Appunti delle lezioni di Gestione di Sistemi e Reti

6. MIB Standard

Con il diffondersi e con il successo di SNMP come architettura di gestione, è aumentato il numero di tipi di apparecchiature, servizi e risorse in genere che il mercato è interessato a gestire. Per poter gestire in modo robusto e conveniente risorse prodotte da diversi produttori, occorre che esse offrano la stessa MIB: per raggiungere questo obiettivo è necessario che esista, per ciascun tipo di risorsa, una MIB standard, definita da un organismo vendor-independent, che gli utilizzatori possano chiedere ai produttori di implementare per i loro prodotti.

Gli organismi di standardizzazione di Internet hanno sviluppato numerose *MIB standard*, perché molti sono i sistemi che il mercato richiede che siano gestiti in modo indipendente dal venditore. Cercate sul sito dell'editor degli RFC (http://www.rfc-editor.org) i documenti che contengono MIB nel titolo e si vedrà il grande numero di MIB standardizzate o in via di standardizzazione.

In questo capitolo esaminiamo brevemente due MIB standard, per avere una idea della gestione che si può effettuare tramite una MIB.

6.1. MIB-II

Lo scopo di questa MIB (in pratica priva di nome) è di permettere la gestione delle risorse che devono essere presenti su un agente SNMP per il solo fatto che l'agente esiste, indipendentemente dal sistema che l'agente permette di gestire. Quindi la implementazione della MIB-II è obbligatoria per ogni genere di agente SNMP. Questa MIB è stata definita RFC 1213, ed è la seconda versione di questa MIB minimale; la prima era stata definita in RFC 1156, ma essendo la prima standard mai definita da Internet presentava tanti problemi che fu presto superata da MIB-II.

Nel decidere quali oggetti inserire in questa MIB, i progettisti hanno seguito un certo numero di criteri, alcuni dei quali sono importanti per qualunque MIB, altri invece motivati specificamente dall'intenzione di definire una MIB minimale. Esaminiamo questi criteri, perché permettono anche di comprendere le problematiche che i progettisti di una MIB devono affrontare per assolvere al loro compito.

6.1.1. Criteri di Progetto della MIB-II

I criteri adottati, esplicitamente citati in RFC 1213, sono i seguenti.

1. *Un oggetto deve essere necessario per fault o per configuration management.* Gli obiettivi di MIB-II sono molto limitati, in particolare per quanto riguarda le aree funzionali di gestione che si vuole indirizzare. Ovviamente questo criterio non è valido in generale, perché ci si dovrebbe porre l'obiettivo di realizzare tutte le aree funzionali di gestione.
2. S*ono permessi solo oggetti di controllo "deboli"*. Per oggetti deboli si intende oggetti che non possano causare gravi danni se mal gestiti. Questa strano requisiti è in realtà dovuto alla debolezza di SNMP dal punto di vista della sicurezza (vederemo bene quando parleremo del protocollo): è quindi facile violare le protezioni di SNMP e utilizzare gli oggetti di gestione per danneggiare il sistema gestito Se gli oggetti non permettono di fare "danni", anche l'attacco alla sicurezza del

sistema di gestione non potrà fare danni. I noti che con un oggetto debole non si può però neanche fare molto del bene; quindi questo obbiettivo è una dichiarazione di "impotenza" molto significativa da parte dei progettisti dell'architettura di gestione di SNMP.

3. *Ci devono essere prove certe di utilità e di uso corrente.* Il numero di oggetti in una MIB e la complessità del loro comportamento introduce inevitabilmente dei costi nello sviluppo degli agenti di gestione. Perché un sistema di gestione sia accettato dal mercato occorre non solo che gli utilizzatori lo richiedano ai produttori, ma anche che i produttori riescano a offrirlo a costi ragionevoli.
4. *Quando fu sviluppata MIB-I c'erano solo circa 100 oggetti; limite eliminato in MIB-II.* Il vincolo sul numero di oggetti in MIB-I era dovuto alla necessità di tenere moto semplice la MIB per farla accettare ai produttori. Evidentemente i primi successi di MIB-I fra gli utilizzatori ha permesso agli standardizzatori di essere più "ambiziosi" nel progetto di MIB-II.
5. *Niente oggetti ridondanti.* Un oggetto non deve essere incluso nella MIB se il suo valore è ricavabile dal valore di altri oggetti. Questa è una buona regola nella definizione di ogni database, e questo requisito ha quindi una validità estremamente generale.
6. *Oggetti che sono specifici solo di alcune implementazioni (ad es. Unix-BSD) non devono essere inclusi.* Questo è un vincolo "storico", ma che illustra una tematica che si ha sempre di fronte nel progettare una MIB standard. Alcune esigenze nella gestione della risorsa sono certamente indipendenti dalla implementazione, in quanto sono legate alla funzionalità che la risorsa rende disponibile. Ad es., se dobbiamo standardizzare una MIB per TCP, è evidente che dovremo introdurre oggetti per descrivere una connessione, perché TCP è connection-oriented. Ma se per implementare il servizio della connessione una implementazione utilizza risorse che sono molto particolari, tanto che altre implementazioni non ne fanno uso, è ragionevole introdurre oggetti per descrivere questi aspetti molto particolari di una implementazione? Se la risposta fosse affermativa, la MIB diventerebbe rapidamente piena di oggetti la maggior parte dei quali non ha nessun senso in una qualunque implementazione. È però anche vero che se nella MIB inseriamo solo oggetti che hanno senso in tutte le implementazioni note, la MIB potrebbe risultare troppo povera di informazione. Questo è il dilemma che ci si trova sempre davanti nella definizione di una MIB standard. In generale, Internet ha sempre privilegiato la semplicità.
7. *Occorre evitare di strumentare eccessivamente sezioni critiche di codice.* Strumentare il codice significa inserire al suo interno elementi di intervento in lettura o in scrittura, o addirittura aggiungere software di misura (contatori o gauge. La strumentazione del codice rallenta il codice, sia perché il software di misura è carica la CPU, sia perché è comunque necessario assicurare mutua esclusione fra il codice eseguito in favore dell'agent e quello che assicura le funzionalità della risorsa. La mutua esclusione rallenta l'esecuzione e carica il sistema operativo che la deve assicurare.

