

# Reti di Calcolatori II

## SNMP (parte II)

Giorgio Ventre  
COMICS LAB  
Dipartimento di Informatica e Sistemistica  
Università di Napoli Federico II

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Nota di Copyright

Quest'insieme di trasparenze è stato ideato e realizzato dai ricercatori del Gruppo di Ricerca sull'Informatica Distribuita del Dipartimento di Informatica e Sistemistica dell'Università di Napoli e del Laboratorio Nazionale per la Informatica e la Telematica Multimediali. Esse possono essere impiegate liberamente per fini didattici esclusivamente senza fini di lucro, a meno di un esplicito consenso scritto degli Autori. Nell'uso dovrà essere esplicitamente riportata la fonte e gli Autori. Gli Autori non sono responsabili per eventuali imprecisioni contenute in tali trasparenze né per eventuali problemi, danni o malfunzionamenti derivanti dal loro uso o applicazione.

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

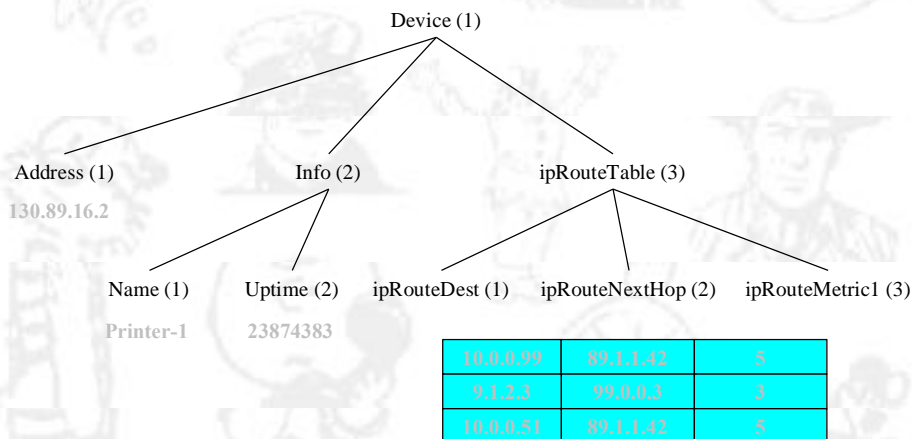
## Obiettivi

- | Approfondimento comando get-next
- | Gestione in SNMP di informazioni tabellari:
  - » scansione di tabelle
  - » inserimento/rimozione
  - » indici complessi
- | Tipi complessi di variabile
- | Dettaglio MIB II

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB di Esempio



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Approfondimento get-next (1)

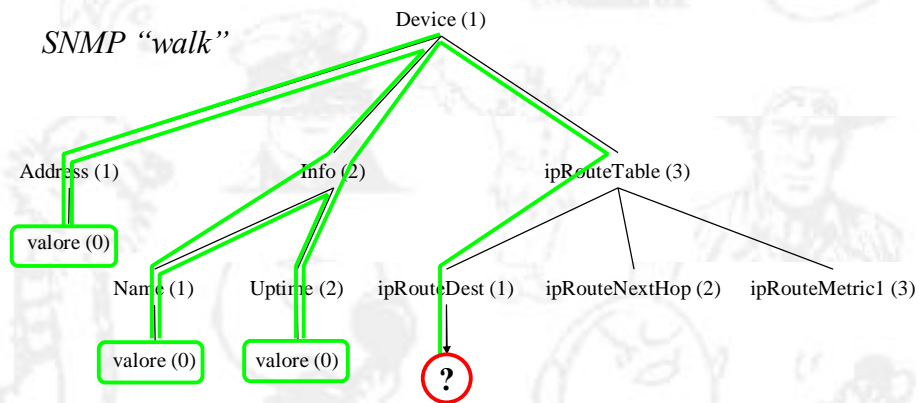
- | Un engine SNMP deve essere in grado di interpretare qualsiasi OID e stabilirne la posizione (ordine) rispetto alle OID implementate
- | Lo stesso algoritmo si applica indipendentemente su tutte le variabili della stessa richiesta
- | Esempio (caricaturale):
  - » `get-next(1, 1.1, 1.2, 1.2.2) ⇒`  
(1.1.0 → 130.89.16.2, 1.1.0 → 130.89.16.2, 1.2.1.0 → Printer-2, 1.2.2.0 → 23874383)
- | A che serve la `get-next`? Come usarla (bene)?

