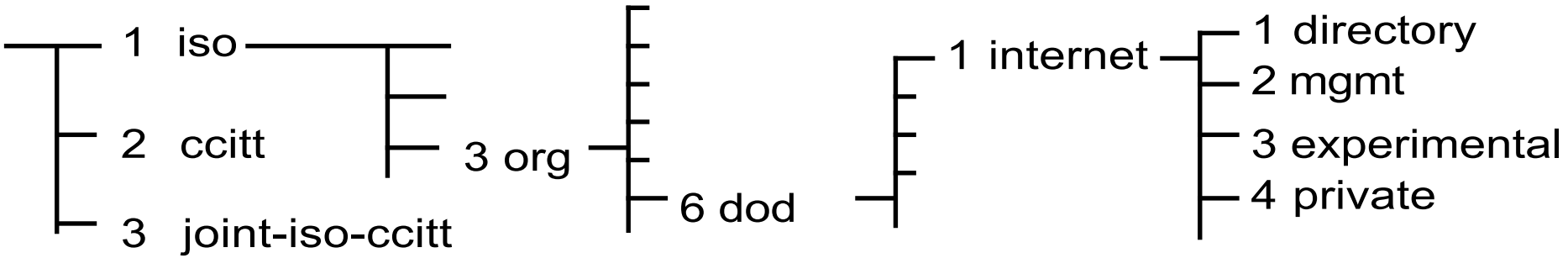
Management Information Base

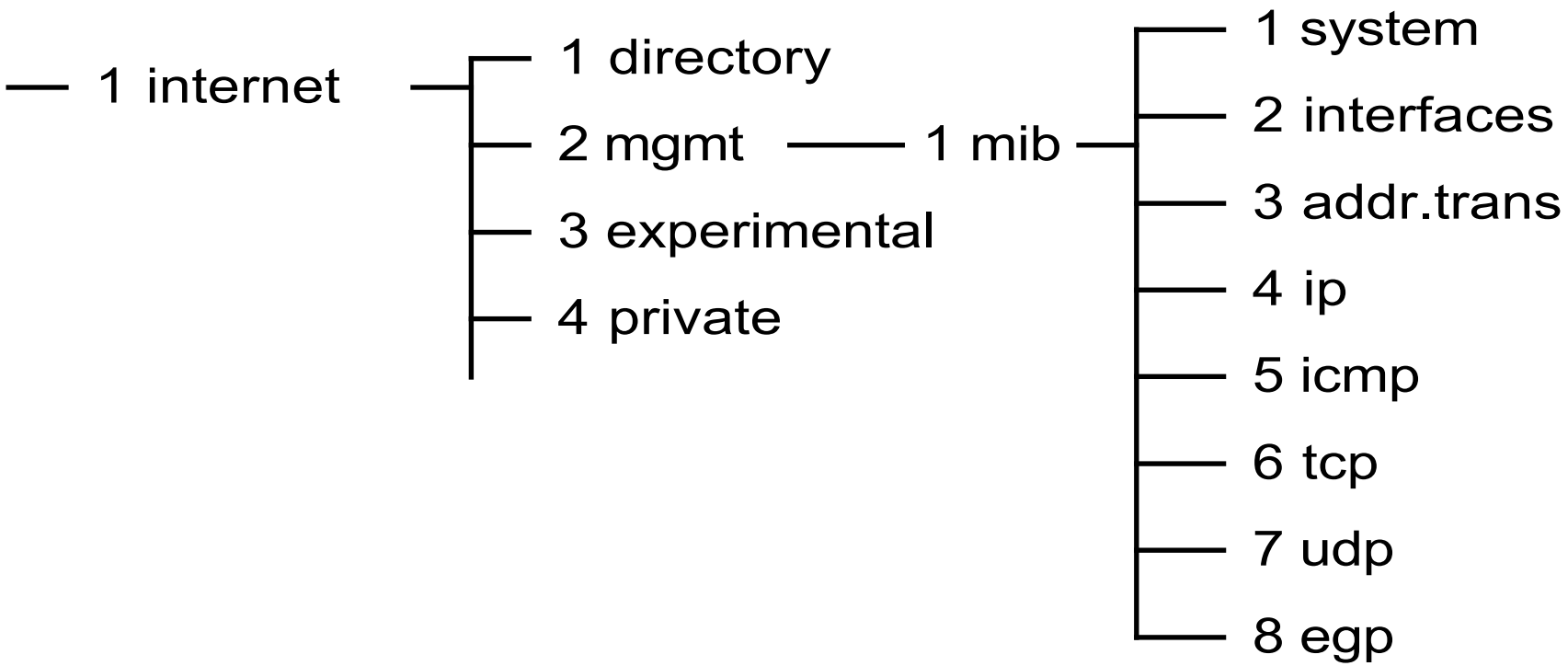
- Management of a network involves reading and setting many network-related values, as we evaluate the network performance and adjust its operation.
- The management values are held in an extensive database using a structure agreed between Internet and OSI network management, known as the *Management Information Base* or *MIB*.
- The MIB is a hierarchical structure, with a name identified by the sequence of node indices as the tree is traversed from the root to the node.



