



第5章 远程网络监视

- RMON 的基本概念
- RMON 1 MIB
- RMON 2 MIB
- 其他RMON MIB



RMON的基本概念

为什么需要RMON

1.什么是RMON?

- RMON是Remote Monitoring的缩写，它最初由IETF（互联网工程任务组）定义，主要用于从远程监视LAN和VLAN（虚拟局域网）的通信。



网络监视

- 网络监视（Monitoring or Probing）
 - 捕获局域网上的每个包，并对其进行统计、分析
 - 包仍按原有的方式传递，其内容不发生变化
- 网络监视器（Monitor, Analyzer, Probe）
 - 在局域网中执行网络监视功能的设备
- 网络监视功能：用于局域网整体流量参数统计、分析
 - 错误统计：如小于规定大小的包数或冲突数量
 - 性能统计：每秒传递的包数以及包的大小分布



远程网络监视

- 远程网络监视
 - 由网络监视器在本地对局域网的参数进行收集、统计和分析
 - 收集的参数和分析结果可以通过某种方式传送给远程网络管理站点
 - 远程网络管理站点可以通过一定的方式控制网络监视器



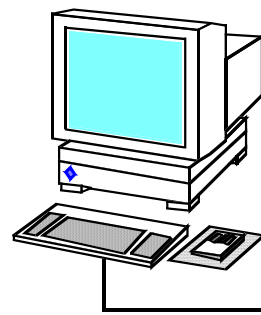
远程网络监视

- 远程网络监视的优点
 - 降低网络管理信息的通信量
 - 减轻网络管理站的负担
 - 本地监视可以增加信息的可靠性
 - 通过对局域网段所有包的捕获，能够有效地获取局域网段的总体流量参数
 - 通过对局域网段的连续监控，可以获得更为及时的统计信息，从而能够对网络故障做出及时诊断

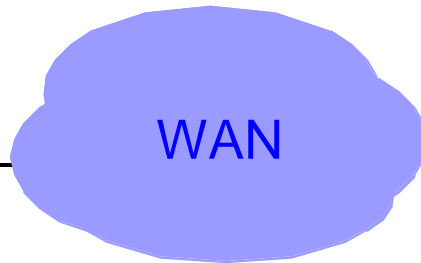


RMON

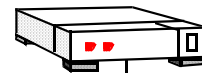
MANAGER



WAN



RMON



ETHERNET





2. 为何要RMON?

- SNMP轮询有两个明显的弱点：
 - (1) 它没有伸缩性。在大型的网络中，轮询会产生巨大的网络管理通信量，因而导致通信拥挤情况的发生。
 - (2) 它将收集数据的负担加在网络管理控制台上。如果轮询多个网络，管理工作站的负载加重，影响任务的完成。

RMON MIB的出现解决了该问题

- RMON MIB弥补了轮询的弊端，扩充了SNMP的管理信息库MIB-II，使SNMP更为有效、更为积极主动地监控远程设备。



RMON的基本概念

- RMON（远程网络监视）MIB是对于SNMP的一个重要增强。定义了标准网络监视功能，以及网络管理进程与远程网络监视器之间的通信接口。
- RMON最大优点就在于它与现存的SNMP框架相兼容，不需对此协议进行任何修改。
- 当前 RMON 有两种版本：RMON v1 和 RMONv2。RMON v1在目前使用较为广泛的网络硬件中都能发现，它定义了9个MIB 组服务于基本网络监控；RMON v2是RMON v1的扩展，专注于MAC层以上更高的流量层，它主要强调IP流量和应用程序层流量。允许网络管理应用程序监控所有网络层的信息包， RMON v1只允许监控MAC及其以下层的信息包。
- RMON监视系统有两部分构成：探测器（代理或监视器）和管理站。RMON代理在 RMON MIB中存储网络信息，它们被直接植入网络设备（如路由器、交换机等），代理也可以是PC机上运行的一个程序。代理只能看到流经它们的流量，所以在每个被监控的 LAN 段或WAN 链接点都要设置RMON代理，网管工作站用SNMP获取RMON⁸数据信息。

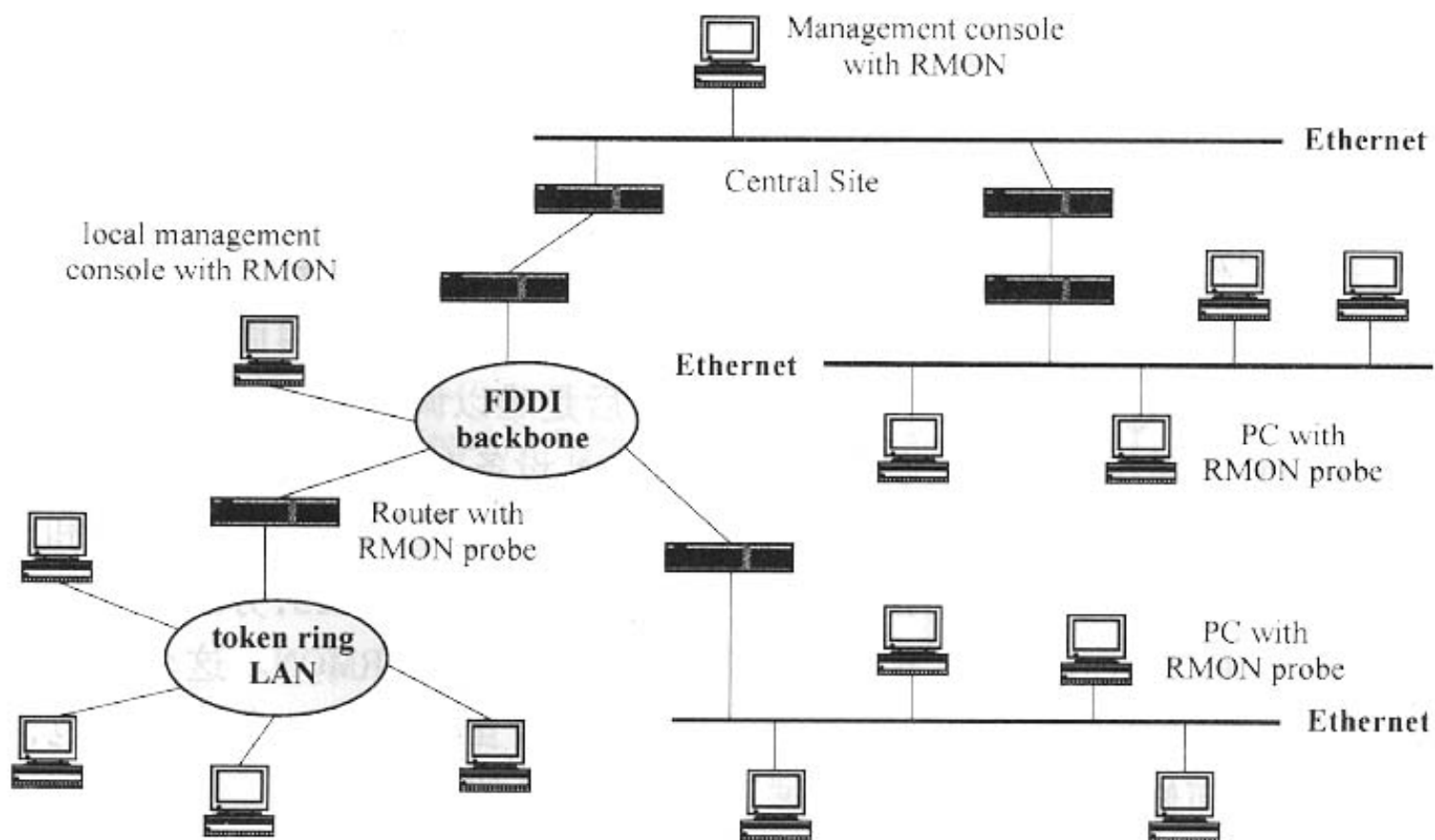


RMON简介

- 利用MIB II，网络管理员可以获得对单个设备的本地信息，如可以获取出入每个设备的信息流量，但不容易获知LAN上的整体流量
- 远程网络监视（RMON）在互联网中的每个子网设置一个网络监视器（Probe），以在该子网中捕获并分析流量
- 网络管理进程与远程网络监视器通信，以获取各个子网的流量信息
- RMON可以在降低其他代理和管理工作站负载的情况下监视子网的行为



RMON配置示例





RMON的目标

(1) 离线操作：由于偶然的网络故障影响，或者为了降低通信成本,管理站可以停止对监视器的轮询，从而提高带宽利用率。即使不被管理站查询，监视器也能不断收集子网故障、性能和配置方面信息，统计和积累数据，以便管理站查询时及时提供管理信息。

(2) 主动监视：如监视器有足够资源，通信负载允许，监视器可以连续地或周期地运行诊断程序，获得并记录网络性能参数。如网络管理者想进一步诊断问题所在时，可以通过管理工作站回放这些历史信息