## Approfondimento get-next (2)

- | Per scandire una MIB intera:

```
oid = 1
while true:
    (oid, value) = getnext(oid)
    print oid + "=" + value;
```
- | La prossima OID utilizzata nella `get-next` è quella ritornata dal comando `get-next` appena realizzato
- | Una OID ritornata dalla `get-next` è sempre una OID foglia; la `get-next` su quella OID ci torna sempre la prossima foglia (nell'ordine lessicografico della MIB)

## Approfondimento get-next (3)



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Dati tabellari (1)

- | Permettono la rappresentazione di un vettore di oggetti strutturati
- | Navigabile con le stesse primitive delle variabili *scalari*
- | E' possibile (sempre con le primitive standard) aggiungere e rimuovere righe

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Dati tabellari (2)

### Definizione in ASN.1:

- » nodo "root" della tabella, una lista di "entries":

```
<table> OBJECT-TYPE SYNTAX SEQUENCE OF <Entry>
```

- » "interface entry", che riferisce all'indice:

```
<entry> OBJECT-TYPE SYNTAX <Entry>  
INDEX { <index> }  
 ::= { <table> 1 }
```

- » definizione struttura dei record:

```
<Entry> ::= SEQUENCE {  
<index> INTEGER,  
<field1> DisplayString, ... }
```

- » definizione dell'indice:

```
<index> ... ::= { <entry> 1 }
```

- » definizione di ogni campo della struttura (come nodo ASN.1)

## Esempio dalla MIB-II (1)

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

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

```
IfEntry ::=  
SEQUENCE {  
    ifIndex  
        INTEGER,  
    ifDescr  
        DisplayString,  
    ifType  
        INTEGER,  
    ifMtu  
        INTEGER,  
    ifSpeed  
        Gauge,  
    ifPhysAddress  
        PhysAddress,  
    ifAdminStatus  
        INTEGER,  
    ifOperStatus  
        INTEGER,  
    ... }
```

## Esempio dalla MIB-II (2)

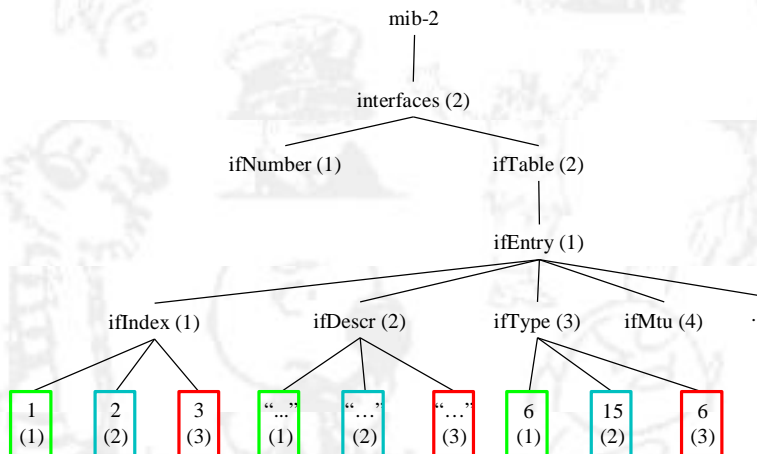
```
ifIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "...
        ::= { ifEntry 1 }
```

```
ifDescr OBJECT-TYPE
    SYNTAX DisplayString (SIZE(0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "...
        ::= { ifEntry 2 }
```

ecc. ecc...

Questa tabella contiene i dati delle interfacce presente sull'apparato.

## Esempio dalla MIB-II (3)



## Navigazione di una tabella (1)

- | OID id una "cella" nella tabella:

```
<oid tabella>.1.<colonna>.<riga>
```

- | Ad esempio:

```
1.3.6.1.2.1.2.2.1.2.3 → "..." (descrizione 3ª interfaccia)
```

```
mib-2.interfaces.ifTable.ifEntry.ifType.2 → 15
```

- | Per recuperare una riga intera con una sola get:

```
get(1.3.6.1.2.1.2.2.1.1.1,  
    1.3.6.1.2.1.2.2.1.2.1,  
    1.3.6.1.2.1.2.2.1.3.1, ...)
```

