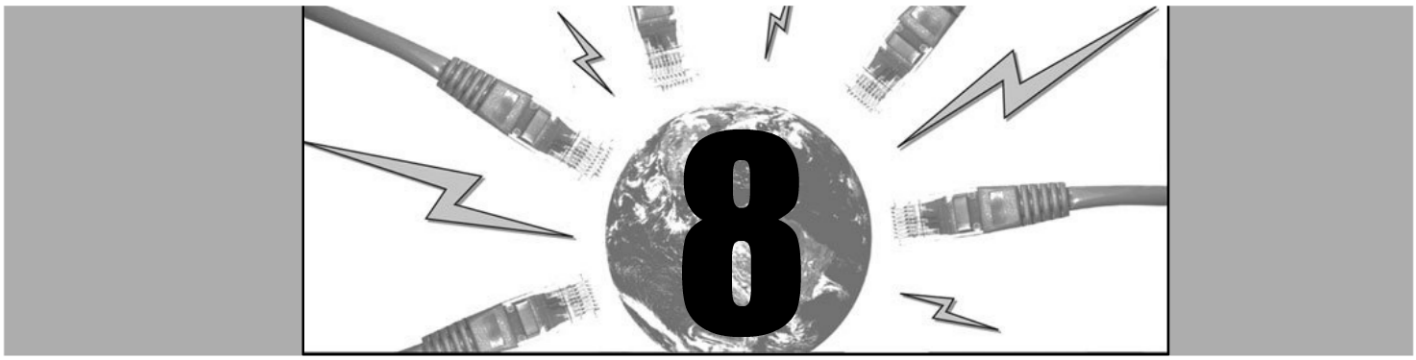

SNMP Management: RMON

OBJECTIVES

- *Remote network monitoring, RMON*
- *RMON1: Monitoring Ethernet LAN and token-ring LAN*
- *RMON2: Monitoring upper protocol layers*
- *Generates and sends statistics close to subnetworks to central NMS*
- *RMON MIBs for RMON group objects*

The success of SNMP management resulted in the prevalence of managed network components in the computer network. SNMPv1 set the foundation for monitoring a network remotely from a centralized network operations center (NOC) and performing fault and configuration management. However, the extent to which network performance could be managed was limited. The characterization of the performance of a computer network is statistical in nature. This led to the logical step of measuring the statistics of important parameters in the network from the NOC and the development of remote monitoring (RMON) specifications.

8.1 WHAT IS REMOTE MONITORING?

We saw examples of SNMP messages going across the network between a manager and an agent in Section 5.1.4. We did this using a tool that "sniffs" every packet that is going across a local area network (LAN), opens it, and analyzes it. It is a passive operation and does nothing to the packets, which continue to proceed to their destinations. This is called monitoring or probing the network and the device that does the function is called the network monitor or the probe. Let us distinguish between the two components of a probe: (1) physical object that is connected to the transmission medium and (2) processor, which analyzes the data. If both are at the same place geographically, it is a local probe, which is how sniffers used to function. We will discuss this further in Chapter 9, when we consider management systems and tools.

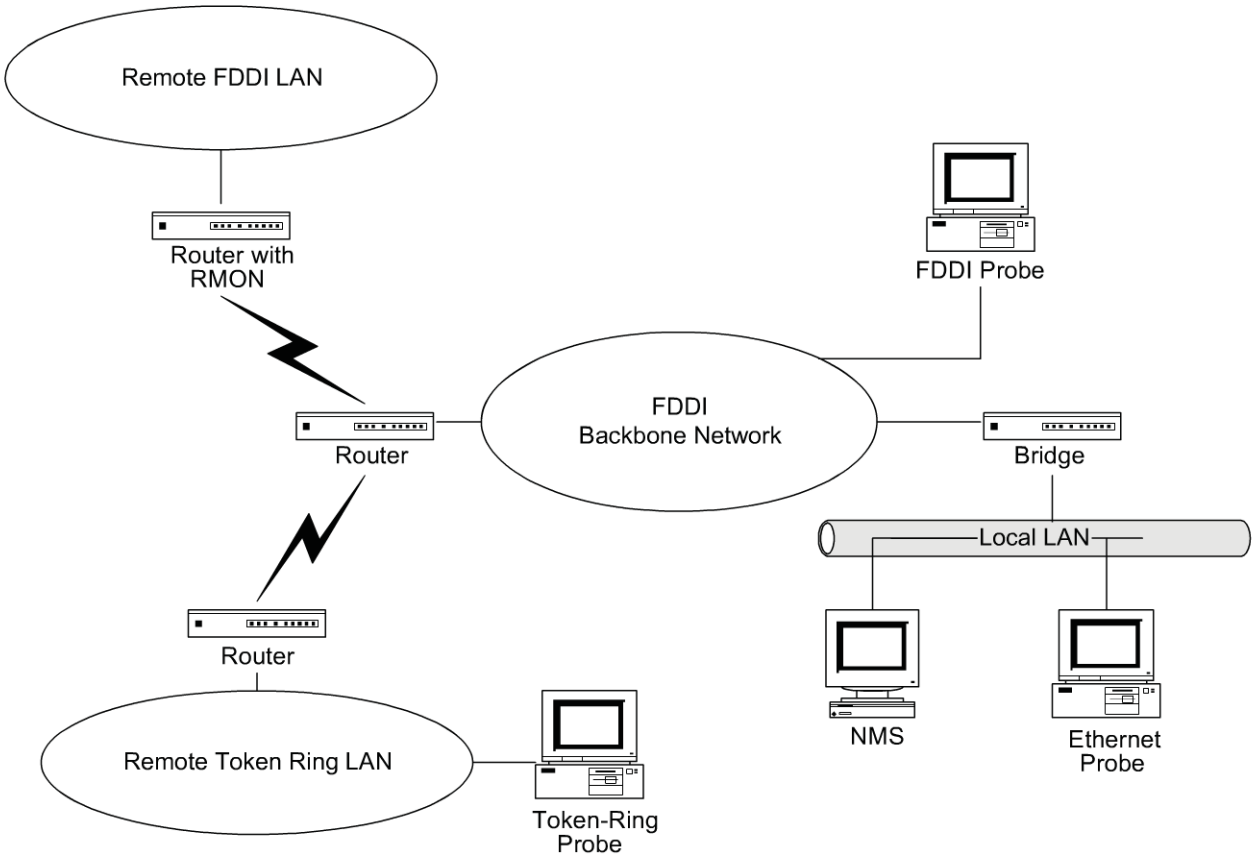

Figure 8.1 Network Configuration with RMONs

The monitored information gathered and analyzed locally can be transmitted to a remote network management station. In such a case, remotely monitoring the network using a probe is referred to as remote network monitoring or RMON. Figure 8.1 shows a fiber-distributed data interface (FDDI) backbone network with a local Ethernet LAN. There are two remote LANs, one a token-ring LAN and another, an FDDI LAN, connected to the backbone network. The network management system (NMS) is on the local Ethernet LAN. There is either an Ethernet probe or an RMON on the Ethernet LAN monitoring the local LAN. The FDDI backbone is monitored by an FDDI probe via the bridge and Ethernet LAN. A token-ring probe monitors the token-ring LAN. It communicates with the NMS via routers and the wide area network (WAN) (shown by the lightening bolt symbol of the telecommunications link). The remote FDDI is monitored by the built-in probe on the router. The FDDI probe communicates with the NMS via the WAN. All four probes that monitor the four LANs and communicate with the NMS are RMON devices.

The use of RMON devices has several advantages. First, each RMON device monitors the local network segment and does the necessary analyses. It relays the necessary information in both solicited and unsolicited fashion to the NMS. For example, RMON could be locally polling network elements in a segment. If it detects an abnormal condition, such as heavy packet loss or excessive collisions, it would send an alarm. Because the polling is local, the information is more reliable. This example of local monitoring and reporting to a remote NMS significantly reduces SNMP traffic in the network. This is especially true for the segment in which the NMS resides, as all the monitoring traffic would otherwise converge there.

The following case history illustrates another advantage. RMON reduces the necessity of agents in the network to be visible at all times to the NMS. One of the NMSs would frequently indicate that one of the hubs would show failure, but the hub recovered itself without any intervention. The performance study of the hub that the LAN was part of indicated that the LAN would frequently become overloaded with heavy traffic, and would have a significant packet loss. That included the ICMP packets that the NMS was using to poll the hub. The NMS was set to indicate a node failure if three successive ICMP packets did not receive responses. Increasing the number of packets needed to indicate a failure stopped the failure indication. This demonstrates the third advantage.

