MIB snmp per la gestione e il monitoraggio di un servent Gnutella.

Rossano Venturini

September 8, 2003

1 Protocollo Gnutella

Gnutella è un modello peer-to-peer completamente decentralizzato. Ogni servent[1] è connesso ad altri servent, la struttura della rete che si viene a creare è un grafo sparso del quale non è garantita la connessione. L'insieme di nodi connessi si scambiano messaggi Gnutella su connessioni TCP. I messaggi sono query, risposte a query o altri messaggi di controllo che facilitano l'individuazione di altri nodi. Il protocollo Gnutella è utilizzato dai nodi per trovare altri servent a cui connettersi e per effettuare ricerche tra le risorse condivise. Lo scambio di risorse tra due servent viene invece negoziato con il protocollo HTTP.

1.1 Descrizione dei Messaggi

I Messaggi generati da un nodo vengono inoltrati lungo tutte le connessioni attive. I nodi che ricevono tali messaggi devono rispedirli su tutte le loro connessioni ad esclusione di quella da cui il messaggio è pervenuto. Ogni messaggio è dotato di due contatori Hops e TTL. Ogni nodo decrementa il TTL di ogni messaggio che inoltra e aumenta il numero di Hops. Un messaggio con TTL uguale a zero non viene inoltrato. A causa della natura non centralizzata della rete un pacchetto tende a passare più volte dallo stesso nodo perció è necessaria l'introduzione di un identificativo di messaggio unico nella rete. L'identificativo permette di riconoscere i pacchetti che sono giá stati inoltrati. L'header di un messaggio Gnutella contiene questo identificativo (16 byte il cui valore è random), il TTL, l'Hops, la lunghezza del payload e il Payload Descriptor (1 byte) che indica il tipo di messaggio che segue.

I tipi di messaggi Gnutella sono i seguenti[1]:

- *Ping:* è utilizzato per scoprire servent sulla rete. Un servent che riceve un messaggio di questo tipo è tenuto a rispondere con un messaggio di tipo Pong;

[1] Servent è la contrazione delle parole SERVer e cliENT

- *Pong:* risposta ad un Ping. Contiene l'indirizzo di un servent connesso alla rete Gnutella, la porta sulla quale è in ascolto e il numero e l'ammontare in Kbyte dei file che tale servent condivide;
- *Query:* è il meccanismo principale per la ricerca di risorse sulla rete. Il servent che riceve tale messaggio risponde con un QueryHit per ogni risorsa locale che soddisfa la richiesta;
- *QueryHit:* risposta ad una Query. Contiene le informazioni necessarie per raggiungere la risorsa;
- *Push:* un meccanismo che permette la condivisione di file anche ai servent protetti da firewall;
- *Bye*[2] con questo messaggio il mittente informa l'host remoto che vuole chiudere la connessione.

1.2 Connessione alla rete

Un servent si connette alla rete stabilendo una connessione con un altro servent correntemente in rete. Il servent stabilisce una connessione TCP con l'altro servent e inizia la fase di handshaking. Questa fase prevede lo scambio di informazioni tra il servent client e il servent server, l'header dei messaggi è identica al protocollo HTTP.

Un esempio di handshaking è il seguente:

```
Client                              Server
-------------------------------------------------------------
GNUTELLA CONNECT/0.6<cr><lf>
User-Agent: BearShare<cr><lf>
Query-Routing: 0.2<cr><lf>
Pong-Caching: 0.1
<cr><lf>
                                    GNUTELLA/0.6 200 OK<cr><lf>
                                    User-Agent: BearShare<cr><lf>
                                    Query-Routing: 0.1<cr><lf>
                                    Private-Data: 5ef89a<cr><lf>
                                    Pong-Caching: 0.1
                                    <cr><lf>
GNUTELLA/0.6 200 OK<cr><lf>
Private-Data: a04fce<cr><lf>
<cr><lf>
```

[2]Bye è opzionale e non previsto nella versione 4.0 del protocollo.:

[binary messages] [binary messages]

Esistono altri campi che indicano caratteristiche del servent o che forniscono informazioni su altri punti di accesso alla rete nel caso in cui il server[3] non accetti la connessione (il campo X-Try: contiene l'indirizzo di altri servent che potrebbero accettare una connessione). Il codice ritornato per il rifiuto della connessione è il "503" seguito da "Busy" o un'altra stringa.

Una volta connesso con successo alla rete, un servent comunica con gli altri servent spedendo e ricevendo messaggi Gnutella. Il numero di connessioni Gnutella attive dipende dalla velocitá della connessione ad internet. Molti servent scelgono un numero fisso di connessoni basandosi sulla velocitá della connessione indicata dall'utente. Un metodo migliore puó essere quello di regolare automaticamente il numero di connessioni in modo da utilizzare la giusta banda per il traffico di rete Gnutella. Una tecnica comune prevede di aprire 2 o 3 connessioni in uscita e attendere connessioni in ingresso fino al riempimento dei rimanenti slot.