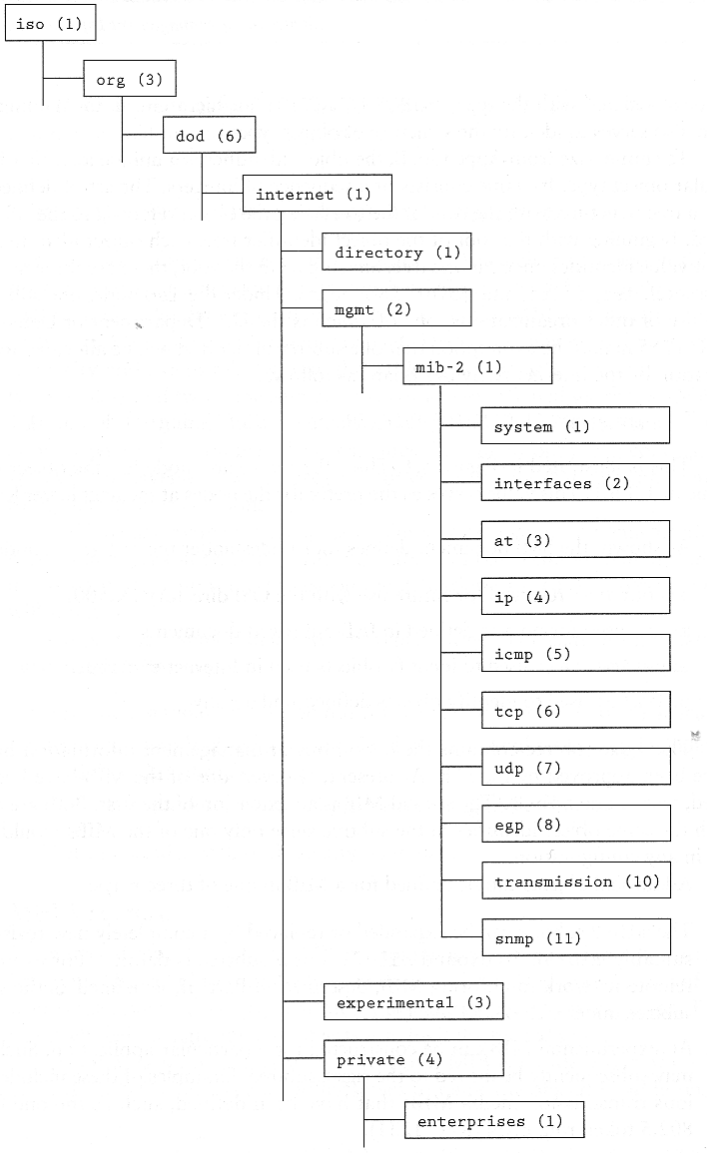
Una ovvia conseguenza dei requisiti di cui sopra è che la MIB-II contiene solo oggetti essenziali e quindi nessun oggetto è opzionale.

Gruppi presenti nella MIB-II

Gli oggetti della MIB-II sono suddivisi in sottoalberi, per aumentare la chiarezza e aiutare sia gli implementatori che gli utilizzatori della MIB (i gestori del sistema) a comprendere il significato degli oggetti presenti. La suddivisione in gruppi viene ottenuta introducendo per tali gruppi opportuni OID, e poi ovviamente introducendo gli OID per gli object type al di sotto di tali OID.

Se la semantica di un gruppo si applica al sistema gestito, deve essere implementato tutto il gruppo: ad es. se sulla macchina su cui esegue l'agent c'è TCP, tutto il gruppo che descrive TCP deve essere presente nella implementazione della MIB.

Gli identificatori principali della MIB-II sono illustrati nella figura seguente.



Si noti la collocazione dell'albero della MIB-II all'interno dell'albero globale degli OID. Questa collocazione fa comprendere che ISO aveva già previsto che organizzazioni private avessero necessità di introdurre OID "privati", e aveva riservato per tali scopi l'albero `iso.org`. L'organizzazione che ha chiesto un albero per assegnare al suo interno OID privati è stato il DOD, Department of Defense del governo degli Stati Uniti. L'ISO ha concesso a tale organizzazione l'uso dell'albero `iso.org.dod`, che può anche essere denotato come `1.3.6`. Prevedendo che più agenzie del DOD potessero avere bisogno di assegnare OID, il DOD ha riservato per l'IETF (l'organismo di standardizzazione di Internet) l'albero `iso.org.dod.internet`. IETF a sua interno ha partizionato il suo albero in tre sottoalberi, introducendo `directory`, `mgmt`, `experimental`, e `private`. Lo scopo dei primi tre sottoalberi è chiaro. Il quarto sottoalbero `private` mostra che anche IETF ha previsto che proprio per realizzare i suoi standard le aziende dovessero dotarsi di OID "privati" e quindi ha riservato un sottoalbero per assegnarvi spazi privati per le aziende. Finalmente, nell'area del management (`mgmt`) la prima MIB definita e quindi che aveva bisogno di un sottoalbero per i propri OID è stata MIB-II. Vedremo che in realtà le altre MIB relative a risorse più specifiche di quelle minimali che comunque debbano essere gestite saranno introdotte come *estensione* di MIB-II, quindi all'interno dell'albero

