2.3 SMI

Cada objeto a gestionar debe ser representado por un objeto. El MIB es una colección estruturada de objetos. Cada nodo en el sistema será mantenido en el MIB, que reflejará el estado del recurso gestionado en ese nodo. Para que esto sea posible, se necesita un esquema común de representación que soporte la interoperatibilidad entre los nodos.

2.3.1 Estructura de la información de gestión

Determina la forma en la que debe definirse y construirse una MIB.

Proporciona técnicas de estandarización para:

- Definir la estructura de un determinado MIB.
- Definir los objetos individuales, incluyendo la sintaxis y valor.
- Codificación de los valores de estos objetos.

Estructura de la MIB

Cada tipo de objeto de un MIB tiene asociado un `OBJECT IDENTIFIER` que lo nombra y cuya organización es jerárquica (en forma de árbol). Este valor consiste en una secuencia de enteros no negativos.

GESTIÓN INTERNET

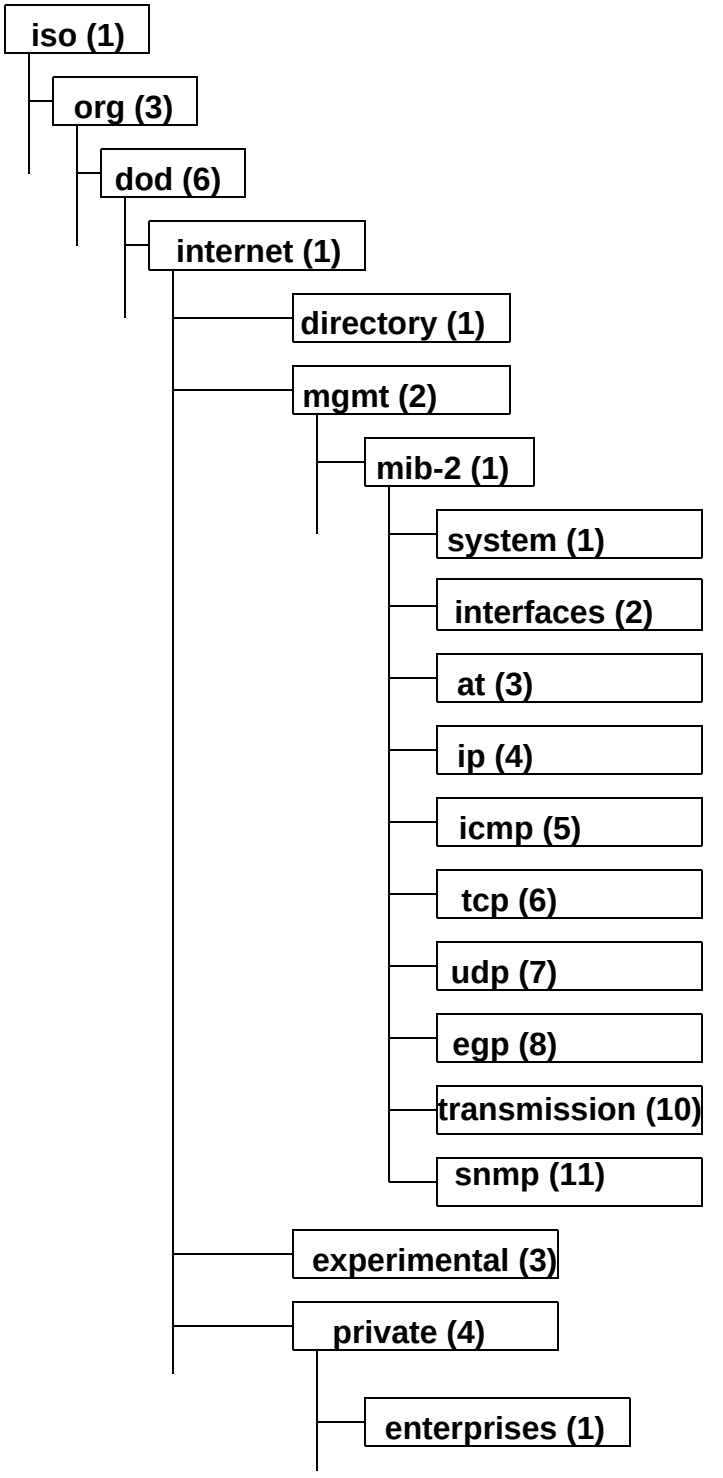
El subárbol 1.3.6.1 se refiere a la subrama de internet que define cuatro tipos de nodos internet (grupos):

- ***directory*:** Reservado para OSI directory (X.500).
- ***mgmt*:** Destinado a objetos definidos por la IAB (MIB I y MIB II).
- ***experimental*:** Objetos usados en temas experimentales.
- ***private*:** Objetos definidos de manera unilateral (empresas).

Las estructura de estas MIBs son definidas utilizando la sintaxis ASN.1

GESTIÓN INTERNET

ARBOL DE REGISTRO ISO



GESTIÓN INTERNET

Contructores de la Macro del RFC 1212:

SYNTAX: Sintaxis abstracta del tipo de objetos. Los tipos de datos aceptables son un subconjunto muy reducido de los valores permisibles en ASN.1

Tipos Básicos:

INTEGER (UNIVERSAL 2), OCTET STRING (UNIVERSAL 4), NULL (UNIVERSAL 5), OBJECT IDENTIFIER (UNIVERSAL 6), SEQUENCE, SEQUENCE OF (UNIVERSAL 16).

Tipos Compuestos:

- Network Address: CHOICE de direcciones de red. Actualmente, sólo...
- IpAddress: Dirección IP en 32 bits.
- Counter: Entero no negativo que sólo puede incrementarse (hasta 2E32-1). Si se alcanza el máximo, vuelve a 0.
- Gauge: Entero no negativo que puede crecer o decrecer limitado a 2E32-1. Si se alcanza, permanece ahi hasta un reset.
- TimeTicks: Entero no negativo que representa el tiempo transcurrido en centésimas de segundo desde un determinado instante identificado en la definición del objeto que lo utiliza.
- Opaque: Para enviar datos de tipos arbitrarios como secuencias de octetos.

ACCESS: Define la forma en que una instancia de un objeto puede ser accedida. permite restricciones específicas para una implementación.

STATUS: Especifica si se requiere que sea soportado por las implementaciones.

DescrPart: Descripción textual de la semántica del objeto.

ReferPart: Referencia textual a otro objeto definido en otros modulos.

IndexPart: Requerido cuando se definen tablas.

DefValPart: Se utiliza para definir valores por defecto para los tipos de datos. Se utilizan cuando se crean las instancias en el agente.

VALUE NOTATION: Nombre del objeto necesario para acceder a él cuando se utiliza SNMP (Su posición dentro del árbol de registro).

Definición de tablas:

El soporte lo van a dar:

- El tipo SEQUENCE y SEQUENCE OF de ASN.1.
- El constructor INDEX de la Macro Object Type.

No se permiten anidaciones de tabla.