[3]server è da intendersi come il servent che accetta la connessione

2 Uso dei Messaggi Gnutella

2.1 Messaggi Ping e Pong

Nelle prime versioni del protocollo Gnutella i messaggi di tipo Ping erano inviati in broadcast sulla rete e i messaggi Pong erano inoltrati indietro verso l'origine del messaggio Ping. Dopo aver ricevuto un messaggio Ping il servent lo inoltrava su ogni connessione aperta eccetto quella da cui proveniva. Su quest'ultima veniva inviato un messaggio Pong con le informazioni del servent locale. Questo sistema consuma quasi il 50% della banda di Gnutella[4], tale consumo cresce con l'aumentare dei servent connessi limitando quindi la scalabilità del protocollo. Inoltre, gli indirizzi contenuti nei messaggi Pong spesso non accettano connessioni. Servono quindi tecniche alternative piú efficaci per trattare i messaggi Ping e Pong.

Gli obiettivi di questi nuovi metodi sono i seguenti

- Quando un servent riceve un messaggio Ping deve, se possibile, rispondere con un numero di messaggi Pong (10 è un numero ragionevole). Questi Pong devono avere lo stesso Message ID del Ping ricevuto.
- I Pong spediti devono avere un buona qualità, devono cioè garantire un alta probabilità di connessione e una buona dispersione degli host nella rete;
- La banda usata dai Ping e i Pong deve essere minimizzata.

Per raggiungere questi obiettivi si usano cache di messaggi Pong. Esistono diversi schemi per creare una cache di messaggi, quello che segue è uno dei piú semplici.

Per ogni connessione viene mantenuto un array contenente i messaggi Pong ricevuti. Se si riceve un messaggio Pong e la cache è piena si deve sovrascrivere il piú vecchio tra quelli memorizzati per quella connessione. Per ogni Pong vengono mantenute le seguenti informazioni:

- Indirizzo IP;
- Numero di Porta;
- Numero di file condivisi;
- Quantitá di file condivisi in kbytes;
- Blocco estensione GGEP (se presente);
- Valore dell'Hops.

Quando il servent riceve un messaggio Ping P da una connessione C risponde con un numero di Pong (es. 10) scelti tra quelli memorizzati in ogni connessione eccetto C. In aggiunta, il servent puó aggiungere un Pong con le

informazioni su se stesso ma solo se è in grado di accettare connessioni. I Pong da inviare possono essere scelti tra quelli ricevuti da connessioni diverse e per garantire la dispersione si potrebbe selezionare quelli con Hops diversi. La banda usata con questo schema è limitata. Supponiamo, infatti, che un messaggio Ping sia spedito ogni 3 secondi e che si risponda con 10 Pong ogni Ping. Un Ping (senza estensioni) è grande 23 bytes e un Pong (senza estensioni) è 37 bytes, l'ammontare della banda usata per ogni connessione è data da (23+10*37)/3 = 131 bytes/sec/connessione.

2.2 Query e QueryHint

Un servent inoltra un messaggio Query ricevuto su tutte le connessioni eccetto quella da cui proviene. I servent che usano Flow Control o Ultrapeer non sempre inoltrano tutte le Query su tutte le connessioni. Un servent che riceva un messaggio con lo stesso payload e lo stesso Messagge ID di un messaggio ricevuto precendentemente deve scartalo. I messaggi QueryHit devono essere spediti solo sul percorso dal quale proviene il messaggio Query. In questo modo solo i servent che precedentemente abbiano inoltrato il messaggio Query vedranno il messaggio QueryHit. Un servent che riceva un messaggio QueryHit con Messagge ID = n, ma che non abbia visto un messaggio Query con Messagge ID = n è tenuto a rimuoverlo dalla rete. Un servent puó decidere di scartare alcune query o chiudere una connessione se le query generate dai suoi vicini (Hops=0) sono troppo frequenti, sono duplicate o in altro modo indicano un cattivo comportamento. Quando un servent riceve una Query deve confrontare il Search Criteria con i file che condivide ed eventualmente generare uno o piú QueryHit.

3 Uso del protocollo

3.1 Flow control

Tutti i servent dovrebbero aver un sistema in grado di regolare i dati che passano attraverso una connessione. Il metodo piú semplice è chiudere una connessione se viene generato troppo traffico. Un modo migliore prevede di scartare alcuni pacchetti broadcast per ridurre l'utilizzo di banda. Un metodo ancora migliore prevede l'implementazione di una coda in uscita in ordine FIFO per ogni connessione. Le code contengono tutti i messaggi che devono essere spediti sulla relativa connessione. Se una coda viene riempita per piú del 50% si entra in flow-control mode. Esistono molte implementazioni di Flow Control, una di queste prevede il seguente comportamento. In FC mode tutte le query in ingresso su una connessione sono scartate per evitare di ricevere un gran numero di risultati in risposta. I messaggi che devono essere spediti dal nodo acquisiscono una prioritá. Tale prioritá è stabilita come segue:

- Per messaggi in broadcast, il messaggio con il numero di hops minore ottiente una prioritá maggiore. In questo modo le Query richieste dall'utente verranno inoltrate;
- Tra le risposte vengono privilegiati quei pacchetti che hanno un numero di hops maggiore. Questo vuol dire che si preferisce rispondere a pacchetti provenienti dai servent piú lontani;
- Ogni pacchetto ha comunque una sua prioritá in base al tipo messaggio secondo il seguente ordine: Push, QueryHit, Pong, Query, Ping. Un messaggio di tipo Bye non puó mai essere scartato.