(3) 问题检测和报告：如果主动监视消耗网络资源太多，监视器也可以被动地获得网络数据。可以配置监视器，使其连续观察网络资源的消耗情况，记录随时出现的异常条件，并在出现异常时向管理站报告。

(4) 提供增值数据：监视器可以分析收集到的子网数据，从而减轻管理站的计算任务。如，监视器可以分析子网的通信情况，计算出哪些主机通信最多，哪些主机出错最多等。这些数据的收集和计算由监视器来做，比由远处的管理站来做更有效。

(5) 多管理站操作：一个网络可以有多个管理站，这样可以提高可靠性，或者分布地实现不同的网络管理功能。可以配置监视器使其能够并发地工作，为不同的管理提供不同的信息。



远程监视器的实现有两种方式

- (1) 使用一个专用设备来实现远程监视器，其唯一目的就是捕获并分析流量；
- (2) 由具有其它任务的设备如工作站、服务器或路由器来执行。把它作为系统功能的一部分，该系统有专用于监视功能的处理器和内存资源，有了这些专用资源，远程监视器可以执行比一个只支持MIB-II的代理更复杂、更广泛的功能。为了有效地进行网络管理，这些监视器需要与一种中央网络管理站通信。

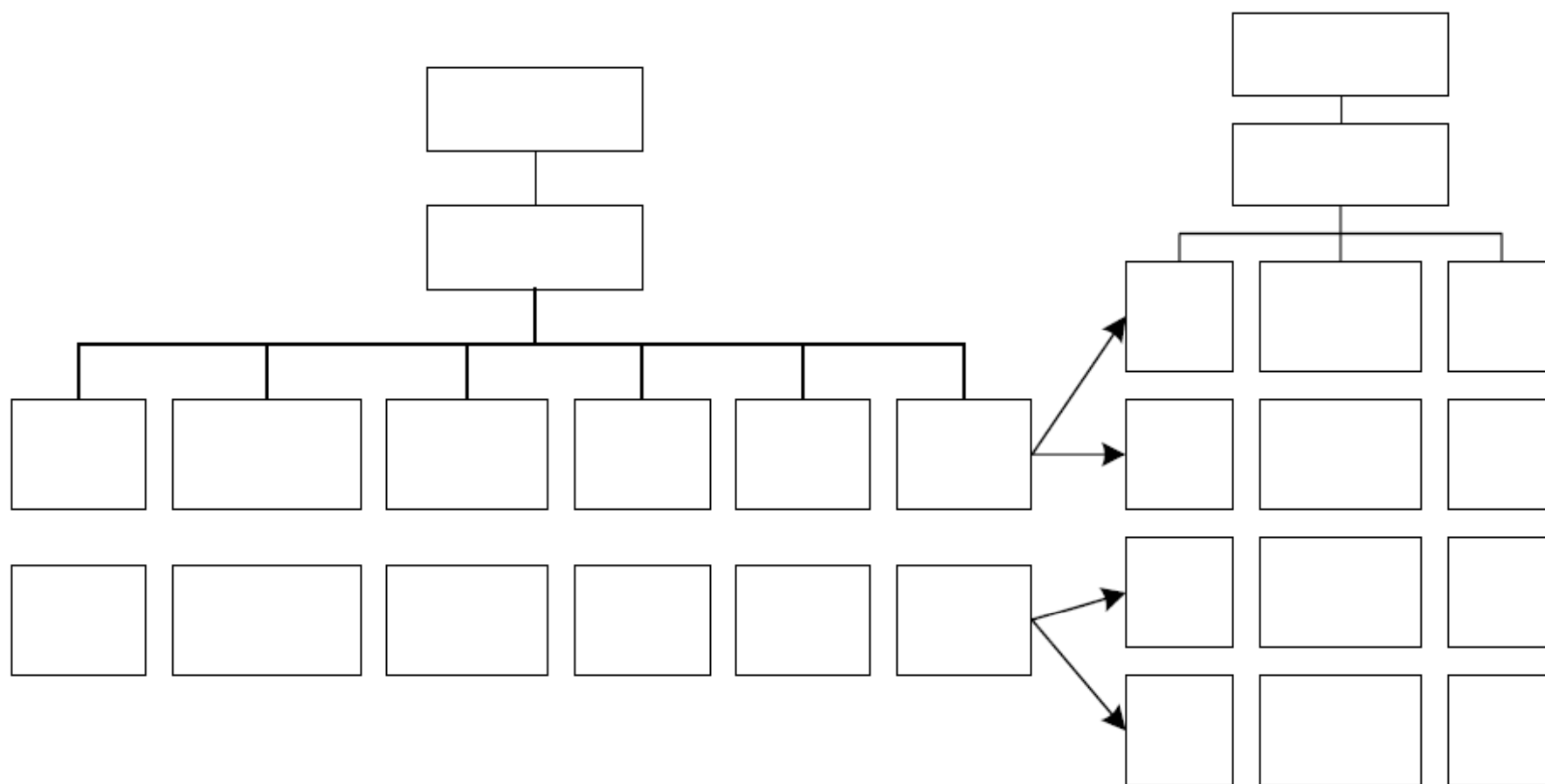
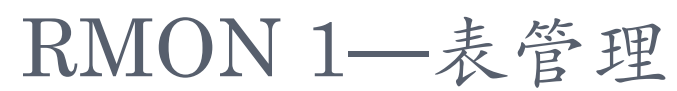


RMON监视器的控制

1. 监视器的配置

对远程的RMON监视器进行配置，以指明监视器要收集的数据类型和形式。RMON MIB的监视器配置方案如下：在每一个功能组内，包含一个或多个控制表和数据表，控制表定义数据表的结构，数据表用于存储数据。

- 控制表：包含描述数据表中数据的参数，管理站通过设置相应的控制参数来配置远程监视器使其收集所需的数据，管理站可以在控制表中加入一个新行或更改已有的一行来设置参数，如可以指明收集数据的来源、数据类型及收集的时间等，可读可写。
- 数据表：监视器按控制表中的参数设置收集信息，并存放在相应的数据表的行中，只可读。
- 在有些情况下，定义数据收集功能的控制参数和用于存放收集来数据的一行对象是一一对应的，控制表和数据表合二为一
- 每个控制纪录及其对应的数据可以通过一个互锁指针绑定起来。
- 修改控制表的方法：先删除后插入





监视器的操作控制

- SNMP的主要功能就是读取（Get）和设置（Set）MIB视图内对象的值。
- 通过SNMP的Set操作达到发布命令和控制的目的，一个对象可以用来代表一个命令。如果这个对象被设置为一个特殊的数值，则表明一个特定动作就会被执行。如果设置对象的值为其当前值，将不会导致操作被执行。



多管理者访问

RMON监视器应允许多个管理站并发地访问，当多个管理站访问时可能出现以下问题：

- (1) 多个管理站对资源的并发访问可能超过监视器的能力。
- (2) 一个管理站可能长时间占用监视器资源，使得其它站得不到访问。
- (3) 占用监视器资源的管理站可能出现崩溃，而没有释放资源。



解决方法:

在控制表中引入一个列对象,用于指定表中特定的记录以及相关功能的属主。

控制表中的一行只能被其属主修改和删除,其它管理站只能读。

通过所有者标号:

A. 管理站可以识别它所有的不再使用的资源。

B. 如果管理站经过了重新启动,它可以识别其在过去预定的资源,并释放它不再需要的那些部分。

(1) 管理站能认得自己所属的资源,也知道自己不再需要的资源。

(2) 网络管理员可以知道管理站占有的资源,并决定是否释放这些资源。

(3) 一个被授权的网络管理员可以自主决定是否释放其它网络管理员的资源。

(4) 如果管理站重新启动,它应该首先释放不再使用的资源。

引入共享来实现多个网络管理员访问同一控制表



RMON的表管理

1. 文本约定:

两种新的数据类型

- 行所有者对象: 类型为OwnerString, 与DisplayString相同, 为0~256之间的字符串。便于记忆, 这类对象名以Owner结尾

OwnerString::=DisplayString

每一个读写表都有一个指示每行属主的对象, 该对象类型为OwnerString。

- 行状态对象: 类型为EntryStatus, 可取{valid(1)、createRequest(2)、underCreation(3)、Invalid(4)}四值之一, 这类对象名以Status结尾, 用于创建、修改和删除行

EntryStatus::=INTEGER{valid(1)

createRequest(2)

underCreation(3)

Invalid(4)}

该对象数值给出了包含其对象实例的状态纪录



2. 增加行

利用SNMP的Set命令在RMON表中增加新行。同时，针对多个管理站并发表增加操作所带来的冲突问题，RMON通过设置表示行状态的*EntryStatus*对象的值，从而安全地控制行的创建。

- (1) 管理站发出创建请求，并索引对象值不存在，则以createRequest(2)为状态值开始创建一新行
- (2) 创建操作完成后，代理把状态对象值设置成underCreation(3)
- (3) 在管理站创建完其配置所需的所有行之前，这些行应该一致处于underCreation状态，直至所有记录创建完毕，管理站将新创建行的状态值设置为valid(1)
- (4) 如果其他管理站试图以createRequest状态创建新行时，而该行已存在，则会返回错误信息