Para definir una tabla en SNMP:

1.- Definición de un objeto xxxTable:

- Sintaxis: SEQUENCE OF Tipo
- Not accessible.

2.- Definición de un objeto xxxEntry debajo de xxxTable:

- Sintaxis: Tipo ::= SEQUENCE. Un campo por columna de la tabla.
- Not accessible.
- Constructor INDEX indicando el(los) campo(s) que identifican unívocamente cada entrada de la tabla.

3.- Definición de N objetos columna debajo de xxxEntry:

- Sintaxis: coincidente con el campo correspondiente del SEQUENCE anterior.

- Regla de acceso dependiente de la semántica de la columna.

Ejemplo: La tabla de conexiones de TCP.

Codificación de los valores:

Siguiendo las Basic Encoding Rules de ASN.1

2.3.2 Management Information Base I (RFC 1155)

Constituyó la primera MIB normalizada y estandarizaba la definicion de los objetos de la torre de protocolos TCP/IP:

GRUPO	NUMERO	DESTINO
system	3	El propio sistema
interfaces	22	Interfaces de red
at	3	Correspondencia de direcciones IP.
ip	33	Internet Protocol.
icmp	26	Internet Control Message Protocol.
tcp	17	Transmision Control Protocol
udp	4	User Datagram Protocol
egp	6	Exterior Gateway Protocol

2.3.3 Management Information Base II (RFC 1213)

GRUPO	NUMERO	DESTINO
system	3	El propio sistema
interfaces	22	Interfaces de red
at	3	Correspondencia de direcciones IP.
ip	33	Internet Protocol.
icmp	26	Internet Control Message Protocol.
tcp	17	Transmision Control Protocol
udp	4	User Datagram Protocol
egp	6	Exterior Gateway Protocol

Define un superconjunto de la MIB I pero con objetos y grupos adicionales. Se desestimó la tabla at (address translation).

GRUPO	NUMERO	COMENTARIOS
system	7	Eran 3
interfaces	223	Eran 22
at	3	Mantenidos por compatibilidad
ip	38	Eran 33
icmp	26	Sin cambio
tcp	19	Eran 17
udp	7	Nueva tabla
egp	18	Expansión de tabla

GRUPO	NUMERO	COMENTARIOS
trnasmission	0	Nuevo grupo
snmp	30	Nuevo grupo

Criterios de diseño de la MIB-II:

- Se incluyen objetos esenciales para gestión de configuración y fallos. Y de los que sea evidente su necesidad actual.
- Sólo se permiten objetos de control débiles (o sea, que no puedan causar muchos destrozos si son mal utilizados)
- Límite en el número de objetos para adecuarse a la tecnología.
- Se evita al máximo la redundancia de información.
- Sólo se incluyen aspectos genéricos, no dependientes de la implementación => No hay objetos opcionales.

Los objetos gestionados están divididos en grupos por cercanía semántica. La idea es que un agente debe implementar un grupo completo si es aplicable al equipo que controle.

2.3.4 MIBs experimentales

Son MIBs en desarrollo por los grupos de trabajo de Internet. Actualmente existen multitud de MIBs

experimentales:

TITULO	RFC	FECHA
IEEE 802.5 Token Ring	1231	Mayo 1991
IEEE 802.4 Token Bus	1230	Mayo 1991
IEEE 802.3 Repeater Devices	1516	Septiembre 1993
Ethernet	1398	Enero 1993
FDDI	1512	Septiembre 1993
RMON	1271	Noviembre 1991
ATM	1695	Agosto 1994
Modem	1696	Agosto 1994

2.3.5 MIBs privadas

MIBs de productos específicos, que añaden funcionalidad a las MIB estándar.

Los fabricantes las hacen públicas por Internet (depósito común en `isi.edu`).

MIBs para productos de Cabletron, Synoptis, ATT, Cisco, IBM, etc....

2.3.6 EL PROTOCOLO SNMP

RFC 1157

Sólo se permiten GET, SET y TRAP.

No se permiten cambios en la estructura de la MIB (añadir, eliminar objetos), invocar acciones...

Sencillez de implementación, limitaciones en la funcionalidad.

Communities:

Relación 1-n entre agente y gestores, con el agente manteniendo la información de la MIB (necesidad de mecanismos de seguridad).

Community: Relación entre un agente y varios gestores con unas características determinadas de control de acceso y autentificación.

La definición de las *communities* es local al agente y llevan asociado un nombre.

Autentificación:

¡Se usa simplemente el *community name* sin cifrar!

Política de control de acceso:

Definiendo diferentes *communities* podemos permitir distintas **vistas de la MIB,** con diferentes objetos y diferentes permisos de acceso (READ-ONLY, READ-WRITE) -> Es el **community profile.**

Hay que combinar la declaración de acceso realizada con SMI al definir la MIB con la del community profile del gestor que accede:

Según la MIB	**Según la Community**	**Según la Community**
	READ-ONLY	READ-WRITE
READ-ONLY	GETs y TRAPs.	GETs y TRAPs.
READ-WRITE	GETs y TRAPs.	GETs, SETs y TRAPs.
WRITE-ONLY	GETs y TRAPs El valor depende de la implementación.	GETs SETs y TRAPs El valor depende de la implementación en los GETs y TRAPs.
NOT-ACCESSIBLE	No disponible.	No disponible.

Identificación de instancias:

En base al `OBJECT IDENTIFIER`, pero....

...¿cómo lo hacemos en las tablas?

Cada tabla contiene cero o más filas con conjuntos de objetos columna. Cada objeto columna lleva asociado su OID (OID de la tabla + Identificador).

Falta conocer a qué fila nos referimos: Para ello se utiliza la clave que permite identificar cada fila, indicada por el constructor `INDEX` de SMI (puede ser múltiple).

`y.(i1).(i2). . . . .(in)` , donde `y` es el OID del objeto columna y `ij`, son subidentificadores con los valores de sus campos índice para ese objeto.

Representación de los valores de los objetos como subidentificadores:

- Enteros: Un subidentificador con el valor.
- Cadenas de longitud fija: n subidentificadores, uno por caracter.
- Cadenas de longitud variable: un subidentificador con la longitud (n) más n para la cadena.
- Oids: un subidentificador con la longitud (n) más n para la cadena de números.
- Direcciones IP: Cuatro identificadores con el formato habitual.

Para algunas tablas antiguas esto no se cumplia: una solución era marcar las líneas con la misma clave con otro identificador, pero la tendencia ha sido irlas eliminando de la MIB.

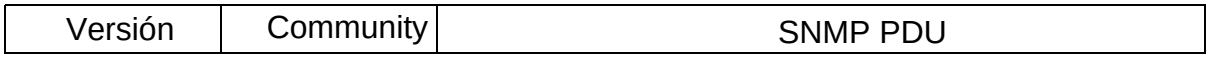
No hay identificadores para tablas ni filas (no son accesibles).

Los objetos escalares llevan el subidentificador 0 para diferenciar tipo e instancia.