`iso.org.dod.internet.mgmt.mib2`, come fratelli dei nodi che descriviamo nel seguito. Questo è un approccio tipico di Internet che preferisce "estendere" la MIB-II piuttosto che introdurre alberi separati allo stesso livello di MIB-II.

I gruppi di MIB-II sono:

1. `system`: informazione generale
2. `interfaces`: descrive le interfacce di rete presenti (almeno una c'è altrimenti non ci sarebbe l'agent)
3. `at`: (address translation) deprecata e quindi non viene più usata perché rimpiazzata da altre informazioni
4. `ip`: descrive la implementazione e "esperienza" di IP
5. `icmp`: implementazione e esperienza di ICMP
6. `tcp`: implementazione e esperienza di TCP
7. `udp`: implementazione e esperienza di UDP
8. `egp`: implementazione e esperienza di EGP (protocollo di routing, potrebbe benissimo non esserci)
9. `transmission`: descrive gli schemi di trasmissioni e i protocolli di accesso di ciascuna interfaccia
10. `snmp`: implementazione e esperienza di SNMP

6.1.2. Il Gruppo `system`

L'albero degli object type del gruppo `system` è descritto dalla prossima figura. Si noti la convenzione di iniziare tutti i nome degli object type con il prefisso `sys`, convenzione che non solo aiuta a ricordare dove trovare la definizione dell'object type all'interno dell'RFC, ma facilita il lavoro in parallelo di più gruppi nel definire nuovi nomi per nuovi OID. Dalla lettura dell'RFC della MIB-II si vede che sono tutti object type scalari.

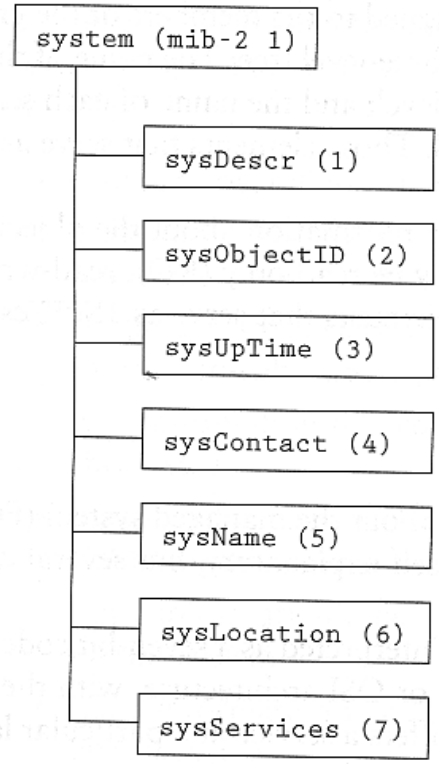


FIGURE 6.1 MIB-II `system` group

L'oggetto (ma dovremmo sempre dire "l'object type") `sysServices` si interpreta come una maschera a 7 bit, in cui ogni bit settato indica che il corrispondente livello dello stack ISO-OSI è implementato sulla macchina (il livello 1 è rappresentato dal bit di peso 0, e così via). Interpretando invece i 7 bit come un numero intero senza segno di ottiene una valore numerico, che è rappresentato nella istanza di questo object type.

$$sysServices = \sum 2^{L-1},\ L \in S$$

Si contano solo i servizi *principali* offerti al resto della rete, i servizi end-to-ed di un host (IP+trasporto) vengono considerati al livello 4. Il livello 5 (sessione) e 6 (presentazione) in internet non sono presenti, e quindi una macchina con applicazioni distribuite offre i livelli 4 e 7 per cui `sysServices` vale 72.

`sysUpTime` contiene il tempo passato dal boot dell'agent, misurato in centesimi di secondo. Viene quindi resettato quando l'agent viene re-inizializzato. Questo oggetto è molto importante perché permette al manager di rilevare che l'agent ha resettato (è difficile che un agent rimanga in operazione per tanto tempo da arrivare al numero massimo esprimibile di timeticks!) e quindi tutti gli oggetti che hanno come sintassi Counter sono altresì stati resettati.

`sysObjectID` contiene un OID che identifica il costruttore e il modello del sistema gestito: deve essere allocato nello spazio degli OID del costruttore, che quindi ne deve possedere uno. Alcuni lo intendono come l'OID che identifica il costruttore dell'agent di gestione, ma leggendo bene l'RFC (il parametro DESCRIPTION della macro OBJECT-TYPE) si capisce che è una interpretazione scorretta.

