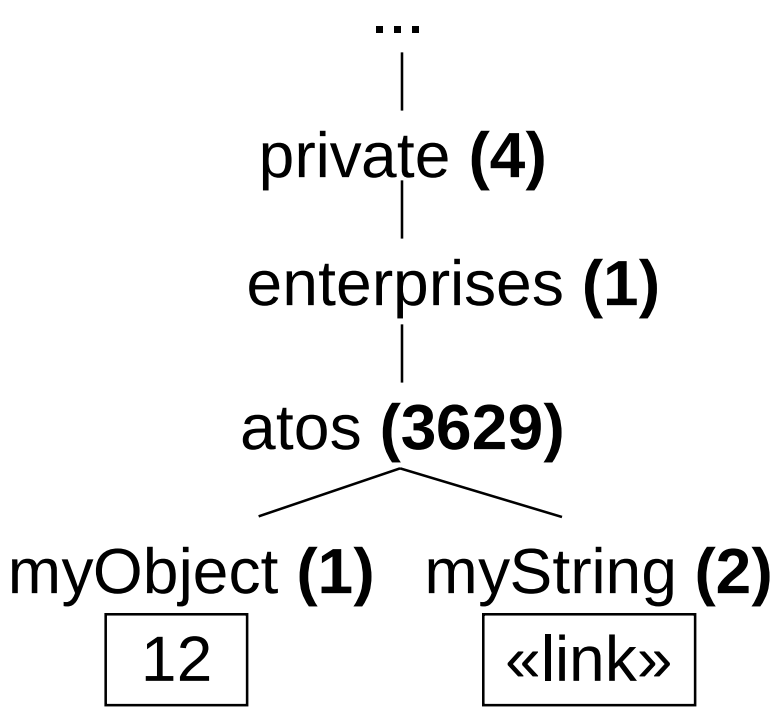




The Get Next Request

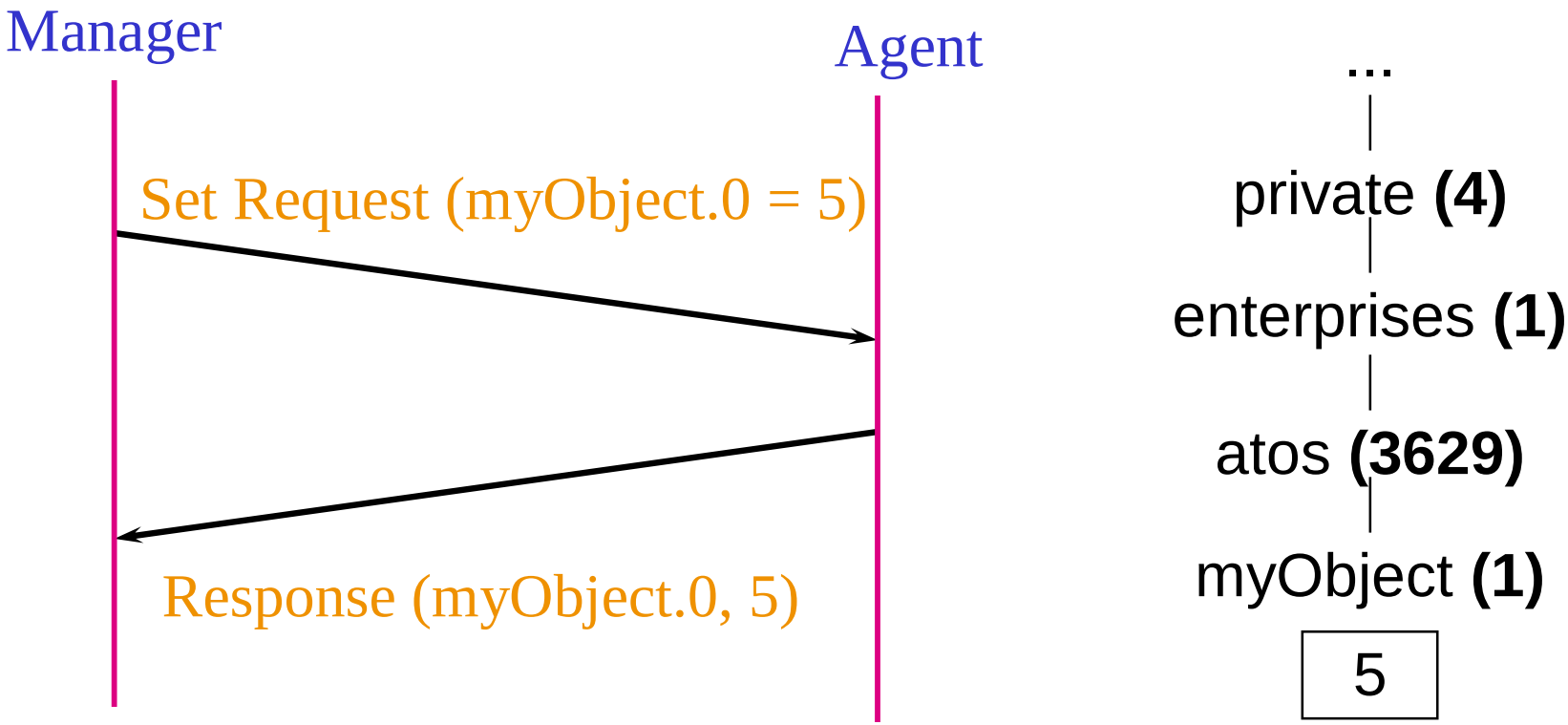
Used to obtain the value of the next object instance from an agent





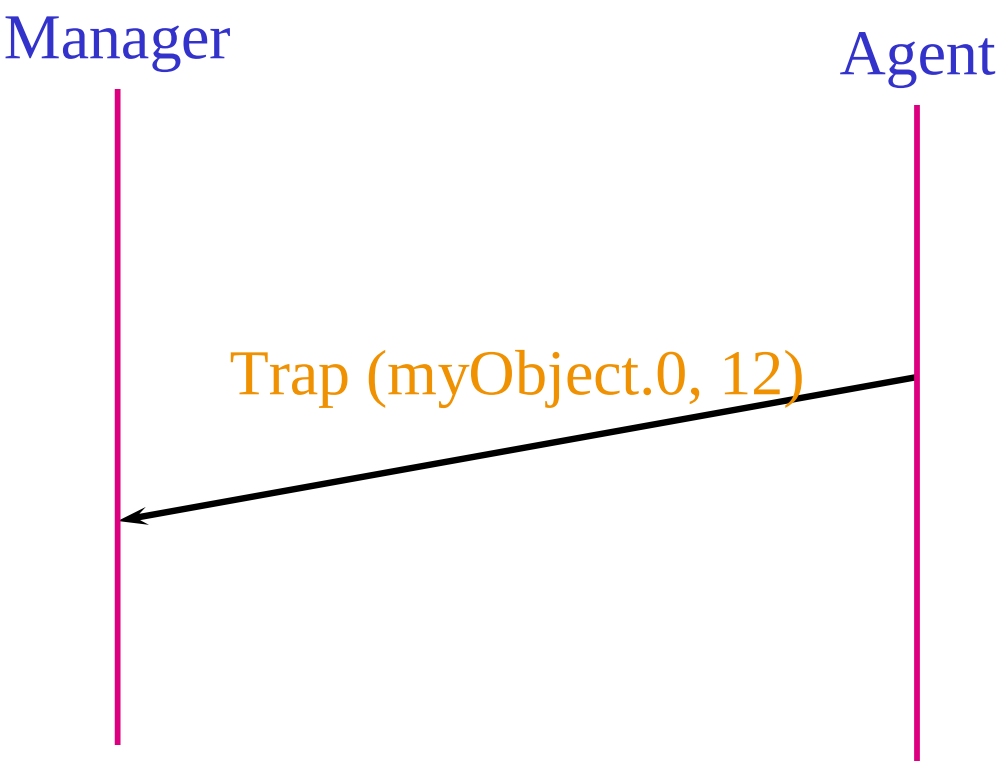
The Set Request

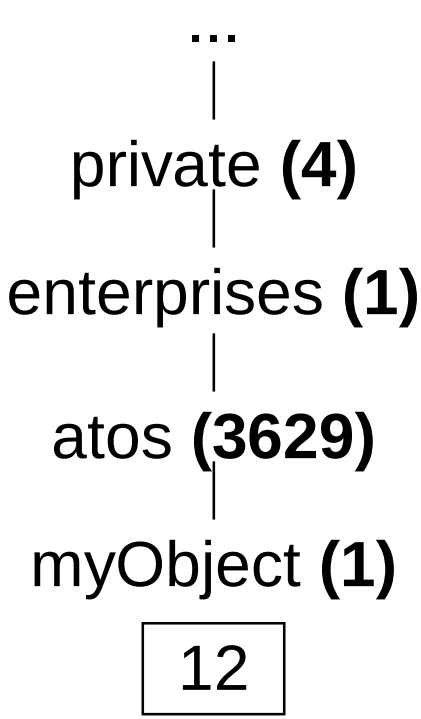
Used to change the value of an object instance within an agent



The Trap Notification

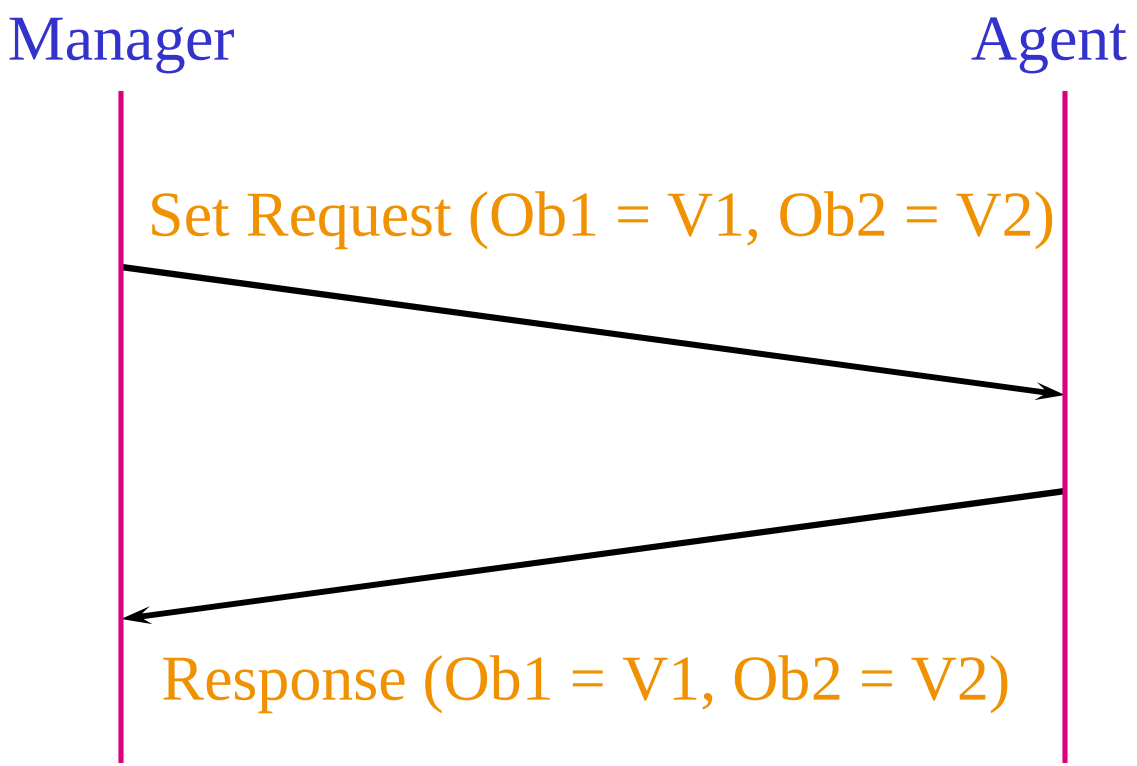
Used by agents to report events to managers





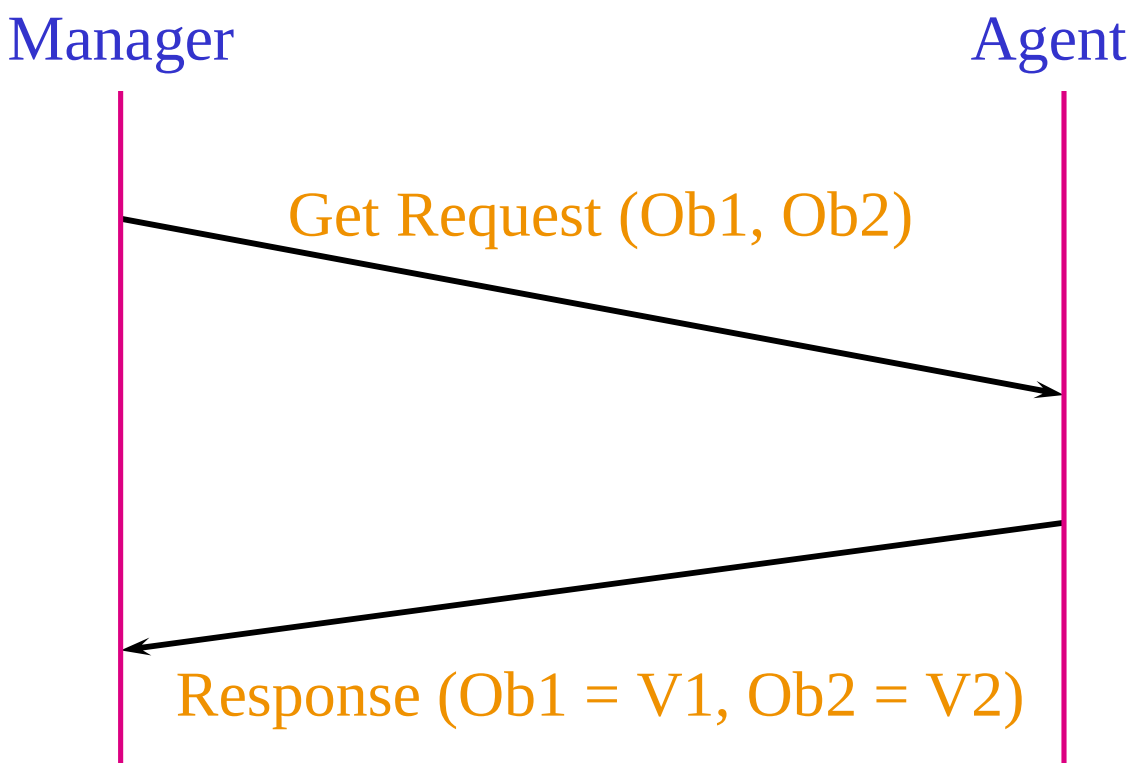
Multiple Requests

The *Get*, *Get Next* and *Set* Requests may contain several objects to retrieve or to set



Atomic Requests (1/2)

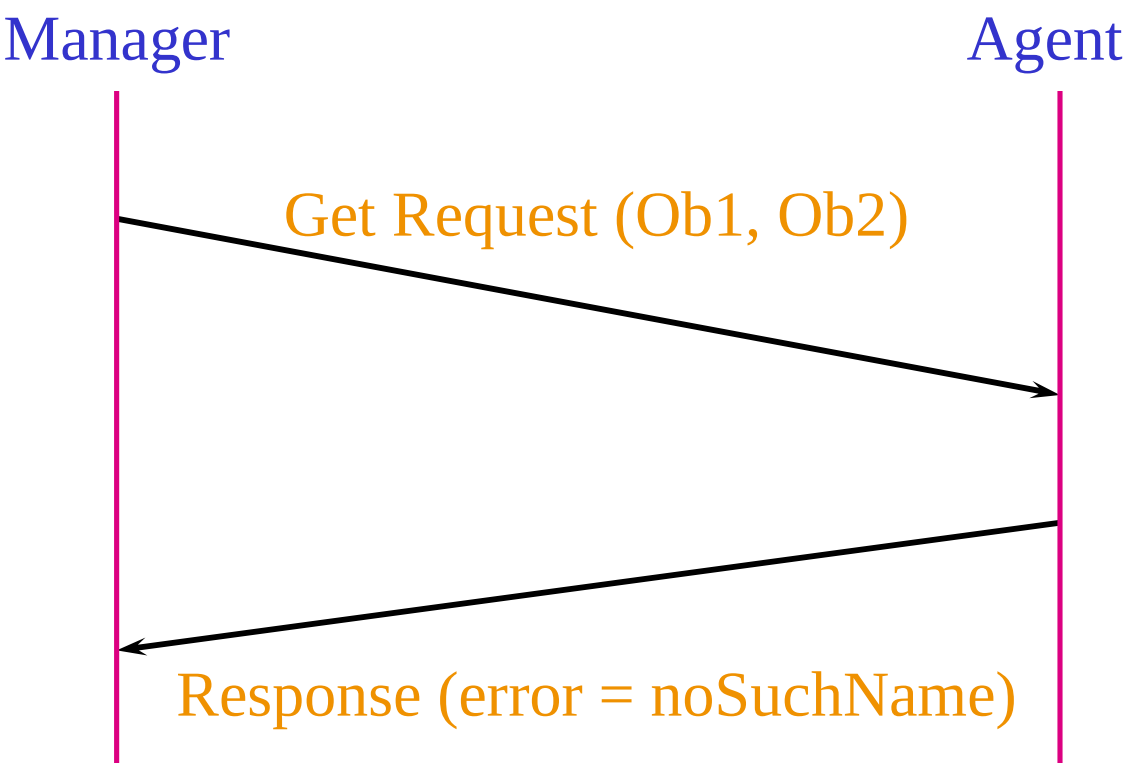
The multiple *Get*, *Get Next* and *Set* Requests are <u>atomic</u> : either all of the values are retrieved/updated or none is



<u>Case 1</u> :

the request is performed

Atomic Requests (2/2)



<u>Case 2</u> :

Ob1 is not implemented,

the request is not performed

SNMP Port Numbers (1/2)

By convention, the UDP port numbers used for SNMP are :

161 (Requests) and 162 (Traps)