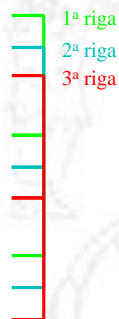
```
⇒ (1.3.6.1.2.1.2.2.1.1.1 → 1,  
    1.3.6.1.2.1.2.2.1.2.1 → "...",  
    1.3.6.1.2.1.2.2.1.3.1 → 6, ...)
```



## Navigazione di una tabella (2)

- | Se utilizzo la get-next, nell'ordine troverò:

```
ifTable.ifEntry.ifIndex.1  
ifTable.ifEntry.ifIndex.2  
ifTable.ifEntry.ifIndex.3  
...  
ifTable.ifEntry.ifDescr.1  
ifTable.ifEntry.ifDescr.2  
ifTable.ifEntry.ifDescr.3  
...  
ifTable.ifEntry.ifType.1  
ifTable.ifEntry.ifType.2  
ifTable.ifEntry.ifType.3  
...
```



## Navigazione di una tabella (3)

- Come si fa? Si utilizza la `get-next` su tutta la riga, e si recuperano le righe intere una dopo l'altra...

```
get-next( 1.3.6.1.2.1.2.2.1.1.0,  
          1.3.6.1.2.1.2.2.1.2.0,  
          1.3.6.1.2.1.2.2.1.3.0, ... )  
⇒ ( 1.3.6.1.2.1.2.2.1.1.1 → 1,  
     1.3.6.1.2.1.2.2.1.2.1 → "...",  
     1.3.6.1.2.1.2.2.1.3.1 → 6, ... )  
get-next( 1.3.6.1.2.1.2.2.1.1.1,  
          1.3.6.1.2.1.2.2.1.2.1,  
          1.3.6.1.2.1.2.2.1.3.1, ... )  
⇒ ( 1.3.6.1.2.1.2.2.1.1.2 → 2,  
     1.3.6.1.2.1.2.2.1.2.2 → "...",  
     1.3.6.1.2.1.2.2.1.3.2 → 15, ... )
```

e così via...

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Inserimento/Cancelazione righe

- Non è supportato nativamente dal protocollo: bisogna usare una combinazione di primitive e di variabili aggiuntive
- Non c'è una RFC che definisce un meccanismo standard
- Nella RFC 1271 (RMON), vi è descritto un meccanismo generico che può essere riutilizzato in altri contesti
- Si aggiunge una colonna in più, ad esempio "entryStatus", di tipo read-write, che può valere un intero tra:  
*valid(1), createRequest(2), underCreation(3), invalid(4)*
- Si usa la `SetRequest` per dare la possibilità al Manager di agire sulla riga

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Inserimento nuova riga

- | Il Manager deve trovare un valore di indice ancora non presente in tabella
- | Invia all'Agent una SetRequest con, per tutte o solo una parte delle colonne, e per quell'indice
- | Una SetRequest su un valore di indice che non esiste viene interpretato come una richiesta di creazione della riga con quell'indice; `entryStatus` viene posto al valore `createRequest(2)`
- | Per confermare la creazione della riga, una volta completati i dati, il Manager invia una SetRequest di `entryStatus` al valore `valid(1)`

## Cancellazione riga

- | Il Manager stabilisce l'indice della riga che deve essere cancellata
- | Invia all'Agent una SetRequest con la colonna `entryStatus` viene posta al valore `invalid(4)`
- | L'Agent può decidere di mantenere la riga nella tabella con lo status di `invalid`, oppure di fare sparire la riga dalla tabella

## Definizione di indici complessi

- | L'indice di una tabella di MIB deve essere una *chiave* per accedere all'informazione nella tabella
- | Non è per forza di tipo INTEGER
- | Non è sempre una sola colonna...

## Indice di tipo stringa

- | Ad esempio una tabella che rappresenti il file `/etc/hosts`: la chiave primaria è l'*hostname*
- | In questo caso l'indice è costituito da la sequenza `<lunghezza>.<car 1>.<car 2>... . <car n>`
- | Ad esempio:

```
hostTable.hostEntry.hostName.5.85.78.73.84.78
```

↑  
lunghezza

“UNITN” in ASCII

## Indice di tipo IP Address

- | Ad esempio la tabella ipForward
- | In questo caso l'indice è costituito da  
`<byte1>.<byte2>.<byte3>.<byte4>`
- | Ad esempio:  
`ipForward.1.193.17.25.230`

