Remote Network Monitoring: Statistics Collection

Chu-Sing Yang

Department of Electrical Engineering
National Cheng Kung University
Outline

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
Introduction

- SNMP standards
  - SMI, MIB and SNMP protocol

- Remote Network Monitoring (RMON)
  - Is the most important addition to the basic set of SNMP standards
  - Is a major step forward in internetwork management
  - Defines a remote monitoring MIB
    - Supplement MIB-II
    - Provides the internetwork manager and information
  - Provides a significant expansion in SNMP functionality
    - Is simply a specification of a MIB
    - No changes in the underlying SNMP protocol
RMON- Related RFCs

- RFC 1513, Sep. 1993
  - Token Ring Extensions to the Remote network Monitoring MIB
- RFC 1757, Feb. 1995
  - Remote Network Monitoring Management Information Base
  - Remote Network Monitoring Management Information Base II
- RFC 2074, Jan. 1997
  - Remote Network Monitoring MIB Protocol Identifiers
Outline

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
Basic Concepts

A LAN has a number of devices with SNMP agent

- An SNMP manager can learn of the amount of traffic into and out of each device from MIB-II
- An SNMP can’t easily learn about the traffic on the LAN as a whole
- **Network monitors** are devices that employ to study the traffic on a network as a whole
Network Monitor

- Are referred to as network analyzer or probe
- Operates on a LAN in “promiscuous” mode
- Views every packets on the LAN
- Produce summary information
  - Error statistics
    - A count of undersized packets
    - The number of collisions
  - Performance statistics
    - The number of packets delivered per second
    - The packet size distribution
- Store (partial) packets for later analysis
  - Uses filters to limit the number of packet counted or captured
    - Packet type
    - Packet characteristics
Network Management

In an internetworked environment

- Need one monitor per subnetwork
  - Standalone device
    - Capture and analyze traffic
  - A device with other functions
    - A workstation, server or router
    - Perform monitoring function
- These monitors need to communicate with a central network management station
  - Refer to as a remote monitor
RMON Specification

- Defines standard network monitoring functions and interfaces for communicating between SNMP-based management consoles and remote monitors.
- Provides an effective and efficient way to monitor subnetwork-wide behavior while reducing the burden both on other agents and on management stations.
Design Goals of RMON

- **Off-line operation**
  - Network manager may be desirable or necessary to limit or halt the routine polling of a monitor
  - Save communication costs
  - Polling may cease if there is a communication failure or if the manager fails
  - The monitor collects fault, performance, and configuration information continuously
  - Attempt to notify the manager if an exceptional event occurs

- **Proactive monitoring**
  - Monitor continuously runs diagnostics and log network performance
    - Be able to notify management station of the failure and provides the management station with information useful in diagnosing the failure

- **Problem detection and reporting**
  - Preemptive monitoring
    - Involves an active probing of the network and the consumption of network resources to check for error and exception conditions
  - Passive monitor (without polling) recognize certain error conditions and other conditions such as congestion on the basis of the traffic that it observes
  - If conditions occur, monitor logs the condition and attempts to notify the manager
Design Goals of RMON (cont’)

- **Value-added data**
  - The network monitor can perform analyses specific to the data collected on this subnetwork, thus relieving the management station of this responsibility
    - To determine which hosts generate the most traffic or errors on the subnetwork

- **Multiple managers**
  - An internetworking configuration may have more than one management station
    - Improves reliability
    - Perform different functions (engineering and operations)
    - Provides management capability to different units within an organization
  - The monitor can be configured to deal with more than one management station concurrently
Remote Monitoring

- An example configuration for remote monitoring
- An internetwork with 5 subnetworks
  - The subnetwork at the top is the central site
    - A dedicated MS with RMON management capability
  - Two Ethernets
    - RMON MIB implemented in a PC
    - Dedicated to remote monitoring
    - Or may perform other duties, such as local NM or a server function
  - FDDI backbone
    - The 2nd MS with RMON management capability
  - Token ring
    - RMON MIB functions performed by the router
Remote Monitoring
A system implements the RMON MIB
Has an agent
Has an **RMON probe process entity**
  - Provides the RMON-related functionality
  - Is capable of reading/writing the local RMON MIB in response to management action and in performing the various RMON-related functions
RMON MIB

- Contained features to support extensive control from management station

  - **Configuration**
    - A remote monitor will be configured to collect data
    - MIB is organized into a number of functional groups
      - Each group may have several control tables and data tables
      - A control table contains parameters that describe the data in a data table
      - Control table: read-write
      - Data table: read-only

  - **Action invocation**
Configuration

- Dictates the type and form of data to be collected
- The management station sets control parameters to configure remote monitors to collect the desired data
  - The parameters are set by adding a new row to the control table or modifying an existing row
  - As information is collected according to the control row, the data are stored in rows of the corresponding data table
  - Functions to be performed by a monitor are defined and implemented in terms of table rows
  - Associate with a single control row are one or more rows in one or more data tables
  - The individual control row and its associated data rows are tied together by **interlocking pointers**
Configuration (cont.)