注：管理站也可以通过将状态对象从无效改为有效将一行加入表中



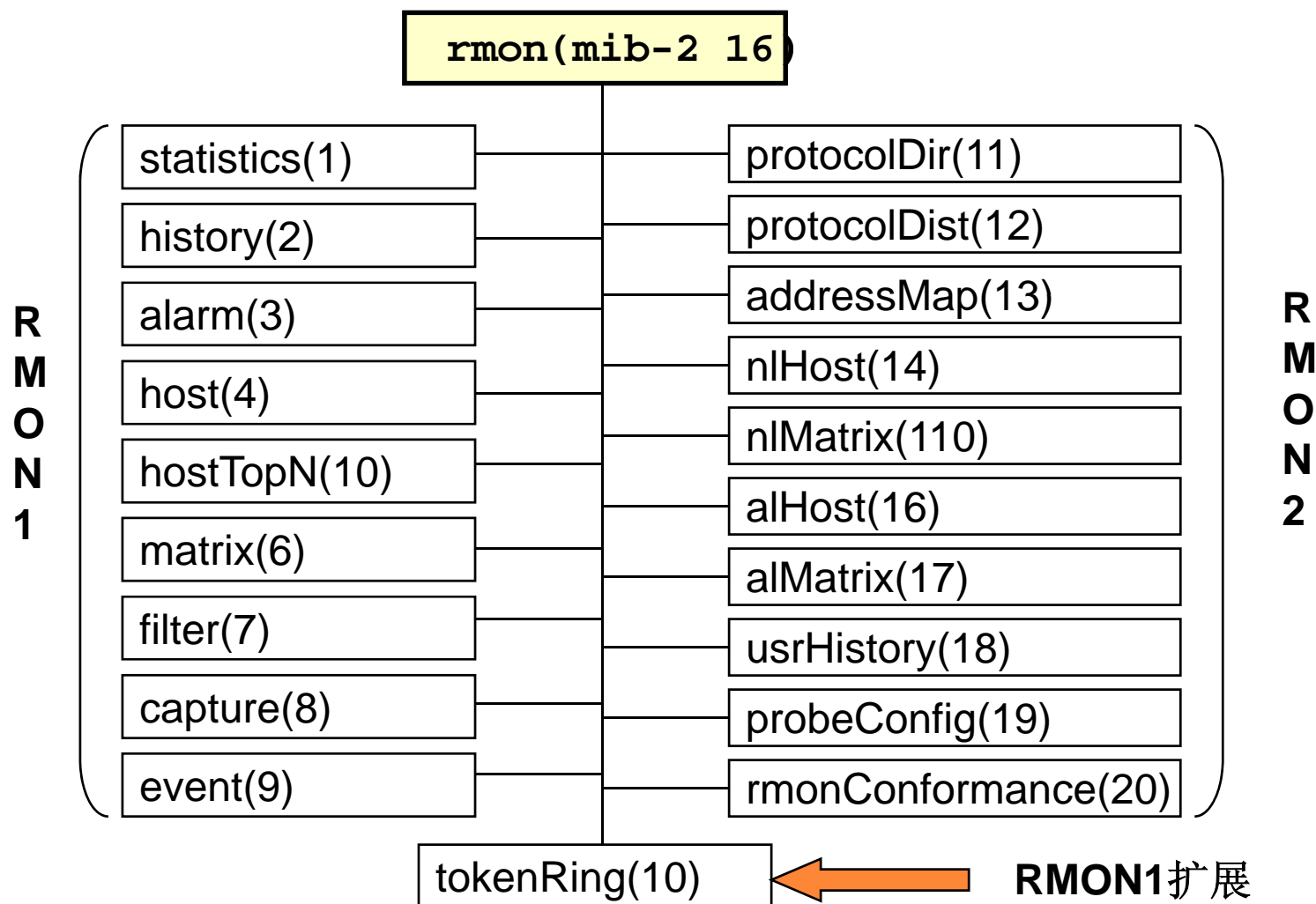
3. 行的修改和删除

删除：通过**SetRequest**设置状态对象值无效**invalid(4)**

修改：首先使之无效，然后赋予新的参数值（新增）



RMON的管理信息库





RMON的管理信息库

○ RMON1和RMON2 对比

网络管理问题	相关 OSI 层	管理标准
物理故障与利用	介质访问控制层	RMON1
局域网网段	数据链路层	RMON1
网络互连	网络层	RMON2
应用程序的使用	应用层	RMON2

○ 联系： RMON2是RMON1的技术补充



RMON1 MIB

- 包括9个以太网功能组和1个令牌环网功能组：

统计组（Statistics）	以太网流量统计
历史组（History）	一段时间的统计（流量和错误率）
警报组（Alarm）	用户定义任何RMON变量的阈值的告警
主机组（Hosts）	由MAC地址区分的每个主机流量和错误率
最高N台主机组（Host Top N）	基于最大流量和/或错误率的主机排序
矩阵组（Matrix）	主机之间的会话矩阵
过滤组（Filter）	定义哪些数据报文的类型进行捕获和存储
捕获组（Capture）	建立一组缓冲区，用于存储从通道中捕获的报文
事件组（Event）	产生日志条目和/或 SNMP陷阱
令牌环网组（Token Ring）	RFC1513扩展了RMON MIB，增加了有关IEEE 802.5令牌环网的管理信息



RMON1 MIB

- RMON MIB中的所有组都是可选的，但是，它们之间有一些依赖性
- 实现RMON 1 MIB必须实现MIB-II（RFC 1213）和IF-MIB（RFC 2233）的系统组
- 警报组需要事件组的实现。
- 最高N台主机组（hostTopN）需要主机组的实现
- 包捕获组需要过滤组的实现。



STATISTICS统计组

- 每个监视子网的基本统计信息，包括子网负载和子网总体健康状况的信息
- 与接口组类似，二者有重叠，但提供了子网关于行为更详细的信息
- **RFC1757** 定义的统计组包含一个表（数据表和控制表合并），每个监视接口（子网）对应一个表项，**RFC1713** 增加两个表。
- 统计用计数器的形式表示，初始值为0



STATISTICS统计组

提供了很多以太网数据包的统计信息，例如接收到的数据包数、字节数、广播包数、组播包数，CRC校验错误、对准错误、冲突：

- **PACKETS**
- **OCTETS**
- **BROADCASTS**
- **MULTICASTS**
- **COLLISIONS**
- **ERRORS**

提供了超大或超小数据包分布情况：

- **65 - 127 OCTETS**
- **128 - 255 OCTETS**
- **256 - 511 OCTETS**
- **512 - 1023 OCTETS**
- **1024 - 1518 OCTETS**



STATISTICS统计组

statistics (rmon 1)

- etherStatsTable(1) 表示一个以太网的统计信息
- tokenRingMLStatsTable(2) 统计令牌环中各种MAC控制分组
- tokenRingPStatsTable(3) 统计各种数据分组



etherStatsTable(1)

└ etherStatsEntry(1)

- └ etherStatsIndex(1) 索引，对应一个子网
- └ etherStatsDataSource(2) 监视器接收数据的以太网接口
- └ etherStatsDropEvents(3) 因资源不足而丢弃的分组数
- └ etherStatsOctets(4) 接收到的字节总数
- └ etherStatsPkts(5) 接收到的分组总数
- └ etherStatsBroadcastPkts(6) 接收到的广播分组数
- └ etherStatsMulticastPkts(7) 接收到的组播分组数
- └ etherStatsCRCAlignErrors(8) 接收到的CRC错误或有对准错误的分组数
- └ etherStatsUndersizePkts(9) 不足64字节的分组数
- └ etherStatsOversizePkts(10) 大于1518字节的分组数
- └ etherStatsFragments(11) 不足64字节且CRC错误或有对准错误的分组数
- └ etherStatsJabbers(12) 大于1518字节且CRC错误或有对准错误的分组数
- └ etherStatsCollisions(13) 子网上发生冲突的次数
- └ etherStatsPkts64Octets(14) 长度为64字节的分组数
- └ etherStatsPkts65To127Octets(15) 长度为65~127字节的分组数
- └ etherStatsPkts128To255Octets(16) 长度为128~255字节的分组数
- └ etherStatsPkts256To511Octets(17) 长度为256~511字节的分组数
- └ etherStatsPkts512To1023Octets(18) 长度为512~1023字节的分组数
- └ etherStatsPkts1024To1518Octets(19) 长度为1024~1518字节的分组数
- └ etherStatsOwner(20) 行的所有者
- └ etherStatsStatus(21) 行的状态



tokenRingMLStatsTable(2)

└ tokenRingMLStatsEntry(1)