There are more chances that the monitoring packets, such as ICMP pings, may get lost in long-distance communication, especially under heavy traffic conditions. This may wrongly be interpreted by the NMS as the managed object being down. RMON pings locally and hence has less chance of losing packets, thus increasing the reliability of monitoring.

Another advantage of local monitoring using RMON is that individual segments can be monitored on a more continuous basis. This provides better statistics and greater ability for control. Thus, a fault could be diagnosed quicker by the RMON and reported to the NMS. In some situations, a failure could even be prevented by proactive management.

The overall benefits of implementing RMON technology in a network are higher network availability for users and greater productivity for administrators. A study report [CISCO/RMON] indicates increased productivity of several times for network administrators using RMON in their network.

8.2 RMON SMI AND MIB

For a network configuration system, like the one shown in Figure 8.1, to work successfully, several conditions need to be met. Network components are made by different vendors. Even the RMON devices may be from different vendors. Thus, just as in the communication of network management information, standards need to be established for common syntax and semantics for the use of RMON devices. The syntax used is ASN.1. The RMON structure of management information is similar to SMIv2 in defining object types. The Remote Network Monitoring Management Information Base (RMON MIB) defining RMON groups has been developed and defined in three stages. The original RMON MIB, now referred to as RMON1 was developed for Ethernet LAN in November 1991 RFC 1271, but was made obsolete in 1995 RFC 1757. Token-ring extensions to RMON1 were developed in September 1993 [RFC 1513]. The use of RMON1 for remote monitoring was found to be extremely beneficial. However, it addressed parameters at the OSI layer 2 level only. Hence, RMON2 [RFC 2021] was developed and released in January 1997, which addressed the parameters associated with OSI layers 3 through 7.

The RMON group is node 16 under MIB-II (mib-2 16), as shown in Figure 6.36. All the groups under the RMON group are shown in Figure 8.2. It consists of nine Ethernet RMON1 groups (rmon 1 to rmon 9), one token-ring extension group to RMON1 (rmon 10), and nine RMON2 groups (rmon 11–20) for the higher layers.

RMON1 is covered in Section 8.3 and RMON2 in Section 8.4. We will discuss the applications of RMON in Part III when we discuss applications, systems, and tools.

8.3 RMON1

RMON1 is covered by RFC 1757 for Ethernet LAN and RFC 1513. There are two data types introduced as textual conventions, and ten MIB groups (rmon 1 to rmon 10), as shown in Figure 8.2.

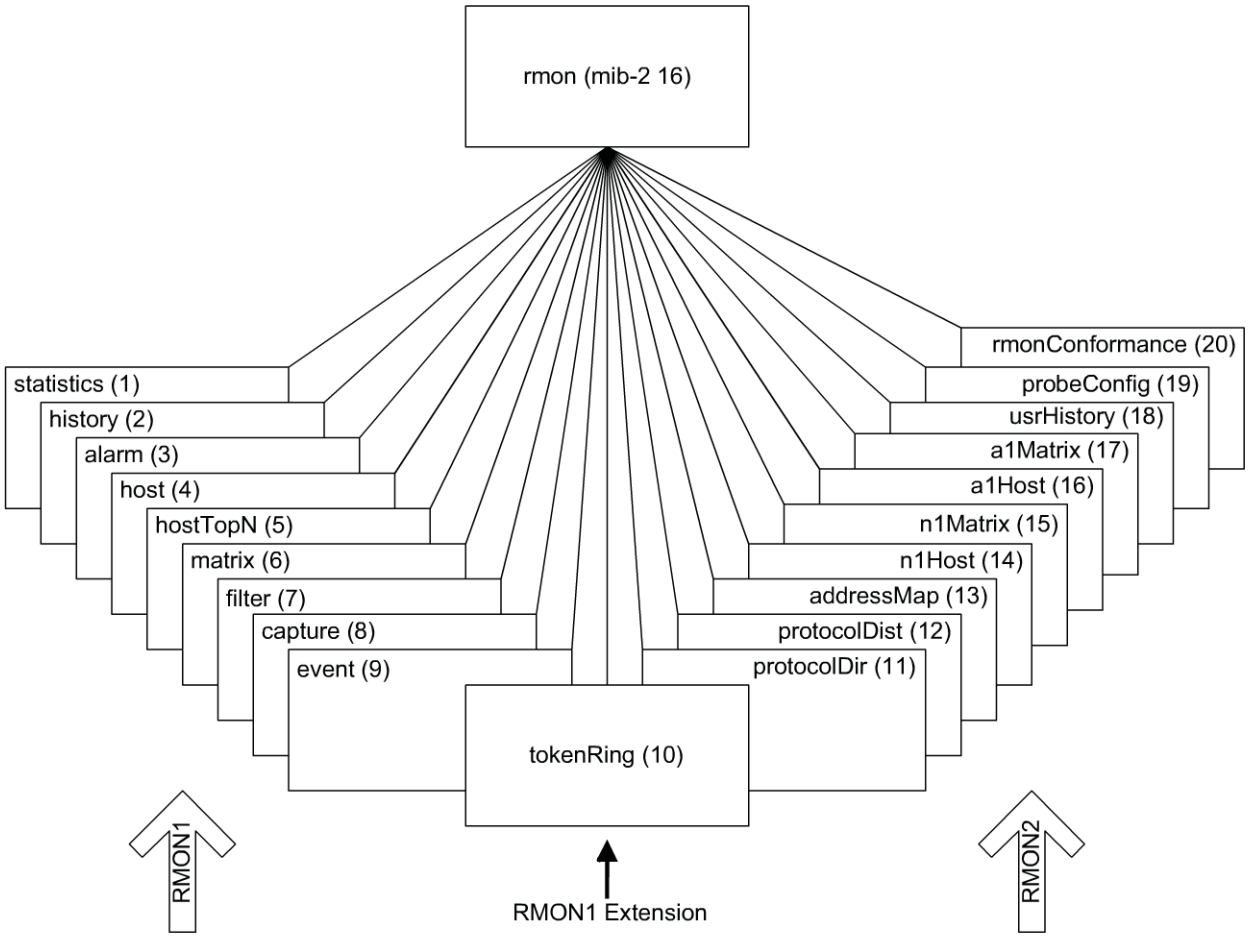

Figure 8.2 RMON Group

8.3.1 RMON1 Textual Conventions

Two new data types that are defined in RMON1 textual conventions are *OwnerString* and *EntryStatus*. Both these data types are extremely useful in the operation of RMON devices. RMON devices are used by management systems to measure and produce statistics on network elements. We will soon see that this involves setting up tables that control parameters to be monitored. Typically, there is more than one management system in the network, which could have permission to create, use, and delete control parameters in a table. Or, a human network manager in charge of network operations does such functions. For this purpose, the owner identification is made part of the control table defined by the *OwnerString* data type. The *EntryStatus* is used to resolve conflicts between management systems in manipulating control tables.

The *OwnerString* is specified in the NVT ASCII character set specified by *DisplayString*. The information content of *OwnerString* contains information about the owner: IP address, management station name, network manager's name, location, or telephone number. If the agent itself is the owner, as for example in the addition of an interface card, the *OwnerString* is set to "monitor."

In order to understand the data type, *EntryStatus*, we need to understand the concept of creation and deletion of rows in tables, which was discussed in Section 6.4.7. For a table to be shared by multiple users, a columnar object *EntryStatus*, similar to *RowStatus* in SNMPv2, is added to the table that

Table 8.1 EntryStatus Textual Convention

STATE	ENUMERATION	DESCRIPTION
valid	1	Row exists and is active. It is fully configured and operational
createRequest	2	Create a new row by creating this object
underCreation	3	Row is not fully active
invalid	4	Delete the row by disassociating the mapping of this entry

contains information on the status of the row. The *EntryStatus* data type can exist in one of four states: (1) *valid*, (2) *createRequest*, (3) *underCreation*, and (4) *invalid*. The four states of *EntryStatus* are shown in Table 8.1. Under the *valid* state condition, the instantiation or row of the table is operational and is probably measuring the number of input octets in the IF group on an interface. Any management system, which is authenticated to use the RMON device, may use this row of data. Of course, if the owner of the row decides to make it invalid, other systems lose the data. The *invalid* state is the way to delete a row. Based on implementation, the row may be immediately deleted and the resource claimed, or it may be done in a batch mode later. If the desired row of information does not already exist, the management system can create a row. The *EntryStatus* is then set to *createRequest*. The process of creation may involve more than one exchange of PDUs between the manager and the agent. In such a situation, the state of the *EntryStatus* is set to *underCreation* so that others won't use it. After the creation process is complete, it is set to the *valid* state.

