Grenoble INP ENSIMAG

SNMP
Network Management

Prof. Andrzej Duda

duda@imag.fr

http://duda.imag.fr

Contents

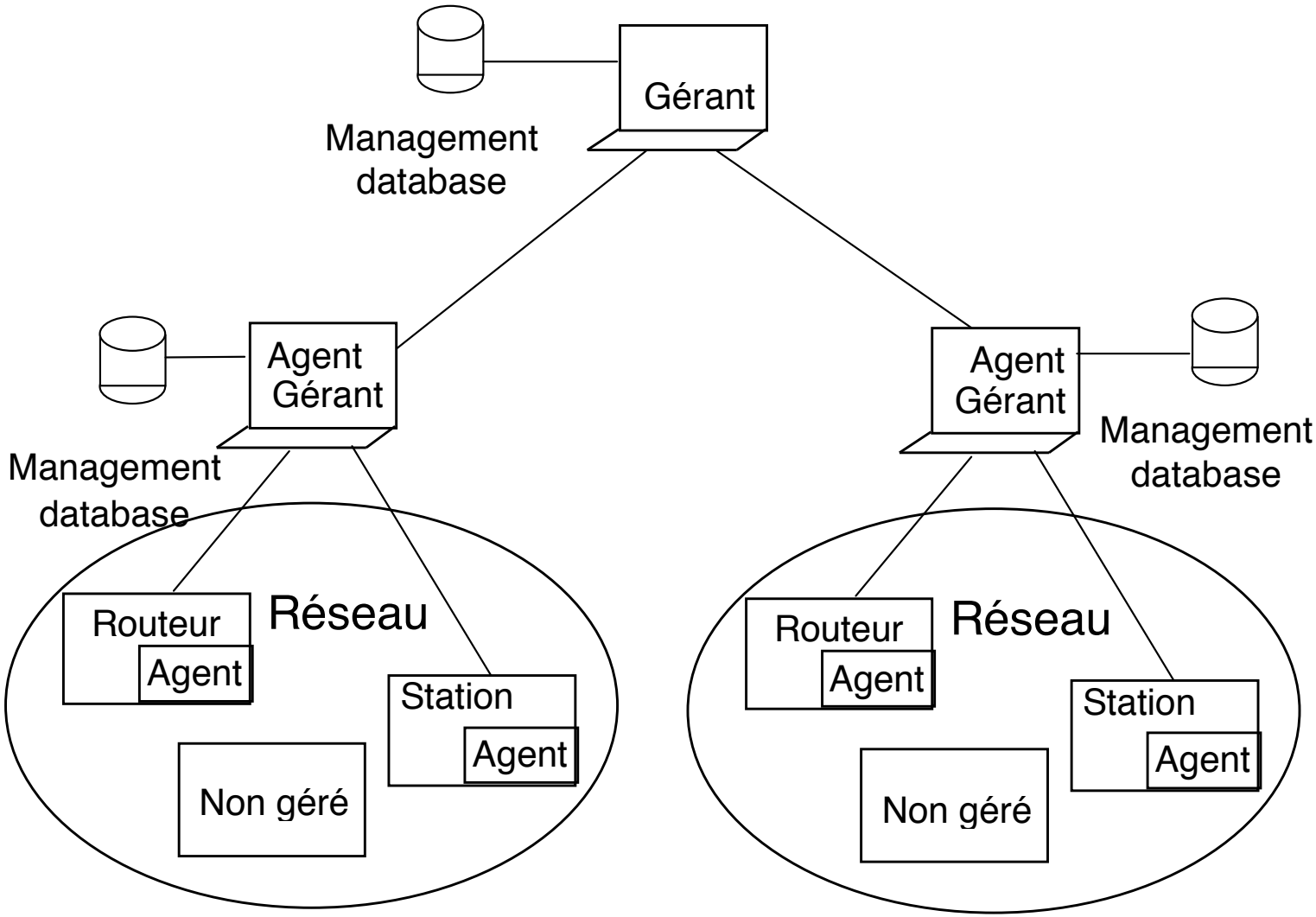
- Principles of network managment
- Information model
  - ASN.1
  - BER
- Management Information Base (MIB)
  - Type definitions
  - Object identifiers
  - Object instances
  - Index
- SNMP v1, v2, v3
- Slides based on the textbook "Network Management, Principles and Practice" by Mani Subramanian

Main standards

- OSI/CMIP (Common Management Information Protocol)
  - powerful, structured
  - requires a lot of ressources

- IETF/SNMP (Simple Network Management Protocol)
  - simple (too simple? SNMPv1) – weak security of SNMPv1
  - lightweight agents, popular
  - versions: SNMP v1, v2, v3

Network Management Systems



Management Model

- **Organization Model**:
  - client-server architecture
  - components and their functionality
  - agents, managers
- **Information Model**:
  - managed objects
  - information base (MIB)
- **Communication model:**
  - protocol and its functions
- **Functional model**:
  - how to use the system for configuration, management, fault detection and repair
  - performance, security, accounting

Information Model

- Devices represented as objects
- NMS operates on objects having attributs
- Objects stored in a Management Information Base (MIB)
  - views: agent MIB, manager MIB (union of all MIBs)
- SMI (Structure of Management Information)
  - defines the syntax and the semantics of MIB objects
  - uses ASN.1: Abstract Syntax Notation One
  - semantics – textual description

Examples

- SNMP (scalar objects)
  - Identifier: sysUpTime or 1.3.6.1.2.1.1.3 -- OID
  - Syntax: TimeTicks – data type
  - Access: read-only
  - Status: mandatory
  - Description: "time in 1/100s since reboot" -- semantics
  - TimeTicks ::= [**APPLICATION** 3] **IMPLICIT INTEGER** (0..4294967295)
- CMIP (complex object)
  - Object class: packetCounter
  - Attributes: single-valued
  - Operations: get, set
  - Behavior: retrieves or reset values
  - Notification: generate notifications on new value

Communication model

- Client-server
  - Manager sends queries (SNMP: Get, Get-Next, Set)
  - Agent sends responses
- Asynchronous
  - Agent sends notifications (SNMP: Trap)
- Security
  - defined by a given protocol
- Coding and syntax (presentation layer)
  - ASN.1 (Abstract Syntax Notation) (used in the Information Model)
  - BER (Basic Encoding Rules), exchange format independent of the machine representation

Information Model

- Managed Objects
  - ASN.1, BER, OID
- MIB
- SMI

Managed Objects

- Managed objects can be
  - Network elements (hardware, system)
    - hubs, bridges, routers, transmission facilities
  - Software (non-physical)
    - programs, algorithms
  - Administrative information
    - contact person, name of group of objects (IP group)

Managed Object

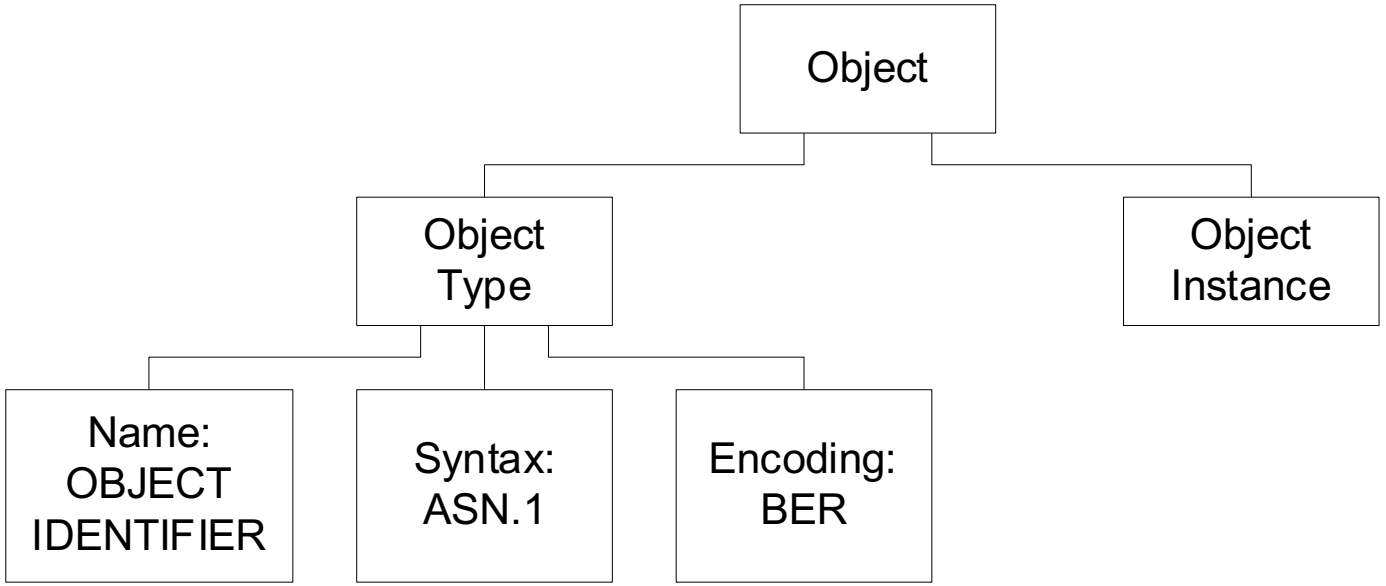


Figure 4.10 Managed Object : Type and Instance

Example of object definition

- An invoice defined in ASN.1

```
PageNumber ::= INTEGER
Trade-Message ::= SEQUENCE
        {invoice-no   INTEGER,
        name  GraphicString,
       details SEQUENCE OF
                SEQUENCE
                {part-no INTEGER, quantity INTEGER},
        charge REAL }
```

- value: { 10, "HUB purchase", { {2, 1}, {3, 4}}, 100.1 }

ASN.1

Primitive Data Types

Structure	Data Type	Comments
Primitive types	INTEGER	Subtype INTEGER (n1..nN) Special case: Enumerated INTEGER type
	OCTET STRING	8-bit bytes binary and textual data Subtypes can be specified by either range or fixed
	OBJECT IDENTIFIER	Object position in MIB
	NULL	Placeholder