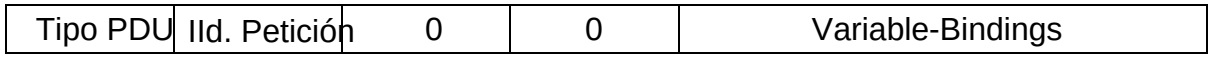
Mediante este sistema, podemos establecer una ordenación entre todos los objetos existentes en la MIB.

PDUs

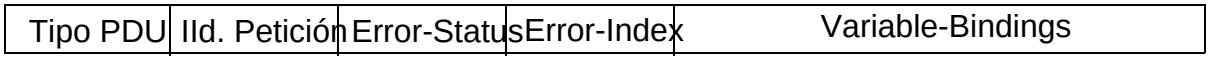
Formato del mensaje:



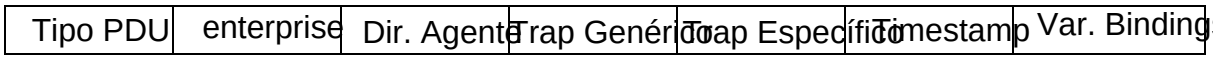
Mensaje SNMP



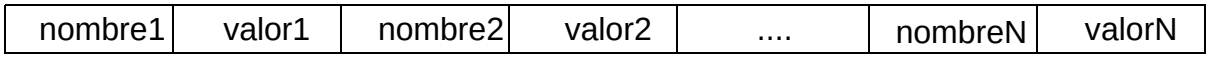
GetRequest, GetNextRequest, SetRequest PDUs



GetResponse PDU



Trap PDU



Variable-Bindings

Algunos campos:

- **Id. Petición**: para distinguir entre múltiples peticiones, eliminar duplicados.

- **errorStatus:** Usado para indicar excepción al procesar una petición: noError(0), tooBig(1), nosuchName(2), badValue(3), readOnly(4), genErr(5).

- **errorIndex:** en caso de error, indica la variable que lo causó.

- **variableBindings:** lista de instancias con sus valores. Para efetuar peticiones sobre múltiples instancias. En

algunos casos los valores van vacíos (GET requests), recomendándose NULL.

- **enterprise:** tipo de objeto generador de la trap (sysObjectId).

- **Dir. Agente:** dirección del agente que genera la trap.

- **Trap genérico:** tipo de trap: coldStart(0), warmStart(1), linkDown(2), linkUp(3), authenticationFailure(4), egpNeighborLoss(5), enterpriseSpecific(6).

- **Trap específico:** Código específio de la trap.

- **timeStamp:** Tiempo desde la última reinicialización del agente (sysUpTime).

Transmisión:

1.- Se construye la PDU.

2.- Se envía al servicio de autentificación, junto con las direcciones de transporte correspondientes y el community name. Puede encriptarlo..., aunque usualmente no hace nada.

3.- Se construye el mensaje con la versión, el community y el resultado del proceso 2.

4.- Se codifica según las BER.

Recepción:

1.- Comprobación sintáctica (eventual descarte).

2.- Verificación de la versión utilizada.

3.- Se envía el community, la PDU y las direcciones al servicio de autentificación, que realiza las correspondientes comprobaciones. Si falla se genera un trap de autentificación, si no se retorna la PDU en formato RFC 1157.

4.- Si la sintaxis es correcta, se aplica la política de acceso correpondiente a la community y se procesa la petición.

GetRequest:

Las variables pedidas están en el variableBindings.

Atómico: O se obtiene valor de todas o de ninguna en un GetResponse.

Posibles errores:

- No existe instancia o es de un tipo agregado (no un nodo hoja): Se retorna noSuchName y errorIndex=posición de la variable errónea.
- Respuesta muy grande: tooBig.
- Otros motivos: genErr y la posición en errorIndex.

GetNextRequest:

Muy similar, produce un getResponse, mismos errores.

Retorna el valor del objeto siguiente según el orden establecido entre las instancias.

Atómico.

Si no hay siguiente, noSuchName.

Utilidad: Descubrir una MIB no conocida y recorrer tablas de la MIB.

Hay menos posibilidades de fallo que con la anterior,

porque si un objeto no es visible, obtendremos el siguiente.

SetRequest:

Formato similar, pero aquí en la parte de valores de la variableBindings tenemos los valores que queremos alterar en la MIB.

Se retorna un GetResponse con la lista de valores si todo ha ido bien o vacía si algún error ha ocurrido.

Atómico.

Los mismos errores que las anteriores, pero además puede que el valor sea inconsistente con el tipo (badValue, con la posición en errorIndex).

Para alterar una tabla, podemos usar SetRequest en los objetos columna.

También podemos añadir filas enteras, alterando en un único SetRequest valores a todos los objetos columna de una tabla, incluído el índice.

Si alteramos un índice inexistente, puede (depende de la implementación) crear una línea con valores por defecto.

Podemos también borrar líneas de tablas con ciertos valores especiales en determinados objetos columna

(dependiente de la implementación). Ej., ipRouteType = invalid.

La invocación de acciones puede también simularse con un setRequest. Ej., un objeto reBoot.

Trap:

Envío asíncrono de información al gestor.

variableBindings tiene aquí un significado dependiente de la implementación.

No requiere respuestas.

Significado de los definidos en la norma:

coldStart: reinicialización de la entidad SNMP no esperada.

warmStart: reinicialización de la entidad SNMP esperada.

linkDown: error en alguna de las líneas de comunicaciones del agente. En la lista de variableBindings nos añade la información del interfaz (ifIndex) con problemas.

linkUp: Fin de los problemas en alguna de las líneas de comunicaciones del agente. En la lista de

variableBindings nos añade la información del interfaz (ifIndex) correspondiente.

authenticationFailure: error de autentificación en algún mensaje llegado al agente.

egpNeighborLoss: indica la caída de algún vecino egp.

enterpriseSpecific: Otros, indicado en specificTrap.

Soporte del nivel de transporte:

UDP: Puertos 161 (Agente), 162 (Managers, para traps).

No fiable, no orientado a conexión, pero no precisa de conexiones establecidas.

También se ha definido el uso de CLTS, COTS.

Si el servicio no es fiable:

- Gets: reinvocarlos. requestId filtra duplicados.
- Sets: asegurarse de que el valor no ha cambiado.
- Traps: sondear al agente por los fallos relevantes.

Diferencias en las implemenatciones deben ser vigiladas (traps que no se emiten, valores de atributos falsos...)

Debido a la escasez de traps estándares y la posibilidad de que o no se emitan o se pierdan, la monitorización SNMP debe basarse en un esquema de sondeo.

GESTIÓN INTERNET

El número de agentes manejables depende drásticamente de varios factores:

- Número de Agentes (N).
- Tiempo de Respuesta del Agente a una petición SNMP.
- Retardo de la Red.
- Intervalo de Sondeo (Delta).

$$N \leq \frac{T}{\Delta}$$

No olvidar los posibles efectos del exceso de carga en la red por el tráfico del sondeo.