8.3.2 RMON1 Groups and Functions

RMON in general, and RMON1 specifically, performs numerous functions at the data link layer. Figure 8.3 shows a pictorial representation of RMON1 groups and functions. The data-gathering modules, which are LAN probes, gather data from the remotely monitored network comprising Ethernet and token-ring LANs. The data can serve as inputs to five sets of functions. Three of those comprise monitoring of traffic statistics. The host and conversation statistics group deals with traffic data associated with the hosts, ranking of traffic for the top N hosts, and conversation between hosts. The group of statistical data associated with Ethernet LAN, namely Ethernet statistics and Ethernet history statistics, is addressed by the groups and functions in the Ethernet statistics box. The history control table controls the data to be gathered from various networks. It is also used by the token-ring statistics modules in the token-ring statistics box. Outputs of various modules are analyzed and presented in tabular and graphical forms to the user by the network manager in the NMS.

The filter group is a cascade of two filters. The packet filter filters incoming packets by performing a Boolean and/or XOR with a mask specified. This could be quite complex. The filtered packet stream is considered a channel. We can make further selections based on the channel mask. The filtered outputs may generate either alarms or events. These are reported to the network manager. The output of the data gatherer could also generate an alarm directly.

The output of the filter group could be stored in the packet capture module for further analysis by the network manager. This could be associated with a special study of the traffic pattern or troubleshooting of an abnormality in the network.

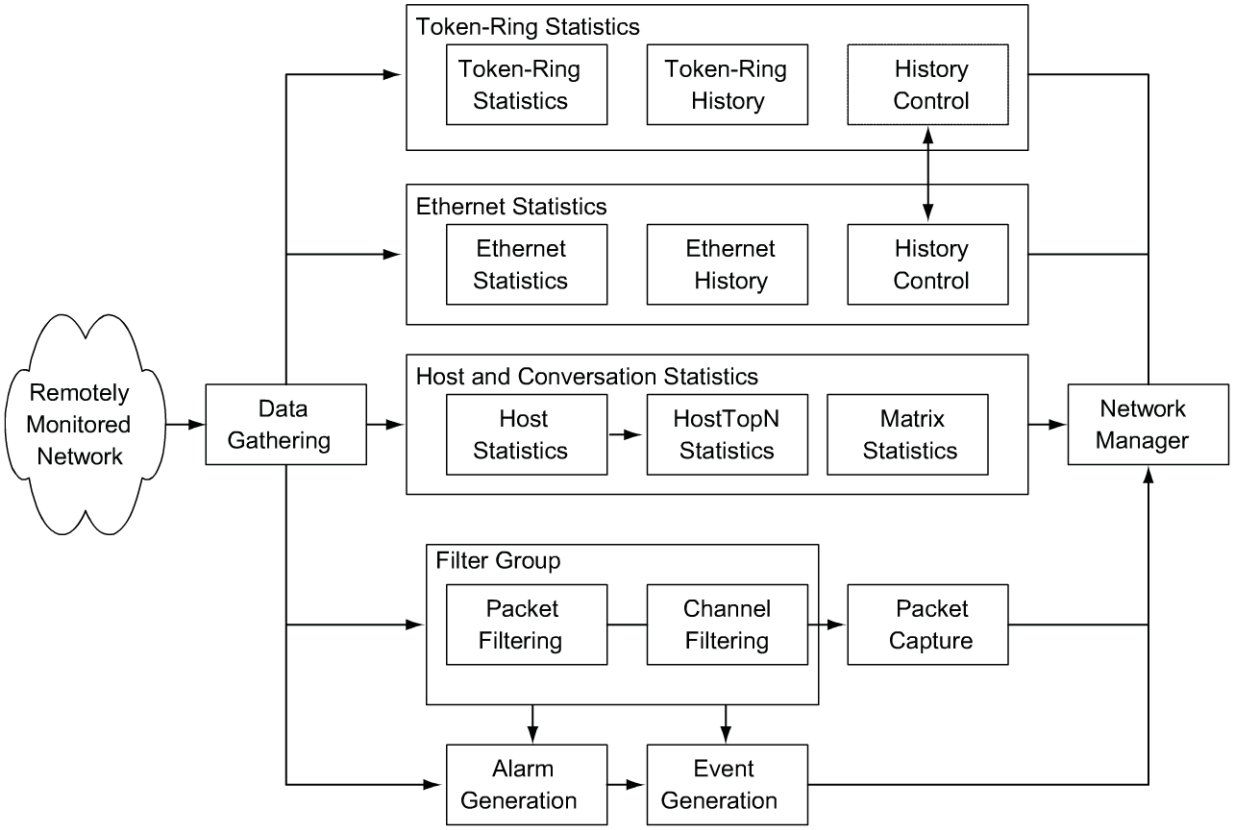

Figure 8.3 RMON1 Groups and Functions

The above functions associated with the various groups are accomplished using ten groups associated with the RMON1 MIB, as shown in Table 8.2. The first nine groups are applicable to common data and to Ethernet LAN, and the tenth group extends it to token-ring LAN. Most of the groups have one or more tables. The groups fall into three categories. The largest category is the statistics-gathering groups. These are the Statistics groups, the History groups, the Host group, the Host Top N group, and the Matrix group. The second category deals with the network event reporting functions. These are the Alarm group and the Event group. The third category deals with filtering the input packets according to selected criteria and capturing the data if desired for further analysis. These are the Filter group and the Packet Capture group. We will consider RMON1 groups and the token-ring extension to RMON1 in Sections 8.3.4 and 8.3.5, respectively.

In Table 8.2, we notice in the Tables column that some of the groups have tables with "2" as part of the name; for example, *etherStats2Table* in the Statistics group. These are additional tables created during RMON2 specifications development and are enhancements to RMON1. Hence, they are included here as part of RMON1. The enhancements to RMON1 include the standard *LastCreateTime* textual convention for all control tables and *TimeFilter* textual convention that provides capability for the filter to handle rows to be used for the index to a table. The *LastCreateTime* enhancement helps keep track of data with the changes in control. The *TimeFilter* enables an application to download only those rows that changed since a particular time. The agent returns a value only if the time mark is less than the last update time.

As an example, let us consider a *fooTable* with two rows and three columnar objects, *fooTimeMark* (with *TimeFilter* as the data type), *fooIndex*, and *foocounts*. The indices defining a row are *fooTimeMark*

Table 8.2 RMON1 MIB Groups and Tables

GROUP	OID	FUNCTION	TABLES
Statistics	rmon 1	Provides link-level statistics	–etherStatsTable
			–etherStats2Table
History	rmon 2	Collects periodic statistical data and stores for later retrieval	–historyControlTable
			–etherHistoryTable
			–historyControl2Table
			–etherHistory2Table
Alarm	rmon 3	Generates events when the data sample gathered crosses pre-established thresholds	–alarmTable
Host	rmon 4	Gathers statistical data on hosts	–hostControlTable
			–hostTable
			–hostTimeTable
			–hostControl2Table
Host Top N	rmon 5	Computes the top N hosts on the respective categories of statistics gathered	–hostTopNcontrolTable
Matrix	rmon 6	Gathers statistics on traffic between pairs of hosts	–matrixControlTable
			–matrixSDTable
			–matrixDSTable
			–matrixControl2Table
Filter	rmon 7	Performs filter function that enables capture of desired parameters	–filterTable
			–channelTable
			–filter2Table
			–channel2Table
Packet capture	rmon 8	Provides packet capture capability to gather packets after they flow through a channel	–buffercontrolTable
			–captureBufferTable
Event	rmon 9	Controls the generation of events and notifications	–eventTable
Token ring	Rmon 10	See Table 8.3	See Table 8.3

and *fooIndex*. Let the *TimeFilter* index start at 0, the last update of *fooCounter* in row #1 occur at time 3, and its value is 5. Assume the update to row #2 occurred at time 5 and the value was updated to 9. This scenario would yield the following instance of *fooCounts* in the *fooTable*:

fooCounts.0.1 5
fooCounts.0.2 9

fooCounts.1.1 5
fooCounts.1.2 9
fooCounts.2.1 5
fooCounts.1.2 9
fooCounts.3.1 5
fooCounts.3.2 9
fooCounts.4.2 9 (Note that row #1 does not exist for times 4 and 5 since the last update occurred at timemark 3.)
fooCounts.5.2 9
(Both rows #1 and #2 do not exist for timemark greater than 5.)