Data types

Constructor or Structured Data Type: SEQUENCE

- Like C record

SEQUENCE { <type1>, <type2>,…, <typeN> }

	Object	OBJECT IDENTIFIER	ObjectSyntax
1	ipAdEntAddr	{ipAddrEntry 1}	IpAddress
2	ipAdEntIfIndex	{ipAddrEntry 2}	INTEGER
3	ipAdEntNetMask	{ipAddrEntry 3}	IpAddress
4	ipAdEntBcastAddr	{ipAddrEntry 4}	INTEGER
5	ipAdEntReasmMaxSize	{ipAddrEntry 5}	INTEGER
6	ipAddrEntry	{ipAddrTable 1}	SEQUENCE

```
List:   IpAddrEntry  ::=
            SEQUENCE {
                ipAdEntAddr             IpAddress
                ipAdEntIfIndex          INTEGER
                ipAdEntNetMask          IpAddress
                ipAdEntBcastAddr        INTEGER
                ipAdEntReasmMaxSize     INTEGER (0..65535)
            }
```

Managed Object IpAddrEntry as a list

Data types

Constructor or Structured Data Type: SEQUENCE OF

- Like C array

SEQUENCE OF <entry>
where <entry> is a list constructor

	Object Name	OBJECT IDENTIFIER	Syntax
7	ipAddrTable	{ip 20}	SEQUENCE OF

Table: IpAddrTable ::=
SEQUENCE OF IpAddrEntry

Managed Object ipAddrTable as a table

Data types

SEQUENCE OF Example

Title: System Information : router1.gatech.edu
Name or IP Address: 172.16252.1

Index	Interface	IP address	Network Mask	Network Address	Link Address
23	LEC.1.0	192.168.3.1	255.255.255.0	192.168.3.0	0x00000C3920B4
25	LEC.3.9	192.168.252.15	255.255.255.0	192.168.252.0	0x00000C3920B4
13	Ethernet2/0	172.16..46.1	255.255.255.0	172.16..46.0	0x00000C3920AC
16	Ethernet2/3	172.16.49.1	255.255.255.0	172.16.49.0	0x00000C3920AF
17	Ethernet2/4	172.16.52.1	255.255.255.0	172.16.52.0	0x00000C3920B0
9	Ethernet1/2	172.16.55.1	255.255.255.0	172.16.55.0	0x00000C3920A6
2	Ethernet 0/1	172.16.56.1	255.255.255.0	172.16.56.0	0x00000C39209D
15	Ethernet2/2	172.16.57.1	255.255.255.0	172.16.57.0	0x00000C3920AE
8	Ethernet1/1	172.16.58.1	255.255.255.0	172.16.58.0	0x00000C3920A5
14	Ethernet2/1	172.16.60.1	255.255.255.0	172.16.60.0	0x00000C3920AD

- The above example (Figure 4.3) uses part of the IP MIB discussed for SEQUENCE OF construct

Data types

Defined or Application Data Type

Defined types	NetworkAddress	Not used
	IpAddress	Dotted decimal IP address
	Counter	Wrap-around, non-negative integer, monotonically increasing, max 2^32 -1
	Gauge	Capped, non-negative integer, increase or decrease
	TimeTicks	Non-negative integer in hundredths of second units
	Opaque	Application-wide arbitrary ASN.1 syntax, double wrapped OCTET STRING

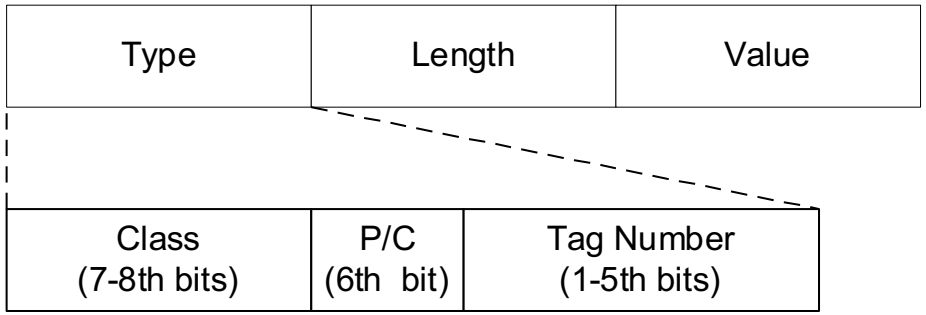
- Defined data types are simple or base types
- Opaque is used to create data types based on previously defined data types

Example

```
ApplicationSyntax ::= CHOICE {
                        address   NetworkAddress,
                        counter   Counter,
                        gauge     Gauge,
                        ticks     TimeTicks,
                        arbitrary Opaque
Counter ::=  [APPLICATION 1] IMPLICIT INTEGER
(0..4294967295)
Gauge ::= [APPLICATION 2] IMPLICIT INTEGER
(0..4294967295)
TimeTicks ::= [APPLICATION 3] IMPLICIT INTEGER
(0..4294967295)
```

Encoding

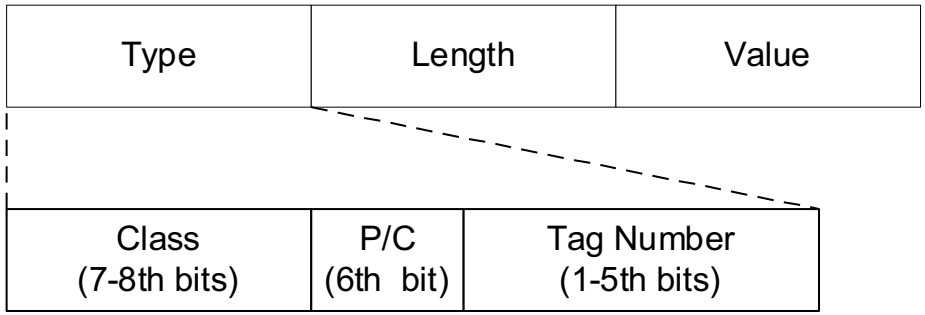
- Basic Encoding Rules (BER)
- Type, Length, and Value (TLV)



- Class
  - 00 UNIVERSAL
  - 01 APPLICATION
  - 10 Context-specific
  - 11 Private
- P/C – Primitive or Composite
- Examples:
  - BOOLEAN – UNIVERSAL 1 -- 1 is a Tag
  - INTEGER – UNIVERSAL 2 -- 2 is a Tag

Encoding

- Basic Encoding Rules (BER)
- Type, Length, and Value (TLV)



- Length
  - short: bit 7 = 0, Length on 7 bits
  - long: bit 7 = 1, number of bytes for Length on 7 bits
  - then Length

- SNMP Data Types and Tags

Type	Tag
OBJECT IDENTIFIER	UNIVERSAL 6
SEQUENCE	UNIVERSAL 16
IpAddress	APPLICATION 0
Counter	APPLICATION 1
Gauge	APPLICATION 2
TimeTicks	APPLICATION 3
Opaque	APPLICATION 4

Example

IMPLICIT : replaces the value of the tag (TLV)

EXPLICIT: tag is added in front of the existing tag (TL TLV)

Example:

TimeTicks 132320314 (15 days 7:33:22) coded as:

0x43 2bits 01: APPLICATION, 1bit: simple type, 3 - tag

0x4 length

0x07 0xe3 0x0c 0x3a value

OBJECT IDENTIFIER

OSI Management Information Tree

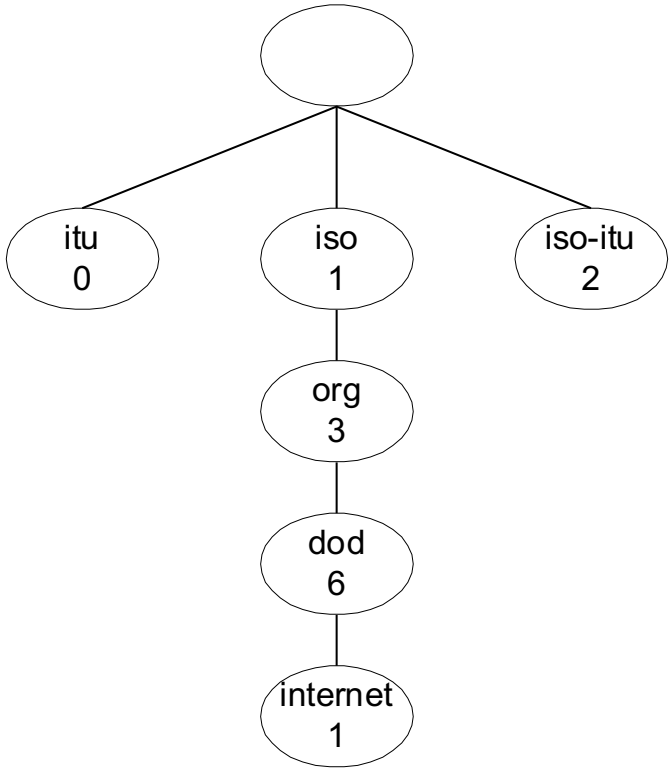


Figure 3.8 **OSI Management Information Tree**

OBJECT IDENTIFIER

OSI Management Information Tree

- iso International Standards Organization
  itu International Telecommunications Union
  dod Department of Defense

- Designation:

iso 1
org 1.3
dod 1.3.6
internet 1.3.6.1 – all internet managed objects will start with this