- └ tokenRingMLStatsIndex(1) 索引，对应一个子网
- └ tokenRingMLStatsDataSource(2) 监视器接收数据的令牌网接口
- └ tokenRingMLStatsDropEvents(3) 因资源不足而丢弃的分组数
- └ tokenRingMLStatsMacPkts(4) 接收到的MAC分组总数
- └ tokenRingMLStatsMacOctets(5) 好的MAC分组中的字节数
- └ tokenRingMLStatsRingPurgeEvents(6) 环进入清除状态的次数
- └ tokenRingMLStatsRingPurgePkts(7) 检测到的清除分组数
- └ tokenRingMLStatsBeaconEvents(8) 环进入信标状态的次数
- └ tokenRingMLStatsBeaconTime(9) 环进入信标状态的时间
- └ tokenRingMLStatsBeaconPkts(10) 检测到的信标分组数
- └ tokenRingMLStatsClaimTokenEvents(11) 环进入声明令牌状态的次数
- └ tokenRingMLStatsClaimTokenPkts(12) 检测到的声明令牌分组数
- └ tokenRingMLStatsNAUNChanges(13) 检测到NAUN改变的次数
- └ tokenRingMLStatsLineErrors(14) 检测到错误行的数
- └ tokenRingMLStatsInternalErrors(15) 适配器内部错误数
- └ tokenRingMLStatsBurstErrors(16) 突发的内部错误数
- └ tokenRingMLStatsACErrors(17) 地址拷贝错误数
- └ tokenRingMLStatsAbortErrors(18) 夭折错误数
- └ tokenRingMLStatsLostFrameErrors(19) 丢失帧错误数
- └ tokenRingMLStatsCongestionErrors(20) 接收拥挤错误数
- └ tokenRingMLStatsFrameCopiedErrors(21) 帧拷贝错误数
- └ tokenRingMLStatsFrequencyErrors(22) 频率错误数
- └ tokenRingMLStatsTokenErrors(23) 令牌错误数
- └ tokenRingMLStatsSoftErrorReports(24) 软件错误数
- └ tokenRingMLStatsRingPollEvents(25) 环查询数
- └ tokenRingMLStatsOwner(26) 行的所有者
- └ tokenRingMLStatsStatus(27) 行的状态



tokenRingPStatsTable(3)

└ tokenRingPStatsEntry(1)

- tokenRingPStatsIndex(1) 索引，对应一个子网
- tokenRingPStatsDataSource(2) 监视器接收数据的令牌网接口
- tokenRingPStatsDropEvents(3) 因资源不足而丢弃的分组数
- tokenRingPStatsDataOctets(4) 好的MAC分组中的字节数
- tokenRingPStatsDataPkts(5) 好的MAC分组数
- tokenRingPStatsDataBroadcastPkts(6) 广播分组数
- tokenRingPStatsDataMulticastPkts(7) 组播分组数
- tokenRingPStatsDataPkts18to63Octets (8) 18~63字节的分组数
- tokenRingPStatsDataPkts64to127Octets (9) 64~127字节的分组数
- tokenRingPStatsDataPkts128to255Octets (10) 128~255字节的分组数
- tokenRingPStatsDataPkts256to511Octets (11) 256~511字节的分组数
- tokenRingPStatsDataPkts512to1023Octets (12) 512~1023字节的分组数
- tokenRingPStatsDataPkts1024to2047Octets (13) 1024~2047字节的分组数
- tokenRingPStatsDataPkts2048to4095Octets (14) 2048~4095字节的分组数
- tokenRingPStatsDataPkts4096to8191Octets (15) 4096~8191字节的分组数
- tokenRingPStatsDataPkts8192to18000Octets (16) 8192~18000字节的分组数
- tokenRingPStatsDataPktsGreaterThan18000Octets(17) 大于18000字节的分组数
- tokenRingPStatsOwner(18) 行的所有者
- tokenRingPStatsStatus(19) 行的状态



HISTORY 历史组

- 历史组存储的是以固定间隔取样所获得的子网数据，定义一个或多个接口的采样功能
- 管理者可以设置接口(the ethernet segments interfaces)和采样间隔
 - 历史控制表：详细说明接口及其采样功能
 - 接口+采样间隔：标记一行
 - 对于一个子网可以启动多个采样过程，但每个采样过程需要设置不同的采样间隔
 - 规范建议每个监视接口至少有两个历史控制项，采样间隔分别为30秒和30分钟
 - 短的采样间隔可以检测到流量模式的突变，较长间隔可以监视接口长期状态是否稳定
 - 历史数据表：记录数据
 - 一行：一次采样
 - 使用循环缓冲区(桶)存储统计信息



HISTORY 历史组

history (rmon 2)

- historyControlTable(1) 历史控制表
- etherHistoryTable(2) 以太网介质数据表
- tokenRingMLHistoryTable(3) 令牌网介质数据表
- tokenRingPHistoryTable(4) 令牌网介质数据表



HISTORY 历史组

historyControlTable (1)

└ historyControlEntry(1)

- └ historyControlIndex(1) 索引
- └ historyControlDataSource(2) 被采样接口编号
- └ historyControlBucketsRequested(3) 请求的吊桶数(默认值为50)
- └ historyControlBucketsGranted(4) 实际得到的吊桶数
- └ historyControlInterval(5) 采样间隔长度(默认值为1800s)
- └ historyControlOwner(6)
- └ historyControlStatus(7)

etherHistoryTable (2)

└ etherHistoryEntry(1)

- etherHistoryIndex(1) 索引，与historyControlIndex相同
- etherHistorySampleIndex(2) 索引，唯一标识一个样品
- etherHistoryIntervalStart(3) 采样开始时sysUpTime的值
- etherHistoryDrapEvents(4) 因资源不足而丢弃的分组数
- etherHistoryOctets(5) 接收到的字节总数
- etherHistoryPkts(6) 接收到的分组总数
- etherHistoryBroadcastPkts(7) 接收到的广播分组数
- etherHistoryMulticastPkts(8) 接收到的组播分组数
- etherHistoryCRCAlignErrors(9) 接收到的CRC错误或有对准错误的分组数
- etherHistoryUndersizePkts(10) 不足64字节的分组数
- etherHistoryOversizePkts(11) 大于1518字节的分组数
- etherHistoryFragments(12) 不足64字节且CRC错误或有对准错误的分组数
- etherHistoryJabbers(13) 大于1518字节且CRC错误或有对准错误的分组数
- etherHistoryCollisions(14) 子网上发生冲突的次数
- etherHistoryUtilization(15) 表示子网利用率



HistoryControlTable

HistoryControlIndex HistoryControlDataSourceHistoryControlBucketsGranted HistoryControlInterval

1	D_1	B_1	I_1
2	D_2	B_2	I_2
...
k	D_k	B_k	I_k

etherHistoryTable

etherHistoryIndex

etherHistorySampleIndex

1	$x+1$
1	$x+2$
1	$x+3$
...	...
1	$x+B_1$
2	$y+1$
2	$y+2$
2	$y+3$
...	...
2	$y+B_2$
...	...



主机信息

两个组:

- **HOST**
- **HOST TOP N**

IN / OUT:

PACKETS / OCTETS

OUT:

BROADCASTS

MULTICASTS

ERRORS

INFORMATION INDEXED BY:

- **INTERFACE AND MAC ADDRESS (hostTable)**
- **CREATION TIME (hostTimetable)**
- **SORTED ON SOME VARIABLE VALUE (hostTopN)**



HOST主机组

- 主机组用于收集LAN上特定主机的统计信息

host (rmon 4)

- hostControlTable(1) 控制表
- hostTable(2) 保存主机统计数据
- hostTimeTable(3) 数据表



hostControlTable(1)

└─ hostControlEntry(1)

→ hostControlIndex(1) 索引项

— hostControlDataSource(2) 标识网络接口

— hostControlTableSize(3) 主机数据表的行数，即子网上的主机数

— hostControlLastDeleteTime(4) 主机表删除一行的时间，与
SysUpTime相同

— hostControlOwner(5)

— hostControlStatus(6)



hostTable(2)

└─hostEntry(1)

→	hostAddress(1)	主机的MAC地址
	hostCreationOrder(2)	主机被发现的顺序号
→	hostIndex(3)	与hostControlIndex匹配
	hostInPkts(4)	输入分组数
	hostOutPkts(5)	输出分组数
	hostInOctets(6)	输入字节数
	hostOutOctets(7)	输出字节数
	hostOutErrors(8)	输出出错分组数
	hostOutBroadcastPkts(9)	输出广播分组数
	hostOutMulticastPkts(10)	输出多播分组数

用途：保存相应主机的统计数据

hostTimeTable(3)

└─ hostTimeEntry(1)