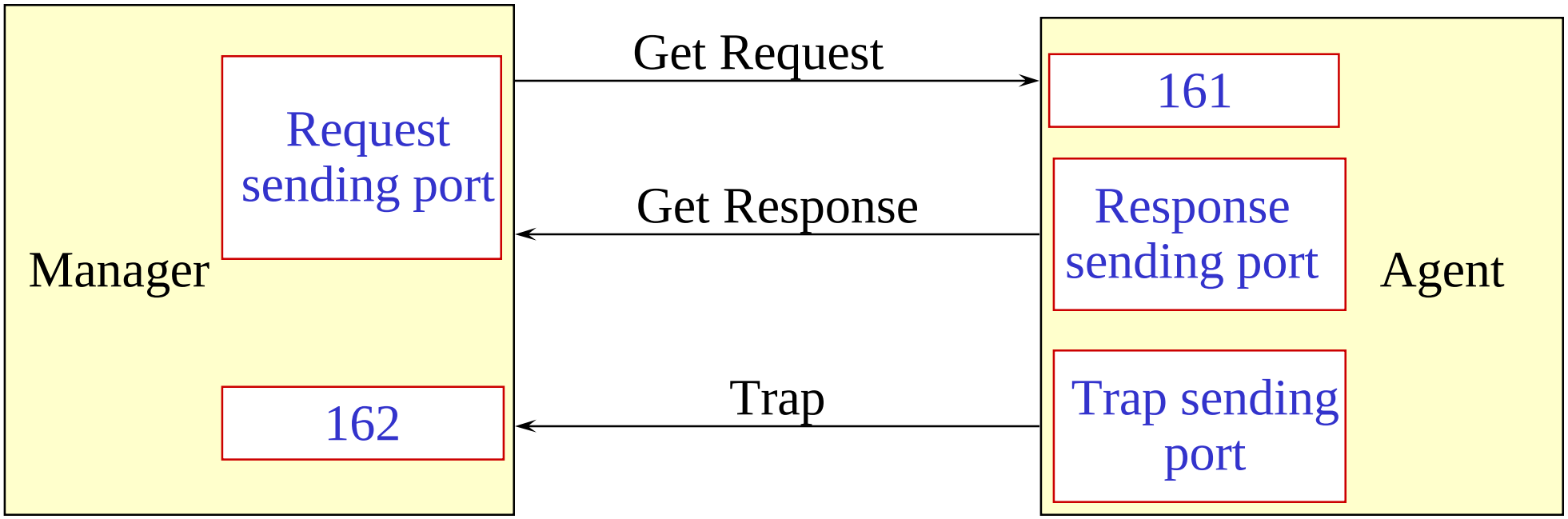
Manager behaviour :

- listens for agent traps on local port 162
- sends requests to port 161 of remote agent

Agent behaviour :

- listens for manager requests on local port 161
- sends traps to port 162 of remote manager

SNMP Port Numbers (2/2)



Loss of PDUs

The actions to be taken are not normalised -> common-sense actions

- In case of *Get* and *Get-Next* requests :
  - The manager can repeat the request one or more times
  - No problem with duplicate messages because of the request-id
- In case of *Set* requests :
  - The manager can test the object with a *Get* to determine whether the *Set* was performed
- In case of *Traps* :
  - The manager should periodically poll the agent for relevant problems

Outline

- **SNMP Architecture**
- **SNMP Protocol**
  - SNMP Operations
  - SNMP Protocol Data Units
  - **SNMP PDUs Format**
  - SNMP PDUs Avanced Concepts
  - SNMP PDUs Encoding
- **SNMP Security Mechanisms**

SNMP Overall Message Format

All SNMP PDUs are built in the same way :

Version	Community	SNMP V1 PDU

SNMP version

(SNMP V1 is version 0)

Community name

PDU-type dependant

Community Name

- Local concept, defined at each agent
- SNMP community = set of SNMP managers allowed to access to this agent
- Each community is defined using a unique (within the agent) name
- Each manager must indicate the name of the community it belongs in all get and set operations

Overall Message ASN.1 Definition

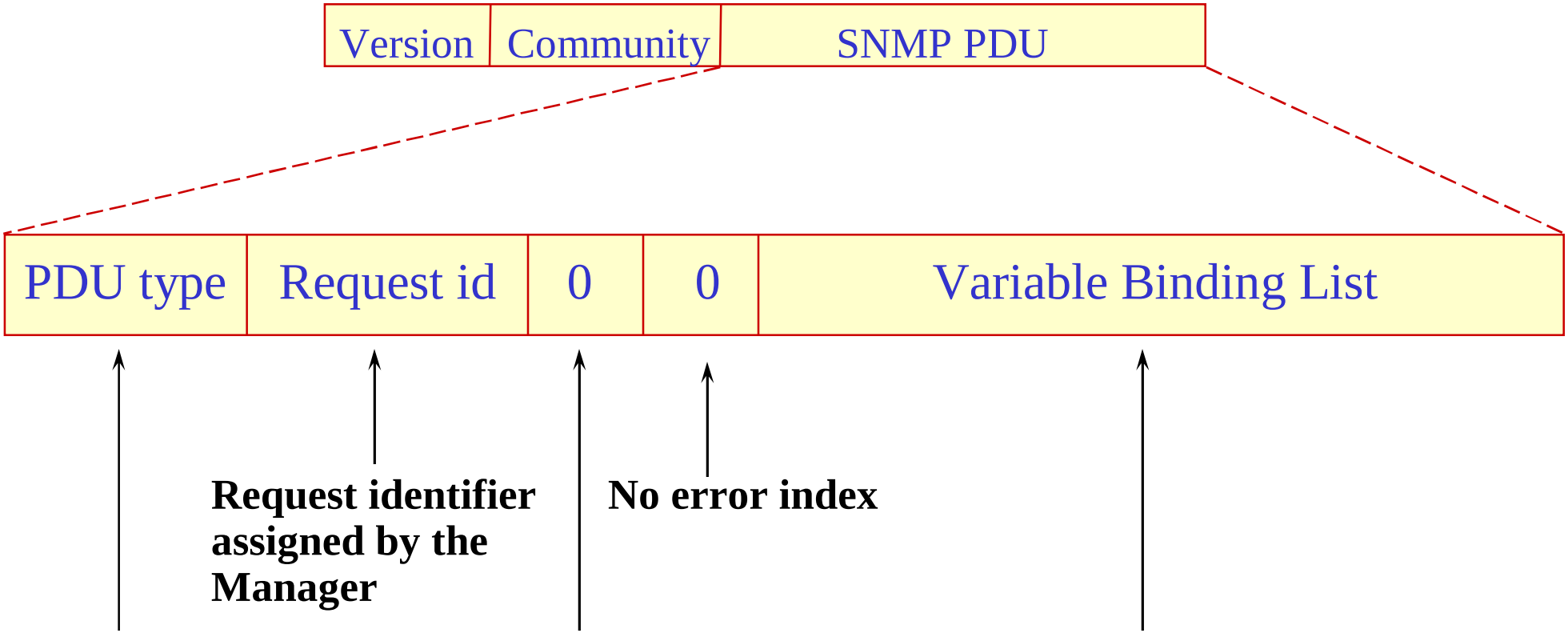
RFC1157-SNMP DEFINITIONS ::= BEGIN

IMPORTS *ObjectName*, *ObjectSyntax*, ... FROM *RFC1155-SMI*;

Message ::= SEQUENCE {

version INTEGER,	Version
community OCTET STRING,	Community
data ANY}	SNMP PDU

Get, Get-Next and Set Format



Request identifier assigned by the Manager

No error index

PDU type

Get Request : 0

Get-Next Request : 1

Set Request : 3

No error status

List of object instances whose values are requested (Get and Get-Next Requests)

List of object instances and corresponding values to set (Set Request)

Get, Get Next and Set ASN.1 Definitions

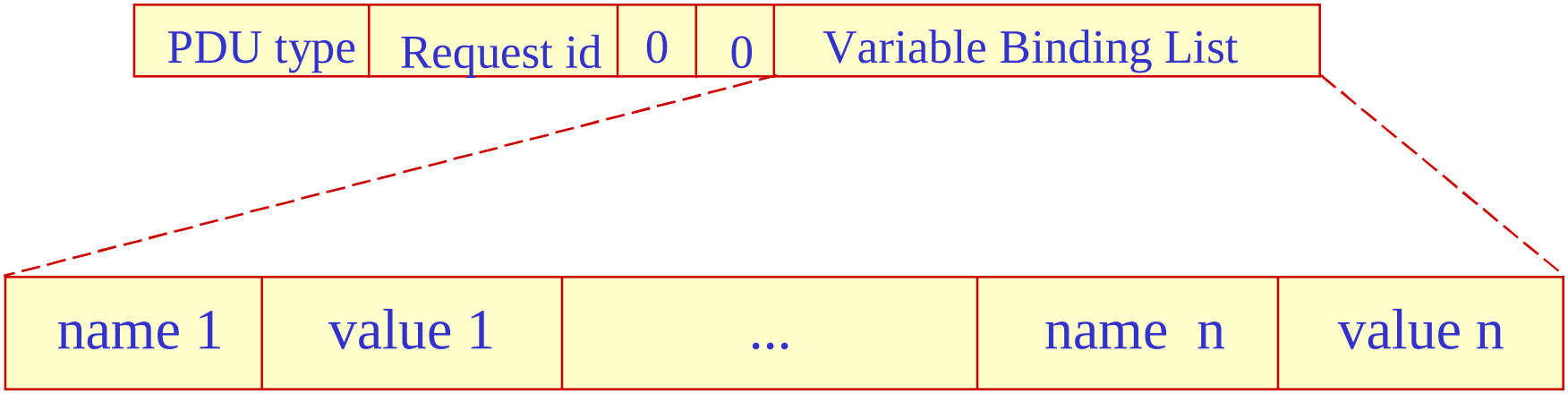
```
PDUs ::= CHOICE { get-request          GetRequest-PDU,
                  get-next-request     GetNextRequest-PDU,
                  response             Response-PDU,
                  set-request          SetRequest-PDU,
                  trap                 Trap-PDU}
GetRequest-PDU      ::= [0] IMPLICITE PDU
GetNextRequest-PDU  ::= [1] IMPLICITE PDU
Response-PDU        ::= [2] IMPLICITE PDU
SetRequest-PDU      ::= [3] IMPLICITE PDU
PDU ::= SEQUENCE {
  request-id INTEGER,
  error-status INTEGER,
  error-index INTEGER,
  variable-binding VarBindList }
```

Request id
0
0
Variable Binding List

Variable Binding List

- Goal : group a number of operations of the same type (get, set, trap) into a single message
- The operation is named a multiple operation
- Advantage : reduce the communication burden of network management
- The Variable Binding field contains the object instances (all PDUs) and the associated values (set and trap only)

The Variable Binding List Format



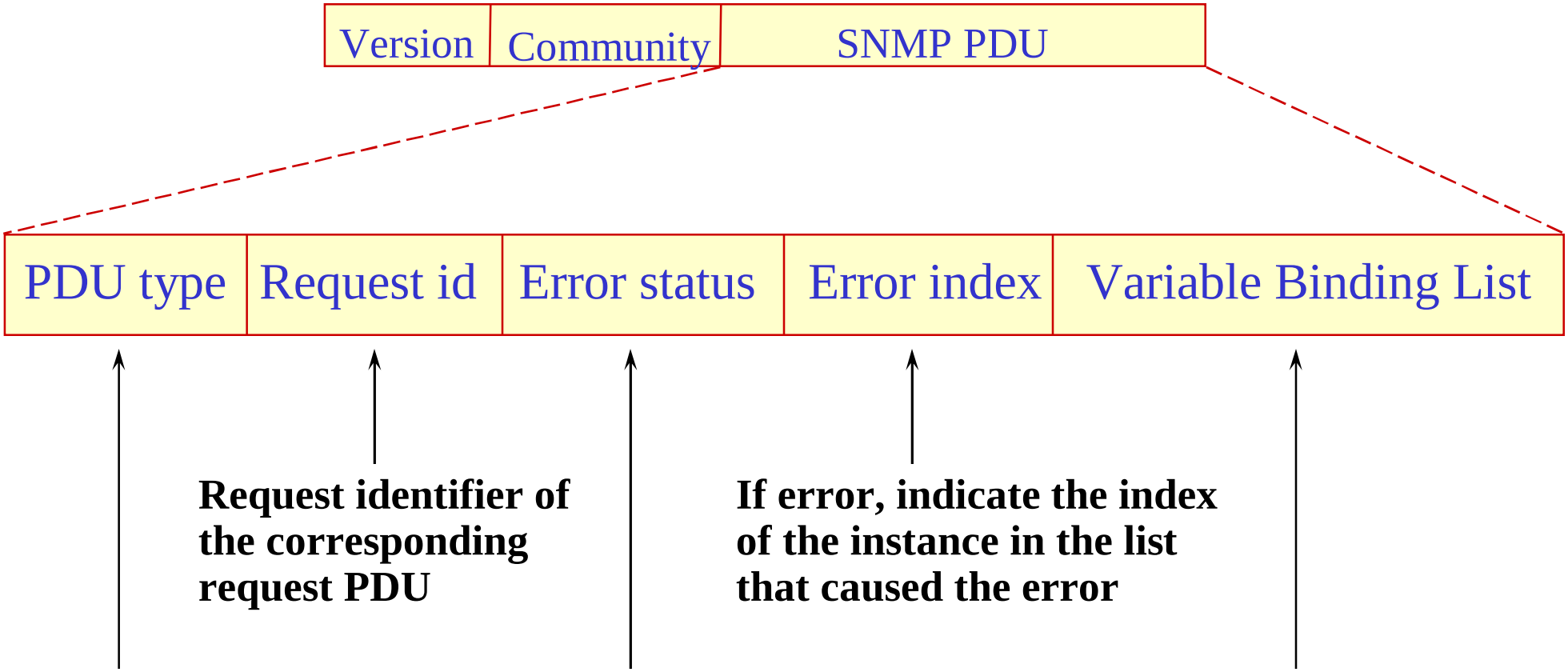
VarBind ::= **SEQUENCE** {

name ObjectName,

value ObjectSyntax }

VarBindList ::= **SEQUENCE OF** *VarBind*

The Response Format



Request identifier of the corresponding request PDU

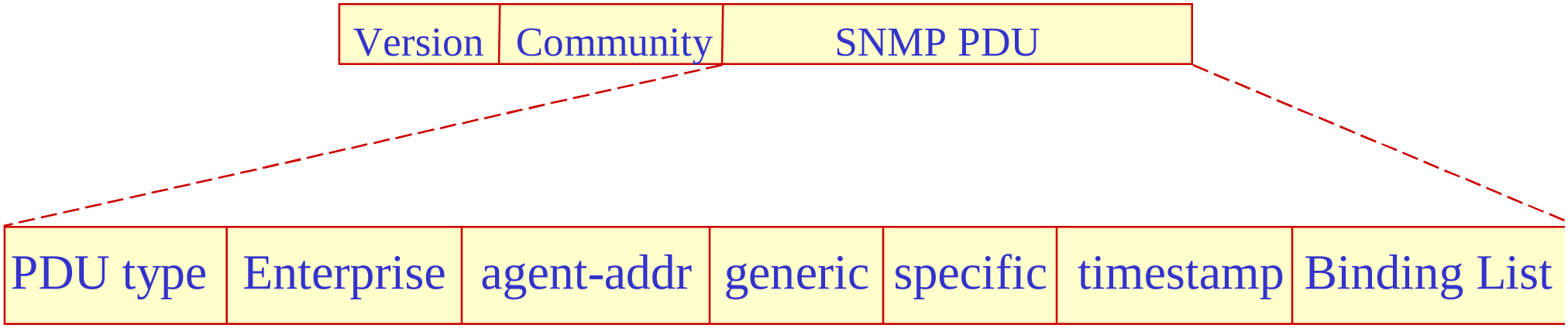
If error, indicate the index of the instance in the list that caused the error

PDU type Response : 2

Indicate that an error occured while processing the request : noError, tooBig, badValue, readOnly and genErr

List of object instances whose values are requested

The Trap Format



System generating the trap (*sysObjectID* of *system* group) or value defined in the MIB

Information about the nature of the event

Time elapsed between the last initialization of the agent and the generation of the trap (*sysUpTime*)

PDU type
Trap : 4

Agent IP address

Information about enterprise specific event

Additional information about the event (implementation specific)

The Generic and Specific Fields (1)

The Generic field may take on one of the following values :

- ***coldStart*** **(0) :**

  An unexpected reinitialization occurs within the agent, due to a crash or major fault

- ***warmStart*** **(1) :**

  A minor fault occurs within the agent

- ***linkDown*** **(2) :**

  A failure occurs in one of the agent communication links; the variable binding area contains the name and value of the affected interface

- ***linkUp*** **(3) :**

  One of the agent communication links has come up; the variable binding area contains the name and value of the affected interface

The Generic and Specific Fields (2)

- ***authenticationFailure*** **(4) :**

  The agent has received a protocol message that it cannot authenticate properly

- ***egpNeighborLoss*** **(5) :**

  An EGP (External Gateway Protocol) neighbor has been declared down; the variable binding area contains the name and value of the egpNeighAddr of the neighbor

- ***enterpriseSpecific*** **(6) :**

  Some enterprise-specific event has occured; the Specific field indicates the type of event

The Trap ASN.1 Definition

***PDUs* ::= CHOICE {get-request *GetRequest-PDU*,**

...

trap *Trap-PDU*}

***Trap-PDU* ::= [4] IMPLICIT SEQUENCE {**

enterprise OBJECT IDENTIFIER,
agent-addr NetworkAddress,
generic-trap INTEGER {
coldStart (0),
...
enterpriseSpecific (6) },
specific-trap INTEGER,
time-stamp TimeTicks,
variable-bindings VarBindList }

Enterprise
agent-addr
generic
specific
timestamp
Variable Binding List

Trap Example

Trap	Enterprise	agent-addr	generic	specific	timestamp
4	*1.3.6.1.4.1.20.1*	*132.18.54.21*	*3*	*0*	*22759400*
ipInReceives.0		*956340*			