- To modify any parameters in a control row
  - Invalidate the control entry (row)
    - the deletion of the control row
    - the deletion of all associated rows in data tables
  - Create a new control row with the modified parameters

- When a control row is deleted
  - The associated data tables rows are deleted
  - The resources used by the data tables rows are reclaimed

- One-to-one relationship
  - One data row is related to one control row
  - The control and data tables are combined into a single table
Action Invocation

- Use the SNMP Set operation to issue a command, a process called action invocation
  - An object can be used to represent a command
  - These objects represent states, and the states can be changed by the management station
  - A specific action is taken if the object is set to a specific value (state)
  - A request to set an object to its current value does not cause an action to be performed
Multiple Managers

- An RMON probe may be managed by multiple management stations
- Difficulties for a shared RMON probe
  - Concurrent requests for resources could exceed the capability of the monitor to supply those resources
  - A management station could capture and hold monitor resources for a long period of time
  - Resources could be assigned to a management station that crashes without releasing the resources
Avoidance and resolution
- Add a columnar object to each control table to identify the owner of a particular row of the table and of the associated function

Ownership label
- Contain IP address, management station name, network manager’s name, location, phone number
- Each relevant ownership label is set to a string starting with monitor
- A management station should not alter or delete a monitor-owned function
Multiple Managers (cont.)

- A row of a control table should be altered or deleted only by its owner and read only by other management stations.
- Share the functions already defined by other management stations:
  - Simply observe the corresponding read-only data rows associated with the control row.
  - May be modified or terminated by its owner.
Table Management

- RMON specification
  - include a set of
    - Textual conventions
    - Procedure rules
  - not violate or modify the SNMP framework
  - provide a clear and disciplined technique for row addition and deletion
RFC 1757

No changes to the SMI or SNMP are needed to accommodate the RMON MIB

Two new data types are defined to enhance the readability of the specification

OwnerString ::= DisplayString
- Associated with each read-write table
- 0 to 255 octets
- Used in the creation, modification, and deletion of rows
EntryStatus ::= INTEGER { valid (1),
    createRequest (2),
    underCreation (3),
    invalid (4) }

- Associated with each read-write table
- Give the status of the row containing the object instance
rmiControlTable OBJECT-TYPE
SYNTAX  SEQUENCE OF RMIControlEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION  "A control table."
 ::= { ex 1 1 }

rmiControlEntry OBJECT-TYPE
SYNTAX  RMIControlEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION  "Defines a parameter that controls a set of data table entries."
INDEX  { rmiControlIndex }
 ::= { rmiControlTable 1 }

rmiControlEntry ::= SEQUENCE {
  rmiControlIndex  INTEGER,
  rmiControlParameter  Counter,
  rmiControlOwner  OwnerString,
  rmiControlStatus  RowStatus
}

rmiControlIndex OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION  "The value of this object uniquely identifies this rmiControl entry."
 ::= { rmiControlEntry 1 }

rmiControlParameter OBJECT-TYPE
SYNTAX  Integer
ACCESS  read-write
STATUS  mandatory
DESCRIPTION  "The value of this object characterizes data table rows associated with this entry."
 ::= { rmiControlEntry 2 }

rmiControlOwner OBJECT-TYPE
SYNTAX  OwnerString
ACCESS  read-write
STATUS  mandatory
DESCRIPTION  "The entity that configured this entry."
 ::= { rmiControlEntry 3 }

rmiControlStatus OBJECT-TYPE
SYNTAX  EntryStatus
ACCESS  read-write
STATUS  mandatory
DESCRIPTION  "The status of this rmiControl entry."
 ::= { rmiControlEntry 4 }

rmiDataTable OBJECT-TYPE
SYNTAX  SEQUENCE OF RMICtrlEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION  "A data table."
 ::= { ex 2 1 }

rmiDataEntry OBJECT-TYPE
SYNTAX  RMIDataEntry
ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION  "A single data table entry."
INDEX  { rmiDataControlIndex, rmiDataIndex }
 ::= { rmiDataTable 1 }

RMIDataEntry ::= SEQUENCE {
  rmiDataControlIndex  INTEGER,
  rmiDataIndex  INTEGER,
  rmiDataValue  Counter
}

rmiDataControlIndex OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION  "The control set of which this entry is a part. The control set identified by a value of this index is the same control set identified by the same value of rmiControlIndex."
 ::= { rmiDataEntry 1 }

rmiDataIndex OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
DESCRIPTION  "An index that uniquely identifies a particular entry among all data entries associated with the same rmiControlEntry."
 ::= { rmiDataEntry 2 }

rmiDataValue OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
DESCRIPTION  "The value reported by this entry."
 ::= { rmiDataEntry 3 }

FIGURE 8.2  Control and data tables in RMON1 style
<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
<th>monitor</th>
<th>valid(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>26</td>
<td>manager alpha</td>
<td>valid(1)</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>manager beta</td>
<td>valid(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rm1DataControlIndex</th>
<th>rm1DataIndex</th>
<th>rm1DataValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>96</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>92</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>86</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>26</td>
</tr>
</tbody>
</table>

**FIGURE 8.3** Instance of tables defined in Figure 8.2
Row Addition

- Object instance ID
  \[ \text{Object instance ID} = \text{Object ID} + \text{instance value for the indices} \]

- A `SetRequest` PDU is issued
  - Include a list of columnar object identifiers for the table
    (variable bindings)
  - `SetRequest variable-bindings list` should include all of the columnar objects in the table

- If row addition is not possible
  - `GetResponse` with a `badValue` error status
  - `error-index` field indicates the 1st field in the `variable-bindings list` for which the requested setting was invalid
Row Addition (cont.)

