SNMP Extensions for a Self Healing Network

Background

Patent 6,088,141: This is a self healing network depending on additional hardware. It requires a second ring of connection to handle recovery operations. Modeled after the token ring network wherein the network has two rings, one as a backup.

Patent 5,235,99: This patent is about failed transmissions and restoring those transmissions.

Automatic Healing-Based Self Management for Network Management in Hybrid Networks (http://www.springerlink.com/content/435233484k12g521/) This is a server based solution wherein the software monitors SNMP messages sent from agents and then based on that information, responds.

Policy-Based Mobile Ad Hoc Network Management (http://www.computer.org/portal/web/csdl/doi/10.1109/POLICY.2004.1309148) paper describes a policy-based mobile ad hoc network management system. The system provides the capability to express networking requirements in the form of policies at a high level and have them automatically realized in the network by intelligent agents. This system is strictly monitoring, it takes no corrective action.

Simple Network Management Protocol: as this invention is an extension of SNMP, the various SNMP standards would constitute related prior art.

RFC 1065 — Structure and identification of management information for TCP/IP-based internets
RFC 1066 — Management information base for network management of TCP/IP-based internets
RFC 1067 — A simple network management protocol

These protocols were made obsolete by version 2 of SNMP defined in these RFC's:

RFC 1155 — Structure and identification of management information for TCP/IP-based internets
RFC 1156 — Management information base for network management of TCP/IP-based internets
RFC 1157 — A simple network management protocol
RFC 1156 (MIB-1):
RFC 1213 — Version 2 of management information base (MIB-2) for network management of TCP/IP-based internets

Discussion

Introduction

SNMP is currently used to monitor networks. The current invention (SNMP Extensions for a Self Healing Network) is about extending the SNMP protocol to allow the creation of networks that are self healing. For that reason we should begin with a discussion of SNMP.
Simple Network Management Protocol (SNMP): This is a UDP-based network protocol. It is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention. An SNMP-managed network consists of three key components:

- Managed device
- Agent — software which runs on managed devices
- Network management system (NMS) — software which runs on the manager

SNMP itself does not define which information (which variables) a managed system should offer. Rather, SNMP uses an extensible design, where the available information is defined by management information bases (MIBs). MIBs describe the structure of the management data of a device subsystem; they use a hierarchical namespace containing object identifiers (OID). SNMP operates in the Application Layer of the Internet Protocol Suite (Layer 7 of the OSI model). The SNMP agent receives requests on UDP port 161. The manager may send requests from any available source port to port 161 in the agent. The agent response will be sent back to
the source port on the manager. The manager receives notifications (Traps and InformRequests) on port 162. The agent may generate notifications from any available port. All SNMP PDUs are constructed as follows:

IP header	UDP header	version	community	PDU-type	request-id	error-status	error-index	variable bindings

The seven SNMP protocol data units (PDUs) are as follows:

GetRequest: Retrieve the value of a variable or list of variables. Desired variables are specified in variable bindings (values are not used).

SetRequest: Change the value of a variable or list of variables. Variable bindings are specified in the body of the request.

GetNextRequest: Returns a Response with variable binding for the lexicographically next variable in the MIB. The entire MIB of an agent can be walked by iterative application of

GetNextRequest starting at OID 0. Rows of a table can be read by specifying column OIDs in the variable bindings of the request.

GetBulkRequest: optimized version of GetNextRequest. Requests multiple iterations of GetNextRequest and returns a Response with multiple variable bindings walked from the variable binding or bindings in the request.

SNMP version two expanded this list to include the following:

Response Returns variable bindings and acknowledgement for GetRequest, SetRequest, GetNextRequest, GetBulkRequest and InformRequest. Error reporting is provided by error-status and error-index fields. Although it was used as a response to both gets and sets, this PDU was called GetResponse in SNMPv1.

Trap: Asynchronous notification from agent to manager. Includes current sysUpTime value, an OID identifying the type of trap and optional variable bindings.

SNMP can be used to gather extensive information about various elements on a network. That is depicted in figure 1-1.