HOST主机组

- hostControlTable和hostTable是直接对应关系，因而hostTable中行数为：

$$N = \sum_{i=1}^k N_i$$

N: hostTable中的总行数

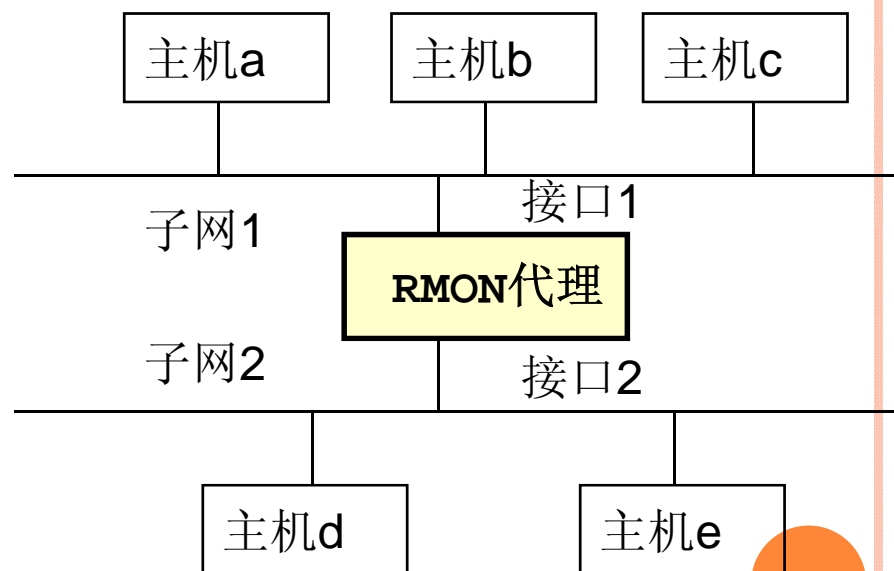
i: hostControlIndex

k: hostControlTable的行数

N_i : 为hostControlTable中第i行的
hostControlTableSize值

例：如右图

k=2、N1=3、N2=2

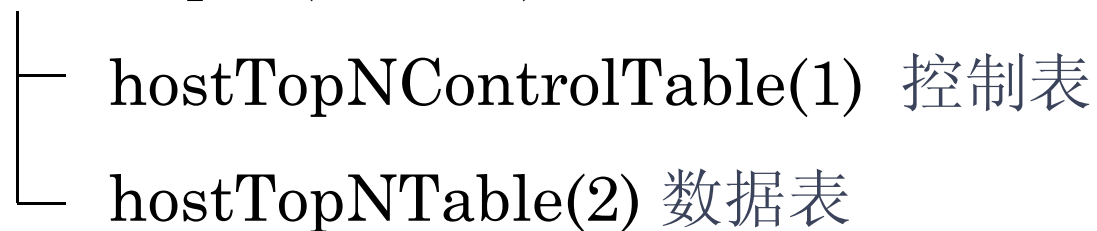




HOSTTOPN 最高N台主机组

- 维持对子网上一组主机的统计
 - 这些主机在用某种参数作比较时位于列表的最高处
 - 统计的数据来自主机组

hostTopN (rmon 5)





其中：

hostTopRateBase的值为整数，可以取7种值，如右图所示。

```
SYNTAX INTEGER {  
    hostTopNInPkts(1),  
    hostTopNOutPkts(2),  
    hostTopNInOctets(3),  
    hostTopNOutOctets(4),  
    hostTopNOutErrors(5),  
    hostTopNOutBroadcastPkts(6),  
    hostTopNOutMulticastPkts(7) }
```



hostTopNTable(2)

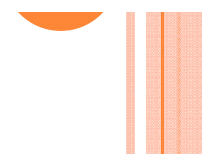
└─hostTopNEntry(1)

→ └─hostTopNReport(1) 与控制表索引相同

→ └─hostTopNIndex(2) 表示惟一的主机

└─hostTopNAddress(3) 主机的MAC地址

└─hostTopNRate(4) 采样区间内采样变量变化的数量





MATRIX矩阵组

- 用于纪录一个子网内各对主机间的流量信息 (packets,octets,errors)。
- 可用于入侵检测

matrix (rmon 6)

- matrixControlTab(1)
- matrixSDTable(2)
- matrixDSTable(3)



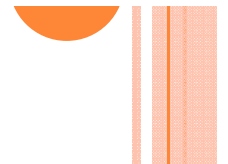
matrixControlTab(1)

└─ matrixControlTable(1)





索引顺序: matrixSDTable 依次用 matrixSDIndex,
matrixSDSourceAddress, matrixSDDestAddress来索引



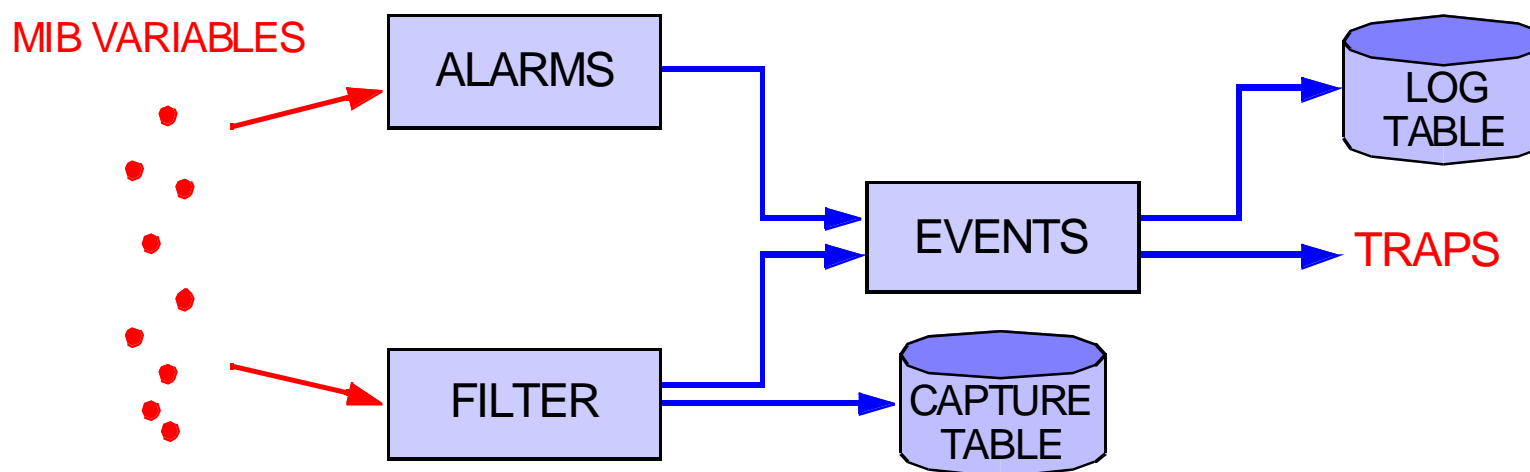


索引顺序: matrixDSTable 依次用 matrixDSIndex,
matrixDSDestAddress , matrixDSSourceAddress来索引





其它组



FILTER GROUP

- TO COUNT PACKETS THAT CARRY A SPECIFIC BIT-PATTERN

PACKET CAPTURE GROUP

- TO STORE SPECIFIC PACKETS

EVENT GROUP

- TO DEFINE THE VARIOUS EVENTS
- TO DETERMINE ON LOGGING AND / OR TRANSMISSION OF TRAPS



FILTER过滤组

- filter组提供管理站向监视器发送命令的机制，用以指示监视器有选择地观测特定网络接口上的数据包。
- 过滤组定义了数据过滤器和状态过滤器两种过滤器，其中数据过滤器使得监视器可以根据数据包的位模式（匹配或不匹配）来观察数据包；状态过滤器使得监视器可以根据数据包的状态（如有效、CRC错误等）来观察数据包。

filter (rmon 7)

- filterTable(1) 定义了相关的过滤器
- channelTable(2) 表的每行定义了惟一的一个通道

channelTable(2)

└─ channelEntry(1)

- └─ channelIndex(1) 索引, 定义一个通道
- └─ channelIfIndex(2) 说明监视器的接口
- └─ channelAcceptType(3) 通道接受类型
- └─ channelDataControl(4) 通道开关
- └─ channelTurnOnEventIndex(5) 事件索引
- └─ channelTurnOffEventIndex(6) 事件索引
- └─ channelEventIndex(7) 事件索引, channelEventIndex(7) 为 channelEventIndex(7) 分组被匹配时产生, 0..65535
- └─ channelEventStatus(8) 事件状态
- └─ channelMatches(9) 记录匹配分组数
- └─ channelDescription(10) 通道的文本描述
- └─ channelOwner(11)
- └─ channelStatus(12)

取值eventMatched(1)或
(2)

取值on(1)
或 off(2)

取值eventReady(1),
eventFired(2),
eventAlwaysReady(3)

filterTable(1)

└─ filterEntry(1)

- —filterIndex(1) 索引，每行定义了一个数据过滤器和状态过滤器
- filterChannelIndex(2) 指向该过滤器所属的通道
- filterPktDataOffset(3) 被测试数据距分组头的位移
- filterPktData(4) 与输入数据进行匹配的数据
- filterPktDataMask(5) 用于匹配过程的掩码
- filterPktDataNotMask(6) 用于匹配过程的掩码反码
- filterPktStatus(7) 与输入数据进行匹配的状态
- filterPktStatusMask(8) 用于状态匹配过程的掩码
- filterPktStatusNotMask(9) 用于状态匹配过程的掩码反码
- filterOwner(10)
- filterStatus(11)