- If two or more mgnt stations attempt to create a row with the same parameters, including index parameters:
  - A state machine built into MIB structure defined by *Status* object
  - RMON Polka

- RMON Polka
  - createRequest (2)
  - underCreation (3)
  - valid (1)
  - An error may be returned for createRequest if the row already exists
If a manager attempts to create a new row, and the index object value or values don’t already exist, the row is created with a status object value of `createRequest (2)`.

Immediately after completing the create operation, the agent sets the status object value to `underCreation (3)`.

Rows shall exist in the `underCreate(3)` state until the manager is finished creating all of the rows that it desires for its configuration. At that point, the manager sets the status object value in each of the created rows to `valid (1)`.

If an attempt is made to create a new row, with a `createRequest` status, and the row already exists an `error` will be returned.
Row Modification and Deletion

- **Row deletion**
  - By setting the status object value for that row to invalid
  - The owner deletes the row by issuing SetRequest PDU

- **Row modification**
  - Invalidates the row first
  - Provides the row with new parameter values

- No transition from the createRequest state to any other state
  - The manager will never find the EntryStatus variable for a row in state createRequest
Transitions of EntryStatus State

FIGURE 8.4 Transitions of EntryStatus state
The RMON MIB

- Incorporated into MIB-II with a subtree Identifier of 16
- Divided into 10 groups
  - Each group is used to store data and statistics derived from data collected by the monitor
  - A monitor may have more than one physical interface
  - All of the groups are optional
- There are some dependencies
  - Alarm group vs. event group
  - hostTopN group vs. host group
  - Packet capture group vs. filter group
FIGURE 8.5  Remote network monitoring MIB
The RMON MIB

- **Statistics**
  - Maintains low-level utilization and error statistics for each subnetwork monitored by the agent

- **History**
  - Records periodic statistical samples from information available in the statistics group

- **Alarm**
  - Allows the manager to set a sampling interval and alarm threshold for any counter or integer recorded by the RMON probe

- **Host**
  - Contains counters for various types of traffic to and from hosts attached to the subnetwork

- **hostTopN**
  - Contains sorted host statistics that report on the hosts that top a list based on some parameter in the host table
The RMON MIB (cont.)

- Matrix
  - Shows error and utilization information in matrix form

- Filter
  - Allows the monitor to observe packets that match a filter

- Packet capture
  - Governs how data is sent to a management console

- Event
  - Gives a table of all events generated by the RMON probe

- tokenRing
  - Maintains statistics and configuration information for token ring subnetwork
The RMON MIB (cont.)

- Dependencies
  - The alarm group requires the implementation of the event group
  - The hostTopN group requires the implementation of the host group
  - The packet capture group requires the implementation of the filter group
  - Collection of traffic statistics for one or more subnetworks
    - Statistics, history, host, hostToN, matrix, and tokenRing
  - Concern with various alarm conditions and packet filtering
    - Alarm, filter, packet capture, event
Outline

- Introduction
- Basic Concepts
- **statistics** Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
Statistics Group

- Contain the basic statistics for each monitored subnetwork
  - A single table
    - One entry (row) for each monitored interface (subnetwork)
    - The statistics are in the form of counters starting from 0

- etherStatsTable
  - Is for Ethernet interfaces
  - Collects a variety of counts for each attached subnetwork
  - Including byte, packet, error, and frame size counts

- Provides information about the load on a subnetwork and the overall health of the subnetwork
  - Various error conditions are counted
    - CRC alignment errors, collisions, undersized and oversized packet
FIGURE 8.6  RMON statistics group
<table>
<thead>
<tr>
<th>Table 8.2  Counters in <code>etherStatsTable</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>etherStatsDropEvents</strong></td>
</tr>
<tr>
<td><strong>etherStatsOctets</strong></td>
</tr>
<tr>
<td><strong>etherStatsPkts</strong></td>
</tr>
<tr>
<td><strong>etherStatsBroadcastPkts</strong></td>
</tr>
<tr>
<td><strong>etherStatsMulticastPkts</strong></td>
</tr>
<tr>
<td><strong>etherStatsCRCAlignErrors</strong></td>
</tr>
<tr>
<td><strong>etherStatsUndersizePkts</strong></td>
</tr>
<tr>
<td><strong>etherStatsOversizePkts</strong></td>
</tr>
<tr>
<td><strong>etherStatsFragments</strong></td>
</tr>
<tr>
<td><strong>etherStatsJabbers</strong></td>
</tr>
<tr>
<td><strong>etherStatsCollisions</strong></td>
</tr>
<tr>
<td><strong>etherStatsPkts640ctets</strong></td>
</tr>
<tr>
<td><strong>etherStatsPkts65to127Octets</strong></td>
</tr>
<tr>
<td><strong>etherStatsPkts128to255Octets</strong></td>
</tr>
<tr>
<td><strong>etherStatsPkts256to511Octets</strong></td>
</tr>
<tr>
<td><strong>etherStatsPkts512to1023Octets</strong></td>
</tr>
<tr>
<td><strong>etherStatsPkts1024to1518Octets</strong></td>
</tr>
</tbody>
</table>
Two of the noncounter objects in the group warrant further explanation

- etherStatsIndex
  - Is an integer index for this row
  - One row is defined for each monitored Ethernet interface

- etherStatsDataSource
  - Identify the interface that is the source of the data in this row
  - The value of this object instance is an object ID that identifies the instance of ifIndex in the interface group of MIB-II
  - ifIndex.1

Compared to EtherLike MIB group

- The Ethernet interface MIB
- dot3Stats table collects statistics for a single system on an Ethernet
Statistics Group (cont.)