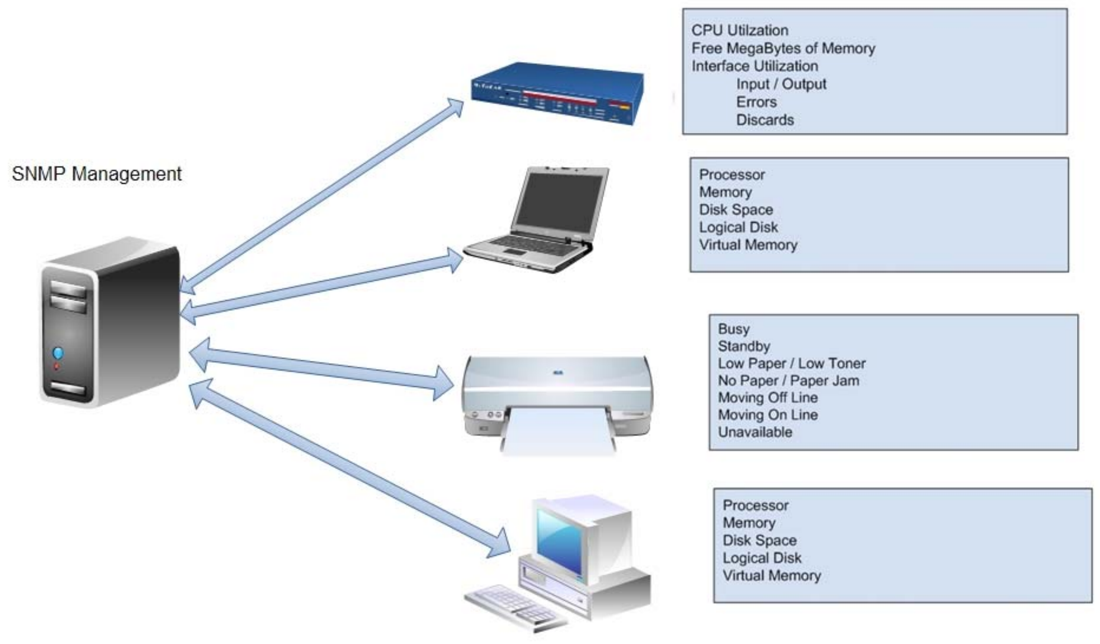

Figure 1-1 SNMP Basic Functionality

It should also be noted that SNMP itself does not define which information (which variables) a managed system should offer. Rather, SNMP uses an extensible design, where the available information is defined by management information bases (MIBs). MIBs describe the structure of the management data of a device subsystem; they use a hierarchical namespace containing object identifiers (OID). Each OID identifies a variable that can be read or set via SNMP. The essence of this invention is to

a. Expand the Management Information Blocks
b. Allow the agents to act on the information found.

What the current invention proposes is extending SNMP beyond merely reporting data, but rather making modifications to the system to improve network health. This would require modifying the SNMP agents that are placed on each client that is being monitored.

SNMP agents currently have a number of 'get' methods that allow the SNMP management server to query the agent for data. This invention necessitates adding at least one 'set' method.
In one embodiment of this invention there would be a single set method with a flag that determines what is to be set, and in the SNMP agent set functions different action would be taken based on that flag. Pseudocode for that is shown here:

```
Function SetClientValue(integer flag)

  If flag = 1 ' i.e. virtual memory low
      Increment virtual memory/swap file size to max allowed
  If flag =2 ' i.e. low memory
     Unload the service or application with the lowest priority
  If flag = 3 'i.e. excessive errors on one port of router/switch
      Attempt to route traffic through a different port
   If flag = 4' i.e. excessive traffic on one port of router/switch
      Attempt to route traffic through a different port.
```

In another embodiment of this invention, some, or all of these tasks would first prompt the user for permission to unload. For example if the memory was too low, then the user would be notified that they are low on memory and would be told what the lowest priority service or application is, and asked if they wish to unload that to improve memory performance.
In yet another embodiment of the invention, there would be separate set functions for each activity. Pseudocode for that is shown here:

```
Function SetVirtualMemory(integer level)
 Note: this could also be implemented as function that takes a value that is a percentage
whereby to increase memory, or taking no arguments and simply taking virtual memory
to the maximum allowed.
Function SetMemory()
Function ReRouteTraffic()
Note: this could also be implemented as a function that takes a value determining which
alternate port to use.
```

In one embodiment of this invention the SNMP agents would be queried from the SNMP management server and return the requested data (as per normal SNMP). The SNMP server would then check that returned data against a table or list of acceptable values. If the reported values for a given parameter exceeded acceptable limits, then a command would be issued back to the agent to alter that parameter. This is shown in figure 1-2.