{ 1 3 6 1 }

OBJECT IDENTIFIER

Internet Subnodes

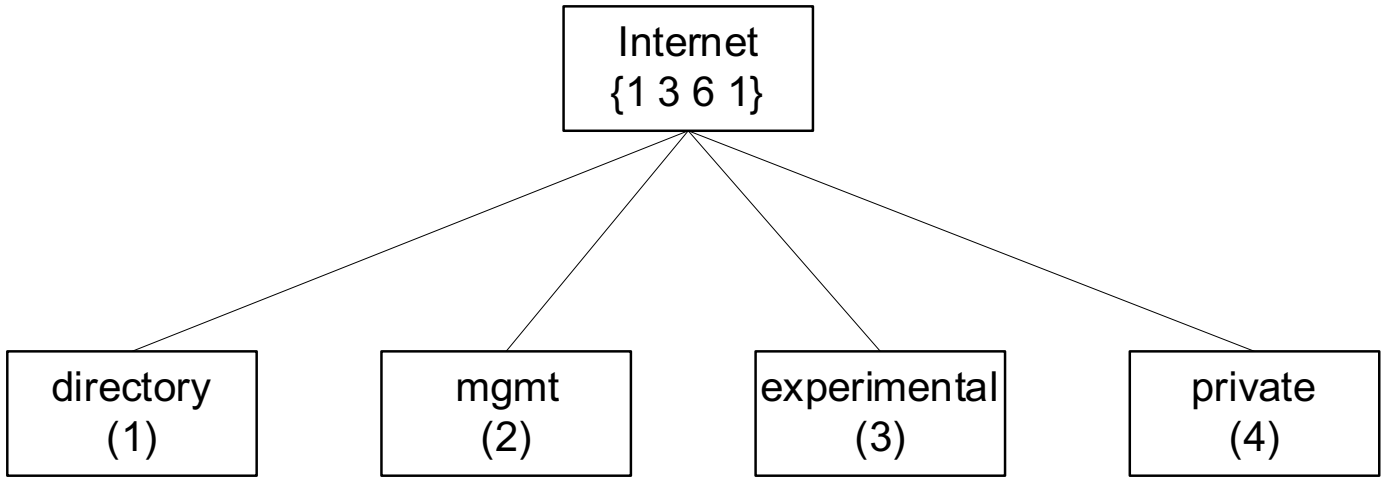


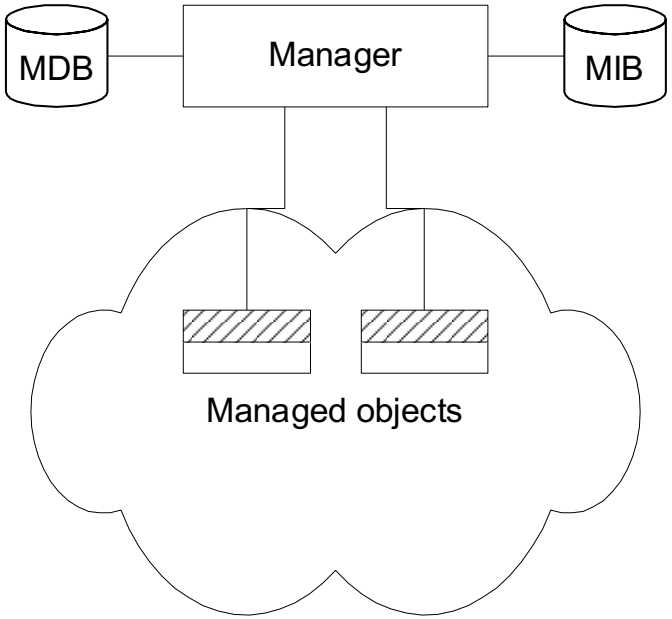
Figure 4.13 Subnodes under Internet Node in SNMPv1

OBJECT IDENTIFIER

Internet Subnodes

```
directory       OBJECT IDENTIFIER ::= {internet 1}
mgmt            OBJECT IDENTIFIER ::= {internet 2}
experimental    OBJECT IDENTIFIER ::= {internet 3}
private         OBJECT IDENTIFIER ::= {internet 4}
```

MIB



MDB Management Database
MIB Management Information Base

Agent process

MIB

- Properties associated with nodes
  - Name – Object Identifier
  - Access mode (read-only, RW, Write-only)
  - Structure, syntax, type
- Names

internet **OBJECT IDENTIFIER** ::= { iso org(3) dod(6) 1 }

mgmt **OBJECT IDENTIFIER** ::= { internet 2 }

- Values
  - only at leaves

MIB - OBJECT IDENTIFIER

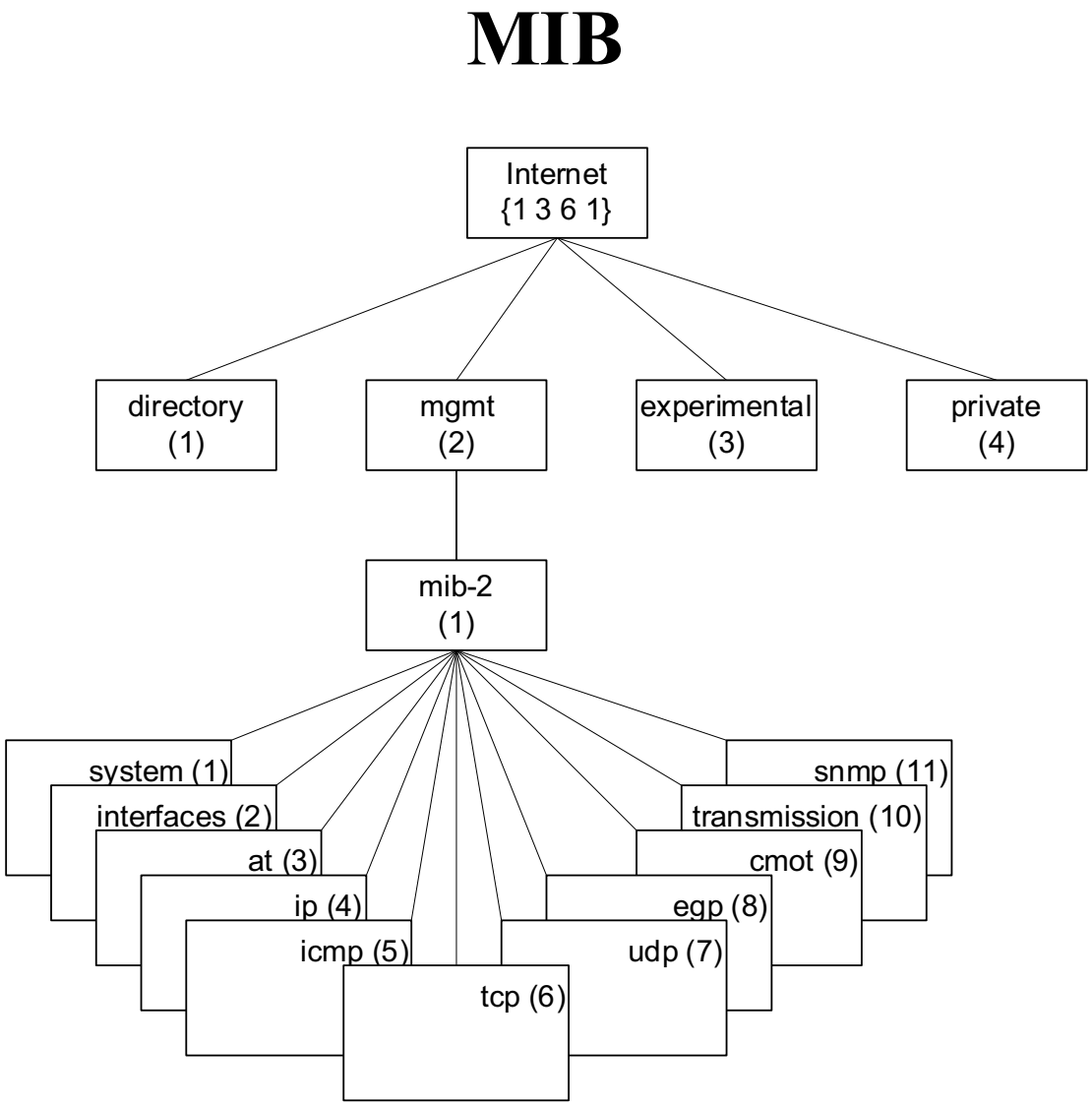


Figure 4.26 Internet MIB-II Group

SMI – type definition

```
RFC1155-SMI DEFINITIONS ::= BEGIN
        EXPORTS -- EVERYTHING
                internet, directory, mgmt,
                experimental, private, enterprises,
                OBJECT-TYPE, ObjectName, ObjectSyntax, SimpleSyntax,
                ApplicationSyntax, NetworkAddress, IpAddress,
                Counter, Gauge, TimeTicks, Opaque;
        -- the path to the root
        internet        OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
        directory       OBJECT IDENTIFIER ::= { internet 1 }
        mgmt            OBJECT IDENTIFIER ::= { internet 2 }
        experimental    OBJECT IDENTIFIER ::= { internet 3 }
        private         OBJECT IDENTIFIER ::= { internet 4 }
        enterprises     OBJECT IDENTIFIER ::= { private 1 }
```

SMI – type definition

```
-- definition of object types
OBJECT-TYPE MACRO ::=
BEGIN
        TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                          "ACCESS" Access
                          "STATUS" Status
        VALUE NOTATION ::= value (VALUE ObjectName)
        Access ::= "read-only"
                 | "read-write"
                 | "write-only"
                 | "not-accessible"
        Status ::= "mandatory"
                 | "optional"
                 | "obsolete"
END
```

SMI – type definition

```
-- names of objects in the MIB
        ObjectName ::= OBJECT IDENTIFIER
-- syntax of objects in the MIB
        ObjectSyntax ::=  CHOICE {
            simple SimpleSyntax,
            application-wide ApplicationSyntax
            }
        SimpleSyntax ::=
            CHOICE {
                number      INTEGER,
                string      OCTET STRING,
                object      OBJECT IDENTIFIER,
                empty       NULL
            }
```

SMI – type definition

```
ApplicationSyntax ::=
        CHOICE {
            address         NetworkAddress,
            counter         Counter,
            gauge           Gauge,
            ticks           TimeTicks,
            arbitrary       Opaque
        -- other application-wide types, as they are
        -- defined, will be added here
            }
        -- application-wide types
        NetworkAddress ::=
            CHOICE {
            internet        IpAddress
            }
```

SMI – type definition

```
IpAddress ::=   [APPLICATION 0]          -- in network-byte
order
        IMPLICIT OCTET STRING (SIZE (4))
Counter ::= [APPLICATION 1]
        IMPLICIT INTEGER (0..4294967295)
Gauge ::=   [APPLICATION 2]
        IMPLICIT INTEGER (0..4294967295)
TimeTicks ::=   [APPLICATION 3]
        IMPLICIT INTEGER (0..4294967295)
Opaque ::=   [APPLICATION 4]          -- arbitrary ASN.1 value,
        IMPLICIT OCTET STRING
 END
```

Example

sysDescr **OBJECT-TYPE**

SYNTAX DisplayString (SIZE (0..255))

ACCESS read-only

STATUS mandatory

DESCRIPTION"A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters."