IP Address

## Indice multiplo

- | Ad esempio la tabella tcpConnection
- | In questo caso l'indice è costituito da:  
`<local IP Add.>.<local port>.<remote IP Add.>.<local port>`
- | Ad esempio:  
`tcpConnState.89.1.1.42.21.10.0.0.51.2059`

Local IP Address

Remote IP Address

Local Port

Remote Port

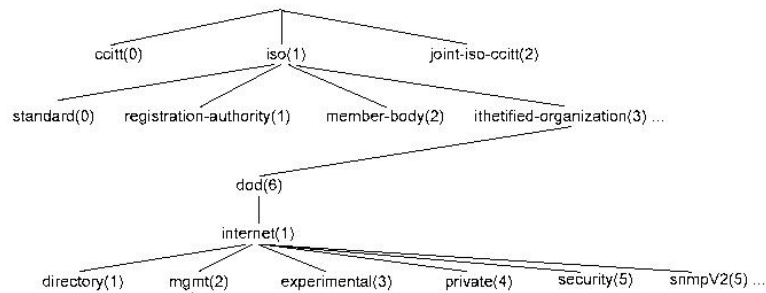
## MIB Naming Conventions

- | Prefisso comune a variabili dello stesso gruppo: sysDescr
- | Oggetti di tipo Counter al plurale
- | Nomi di tabelle nella forma xxxTable: ifTable
- | Nomi di riga nella forma xxxEntry: ifEntry
- | Prefisso comune a tutti i nodi di una tabella, comprese le foglie: e.g. per interfaces, ifIndex, ifType, ifState...

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Approfondimento: MIB-I I



MIB-II

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB-II : generalità

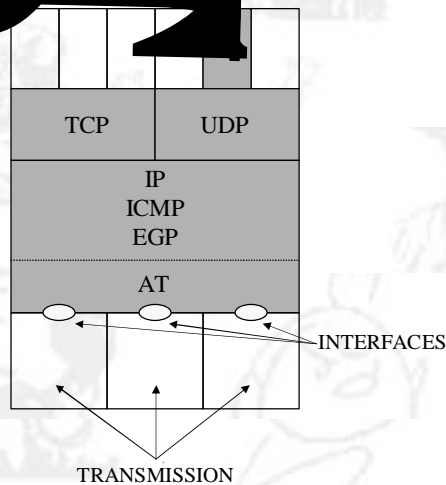
- | La RFC 1156 definì la prima versione di una struttura generica per apparati di rete TCP/IP
- | Nella RFC 1213 la MIB-I fu estesa e completata per diventare la *MIB-II*
- | Essenzialmente una vista in lettura a supporto del *Fault* e *Configuration Management* del protocollo TCP/IP
- | Possibilità di configurazione (controllo) limitata (preferibile per i limiti di SNMPv1 a livello di sicurezza)
- | Inadeguato per IPv6 (indirizzi IP codificati a 4 byte)

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB-II : gruppi di...

SYSTEM



Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB-II : gruppo "system"

- Informazioni generali sull'apparato, ad esempio:

- » sysDescr: "Cisco Gateway"
- » sysObjectID: 1.3.6.1.4.1.9.1.1
- » sysUpTime: 37153422 (in secondi)
- » sysContact: "helpdesk@science.unitn.it"
- » sysName: "bordergateway.unitn.it"
- » sysLocation: "Via Sommarive, ; 2. piano"
- » sysServices: 6 (bridge and router functions)

				√	√	
Application (e.g., NFS Server)	End-to-end (e.g., Host)	Internet Layer (e.g., router)	Site-Link Layer (e.g., Bridge)	Physical Layer (e.g., repeater)		

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB-II : gruppo "interfaces" (1)

- Informazioni relative alle interfacce dell'apparato
- Le interfacce possono essere fisiche o logiche
- Contiene un contatore (ifNumber) e una tabella:  
ifNumber non è altro che il numero di righe nella tabella
- Informazioni di stato: ifAdminStatus e ifOperStatus  
(valori possibili: up (1), down (2), testing (3))
- Informazione di tipo: ifType. Ad esempio:

6	Ethernet	23	PPP
15	FDDI	24	Loopback
20	ISDN Basic	32	Frame Relay

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB-II: gruppo "interfaces" (2)

- | Le altre variabili sono dei contatori dei pacchetti ricevuti e inviati, con statistiche su:
  - » pacchetti scartati
  - » pacchetti in errore
  - » pacchetti con protocollo sconosciuto, ecc.
- | Limitazioni della prima versione (RFC 1213):
  - » contatori a 32-bit rapidamente saturi
  - » non c'è distinzione tra interfacce fisiche e logiche
  - » non c'è distinzione tra pacchetti di multicast e di broadcast
  - » è difficile catturare le "layered interfaces"
- | Alcune limitazioni sono state superate in RFC successive.