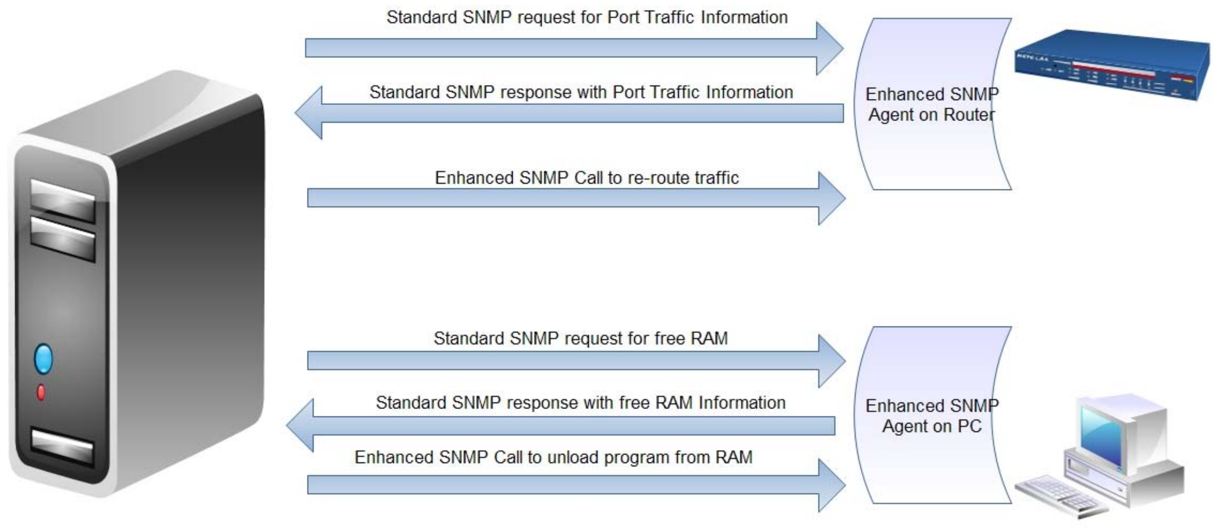
Standard SNMP request for Port Traffic Information
Standard SNMP response with Port Traffic Information
Enhanced SNMP Call to re-route traffic
Enhanced SNMP Agent on Router
Standard SNMP request for free RAM
Standard SNMP response with free RAM Information
Enhanced SNMP Call to unload program from RAM
Enhanced SNMP Agent on PC

Figure 1-2 Communication with the SNMP self healing server

Acceptable limits and the action to be taken would be set by a network administrator during setup.

In another embodiment of this invention, the individual agents would be programmed to take corrective action when certain threshold limits are met, without first having a centralized server initiate action. In that scenario the server would still be notified of the change, so there is a centralized record of changes that have been made. This is shown in figure 1.3.

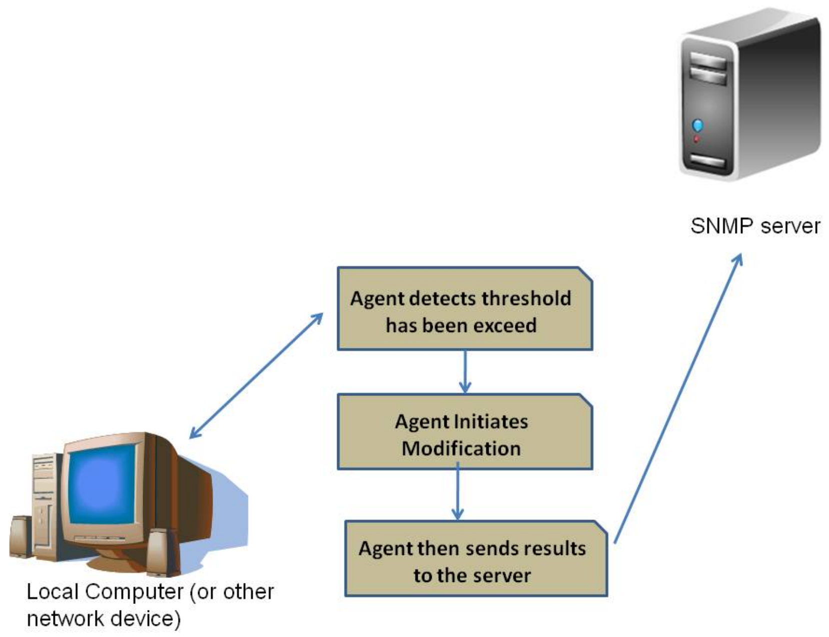
SNMP server
Agent detects threshold has been exceed
Agent Initiates Modification
Agent then sends results to the server
Local Computer (or other network device)

Figure 1.3 Local Agent corrects problem

In any embodiment of this invention, there are certain requirements that the agents must meet.