::= { system 1 }

Example

sysObjectID **OBJECT-TYPE**

SYNTAX **OBJECT IDENTIFIER**

ACCESS read-only

STATUS mandatory

DESCRIPTION "The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining `what kind of box' is being managed. For example, if vendor `Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its `FredRouter'."

::= { system 2 }

MIB-II

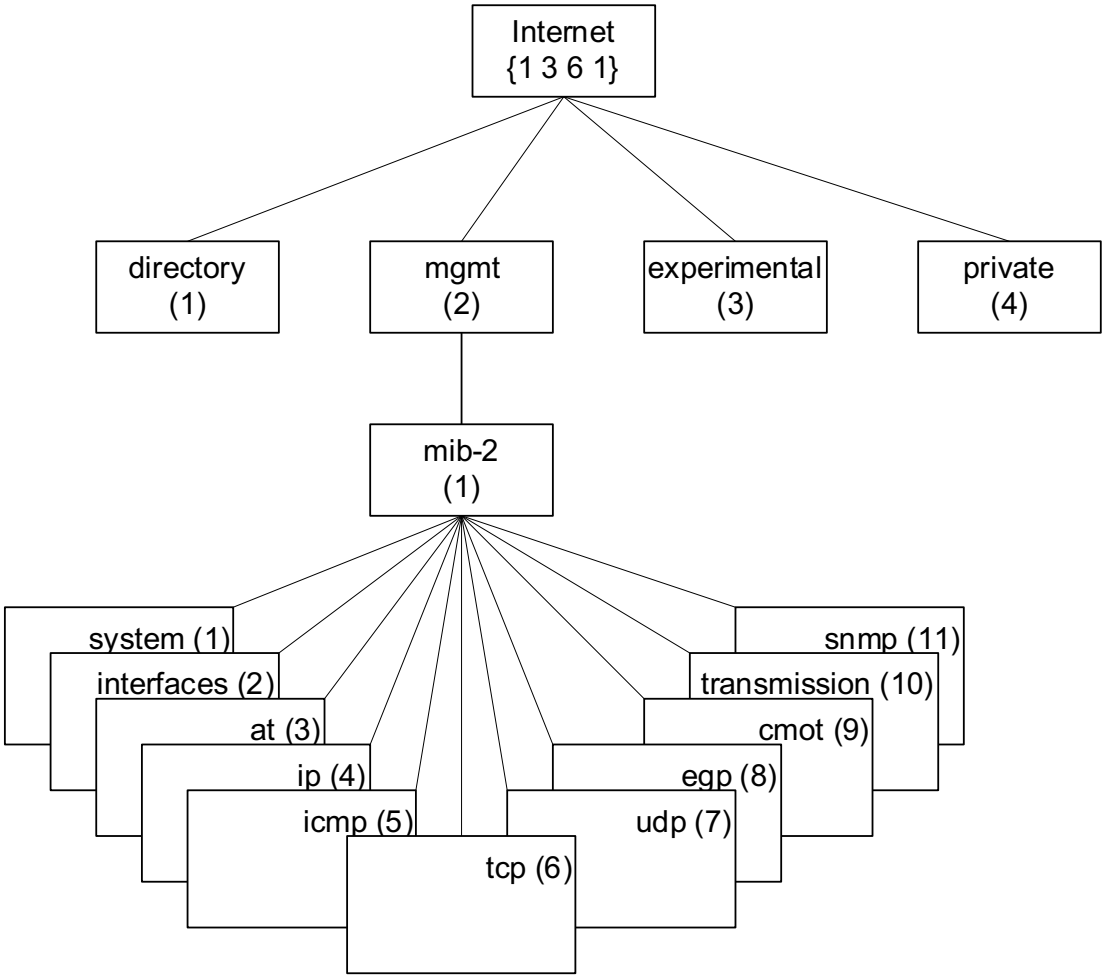


Figure 4.26 Internet MIB-II Group

- MIB-II (RFC 1213) is a superset of MIB-I
- Related objects grouped into object groups
- RFC 1213 defines eleven groups

MIB

```
RFC1213-MIB
        DEFINITIONS ::= BEGIN
         IMPORTS
                mgmt, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks
                        FROM RFC1155-SMI
                OBJECT-TYPE
                        FROM RFC-1212;
        --  MIB-II (same prefix as MIB-I)
        mib-2       OBJECT IDENTIFIER ::= { mgmt 1 }
        -- textual conventions
        DisplayString ::= OCTET STRING
        -- SIZE (0..255)
        PhysAddress ::= OCTET STRING
        -- an ethernet address would be represented as a string of 6 octets.
```

MIB

```
-- groups in MIB-II
        system          OBJECT IDENTIFIER ::= { mib-2 1 }
        interfaces      OBJECT IDENTIFIER ::= { mib-2 2 }
        at              OBJECT IDENTIFIER ::= { mib-2 3 }
        ip              OBJECT IDENTIFIER ::= { mib-2 4 }
        icmp            OBJECT IDENTIFIER ::= { mib-2 5 }
        tcp             OBJECT IDENTIFIER ::= { mib-2 6 }
        udp             OBJECT IDENTIFIER ::= { mib-2 7 }
        egp             OBJECT IDENTIFIER ::= { mib-2 8 }
        -- historical (some say hysterical)
        -- cmot         OBJECT IDENTIFIER ::= { mib-2 9 }
        transmission    OBJECT IDENTIFIER ::= { mib-2 10 }
        snmp            OBJECT IDENTIFIER ::= { mib-2 11 }
```

cmot - CMIP over TCP

System Group

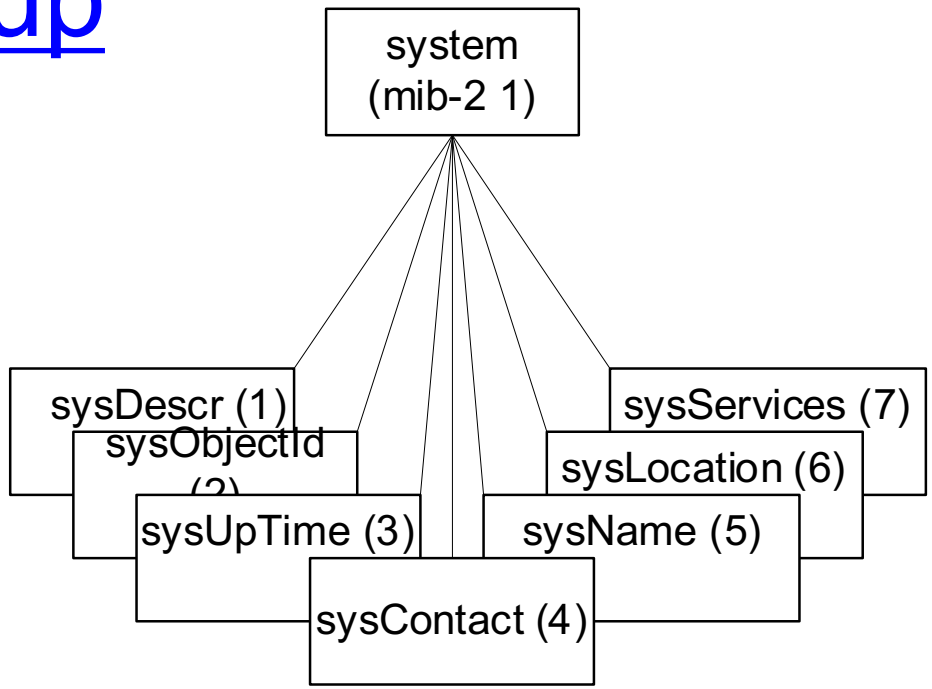


Figure 4.27 System Group

Entity	OID	Description (brief)
sysDescr	system 1	Textual description
sysObjectID	system 2	OBJECT IDENTIFIER of the entity
sysUpTime	system 3	Time (in hundredths of a second since last reset)
sysContact	system 4	Contact person for the node
sysName	system 5	Administrative name of the system
sysLocation	system 6	Physical location of the node
sysServices	system 7	Value designating the layer services provided by the entity

Object Instances

- MIB – generic objects
- Agents – object instances with values
- **Unique instance** (e.g. sysDescr, ifNumber)
  - oid of the instance is <id-leaf>.0

sysDescr ::= 1.3.6.1.2.1.1.1
→ instance sysDescr.0

Get(sysDescr.0) = Get (1.3.6.1.2.1.1.1.0)
→ {1.3.6.1.2.1.1.1.0, String SunOs4.1}

Get-Next(sysDescr) = Get-Next (1.3.6.1.2.1.1.1)
→ {1.3.6.1.2.1.1.1.0, String SunOs4.1}