8.3.3 Relationship Between Control and Data Tables

Observing the Tables column in Table 8.2, you will notice several of the groups have a data table and a control table. The data table contains rows (instances) of data. The control table defines the instances of the data rows in the data table and is settable to gather and store different instances of data. The relationship between the control table and the data table is illustrated in a generic manner in Figure 8.4. The value of the *dataIndex* in the data table is the same as the value of *controlIndex* in the control table.

Let us understand how the data table and the control table work together using the matrix group in Table 8.2. We can collect data based on source and destination addresses appearing in the packets on a given interface using the *matrixSDTable* (matrix source–destination table). The control index is an integer uniquely identifying the row in the control table. It would have a value of 1 for the first interface of a managed entity. The value of the columnar object, *controlDataSource*, identifies the source of the data that is being collected. In our example, if the interface #1 belongs to the interfaces group, then *controlDataSource* is *ifIndex.1*.

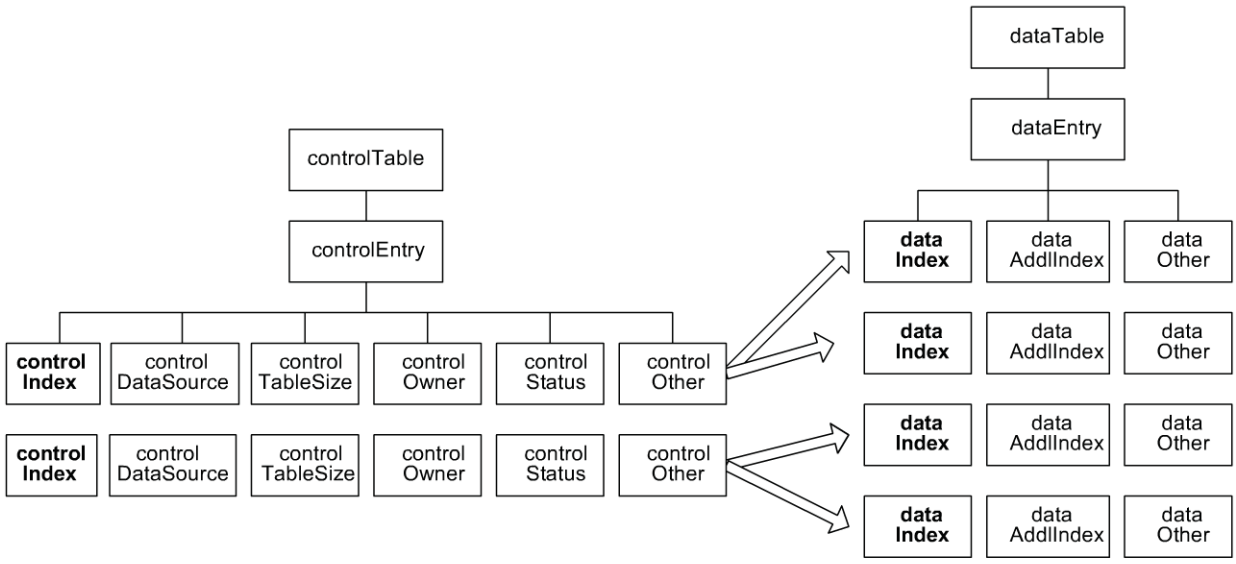

Figure 8.4 Relationship Between Control and Data Tables

The *controlTableSize* identifies entries associated with this data source. In our matrix source–destination table example, this would be the source–destination pair in each row of the table.

The *controlOwner* columnar object is the entity or person who created the entry. The entity could be either the agent or NMS, or a management person. The *controlStatus* is one of the entries listed in Table 8.1. The *controlOther* could be any other object.

To uniquely identify a conceptual row in the data table, we may need to specify more indices than the *dataIndex*. This is indicated as *dataAddlIndex* in Figure 8.4. In our matrix source–destination example, additional indices are source and destination address objects. The *dataOther* in the data table indicates data being collected, such as the number of packets.

8.3.4 RMON1 Common and Ethernet Groups

We have so far covered the global picture of RMON1 Ethernet MIB and how data and control tables are related to each other. Let us now address the nine RMON1 common and Ethernet groups.

Statistics Group. The statistics group contains statistics measured by the probe for each monitored Ethernet interface on a device. The *etherStatsTable* in this group has an entry for each interface. Data include statistics on packet types, size, and errors. It also provides capability to gather statistics on the collision of the Ethernet segment. The number of collisions is a best estimate, as the number of collisions detected depends on where the probe is placed on the segment.

The statistics group is used to measure live statistics on nodes and segments. Commercial NMSs include features such as dynamic presentation of various traffic patterns. The number of MIB collisions could also be used for alarm generation when it exceeds a set high threshold value.

History Group. The history group consists of two subgroups: the history control group and the history (data) group. The history control group controls the periodic statistical sampling of data from various types of networks. The control table stores configuration entries comprising interface, polling period, and other parameters. Information is stored in a media-specific table, the history table, which contains one entry for each specific sample. A short-term and a long-term interval, such as 30-second and 30-minute intervals, may be specified to obtain two different statistics. The data objects defined are dropped events, number of octets and packets, different type of errors, fragments, collisions, and utilization.

The history group is extremely useful in tracking the overall trend in the volume of traffic. Since historical data are accumulated at the data link layer, they include traffic caused by all higher-layer protocols. Short-term history statistics can also be used to troubleshoot network performance problems. For example, in one study of traffic pattern that the author participated in, short-term history statistics revealed that a significant volume of "transparent" data was contributed by servers in the network, which were functioning as "mirrors" for a public news service on the Internet. Although the service was considered to be desirable, since it was generated and consumed externally, it behaved somewhat transparently with regard to the local network traffic.

Alarm Group. The alarm group periodically takes statistical samples on specified variables in the probe and compares them with the pre-configured threshold stored in the probe. Whenever the monitored variable crosses the threshold, an event is generated. To avoid excessive generation of events on the threshold border, rising and falling thresholds are specified. This works in the following manner. Suppose an alarm event is generated when the variable crosses the falling threshold while going down in value. Another event would be generated only after the value crosses the rising threshold at least once.

The group contains an *alarm table* that has a list of entries defining the alarm parameters. The columnar objects *alarmVariable* and *alarmInterval* are used to select the variable and the sampling interval. The sampling type is either the absolute or delta value. In the former, the absolute value of the variable at the end of the previous period is stored as an alarm value. In the latter type, the absolute value at the end of a period is subtracted from the beginning of the period and the computed value is stored. These values are compared with the rising and falling thresholds to generate alarms.

An example of an absolute value would be a new interface card on test for infant mortality. The threshold of the sum of outgoing and incoming packets could be set to 1 gigaoctects and the RMON would generate an alarm/event when the threshold is reached. An example of delta type is threshold set to 10,000 packets in a 10-second interval for excessive packet loss.

Host Group. The host group contains information about the hosts on the network. It compiles the list of hosts by looking at the good packets traversing the network and extracting the source and destination MAC addresses. It maintains statistics on these hosts. There are three tables in the group: *hostControlTable*, *hostTable*, and *hostTimeTable*. The *hostControlTable* controls the interfaces on which data gathering is done. The other two tables depend on this information. The *hostTable* contains statistics about the host. The *hostTimeTable* contains the same data as the host table, but is stored in the time order in which the host entry was discovered. This helps in the fast discovery of new hosts in the system. The entries in the two data tables are synchronized with respect to the host in the *hostControlTable*. We can obtain statistics on a host using this MIB.

Host Top N Group. The host top N group performs a report-generation function for ranking the top N hosts in the category of the selected statistics. For example, we can rank–order the top ten hosts with maximum outgoing traffic. The *HostTopNControlTable* is used to initiate generation of such a report.

As an example of the type of data that can be acquired using an RMON probe, Figure 8.5 shows a chart derived using an RMON probe for the output octets of the top ten hosts in a network. The names of the hosts have been changed to generic host numbers for security reasons.

Matrix Group. The matrix group stores statistics on the conversation between pairs of hosts. An entry is created for each conversation that the probe detects. There are three tables in the group. The *matrixControlTable* controls the information to be gathered. The *matrixSDTable* keeps track of the source to destination conversations; and the *matrixDSTable* keeps data based on destination to source traffic. We can obtain a graph similar to Figure 8.5 for the conversation pairs in both directions using this group.