CAPTURE-捕获组

- packet-capture组可以用于设置一种缓冲机制，以捕获流经filter组的某个通道的数据分组，该组的实现依赖于filter组的实现。

capture (rmon 8)

- bufferControlTable(1) 每一行定义一个缓冲区用于捕获和存储一个通道中的分组
- captureBufferTable(2) 每一行对应一个捕获分组

bufferControlTable(1)

└ bufferControlEntry(1)

- └ bufferControlIndex(1) 索引
- └ bufferControlChannelIndex(2) 指向一个通道
- └ bufferControlFullStatus(3)表示缓冲区是否用完: spaceAvailable(1)或full(2)
- └ bufferControlFullAction(4)表示缓冲区是否循环使用: lockWhenFull(1)或warpWhenFull(2)
- └ bufferControlCaptureSliceSize(5) 被捕获分组可存入缓冲区中的最大字节数
- └ bufferControlDownloadSliceSize(6) 单个SNMP PDU可从缓冲区取得的最大字节数
- └ bufferControlDownloadOffset(7)SNMP从缓冲区取得的第一个字节距分组头的位移
- └ bufferControlMaxOctetsRequested(8) 请求的缓冲区大小
- └ bufferControlMacOctetsGranted(9) 得到的缓冲区大小
- └ bufferControlCapturePkts(10) 当前在缓冲区中的分组数
- └ bufferControlTurnTime(11) 打开缓冲区的时间
- └ bufferControlOwner(12)
- └ bufferControlStatus(13)

captureBufferTable(1)

└ captureBufferEntry(1)

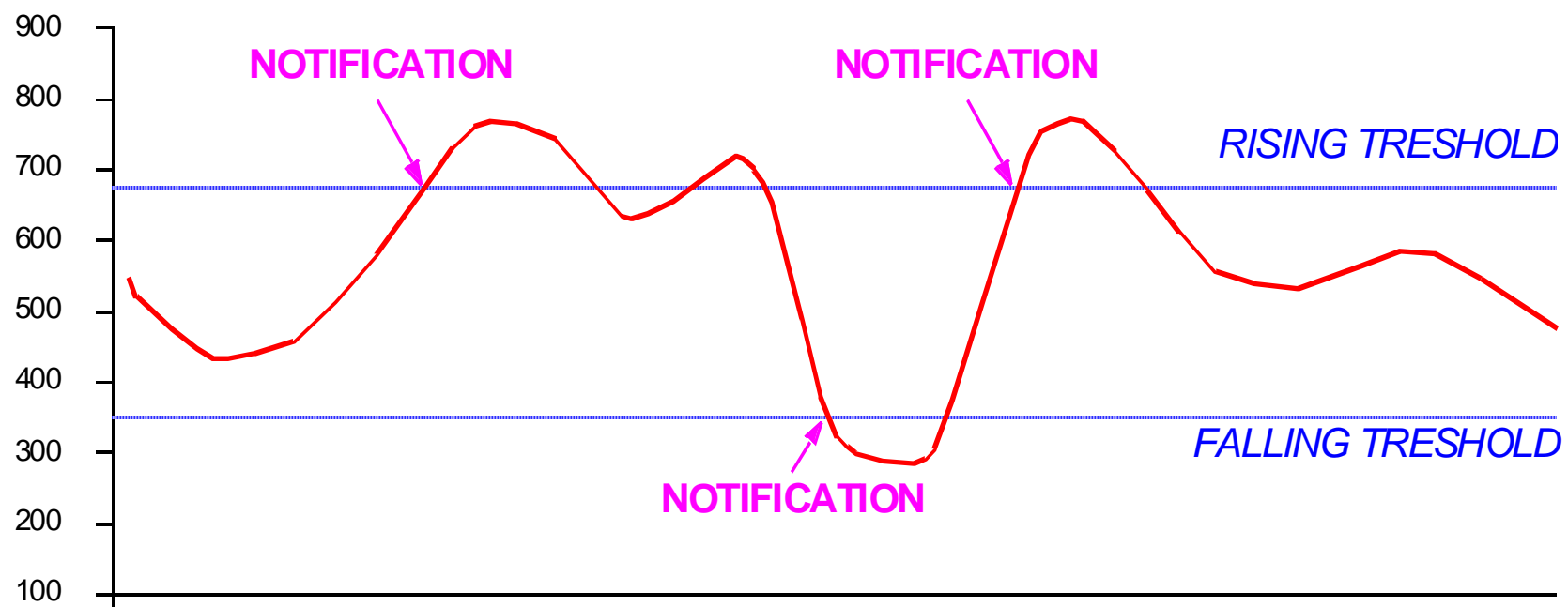


bit # Error

- 0 Packet is longer than 1518 octets
- 1 Packet is shorter than 64 octets
- 2 Packet experienced a CRC or Alignment error
- 3 First packet in this capture buffer after it was detected that some packets were not processed correctly.
- 4 Packet's order in buffer is only approximate (May only be set for packets sent from the probe)



ALARM 警报组



ABSOLUTE OR DELTA VALUES

TRIGGERS ON:

- RISING ALARM
- FALLING ALARM
- RISING OR FALLING ALARM



ALARM 警报组

- alarm组定义了一组网络性能的门限值。如果某变量超过门限值时，将会产生一个报警事件并报告控制台。
- 警报组由单个表组成，每个表项指定
 - 被监视的特定变量
 - 采样间隔
 - 相应的阈值参数
- 表中只记录最新的采样数据

alarm (rmon 3)

alarmTable(1) 表中每个记录都定义了一个特定的监视器变量、采样区间和门限值。

alarm(rmon 3)

└─alarmTable(1)

└─alarmEntry(1)

→

- alarmIndex(1) 索引
- alarmInterval(2) 采样区间(s)
- alarmVariable(3) 采样的变量(对象标识符)
- alarmSampleType(4) 采样类型, absoluteValue(1), deltaValue(2)
- alarmValue(5) 最近一次采样中得到的统计值
- alarmStartupAlarm(6) 行生效后的第一次采样值是否产生报警
- alarmRisingThreshold(7) 上升门限
- alarmFallingThreshold(8) 下降门限
- alarmRisingEventIndex(9) 超过上升门限时事件表的索引值
- alarmFallingEventIndex(10) 超过下降门限时事件表的索引值
- alarmOwner(11)
- alarmStatus(12)



ALARM警报组

alarm组的对象

(1) alarmSampleType决定是以采样绝对值还是以差值来与阈值比较。其取值如下。

- ① absoluteValue(1) 表示对象采样时的值直接与门限值进行比较。
- ② deltaValue(2) 代表了对象在两次连续采样时间段内的差与门限值进行比较，它与变化率有关。



ALARM警报组

alarm组的对象

(2) **alarmStartupAlarm**为由管理站设定的警报类型，在监视过程中不变。取值如下。

- ① **risingAlarm(1)** 该行生效后第一个采样值 \geq 上升门限，产生警报
- ② **fallingAlarm(2)** 该行生效后第一个采样值 \leq 下降门限，产生警报
- ③ **risingOrFallingAlarm(3)** 该行生效后第一个采样值 \geq 上升门限或者 \leq 下降门限，产生警报



ALARM警报组

alarm组的报警机制

1. 产生上升警报的规则

(1) 在此行成为有效行后，若第一次采样的值：

- ① $<$ 上升门限值，当第一次采样值 \geq 上升门限时，产生上升警报
- ② \geq 上升门限，且alarmStartUpAlarm=risingAlarm(1)或risingOrFallingAlarm(3)，产生上升警报
- ③ \geq 上升门限，且alarmStartUpAlarm=falingAlarm(2)，那么当采样值降低到低于上升门限后，第一次 \geq 上升门限时，产生上升警报

(2)产生上升警报后，直到采样值下降到下降门限值，然后重新到达上升门限值的时候，才会产生另一次这样的事件。



ALARM警报组

- alarm组的报警机制

2. 产生下降警报的规则与上述过程相反。

- ① $>$ 下降门限值，当第一次采样值 \leq 下降门限时，产生下降警报
- ② \leq 下降门限，且alarmStartUpAlarm=falingAlarm(2)或risingOrFallingAlarm(3)，产生下降警报
- ③ \leq 下降门限，且alarmStartUpAlarm=risingAlarm(1)，那么当采样值升到 $>$ 上升门限后，第一次 \leq 下降门限时，产生下降警报