But

Get-Next(sysDescr.0) → {sysObjectID.0, OID: enterprise.Sun.2.1.1}

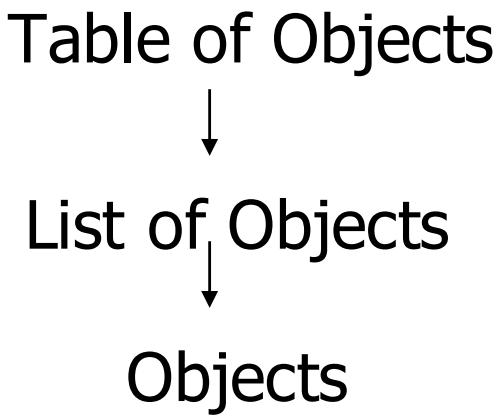
Get(sysDescr) = Get (1.3.6.1.2.1.1.1) → error "noSuchName"

Object Instances

- **Tabular Objects** (e.g. ipAddrTable)
  - type is a SEQUENCE OF xx and xx an OBJECT-TYPE with a field INDEX
  - different instances determined by the value of the *index*,
  - OID of the instance is OID(xx).index
  - instances grouped in a table
- e.g. Table of IP interfaces
  - ifInOctets.2 → Integer 102338
- Tabular objects
  - form a SEQUENCE OF rows (i.e. C array)
- Rows composed of various elements
  - row is a SEQUENCE of elements (i.e. C record)
  - one or several elements of the SEQUENCE has a INDEX field

Aggregate object

- A group of objects
- Also called tabular objects
- Can be represented by a table with
  - Columns of objects
  - Rows of instances



Tabular Representation

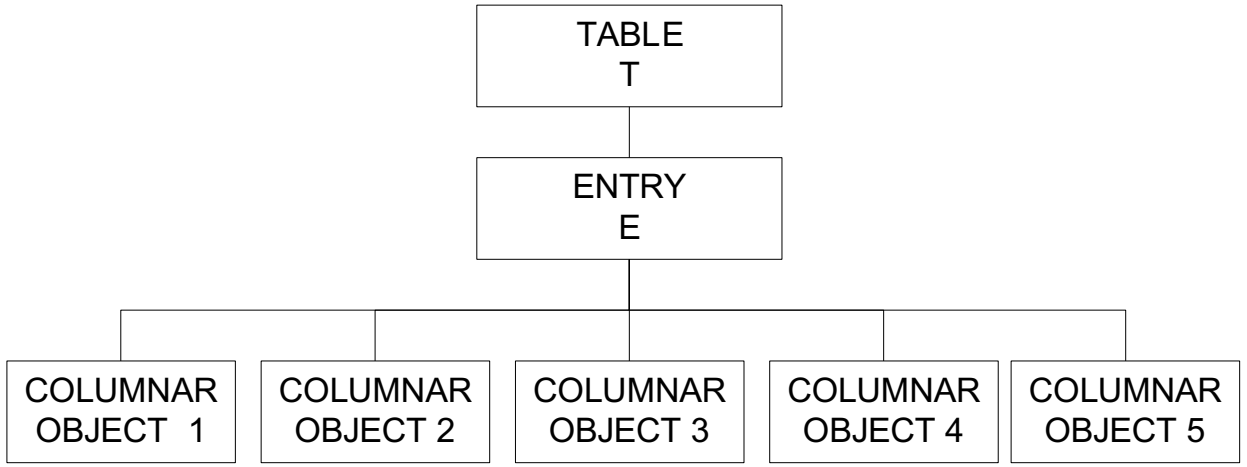


Figure 4.22(a) Multiple Instance Managed Object

- The objects TABLE T and ENTRY E are objects that are logical objects. They define the grouping and are not accessible
- Columnar objects are objects that represent the attributes and hence are accessible
- Each instance of E is a row of columnar objects 1 through 5
- Multiple instances of E are represented by multiple rows

Tabular Representation

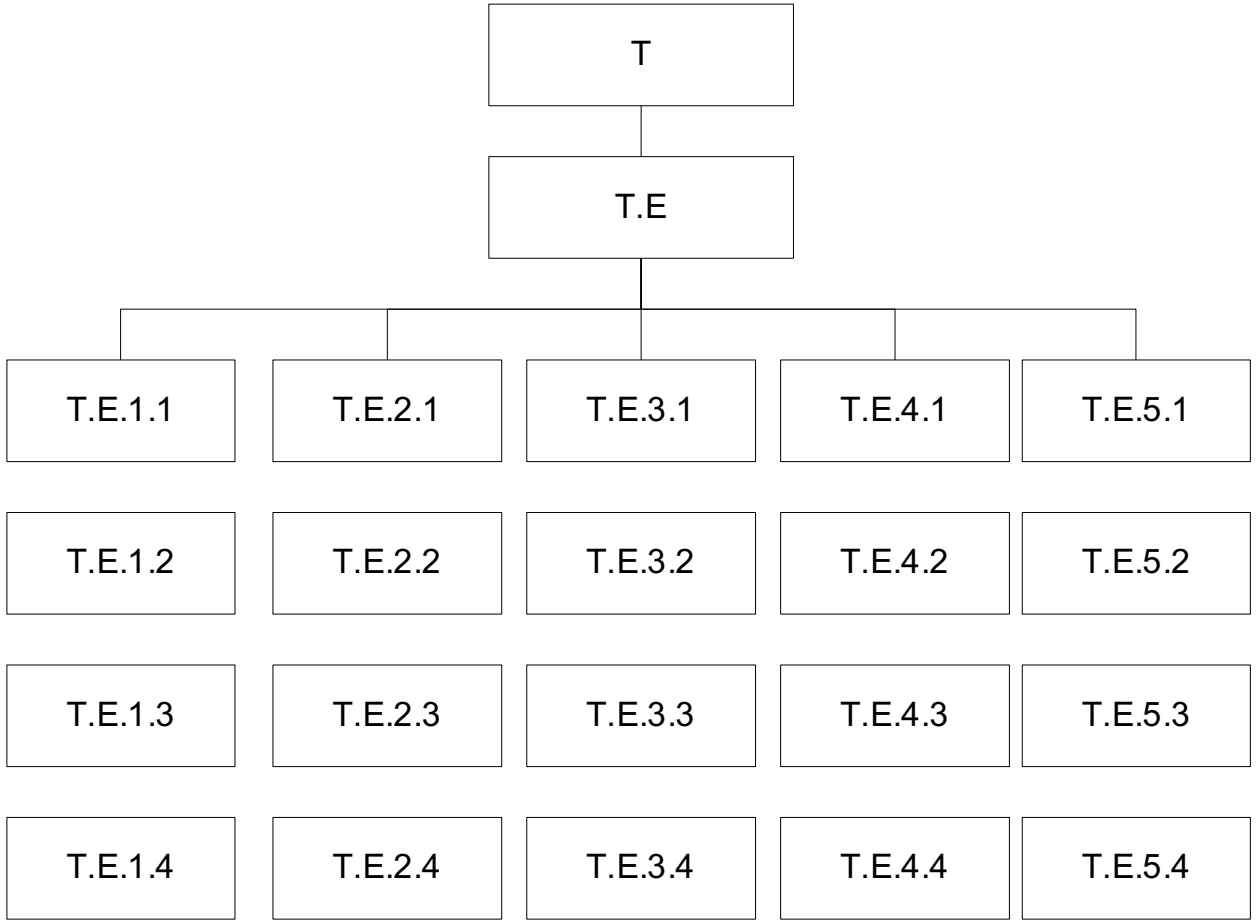


Figure 4.22(b) Example of 5 Columnar Object with 4 Instances (rows)

- Notice that the column-row numeric designation is reverse of what we are used to as row-column

INDEX

- Instanciated INDEX
  - concatenation of the transformation of the INDEX value into a sequence of labels

INDEX {ifIndex} and ifIndex INTEGER ::=3 → .3

INDEX { ipAdEntAddr } and ipAdEntAddr IpAddress::=127.0.0.1

→ .127.0.0.1

INDEX { name } and name OCTET STRING (0..20)::="lp2"

→ .3.108.112.50 (len=3, 'l'=108, 'p'=112, '2'=50)□□

INDEX { udpLocalAddress, udpLocalPort } and

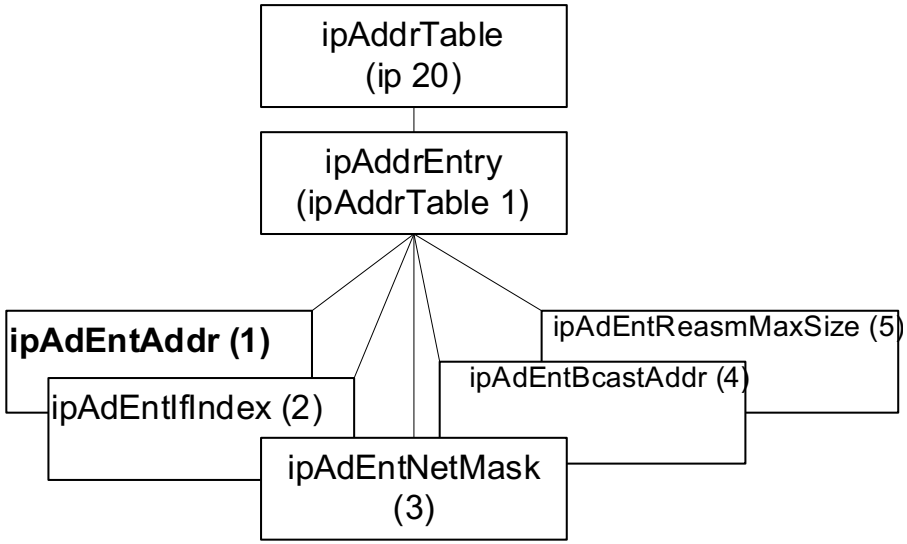
udpLocalAddress IpAddress::=127.0.0.1,

udpLocalAddress INTEGER (0..65535) = 2000 → .127.0.0.1.2000

Example

```
ipAddrTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IpAddrEntry ::= { ip 20 }
ipAddrEntry OBJECT-TYPE
    SYNTAX  IpAddrEntry   INDEX   { ipAdEntAddr }  ::= { ipAddrTable 1 }
ipAddrEntry ::=
    SEQUENCE {ipAdEntAddr              IpAddress,
              ipAdEntIfIndex           INTEGER,
              ipAdEntNetMask           IpAddress,
              ipAdEntBcastAddr         INTEGER,
              ipAdEntReasmMaxSize      INTEGER (0..65535)  }
ipAdEntAddr     OBJECT-TYPE SYNTAX  IpAddress::= { ipAddrEntry 1 }
ipAdEntIfIndex  OBJECT-TYPE SYNTAX  INTEGER::= { ipAddrEntry 2 }
ipAdEntNetMask OBJECT-TYPE SYNTAX  IpAddress::= { ipAddrEntry 3 }
ipAdEntBcastAddr OBJECT-TYPE SYNTAX  INTEGER::= { ipAddrEntry 4 }
ipAdEntReasmMaxSize OBJECT-TYPE SYNTAX  INTEGER (0..65535)::=
                                              { ipAddrEntry 5 }
```