All internet Management variables start with
1.3.6.1.2. … **or** *iso.org.dod.internet.mgmt.*

First levels of Management Information Base tree

- There are three basic trees, from an unnamed root node.
- The one of interest is "internet", which is managed by "dod" (the Department of Defense), which is an "org" (Organisation) known to the International Standards Organisation "iso".



All tcp variables are *1.3.6.1.2.1.6. …*
or *iso.org.dod.internet.mgmt.mib.tcp. …*

Namespace for variables within Internet MIB

- All MIB variables fall into one of eight categories, corresponding to the rightmost column of the tree (system to egp).
- Simple (scalar) variables are defined by the list of names or indices, followed by a zero.
- Thus the MIB variable *ipInReceives* (identifier 3 under the *ip* node) is either *1.3.6.1.2.1.4.3.0* or *iso.org.dod.internet.mgmt.mib.ip.ipLnReceives*. (It counts the number of IP packets received.)
- The mappings are defined entirely by tables – there is no other defined correspondence.
- The internal storage format is unspecified, but it must be accessible by addresses as specified.

Abstract Syntax Notation.1 (ASN.1) and MIB definition

The MIB entries are defined by a special syntax. For example, the variable *ipInReceives* has the definition

```
ipInReceives OBJECT-TYPE
                    SYNTAX   Counter
                    ACCESS   read-only
                    STATUS   mandatory
                    ::= { ip 3 }
```

It is a read-only counter, which is must be implemented and has an index of 3 within *ip*.

Entries which define the lower nodes of the tree are, in part,

```
mgmt OBJECT IDENTIFIER ::= { iso org(3) dod(6)
                             internet(1) mgmt(2) }
     mib   OBJECT IDENTIFIER ::= { mgmt 1 }
     ip    OBJECT IDENTIFIER ::= { mib 4 }
```

Other examples come from the *ip* section and the definition of the *ipAddrTable*.

- SEQUENCE OF in *ipAddrTable* defines a linear array of entries
- SEQUENCE defines a data structure (such as *ipAddrEntry*).

```
-- the IP Interface table

        ipAddrTable OBJECT-TYPE
                SYNTAX  SEQUENCE OF IpAddrEntry
                ACCESS  read-only
                STATUS  mandatory
                ::= { ip 20 }

        ipAddrEntry OBJECT-TYPE
                SYNTAX  IpAddrEntry
                ACCESS  read-only
                STATUS  mandatory
                ::= { ipAddrTable 1 }
```

```
IpAddrEntry ::= SEQUENCE {
    ipAdEntAddr
        IpAddress,
    ipAdEntIfIndex
        INTEGER,
    ipAdEntNetMask
        IpAddress,
    ipAdEntBcastAddr
        INTEGER
}
```

Each of the elements of the structure is then defined –

```
ipAdEntAddr OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::=  { ipAddrEntry 1 }

ipAdEntIfIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::=  { ipAddrEntry 2 }

ipAdEntNetMask OBJECT-TYPE
        SYNTAX  IpAddress
        ACCESS  read-only
        STATUS  mandatory
        ::=  { ipAddrEntry 3 }
```

```
ipAdEntBcastAddr OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        ::= { ipAddrEntry 4 }
```

Accessing an element of a structure is rather unusual, in that a suffix is appended to the name. For the *ipAddrTable*, the suffix is the IP address. Thus to find the network mask field in the table entry for address 128.10.2.3, the full name is

iso.org.dod.internet.mgmt.mib.ip.
ipAddrTable.ipAddrEntry.ipAdEntNetMask.128.10.2.3

or, in numeric form,

1.3.6.1.2.1.4. 20.1.3.128.10.2.3.

Stepping to the next entry, when neither its index nor the total number of elements is known, is a function of the network management protocol.

Simple Network Management Protocol (SNMP)

- The SNMP protocol provides the user interface to the MIB and is in many respects an extension of the MIB.
- It is based on a request/response, or fetch/store paradigm, in which the manager may request values for variables or supply values; actions may occur as a side-effect of writing values.