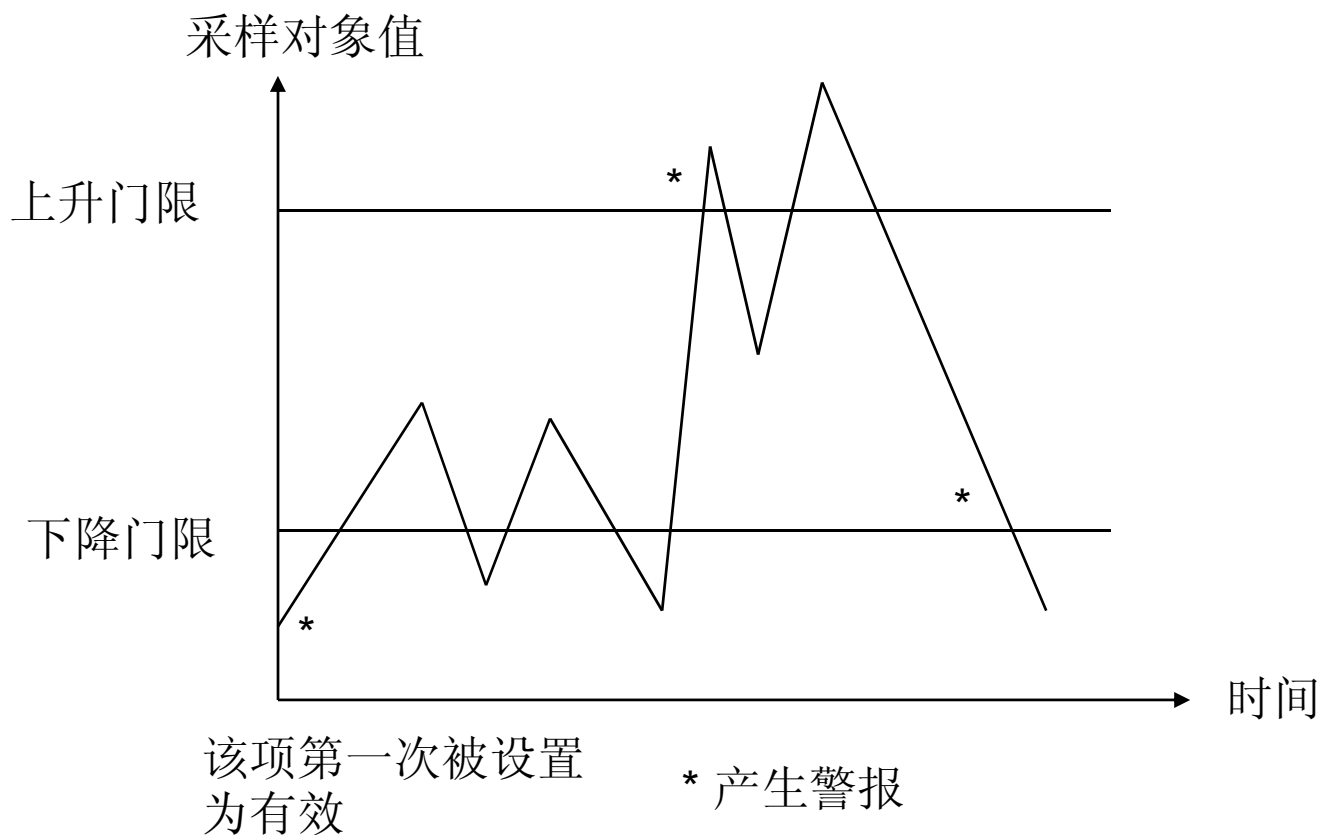
(2)产生下降警报后，直到采样值上升到达到上升门限值，然后重新下降到下降门限值的时候，才会产生另一次这样的事件。

- 警报机制目的：防止在门限附近波动时警报反复产生加重网络负担，又称为hysteresis(磁滞现象)机制。



ALARM 警报组

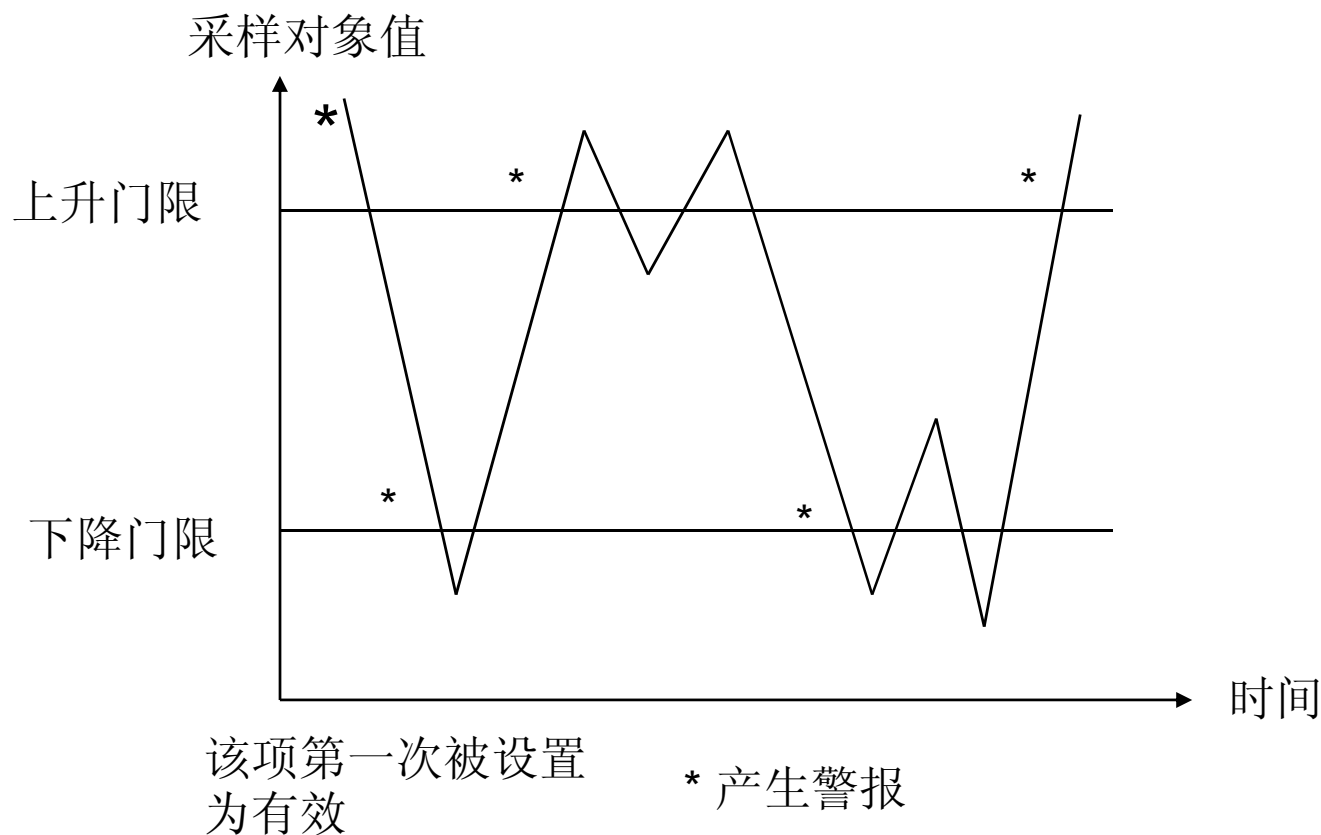
- 例1: $\text{alarmStartUpAlarm}=2$, 发出报警信号的时刻, 如图





ALARM 警报组

- 例2: alarmStartUpAlarm=1或3, 发出报警信号的时刻, 如图





ALARM警报组

当alarmSampleType=deltaValue(2)时为增量报警方式。

双重采样规则：每个周期采样两次，把最近两次采样值的和与门限比较。



ALARM警报组

例，有如下采样值：

时间(s)	0	10	20
-------	---	----	----

观察的值	0	19	32
------	---	----	----

增量值	0	19	13
-----	---	----	----

若上升门限为20，不会报警

若采用双重采样：

时间(s)	0	5	10	15	20
-------	---	---	----	----	----

观察的值	0	10	19	30	32
------	---	----	----	----	----

增量值	0	10	9	11	2
-----	---	----	---	----	---

15秒时有一次报警($11+9=20$)



EVENT 事件组

- event组支持事件的定义。

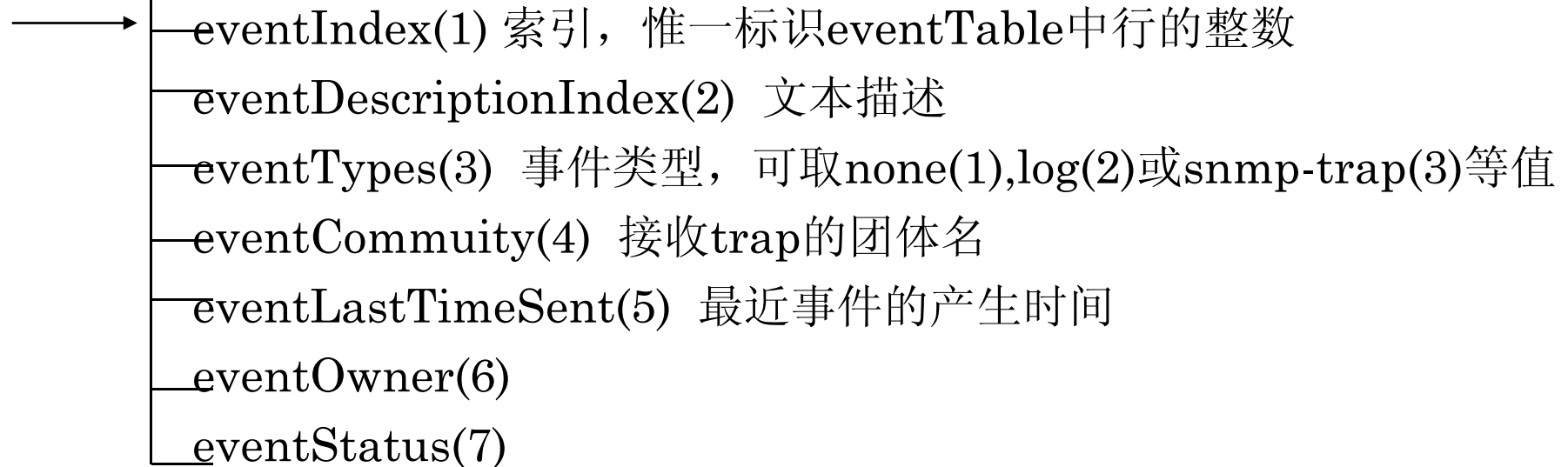
event (rmon 9)