Example



Legend: INDEX in bold

Figure 4.30 IP Address Table

Entity	OID	Description (brief)
ipAddrTable	ip 20	Table of IP addresses
ipAddrEntry	IpAddrTable 1	One of the entries in the IP address table
ipAdEntAddr	IpAddrEntry 1	The IP address to which this entry's addressing information pertains
ipAdEntIfIndex	IpAddrEntry 2	Index value of the entry, same as ifIndex
ipAdEntNetMask	IpAddrEntry 3	Subnet mask for the IP address of the entry
ipAdEntBcastAddr	IpAddrEntry 4	Broadcast address indicator bit
ipAdEntReasmMaxSize	IpAddrEntry 5	Largest IP datagram that can be reassembled on this interface

ipAddrTable

Row	ipAdEntAddr	ipAdEntIf-Index	ipAdEntNet-Mask	ipAdEntBcast-Addr	ipAdEntReasm-MaxSize
1	127.0.0.1	1	255.0.0.0	1	1536
2	129.88.34.1	2	255.255.255.0	1	1500

- Columnar object ipAdEntAddr
  - 1.3.6.1.2.1.4.20.1.1
  - row 2, OID 1.3.6.1.2.1.4.20.1.1.129.88.34.1
- Columnar object ipAdEntBcastAddr
  - 1.3.6.1.2.1.4.20.1.4
  - row 1, OID 1.3.6.1.2.1.4.20.1.4.127.0.0.1

Example

- 2 interfaces, 127.0.0.1 and 129.88.34.1 → snmpwalk in the tree

ip.ipAddrTable.ipAddrEntry.ipAdEntAddr.127.0.0.1 → ipAddress 127.0.0.1
ip.ipAddrTable.ipAddrEntry.ipAdEntAddr.129.88.34.1 → ipAddress 129.88.34.1
ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex.127.0.0.1 → Integer 1
ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex.129.88.34.1 → Integer 2
ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.127.0.0.1→ ipAddress 255.0.0.0
ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.129.88.34.1→ ipAddress 255.255.255.0
ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr.127.0.0.1 → Integer 1
ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr.129.88.34.1 → Integer 1
ip.ipAddrTable.ipAddrEntry.ipAdEntReasmMaxSize.127.0.0.1 → Integer 1536
ip.ipAddrTable.ipAddrEntry.ipAdEntReasmMaxSize.129.88.34.1→Integer 1500

Example

- 2 interfaces, 127.0.0.1 and 129.88.34.1
- Going through a table (all values in two columns)
  - from a parent, Get-Next give the 1st element (twice Get-Next)

Get-Next(ipAdEntAddr, ipAdEntNetMask) ->

{ {ipAdEntAddr.127.0.0.1, ipAddress 127.0.0.1}
{ipAdEntNetMask.127.0.0.1, ipAddress 255.0.0.0} }

  - from an element, Get-Next give the next element

Get-Next(ipAdEntAddr.127.0.0.1, ipAdEntNetMask.127.0.0.1) →

{ {ipAdEntAddr. 129.88.34.1, ipAddress 129.88.34.1}
{ipAdEntNetMask. 129.88.34.1, ipAddress 255.255.255.0} }

  - done, when not in the subtree

Get-Next(ipAdEntAddr.129.88.34.1, ipAdEntNetMask.129.88.34.1) →

{ { ipAdEntIfIndex.127.0.0.1, Integer 1}
{ ipAdEntBcastAddr.127.0.0.1, Integer 1} }