Binding List

- **IP address of the sending agent : 132.18.54.21**
- **Object concerned by the trap : 1.3.6.1.4.1.20.1 (private MIB)**
- **Problem type : a communication link has been reinitialised**
- **Indication : the number of received IP paquets is 956340**
- **Last reinitialisation of the agent : 6 hours ago**

Outline

- **SNMP Architecture**
- **SNMP Protocol**
  - SNMP Operations
  - SNMP Protocol Data Units
  - SNMP PDUs Format
  - **SNMP PDUs Avanced Concepts**
  - SNMP PDUs Encoding
- **SNMP Security Mechanisms**

Get Request Operation

The *Get Request* operation accesses only to <u>instances of leaf objects</u>

mib2**(1.3.6.1.2.1)**

interfaces**(2)**

ifTable**(2)**

ifEntry**(1)**

ifIndex**(1)** ifPhysAddress**(6)** ifAdminStatus**(7)**

1	00:00:39:20:04	1 (up)
2	08:00:56:16:11	3 (testing)
8	00:00:b4:02:33	2 (down)

GetRequest (ifPhysAddress.2)

Response (ifPhysAddress.2 = 08:00:56:16:11)

Get Request in Tabular Objects

The *Get Request* operation only allows the retrieval of <u>leaf objects</u>

<u>Consequence</u> : it is not possible to retrieve

- an entire row of a table (by referencing the entry object)
- an entire table (by referencing the table object)

<u>Solution</u> : retrieve an entire row by including each object instance of the table in the Variable Binding field

Get Request Example

mib2**(1.3.6.1.2.1)**

interfaces**(2)**

ifTable**(2)**

ifEntry**(1)**

ifIndex**(1)** ifPhysAddress**(6)** ifAdminStatus**(7)**

1	00:00:39:20:04	1 (up)
2	08:00:56:16:11	3 (testing)
8	00:00:b4:02:33	2 (down)

To get the second row

GetRequest (ifIndex.2, ifPhysAddress.2, ifAdminStatus.2)

Get Request Error Status

Error Situations	Error Status	Error Index
An object of the Variable Binding field does not match any object leaf in the MIB tree	noSuchName	index of the object
The size of the resulting Get Response PDU exceeds the local limitation	tooBig	-
Other reason	genErr	index of the object

GetNext Request Operation

The *Get Next* Request has <u>three advantages</u>, compaired to *Get* :

- Allows the retrieving of unknown objects
- More efficient way to retrieve a set of object values when some are not implemented by the agent
- Allows the retrieving of an entire table, without knowing its content

Retrieving Unknown Objects

No requirement that the supplied identifier represents an object instance

The *Get Next* operation can be used to discover the MIB structure

mib2**(1.3.6.1.2.1)**

interfaces**(2)**

ifTable**(2)**

ifEntry**(1)**

ifIndex**(1)** ifPhysAddress**(6)** ifAdminStatus**(7)**

1	00:00:39:20:04	1 (up)
2	08:00:56:16:11	3 (testing)
8	00:00:b4:02:33	2 (down)

GetNextRequest (interfaces)

Response (ifIndex.1 = 1)

The manager learns that the first supported object in the *interfaces* sub-tree is *ifIndex*

Retrieving a Set of Objects (1/2)

mib**(1)**

udp**(7)**

udpInDatagrams**(1)**	udpNoPorts**(2)**	udpInErrors**(3)**	udpOutDatagrams**(4)**
43258	**433**	**5021**	**76320**

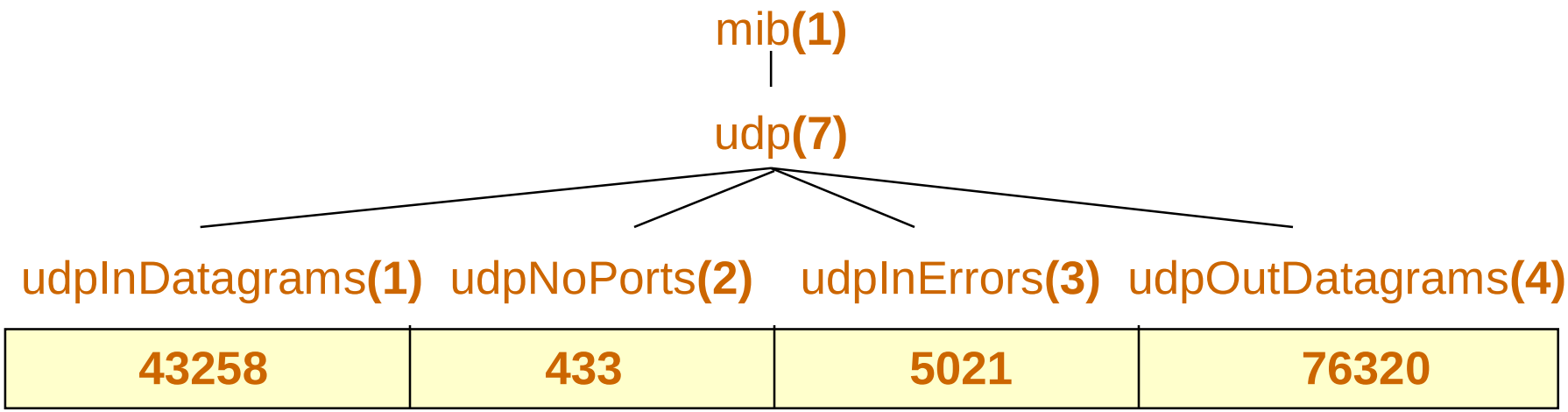
If *udpNoPorts* is <u>not implemented</u> in the agent MIB :

GetRequest (udpInDatagrams.0, udpNoPorts.0,
udpInErrors.0, udpOutDatagrams.0)

↓

Response (noSuchName)

Retrieving a Set of Objects (2/2)



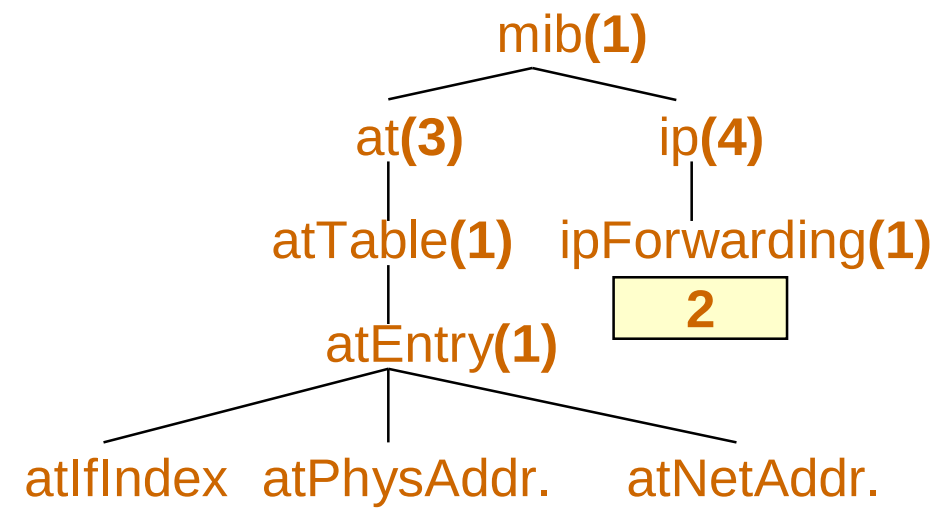
If *udpNoPorts* is <u>not implemented</u> in the agent MIB :

GetNextRequest (udpInDatagrams, udpNoPorts, udpInErrors, udpOutDatagrams)

Response (udpInDatagrams.0 = 43258, udpInErrors.0 = 5021, udpInErrors.0 = 5021, udpOutDatagrams.0 = 76320)

Retrieving Unknown Tables (1/4)

The *Get Next* operation can be used to <u>retrieve an entire table</u>

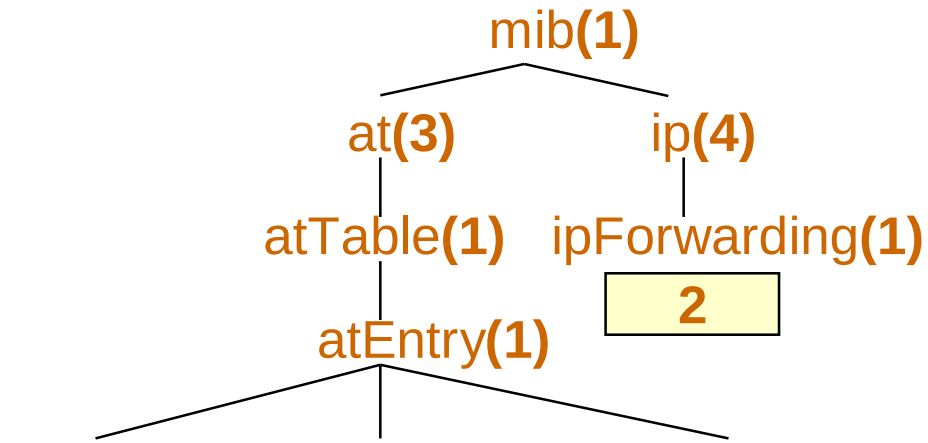


1	00:00:39:20:04	194.2.6.10
4	08:00:56:16:11	194.22.67.45
5	00:00:b4:02:33	194.7.53.11

GetNextRequest (atIfIndex, atPhys, atNet)

Response (atIfIndex.1 = 1,
atPhys.1 = 00:00:39:20:04,
atNet.1 = 194.2.6.10)

Retrieving Unknown Tables (2/4)

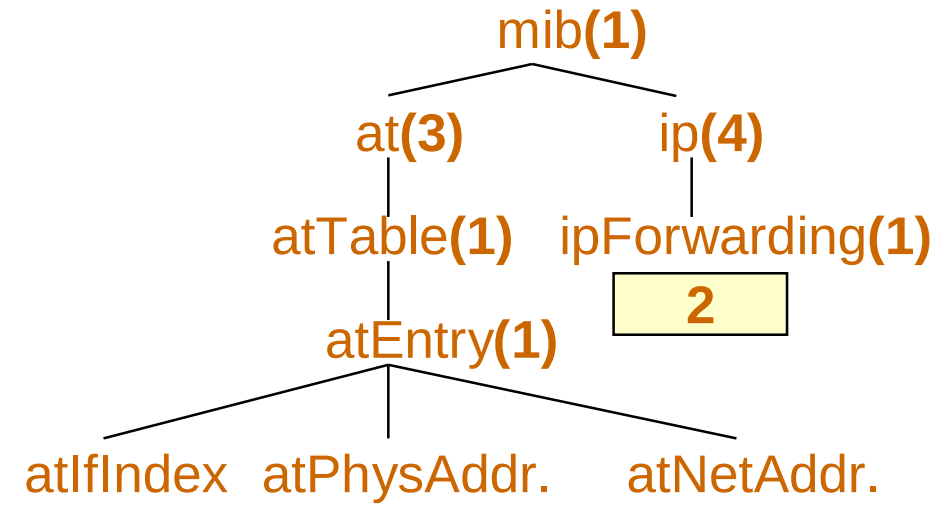


1	00:00:39:20:04	194.2.6.10
4	08:00:56:16:11	194.22.67.45
5	00:00:b4:02:33	194.7.53.11

GetNextRequest (atIfIndex.1, atPhys.1, atNet.1)

Response (atIfIndex.4 = 4,
atPhys.4 = 08:00:56:16:11,
atNet.4 = 194.22.67.45)

Retrieving Unknown Tables (3/4)

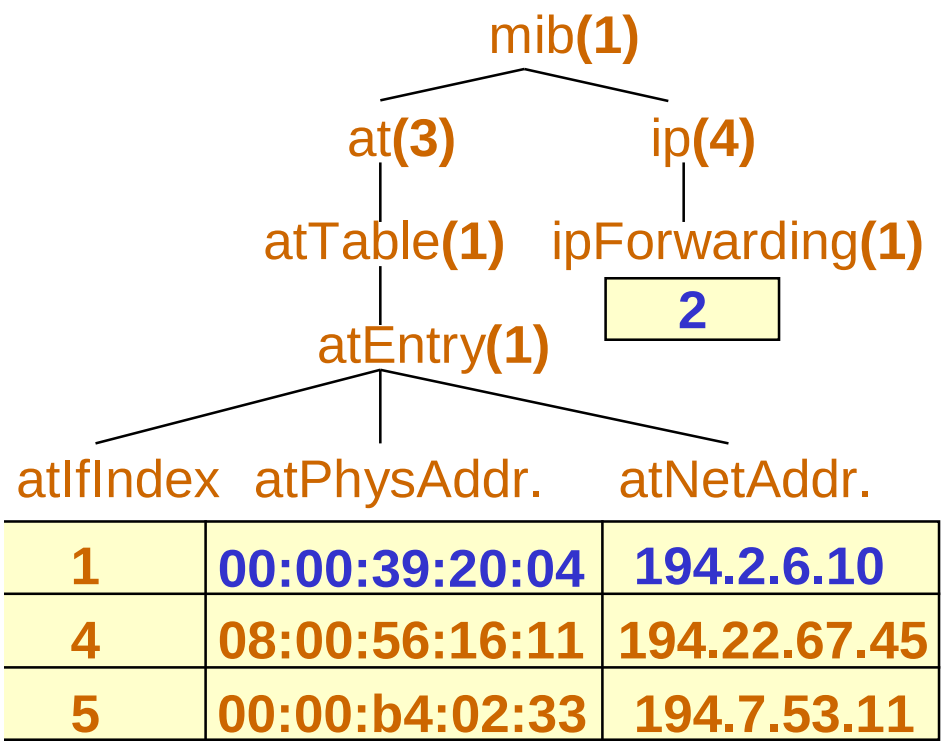


1	00:00:39:20:04	194.2.6.10
4	08:00:56:16:11	194.22.67.45
5	00:00:b4:02:33	194.7.53.11

GetNextRequest (atIfIndex.4, atPhys.4, atNet.4)

Response (atIfIndex.5 = 5,
atPhys.5 = 00:00:b4:02:33,
atNet.5 = 194.7.53.11)

Retrieving Unknown Tables (4/4)



GetNextRequest (atIfIndex.5, atPhys.5, atNet.5)

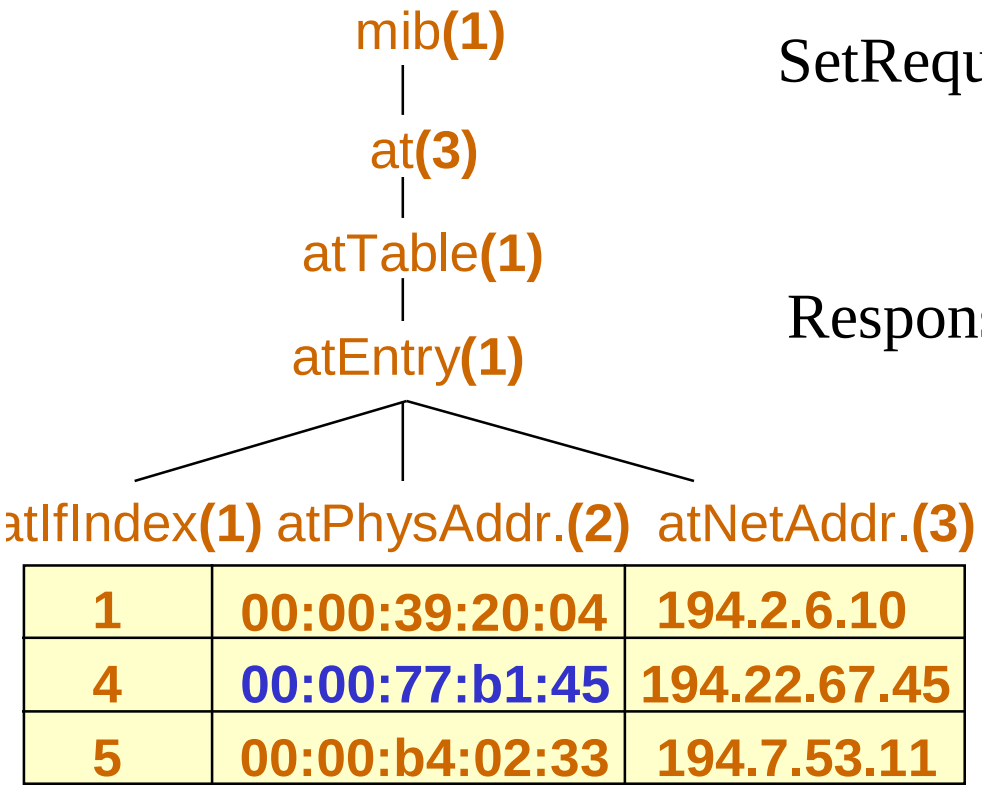
Response (atPhys.1 = 00:00:39:20:04,
atNet.1 = 194.2.6.10,
ipForwarding.0 = 2)

The object names in the response do not match those in the request :

The manager learns that it has reached the end of the *at* table

Set Request Operation

The *Set Request* operation accesses only to <u>instances of leaf objects</u>



SetRequest (atPhysAddress.4 = 00:00:77:b1:45)

Response (atPhysAddress.4 = 00:00:77:b1:45)

Set Request Limitations

RFC 1157 does not provide any specific guidance about *Set Request* operations on tabular objects :

- updating tables
- row deletion
- performing an action within the agent

The SNMP agents are free to implement these points in several ways

Row Adding (1/2)

mib**(1)**

at**(3)**

atTable**(1)**

atEntry**(1)**

atIfIndex**(1)**	atPhysAddr.**(2)**	atNetAddr.**(3)**
1	**00:00:39:20:04**	**194.2.6.10**
4	**08:00:56:16:11**	**194.22.67.45**
5	**00:00:b4:02:33**	**194.7.53.11**

SetRequest (atIfIndex.9 = 9,
atPhys.9 = 08:00:9e:00:23,
atNet.9 = 196.44.98.03)

