Systems Management and the Internet Management Framework

Bobby Krupczak
Empire Technologies, Inc.
541 Tenth St, NW.
Suite 169
Atlanta, GA 30318-5713

1 Introduction

The success, popularity, and importance of using the Internet Management Framework [3, 14, 13] for network management has been unparalleled by any other management framework. While its efforts were originally focused on the monumental task of managing the "network", managing the "systems" *connected to* the network has become increasingly important (Managing the systems has always been important but industry and the IETF focused on network management first mainly because of its more pressing needs.). Indeed, experience shows that without the correct functioning of key systems, the network becomes unusable from a user, or application, perspective. The need for a consistent method for the management of valuable system resources and mission-critical applications has become increasingly urgent as our dependence on distributed computing continues to grow.

Fortunately, the Internet community has made great strides toward specifying a common interface for performing systems management by defining a number of experimental and standards-track MIB modules [1, 2, 8, 9] of which the Host Resources MIB [10], on which we focus here, is particularly important for managing systems and their resources. However, in order to fully support the Host Resources MIB, developers of the underlying systems (OS, device drivers, and system daemons) must provide some means of accessing information.

We focus on the strengths and weaknesses of the Host Resources MIB in regards to systems management and, based on implementation experience presented here, identify those areas which require more consistent support by underlying systems. In addition, we outline extensions made within the context of our own private-enterprise MIB which expand the scope of the Host MIB (targeted at "generic" hosts) to provide detailed management of **UNIX** systems. This article is intended for those interested in network and systems management, and especially those developing hardware, operating systems, and systems applications. It is hoped that these experiences will provide sufficient feedback so that in the future these developers may include better support for accessing systems management information.

This article is organized as follows. First we review systems management and the SNMP in Section 2. Then, in Section 3 we present implementation-based feedback on the Host Resources MIB while Section 4 provides an overview of our UNIX management extensions. Section 5 concludes the article.

2 Systems Management and SNMP

In this section we first define *systems* and *network* management and then provide some motivations for their integration. We then review some of the previous work on systems management within the context of the Internet Management Framework.

The OSI network management literature [5, 6, 7] defines *systems* management as the management of the OSI environment (i.e., the components providing OSI services). It does not necessarily include the underlying hardware and systems software supporting the provision of OSI services. While much of the literature has used the terms *network* and *systems* management interchangeably, they have taken on different meanings than that used in the OSI literature. We utilize the term *network* management to include the management and operation of a communications subnetwork and the services it provides. Included in this category are the network interface cards, the protocol software, routers, bridges, etc. We use the term *systems* management to denote the management and operation of the systems in which networked components reside. Included in this category are the underlying operating system, the hardware and devices on which the system operates, its applications, file systems, and its system daemons.

The proliferation of scientific and engineering workstations running complex operating system software has created a nightmare for system administrators. Each operating system defines its own set of system administration utilities and tools, thus requiring administrators to learn multiple interfaces and commands. This lack of a consistent means of management results in increased costs due to training, additional administration personnel, and vendor-specific administration software. However, through the integration of systems and network management, administrators can leverage a common, interoperable and standard interface for the administration of systems. Further, their intergation can also permit the leveraging of network management software to perform systems management functions. Lastly, this integration can result in a reduced burden on system support staff and, ultimately, lower management costs.

Interest in systems management via SNMP has been rising since the successful deployment of MIB-II (as early as 1991) although not much has been published regarding its use. However, a few MIBs and papers have been published. The overall applicability of the Internet Management Framework to systems management is explored in [11] along with a discussion on a UNIX systems management MIB. The Host Resources MIB [10] (See Figure 1) defines a set of managed objects useful for systems management. It is defined to be operating system independent and contains objects defined for use in the monitoring of devices, storage areas, file systems, and installed software. An excellent overview of this MIB and its potential use can be found in an earlier *Connexions* article [16]. The monitoring of critical applications has been proposed in several experimental, application-specific MIBs [8, 12, 1, 2, 9]. However, as of yet, no general, flexible framework has been proposed for application management.

3 Implementation Experiences

In this section we present experiences gained during the implementation of the Host Resources MIB and a private-enterprise UNIX management MIB on two of the major flavors of UNIX (BSD and System V).