Limitaciones SNMP:

- No es adecuado a la gestión de redes grandes, por los aspectos del sondeo: requiere grandes volúmenes de tráfico.
- No está pensado para enviar grandes cantidades de información.
- Los traps no son confirmados y utilizan un servicio no fiable, por lo que pierden casi su sentido.
- Problemas de seguridad, utiliza una autentificación casi trivial, lo que le hace muy poco adecuado para control.
- No dispone de un mecanismo potente de invocación de operaciones, con parámetros y valores de retorno, sino que se simulan con SETs sobre algunos objetos.
- Limitaciones del modelo de información.
- No está pensado para disponer de jerarquías de gestores: comunicación manager-manager.

```
IMPORTS   ObjectName, Object Syntax FROM RFC-1155-SMI

OBJECT-TYPE MACRO ::=
BEGIN
     TYPE NOTATION ::=   "SYNTAX"   type (TYPE ObjectSyntax)
                         "ACCESS"   Access
                         "STATUS"   Status
                         DescrPart
                         ReferPart
                         IndexPart
                         DefValPart
     VALUE NOTATION ::= value (VALUE ObjectName)

     Access ::= "read-only"|"read-write"|"write-only"|"not-accessible"

     Status ::= "mandatory|"optional"|"obsolete"|"deprecated"

     DescrPart ::= "DESCRIPTION" value (description DisplayString)|empty

     ReferPart ::= "REFERENCE" value (reference DisplayString)|empty

     IndexPart ::= "INDEX" "{" IndexTypes "}"

     IndexTypes ::= IndexType|IndexTypes "," IndexType

     IndexType ::= value (indexobject ObjectName)  --if indexobject, use the SYNTAX
                                                   --value of the correspondent
                                                   --OBJECT-TYPE invocation
                     |type (indextype)             --otherwise use named SMI type;
                                                   --must conform to IndexSyntax below

     DefValPart ::= "DEFVAL" "{" value (defvalue ObjectSyntax) "}" |empty

     DisplayString ::= OCTET STRING SIZE (0..255)

END

IndexSyntax ::= CHOICE { number INTEGER (0..MAX),
                         string OCTET STRING,
                         object OBJECT IDENTIFIER,
                         address NetworkAddress,
                         IpAddress IpAddress }
```

FIGURE 5.3 Macro for managed objects (RFC 1212)

```
RFC1155-SMI DEFINITIONS ::= BEGIN

EXPORTS -- EVERYTHING
            internet, directory, mgmt, experimental, private, enterprises, OBJECT-TYPE,
            ObjectName, ObjectSyntax, SimpleSyntax, ApplicationSyntax, Network Address,
            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 }

-- 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

-- names of objects in the MIB

ObjectName ::= OBJECT IDENTIFIER

--syntax of objects in the MIB

ObjectSyntax ::= CHOICE {simple SimpleSyntax,

       --note that simple SEQUENCES are not directly mentioned here to keep things
       --simple (i.e., prevent misuse). However, application-wide types which are
       --IMPLICITly encoded simple SEQUENCEs may appear in the following CHOICE

                          application-wide ApplicationSyntax}

SimpleSyntax ::= CHOICE {number INTEGER,
                         string OCTET STRING,
                         object OBJECT IDENTIFIER
                         empty NULL}
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] OCTET STRING      --arbitrary ASN.1 value, "double-wrapped"

END
```

FIGURE 5.4 Structure of management information (RFC 1155)

```
tcpMaxConn OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
         "The limit on the total number of TCP connections the entity can
         support. In entities where the maximum number of connections is
         dynamic, this object should contain the value -1."
     ::= { tcp 4 }
```

FIGURE 5.5 Example of an object definition

```
tcpConn Table OBJECT-TYPE
        SYNTAX   SEQUENCE OF TcpConnEntry
        ACCESS   not-accessible
        STATUS   mandatory
        DESCRIPTION
              "A table containing TCP connection-specific information."
        ::= { tcp 13 }

tcpConnEntry OBJECT-TYPE
       SYNTAX    TcpConnEntry
       ACCESS    not-accessible
       STATUS    mandatory
       DESCRIPTION
             "Information about a particular TCP connection. An object of this type is
             transient, in that it ceases to exist when (or soon after) the connection
             makes the transition to the CLOSED state."
       INDEX     { tcpConnLocalAddress,
                   tcpConnLocalPort,
                   tcpConnRemAddress,
                   tcpConnRemPort }
       ::= { tcpConnTable 1 }

TcpConnEntry ::= SEQUENCE { tcpConnState INTEGER,
                            tcpConnLocalAddress IpAddress,
                            tcpConnLocalPort INTEGER (0..65535),
                            tcpConnRemAddress IpAddress
                            tcpConnRemPort INTEGER (0..65535)}

tcpConnState OBJECT-TYPE
       SYNTAX   INTEGER {closed (1),
                         listen (2),
                         synSent (3),
                         synReceived (4),
                         established (5),
                         finWait1 (6),
                         finWait2 (7),
                         closeWait (8),
                         lastAck (9),
                         closing (10),
                         timeWait (11),
                         delete TCB (12) }

       ACCESS      read-write
       STATUS      mandatory
       DESCRIPTION
             "The state of this TCP connection.
```

continued

FIGURE 5.6 MIB-II specification of TCP connection table (RFC 1213)

```
                The only value which may be set by a management station is deleteTCB(12)."
                Accordingly, it is appropriate for an agent to return a 'badValue'
                response if a management station attempts to set this object to any other
                value.

                If a management station sets this object to the value deleteTCB(12), then
                this has the effect of deleting the TCB (as defined in RFC 793) of the
                corresponding connection on the managed node, resulting in immediate ter-
                mination of the connection.

                As an implementation-specific option, a RST segment may be sent from the
                managed node to the other TCP endpoint (note however that RST segments are
                not sent reliably)."
        ::= { tcpConnEntry 1 }

tcpConnLocalAddress OBJECT-TYPE
        SYNTAX      IpAddress
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The local IP address for this TCP connection. In the case of a connec-
                tion in the listen state which is willing to accept connections for any
                IP interface associated with the node, the value 0.0.0.0 is used."
        ::= { tcpConnEntry 2 }

tcpConnLocalPort OBJECT-TYPE
        SYNTAX      INTEGER (0..65535)
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The local port number for this TCP connection."
        ::= { tcpConnEntry 3 }

tcpConnRemAddress OBJECT-TYPE
        SYNTAX      IpAddress
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The remote IP address for this TCP connection."
        ::= { tcpConnEntry 4 }

tcpConnRemPort OBJECT-TYPE
        SYNTAX       INTEGER (0..65535)
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The remote port number for this TCP connection."
        ::= { tcpConnEntry 5 }
```