The agent developer can choose to :

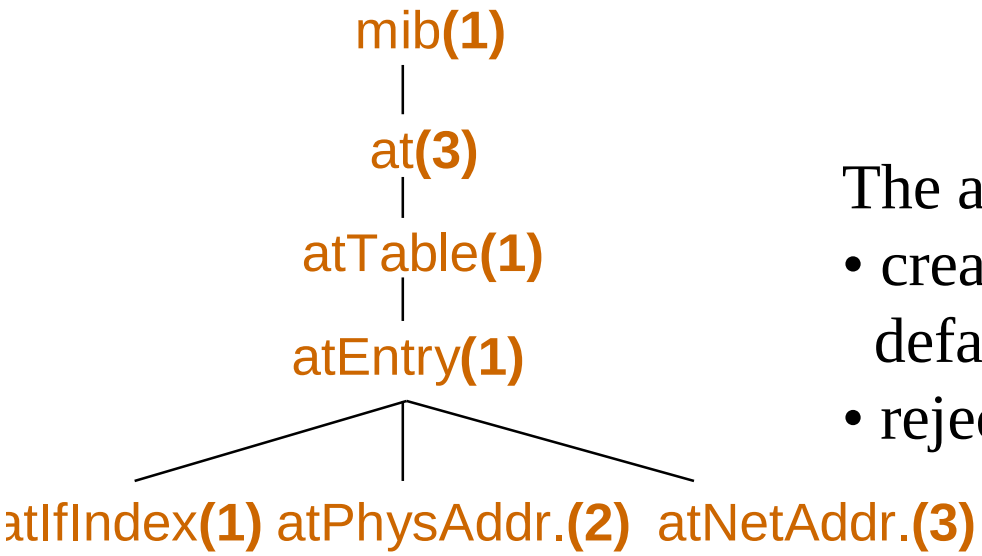
- reject the operation (noSuchName)
- create a new row, if the assigned values are consistent
- reject the operation (badValue) if not

Row Adding (2/2)

SetRequest (atIfIndex.9 = 9)

The agent developer can choose to :

- create a new row by supplying default values for the objects not listed
- reject the operation (badValue)



1	00:00:39:20:04	194.2.6.10
4	08:00:56:16:11	194.22.67.45
5	00:00:b4:02:33	194.7.53.11

Row Deletion

mib**(1)**

ip**(4)**

ipRouteTable**(21)**

ipAddrEntry**(1)**

pRouteDest	ipRouteMetric1	ipRouteType
194.2.6.10	4	1
194.0.67.5	3	1
194.71.3.1	9	1

SetRequest (ipRouteType.194.2.6.10 = 2)

The agent developer can choose the following convention :

- ipRouteType = 1 : valid row
- ipRouteType = 2 : invalid row

When receiving the request, it marks the first row as null

Performing an Action

The agent developer can use a proprietary object to <u>represent an action</u>

...

ReBoot **(1)**

0

SetRequest (ReBoot.0 = 1)

The agent developer can choose to reboot the system when receiving this request

Set Request Error Status

Error Situations	Error Status	Error Index
An object named in the Variable Binding field does not match any object leaf in the MIB tree	noSuchName	index of the object
The size of the resulting Get Response PDU exceeds the local limitation	tooBig	-
A variable name and value are inconsistent (type, length, value...)	badValue	index of the object
Other reason	genErr	index of the object

Outline

- **SNMP Architecture**
- **SNMP Protocol**
  - SNMP Operations
  - SNMP Protocol Data Units
  - SNMP PDUs Format
  - SNMP PDUs Avanced Concepts
  - **SNMP PDUs Encoding**
- **SNMP Security Mechanisms**

What are the Basic Encoding Rules ?

- **Standardized by CCITT (X.209) and ISO (ISO 8825)**
- **Provides a set of rules to develop an unambiguous, bit-level description of data :**
- **How data are represented during the**
- **communication transfer process of SNMP PDUs ?**

The Basic Encoding Rules (BER)

Any ASN.1 value is encoded as an octet string :

- The encoding is based on the use of a Type-Length-Value (TLV) structure
- This structure is recursive : the «V» portion may consist of one or more TLV structures

Value Encoding

1 to n bytes	1 to n bytes	1 to n bytes
Identifier	Length	Content

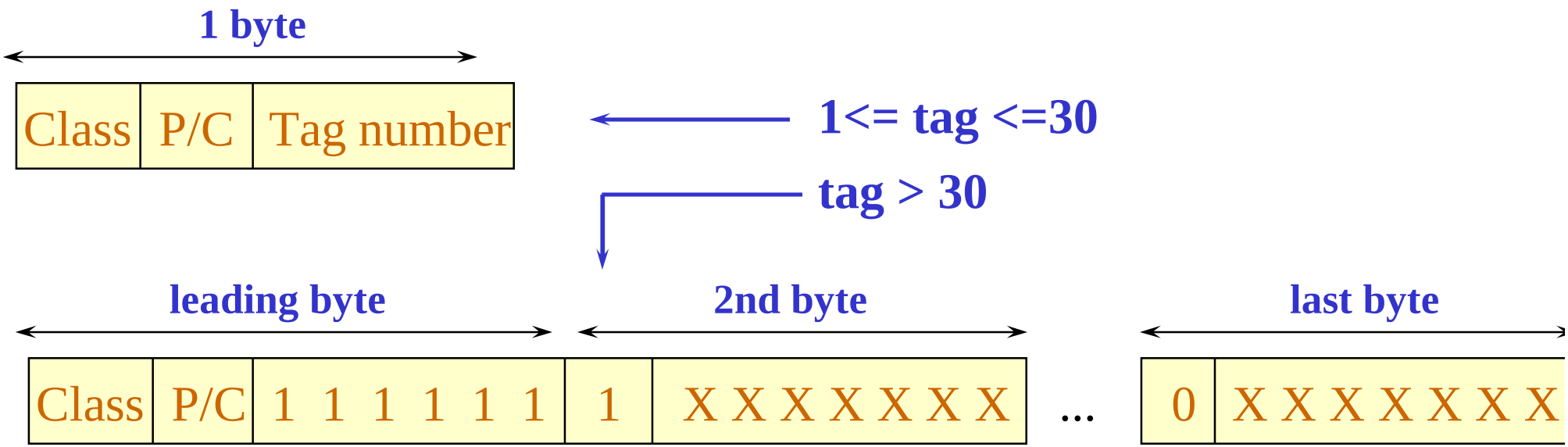
the length of the value is known in advance

1 to n bytes	1 to n bytes	1 to n bytes	1 byte
Identifier	Length	Content	EOC

the length of the value is not known in advance

EOC = 00000000

Identifier Field



<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> :

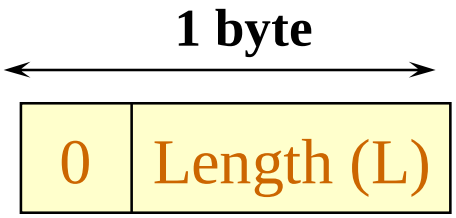
1 = Boolean type

2 = Integer type

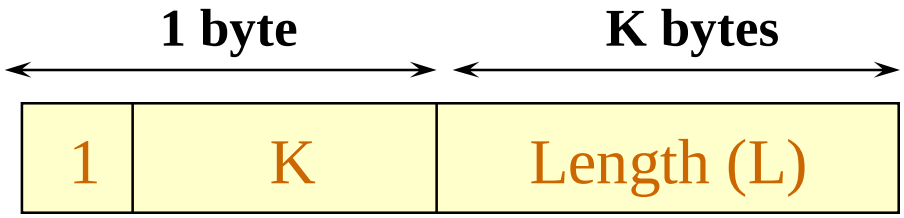
...

> 30 : X...X = tag number

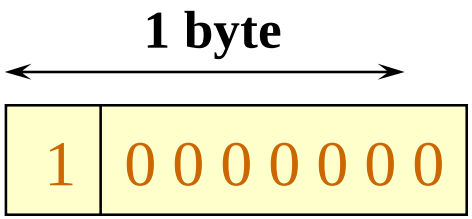
Length Field



short definite length :
1<= L <= 127



long definite length :
128 <= L <= 2^{1008}



undefinite length : value terminated by EOC

Simple Encoding Examples

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

GET Request Encoding Example

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
```

GET Response Encoding Example

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...»
```

Outline

- **SNMP Architecture**
- **SNMP Protocol**
  - SNMP Operations
  - SNMP Protocol Data Units
  - SNMP PDUs Format
  - SNMP PDUs Avanced Concepts
  - SNMP PDUs Encoding
- **SNMP Security Mechanisms**

SNMP Security Mechanisms

The basic SNMP standard provides only trivial security mechanisms, based on:

- Authentication Mechanism
- Access mode Mechanism

Authentication Mechanism

- Goal of the Authentication Service : assure the destination that the SNMP message comes from the source from which it claims to be
- Based on community name, included in every SNMP message from a management station to an agent
- This name functions as a password : the message is assumed to be authentic if the sender knows the password
- No encryption/decryption of the community name

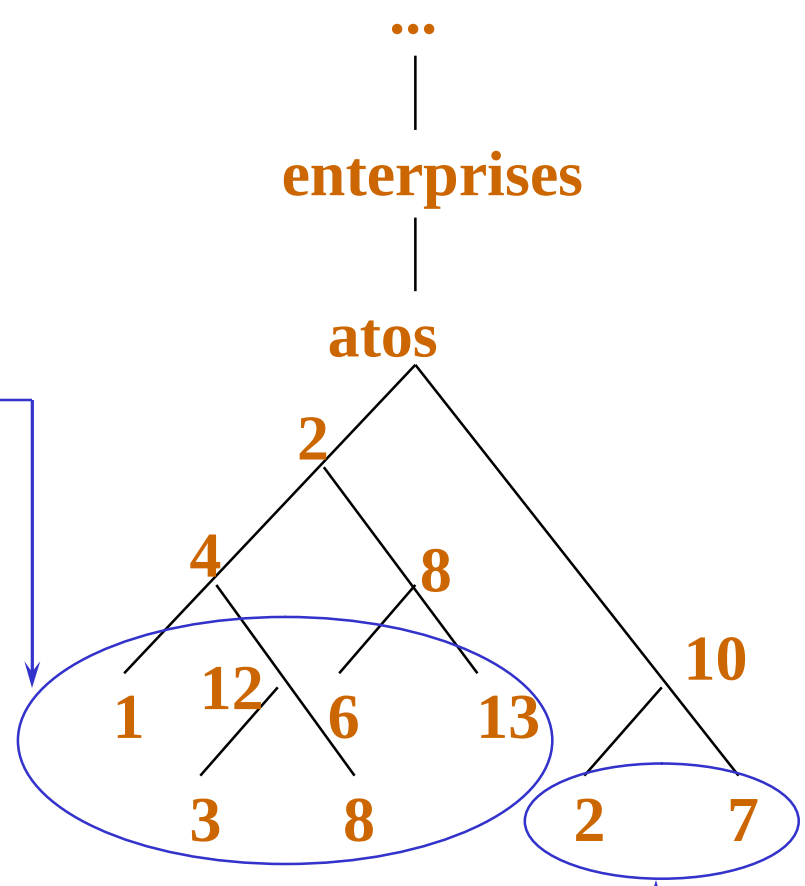
Access Mode Mechanism

- Based on community profiles
- A community profile consists of the combinaison of :
  - a defined subset of MIB objects (MIB view)
  - an access mode for those objects (READ-ONLY or READ-WRITE)
- A community profile is associated to each community defined by an agent

Access Mode Example

community profile =
«public» : READ-ONLY
«atos_com» : READ-WRITE

community profile =
«public» : READ-ONLY
«atos_com» : READ-ONLY



SNMP V1 : Standard MIBs

Outline

- **General MIB Structure**
- MIB-I and MIB-II Presentation
  - Overview
  - MIB-II Groups
- The Private MIBs

SNMP MIB Features

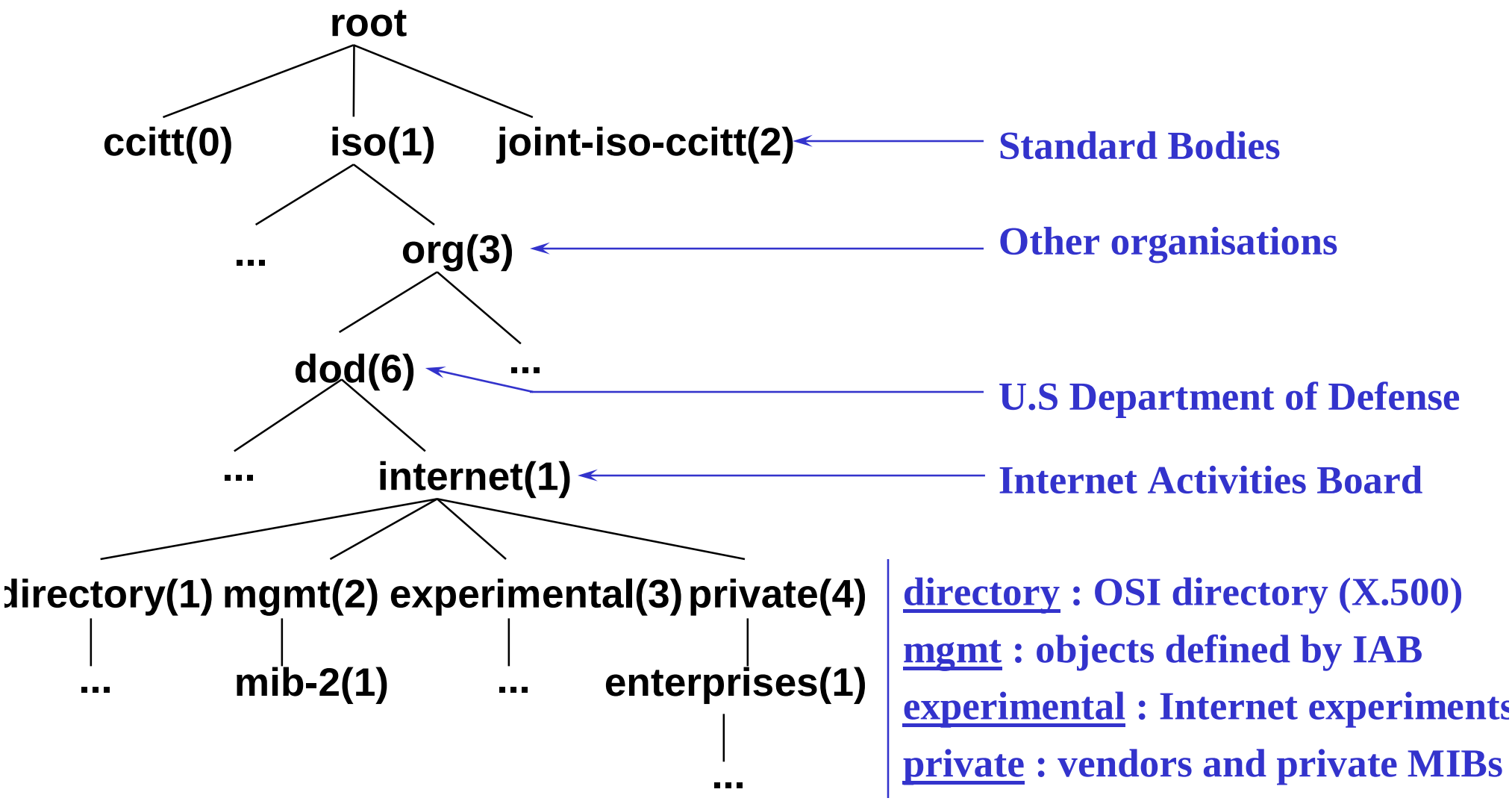
- Describes standardised objects
- Flexible enough to accompany technology changes
- Flexible enough to adapt to specific product offerings

Standardised MIBs

The International Architective Board (IAB) organization and other cooperating organisms have standardised several MIBs :

- **MIB-II**
- **Frame Relay**
- **FDDI**
- **AppleTalk**
- **OSI CMIP**
- **Token Ring**
- **Token Bus**
- **Ethernet**
- **ATM**
- **...**

Overall MIB Structure



Outline

- General MIB Structure
- **MIB-I and MIB-II Presentation**
  - **Overview**
  - MIB-II Groups
- The Private MIBs

MIB-I and MIB-II Overview

- MIB-I is defined in RFC 1156 :
  - 114 objects defined within 8 groups
- MIB-II is defined in RFC 1213 :
  - superset of MIB-I (2nd version)
  - 171 objects defined within 10 groups
- MIB-II is the most important MIB specification, covering a broad range of managed objects

MIB-I/MIB-II Objects

groups	MIB-I	MIB-II
system	3	7
interfaces	22	23
at	3	3
ip	33	38
icmp	26	26
tcp	17	19
udp	4	7
egp	6	18
transmission	X	0
snmp	X	30

MIB-II defines two new groups: transmission and snmp

Outline

- General MIB Structure
- **MIB-I and MIB-II Presentation**
  - Overview
  - **MIB-II Groups**
- The Private MIBs

MIB-II Groups

mib-2 (mgmt 1)

system (1)	**General information about the managed system**
interfaces (2)	**Generic information about the physical interfaces**
at (3)	**Address translation table (network addr. to physical addr.)**
ip (4)	**Information about the IP implementation of the system**
icmp (5)	**Information about the ICMP implementation of the system**
tcp (6)	**Information about the TCP implementation of the system**
udp (7)	**Information about the UDP implementation of the system**
egp (8)	**Information about the EGP implementation of the system**
transmission (10)	**Information about the transmission medium of each interface**
snmp (11)	**Information about the SNMP implementation of the system**

The System Group

system (mib-2 1)

sysDescr (1)	**Description of the managed system (hardware, O.S., ...)**
sysObjectID (2)	**Vendor's authoritative identification of the managed system**
sysUpTime (3)	**Time since the managed system was last reinitialised**
sysContact (4)	**Identification of the person responsible for this system**
sysName (5)	**Administratively assigned name for the managed system**
sysLocation (6)	**Physical location of the managed system**
sysServices (7)	**Set of services that the managed system offers**

The Interfaces Group

- interfaces (mib-2 2)
  - ifNumber (1) — Total number of network interfaces of the system
  - ifTable (2) — Interface table (one row per interface)
    - ifEntry (1) — Interface entry
      - ifIndex (1) — Unique value for each interface (betw. 1 and ifNumber)
      - ifDescr (2) — Information about the interface (name,vendor,version, ...)
      - ifType (3) — Type of the interface (Ethernet,Tokenring,Framerelay,...)
      - ifSpeed (5) — Estimate of the interface's current data rate capacity
      - ifPhysAddres (6) — Interface's address
      - ifInOctets (10) — Total number of octets received on the interface
      - ...