Lo scopo di questo object type è permettere al manager di capire che sistema è gestito. L'uso di MIB standard non permette di capire la natura del sistema gestito proprio perché tutti dovrebbero avere la stessa MIB. È però anche vero che la conformità delle implementazioni allo standard può non essere perfetta e quindi è bene avere uno strumento per individuare la esatta natura del sistema gestito. Si noti l'introduzione di questo object type obbliga i costruttori di sistemi gestiti mediante SNMP ad avere uno spazio di OID a loro riservato. Per non "disturbare" ISO, IETF ha introdotto un sottoalbero apposito nel suo spazio di OID dedicato a questo scopo. L'OID del sottoalbero è `iso.org.dod.internet.private.enterprises`. Le aziende che vogliono mettere sul mercato un prodotto gestito mediante un agent SNMP devono chiedere alla IANA il conferimento di un OID per poter assegnare OID ad ogni nuovo prodotto che commercializzano. Tale OID dovrebbe identificare il tipo, il modello e la versione del prodotto, e assegnarne uno nuovo solo quando il prodotto si comporta in modo diverso da un altro dello stesso tipo e modello.

6.1.3. Il Gruppo interfaces

Questo gruppo contiene informazioni generiche sulle interfacce fisiche di rete del sistema, incluse informazioni di configurazione e statistiche sul traffico. La prossima figura rappresenta la struttira di questo gruppo; la freccia laterale indica i campi INDEX nelle tabelle.

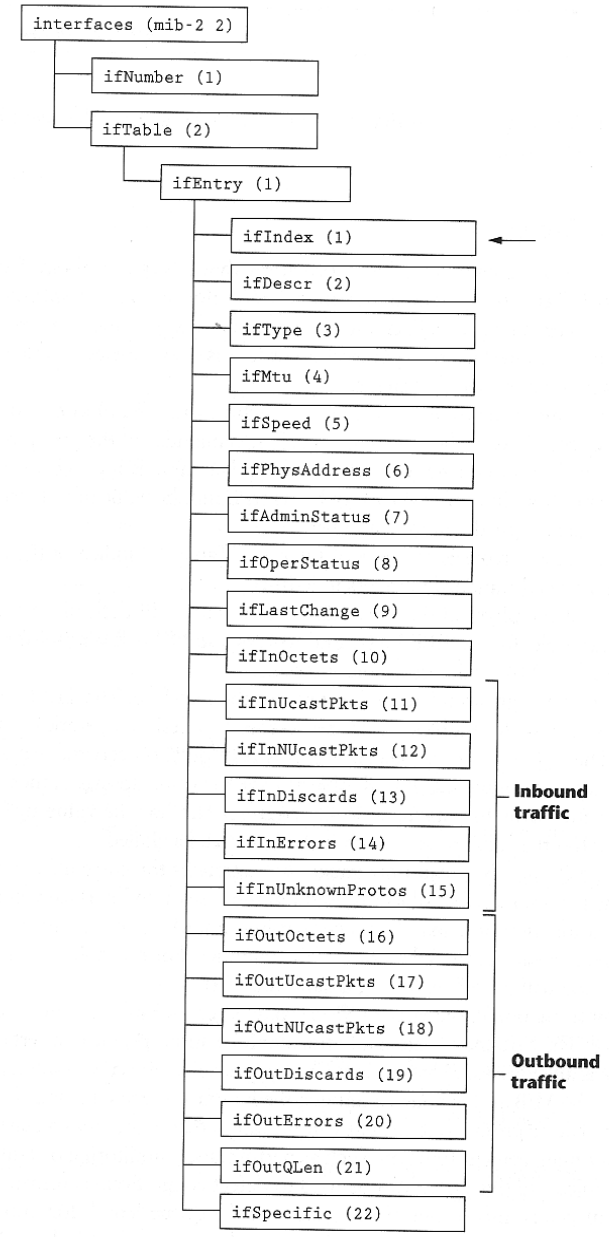


FIGURE 6.2 MIB-II `interfaces` group

Ogni interfaccia è considerata collegata ad una rete fisica, eventualmente anche punto-punto. Il gruppo è costituito da uno scalare, `ifNumber`, e da una tabella `ifTable`. `ifNumber` è un object type che contiene il numero delle interfacce fisicamente presenti nel sistema, qualunque sia lo stato in cui sono. Nella tabella vi sarà una riga per ogni interfaccia presente nel sistema e quindi `ifNumber` esprime il numero di righe che sono presenti in `ifTable`.

`ifTable` è indiciata da `ifIndex`, che è semplicemente un intero, progressivo fra 1 e `ifNumber` inclusi, assegnato arbitrariamente dall'agent, con l'unico vincolo che non cambia fra un reboot dell'agent e il successivo. L'agent ha però la libertà di assegnare ad una certa interfaccia un diverso valore di `ifIndex` ogni volta che si re-inizializza: per questa ragione `ifIndex` non ha alcun significato per il manager, che deve trovare un altro modo per "riconoscere" la stessa interfaccia fra un reboot dell'agent e il successivo. Si noti che il tipo dell'interfaccia non è sufficiente perché il sistema può avere due interfacce dello stesso tipo e neanche la descrizione testuale dell'interfaccia fornita dal costruttore; alla fine l'unico fattore veramente utile a individuare la singola interfaccia è il suo indirizzo fisico (e non in ogni caso!).

`ifAdminStatus` specifica lo stato dell'interfaccia che il manager ha richiesto settando il valore di questo oggetto: per disabilitare l'interfaccia occorre settarlo a `down (2)`, per riabilitarla settarlo ad `up (1)`. Il campo `ifOperStatus` indica invece lo stato in cui si trova l'interfaccia, e ovviamente è read-only. Il manager setta `ifAdminStatus` a `up`, e invece la interfaccia va in `ifOperStatus` uguale a `down`, allora vuol dire che l'interfaccia è guasta. Quindi è sufficiente confrontare questi due valori per diagnosticare un guasto dell'interfaccia. `ifLastChange` riporta il valore che `sysUpTime` aveva quando è avvenuto l'ultimo cambiamento di stato dell'interfaccia.