1. They must run as a service (Windows) or a daemon (Unix/Linux). This allows them to be constantly running and monitoring the system.
2. They must run with enough privileges to make adjustments to the local system. The level of privilege required would very much depend on the types of issues the agent would be expected to execute.
3. It is recommended that communication between the agents and the SNMP server be encrypted. One major current weakness to the current SNMP is that it sends data in clear text that can be intercepted and read. Version 3 of SNMP now includes encryption and authentication. The preferred embodiment of this current invention would also include encryption and authentication.

In another embodiment of the invention it would be expanded so that it could monitor application changes, operating system changes, and hardware changes. This is shown in figure 1-4.

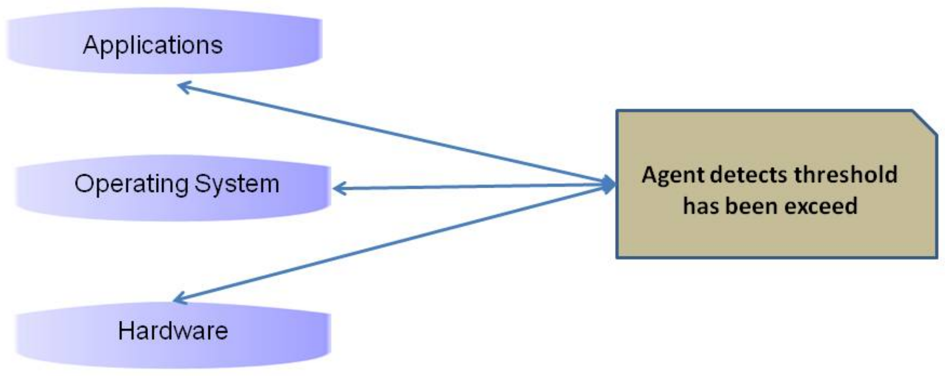

Figure 1-4 Monitoring Hardware, Software, and Operating System

The issue is how specifically the SNMP agents would go about modifying various parameters to return the system to acceptable levels. This will, of course, vary in different embodiments, but some general guidelines can be explained.

Memory Issues: If a system is having difficulty due to memory issues, there are any number of corrective actions that can be taken by software agents. Those include but are not limited to:

1. Expanding Virtual Memory/Swap file
2. Removing multiple versions of the same process. This is particularly important in Windows were it is not uncommon for an application to close, but accidently leave a process running. The extended SNMP agents could monitor running processes and applications in several ways including:
    a. When an application first launches, record the various processes that launch. Then when the application is unloaded ensure those matching processes are also unloaded. This would take corrective action to handle programming flaws in the various applications.
    b. Search running processes for duplicate processes. It must be noted that certain Windows processes do run multiple instances. However the SNMP agents could be programmed with a list of processes that would be allowed multiple instances.

3. Unloading one or more applications based on least used. In other words if the system is running out of memory, whatever currently running application is being used the least, will be unloaded from memory.

Bandwidth Issues: If a router is equipped with the expanded SNMP agents, the agents would be able to monitor for bandwidth problems. In the case that bandwidth is being exceeded, there are a number of actions an agent could take:

1. If the excessive activity is on a small number of its ports, then it could use the routing maps that all internal network routers have, and direct some of the traffic via a different path.
2. If the excessive activity is from a single application, the agent could notify the network administrator and give them the option of shutting down that application.

In yet another embodiment of this invention, agents on routers and switches would exchange utilization information so that traffic could be more effectively routed to avoid congestion in the first place. Essentially the router agents could exchange with each other (and the SNMP server) details about traffic patterns. This would allow eventual extensions to this invention that would have the centralized server generate statistical models of network usage and route network traffic accordingly. This is much like the concept used in automobile traffic routing software.

The traffic between the agents and the extended SNMP service can be handled with any protocol, however standard SNMP uses UDP, a connectionless protocol, in order to reduce overhead. This would be appropriate for the extended SNMP as well.