Command	Meaning
get-request	Fetch a value from a specific variable
get-next-request	Fetch the value following that fetched by last get-request
set-request	Store a value in a specific variable
get-response	Reply to a get operation
trap	Reply triggered by an event

- SNMP operations are atomic, meaning that all of the actions of a message must occur, or none will occur; any error causes the whole request to be abandoned.
- The problem of scanning through tables is solved by the *get-next-request* which has an identifier for a known item in a table, but triggers a *get-response* corresponding to the next table entry. A *get-next-request* to the table itself returns the first entry.
- SNMP is often implemented as a datagram protocol, using UDP, with no explicit acknowledgement of commands.
  ° Each command is given a 32-bit sequence number which is returned in replies.
  ° Replies can be associated with requests, or repeated commands ignored.

Details from here on are not examinable, but principles should be known

- The GetRequest-PDU for example contains a 32-bit *requestID* which is essentially a sequence number of the request to match requests and responses, error indicators and a list of objects for which values are wanted.
- The messages tend to be complex. The next slide contains the lines —

```
   A0      1C      02      04      05      AE      56      02
getreq. len=28 INTEGER len=4   |------- request id --------|
```

A0	identifies a Get-Request command
1C	length of the command
02	an integer follows (known to be the request ID)
04	the length of the integer
05 AE 56 02	the 4 bytes of the integer value

```
   30        29        02        01        00
SEQUENCE  len=41   INTEGER   len=1    vers=0

   04        06        70        75        62        6C        69        63
 string    len=6       p         u         b         l         i         c

   A0        1C        02        04        05        AE        56        02
 getreq.   len=28   INTEGER   len=4   |------- request id -------|

   02        01        00        02        01        00
 INTEGER   len=1       00    INTEGER   len=1 err. index

   30        0E        30        0C        06        08
SEQUENCE  len=14  SEQUENCE  len=12  objectid  len=8

   2B        06        01        02        01        01        01        00
  1.3         6         1         2         1         1         1         0

   05        00
 null      len=0
```

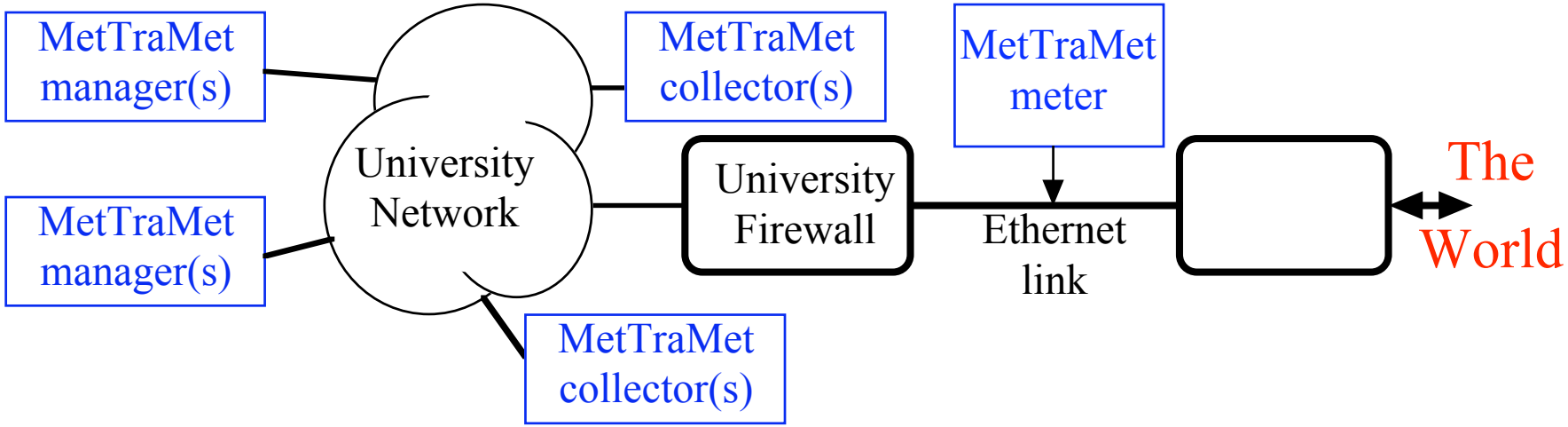
Example of SNMP message – *get-request* for *sysDescr* (*1.3.6.1.2.1.1.1*)

Fortunately, SNMP is supported by a suite of public-domain routines which do most of the message formatting and interpretation for the user, so that even people working at the detailed level seldom need to know all the gory details.

The NetTraMet system (Network Traffic Meter)

A system of

- meters, to collect measurements on the attached networks
- collectors, to collect data from meters (meters & collectors may be many to many), and
- managers, to coordinate and process data from collectors
- communication is by SNMP.



Measuring University external traffic