`ifSpeed` è un gauge che stima la velocità della interfaccia: utile per quelle reti in cui la velocità può cambiare in diverse condizioni di utilizzo (per Ethernet è fisso).

Si noti la significativa quantità di counter che riportano statistiche sul traffico sull'interfaccia (e nella figura vi è un errore nel marcare i dati relativo al traffico in ingresso dalla rete fisica e quelli relativi al traffico in uscita verso la rete fisica). È curioso che vi riportata la lunghezza della coda di uscita verso la rete, mentre non vi è una simile coda in ingresso: forse è stato ritenuto che non tutte le implementazioni di interfacce di rete sono dotate di coda in ingresso!

Interessante è l'object type ifSpecific che contiene un pointer ad un'altra parte della MIB che contiene informazioni specifiche a seconda del tipo della interfaccia. Come ci si poteva aspettare tale pointer è di tipo `OBJECT IDENTIFIER`; se assente deve valere `0.0`, che è un valido OID che ogni implementazione dovrebbe essere in grado di accettare e di generare.

Il gruppo `interfaces` contiene quindi informazione basilare utile per iniziare a fare una qualunque forma di gestione, ad es. performance o fault management.

6.1.4. Il Gruppo `at`

Il gruppo `Address Translation` è oramai abbandonato perché insufficiente a fornire informazioni sul meccanismo di traduzione da indirizzo logico (IP) a indirizzo fisico.

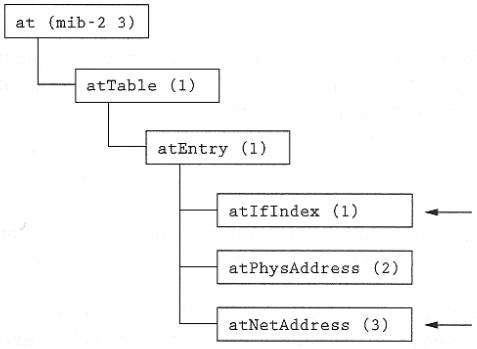


FIGURE 6.4 MIB-II address translation group

Notiamo solo che la corrispondenza fra parametri di traduzione e scheda di rete è data inserendo in `atIndex`, usato per indiciare la `atTable`, lo stesso intero che viene usato per indicare la interfaccia nella `ifTable`. Si usa anche l'indirizzo di rete come indice (più righe possono avere lo stesso `atIndex`), ma se non si conosce l'indirizzo di rete non si riesce ad accedere ad una riga. Più tardi spiegheremo a cosa serve tale campo indice!

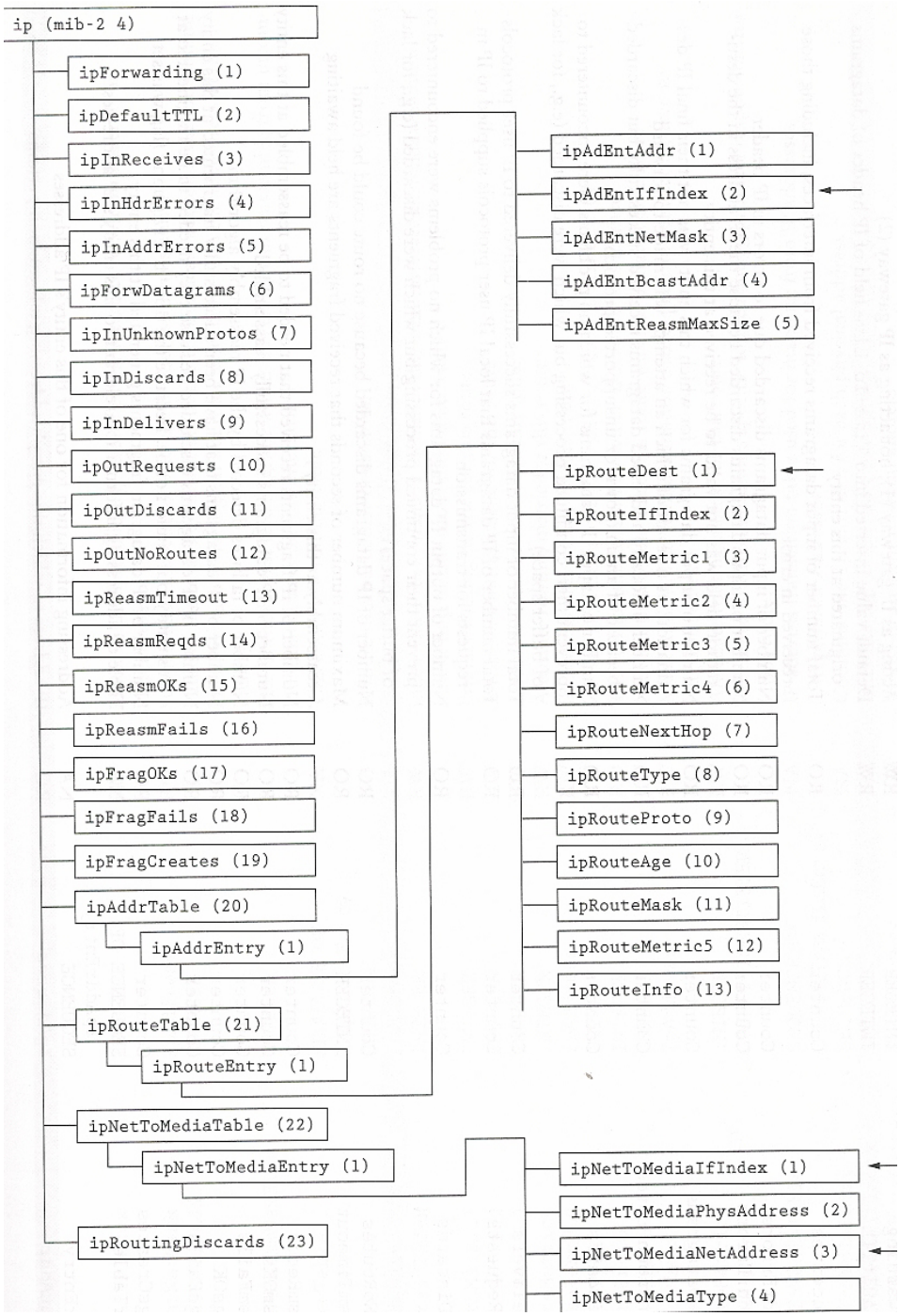
Questa tabella di traduzione è stata abbandonata perché:

- la stessa interfaccia può supportare più protocolli di rete ci sono più indirizzi per la stessa interfaccia
- serve anche il mapping da fisico a logico
- la tabella rappresenta solo metodi di traduzione tabellari

La traduzione è ora affidata al gruppo che descrive il protocollo di rete adottato, e in questo modo il meccanismo di traduzione può essere più specifico.

6.1.5. Il Gruppo `ip`

Sono interessanti le due tabelle qui contenute:



`ipAddrTable` è la tabella che descrive il mapping fra indirizzi logici e indirizzi fisici assegnati alla scheda. Notare che anche in questo caso il "link" fra questa tabella e quella delle interfacce è dato dalla posizione che la scheda occupa nella `ifTable`, contenuto nel campo `ipAdEntIfIndex`. I campi della tabella sono tutti read-only: non si può usare per fare configurazione (applicazione del principio di *"non nuocere"*!). L'INDEX è `ipAdEntAddr` e non `ipAdEntIfIndex` come dice il libro: ci possono essere più indirizzi assegnati ad ogni scheda!

La seconda tabella di interesse è `ipRouteTable`. Contiene sia informazione usata da IP per instradare (*tabella di routing*) che informazione usata/calcolata dai protocolli applicativi di routing (vedi Reti 2), quindi *NON* da IP. La rete da usare per il next hop è espressa in termini di indirizzo del router, `ipRouteNextHop`, ma anche dall'indice della interfaccia da usare, contenuto in `ipRouteIfIndex`. Infatti, in molte implementazioni di IP, in particolar modo quelle per le linee punto-punto (che possono anche essere anonime) il next hop viene in realtà configurato indicando semplicemente la interfaccia da usare per raggiungerlo.

`ipRouteTable` è indiciata da `ipRouteDest`, ma questo è stato un grave errore di specifica, perché la tabella può contenere multiple route per la stessa destinazione per

- Differenti TOS
- Load sharing/balancing

Nell'unico RFC della MIB-II approvato non è specificato come trattare questo caso. Gli RFC 1354 (nel 1992), 2096, 4292 (ancora tutti "proposed") definiscono una nuova tabella:

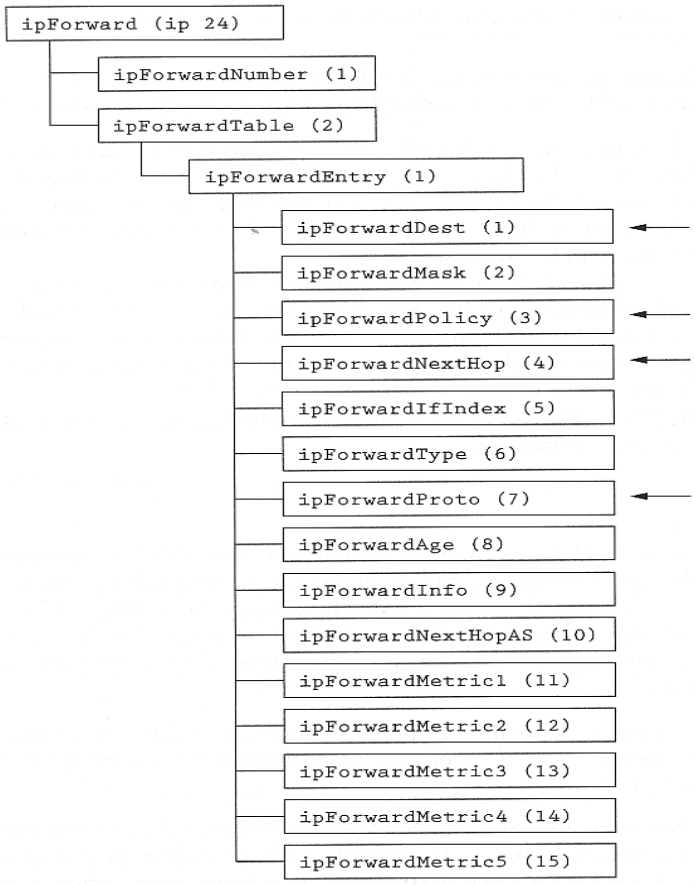


FIGURE 6.7 IP forwarding table MIB

`ipForwardPolicy` specifica per quale TOS questa è la route. `ipForwardProto` specifica come è stata "appresa" questa strada: *locale* (vari tipi) o vari *protocolli di routing* (vedi RdE2). Gli INDEX di questa tabella sono `ipForwardDest`, `ipForwardNextHop`, `ipForwardPolicy` e `ipForwardProto`.