The Address Translation Group

- at (mib-2 3)
  - atTable (1) — Address translation table (one row per physical interface)
    - atEntry (1) — Address translation entry
      - atifIndex (1) ← ifIndex value of the current interface
      - atPhysAddres(2) — Media-dependent physical address (ex. : MAC, X.121)
      - atNetAddress(3) ← Network address corresponding to the physical address (e.g., IP, X25)

The IP Group

ip (mib-2 4)

- **ipForwarding (1)** — The system is acting as gateway (1) or not (2)
- **ipInReceives (3)** — Total number of IP datagrams received from interfaces
- **ipOutRequests (10)** — Total number of IP datagrams that IP users supplied to IP layer
- **ipAddrTable (20)** — Table of the IP addresses assigned to each physical interface (described in the ifTable)
- **ipRouteTable (21)** — IP routing table (for each route : destination IP address of the route, physical interface of the next node, ...)
- **ipNetToMediaTable(22)** — Address translation table that provides correspondence between physical and IP addresses

...

The ICMP Group

ICMP (Internet Control Message Protocol) provides feedback about communication problems

icmp (mib-2 5)

- **icmpInMsgs (1)** — **Total number of ICMP messages received by the system**
- **icmpInErrors (2)** — **Total number of ICMP messages received with error**
- **icmpOutMsgs (14)** — **Total number of ICMP messages that the system attempted to send**
- **icmpOutErrors (15)** — **Total number of ICMP messages that the system did not send due to problems discoved within ICMP**

...

The TCP Group

- tcp (mib-2 6)
  - tcpCurrEstab (9) — Number of currently established TCP connections
  - tcpInSegs (10) — Total number of segments received
  - tcpConnTable(13) — TCP connection table (one row per TCP connection)
    - .. tcpConnEntry (1) — Connection entry
      - tcpConnState (1) — TCP connection state : closed, listen, established, ...
      - tcpConnLocalAdd (2) ← Local IP address of the connection
      - tcpConnLocalPort (3) ← Local TCP port of the connection
      - tcpConnRemAdd (4) ← Remote IP address of the connection
      - tcpConnRemPort (5) ← Remote TCP port of the connection

The UDP Group

- udp (mib-2 7)
  - udpInDatagrams (1) — Total number of UDP datagrams delivered to UDP users
  - udpNoPorts (2) — Total number of UDP datagrams for which there was no application at the destination port
  - udpInErrors (3) — Total number of datagrams received with errors
  - udpOutDatagrams(4) — Total number of UDP datagrams sent from the system
  - udpTable (5) — UDP users information table
    - udpEntry (1) — UDP users information entry
      - udpLocalAddress (1) ← Local IP address for this UDP user
      - udpLocalPort (2) ← Local port number for this UDP user

The EGP Group

EGP (External Gateway Protocol) is a route discovery protocol

- egp (mib-2 8)
  - egpInMsgs (1) — Number of EGP messages received without error
  - egpInErrors (2) — Number of EGP messages received with errors
  - egpOutMsgs (3) — Total number of locally generated EGP messages
  - egpOutErrors (4) — Total number of locally generated EGP messages not sent due to resource limitations
  - egpNeighTable (5) — Information neighbor gateways known by the system
    - ...
    - egpNeighEntry (1) — Neighbor gateway information entry
      - egpNeighAddr (2) ← IP address of the neighbor gateway
      - egpNeighIntervalHello(12) — Interval between Hello message retransmissions
      - ...

The Transmission Group

- The Interface group contains generic information that applies to all interfaces
- The Transmission group contains information that relates to a specific type of communication medium
- Example : the Ethernet Interface MIB
  - coaxial cable bus
  - optical fiber
  - twisted pair

The Ethernet Interface MIB

- **dot3 (transmission 7)**
  - **dot3StatsTable (2)**
    - **...**
  - **dot3CollTable (5)**
    - **...**
  - **dot3Tests (6)**
    - **...**
  - **dot3Errors (7)**
    - **...**

Statistics on the trafic for each physical interface : number of collisions, number of MAC transmit errors, number of frames exceeding maximum size, ...

Histogram of collision activity, showing the number of frames that have experienced a given number of collisions

Testing actions at the agent : when a manager accesses them, the corresponding test is performed (example : loopback test)

Error information that occured during a test (example : expected data not received correctly in loopback test)

The SNMP Group

snmp (mib 11)

- snmpInPkts (1) — Nb of PDU delivered to the SNMP entity from transport
- snmpOutPkts (2) — Nb of PDU passed from the SNMP entity to transport
- snmpInBadComName(4) — Nb of PDU delivered to SNMP with unknown comm. name
- snmpInTooBigs (8) — Nb of PDU delivered with tooBig error-status field
- snmpInGetReq (15) — Nb of Get-request PDU processed by the SNMP entity
- snmpInSetReq (17) — Nb of Set-request PDU processed by the SNMP entity
- snmpOutTooBigs (20) — Nb of PDU generated with tooBig error-status field
- snmpOutGetReq (25) — Nb of Get-request PDU generated by the SNMP entity
- snmpOutSetReq (27) — Nb of Set-request PDU generated by the SNMP entity
- snmpEnableAuthenTraps(30) — Authentication-failure traps enabled or disabled (RW)

..

Outline

- General MIB Structure
- **MIB-I and MIB-II Presentation**
  - Overview
  - MIB-II Groups
- **The Private MIBs**

Private MIBs Location

One advantage of SNMP : The SNMP MIB has been designed to provide flexibility for adding new objects

The private.enterprises subtree is used by :

- vendors who might to enhance the management of their devices and make them visible to a management station
- other users who might to experiment proprietary MIB objects

Private MIBs Development

- The vendor generate the <u>formal description</u> of its MIB extension
- He requests a node under the enterprises subtree from the Internet Assigned Numbers Authority, in order to get an <u>unambiguous identification</u> :

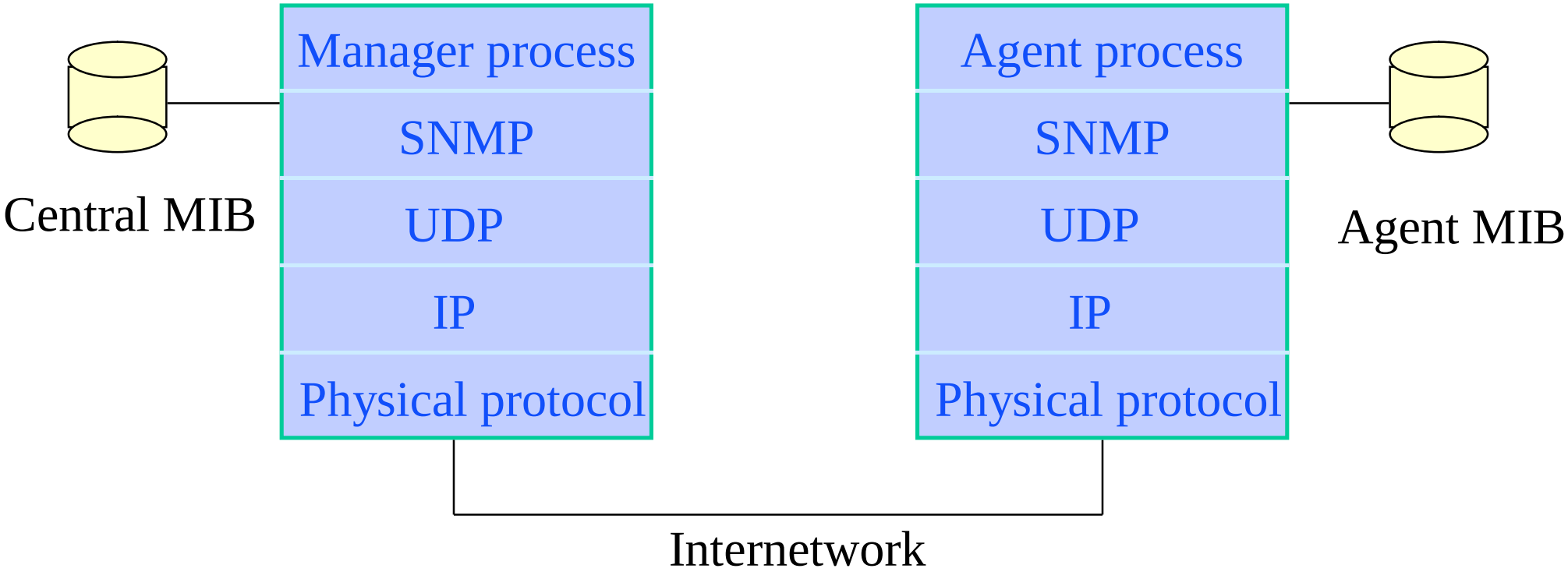
```
myPrivateMib OBJECT IDENTIFIER ::= { enterprises 75 }
```

- He provides this private MIB to clients, in addition to its product
- This private MIB must be loaded in the management station

SNMP V1
Administration ++

SNMP Basic Architecture

- **SNMP is designed to run on the top of the User Datagram Protocol**



Resources, Managed Objects, MIB (1/5)

How do we Model the Management Information ?

Network Management World | **«Real» World**

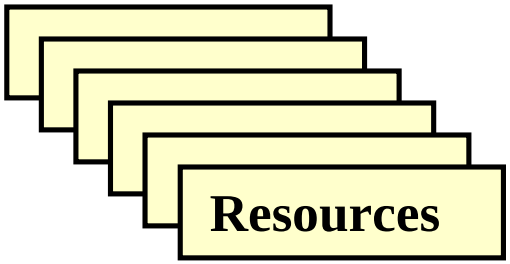
Manager | **Agent**

Resources, Managed Objects, MIB (2/5)

How do we Model the Management Information ?

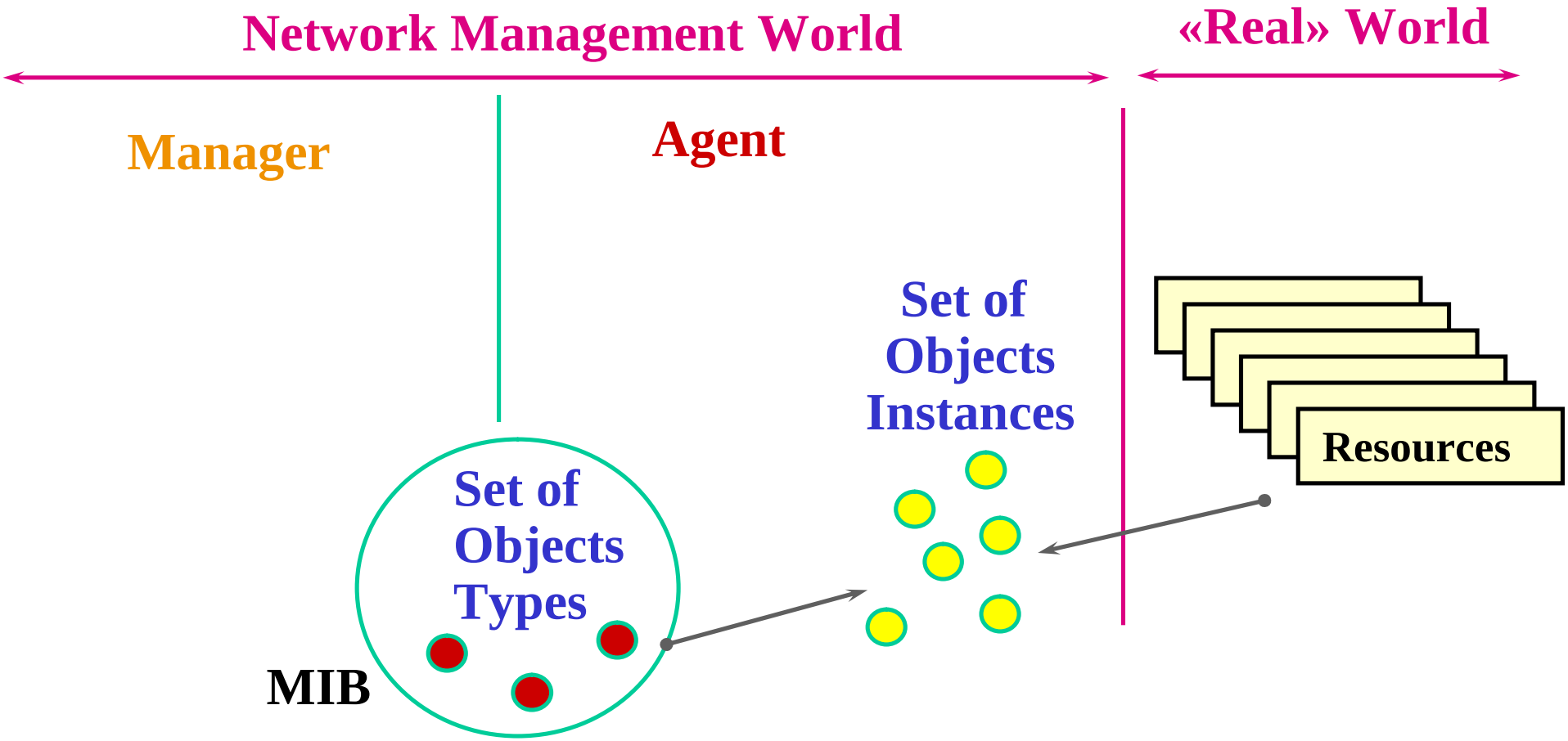
Network Management World | **«Real» World**

Manager | **Agent**



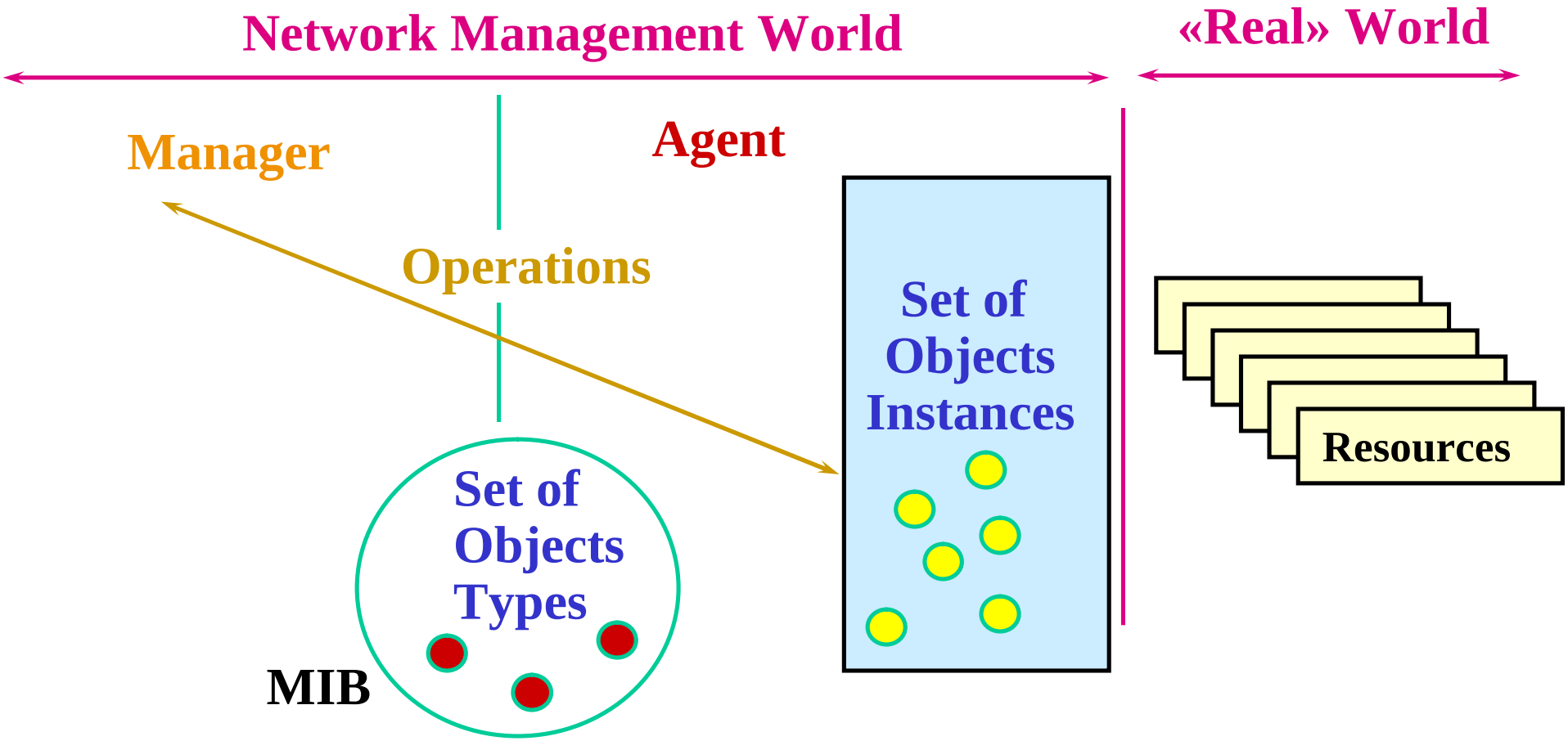
Resources, Managed Objects, MIB (3/5)

How do we Model the Management Information ?