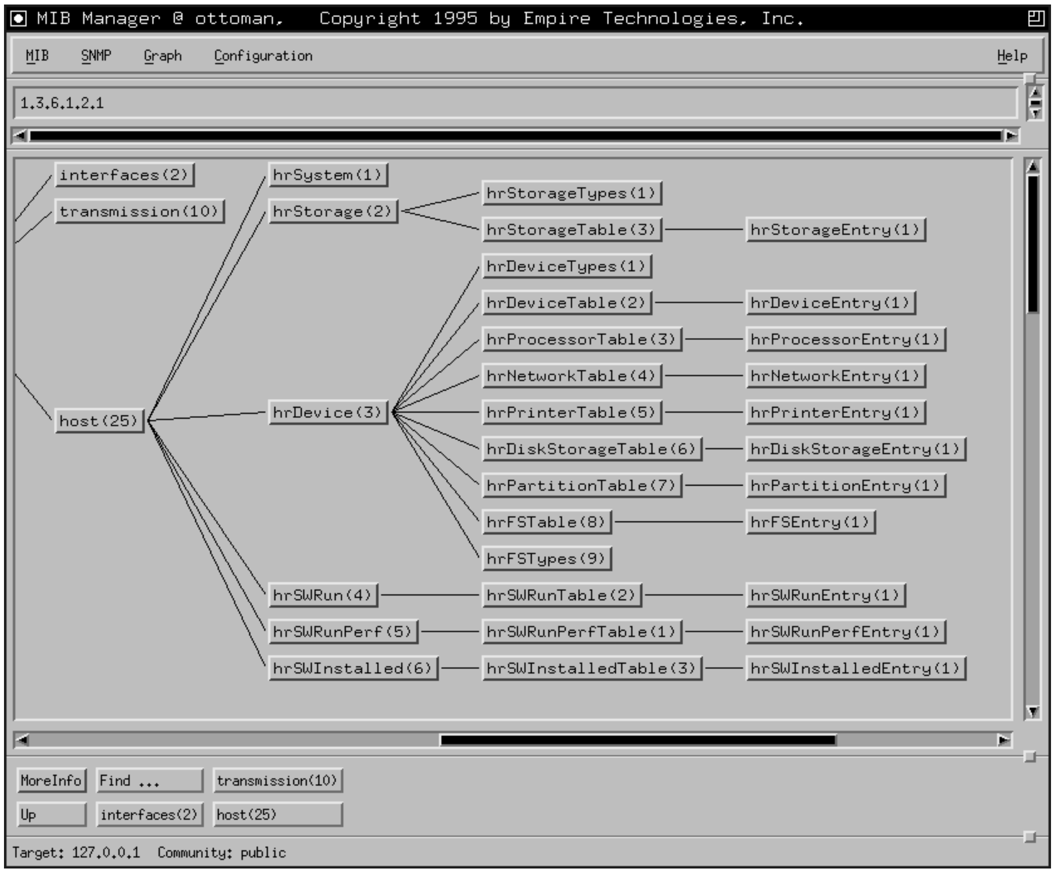

Figure 1: Host Resources MIB, RFC 1514

This implementation has been deployed across a large number of systems in a variety of configurations. We first concentrate on the Host Resources MIB and then discuss its extensions made within our own private-enterprise MIB in Section 4. Again, while an earlier article ([16]) presents the many benefits of the Host Resources MIB, we mainly focus on the experiences gained during its implementation on real systems.

3.1 The Host Resources MIB

Device Table The centerpiece of the Host Resources MIB is the *hrDevice* group and table. This group and table, which provide information about the devices (and their status) contained within a managed system, are extremely useful for asset management. However, complete support for this group is entirely dependent on the underlying operating system and its support for an API to the kernel's information on the hardware and devices currently in operation. While an application can discover some devices by looking through certain operating system configuration files (e.g. */etc/mnttab* for disks and partitions) or by simply attempting to access known devices, this strategy is unreliable and inherently non-scalable. Accessing static configuration files is often incomplete and prone to errors introduced by configuration changes. Attempting to access known devices is inherently non-scalable since the agent can only report devices it knows about at compile-time. *Some* operating systems *do provide* access to the resources known by the operating system. On these systems, agents can rely on the operating system to inform them of all known devices and their configuration. However, current operating systems do not generally provide runtime access to each device's description, manufacturer, version, etc. – information important for asset management. Some devices, notably SCSI disks and tape drives, do provide such access through well-defined APIs. It is our belief that all devices should provide some well-known API for accessing basic information about its manufacturer, revision, type, etc. and that operating systems should make all relevant device information available to application or systems programmers. On UNIX systems, a system defined *ioctl* would probably be sufficient.