- Compared to EtherLike MIB group
  - dot3Stats table collects statistics for a single system on an Internet
  - etherStatsTable collects statistics for all systems attached to an ethernet

- The functions of control table and data table are combined
  - Only read-write objects in the table are
    - etherStatsDataSource, etherStatsOwner, etherStatsStatus
Outline

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
History Group

- Defines sampling functions for one or more of the interfaces of the monitor

historyControlTable
  - Specifies the interface and the details of the sampling function
  - Each row defines a set of samples at a particular sampling interval for a particular interface
  - Each sample is stored in a new row of etherHistoryTable

historyControlRowIndex
  - An integer uniquely identifies a row

historyControlDataSource
  - Identifies the interface that is the source of the data for samples
historyControlTable (cont.)

- historyControlBucketRequested
  - The request number of discrete sampling intervals over which data are to be saved in the part of the data table
  - Default value is 50

- historyControlBucketGranted
  - The actual # discrete sampling intervals over which data will be saved
  - Circular buffer

- historyControlInterval
  - The interval over which data are sampled for each bucket
  - The interval can be set to any number between 1 and 3600 seconds, default = 1800
  - Monitor takes a sample once every 1,800 seconds by default
    - Each sample is stored in a row of etherHistoryTable, and the most recent 50 rows are retained
etherHistoryTable

- Record the data
- A media-specific table for Ethernet
- etherHistoryIndex
  - The history of which this entry is a part
- etherHistorySampleIndex
  - An index that uniquely identifies the particular sample this entry
  - Starts at 1 and increases by one
- etherHistoryIntervalStart
  - The value of sysUpTime at the start of the interval
- etherHistoryUtilization
  - INTERGER object
  - Packets = ΔetherStatsPkts, Octets = Δ etherStatsOctets
  - \{ [Packets*(96+64) + Octets*8] / [etherHistoryControlInterval*bandwidth] \} *100%
etherHistoryTable (cont.)
  - etherHistoryPkts
    - \(\Delta\) etherStatPkts
History Group (cont.)

- No two rows have the same combination of values of `historyControlDataSource` and `historyControlInterval`
  - A history is unique in terms of the interface and the sampling interval
- Specification recommends at least 2 history control entities per monitored interface
  - 30-second sampling rates
    - Detect sudden changes in traffic patterns
  - 30-minute sampling rates
    - Detect the steady-state behavior of the interface
- Bucket
  - Each row of `etherHistoryTable`
  - Holds the statistics gathered during one sampling interval
  - `etherHistoryPkts`
    - Is a counter equal to # packets (including error packets) received during the corresponding sampling interval
FIGURE 8.7  RMON history group
### historyControlTable

<table>
<thead>
<tr>
<th>historyControl-Index</th>
<th>historyControl-Datasource</th>
<th>historyControl-BucketsGranted</th>
<th>historyControl-Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$D_1$</td>
<td>$B_1$</td>
<td>$I_1$</td>
</tr>
<tr>
<td>2</td>
<td>$D_2$</td>
<td>$B_2$</td>
<td>$I_2$</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>$K$</td>
<td>$D_K$</td>
<td>$B_K$</td>
<td>$I_K$</td>
</tr>
</tbody>
</table>

### etherHistoryTable

<table>
<thead>
<tr>
<th>etherHistoryIndex</th>
<th>etherHistorySampleIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x + 1$</td>
</tr>
<tr>
<td>1</td>
<td>$x + 2$</td>
</tr>
<tr>
<td>1</td>
<td>$x + 3$</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>$x + B_1$</td>
</tr>
<tr>
<td>2</td>
<td>$y + 1$</td>
</tr>
<tr>
<td>2</td>
<td>$y + 2$</td>
</tr>
<tr>
<td>2</td>
<td>$y + 3$</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>$y + B_2$</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
Host Group

- Used to gather statistics about specific hosts on the LAN
  - Monitor learns of new hosts on the LAN by observing source/dest. MAC addresses in good packets
  - For each host known to the monitor, a set of statistics is maintained

- Host group
  - One control table
    - Determine for which interface this function is performed
    - Each interface is specified by a row in hostControlTable
  - Two data tables
Host Group (cont.)

- **hostControlTable**
  - **hostControlIndex**
    - An integer that uniquely identifies a row in the hostControlTable
  - **hostControlDataSource**
    - The identifies the interface and hence the subnetwork that is the source of the data for data-table entries
  - **hostControlTableSize**
    - The # rows in hostTable
    - The # hosts that are within the subnetwork
  - **hostControlLastDeleteTime**
    - The value of sysUpTime corresponding to the last time that an entry was deleted from the portion of hostTable associated to this row
Host Group (cont.)

- hostTable
  - Contain one row for each MAC address discovered on an interface specified in a row of hostControlTable,
  - Contain statistics data about the corresponding host
  - No. of rows in hostTable = SUM (value of hostControlTableSize) for each row of hostControlTable
  - Ordered by MAC address
  - Circular buffer
  - hostAddress
  - hostCreationOrder
  - hostIndex
Host Group (cont.)