3.2 Struttura della rete

In origine tutti i nodi Gnutella potevano connettersi con qualsiasi altro nodo, questo però era un problema per gli utenti con una banda ridotta. Infatti, gran parte della loro banda era utilizzata per la gestione della connessione alla rete. I servent recenti risolvono il problema organizzando la rete in una forma piú strutturata. Il sistema Ultrapeer si è dimostrato efficace per questo scopo. Lo schema prevede la realizzazione di una struttura gerarchica per i nodi dividendoli in client-node e super-node. Un client-node mantiene un numero ridotto di connessioni aperte con i super-node. Un super-node agisce come un proxy verso la rete Gnutella per i client-node ad esso connessi. I super-node, quindi, filtrano parte del traffico della rete e inviando ai client solo i messaggi a quali loro potrebbero essere interessanti. Ad esempio, un super-node inoltrerá una query verso un leaf-node solo se crede che questo potrá rispondere mentre un leaf-node non inolterá mai le query ricevute da altri servent. Il meccanismo prende il nome di Query Routing Protocol

(QRP) ed è una parte essenziale delle specifiche Ultrapeer. Il QRP riesce a ridurre il traffico verso alcuni nodi rendendo la rete piú scalabile ed efficiente.

3.3 Trasferimento File

Un servent che abbia ricevuto una risposta ad una ricerca puó scaricare la risorsa stabilendo una connessione diretta al nodo che la offre. Il protocollo utilizzato per trasferire il file è il protocollo HTTP.

La richiesta di un file ha la seguente forma:

```
GET /get/<indice del file>/<nome del file>/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: byte=0-\r\n
User-Agent: Gnutella\r\n
\r\n
```

Se il servent ha effettivamente il file richiesto e ha la possibilitá di accettare una nuova connessione risponderá come segue:

```
HTTP/1.0 200 OK\r\n
Server: Gnutella\r\n
Content-Type: application/binary\r\n
Content-length: <lunghezza file>\r\n
\r\n
{ dati dati ... }
```

4 MIB per il monitoraggio di un servent Gnutella

Lo scopo del MIB è quello di monitorare un servent per quello che riguarda le connessioni stabilite con altri servent e i messaggi Gnutella scambiati.

Il MIB è diviso nei seguenti gruppi:

- connection;
- gnet;
- alarm;
- gnetTrap.

4.1 Gruppo connection

Il gruppo connection ha lo scopo di raccogliere informazioni su ogni connessione attiva tra il servent locale e gli altri servent della rete Gnutella. Una nuova entry all'interno della connectionTable viene aggiunta dall'agent quando il servent locale termina la fase di handshaking con un host remoto. L'agent raccoglie da questa fase i valori di tutti gli attributi presenti nella entry. L'unico attributo che puó essere modificato dopo la creazione della entry è connectionStatus. Questo oggetto indica lo stato della connessione e puó assumere quattro valori. Active indica che la connessione è aperta e i due servent si scambiano messaggi. Lo stato passa a Closed quando la connessione viene chiusa da uno dei due servent. Active o Closed sono impostabili dal solo agent. Il Manager puó forzare il cambiamento di stato della connessione da Active a Closed settando il valore dell'oggetto a Close. L'agent deve chiudere la connessione e impostare lo stato a Closed. Il Manager puó settare il valore a Destroy indicando all'agent di eliminare la riga. Se la connessione è ancora aperta viene chiusa. L'attributo connectionStatus ha lo scopo di fornire al manager la capacitá di chiudere connessione non efficienti o dispendiose, oltre ad essere un modo per mantenere le informazioni raccolte anche dopo la chiusura della connessione.

Tra gli altri oggetti del gruppo si hanno connectionInActive e connectionOutActive. Queste variabili indicano il numero di connessioni attive in ingresso e in uscita, se entrambe sono a zero il servent locale non è connesso alla rete.

4.2 Gruppo gnet

Il gruppo gnet mantiene contatori sui messaggi scambiati tra il servent locale e ogni altro servent a cui esso è connesso. La tabella gnetMsgTable ha contatori per ogni tipo di messaggio Gnutella inviato e ricevuto e l'ammontare dei byte ricevuti e inviati su una particolare connessione. L'agent deve inoltre mantenere contatori sul numero totale di byte inviati e ricevuti e sulla quantitá di byte scambiati per messaggi generati dal servent locale.

4.3 Gruppo Alarm

Il gruppo Alarm è preso dal MIB RMONv1 (RFC1571) e del gruppo originale mantiene lo scopo e il funzionamento. Si possono impostare soglie superiori e inferiori, confronto su valore assoluto o differenza, possibilitá di aggiungere soglie. Ho eliminato la generazione dell'evento e la possibilitá di log.

Se un valore supera una soglia viene generata una trap tra fallingAlarm e risingAlarm. L'introduzione del gruppo garantisce la possibilitá di configurare le soglie e le variabili da monitorare tramite trap. In questo modo si possono impostare soglie per qualsiasi variabile del MIB. La tabella rispetto alle consuete soglie scalari è meglio configurabile dal Manager.