  - end – error EndOfMib or response change prefix

MIB for Get-Next-Request

- A
- B
- T
- E
- 1.1
- 1.2
- 2.1
- 2.2
- 3.1
- 3.2
- Z

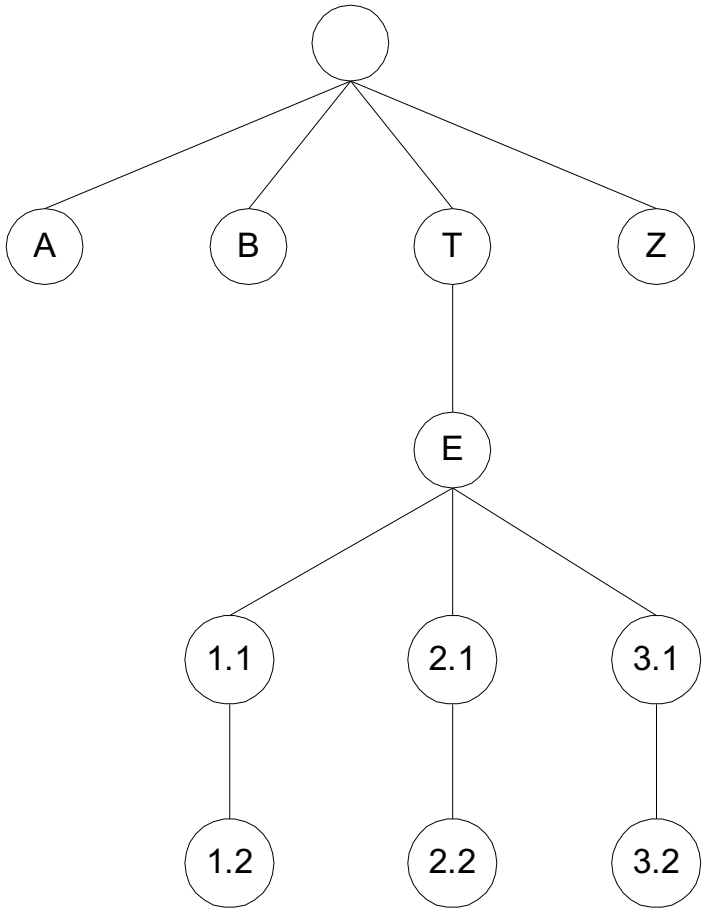


Figure 5.12 MIB for Operation Sequences in Figures 5.13 and 5.15

Get-Next-Request operation

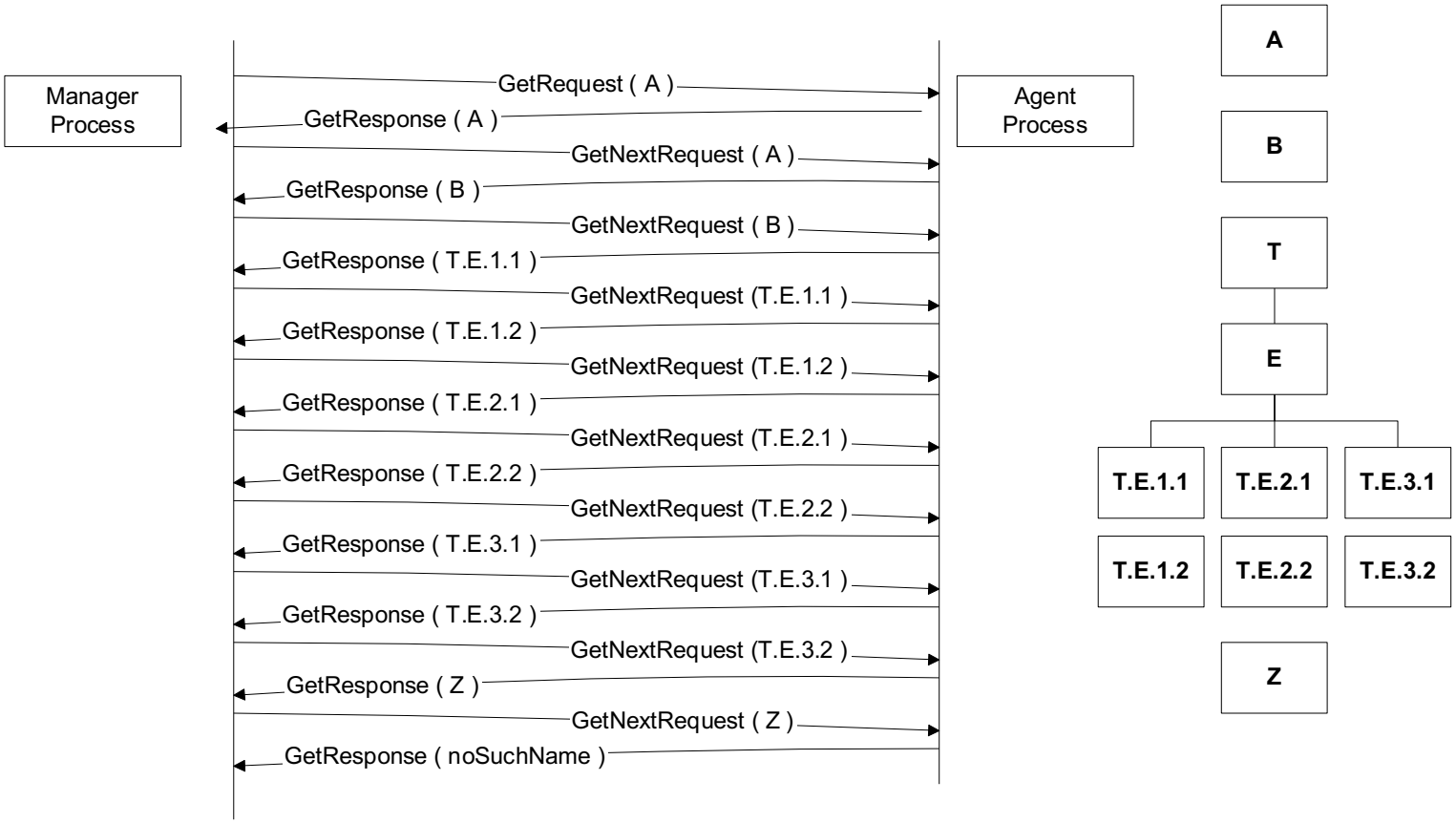


Figure 5.15 Get-Next-Request Operation for MIB in Figure 5.12

Get-Next-Request operation

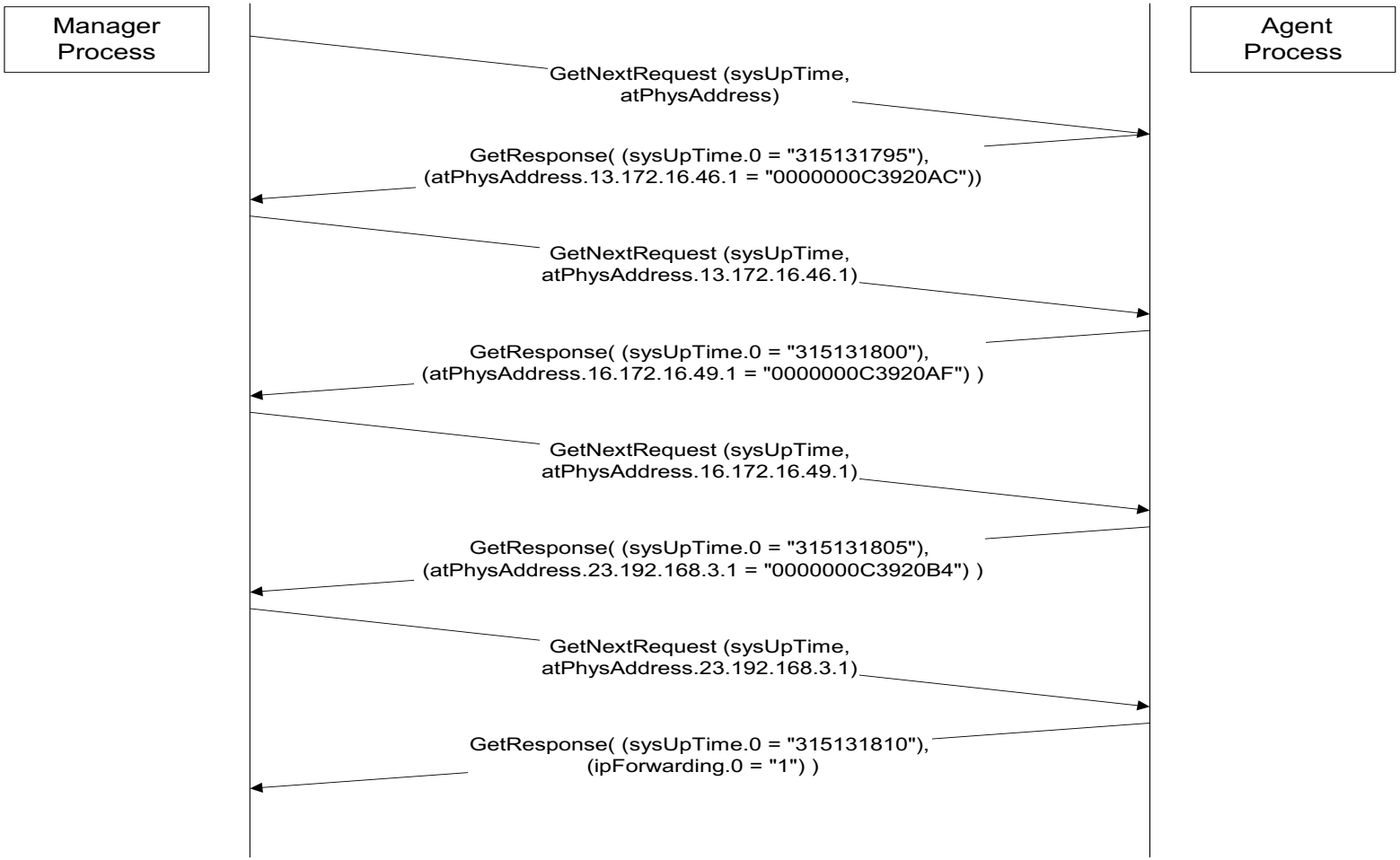


Figure 5.16 GetNextRequest Example with Indices

atIfIndex	atPhysAddress	**atNetAddress**
23	0000000C3920B4	**192.168.3.1**
13	0000000C3920AC	**172.16.46.1**
16	0000000C3920AF	**172.16.49.1**