Filter Group. The filter group is used to filter packets to be captured based on logical expressions. The stream of data based on a logical expression is called a "channel." The group contains a filter table and a channel table. The filter table allows packets to be filtered with an arbitrary filter expression, a set of filters associated with each channel. Each filter is defined by a row in the filter table. A channel may be associated with several rows. For each channel, the input packet is validated against each filter associated with that channel and is accepted if it passes any of the tests. A row in the channel table of the filter group includes the interface ID (same as *ifIndex*) with which the channel is associated, along with acceptance criteria. The combination of the filter and channel filtering provides enormous flexibility to select packets to be captured.

Packet Capture Group. The packet capture group is a post-filter group. It captures packets from each channel based on the filter criteria of packet and channel filters in the filter group. The channel filter criteria for acceptance of the filter group output are controlled by the *bufferControlTable* and

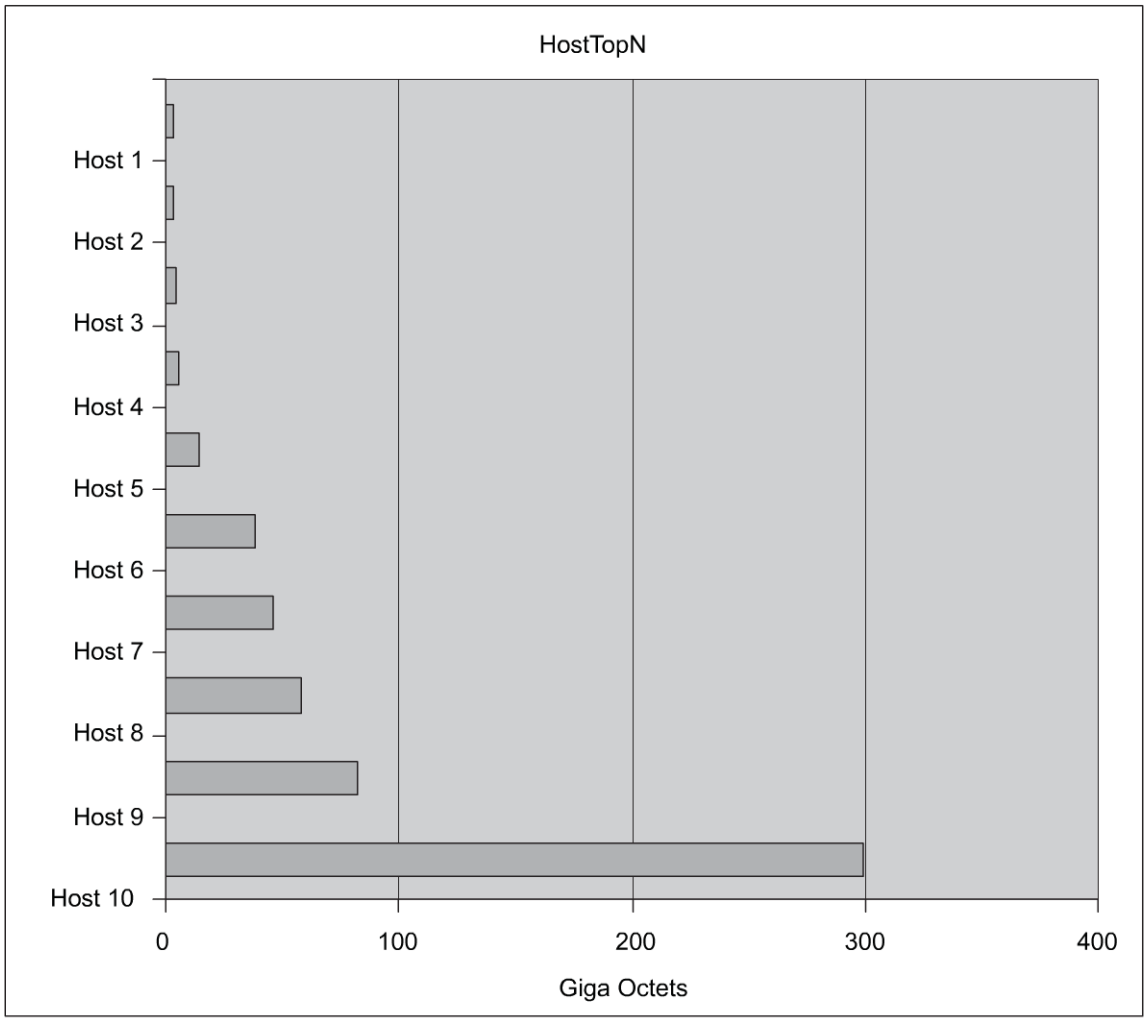

Figure 8.5 HostTop-10 Output Octets

the captured channel data in the *captureBufferTable*. Each packet captured is stored in the buffer as an instance.

Event Group. The event group controls the generation and notification of events. Both the rising alarm and the falling alarm can be specified in the *eventTable* associated with the group. Besides the transmittal of events, a log is maintained in the system.

8.3.5 RMON Token-Ring Extension Groups

As we mentioned earlier, token-ring RMON MIB is an extension to RMON1 MIB and is specified in RFC 1513. Table 8.3 presents the token-ring MIB groups and tables. There are eight groups, each with a data table and two with control tables.

There are two token-ring statistics groups, one at the MAC layer (token-ring statistics group) and a second on packets collected promiscuously (token-ring promiscuous statistics group). They both contain statistics on ring utilization and ring error statistics. The MAC-layer statistics group collects data on token-ring parameters such as token packets, errors in packets, bursts, polling, etc. The promiscuous

Table 8.3 RMON Token-Ring MIB Groups and Tables

TOKEN RING GROUP	FUNCTION	TABLES
Statistics	Current utilization and error statistics of MAC Layer	tokenRingMLStatsTable
		tokenRingMLStats2Table
Promiscuous statistics	Current utilization and error statistics of promiscuous data	tokenRingPStatsTable
		tokenRingPStats2Table
MAC-layer history	Historical utilization and error statistics of MAC layer	tokenRingMLHistoryTable
Promiscuous history	Historical utilization and error statistics of promiscuous data	tokenRingPHistoryTable
Ring station	Station statistics	ringStationControlTable
		ringStationTable
		ringStationControl2Table
Ring station order	Order of the stations	ringStationOrderTable
Ring station configuration	Active configuration of ring stations	ringStationConfigControlTable
		ringStationConfigTable
Source-routing	Utilization statistics of source routing information	sourceRoutingStatsTable
		sourceRoutingStats2Table

statistics group addresses statistics on the number of packets of various sizes and the type of packets as to data—multicast or broadcast. There are two corresponding history statistics groups—current and promiscuous. Each of the four statistics groups has one data table associated with it.

There are three groups associated with the stations on the ring. The ring station group provides statistics on each station being monitored on the ring along with its status. The data are stored in the *ringStationTable*. The rings and parameters to be monitored are controlled by the *ringStationControlTable*. The ring station order group provides the order of the station on the monitored rings and has only a data table. The ring station configuration group manages the stations on the ring.

The last group in the ring groups is the source-routing group. It is used to gather statistics on routing information in a pure source-routing environment.

8.4 RMON2

RMON1 dealt primarily with data associated with the OSI data link layer. The success and popularity of RMON1 led to the development of RMON2. RMON2 [RFC 2021] extends the monitoring capability to the upper layers, from the network layer to the application layer. The term application level is used in the SNMP RMON concept to describe a class of protocols, and not strictly the OSI layer 7 protocol. The error statistics in any layer include all errors below the layer, down to the network layer. For example, the network layer errors do not include data link layer errors, but the transport layer errors include the network layer errors.

Several of the groups and functions in RMON2 at higher layers are similar to that of the data link layer in RMON1. We will discuss the groups and their similarity here. We will cover in detail how protocol analyzer systems incorporate the higher-layer data gathered using RMON2 in Chapter 9 on NMSs and tools.

8.4.1 RMON2 Management Information Base

The architecture of RMON2 is the same as RMON1. RMON2 MIB is arranged into ten groups. Table 8.4 shows the RMON2 MIB groups and tables. We have already discussed enhancements to RMON1 MIB in the previous section.

The protocol directory group is an inventory of the protocols that the probe can monitor. The capability of the probe can be altered by reconfiguring the *protocolDirTable*. The protocols range from the data link control layer to the application layer. This is identified by the columnar object on the unique protocol ID. Each protocol is further subdivided based on parameters, such as fragments. The protocol identifier and protocol parameters are used as indices for the rows of the table. There is one entry in the table for each protocol. The protocols that can be used with the protocol directory have been defined in RFC 2074.