FIGURE 5.6 MIB-II specification of TCP connection table (RFC 1213) *Continued*

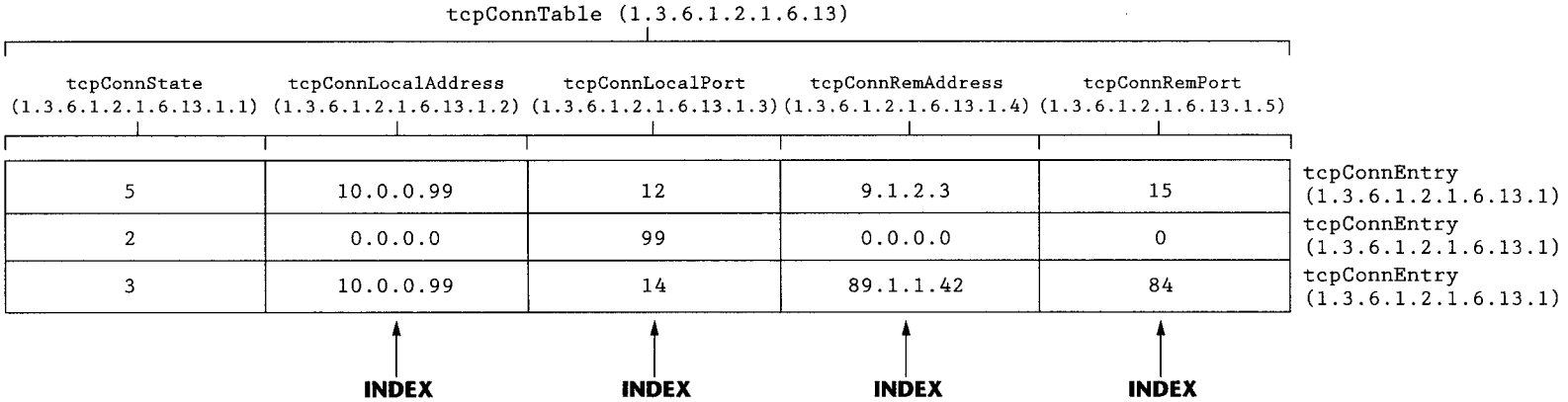


FIGURE 5.7 Instance of a TCP connection table

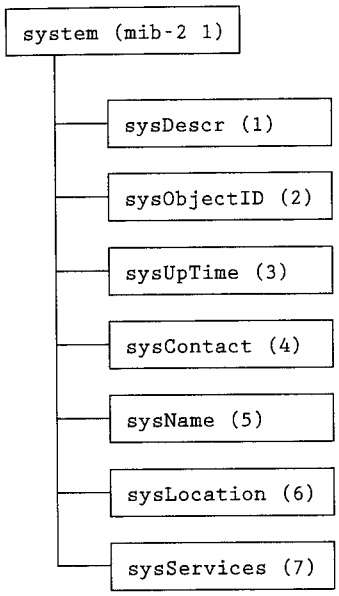


FIGURE 6.1 MIB-II system group

TABLE 6.1 system Group Objects

Object	Syntax	Access	Description
sysDescr	DisplayString (SIZE (0 . . . 255))	RO	A description of the entity, such as hardware, operating system, etc.
sysObjectID	OBJECT IDENTIFIER	RO	The vendor's authoritative identification of the network management subsystem contained in the entity
sysUpTime	TimeTicks	RO	The time since the network management portion of the system was last reinitialized
sysContact	DisplayString (SIZE (0 . . . 255))	RW	The identification and contact information of the contact person for this managed node
sysName	DisplayString (SIZE (0 . . . 255))	RW	An administratively assigned name for this managed node
sysLocation	DisplayString (SIZE (0 . . . 255))	RW	The physical location of this node
sysServices	INTEGER (0 . . . 127)	RO	A value that indicates the set of services this entity primarily offers