—eventTable(1) 包含事件的定义

—logTable(2) 如果需要对一个事件进行记录，将在相应的logTable表中创建一些表项

eventTable(1)

└─ eventEntry(1)





eventTable(1)


└─eventEntry(1)

→ └─logEventIndex(1) 标识生成该日志表项的事件，与EventIndex相同

└─logIndex(2) 事件记录索引

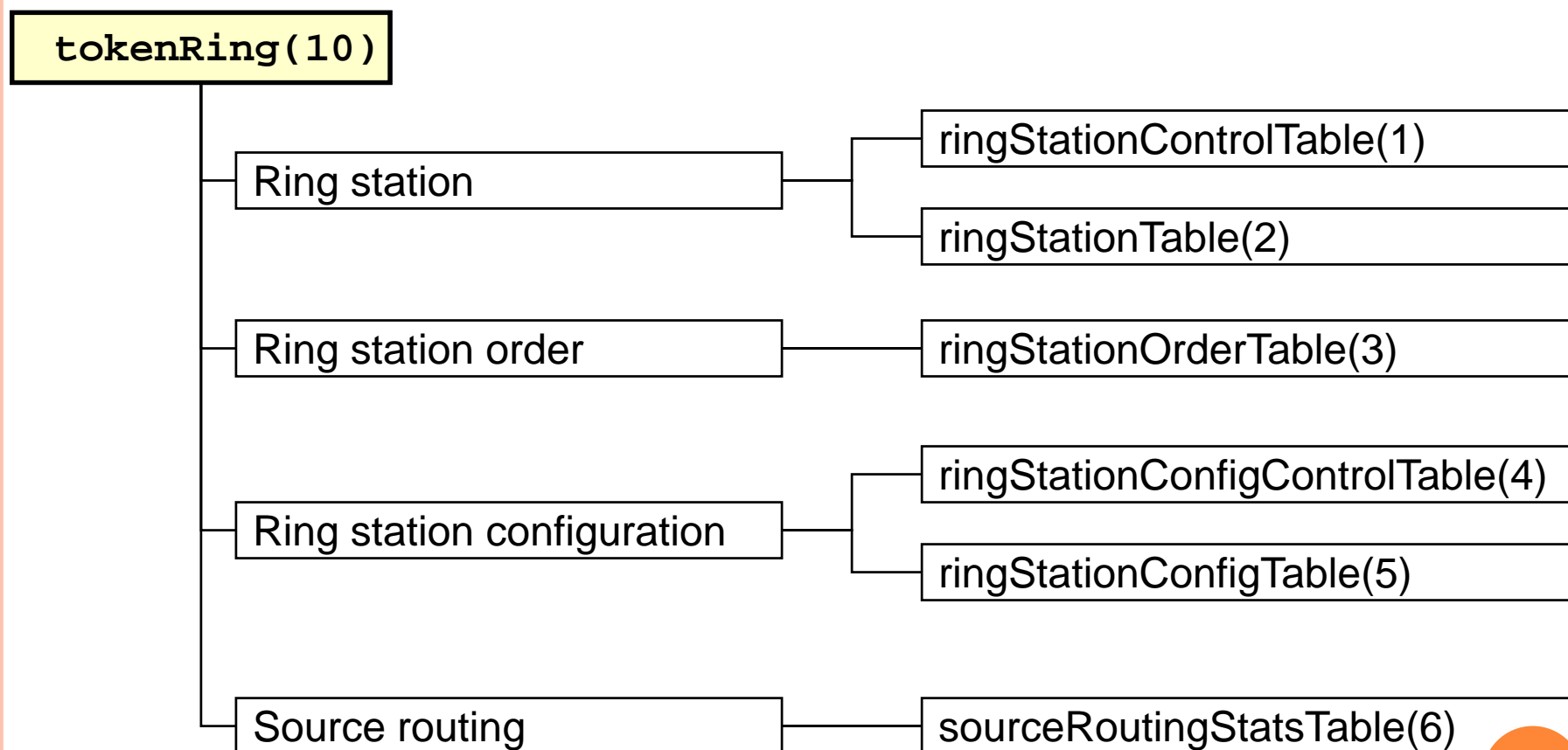
└─logTime(3) 事件记录时间

└─logDescription(4) 与实现有关的文本描述





TOKENRING组





TOKENRING组

- tokenRing组成
 - (1) Ring station: 包含有关每个站的统计数据和状态信息
 - (2) Ring station order: 提供在环上的顺序
 - (3) Ring station configuration: 提供控制环的手段
 - (4) Source routing: 提供源路由信息的使用情况



RMON2 MIB

- RMON1 MIB: 监视 MAC层通信量.
- RMON2 MIB: 监视第3层到第7层的通信。
- 利用RMON2，你能够了解：
 - (1)按照段分析协议；
 - (2)按照网络地址分析协议；
 - (3)为不同网络地址之间的通信分析协议；
 - (4)在应用层一个网络地址分析协议；
 - (5)不同网络地址之间的对话分析应用程序通信。你还可以为该协议和不同网络地址之间的应用层对话生成顶层的N个报表。



RMON2 MIB

1. RMON2 MIB的组成

RMON扩充了RMON MIB，增加了10个新功能组：

- (1) protocolDir(协议目录组)：提供各种网络协议的标准化表示方法。
- (2) protocolDist(协议分布组)：提供每个协议产生的通信统计数据。
- (3) addressMap(地址映射组)：建立IP地址与MAC地址的映像关系。
- (4) nlHost(网络层主机组)：基于IP地址收集网上主机的信息。



RMON2 MIB

- (5) nlMatrix(网络层矩阵组): 记录主机对(源/目标)之间的通信情况。
- (6) alHost(应用层主机组): 对应每个主机的每个应用协议的通信情况。
- (7) alMatrix(应用层矩阵组): 统计一对应用层协议之间的各种通信情况, 或某种参数最大的一对应用层协议之间的各种通信情况。
- (8) userHistory(用户历史组): 按照用户定义的参数周期收集统计数据。
- (9) probeConfig(监视器配置组): 定义了监视器的标准参数集合。
- (10) rmonConformance(一致性规范组): 定义了不同厂商实现RMON2必须达到的最小级别。



RMON2 MIB

2. RMON2的新增功能

- (1) 外部对象索引：RMON2采用了新的表结构，经常使用外部对象索引数据表，以便把数据表与对应的控制表结合起来。
- (2) 时间过滤器索引：网络管理应用需要周期的轮询监视器，以便得到被管理对象的最新状态信息。为了提高效率，使用时间过滤器索引，使监视器每次只返回那些自上次查询以来改变了的值。



其他RMON MIB

- SMON MIB (交换网络监控,包括以下4个不同的组)
 - smonVlanStats 配置和监控VLAN统计表;
 - smonPrioStats 允许配置和监控VLAN 标签控制信息字段 (TCI) 中编码的3比特用户优先级字段的采集;
 - dataSource 描述了数据源以及端口复制能力;
 - portCopy 提供了交换机中复制一个指定源的所有数据帧到另一个指定目的地的能力。
- DSMON MIB(区分服务远程监控)
 - 扩展了RMON 的功能以支持区分服务监控。
 - 通过DSCP计算器, 网络管理器能够监控不同DSCP值对应的网络流量。



本章小结

- 本章首先介绍了RMON的基本概念，然后分别介绍了RMONv1和RMONv2的MIB。需要掌握RMON的概念、作用和功能。
- RMON最初的设计是用来解决从一个中心点管理各局域分网和远程站点的问题，用于监视整个网络通信情况。RMON是对SNMP的补充，扩充了SNMP的MIB-2，基于RMON协议的设备称为网络监视器或网络分析器、探测器等。RMON提供了一种高效、低成本的网络监视方案。



作业

- 1. 什么是RMON? 为什么需要RMON?
- 2. 在RMON规范中增加了哪些新的数据类型?
- 3. RMON是如何解决多个管理站并发访问问题的?
- 4. RMONv2和RMONv1的区别是什么?