The protocol distribution group provides information on the relative traffic of different protocols either in octets or packets. It collects very basic statistics that would help a NMS manage bandwidth allocation utilized by different protocols. The *protocolDistControlTable* is configured according to the data to be collected and *protocolDistStatsTable* stores the data collected. Each row in the *protocolDist StatsTable* is indexed by the *protocolDistControlIndex* in the *protocolDistControlTable* and *protocol DirLocalIndex* in the *protocolDirTable*. The data table stores the packet and octet counts.

The address map group is similar to the address translation table binding the MAC address to the network address on each interface. It has two tables for control and data.

The network-layer host group measures traffic sent from and to each network address representing each host discovered by the probe, as the host group in RMON1 does.

The network-layer matrix group provides information on the conversation between pairs of hosts in both directions. It is very similar to the matrix tables in RMON1. The group also ranks the top N conversations. It has two control tables and three data tables.

The application layer functions are grouped into two groups, the application-layer host group and the application-layer matrix group. They both calculate traffic by protocol units and use their respective control tables in the network-layer host group and the network-layer matrix group. The application-layer matrix group can also generate a report of the top N protocol conversations.

Alarm and history group information have been combined into the user history collection group in RMON2. This function, normally done by NMSs, can be off-loaded to RMON. It has two control tables and one data table. Data objects are collected in bucket groups. Each bucket group pertains to a MIB object, and the elements in the group are the instances of the MIB object. Users can specify the data to be collected by entering data into *usrHistoryControlTable*, which will then be assembled with rows of instances in the *usrHistoryObjectTable*. Each row in the former specifies the number of buckets to be allocated for each object, and the latter contains rows of instances of the MIB object. The data are stored in *userHistoryTable*. There could be one or more instances of *userHistoryTable* associated with each *usrHistoryObjectTable*.

Table 8.4 RMON2 MIB Groups and Tables

GROUP	OID	FUNCTION	TABLES
Protocol directory	rmon 11	Inventory of protocols	protocolDirTable
Protocol distribution	rmon 12	Relative statistics on octets and packets	protocolDistControlTable
			protocolDistStatsTable
Address map	rmon 13	MAC address to network address on the interfaces	addressMapControlTable
			addressMapTable
Network-layer host	rmon 14	Traffic data from and to each host	n1HostControlTable
			n1HostTable
Network-layer matrix	rmon 15	Traffic data from each pair of hosts	n1MatrixControlTable
			n1MatrixSDTable
			n1MatrixDSTable
			n1MatrixTopNControlTable
			n1MatrixTopNTable
Application-layer host	rmon 16	Traffic data by protocol from and to each host	a1HostTable
Application-layer matrix	rmon 17	Traffic data by protocol between pairs of hosts	a1MatrixSDTable
			a1MatrixDSTable
			a1MatrixTopNControlTable
			a1MatrixTopNTable
User history collection	rmon 18	User-specified historical data on alarms and statistics	usrHistoryControlTable
			usrHistoryObjectTable
			usrHistoryTable
Probe configuration	rmon 19	Configuration of probe parameters	serialConfigTable
			netConfigTable
			trapDestTable
			serialConnectionTable
RMON conformance	rmon 20	RMON2 MIB compliances and compliance groups	See Section 8.4.2

The probe configuration group provides the facility to configure the probe. The data can be accessed using a modem connection. The pertinent data are stored in the *serialConfigTable* and *serialConnectionTable*. The *netConfigTable* contains the network configuration parameters, and the *trapDestTable* defines the destination addresses for the traps.

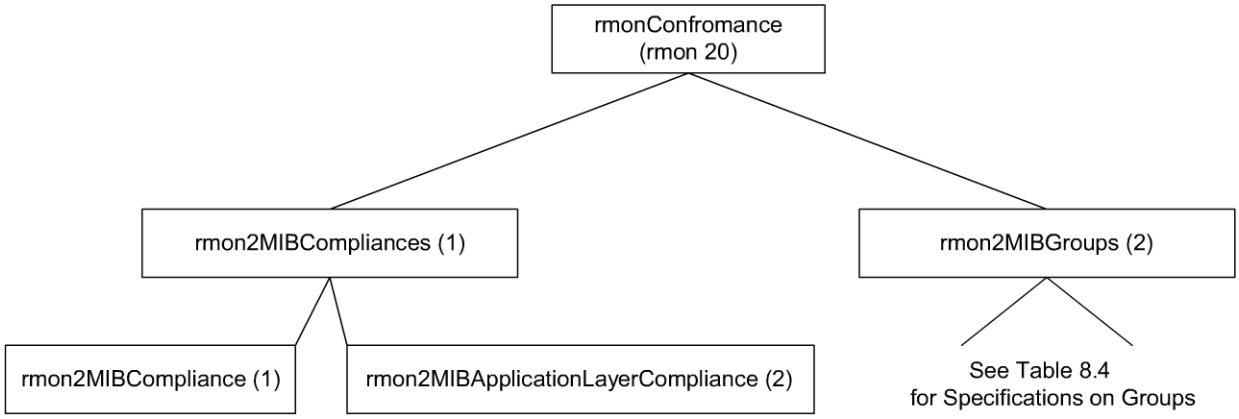

Figure 8.6 RMON2 Conformance Group

8.4.2 RMON2 Conformance Specifications

Conformance specifications were not specified in RMON1. They have been added in RMON2. As shown in Figure 8.6, the RMON2 conformance group consists of two subgroups, rmon2MIBCompliances and rmon2MIBGroups. The compliance requirements are separated into basic RMON2 MIB compliance and application layer RMON2 MIB compliance. Each compliance module defines the mandatory and optional groups. Vendors are required to implement the mandatory groups for compliance; optional groups may be used by vendors to specify additional capabilities.

There are 13 groups in rmon2MIBGroups. They are listed in Table 8.5 along with the mandatory (M) and optional (O) requirements for the basic- and application-level conformance to RMON2. The *rmon1EnhancementGroup* is mandatory for systems that implement RMON1 with RMON2. Notice that *probeConfigurationGroup* is a basic group and hence marked as mandatory, even though it is not specified as such in RFC 2021 definitions. The *rmon1EnhancementGroup* is mandatory for implementation of RMON1. The *rmon1EthernetEnhancementGroup* and *rmon1TokenRingEnhancementGroup* add enhancements to RMON1 that help management stations. The enhancements include filter entry, which provides variable-length offsets into packets and the addition of more statistical parameters.

8.5 ATM REMOTE MONITORING

We will be learning management of ATM in Chapter 9. However, there is a similarity in the use of remote probes for RMON on an ATM network. We will address the commonality and differences here. You may skip this section now, if you so choose, and return to it after you have studied ATM management.

We have thus far learned about RMON and its advantages for gathering statistics on Ethernet and token-ring LANs. RMON1 dealt with the data link layer and RMON2 with higher-level layers. IETF RMON MIBs have been extended to perform traffic monitoring and analysis for ATM networks (see af-nm-test-0080.000 in Table 9.3). Figure 8.7 shows an RMON MIB framework for the extensions, as portrayed by the ATM Forum. Switch extensions for RMON and ATM RMON define RMON objects at the "base" layer, which is the ATM sublayer level. ATM protocol IDs for RMON2 define additional objects needed at the higher-level layers [RFC 2074].

There are several differences between RMON of Ethernet and token ring and monitoring of ATM devices. Extending RMON to ATM requires design changes and new functionality. Particular attention needs to be paid to the following issues: high speed, cell vs. frames, and connection-oriented nature of

Table 8.5 RMON2 Groups and Compliances

OBJECT GROUP	RMON2 MIB	RMON2 MIB APPLICATION LAYER COMPLIANCE
protocolDirectoryGroup	M	M
protocolDistributionGroup	M	M
addressMapGroup	M	M
n1HostGroup	M	M
n1MatrixGroup	M	M
a1HostGroup	N/A	M
a1MatrixGroup	N/A	M
usrHistoryGroup	M	M
probeInformationGroup	M	M
probeConfigurationGroup	M*	M*
rmon1EnhancementGroup	O†	O†
rmon1EthernetEnhancemnetGroup	O	O
rmon1TokenRingEnhancementGroup	O	O

* One of the basic groups in RMON2 and hence is mandatory.

† Mandatory for systems implementing RMON1.