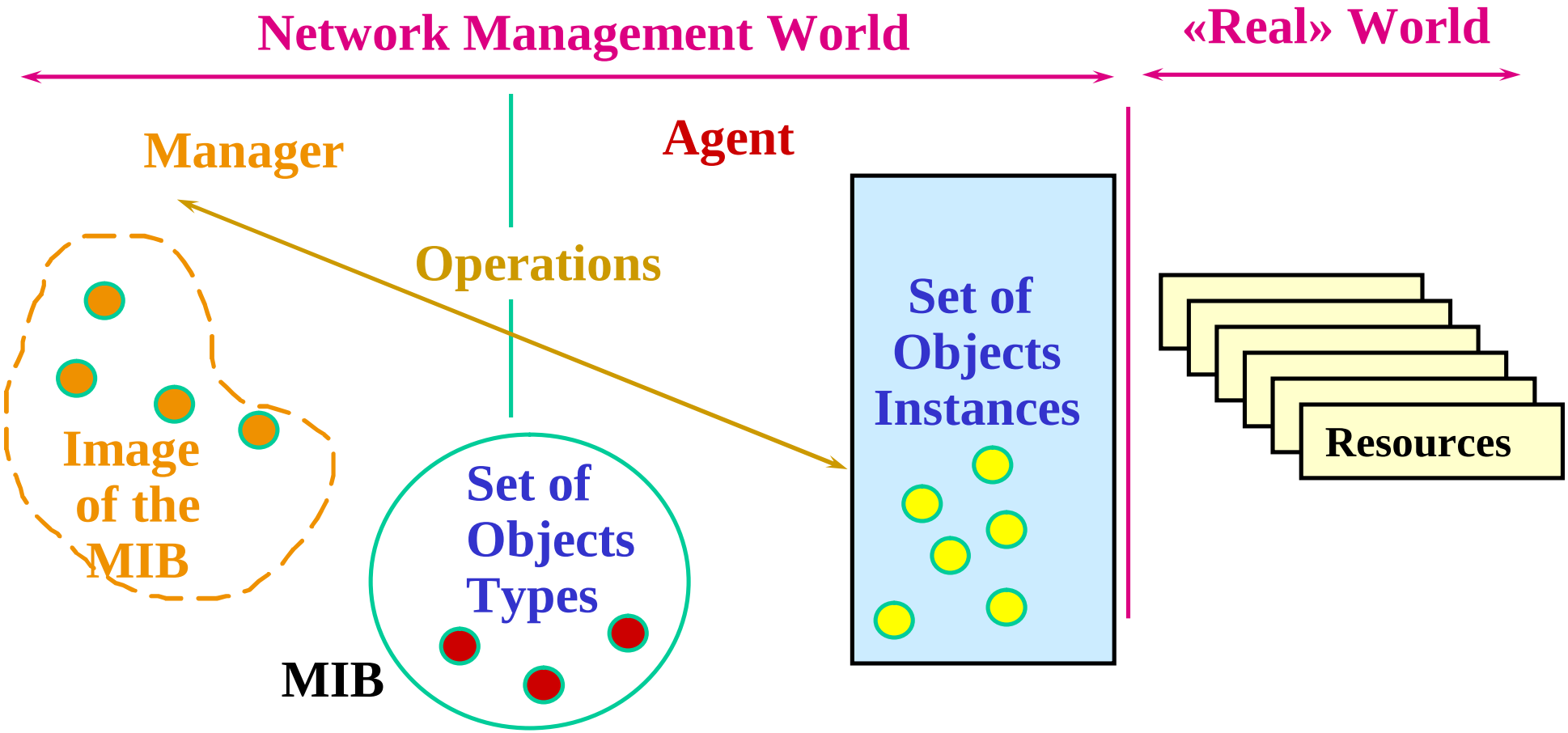
Resources, Managed Objects, MIB (4/5)

How do we Model the Management Information ?



Resources, Managed Objects, MIB (5/5)

How do we Model the Management Information ?



Connectionless Protocol

- Because it uses UDP, SNMP is a connectionless protocol
- No guarantee that the management traffic is received at the other entity
- Advantages :
  - reduced overhead
  - protocol simplicity
- Drawbacks :
  - connection-oriented operations must be built into upper-layer applications, if reliability and accountability are needed

SNMP Operations

- SNMP provides three simple operations :
  - *GET* :

    Enables the management station to retrieve object values from a managed station
  - *SET* :

    Enables the management station to set object values in a managed station
  - *TRAP* :

    Enables a managed station to notify the management station of significant events
- SNMP allows multiple accesses with a single operation
- Adding and deleting object instances (e.g. in tables) is not normalized by RFC : it is an agent-specific implementation

SNMP Protocol Data Units

- ***Get Request*** **:**

  Used to obtain object values from an agent

- ***Get-Next Request*** **:**

  Similar to the Get Request, except it permits the retrieving of the next object instance (in lexicographical order) in the MIB tree

- ***Set Request*** **:**

  Used to change object values at an agent

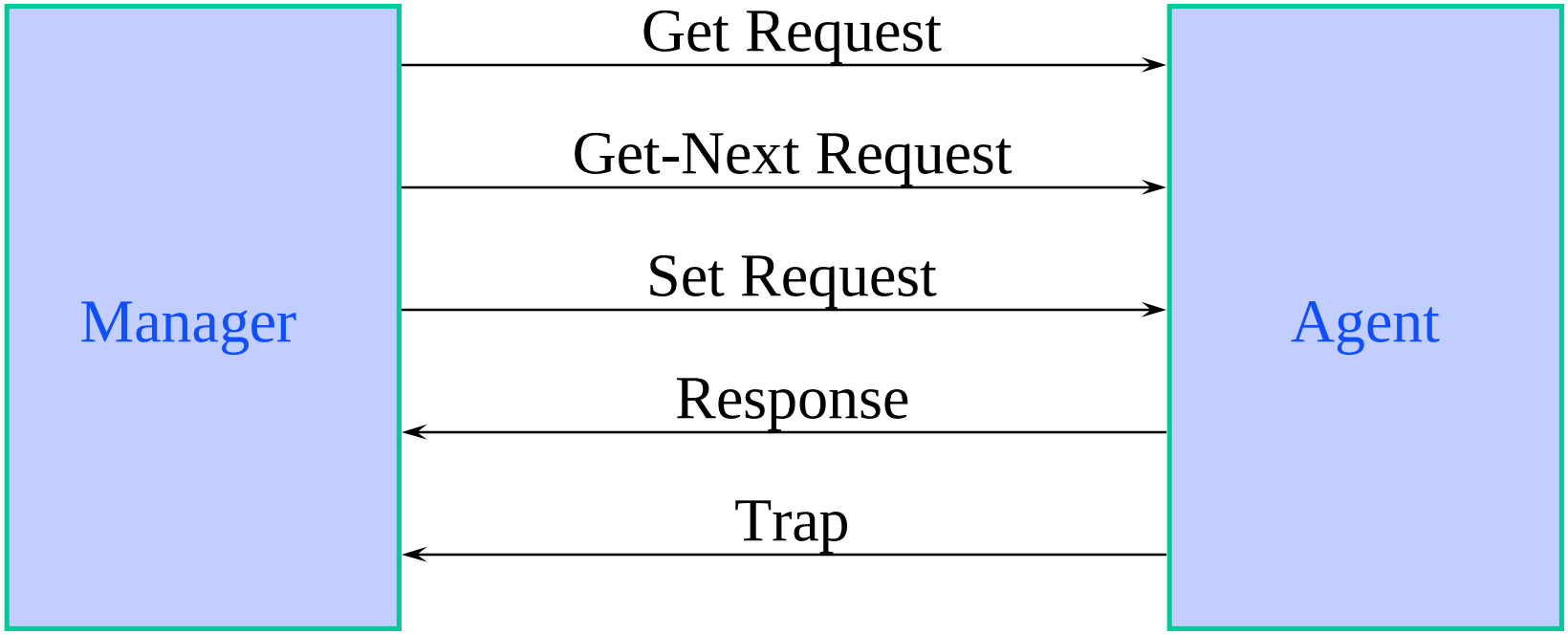
- ***Response*** **:**

  Responds to the Get Request, Get-Next Request and Set Request PDUs

- ***Trap*** **:**

  Enables an agent to report an event to the management station (no response from the manager entity)

SNMP PDUs Direction



SNMP Port Numbers (1/2)

By convention, the UDP port numbers used for SNMP are :

161 (Requests) and 162 (Traps)

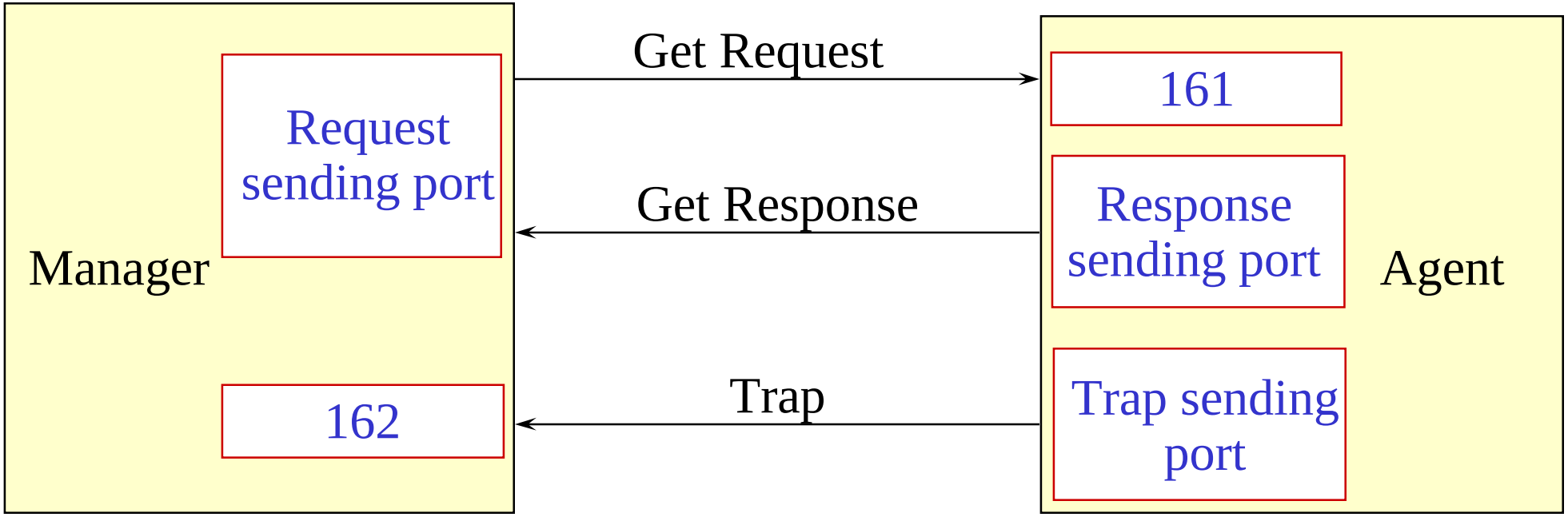
Manager behaviour :

- listens for agent traps on local port 162
- sends requests to port 161 of remote agent

Agent behaviour :

- listens for manager requests on local port 161
- sends traps to port 162 of remote manager

SNMP Port Numbers (2/2)



SNMP Overall Message Format

All SNMP PDUs are built in the same way :

Version	Community	SNMP V1 PDU

SNMP version

(SNMP V1 is version 0)

Community name

PDU-type dependant

Get Request Operation

The *Get Request* operation accesses only to <u>instances of leaf objects</u>

mib2**(1.3.6.1.2.1)**

interfaces**(2)**

ifTable**(2)**

ifEntry**(1)**

ifIndex**(1)** ifPhysAddress**(6)** ifAdminStatus**(7)**

1	00:00:39:20:04	1 (up)
2	08:00:56:16:11	3 (testing)
8	00:00:b4:02:33	2 (down)

GetRequest (ifPhysAddress.2)

Response (ifPhysAddress.2 = 08:00:56:16:11)

Get Request in Tabular Objects

The *Get Request* operation only allows the retrieval of <u>leaf objects</u>

<u>Consequence</u> : it is not possible to retrieve

- an entire row of a table (by referencing the entry object)
- an entire table (by referencing the table object)

<u>Solution</u> : retrieve an entire row by including each object instance of the table in the Variable Binding field

GetNext Request Operation

The *Get Next* Request has <u>three advantages</u>, compaired to *Get* :

- Allows the retrieving of unknown objects
- More efficient way to retrieve a set of object values when some are not implemented by the agent
- Allows the retrieving of an entire table, without knowing its content

Retrieving Unknown Objects

No requirement that the supplied identifier represents an object instance

The *Get Next* operation can be used to discover the MIB structure

mib2**(1.3.6.1.2.1)**

interfaces**(2)**

ifTable**(2)**

ifEntry**(1)**

ifIndex**(1)** ifPhysAddress**(6)** ifAdminStatus**(7)**

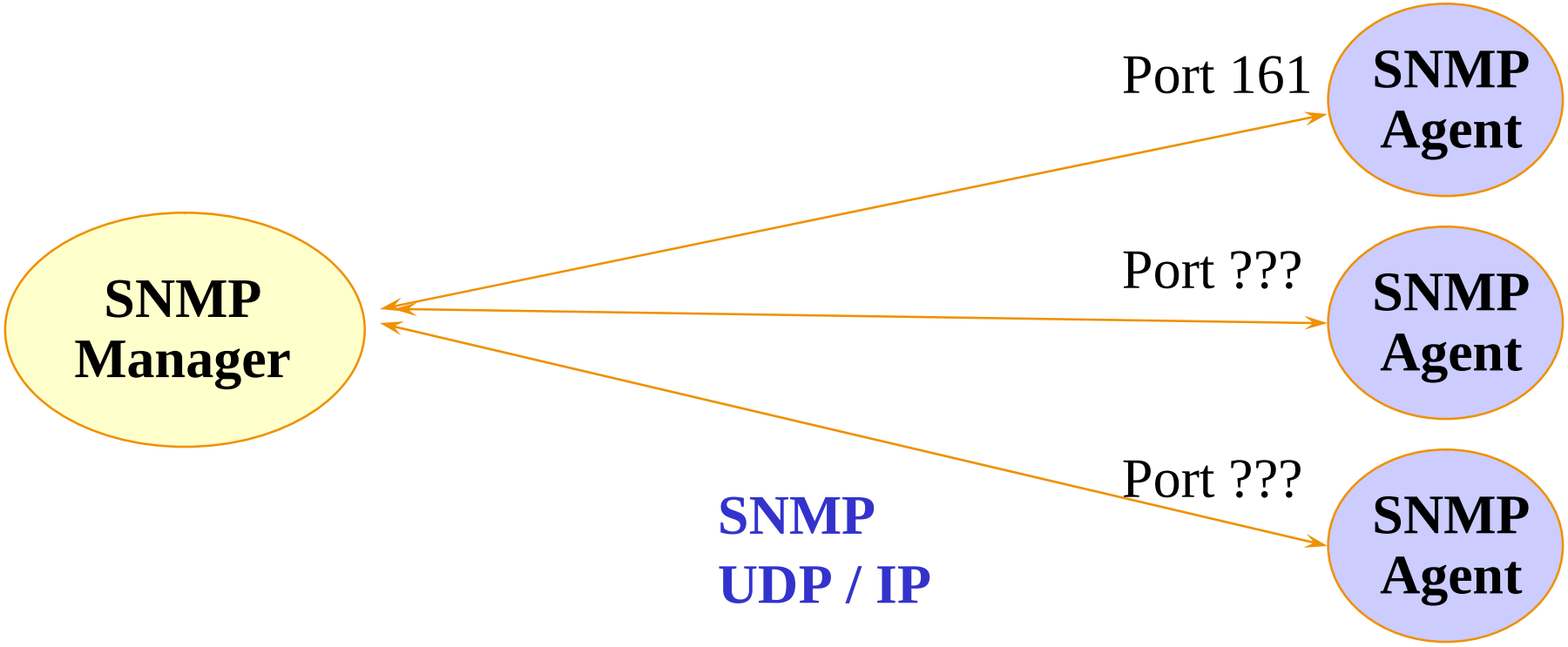
1	00:00:39:20:04	1 (up)
2	08:00:56:16:11	3 (testing)
8	00:00:b4:02:33	2 (down)

GetNextRequest (interfaces)