Buffers and the Storage Table The Host Resources MIB defines a *hrStorage* group and table designed to provide information about the logical storage areas contained within a system. However, the integration of buffer subsystem statistics into the *hrStorageTable* is problematic due to several factors. First, the *hrStorageTable* does not really define a storage type for operating system buffers; a MIB implementor must choose between random-access memory (RAM) or the catch-all "other" for the storage type. Choosing the storage type "RAM" could cause confusion with the system's total RAM, while choosing "other" does not provide much semantic information. Further, if one chose RAM for the storage-type, there could be overlap if different subsystems define buffers of the same size. For example, in SunOS systems, buffer statistics for the Streams and BSD network subsystems as well as the general I/O buffer cache could all be supported. Clumping all buffer information together as RAM would be confusing and could result in the loss of management information. One solution would be to define additional storage types specifically for buffers.

MIB-II Correlation The *hrNetworkTable* defines a table for network devices contained within the system. One of its columns, *hrNetworkIfIndex*, identifies the network device's corresponding MIB-II *ifTable.ifNumber* value. The Host Resources MIB *does not* define a return value to use if this network device has no corresponding MIB-II *ifTable* entry. For example, many SUN workstations contain an ISDN interface which, technically, is a network interface but many times is not contained in the MIB-II *ifTable* because it is not currently used as an IP network interface. In our implementation, we return the value of 0 in such a case. The MIB specification should be modified so that a value of 0 indicates that a network interface is not currently contained in the *ifTable*.

System Configuration The Host Resources MIB defines only a handful of system configuration variables within its *hrSystem* group. Included are the initial boot/load device, the number of users, and the maximum number of processes allowed by the system. In our experience, many more configuration variables are possible and needed to properly manage a system (regardless of its operating system). One alternative is to try to encorporate these variables into the *hrStorage* table. For example, one could construe file and inode lists (or, for example, their DOS and Windows equivalents) to be storage areas in the system and include their allocation information in the *hrStorage* table. However, the same problems arising from the introduction of buffer information in the *hrStorage* also arise in this case as well.

Installed Software The Host Resources MIB defines a table (*hrSWInstalledTable*) containing an entry for each software package installed on the system. There are two problems with its design and implementation. First, the table does not contain a column for the software package's version. Instead, the implementor must combine the software version (if it is known) with the package's name. This oversight hinders automated management since human intervention will probably be necessary in order to interpret what portion of the software package's name is the version. Unfortunately, type-casting software versions into SNMP defined types may not be easy since vendors often use incompatible numbering systems.

Second, not all operating systems (e.g. SunOS and other BSD derived systems) support a unified software installation and packaging format. One work-around is to attempt to do a software "discover"; however, it is a time-consuming task and one is then left with the problem of deciding what is a software package and what is not. Does the agent include shell scripts and system binaries as software packages? Microsoft Windows performs a software "discover" by comparing files it finds against a master list. While this strategy might be practical on PCs with limited disk space, it is not scalable to multi-user systems with large amounts of internal disk storage. Further, with such a strategy, the master list of known software is frozen at the time the agent is distributed. In order to provide support for newer software packages, the agent (or its software list) would also need to be updated.

ProductID Variables The Host Resources MIB defines an object identifier textual convention, *ProductID*, that is intended to identify the manufacturer, model, and version of a hardware item or software package. At present, few hardware or software vendors support it; indeed, conformance with this aspect of the MIB was cause of some debate within the original working group. If vendors begin to comply, it is important that they support the ability to dynamically determine the *ProductID* so as to avoid requiring agents to maintain lists of *ProductID* to vendor mappings. For UNIX systems, a well-known *ioctl* could be defined such that its return value is a device or kernel module's *ProductID*. Application software should support *ProductID*'s as part of installation information.