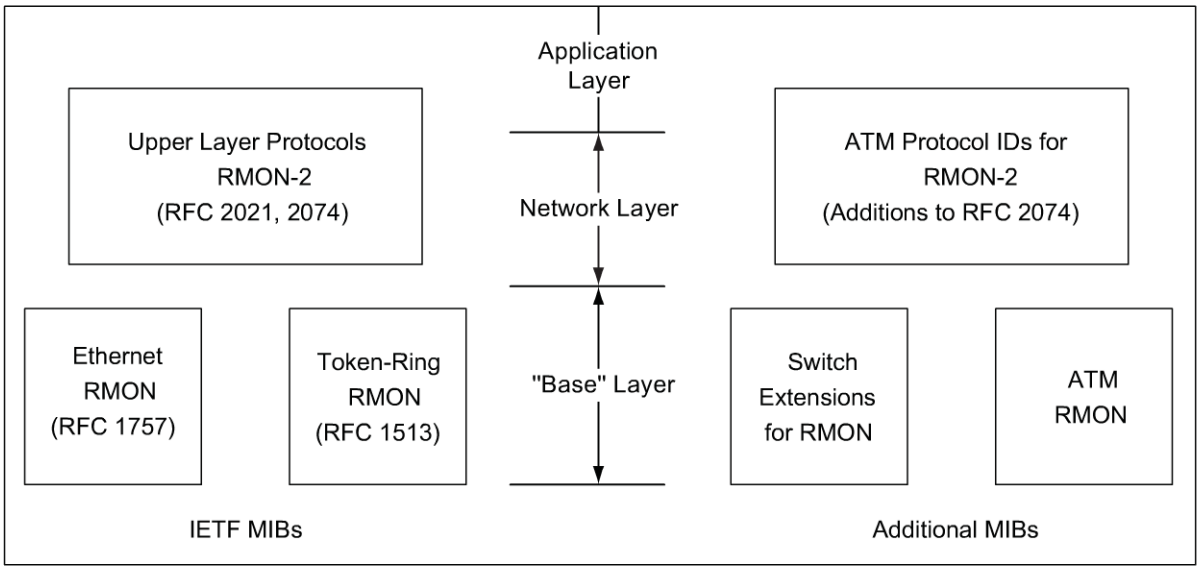

Figure 8.7 RMON MIB Framework

ATM. At the data link sublayer, ATM RMON measures cells instead of packets or frames, and provides cell-based per-host and per-conversation traffic statistics. The high-speed nature of ATM imposes a severe set of requirements in ATM RMON implementation. At the application layer, RMON provides basic statistics for each monitored cell stream, for each ATM host, and for conversation between pair-wise hosts. It also provides capability for flexible configuration mechanisms suited to the connection-oriented nature of ATM.

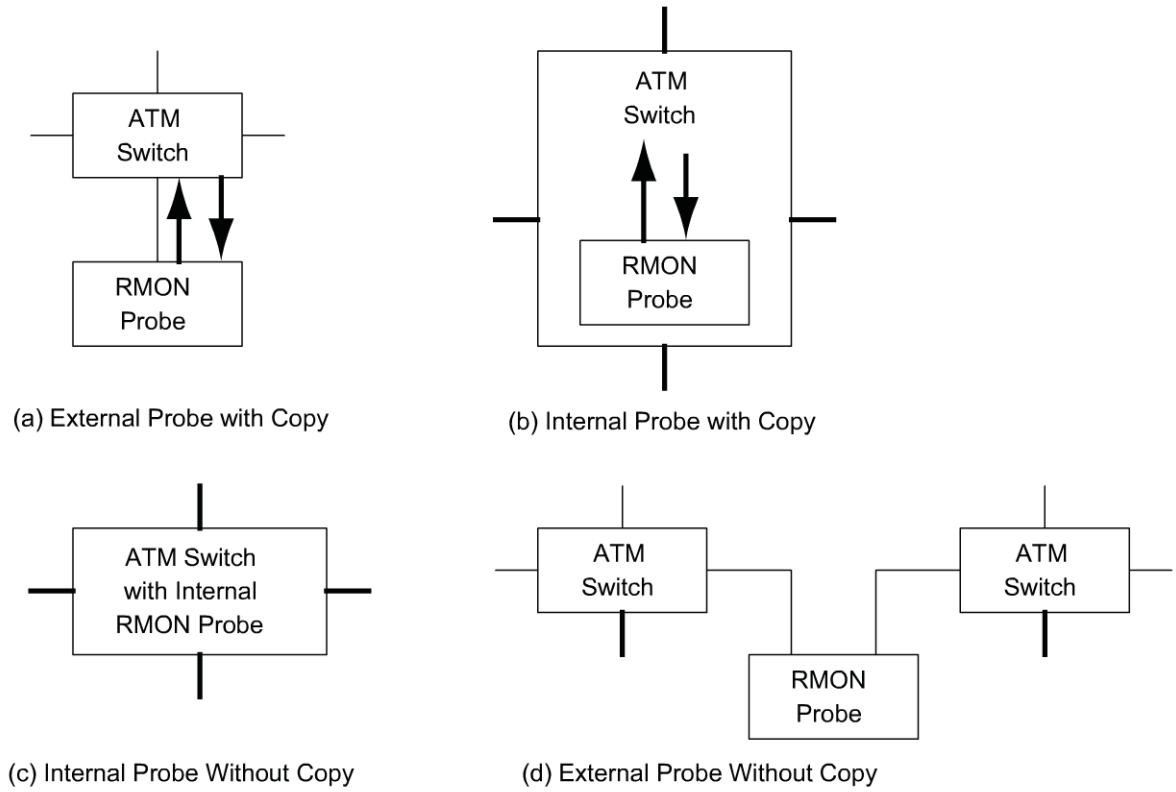

Figure 8.8 ATM Probe Location

There are four different collection perspectives that are possible for ATM RMON, as shown in Figure 8.8. Figure 8.8(a) is a stand-alone probe attached to a single port of a switch. ATM traffic is copied somehow to the RMON probe. Figure 8.8(b) is an embedded probe within a switch, but with no access to the switch fabric. Again, ATM traffic is somehow copied to the RMON probe. Figure 8.8(c) is an embedded probe with access to the switch fabric. However, this type of probe measures traffic at the cell header level only. Figure 8.8(d) is a stand-alone probe, tapping a network-to-network interface between two switches. ATM traffic in both directions is monitored directly without switch intervention. When RMON instrumentation is either embedded into the switch fabric (c) or placed between two switches as in (d), no modification of the circuit is needed. In (a) and (b), circuit steering is needed to copy the cells onto the probe. The two-way arrows in the figures indicate two half-duplex circuits that carry the steered traffic.

The ATM RMON MIB is under the experimental node of the IETF Internet MIB and is shown in Figure 8.9. The functions of the groups and the tables in each group are given in Table 8.6. The MIB contains four groups: *portSelect, atmStats, atmHost,* and *atmMatrix.*

The *portSelect* group addresses port selection. It is used to define the ports to be monitored in a particular statistics, host, or matrix collection. It contains two tables. The *portSelGrpTable* controls the set-up of ports and ATM connection selection criteria used on behalf of any collection associated with entries in this table, such as *atmHostTable*. The *portSelTable* is then used to control the set-up of selection criteria for a single ATM port.

The *atmStats* group collects basic statistics. It counts the total amount of traffic on behalf of one or more *portSelectGroups*. There are two tables in this group: *atmStatsControlTable* and *atmStatsTable*.