- hostTimeTable
  - Indexed by the creation order
  - Table size is known, and it is easy to use GetRequest or GetNextRequest PDU to pack variables efficiently
  - Support efficient discovery by management station of new entries for a particular interface, without having to download the entire table

- It is not necessary to implement two separate tables with duplicate information
  - Logical access to the data
FIGURE 8.10 A simple RMON configuration
<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostInPkts</td>
<td>Number of good packets transmitted to this address</td>
</tr>
<tr>
<td>hostOutPkts</td>
<td>Number of packets, including bad packets, transmitted by this address</td>
</tr>
<tr>
<td>hostInOctets</td>
<td>Number of octets transmitted to this address, not including octets in bad packets</td>
</tr>
<tr>
<td>hostOutOctets</td>
<td>Number of octets transmitted by this address, including octets in bad packets</td>
</tr>
<tr>
<td>hostOutErrors</td>
<td>Number of bad packets transmitted by this address</td>
</tr>
<tr>
<td>hostOutBroadcastPkts</td>
<td>Number of good broadcast packets transmitted by this address</td>
</tr>
<tr>
<td>hostOutMulticastPkts</td>
<td>Number of good multicast packets transmitted by this address</td>
</tr>
<tr>
<td>hostControlIndex</td>
<td>hostControlDataSource</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>1</td>
<td>D₁</td>
</tr>
<tr>
<td>2</td>
<td>D₂</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>K</td>
<td>Dₖ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hostTable</th>
<th>hostTimeTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>host-Address</td>
<td>hostTime-Address</td>
</tr>
<tr>
<td>host-CreationOrder</td>
<td>hostTime-CreationOrder</td>
</tr>
<tr>
<td>host-Index</td>
<td>hostTime-Index</td>
</tr>
</tbody>
</table>

| M(1,1)           | C(1,1)                | 1                     |
| M(1,2)           | C(1,2)                | 1                     |
| M(1,3)           | C(1,3)                | 1                     |
| ...              | ...                   | ...                   |
| M(1,N₁)          | C(1,N₁)               | 1                     |
| M(2,1)           | C(2,1)                | 2                     |
| M(2,2)           | C(2,2)                | 2                     |
| M(2,3)           | C(2,3)                | 2                     |
| ...              | ...                   | ...                   |
| M(2,N₂)          | C(2,N₂)               | 2                     |
| ...              | ...                   | ...                   |

M(i,j) < M(i,j + 1)  
C(i,j) for j = 1, Ni = permutation  
on integers from 1 to Ni

MT(i,j) for j = 1, Ni = permutation  
on M(i,j) for j = 1, Ni

Note: Only selected fields in each table are shown.

**FIGURE 8.11**  An example of host tables
 Lecture Title, Date, Location

Outline:

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
HostTopN Group

- Used to maintain statistics about the set of hosts on one subnetwork that top a list based on some parameter
  - Derived from data in the host group

- Report
  - The set of statistics for one host group object on one interface, collected during one sampling interval
  - Contains the results for only one variable
    - Represents the amount of change in a host group object over the sampling interval
**hostTopNControlTable**

- **hostTopNControlIndex**
  - An integer that uniquely identifies a row in the host-TopNControlTable

- **hostTopNHostIndex**
  - Matches with the value of hostControlIndex and hostIndex

- **hostTopNRateBase**
  - Specifies one of seven variables from hostTable
    - hostInPkts, hostOutPkts, hostInOctets, hostOutOctets, hostOutErrors, hostoutBroadPkts, hostoutMulicastPkts
hostTopNControlTable

- **hostTopNTimeRemaining**
  - Tells the number of seconds left in the sampling interval for the report currently being collected

- **hostTopNDuration**
  - The sampling interval for this report

- **hostTopNRequestedSize**
  - The maximum number of hosts requested for the top-N table for this report

- **hostTopNGrantedSize**
  - Indicates the maximum number of hosts in the topN table for this report

- **hostTopNStartTime**
  - The value of sysUpTime when the top-N report was last started
hostTopNTable

- hostTopNReport
  - The report of which this entry is a part

- hostTopNIndex
  - Uniquely identifies one row among all data rows associated with this report
  - Each row represents a unique host

- hostTopNAddress
  - Gives the MAC address of this host

- hostTopNRate
  - The amount of change in the selected variable during this sampling interval
hostTopN (rmon 5)

hostTopNControlTable (1)

hostTopNControlEntry (1)

- hostTopNControlIndex (1)
- hostTopNHostIndex (2)
- hostTopNRateBase (3)
- hostTopNTimeRemaining (4)
- hostTopNDuration (5)
- hostTopNRequestedSize (6)
- hostTopNGrantedSize (7)
- hostTopNStartTime (8)
- hostTopNOwner (9)
- hostTopNStatus (10)

hostTopNTable (2)

hostTopNEntry (1)

- hostTopNReport (1)
- hostTopNIndex (2)
- hostTopNAddress (3)
- hostTopNRate (4)
### hostTopNControlTable

<table>
<thead>
<tr>
<th>hostTopNIndex</th>
<th>hostTopNHostIndex</th>
<th>hostTopNRateBase</th>
<th>hostTopNGrantedSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H₁</td>
<td>V₁</td>
<td>N₁</td>
</tr>
<tr>
<td>2</td>
<td>H₂</td>
<td>V₂</td>
<td>N₂</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>K</td>
<td>H_K</td>
<td>V_K</td>
<td>N_K</td>
</tr>
</tbody>
</table>

### hostTopNTable

<table>
<thead>
<tr>
<th>hostTopNReport</th>
<th>hostTopNIndex</th>
<th>hostTopNAddress</th>
<th>hostTopNRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>M(1,1)</td>
<td>V₁(1)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>M(1,2)</td>
<td>V₁(2)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>M(1,3)</td>
<td>V₁(3)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>N₁</td>
<td>M(1,N₁)</td>
<td>V₁(N₁)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>M(2,1)</td>
<td>V₂(1)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>M(2,2)</td>
<td>V₂(2)</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>M(2,3)</td>
<td>V₂(3)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>N₂</td>
<td>M(2,N₂)</td>
<td>V₂(N₂)</td>
</tr>
</tbody>
</table>

\[ V₁(j) > V₁(j + 1) \]

Note: Only selected fields in the control table are shown.