4.4 Gruppo Trap

Nel gruppo sono presenti quattro trap. FallingAlarm e RisingAlarm, prese dal MIB RMONv1, sono utilizzate dall'agent per segnalare alla NMS il superamento di una soglia da parte di una variabile. GnetTrapConnection e gnetTrapFlowControl sono utilizzate per notificare due eventi di particolare interesse per il corretto funzionamento del servent locale. I due eventi sarebbero stati comunque impostabili come entry di alarm settando opportune soglie ma, data la loro importanza, ho preferito indicarli come trap separate. La gnetTrapConnection viene generata se uno tra i valori degli oggetti connectionInActive e connectionOutActive passa a 0 o passa da 0 ad altro valore. I valori di questi due oggetti indicano se il servent locale è connesso o meno alla rete. La trap potrebbe essere sostituita dalla seguente entry della alarmTable: alarmSampleType = 1 (absoluteValue), alarmRisingThreshold = 1 e alarmFallingThreshold = 0. La gnetTrapFlowControl viene generata dall'agent se connectionFlowControl di una tra le connessioni presenti nella connectionTable passa da False a True o viceversa. La trap come entry della alarmTable avrebbe i seguenti valori: alarmSampleType = 1 (absoluteValue), alarmRisingThreshold = 1 e alarmFallingThreshold = 0.

5 Lavori Futuri

Nel MIB non c'è la possibilitá di configurare il servent locale e non si hanno informazioni sulle sue caratteristiche. Per mantenere tali informazioni si dovrebbe aggiungere un gruppo contenente oggetti simili a quelli presenti nella connectionTable. La presenza di alcuni oggetti configurabili da remoto permetterebbe di modificare il comportamento del servent. Si potrebbero, ad esempio, aggiungere oggetti per indicare il numero massimo di connessioni in ingresso o in uscita, il numero massimo di download o upload, i nomi delle directory condivise oppure il tipo di servent leaf o super-node.

Il MIB non considera lo scambio di risorse tra servent. Il protocollo Gnutella prevede lo scambio di file con il protocollo HTTP. Il servent è un

client HTTP quando richiede file e un server HTTP quando fornisce risorse e puó quindi essere monitorato vedendolo come un servizio all'interno del MIB WWW Services (RFC 2594).