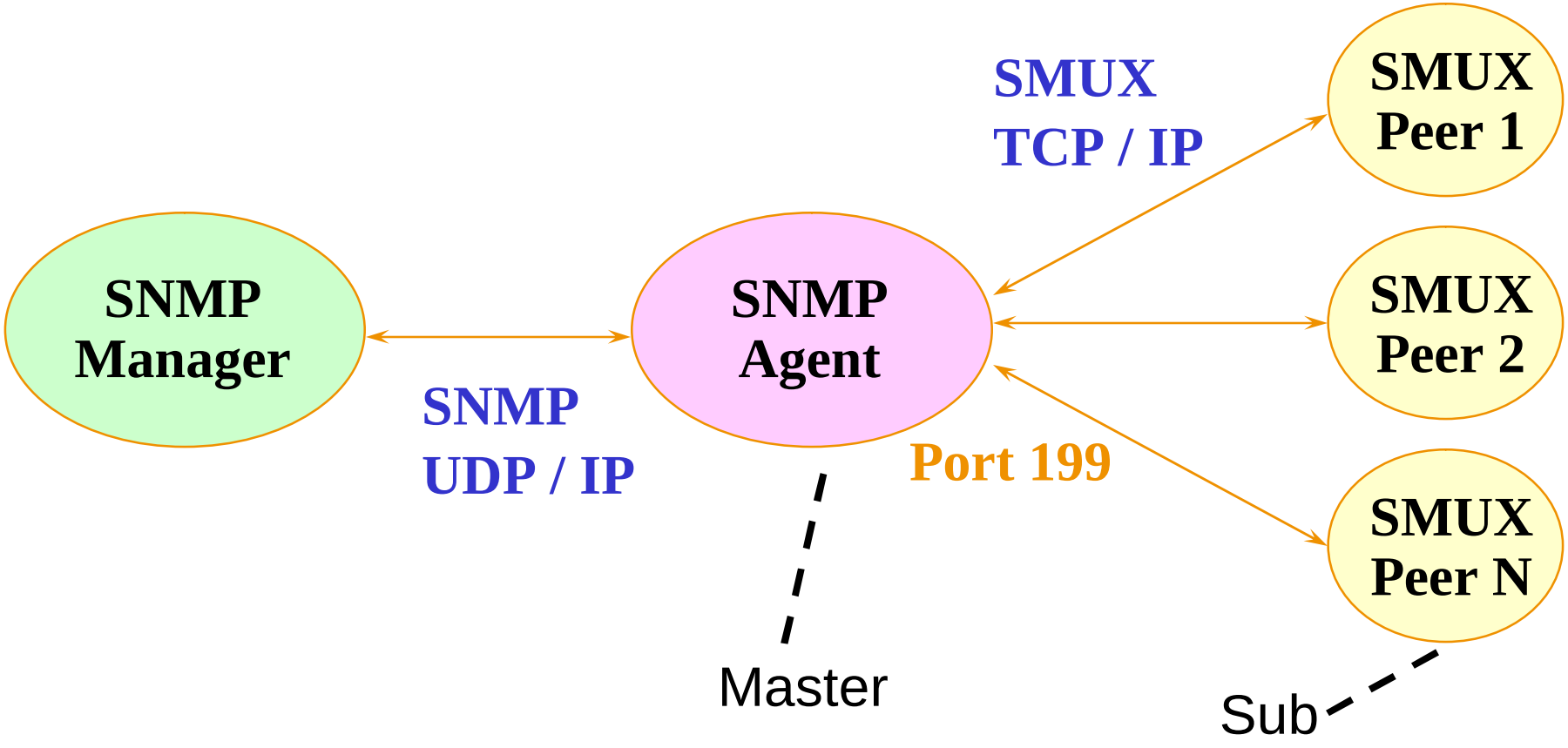
Response (ifIndex.1 = 1)

The manager learns that the first supported object in the *interfaces* sub-tree is *ifIndex*

Multi Agent on same Host



SMUX Architecture Overview
Snmp MUltipleX
(Master / Sub)

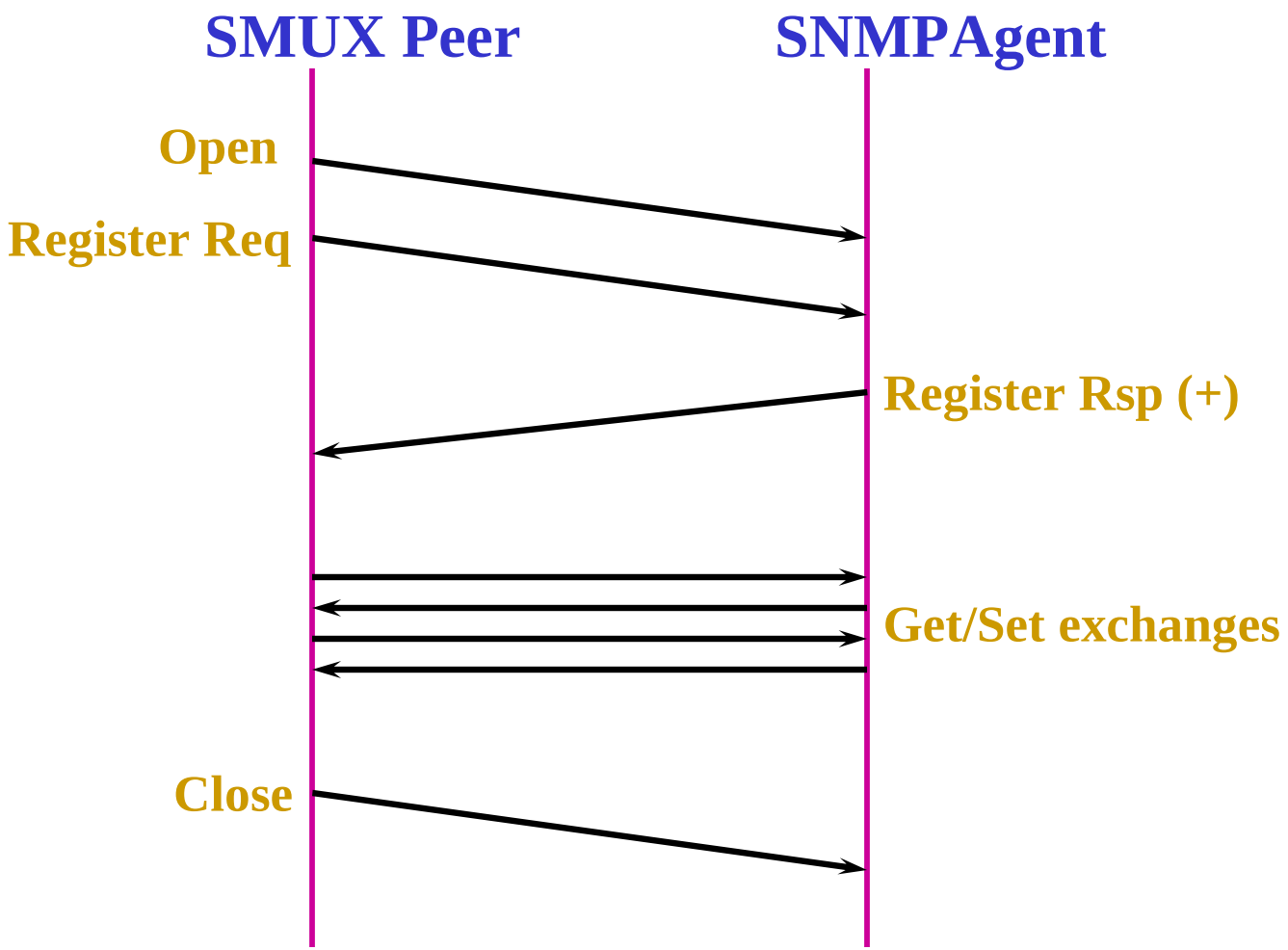


Elements of Protocol

- **Protocol elements derived from SNMP**
  - Get Request
  - Get Next Request
  - Set Request
  - Get Response
  - Trap
- **Additional Protocol elements**
  - Open
  - Close
  - Set Request
  - Register Request
  - Register Response
  - Commit and Rollback

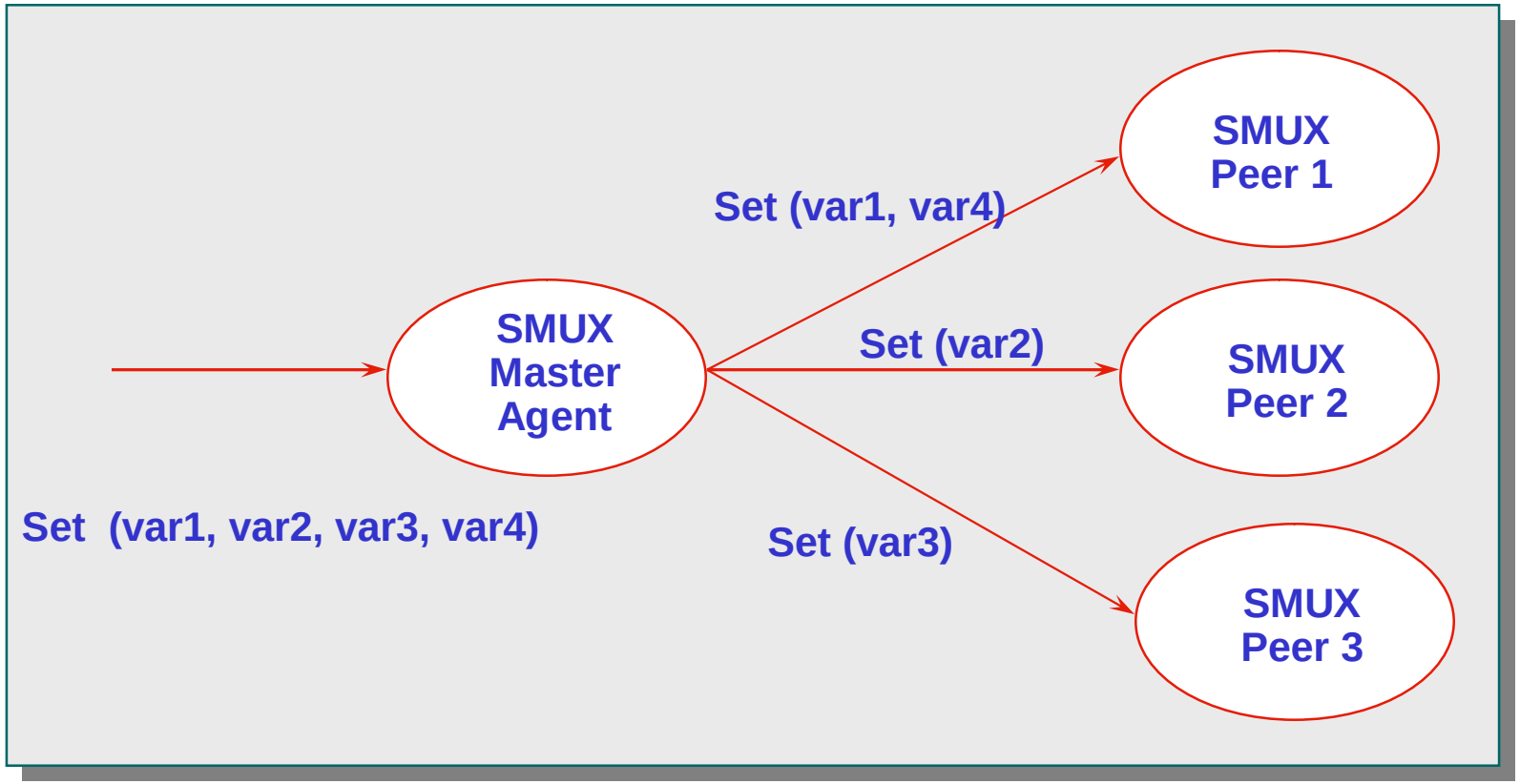
Protocol Overview

Regular Session

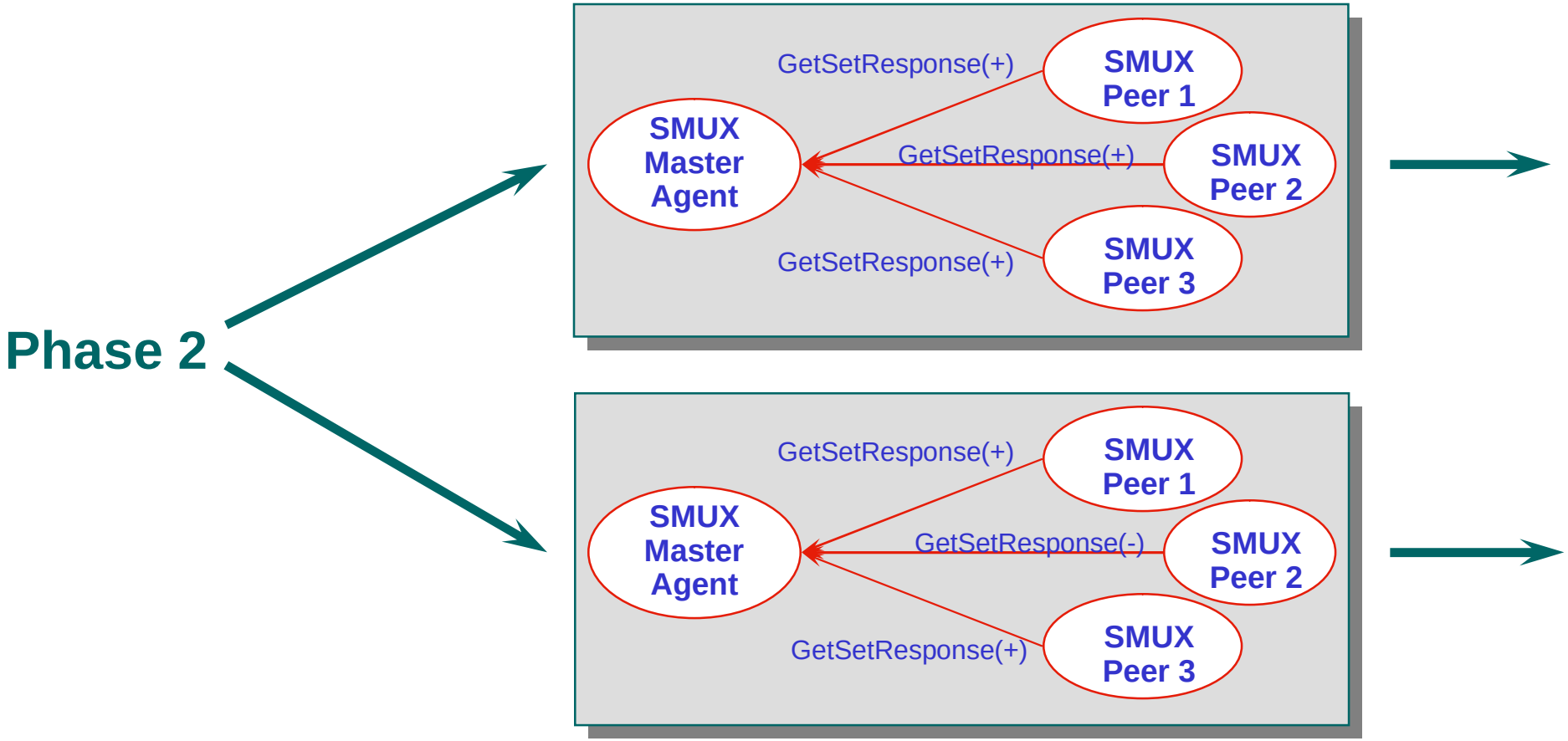


Set Operation (1/3)

Phase 1

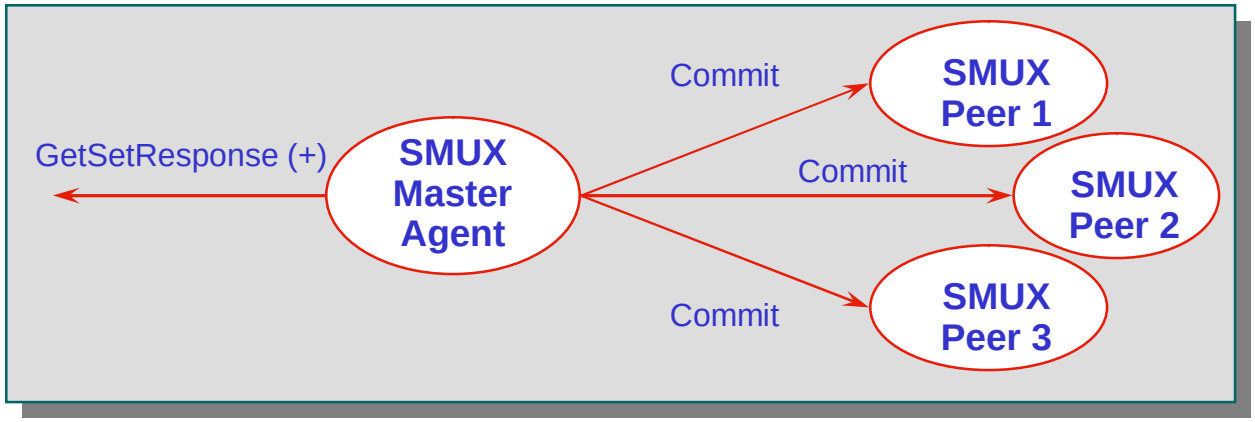


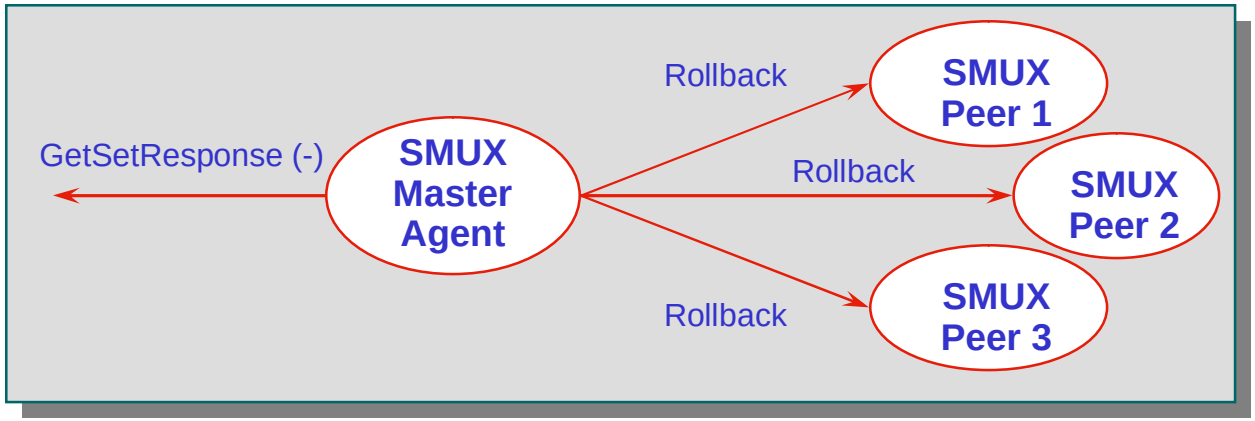
Set Operation (2/3)