**FIGURE 8.13** An example of hostTopN tables

The amount of change in the selected variable during this sampling interval.
Outline

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
Matrix Group

- Used to record information about the traffic between pairs of hosts on a subnetwork
  - Information is stored in the form of a matrix
  - Is useful for retrieving specific pairwise traffic information
    - Finds out which devices are making the most use of a server
  - Consists of 3 tables
    - One control table
      - matrixControlTable
    - Two data tables
      - matrixSDTable, matrixDSTable
Is used to store statistics on traffic from a particular Source host to a number of destinations

Source MAC
Destination MAC
matrixControlTable

- matrixControlIndex
  - An integer that uniquely identifies a row in the table

- matrixControlDataSource
  - Identifies the interface and hence the subnetwork that is the source of the data in this row

- matrixControlTableSize
  - The number of rows in the matrixSDTable (matrixDSTable) that are associated with this row

- matrixControlLastDeleteTime
  - The value of sysUpTime corresponding to the last time that an entry was deleted from the portion of the matrixSDTable and the portion of the matrixDSTable associated with this row
  - Is zero if no deletions have occurred
matrixSDTable

- Is indexed first by matrixSDIndex, then by source address, and then by destination address
- matrixSDSourceAddress
  - The source MAC address
- matrixSDDestAddress
  - The destination MAC address
- matrixSDIndex
  - The set of collected matrix statistics of which this row is a part
- matrixSDPkts
  - Number of packets transmitted from the source to the dest.
- matrixSDOctets
  - # octets contained in all pkts transmitted from this source to the dest.
- matrixSDErrors
  - Number of bad pkts transmitted from source to dest.
matrixDSTable

- Contains the same information as matrixSDTable
- Is indexed first by matrixDSIndex, then by destination address, and then by source address
### Destination address

<table>
<thead>
<tr>
<th>M(i,1)</th>
<th>M(i,2)</th>
<th>M(i,3)</th>
<th>...</th>
<th>M(i,N_j)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>M(i,1)</td>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>M(i,2)</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>M(i,3)</td>
<td>R</td>
<td>R</td>
<td></td>
<td>R</td>
</tr>
</tbody>
</table>

### Source address

<table>
<thead>
<tr>
<th>M(i,N_j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
</tr>
</tbody>
</table>

R = row in both `matrixSDTable` and `matrixDSTable` \( M(i,j) < M(i,j + 1) \)

**FIGURE 8.15** Logical view of the `matrixSDTable` rows and the `matrixDSTable` rows associated with row \( i \) of the `matrixControlTable`
Outline

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
What is Token-Ring?

- was initially defined by IBM at its research facility in Zurich Switzerland in the early 1980s
- IBM pursued standardization of Token-Ring under the 802.5 Working Group of the IEEE
- Is the second most widely used local area network (LAN) technology after Ethernet
- Stations on a Token-Ring LAN are organized in a *ring topology* with data being transmitted sequentially from one ring station to the next
  - The ring is initialized by circulating a *token*
  - A station must capture the token to gain the right to transmit information onto the ring
  - A transmitting station replaces the token with a *frame* which carries the information to be transferred
  - The frame circulates the ring and may be copied by one or more destination stations
  - When the frame returns to the transmitting station, it is removed from the ring and a new token is transmitted
Token-Ring Frame Formats

- The basic transmission unit
  - Is used for transmitting both LLC and MAC frames
  - It may or may not include an information field
  - It may or may not include a routing information field

- Frames are composed of the following fields:
  - Starting Delimiter (1-byte)
  - Access Control (1-byte)
  - Frame Control (1-byte)
  - Destination MAC Address (6-bytes)
  - Source MAC Address (6-bytes)
  - Routing Information (0-30 bytes)
  - Information (0-n bytes)
  - Frame Check Sequence (4-bytes)
  - Ending Delimiter (1-byte)
  - Frame Status (1-byte)
Token Format

- The *token* is the means by which the right to transmit a frame is passed from station to station.
- Stations wishing to transmit capture the token and convert it into a *frame*.
- As the frame circles the ring and returns to its origin, the originating station removes it from the ring and transmits a new token for another station to capture and carry on the process.
  - Starting Delimiter (1-byte)
  - Access Control (1-byte)
  - Ending Delimiter (1-byte)
Abort Sequence Format

- A Token-Ring station may abort a frame it is transmitting at any time by transmitting an *abort sequence*.
- It cause the stations receiving the frame to recognize that it is not a valid frame.
  - Starting Delimiter (1-byte)
  - Ending Delimiter (1-byte)
Field Definitions

- **Starting Delimiter**
  - A frame, token, or abort sequence always starts with a starting delimiter
  - It is one byte in length and consists of a unique sequence of symbols