6 GnutellaMIB

```
GNUTELLA-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    Unsigned32, TimeTicks,
    Counter32, Gauge32, private     FROM SNMPv2-SMI

    DateAndTime, TEXTUAL-CONVENTION,
    TruthValue, DisplayString       FROM SNMPv2-TC

    EntryStatus                     FROM RFC1271-MIB;

-- gnutellaMib

gnutellaMIB MODULE-IDENTITY
    LAST-UPDATED "200309081730Z"
    ORGANIZATION "Universita' di Pisa - SGR2003"
    CONTACT-INFO
        "Rossano Venturini
         Universita' degli Studi di Pisa
         Email:  venturin@cli.di.unipi.it "

    DESCRIPTION
        "Il modulo MIB modella il protocollo Gnutella 0.6 nelle
         parti comuni ai maggiori vendors."
    ::= { private 100 }

-- Textual Conventions

Version ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d-3"
    STATUS    current
    DESCRIPTION
        "Rappresenta la versione del protocollo"
    SYNTAX INTEGER (1..65535)

ConnStatus ::= TEXTUAL-CONVENTION
    STATUS    current
    DESCRIPTION
        "Rappresenta lo stato di una connessione tra due Servent
         Gnutella. I possibili stati di una connessione sono:
         active(0):  i due servent sono attualmente connessi e si
```

```
                    scambiano messaggi Gnutella su tale connessione;
          close(1):   il manager chiede all'agent di chiudere la con-
                      nessione;
          closed(2):  la connessione e' stata chiusa da uno dei due
                      servent anche a causa della richiesta di chiusura
                      da parte del manager;
          destroy(3): l'agent rimuove questa la riga."
      SYNTAX      INTEGER {
                      active(0),
                      close(1),
                      closed(2),
                      destroy(3)
                  }

-- Gruppi definiti

connection      OBJECT IDENTIFIER ::= { gnutellaMIB 1 }
gnet            OBJECT IDENTIFIER ::= { gnutellaMIB 2 }
alarm           OBJECT IDENTIFIER ::= { gnutellaMIB 3 }
gnetTrap        OBJECT IDENTIFIER ::= { gnutellaMIB 4 }

-- connection
-- Il gruppo raccoglie informazioni sulle connessioni tentate e
-- stabilite con altri servent allo scopo di accedere alla rete
-- Gnutella, sono escluse le connessioni con protocollo HTTP per
-- il download/upload di risorse.
-- Con il termine connessione si intende una connessione a livello
-- applicativo quindi questa e' considerata stabilita se viene
-- terminato l'handshaking del protocollo Gnutella.
-- La tabella connectionTable elenca le connessioni attive con altri
-- servent della rete.
-- In questo gruppo sono presenti anche contatori per le connessioni
-- tentate, per quelle riuscite ma terminate e per quelle non andate
-- a buon fine per problemi con protocolli di livello inferiore
-- (trasporto e Ip) o non accettate dal servent.

connectionTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF ConnectionEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "La tabella contiene le connessioni attive con altri servent.
         Sono comprese sia quelle in ingresso che quelle in uscita."
    ::= { connection 1 }
```

```
connectionEntry OBJECT-TYPE
    SYNTAX      ConnectionEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Una riga contiene informazioni su una connessione e sul
         servent remoto."
    INDEX       { connectionIndex }
    ::= { connectionTable 1 }

ConnectionEntry ::= SEQUENCE {
    connectionIndex             Unsigned32,
    connectionSrvAddress        IpAddress,
    connectionSrvPort           INTEGER,
    connectionTime              DateAndTime,
    connectionVendorCode        DisplayString,
    connectionProtocolVers      Version,
    connectionQueryRoutingVers  Version,
    connectionPongCachingVers   Version,
    connectionType              INTEGER,
    connectionServentType       INTEGER,
    connectionStatus            ConnStatus
}

connectionIndex OBJECT-TYPE
    SYNTAX      Unsigned32 (1..4294967295)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Un indice unico per ogni connesione stabilita."
    ::= { connectionEntry 1 }

connectionSrvAddress OBJECT-TYPE
    SYNTAX      IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "L'indirizzo IP del servent remoto con il quale e' stabilita
         la connessione."
    ::= { connectionEntry 2 }

connectionSrvPort OBJECT-TYPE
    SYNTAX      INTEGER (1..65535)
```

```
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "La porta sulla quale e' in ascolto il servent remoto a cui
         siamo connessi."
   ::= { connectionEntry 3 }

connectionTime OBJECT-TYPE
    SYNTAX      DateAndTime
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indica la data e l'ora in cui i due servent hanno terminato
         la fase di handshaking previsto dal protocollo Gnutella.
         Con connessione si indica la connessione a livello applicativo
         non quella TCP."
    ::= { connectionEntry 4 }

connectionVendorCode OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (1..4))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
         "Quattro caratteri case-insensitive rappresentano un vendor
          code.
          Es. BEAR rappresenta il software BearShare in qualunque
          versione."
    ::= { connectionEntry 5 }

connectionProtocolVers OBJECT-TYPE
    SYNTAX      Version
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "La versione del protocollo Gnutella supportata dal servent."
    ::= { connectionEntry 6 }

connectionQueryRoutingVers OBJECT-TYPE
    SYNTAX      Version
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indica la versione del Query Routing Protocol (QRP) supportata
         dal servent remoto.
```

```
          Questo attributo e' necessario per comprendere le statistiche
          sulle Query e le QueryHit ricevute su questa connessione.
          Se la connessione e' outgoing e il servent remoto e' Super-node
          il numero di Query ricevute dovrebbe essere ridotto o comunque
          influenzato dal meccanismo di QRP utilizzato da tale servent.
          Se il QRP non e' supportato o la sua versione non e' nota
          questo attributo ha valore 0.
          Il comportamento tenuto dai nodi leaf e Ultrapeer e le specifiche
          del QRP sono descritte in
          http://www.limewire.com/developer/Ultrapeers.html."
    ::= { connectionEntry 7 }

connectionPongCachingVers OBJECT-TYPE
    SYNTAX      Version
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indica la versione di Pong Caching supportata dal servent
         remoto.
         La versione di Pong Caching del servent remoto influenza la
         quantita' di messaggi Pong ricevuti su questa connessione
         in seguito all'invio di Ping.
         Se la PongCaching non e' supportata o la versione e'