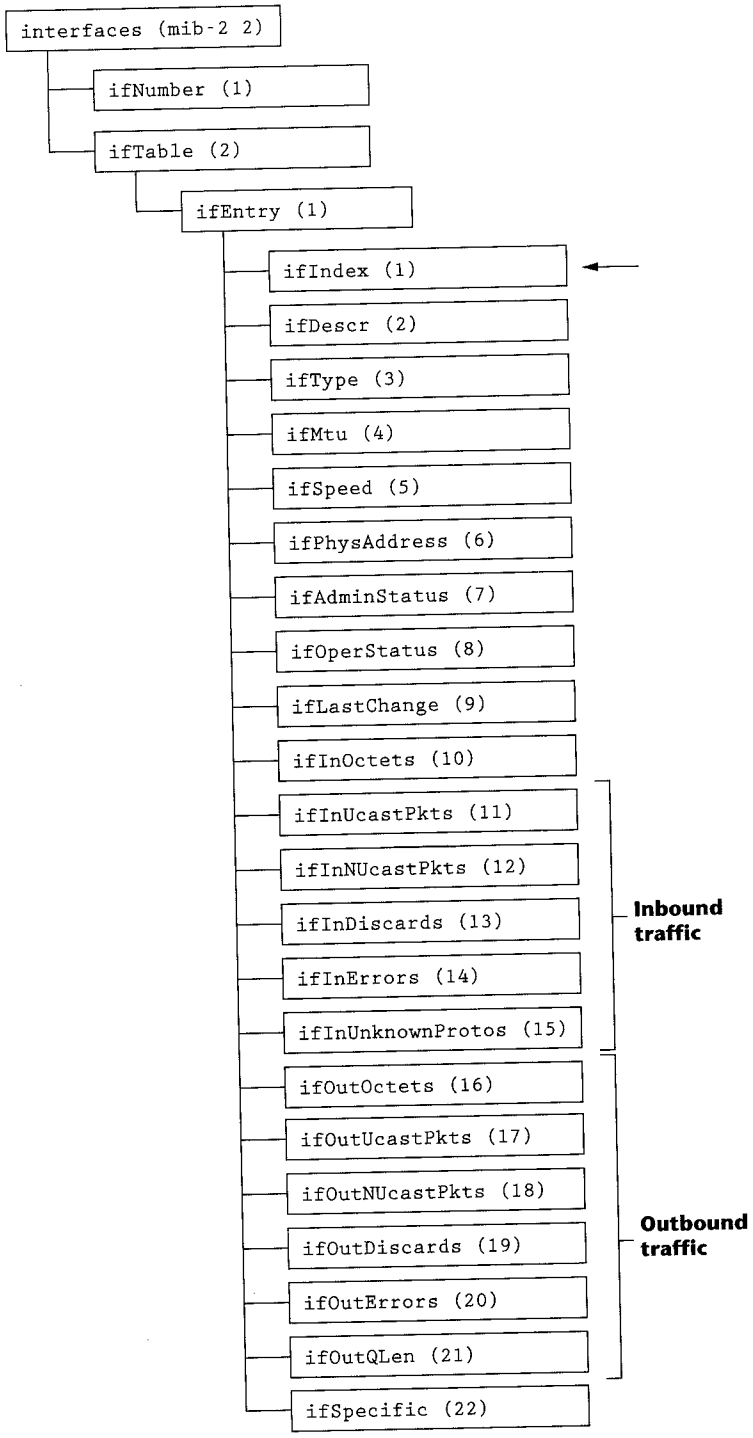


FIGURE 6.2 MIB-II interfaces group

TABLE 6.2 `interfaces` Group Objects

Object	Syntax	Access	Description
`ifNumber`	`INTEGER`	RO	The number of network interfaces
`ifTable`	`SEQUENCE OF ifEntry`	NA	A list of interface entries
`ifEntry`	`SEQUENCE`	NA	An interface entry containing objects at the subnetwork layer and below for a particular interface
`ifIndex`	`INTEGER`	RO	A unique value for each interface
`ifDescr`	`DisplayString (SIZE (0 . . . 255))`	RO	Information about the interface, including name of manufacturer, product name, and version of the hardware interface
`ifType`	`INTEGER`	RO	Type of interface, distinguished according to the physical/link protocol(s)
`ifMtu`	`INTEGER`	RO	The size of the largest protocol data unit, in octets, that can be sent/received on the interface
`ifSpeed`	`Gauge`	RO	An estimate of the interface's current data rate capacity
`ifPhysAddress`	`PhysAddress`	RO	The interface's address at the protocol layer immediately below the network layer
`ifAdminStatus`	`INTEGER`	RW	Desired interface state `(up(1), down(2), testing(3))`
`ifOperStatus`	`INTEGER`	RO	Current operational interface state `(up(1), down(2), testing(3))`
`ifLastChange`	`TimeTicks`	RO	Value of `sysUpTime` at the time the interface entered its current operational state
`ifInOctets`	`Counter`	RO	Total number of octets received on the interface, including framing characters
`ifInUcastPkts`	`Counter`	RO	Number of subnetwork-unicast packets delivered to a higher-layer protocol
`ifInNUcastPkts`	`Counter`	RO	Number of nonunicast packets delivered to a higher-layer protocol
`ifInDiscards`	`Counter`	RO	Number of inbound packets discarded, even though no errors had been detected, to prevent their being deliverable to a higher-layer protocol (e.g., buffer overflow)
`ifInErrors`	`Counter`	RO	Number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol
`ifInUnknownProtos`	`Counter`	RO	Number of inbound packets that were discarded because of an unknown or unsupported protocol
`ifOutOctets`	`Counter`	RO	Total number of octets transmitted on the interface, including framing characters
`ifOutUcastPkts`	`Counter`	RO	Total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or otherwise not sent
`ifOutNUcastPkts`	`Counter`	RO	Total number of packets that higher-level protocols requested be transmitted to a nonunicast address, including those that were discarded or otherwise not sent
`ifOutDiscards`	`Counter`	RO	Number of outbound packets discarded even though no errors had been detected to prevent their being transmitted (e.g., buffer overflow)
`ifOutErrors`	`Counter`	RO	Number of outbound packets that could not be transmitted because of errors
`ifOutQLen`	`Gauge`	RO	Length of the output packet queue
`ifSpecific`	`OBJECT IDENTIFIER`	RO	Reference to MIB definitions specific to the particular media being used to realize the interface

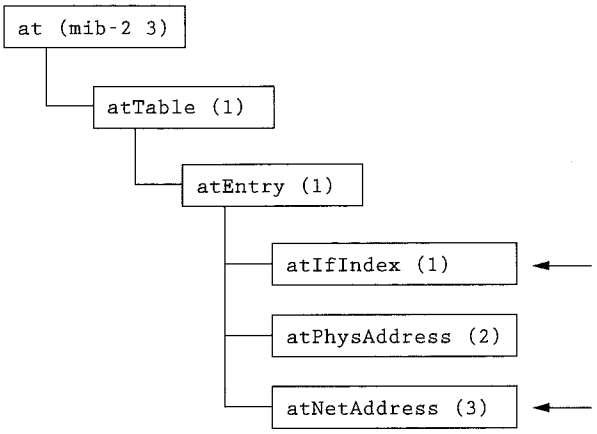


FIGURE 6.4 MIB-II address translation group

TABLE 6.4 Address Translation Group Objects

Object	Syntax	Access	Description
atTable	SEQUENCE OF AtEntry	NA	Contains the NetworkAddress to physical address equivalent
atEntry	SEQUENCE	NA	Contains one NetworkAddress for each physical address
atIfIndex	INTEGER	RW	Interface on which this entry is effective
atPhysAddress	PhysAddress	RW	Media-dependent physical address
atNetAddress	NetworkAddress	RW	The NetworkAddress (e.g., IP address) corresponding to the media-dependent physical address