## MIB-II: gruppo "at"

- | Address Translation: mapping tra gli indirizzi fisici (interfacce fisiche) dell'apparato e gli indirizzi IP
- | "deprecated" in MIB-II (presente per compatibilità con MIB-I)
- | L'informazione è stata spostata sotto il gruppo ip

## MIB-II : gruppo "ip" (1)

- | Informazioni relative al funzionamento del protocollo IP sull'apparato
- | Non tutti gli oggetti definiti sono applicabili dato un tipo di apparato (host vs. router)
- | `ipForwarding`: indica se l'apparato è usato come IP gateway
- | `ipDefaultTTL`: valore di Time To Live inserito nei pacchetti inviati
- | Variabili di statistiche/contatori su pacchetti inviati, ricevuti, scartati, in errore, ecc.
- | 3 tabelle

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB-II : gruppo "ip" (2)

- | `ipAddrTable`: tabella degli indirizzi IP configurati sull'apparato, con la subnet mask e il mapping sulla tabella delle interfacce (attributo `ipAdEntIfIndex`)
- | `ipNetToMediaTable`: tabella di mapping tra indirizzi IP e indirizzi fisici (MAC)
- | `ipRouteTable`: *routing table* dell'apparato; limitata dalla sua indicizzazione sul solo indirizzo IP di destinazione; venne sostituita con la tabella `ipForward` nella RFC 1354, indicizzata per (indirizzo IP dest, protocollo, policy, next hop)

Corso di Reti di Calcolatori II – Anno accademico 2004/2005

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## MIB-II: gruppo "icmp"

- | Informazioni relative al protocollo ICMP (protocollo di controllo, sotto il livello IP)
- | Essenzialmente costituito da contatori:
  - » messaggi ICMP ricevuti
  - » messaggi ICMP inviati
  - » messaggi in errore

## MIB-II: gruppo "tcp"

- | Informazioni relative al funzionamento del protocollo TCP
- | Alcune variabili con informazioni di configurazione (timer di ritrasmissione, limite numero connessioni TCP...)
- | Alcuni contatori di connessioni attive, di connessioni fallite, di segmenti ricevuti e inviati, ecc.
- | Una tabella con lo stato delle connessioni (indicizzate per: indirizzo IP e porta del locale, indirizzo IP e porta del remoto)

## MIB-II : gruppo "udp"

- | Informazioni relative al funzionamento del protocollo UDP
- | Alcuni contatori di datagrams ricevuti, di datagrams scartati per assenza di applicazione su quella porta, di datagrams inviati, ecc.
- | Una tabella con la lista dei *listener* UDP in attesa su una determinata porta

## MIB-II : gruppo "egp"

- | Informazioni relative al funzionamento del protocollo EGP (External Gateway Protocol) sul nodo
- | Alcuni contatori di messaggi ricevuti e inviati, nonché di messaggi in errore
- | Una variabile con l'Autonomous System di appartenenza dell'apparato
- | Una tabella con le informazioni di stato e di statistiche per ognuno dei "neighbor" del nodo
- | Non tanto usato, poiché è più diffuso il protocollo BGP (Border Gateway Protocol), per il quale è stata estesa la MIB-II in una RFC successiva

## MI B-11 : gruppo "transmission"

- | Dettaglio del livello fisico (MAC) dell'apparato
- | Nessun oggetto definito nella RFC 1213
- | Le specifiche sono arrivate in RFC successive (per ogni tecnologia)

## MI B-11 : gruppo "snmp"

- | Informazioni relative al funzionamento del protocollo SNMP
- | Solo variabili scalari (nessuna tabella) con tanto di statistiche su:
  - » pacchetti ricevuti e inviati
  - » richieste ricevute di get, set, ecc.
  - » risposte inviate per tipologie di richieste
  - » casistiche di errori ricevuti (corrispondenti ai vari valori di Error Status della PDU SNMP), e.g.:

`snmpInTooBigs, snmpInNoSuchNames, snmpInBadValues, snmpInReadOnlys,  
snmpInGenErrs`