         sconosciuta l'attributo ha valore 0."
    ::= { connectionEntry 8 }

connectionType OBJECT-TYPE
    SYNTAX      INTEGER {
    outgoing(0),
    incoming(1)
}
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indica il tipo di connessione. Se e' outgoing(0) la connes-
         sione e' stata iniziata dal servent locale, incoming(1)
         indica che e' il servent remoto ad averla iniziata."
    ::= { connectionEntry 9 }

connectionServentType OBJECT-TYPE
    SYNTAX      INTEGER {
    superNode(0),
    leafNode(1),
    normal(2)
```

```
}
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indica il tipo di servent a cui siamo connessi secondo il
         sistema Ultrapeer.
         Il significato dei possibili valori e' il seguente:
         superNode(0): il servent remoto accetta connessioni ed e'
                       un nodo attivo nella rete;
         leafNode(1):  e' un nodo che ha a disposizione una ridotta
                       banda non svolge un ruolo attivo nella rete e
                       si connette ai SuperNodi;
         normal(2):    un nodo che non supporta la struttura gerarchica
                       prevista dallo schema Ultrapeer."
    ::= { connectionEntry 10 }

connectionStatus OBJECT-TYPE
    SYNTAX      ConnStatus
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "Indica lo stato della connessione e della relativa riga.
         Il valore puo' essere active(0) se esiste attiva una connes-
         sione tra i due servent, si passa allo stato closed(2) se
         la connessione viene chiusa.
         Il manager puo' settare l'attributo solo a destroy(3) con
         ogni altro valore o a close(1) ma solo se il valore corrente
         e' active(0), negli altri casi l'agent ritorna 'badValue'.
         Lo stato closed(2) viene impostato dall'agent se la connes-
         sione termina.
         Una connessione puo' essere chiusa forzatamente dal manager
         settando l'attributo a close(1).
         destroy(3) e' utilizzato dal manager per eliminare la riga
         se la connessione e' attiva viene chiusa."
    ::= { connectionEntry 11 }

-- "In" indica connessioni in ingresso nelle quali il servent locale
-- nell'handshaking svolge il protocollo lato server. "Out" indica
-- l'inverso.

connectionInActive OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      current
```

```
    DESCRIPTION
        "Il numero di connessioni accettate e attualmente attive."
    ::= { connection 2 }

connectionInRequests OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di richieste di connessioni ricevute."
    ::= { connection 3 }

connectionInAccepted OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di connessioni accettate."
    ::= { connection 4 }

connectionInRefused OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di connessioni in ingresso rifiutate per qualsiasi
         motivo compresi problemi con protocolli di livello inferiore.
         Es. timeout."
    ::= { connection 5 }

connectionInRefusedBusy OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di connessioni in ingresso rifiutate perche' il
         servent locale non ha slot liberi per poter accettare nuove
         connessioni."
    ::= { connection 6 }

connectionOutActive OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      current
```

```
    DESCRIPTION
        "Il numero di connessioni in uscita accettate e attualmente
         attive."
    ::= { connection 7 }

connectionOutRequests OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di tentativi del servent locale di connettersi
         ad altri servent remoti."
    ::= { connection 8 }

connectionOutAccepted OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di connessioni in uscita che sono state accettate
         dai servent remoti."
    ::= { connection 9 }

connectionOutRefused OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di connessioni in uscita che vengono rifiutate
         (o impedite) per qualsiasi motivo compresi problemi con
         protocolli di livello inferiore. Es. timeout."
    ::= { connection 10 }

connectionOutRefusedBusy OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di connessioni in uscita rifiutate perche' il
         servent remoto al quale il sistema ha tentato di connet-
         tersi non ha slot liberi per poter accettare una nuova
         connessione."
    ::= { connection 11 }
```

```
connectionInSndBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "I byte inviati per rispondere a tentativi di connessione
         di servent remoti. Sono esclusi i messaggi di tipo Ping,
         Pong, Query e QueryHit."
    ::= { connection 12 }

connectionInRcvBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "I byte ricevuti da servent che tentano una connessione.
         Sono esclusi i messaggi di tipo Ping, Pong, Query e
         QueryHit."
    ::= { connection 13 }

connectionOutSndBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "I byte inviati per tentare e stabilire connessioni in uscita.
         Sono esclusi messaggi di tipo Ping, Pong, Query e QueryHit."
    ::= { connection 14 }

connectionOutRcvBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "I byte ricevuti come risposta a tentativi di connessione
         in uscita.
         Sono esclusi messaggi di tipo Ping, Pong, Query e QueryHit."
    ::= { connection 15 }


-- gnet
-- Il gruppo gnet contiene statistiche sui messaggi Gnutella inviati e
-- ricevuti. Per ogni connessione attiva vengono mantenuti contatori
-- sul numero di byte e sul numero di messaggi Gnutella inviati,
```

```
-- ricevuti e scartati. Vengono contati il numero totale di byte
-- ricevuti e inviati per messaggi della rete e per  messaggi generati
-- dal servent locale.

gnetMsgTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF GnetMsgEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "La Tabella mantiene contatori per ogni tipo di messaggio
         Gnutella."
    ::= { gnet 1 }

gnetMsgEntry OBJECT-TYPE
    SYNTAX      GnetMsgEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Una riga contiene informazioni sui messaggi, divisi per
         tipo, inviati e ricevuti da una particolare connessione
         attiva.
         Ogni entry ha una corrispondenza 1:1 con una riga presente
         nella connectionTable.
         L'eliminazione di una riga nella connectionTable produce
         l'eliminazione della riga corrispondente in questa tabella."
    INDEX { connectionIndex }
    ::= { gnetMsgTable 1 }

GnetMsgEntry ::= SEQUENCE {
    gnetMsgLastUpdate           TimeTicks,
    gnetMsgRcvPings             Counter32,
    gnetMsgSndPings             Counter32,