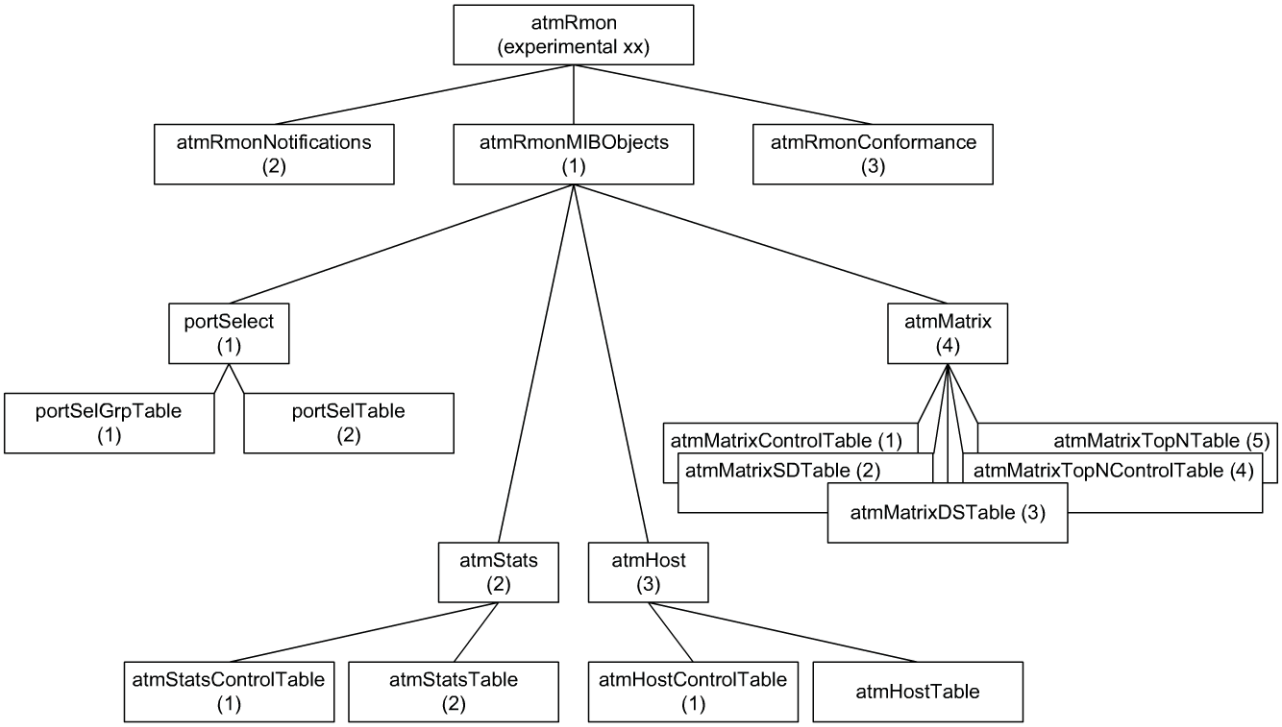

Figure 8.9 ATM RMON MIB

Table 8.6 ATM RMON MIB Groups and Tables

GROUP	OID	FUNCTION	TABLES
portSelect	atmRmonMIBObjects 1	Port selection	portSelGrpTable portSelTable
atmStats	atmRmonMIBObjects 2	Basic statistics	atmStatsControlTable atmStatsTable
atmHost	atmRmonMIBObjects 3	ATM per-host statistics	atmHostControlTable atmHostTable
atmMatrix	atmRmonMIBObjects 4	ATM per-circuit statistics	atmMatrixControlTable atmMatrixSDTable atmMatrixDSTable atmMatrixTopNControlTable atmMatrixTopNTable

The *atmHost* group collects per-host statistics. It counts the amount of traffic sent on behalf of each ATM address discovered by the probe, according to associated *portSelectGroup* criteria. It contains a data table and a control table.

The *atmMatrix* group collects per-circuit statistics and reports the top N circuit traffic. It gathers traffic data on a pair-wise source–destination address, according to *portSelectGroup* criteria, in both directions. It contains three data tables and two control tables. The *atmMatrixControlTable* is used to define the source-to-destination (*atmMatrixSDTable*) and destination-to-source (*atmMatrixDSTable*) traffic. The *atmMatrixTopNControlTable* and *atmMatrixTopNTable* are used to analyze and present the top N traffic carriers.

8.6 A CASE STUDY ON INTERNET TRAFFIC USING RMON

A study was undertaken for planning purposes to gather statistics on the Internet growth at the Georgia Institute of Technology. The technical objectives of the study included traffic growth and trend and traffic pattern. The latter was based on (1) weekly and monthly patterns, (2) diurnal patterns, (3) distribution of traffic by users, packet size, and protocol, and (4) source of traffic based on source–destination data.

The network comprised multiple domains of Ethernet and FDDI LANs. The network complex was connected to the Internet via a high-speed gateway. Data were gathered by measurements made on various domains individually, as well as on the gateway.

Various tools were used to gather data, including RMON statistics. Hewlett-Packard's Netmetrix Protocol Analyzer was used for Ethernet LANs. The statistics were gathered using the host top N and history groups to select the top generators of traffic over the period. The matrix group was used to measure incoming and outgoing traffic. The filter and packet capture groups were beneficial in analyzing the type of traffic based on the application level protocol, such as HTTP, NNTP, etc.

Besides the commercial tools, special tools were developed for the study. For example, the commercial probes were not fast enough to measure the packets traversing an FDDI ring. Hence, a promiscuous mode of counting the packets (function of a probe) was developed to measure traffic on the gateway. We will learn more about management tools and their use in management applications in Chapter 9. However, the case study described here is intended to illustrate the importance of gathering statistics and the use of RMON for that purpose.

A partial summary of the results follows. The names in the results have been changed to protect the privacy and security of the institution.

Results

1. **Growth Rate**: Internet traffic grew at a significant rate from February to June at a monthly rate of 9% to 18%.

February to March	12%
March to April	9%
April to May	18%

Note: There was a sudden drop in June due to end of spring quarter and the beginning of summer quarter.

2. **Traffic Pattern**:
   - Monthly/Weekly: The only discernible variation was lower traffic over weekends.
   - Daily: 2/3 of the top 5% peaks occurred in the afternoon.
   - Users:

Top six domain of users (96%) are

Domain 1	20%
Domain 2	30%
Subdomain 1	(25%)
Subdomain 2	(3%)
Domain 3	34%
Domain 4	7%
Domain 5	3%
Domain 6	2%

Top three hosts sending or receiving data were:

Newsgroups
Mbone
Linux host

What we have learned:

1. The three top groups of users contributing to 84% of the Internet traffic are students (surprise!), Newsgroup services, and Domain 1.
2. The growth rate of Internet use during the study period in the spring quarter was 50%.

Summary

In this chapter we discussed the enhancement to SNMP management by the introduction of remote monitoring, RMON. Remote monitoring is monitoring the network using remotely positioned probes in various segments in the network. RMON1 was initially defined for data link level parameters of Ethernet LAN. It was then extended to token-ring LAN. RMON2 development followed to monitor and produce statistics for parameters associated with the upper layers, from the network to the application level. We will pursue the use of RMON in managing networks from a practical point of view in Part III of this book.

Exercises

1. An NMS connected to a 10-Mbps Ethernet LAN is monitoring a network comprising routers, hubs, and workstations. There are 10,000 nodes in the network to monitor. It sends an SNMP query to each station once a minute and receives a response when the stations are up. Assume that an average frame size is 1,000 bytes long for get-request and response messages.
   (a) What is the maximum traffic load on the LAN that has the NMS?
   (b) Assume that the Ethernet LAN operates at a maximum efficiency of 40% throughput, what is the overhead (SNMP packets/total packets) due to network monitoring?
2. In Exercise 1, assume that the network comprises ten subnetworks, an RMON monitoring each subnet.
   (a) Design a heartbeat monitoring system, using RMONs, that indicates failures to the NMS within a minute of a failure.
   (b) What is the monitoring load on each subnet?
   (c) If the NMS is still expected to detect any failure within 1 minute of occurrence, what is the overhead on the LAN to which the NMS is connected due to this traffic?
3. (a) Describe qualitatively how utilization (number of frames transmitted/number of frames offered) depends on the frame size on an Ethernet LAN.

(b) How would you measure the distribution of the frame size on the LAN?

4. (a) Describe the two methods of measuring collisions on an Ethernet LAN.
   (b) Compare the two methods in terms of what you can measure.
5. Two identical token rings with the same number of stations operate at different efficiencies (ratio of time spent in data transmission to that of the total time). One operates at a higher efficiency than the other. You suspect that it is due to the difference in frame sizes of the data frames in the two rings.
   (a) Why would you suspect the frame size?
   (b) How would you prove your suspicion using RMON?
6. How would you measure the types and distribution of frames in a token-ring LAN?
7. An RMON probe in a network measures Ethernet packets on hub interfaces (*ifIndex*) 1 and 2. The counters are set to zero as the measurement started and interface 1 has counted 1,000 1,500-byte packets and interface 2 has measured 100 64-byte packets. These are stored in rows 1 and 2 of the *protocolDistStatsTable*. They are indexed by the *protocolDistControlIndex* of 1 and 2 and the *protocolDirLocalIndex* of 11 and 12.
   (a) Draw the conceptual rows of the tables involved with the relevant columnar objects and values.
   (b) Write each instance of the columnar object of the data with its associated index and value.