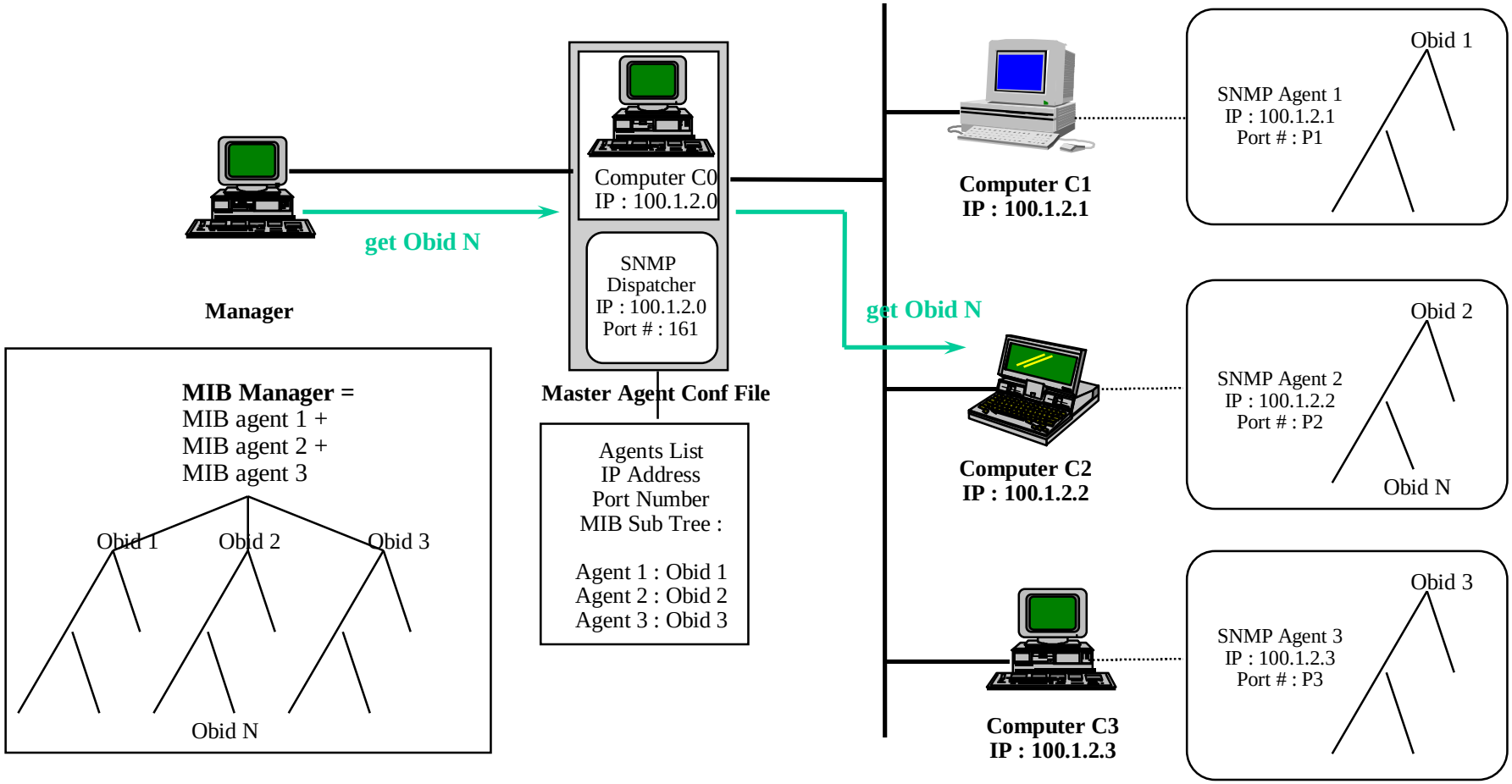
Set Operation (3/3)

Phase 3

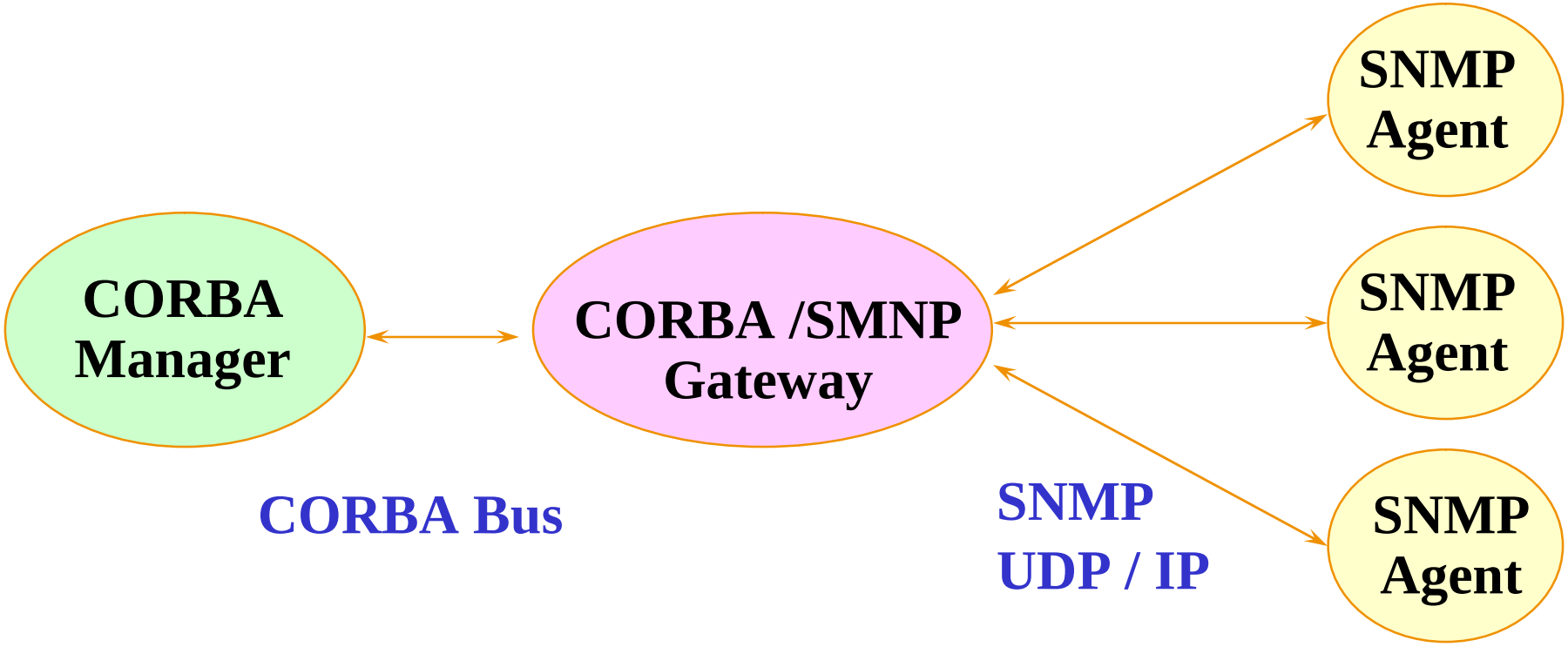




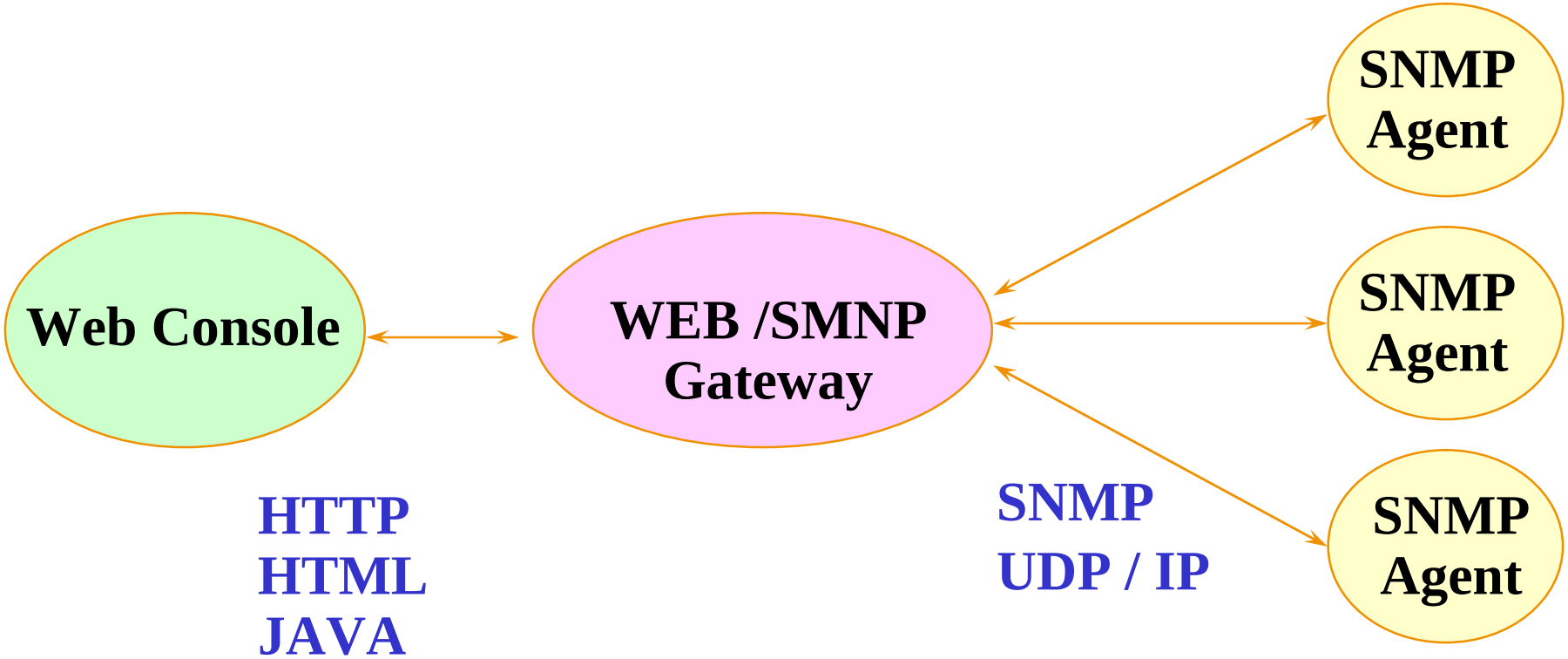
SNMP Dispatcher



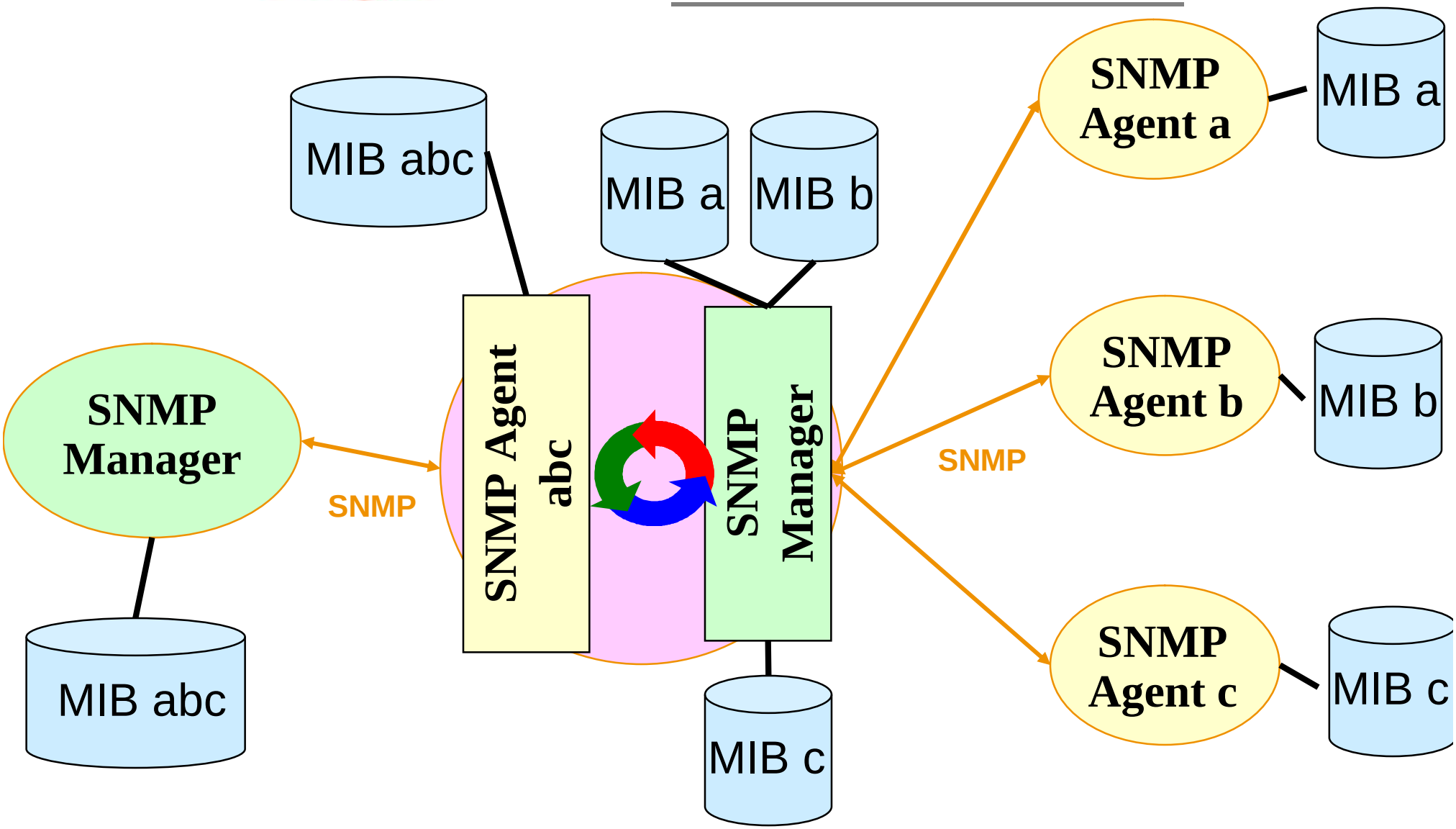
New SNMP Architecture (1/3)



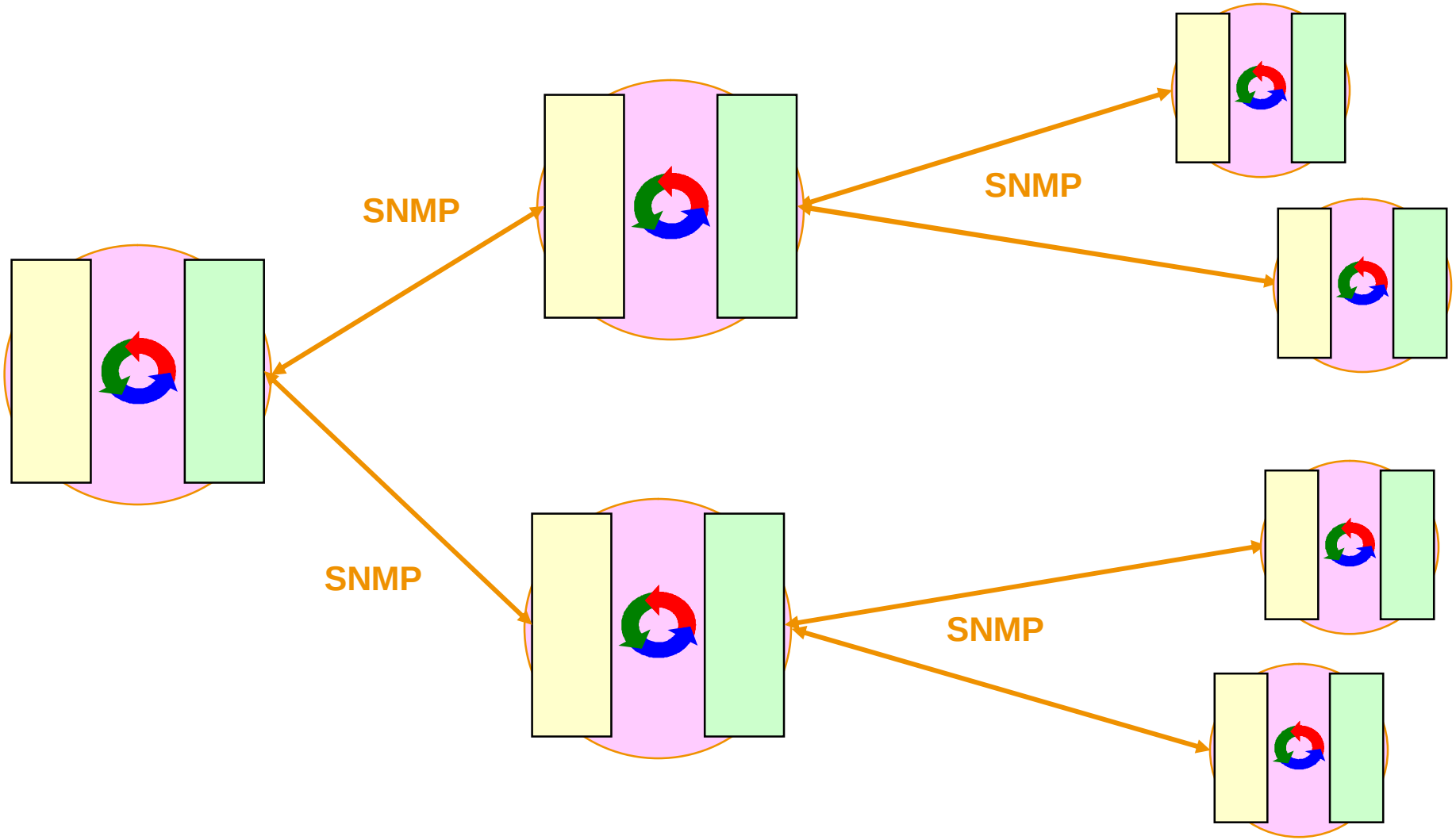
New SNMP Architecture (2/3)



New SNMP Architecture (3/3)
MEDIATION DEVICE



New SNMP Architecture (3/3)
MEDIATION DEVICE



Systems and Technologies
Master / Sub

LINUX	è	SMUX / AgentX (Package UCD-SNMP)
COMPAQ*	è	eSNMP
SUN SOLARIS	è	SNMP Dispatcher / Solstice
HP*	è	No Master / Sub Capabilities
Win NT/2000 on	è	Specific Master Agent (Sub-Agent based DLL)

* Just before fusion J