    gnetMsgRcvPongs             Counter32,
    gnetMsgSndPongs             Counter32,
    gnetMsgRcvQuery             Counter32,
    gnetMsgSndQuery             Counter32,
    gnetMsgRcvQueryHits         Counter32,
    gnetMsgSndQueryHits         Counter32,
    gnetMsgRcvPushs             Counter32,
    gnetMsgSndPushs             Counter32,
    gnetMsgRcvBytes             Counter32,
    gnetMsgSndBytes             Counter32,
    gnetMsgFlowControl          TruthValue
}
```

```
gnetMsgLastUpdate OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indica il tempo dell'ultima modifica dei dati contenuti
         nella riga."
    ::={  gnetMsgEntry 1  }

gnetMsgRcvPings OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo Ping ricevuti da questa
         connessione.
         La ricezione di un messaggio Ping su una connessione produce
         l'invio di un certo numero di messaggi Pong su questa con-
         nessione ed eventualmente Ping sulle altre connessioni."
    ::={  gnetMsgEntry 2  }

gnetMsgSndPings OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo Ping spediti su questa con-
         nessione."
    ::={  gnetMsgEntry 3  }

gnetMsgRcvPongs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo Pong ricevuti da questa con-
         nessione."
    ::={  gnetMsgEntry 4  }

gnetMsgSndPongs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
```

```
    DESCRIPTION
        "Il numero di messaggi di tipo Pong spediti su questa con-
         nessione."
    ::={  gnetMsgEntry 5  }

gnetMsgRcvQuery OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo Query ricevuti da questa con-
         nessione."
    ::={  gnetMsgEntry 6  }

gnetMsgSndQuery OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo Query spediti su questa con-
         nessione."
    ::={  gnetMsgEntry 7  }

gnetMsgRcvQueryHits OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo QueryHit ricevuti da questa
         connessione."
    ::={  gnetMsgEntry 8  }

gnetMsgSndQueryHits OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo QueryHit spediti su questa
         connessione."
    ::={  gnetMsgEntry 9  }

gnetMsgRcvPushs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
```

```
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo Push ricevuti da questa con-
         nessione."
    ::={  gnetMsgEntry 10  }

gnetMsgSndPushs OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di messaggi di tipo Push spediti su questa con-
         nessione."
    ::={  gnetMsgEntry 11  }

gnetMsgRcvBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di Bytes ricevuti dalla connessione per messaggi
         del protocollo Gnutella."
    ::= { gnetMsgEntry 12 }

gnetMsgSndBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero di Bytes spediti sulla connessione per messaggi
         del protocollo Gnutella."
    ::= { gnetMsgEntry 13 }

gnetMsgFlowControl OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indica se il servent locale, al momento, utilizza un mecca-
         nismo di controllo di flusso per ridurre il numero di messaggi
         Gnutella da spedire su questa connessione."
    ::= { gnetMsgEntry 14 }

msgTotRcvBytes OBJECT-TYPE
```

```
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di Bytes ricevuti per messaggi del proto-
         collo Gnutella.
         Comprende anche i byte ricevuti da connessioni gia' chiuse.
         Sono esclusi i bytes ricevuti per stabilire le connessioni."
    ::= { gnet 2 }

msgTotSndBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di Bytes spediti per messaggi del proto-
         collo Gnutella.
         Comprende anche i byte spediti su connessioni gia' chiuse.
         Sono esclusi i bytes inviati per stabilire le connessioni."
    ::= { gnet 3 }

msgTotRcv OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di messaggi ricevuti."
    ::= { gnet 4 }

msgTotSnd OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di messaggi Gnutella spediti."
    ::= { gnet 5 }

msgRcvDiscards OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di messaggi Gnutella ricevuti ma scartati.
         In genere a causa di duplicazione"
```

```
    ::= { gnet 6 }

msgSndDiscards OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di messaggi Gnutella che dovevano essere
         spediti ma sono stati scartati. La causa puo' essere, ad
         esempio, lo stato di Flow Control mode."
    ::= { gnet 7 }

msgRcvUsBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di Bytes per messaggi del protocollo
         Gnutella ricevuti come risposta a messaggi generati dal
         servent locale."
    ::= { gnet 8 }

msgSndUsBytes OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il numero totale di Bytes spediti per messaggi del proto-
         collo Gnutella generati dal servent locale."
    ::= { gnet 9 }


-- Alarm Group
-- Il gruppo deriva dallo stesso gruppo presente nel MIB RMONv1
-- (RFC1757) ho eliminato i collegamenti con  il gruppo Event e
-- OwerString.
-- Il gruppo Alarm periodicamente confronta il valore delle soglie
-- impostate in una entry con la variabile indicata.
-- Sono impostabili una soglia inferiore e una superiore e il periodo
-- che deve trascorrere tra due confronti.
-- Se il valore di una variabile supera una soglia viene generata
-- una trap.
--
-- Un meccanismo e' utilizzato per limitare il numero di allarmi
```

```
-- generati. Un allarme viene generato  se una variabile supera
-- una soglia nella opportuna  direzione nessun altro allarme viene
-- generato finche la soglia non e' superata nella direzione opposta.

alarmTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF AlarmEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Una lista di alarm entry."
    ::= { alarm 1 }

alarmEntry OBJECT-TYPE
    SYNTAX      AlarmEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "La lista di parametri che devono essere settati un particolare
         allarme."
    INDEX       { alarmIndex }
    ::= { alarmTable 1 }

AlarmEntry ::= SEQUENCE {
    alarmIndex              INTEGER,
    alarmInterval               INTEGER,
    alarmVariable              OBJECT IDENTIFIER,
    alarmSampleType             INTEGER,
    alarmValue                  INTEGER,
    alarmStartupAlarm           INTEGER,
    alarmRisingThreshold        INTEGER,
    alarmFallingThreshold       INTEGER,
    alarmStatus                 EntryStatus
}

alarmIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Un indice che identifica univocamente una entry nella
         tabella alarm."
    ::= { alarmEntry 1 }

alarmInterval OBJECT-TYPE
```

```
    SYNTAX      INTEGER
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "L'intervallo in secondi trascorso il quale i dati vengono