SNMP operation

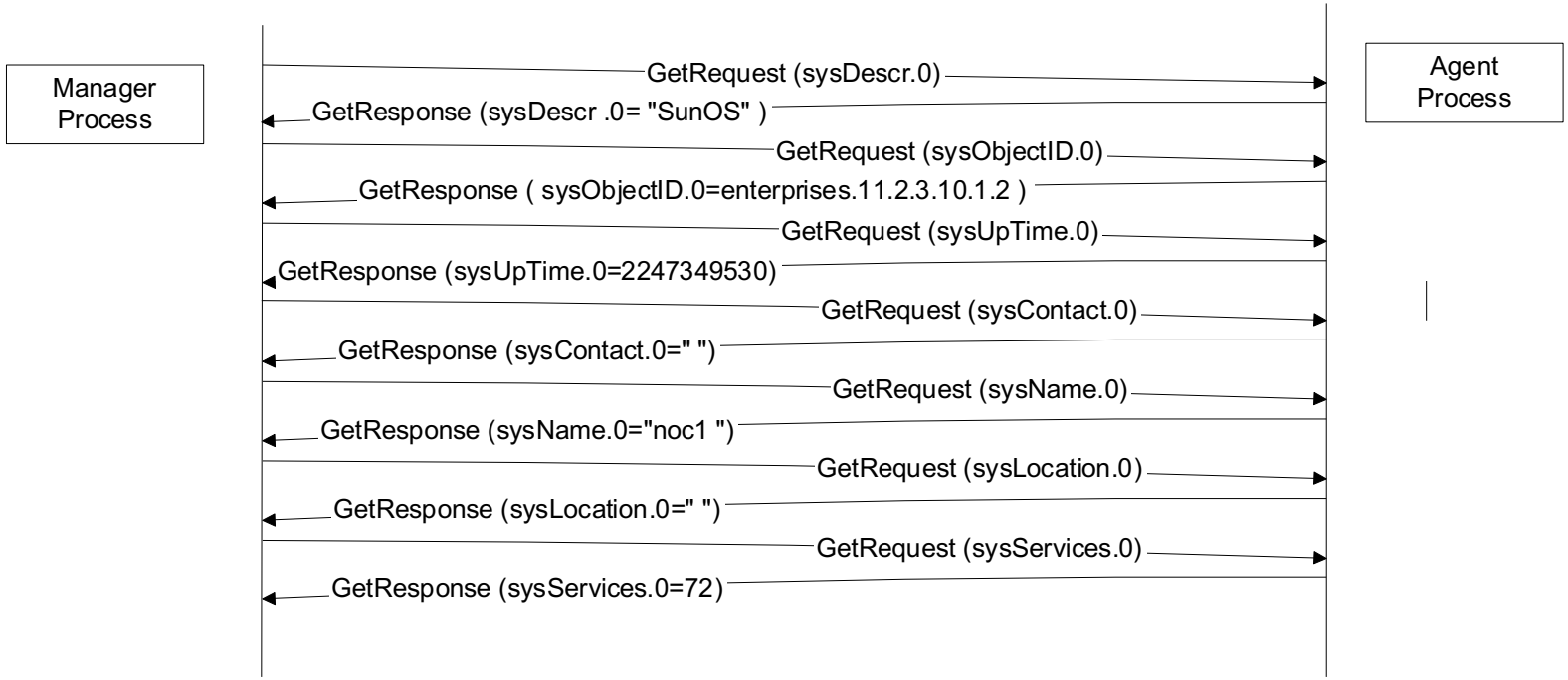


Figure 5.10 Get-Request Operation for System Group

SNMPv1

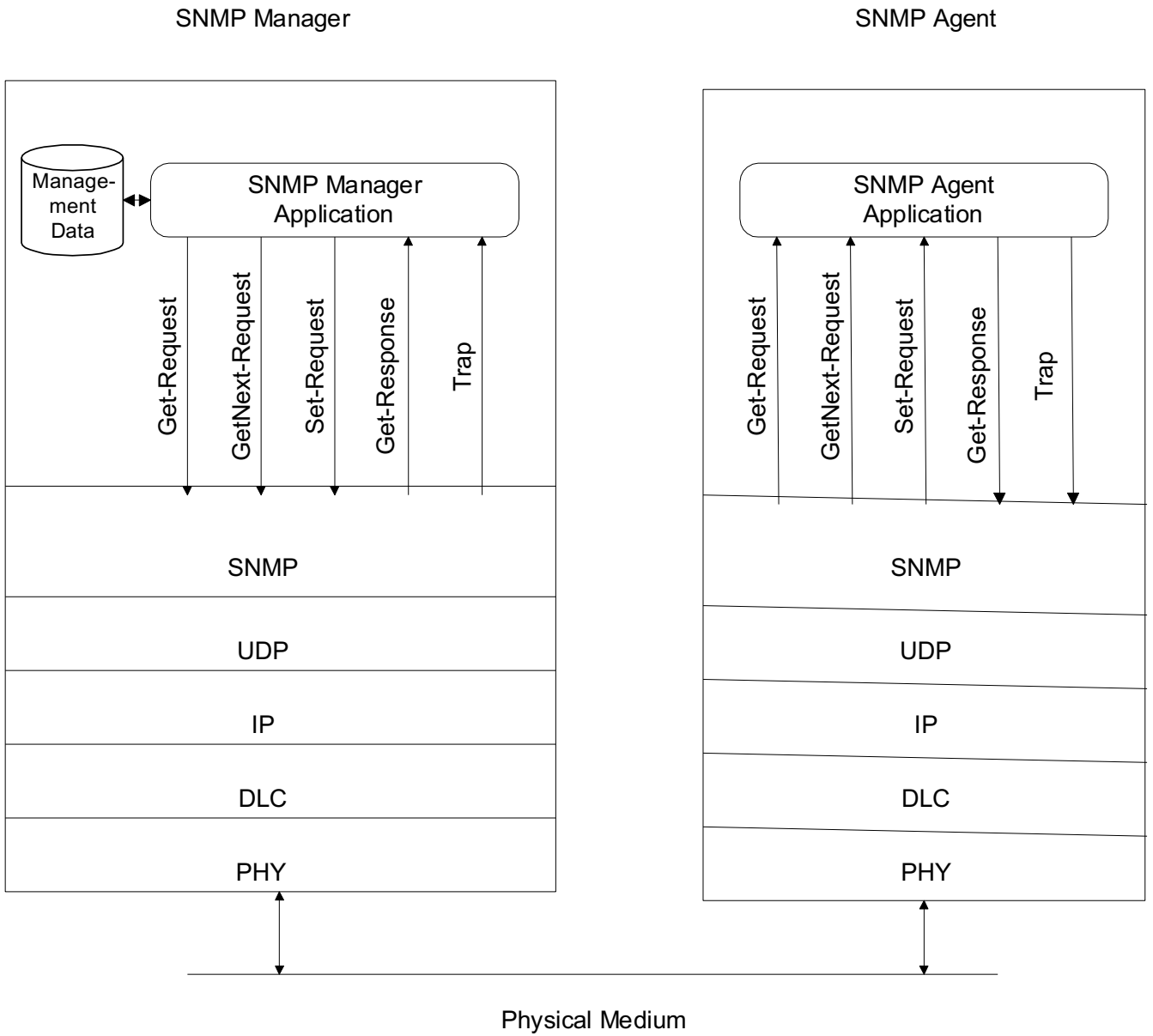


Figure 4.9 SNMP Network Management Architecture

SNMP Agent

- Operations
  - Get, Get-Next, Set, Trap
- Get
  - client sends Get with 1 or several couples **`(name, nul)`**, the value fields set to 'nul', agent sends Response with the same fields and the values
- Get-Next – get the "next" information
  - same format as Get
  - agent sends Response with couple **`(name, nul)`** replaced with couple **`[OID(next(name)), value(next(name)]`**
  - "next" – in the sense of going through the MIB tree
- Set
  - client sends Set with couple **`(name, value)`**, agent sends Response with the same fields and the values
- UDP port 161 for Get, 162 for Trap
- Connection-less
  - retry when a request is lost

SNMP Messages

- Trap
  - Generic trap, Specific trap
  - Time stamp
- Generic trap
  - coldStart, warmStart
  - linkDown, linkUp
  - authenticationfailure
  - egpNeighborLoss
  - enterpriseSpecific
- Specific trap
  - for special measurements such as statistics
- Time stamp: Time since last initialization

SNMP Community

- Security in SNMPv1 is community-based
- Authentication scheme in manager and agent
- Community: password, String of octets
- Community associated with rights on variables at agent
  - "*Read-Only*", "*Read-Write*", "*Write-Only*", "*Not-accessible*"
- Communication is not secured in SNMPv1 - no encryption

SNMP PDUs

type, id, communauté, code d'erreur, (variable, valeur)*

(type = Get, Get-Next, Set, Get-Response)

```
Message ::= { version version-1(0),
        community "public",                          <--- pour la sécurité
        data {
            getResponse {request-id 17,              <-- associe
requête/réponse

            error-status noError(0),      <-- présent dans requête
            error-index 0,                              avec valeur 0
            variable-binding {            <-- liste de couples
              {name 1.3.6.1.2.1.1.1.0,
                value {                   <-- pour get, valeur vide
                    simple {              <-- valeur typée
                            string "unix"
```

SNMP PDUs

```
% snmpget -v 1 -c public horus system.sysUpTime.0
system.sysUpTime.0 = TimeTicks: 132320314 (15 jours 7:33:22)
brahma.imag.fr -> horus.imag.fr UDP D=161 S=35874 LEN=57
 0: 0090 9278 c400 0800 2086 15d0 0800 4500  ...x.... .....E.
16: 004d f49d 4000 ff11 4046 8158 1e0a 8158  .Mô.@...@F.X...X
32: 2601 8c22 00a1 0039 8ab9 3082 002d 0201  &.."...9..0..-..
48: 0004 0670 7562 6c69 63a0 2002 0471 f39f  ...public. ..q..
64: d602 0100 0201 0030 8200 1030 8200 0c06  .......0...0....
80: 082b 0601 0201 0103 0005 00               .+.........
horus.imag.fr -> brahma.imag.fr UDP D=35874 S=161 LEN=55
 0: 0800 2086 15d0 0090 9278 c400 0800 4500  .. ......x....E.
16: 004b a114 4000 fe11 94d1 8158 2601 8158  .K..@......X&..X
32: 1e0a 00a1 8c22 0037 ced9 302d 0201 0004  .....".7..0-....
48: 0670 7562 6c69 63a2 2002 0471 f39f d602  .public. ..q....
64: 0100 0201 0030 1230 1006 082b 0601 0201  .....0.0...+....
80: 0103 0043 0407 e31e ffff ffd3            ...C.....
```

SNMPv2

- New PDUs:
  - inform-request (from manager to manager)
  - get-bulk-request (Get-Next with repetition)
  - trap replaced with notification
- TCP possible
- Security
  - transport with autentication and encryption
- Protocol not widely deployed

SNMPv3

- Architecture model
  - each entity identified by "snmpEngineID"
  - entity composed of components that interact
- Security model
  - 'User based' – rights associated with the User
  - Communication signed (HMAC), encrypted (CBC-DES), timestamped (except for 'denial of service')
  - Agent has a local base that defines Access Rights – VACM model (VACM - View Based Access Control Model)
  - VACM defines the level of security for the User, its execution context, possible views on MIB, and authorization strategies

Facts to remember

- SNMP largely supported by network devices
- Complex use: the user needs to define what to observe
- Security hinders deployment
- Products
  - Nagios – scripts
  - IBM Tivoli
  - BMC Remedy
  - HP Service Manager

Example

```
ipAddrTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IpAddrEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The table of addressing
            information relevant to this entity's IP
            addresses."
    ::= {ip 20}
```

```
ipAddrTable      OBJECT-TYPE ::= {ip 20}
ipAddrEntry      OBJECT-TYPE ::= {ipAddrTable 1}
```

Example

```
ipAddrEntry OBJECT-TYPE
    SYNTAX  IpAddrEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "The addressing information for one of this
entity's IP addresses."

    INDEX   { ipAdEntAddr }
    ::= { ipAddrTable 1 }

IpAddrEntry ::=
    SEQUENCE {
        ipAdEntAddr
            IpAddress,
        ipAdEntIfIndex
            INTEGER,
        ipAdEntNetMask
            IpAddress,
        ipAdEntBcastAddr
            INTEGER,
        ipAdEntReasmMaxSize
            INTEGER (0..65535)
```

Index *ipAdEntAddr* uniquely identifies an instance

Example

ipAddrTable {1.3.6.1.2.1.4.20}
ipAddrEntry (1)
ipAdEntAddr (1)
ipAdEntIfIndex (2)
ipAdEntNetMask (3)
ipAdEntBcastAddr (4)
ipAdEntReasmMaxSize (5)

Columnar object ID of ipAdEntBcastAddr is (1.3.6.1.2.1.4.20.1.4):

iso org dod internet mgmt mib ip ipAddrTable ipAddrEntry ipAdEntBcastAddr
1 3 6 1 2 1 4 20 1 4

Figure 4.23(a) Columnar objects under ipAddrEntry

Row	ipAdEntAddr	ipAdEntIfIndex	IpAdEntNetMask	IpAdEntBcastAddr	IpAdEntReasmMaxSize
1	**123.45.2.1**	1	255.255.255.0	0	12000
2	**123.45.3.4**	3	255.255.0.0	1	12000
3	**165.8.9.25**	2	255.255.255.0	0	10000
4	**9.96.8.138**	4	255.255.255.0	0	15000

Figure 4.23(b) Object instances of ipAddrTable (1.3.6.1.2.1.4.20)

Columnar Object	Row # in (b)	Object Identifier
ipAdEntAddr 1.3.6.1.2.1.4.20.1.1	2	{1.3.6.1.2.1.4.20.1.1.123.45.3.4}
ipAdEntIfIndex 1.3.6.1.2.1.4.20.1.2	3	{1.3.6.1.2.1.4.20.1.2.165.8.9.25}
ipAdEntBcastAddr 1.3.6.1.2.1.4.20.1.4	1	{1.3.6.1.2.1.4.20.1.4.123.45.2.1}
IpAdEntReasmMaxSize 1.3.6.1.2.1.4.20.1.5	4	{1.3.6.1.2.1.4.20.1.5.9.96.8.138}

Figure 4.23(c) Object Id for specific instance

Lexicographic Order

- Start with leftmost digit as first position
- Before increasing the order in the first position, select the lowest digit in the second position
- Continue the process till the lowest digit in the last position is captured
- Increase the order in the last position until all the digits in the last position are captured
- Move back to the last but one position and repeat the process
- Continue advancing to the first position until all the numbers are ordered
- Tree structure for the above process

Numerical Order	Lexicographic order
1	1
2	1118
3	115
9	126
15	15
22	2
34	22
115	250
126	2509
250	3
321	321
1118	34
2509	9