Notate che il gestore difficilmente conosce tutti e quattro i valori: come fa allora ad accedere alla tabella?? abbiamo bisogno di capire di più come è definito il protocollo SNMP di accesso!

6.1.6. Il Gruppo `icmp`

Lo scopo di questo gruppo è ovvio; misura sia ia traffico in ingresso che in uscita.

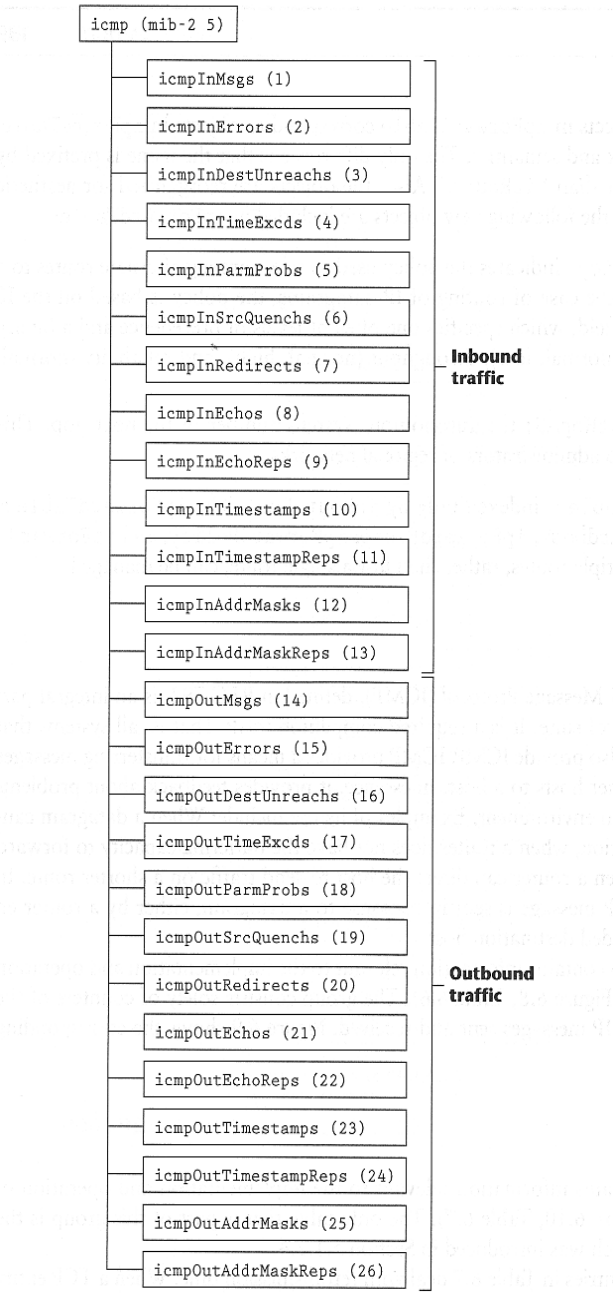


FIGURE 6.8 MIB-II `icmp` group

6.1.7. Il Gruppo `tcp`

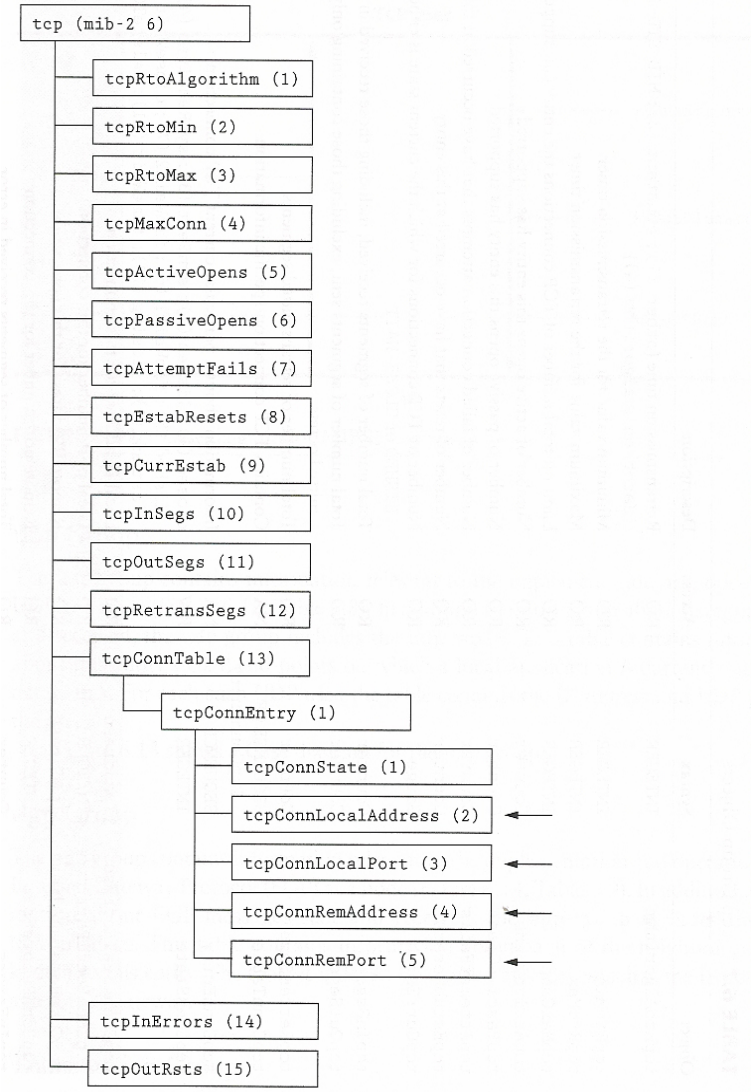


FIGURE 6.10 MIB-II `tcp` group

Sono da notare:

- La specifica dell'algoritmo e dei parametri di ritrasmissione adottati
- Statistiche sulle connessioni e sugli errori
- La tabella delle connessioni di cui abbiamo già parlato

6.1.8. Il Gruppo udp

Molto più semplice del gruppo tcp:

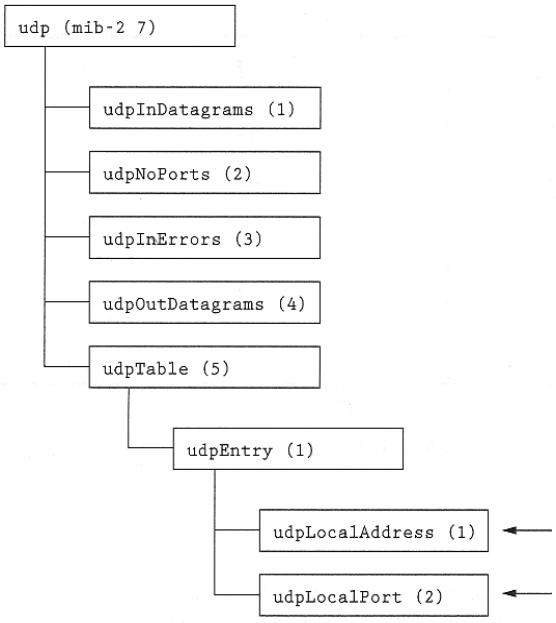


FIGURE 6.12 MIB-II udp group

6.1.9. Il Gruppo dot3

`dot3StateTable` contiene statistiche sul funzionamento di interfacce di rete di tipo CSM/CD. È una tabella perché ci possono essere più schede; il link è fatto tramite il valore di `ifIndex`.

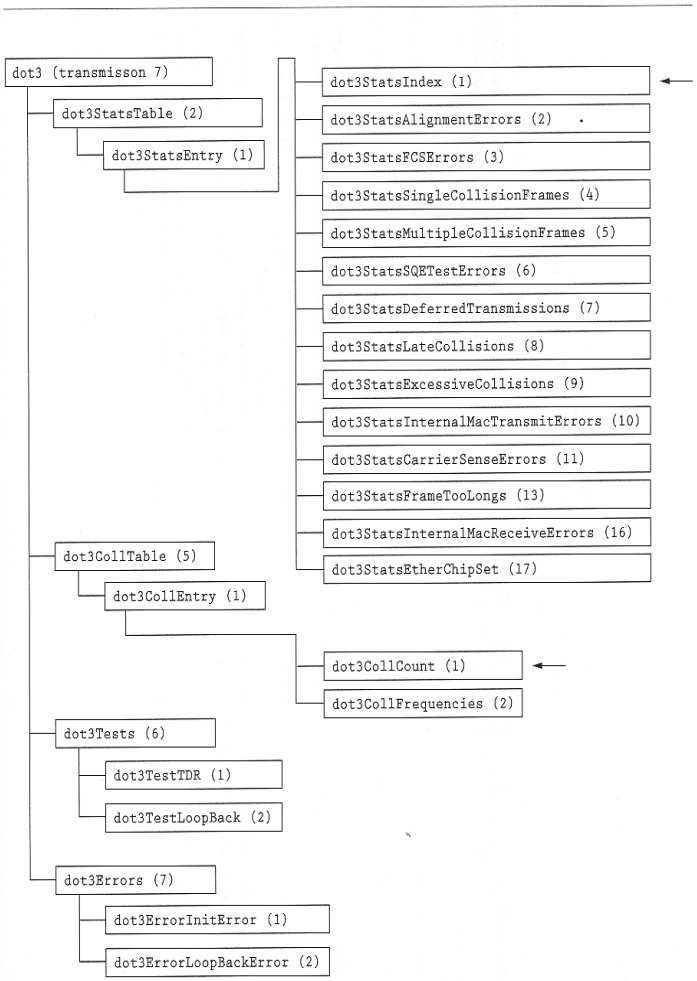


FIGURE 6.15 EtherLike MIB

I contatori che contano i frame che hanno avuto collisioni non permettono di capire come vanno le collisioni, perché sono incrementati anche per numeri diversi di collisioni. Per questo è utile la seconda tabella ...

`dot3CollTable` è una tabella che registra quanti frame sono stati trasmessi con una collisione, con due, e così via. L'INDEX è `dot3CollCount`, ma anche dal valore di `ifIndex` nella tabella delle interfacce e non è mostrato in figura perché è un valore "fuori" dalla tabella: questa è una "feature" di SNMPv2! Con questi dati può costruire un istogramma come quello mostrato in figura