         confrontati con le soglie.
         Quando si setta questa variabile bisogna essere certi  che
         l'intervallo sia abbastanza corto da evitare che la variabile
         aumenti o decresca di un valore maggiore che 2^31 - 1.

         L'oggetto non puo' essere modificato se l'oggetto associato
         alarmStatus e' uguale a valid(1)."
    ::= { alarmEntry 2 }

alarmVariable OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "L'object identifier di una variabile da monitorare.
         I tipi di variabile conforntabili sono solo quelli derivati
         dal tipo ASN.1 INTEGER (INTEGER, Counter, Gauge o TimeTicks).

         L'oggetto non puo' essere modificato se l'oggetto associato
         alarmStatus e' uguale a valid(1)."
    ::= { alarmEntry 3 }

alarmSampleType OBJECT-TYPE
    SYNTAX      INTEGER {
                    absoluteValue(1),
                    deltaValue(2)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "Indica il metodo di confronto utilizzato.
         abosuteValue(1): il valore della variabile selezionata e'
                          confrontato direttamente con le soglie alla
                          fine dell'intervallo;
         deltaValue(2):   il valore della variabile selezionata preso
                          all'intervallo va sottratto con il valore
                          attuale e il risultato confrontato con le
                          soglie.
```

```
        L'oggetto non puo' essere modificato se l'oggetto associato
        alarmStatus e' uguale a valid(1)."
    ::= { alarmEntry 4 }

alarmValue OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Il valore della variabile durante l'ultimo periodo di
        confronto.
        Per esempio, se il tipo di confronto e' deltaValue, questo
        valore sara' la differenza dei valori della variabile
        all'inizio e alla fine del periodo di confronto, se invece
        il confronto e' absoluteValue(1) il valore di questo
        attributo sara' quello della variabile alla fine del
        periodo.
        Questo valore e' confrontato con la soglia superiore e
        inferiore."
    ::= { alarmEntry 5 }

alarmStartupAlarm OBJECT-TYPE
    SYNTAX      INTEGER {
                    risingAlarm(1),
                    fallingAlarm(2),
                    risingOrFallingAlarm(3)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "Un allarme e' spedito se il primo confronto dopo che la
        entry diviene valid appartiene ad uno dei due casi:
        se la variabile e' piu' grande o uguale al valore di rising-
        Threshold e alarmStartupAlarm e' uguale a risingAlarm(1) o
        risingOrFallingAlarm(3) allora sara' generato un singolo
        allarme di tipo rising.
        Se il valore della variabile e' minore o uguale alla
        fallingThreshold e alarmStartupAlarm e' uguale a
        fallingAlarm(2) o a risingOrFallingAlarm(3) allora sara'
        generato un singolo allarme di tipo falling.

        L'oggetto non puo' essere modificato se l'oggetto associato
        alarmStatus e' uguale a valid(1)."
    ::= { alarmEntry 6 }
```

```
alarmRisingThreshold OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "La soglia superiore per il confronto. Quando il valore
         della variabile confrontata e' maggiore o uguale a questa
         soglia e il valore del precedente confronto era minore viene
         generato un allarme.
         Un singolo evento sara' generato anche se il primo confronto
         dopo che la entry diventa valid e' maggiore o uguale alla
         soglia e alarmStartupAlarm e' uguale a risingAlarm(1) o
         risingOrFallingAlarm(3).

         L'oggetto non puo' essere modificato se l'oggetto associato
         alarmStatus e' uguale a valid(1)."
    ::= { alarmEntry 7 }

alarmFallingThreshold OBJECT-TYPE
    SYNTAX      INTEGER
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "La soglia inferiore per il confronto. Quando il valore
         della variabile confrontata e' minore o uguale a questa
         soglia e il valore del precedente confronto e' maggiore,
         viene generato un allarme.
         Un singolo evento sara' generato anche se il primo confronto
         dopo che la entry diventa valid e' maggiore o uguale alla
         soglia e alarmStartupAlarm e' uguale a fallingAlarm(2) o
         risingOrFallingAlarm(3).

         L'oggetto non puo' essere modificato se l'oggetto associato
         alarmStatus e' uguale a valid(1)."
    ::= { alarmEntry 8 }

alarmStatus OBJECT-TYPE
    SYNTAX      EntryStatus
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "Lo stato di questa entry alarm."
    ::= { alarmEntry 9 }
```

```
    -- gnetTrap

    gnetTrapConnection NOTIFICATION-TYPE
        OBJECTS   { connectionOutActive, connectionInActive }
        STATUS      current
        DESCRIPTION
            "La trap viene generata se uno tra i valori di
             connectionOutActive o connectionInActive passa a 0 o passa
             da 0 ad altro valore."
        ::= { gnetTrap 1 }

    gnetTrapFlowControl NOTIFICATION-TYPE
        OBJECTS   { connectionIndex, gnetMsgFlowControl }
        STATUS      current
        DESCRIPTION
            "La trap viene generata se il servent locale entra o esce
             dal Flow Control Mode per una delle connessioni attive.
             L'agent notifica il passaggio dell'attributo
             gnetMsgFlowControl da false a true o viceversa."
        ::= { gnetTrap 2 }

    risingAlarm NOTIFICATION-TYPE
        OBJECTS   { alarmIndex, alarmVariable, alarmSampleType,
                    alarmValue, alarmRisingThreshold }
        STATUS      current
        DESCRIPTION
            "Una trap SNMP e' generata quando un entry alarm supera la
             sua soglia superiore."
        ::= { gnetTrap 3 }

    fallingAlarm NOTIFICATION-TYPE
        OBJECTS     { alarmIndex, alarmVariable, alarmSampleType,
                      alarmValue, alarmFallingThreshold }
        STATUS      current
        DESCRIPTION
            "Una trap SNMP e' generata quando un entry alarm supera la
             sua soglia inferiore."
        ::= { gnetTrap 4 }

END
```

References

[1] T. Klingberg, R. Manfredi *Gnutella 0.6*, Giugno 2002;

[2] *The Gnutella Protocol Specification v0.4.* Document Revision 1.2;

[3] *Gnutella Development Forum (GDF)* http://groups.yahoo.com/group/the_gdf;

[4] J.Ritter *Why gnutella can't scale. No, really*, Febbraio 2001;

[5] Hazewinkel, et al. *Definitions of Managed Objects for WWW Services* RFC 2594;

[6] S. Waldbusser *Remote Network Monitoring Management Information Base* RFC 1757;

[7] McCloghrie, et al. *Structure of Management Information Version 2 (SMIv2)* RFC 2578;