- **Access Control**
  - is found in frames and tokens and is one byte in length
    - 3 priority bits with 000 being the lowest
    - Token bit: “0” in token and “1” in frames
    - Monitor bit
    - 3 reservation bits

- **Frame Control**
  - is one byte in length and indicates the type of frame
    - Frame Type Bits: “00” is a MAC frame, “01” is an LLC frame
    - Control Bits
      - Used for MAC frames to indicate whether the frame is “normal” or express
      - “000000” represents the frame is normal buffered
Field Definitions

- **Destination Address** field
  - is six bytes in length and identifies the station or stations that are to copy the frame

- **Source Address** field
  - is six bytes in length and identifies the station that originated the frame

- **Routing Information (RI) field**
  - Is used as part of the Token-Ring *source routing* protocol for routing frames between rings in multiple-ring networks

- **Frame Check Sequence** field
  - Contains a 4-byte cyclical redundancy check (CRC) value used for error checking

- **Ending Delimiter field**
  - Is one byte in length and consists of a unique sequence of symbols that includes "code violations" in the Manchester encoded data
Active Monitor

- One station on each ring, called the *active monitor*, provides token monitoring and other functions.
- Any operating ring station can be assigned the active monitor responsibility.
- Other ring stations act as *standby monitors*, prepared to take over if the active monitor fails.
- The active monitor has the following duties:
  - maintains the ring's master clock
  - ensures there is enough delay in the ring to support circulation of a token
  - is responsible for detecting "lost tokens"
  - is responsible for detecting frames and priority tokens that circle the ring more than once
  - uses its "receive notification" timer to ensure that an Active Monitor Present MAC frame circles the entire ring periodically
  - deactivates its active monitor functions if it detects another active monitor present on the ring
Standby Monitor

- Continuously check for failures in the active monitor
- Verifies that a good token periodically circulates the ring
  - It does this by restarting its "good token timer" each time it repeats a priority zero token or a priority token greater than zero followed by a frame
  - The timer has a duration greater than the active monitor's "any token timer"
  - If the good token timer expires, the standby monitor initiates the Claim Token Process
- Verifies that the active monitor periodically transmits an Active Monitor Present MAC frame
  - It does this by restarting its "receive notification" timer each time an Active Monitor Present MAC frame is received
  - The timer has a duration greater than the active monitor's "neighbor notification" timer
  - If the receive notification timer expires, the standby monitor assumes the active monitor has malfunctioned or is no longer present, and initiates the Claim Token process
Claim Token Process

- Determines which station becomes the active monitor
  - The Beacon process is resolved
  - The Ring Purge process fails
  - A station detects that the active monitor functions are not being performed properly
- Is started when a station enters the Transmit Claim Token state and begins transmitting Claim Token MAC frames
- Stations not in the Claim Token state enter the Repeat Claim Token state when they receive a Claim Token MAC frame
- If a station in the Transmit Claim Token state receives a Claim Token MAC frame with a source address that has a higher numerical value than its own station address, then it reverts to Repeat Claim Token state
- This ensures that if multiple stations enter the Transmit Claim Token state, then stations will be eliminated until a single "winner" remains
- That last station remaining in the Transmit Claim Token state wins the Claim Token process when it receives its own Claim Token MAC frame
The purpose of the Ring Purge process is to clean up the ring and release a new token.

The Ring Purge process is started when a station wins the Claim Token process and becomes the active monitor.

- when the active monitor detects a failure in the normal token protocol.
- when the active monitor enters the Transmit Ring Purge state and transmits Ring Purge MAC frames.
- When the active monitor receives its own Ring Purge MAC frame, it transitions to the normal repeat state and transmits a token to start normal ring operation.
  - If the active monitor fails to receive its own Ring Purge MAC frame before the "ring purge" timer expires, it disables its active monitor function and starts the Claim Token process.
Beacon Process