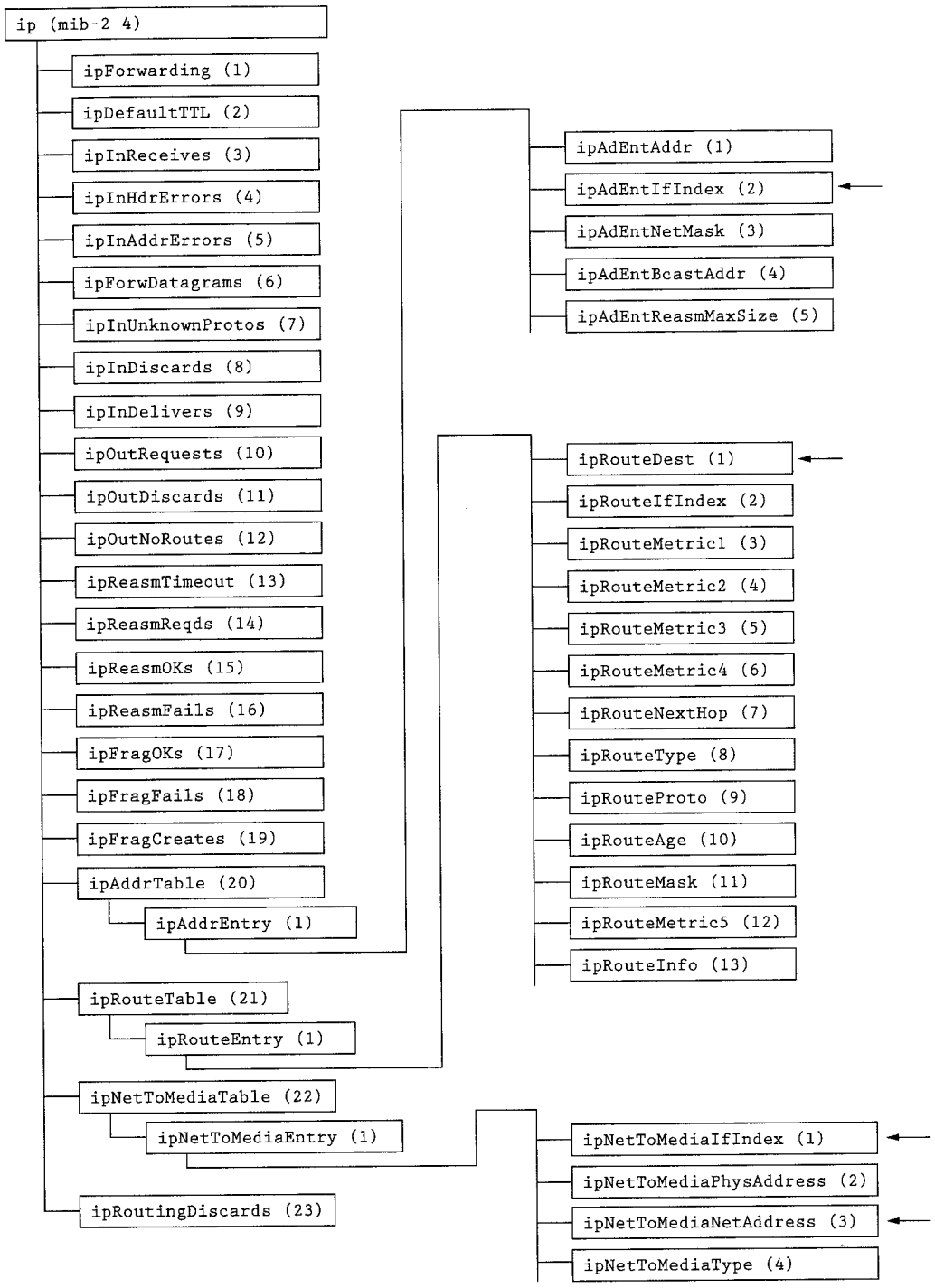


FIGURE 6.5 MIB-II ip group

TABLE 6.5 ip Group Objects

Object	Syntax	Access	Description
ipForwarding	INTEGER	RW	Acting as IP gateway (1); not acting as IP gateway (2)
ipDefaultTTL	INTEGER	RW	Default value inserted into Time-To-Live field of IP header of datagrams originated at this entity
ipInReceives	Counter	RO	Total number of input datagrams received from interfaces, including those received in error
ipInHdrErrors	Counter	RO	Number of input datagrams discarded due to errors in IP header
ipInAddrErrors	Counter	RO	Number of input datagrams discarded because the IP address in the destination field was not valid to be received at this entity
ipForwDatagrams	Counter	RO	Number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to forward
ipInUnknownProtos	Counter	RO	Number of locally addressed datagrams received successfully but discarded because of an unknown or unsupported protocol
ipInDiscards	Counter	RO	Number of input IP datagrams for which no problems were encountered to prevent their continued processing but which were discarded (e.g., for lack of buffer space)
ipInDelivers	Counter	RO	Total number of input datagrams successfully delivered to IP user protocols
ipOutRequests	Counter	RO	Total number of IP datagrams that local IP user protocols supplied to IP in requests for transmission
ipOutDiscards	Counter	RO	Number of output IP datagrams for which no problems were encountered to prevent their continued processing but which were discarded (e.g., for lack of buffer space)
ipOutNoRoutes	Counter	RO	Number of IP datagrams discarded because no route could be found
ipReasmTimeout	INTEGER	RO	Maximum number of seconds that received fragments are held awaiting reassembly at this entity
ipReasmReqds	Counter	RO	Number of IP fragments received that needed to be reassembled at this entity
ipReasmOKs	Counter	RO	Number of IP datagrams successfully reassembled
ipReasmFails	Counter	RO	Number of failures detected by the IP reassembly algorithm
ifFragsOK	Counter	RO	Number of IP datagrams that have been successfully fragmented at this entity
ipFragsFails	Counter	RO	Number of IP datagrams discarded because they needed to be fragmented at this entity but could not be, because the don't-fragment flag was set
ipFragsCreates	Counter	RO	Number of IP datagram fragments generated at this entity
ipAddrTable	SEQUENCE OF IpAddrEntry	NA	Table of addressing information relevant to this entity's IP addresses
ipAddrEntry	SEQUENCE	NA	Addressing information for one of this entity's IP addresses
ipAdEntAddr	IpAddress	RO	IP address to which this entry's addressing information pertains

ipAdEntIfIndex	INTEGER	RO	Index value that uniquely identifies the interface to which this entry is applicable
ipAdEntNetMask	IpAddress	RO	Subnet mask associated with the IP address of this entity
ipAdEntBcastAddr	INTEGER	RO	Value of the least significant bit in the IP broadcast address used for sending datagrams on the logical interface associated with the IP address of this entry
ipAdEntReasmMaxSize	INTEGER	RO	Size of the largest IP datagram which this entity can reassemble from incoming datagrams on this interface
ipRouteTable	SEQUENCE OF IpRouteEntry	NA	This entity's IP routing table
ipRouteEntry	SEQUENCE	NA	A route to a particular destination
ipRouteDest	IpAddress	RW	Destination IP address of this route
ipRouteIfIndex	INTEGER	RW	Index value that uniquely identifies the local interface through which the next hop of this route should be reached
ipRouteMetric1	INTEGER	RW	Primary routing metric for this route
ipRouteMetric2	INTEGER	RW	Alternate routing metric for this route
ipRouteMetric3	INTEGER	RW	Alternate routing metric for this route
ipRouteMetric4	INTEGER	RW	Alternate routing metric for this route
ipRouteNextHop	IpAddress	RW	IP address of next hop of this route
ipRouteType	INTEGER	RW	other (1); invalid (2); direct (3); indirect (4)
ipRouteProto	INTEGER	RO	The routing mechanism by which this route was learned
ipRouteAge	INTEGER	RW	Number of seconds since this route was last updated or verified
ipRouteMask	IpAddress	RW	Mask to be ANDed with destination address before being compared to ipRouteDest
ipRouteMetric5	INTEGER	RW	Alternate routing metric for this route
ipRouteInfo	OBJECT IDENTIFIER	RO	Reference to MIB definitions specific to the routing protocol responsible for this route
ipNetToMediaTable	SEQUENCE OF IpNetToMediaEntry	NA	IP address translation table used for mapping from IP addresses to physical addresses
ipNetToMediaEntry	SEQUENCE	NA	Contains one IpAddress for each physical address
ipNetToMediaIfIndex	INTEGER	RW	Interface for which this entry applies
ipNetToMediaPhysAddress	PhysAddress	RW	Media-dependent physical address
ipNetToMediaNetAddress	IpAddress	RW	IpAddress corresponding to the media-dependent physical address
ipNetToMediaType	INTEGER	RW	Type of mapping: other (1); invalid (2); dynamic (3); static (4)
ipRouting Discards	Counter	RO	Number of routing entries discarded even though valid