Disk, Partition, and File system Tables The Host Resources MIB defines tables containing disk information, disk partitions, and file systems residing within the managed system. Starting with the disk table, managers may then examine that disk's partitions and then examine a given partition's file system. In order to properly support all disks (not just those that have file systems and are mounted), the operating system must provide information about all known devices. Second, they must support manufacturer-independent methods of obtaining configuration information such as capacity, file system layout, etc. Lastly, the *hrFSTable* defines columns identifying the last full and partial backup dates. Unfortunately, there are a myriad (e.g. *bar*, *tar*, *dump*) of backup programs and techniques which each store backup times (if at all) in an incompatible manner. What is needed is a consistent method for storing backup times for partitions and file systems. One solution would be to have a partition's backup times stored within the disk's label along with the normal partition and cylinder information.

Traditional Agent Paradigm The Host Resources MIB continues the tradition of defining MIB objects without endowing the agent or MIB (depending on your perspective) with "intelligence" for distributed self-management. Other MIBs, most notably the Remote Network Monitoring MIB[15] and the Manager-to-Manager MIB[4], are able to endow agents with intelligence, yet still conform to the philosophies of the Internet Management Framework. The Host Resources MIB, however, defines no traps or self-monitoring capabilities; consequently, managers must (perhaps unavoidably) poll the agent.

Advantages Despite these issues, the Host Resources MIB is **extremely** useful for managing systems. Its device and installed-software tables are excellent for asset tracking and management. Its disk, partition, and file system tables coupled with the Host Resources system group are useful for configuration management. Finally, its "standard" nature makes it easier for management station applications to code towards a vendor-independent format.

4 Host Resources MIB Extensions

Because the Host Resources MIB specification focuses on those aspects common to all computer systems, it may not completely satisfy the management requirements of a particular system (e.g. UNIX). Consequently, we have evolved our own private-enterprise MIB to extend the systems management capabilities of the Host Resources MIB. More specifically, we have geared (at present) our MIB specification to the task of managing UNIX and UNIX-like operating systems. Our extensions have been twofold. First, we have defined and implemented additional MIB objects important to the management of UNIX systems. Second, we have begun defining and implementing "RMON"-like extensions for systems management. We discuss these complimentary directions below.

Our first task was the definition and implementation of a range of MIB objects important to the management of UNIX and UNIX-like operating systems. We added extensive kernel configuration parameters covering everything from file table sizes, swap space, and the BSD and Streams I/O subsystems. Next, we added extensive process information to augment that contained in the Host Resources *hrSWRunTable* and *hrSWRunPerfTable*. For example, we report a process' owner and group as well as its process-ID, size, and nice value. Because of the importance of memory buffers, we created a separate group for buffer statistics and separated them by their respective subsystems (e.g. *strbufs* and *mbufs*). We also added information about a system's valid users and groups as well as information about who is currently logged on and using the system. Since information about valid users and groups can be sensitive (for security purposes), we also added the ability to selectively "turn on or off" support for these groups (More sophisticated MIB views within the context of the SNMPv2 administration model would be a more general solution though). While the Host Resources MIB tends to focus on configuration and asset management, we added support for performance management by adding MIB objects for tracking kernel, disk, and memory performance statistics. Lastly, we added a range of counters and statistics necessary for monitoring NFS and RPC services due to their importance in distributed environments.

Our second task was the definition of "RMON"-like extensions for systems management whereby we endow agents with intelligence, yet still conform to the goals of the Internet Management Framework. Our motivations are based primarily on the need to provide a scalable systems and network management solution to large, distributed networks of workstations. By endowing an agent with intelligence and autonomy, we can reduce network and system load due to polling and instruct the agent on how to behave in certain circumstances. Our “RMON”-like extensions have taken (so far) two forms.