- When a station detects a failure in the Claim Token process, it enters the Transmit Beacon state and transmits Beacon MAC frames until its input signal is restored, or until it removes itself from the ring.
- All other stations that receive the Beacon MAC frame enter the Repeat Beacon mode.
- The Beacon MAC frame identifies the reason for the beaconing (Beacon Type) and the address of its last known upstream neighbor.
- When the beaconing station's nearest upstream neighbor has received eight of these Beacon MAC frames, it removes itself from the ring and tests itself to verify it is not the source of the fault.
<table>
<thead>
<tr>
<th>Counter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tokenRingMLStatsDropEvents</code></td>
<td>Number of events in which packets were dropped by the monitor due to lack of resources. This is not necessarily the actual count of packets dropped, but the number of times this condition has been detected.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsMacOctets</code></td>
<td>Number of octets of data in good MAC packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsMacPkts</code></td>
<td>Number of MAC packets received.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsPurgeEvents</code></td>
<td>Number of times ring enters ring purge state from normal ring state.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsPurgePkts</code></td>
<td>Number of ring purge packets detected.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsBeaconEvents</code></td>
<td>Number of times ring enters beaconing state from a nonbeaconing state.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsBeaconPkts</code></td>
<td>Number of beacon packets detected.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsClaimTokenEvents</code></td>
<td>Number of times ring enters claim token state from normal or purge state.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsClaimTokenPkts</code></td>
<td>Number of claim token packets detected.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsNAUNChanges</code></td>
<td>Number of NAUN changes detected.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsLineErrors</code></td>
<td>Number of line errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsInternalErrors</code></td>
<td>Number of adapter internal errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsBurstErrors</code></td>
<td>Number of burst internal errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsACEErrors</code></td>
<td>Number of address-copied (AC) errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsAbortErrors</code></td>
<td>Number of abort delimiters reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsLostFrameErrors</code></td>
<td>Number of lost frame errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsPktsCongestionErrors</code></td>
<td>Number of receive congestion errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsFrameCopiedErrors</code></td>
<td>Number of frame-copied errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsFrequencyErrors</code></td>
<td>Number of frequency errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsTokenErrors</code></td>
<td>Number of token errors reported in error-reporting packets.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsSoftErrorReports</code></td>
<td>Number of soft error report frames.</td>
</tr>
<tr>
<td><code>tokenRingMLStatsRingPollEvents</code></td>
<td>Number of ring poll events (i.e., the number of ring polls initiated by the active monitor that were detected).</td>
</tr>
<tr>
<td>Traffic Direction</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><strong>tokenRingPStatsData0Octets</strong></td>
<td>Number of octets of data in good MAC data packets</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts</strong></td>
<td>Number of good data packets</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataBroadcastPkts</strong></td>
<td>Number of good broadcast data packets</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataMulticastPkts</strong></td>
<td>Number of good multicast data packets</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts18to63Octets</strong></td>
<td>Number of good data packets between 18 and 63 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts64to127Octets</strong></td>
<td>Number of good data packets between 64 and 127 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts128to255Octets</strong></td>
<td>Number of good data packets between 128 and 255 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts256to511Octets</strong></td>
<td>Number of good data packets between 256 and 511 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts512to1023Octets</strong></td>
<td>Number of good data packets between 512 and 1,023 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts1024to2047Octets</strong></td>
<td>Number of good data packets between 1,024 and 2,047 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts2048to4095Octets</strong></td>
<td>Number of good data packets between 2,048 and 4,095 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts4096to8191Octets</strong></td>
<td>Number of good data packets between 4,096 and 8,191 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPkts8192to16384Octets</strong></td>
<td>Number of good data packets between 8,192 and 16,384 octets in length</td>
</tr>
<tr>
<td><strong>tokenRingPStatsDataPktsGreaterThan16384Octets</strong></td>
<td>Number of good data packets greater than 16,384 octets in length</td>
</tr>
<tr>
<td>Counter Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sourceRoutingStatsInFrames</td>
<td>Number of frames sent into this ring from another ring</td>
</tr>
<tr>
<td>sourceRoutingStatsOutFrames</td>
<td>Number of frames sent from this ring to another ring</td>
</tr>
<tr>
<td>sourceRoutingStatsThroughFrames</td>
<td>Number of frames sent from another ring, through this ring, to another ring</td>
</tr>
<tr>
<td>sourceRoutingStatsAllRoutesBroadcastFrames</td>
<td>Number of good frames received that were AllRoutesBroadcast</td>
</tr>
<tr>
<td>sourceRoutingStatsSingleRoutesBroadcastFrames</td>
<td>Number of good frames received that were SingleRoutesBroadcast</td>
</tr>
<tr>
<td>sourceRoutingStatsInOctets</td>
<td>Number of octets in good frames sent into this ring from another ring</td>
</tr>
<tr>
<td>sourceRoutingStatsOutOctets</td>
<td>Number of octets in good frames sent from this ring to another ring</td>
</tr>
<tr>
<td>sourceRoutingStatsThroughOctets</td>
<td>Number of octets in good frames sent from another ring, through this ring, to another ring</td>
</tr>
<tr>
<td>sourceRoutingStatsAllRoutesBroadcastOctets</td>
<td>Number of octets in good frames received that were AllRoutesBroadcast</td>
</tr>
<tr>
<td>sourceRoutingStatsSingleRoutesBroadcastOctets</td>
<td>Number of octets in good frames received that were SingleRoutesBroadcast</td>
</tr>
<tr>
<td>sourceRoutingStatsLocalLLCFrames</td>
<td>Number of frames received with no RIF (RoutingInformationField) and were not AllRouteBroadcast frames</td>
</tr>
<tr>
<td>sourceRoutingStats1HopFrames</td>
<td>Number of frames received whose route had one hop, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStats2HopsFrames</td>
<td>Number of frames received whose route had two hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStats3HopsFrames</td>
<td>Number of frames received whose route had three hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStats4HopsFrames</td>
<td>Number of frames received whose route had four hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStats5HopsFrames</td>
<td>Number of frames received whose route had five hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStats6HopsFrames</td>
<td>Number of frames received whose route had six hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStats7HopsFrames</td>
<td>Number of frames received whose route had seven hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStats8HopsFrames</td>
<td>Number of frames received whose route had eight hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
<tr>
<td>sourceRoutingStatsMoreThan8HopsFrames</td>
<td>Number of frames received whose route had more than eight hops, were not AllRouteBroadcast frames, and whose source or destination were on this ring</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Basic Concepts
- statistics Group
- history Group
- host Group
- hostTopN Group
- matric Group
- tokenRing Extensions to RMON
- Summary
Remote monitoring (RMON) added to SNMP framework

Remote monitoring refers to the use of an agent device connected to a broadcast network to collect statistics concerning traffic on that network.

An agent is responsible for management that relates to the agent’s device.