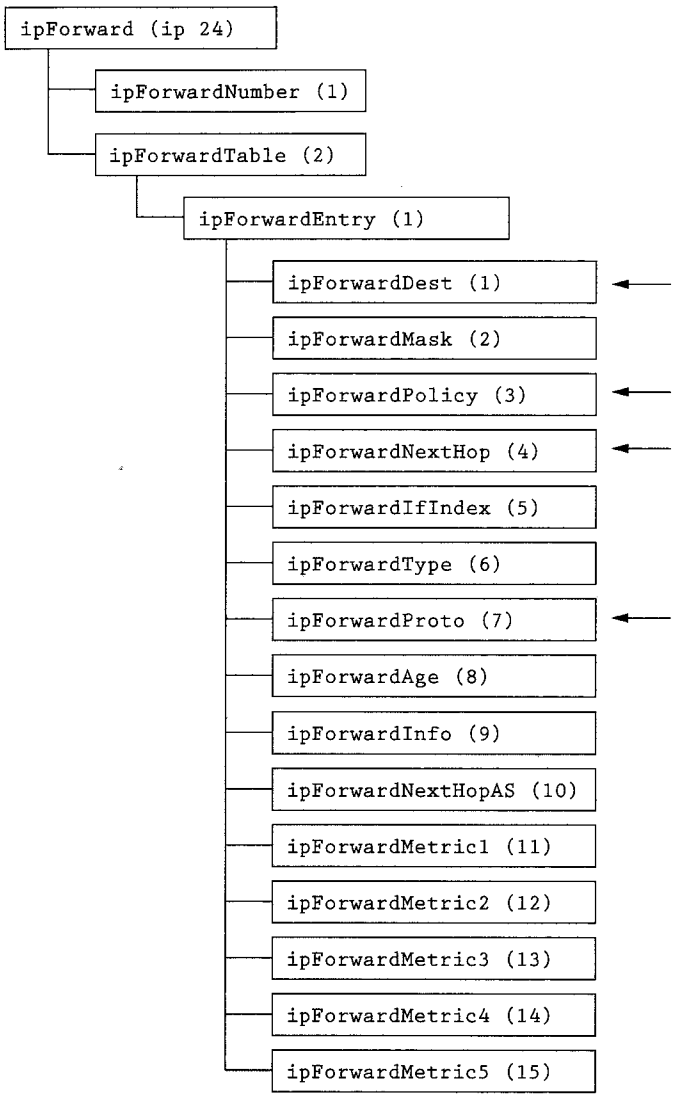


FIGURE 6.7 IP forwarding table MIB

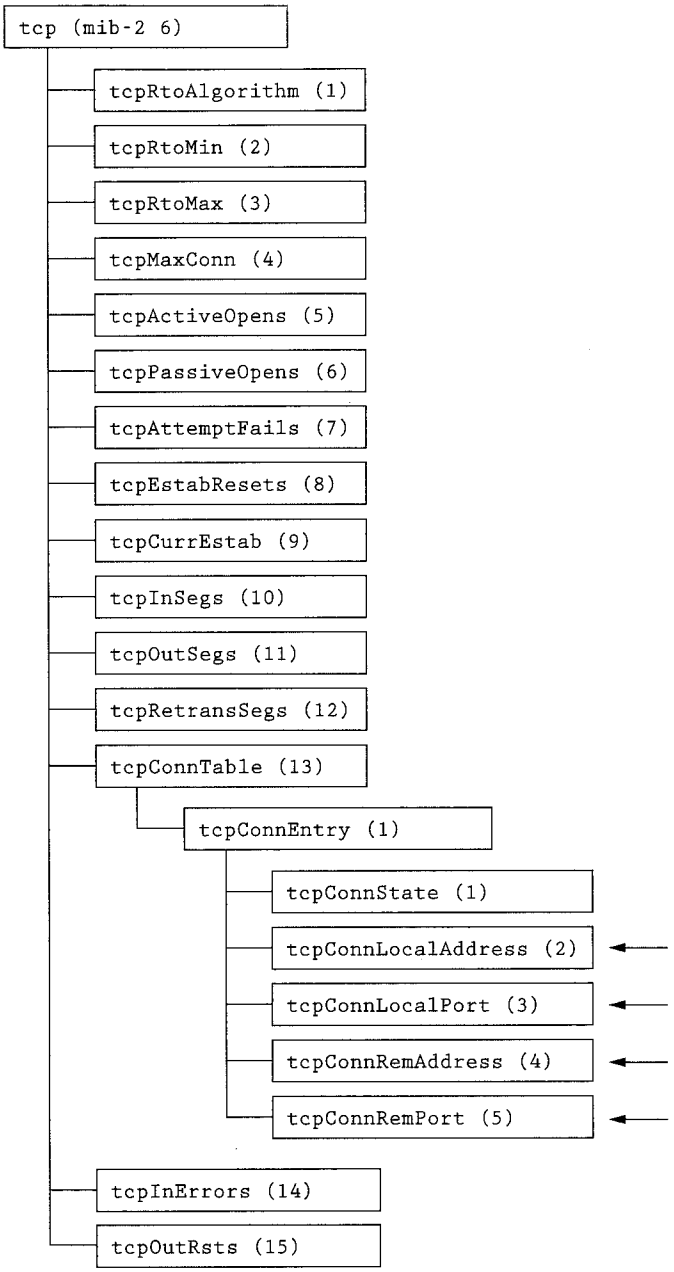


FIGURE 6.10 MIB-II tcp group

TABLE 6.7 tcp Group Objects

Object	Syntax	Access	Description
tcpRtoAlgorithm	INTEGER	RO	Retransmission time (other (1); constant (2); MIL-STD-1778 (3); Jacobson's algorithm (4))
tcpRtoMin	INTEGER	RO	Minimum value for the retransmission timer
tcpRtoMax	INTEGER	RO	Maximum value for the retransmission timer
tcpMaxConn	INTEGER	RO	Limit on total number of TCP connections the entity can support
tcpActiveOpens	Counter	RO	Number of active opens this entity has supported
tcpPassiveOpens	Counter	RO	Number of passive opens this entity has supported
tcpAttemptFails	Counter	RO	Number of failed connection attempts that have occurred at this entity
tcpEstabResets	Counter	RO	Number of resets that have occurred at this entity
tcpCurrEstab	Gauge	RO	Number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT
tcpInSegs	Counter	RO	Total number of segments received, including those received in error
tcpOutSegs	Counter	RO	Total number of segments sent, excluding those containing only retransmitted octets
tcpRetranSegs	Counter	RO	Total number of retransmitted segments
tcpConnTable	SEQUENCE OF TcpConnEntry	NA	Contains TCP connection-specific information
tcpConnEntry	SEQUENCE	NA	Information about a particular current TCP connection
tcpConnState	INTEGER	RW	closed (1); listen (2); synSent (3); synReceived (4); established (5); finWait1 (6); finWait2 (7); closeWait (8); lastAck (9); closing (10); timeWait (11); deleteTCB (12)
tcpConnLocalAddress	IpAddress	RO	Local IP address for this connection
tcpConnLocalPort	INTEGER	RO	Local port number for this connection
tcpConnRemoteAddress	IpAddress	RO	Remote IP address for this connection
tcpConnRemotePort	INTEGER	RO	Remote port number for this connection
tcpInErrors	Counter	RO	Total number of segments received in error
tcpOutRsts	Counter	RO	Number of TCP segments sent containing the RST flag