First, we have defined a monitor table (roughly equivalent to the RMON’s *alarmTable*) that enables a manager to instruct an agent to monitor certain MIB objects and to send a TRAP message when some threshold is crossed. Each monitor entry defines a boolean expression, that when evaluated to True, indicates that an event has occurred. For example, using the monitor table, a manager can instruct an agent to send a TRAP message when a certain disk or file system becomes 95% full; alternatively, a manager can instruct an agent to send a TRAP when an important process (e.g. sendmail) dies or becomes a zombie. Second, we are defining and implementing a history mechanism whereby a manager can instruct an agent to sample and store the value of a MIB object over time. Like the RMON MIB[15], we have defined history and history control tables and the notion of sample buckets. Sample buckets serve to place an upper bound on the resources the agent devotes to a particular history control entry. For example, using this facility, managers can instruct the agent to sample and record the length of the run queue, the amount of network I/O, the size of a given process, or the amount of allocated and free buffer space over some interval. This type of functionality (monitoring and recording) is crucial for the proper tuning of UNIX systems. Although we have initially geared our MIB towards the task of managing UNIX, there is nothing in our “RMON”-like extensions that is necessarily specific to UNIX. Consequently, these extensions may be leveraged for the management of other operating systems. Finally, because these extensions have been patterned after the RMON MIB, we hope to leverage RMON management software.

5 Conclusion

The Host Resources MIB is a great start toward the goal of integrating network and systems management. What we have focused on is experiences gained from its implementation on two major flavors of UNIX. However, due to its goal of being generic, some potential manageability of UNIX systems is not realized. Consequently, we have evolved our own private-enterprise MIB by adding MIB objects more specific to the task of managing UNIX and by defining “RMON”-like functionality for managing systems. It is hoped that these experiences will guide future system developers in making their systems more “management-ready” for those implementing systems management agents.

References

[1] R. Austein and J. Saperia. DNS server MIB extensions. Anonymous FTP, May 1994. RFC 1611.

[2] D. Brower, R. Purvy, A. Daniel, M. Sinykin, and J. Smith. Database management system (RDBMS) management information base (MIB) using SMIv2. Anonymous FTP, August 1994. RFC 1697.

[3] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin. Simple network management protocol. Anonymous FTP, May 1990. RFC 1157, obsoletes RFC 1098.

[4] Jeffrey D. Case, Keith McCloghrie, Marshall T. Rose, and Steven L. Waldbusser. Manager-to-manager management information base. Anonymous FTP, April 1993. RFC 1451.

[5] Information processing systems: Open systems interconnection basic reference model part 4: Management framework, October 1989. ISO/IEC 7498-4.

[6] Information technology: Open systems interconnection: Systems management - part 1: Object management function, August 1991. ISO/IEC 10164-1, CCITT X.730.

[7] Information technology: Open systems interconnection: Systems management overview, August 1991. ISO/IEC 10040, CCITT X.701.

[8] N. Freed and S. Kille. Mail monitoring MIB. Anonymous FTP, January 1994. RFC 1566.

[9] N. Freed and S. Kille. Network services monitoring MIB. Anonymous FTP, January 1994. RFC 1565.

[10] P. Grillo and S. Waldbusser. Host resources MIB. Anonymous FTP, September 1993. RFC 1514.

[11] B. Krupczak. Unix systems management via SNMP. In *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management*, April 1993.

[12] G. Mansfield and S. Kille. X.500 directory monitoring MIB. Anonymous FTP, January 1994. RFC 1567.

[13] M. T. Rose. Concise MIB definitions. Anonymous FTP, March 1991. RFC 1212.

[14] M.T. Rose and K. McCloghrie. Structure and identification of management information for TCP/IP-based Internets. Anonymous FTP, May 1990. RFC 1155, obsoletes RFC 1065.

[15] Steve Waldbusser. Remote network monitoring management information base. Anonymous FTP, November 1991. RFC 1271.

[16] Steven L. Waldbusser. A look at the host resources MIB. *ConneXions: The Interoperability Report*, 7(11):42–43, November 1993.

Bobby Krupczak is Chief Scientist at Empire Technologies, Inc. where he designs and implements intelligent network and systems management agents and managers. He regularly participates on the InteropNet*TM* NOC team where has gained valuable insight into the art of network and systems management of large, heterogeneous networks. He can be reached at **rdk@empiretech.com**. For more information, please consult Empire's homepage at **http://www.empiretech.com/empiretech/**.