Managing Agent Platforms with SNMP

Research Proposal

Candidate
Brian D. Remick

Committee
Robert Kessler, Ph.D.
Matthew Flatt, Ph.D.
Dick Cowan

October 26, 2001
# Table of Contents

1. Introduction .................................................................................................................. 4

2. Motivation ...................................................................................................................... 4
   2.1 Complexity of Agent Platforms ................................................................................. 4
   2.2 Lack of Formal Management Specification ............................................................... 5

3. Management Requirements ............................................................................................ 6
   3.1 Standard Management Tasks .................................................................................... 6
   3.2 Load Balancing .......................................................................................................... 7
      3.2.1 Defining Agent Load .......................................................................................... 7
   3.3 Agent Mobility ............................................................................................................ 9
   3.4 Agent Communication ............................................................................................... 10
   3.5 Unresolved Requirements ......................................................................................... 10
      3.5.1 Security .............................................................................................................. 10
      3.5.2 Advanced Mobility (Agent Scope) ..................................................................... 11

4. Network Management Background ............................................................................... 11
   4.1 Proxy Agents .............................................................................................................. 11
   4.2 Protocols .................................................................................................................... 12
   4.3 Management Techniques .......................................................................................... 13
      4.3.1 Centralized Management .................................................................................... 13
      4.3.2 Decentralized Management ............................................................................... 15

5. Comparing Agent Systems to Networks ...................................................................... 15

6. Research .......................................................................................................................... 17

7. Approach ........................................................................................................................ 18
   7.1 Management Interface Definition ............................................................................... 18
   7.2 Proxy Agent Implementation ...................................................................................... 19
   7.3 Management System Integration ................................................................................. 19
      7.3.1 MIB Browser ....................................................................................................... 19
      7.3.2 Enterprise Management Systems ........................................................................ 20
   7.4 Managing Patterns of Agent Interaction .................................................................... 20
      7.4.1 Platform Load Pattern ....................................................................................... 21
      7.4.2 Client/Server Pattern .......................................................................................... 21
      7.4.3 Agent Mobility Pattern ....................................................................................... 21

8. Assumptions and Limitations ....................................................................................... 21
   8.1 FIPA Platform Specification ...................................................................................... 22
8.2 Access to Managed Data ........................................................................................................ 22

9. Technical Architecture ........................................................................................................... 23
  9.1 Agent Platform Layer ........................................................................................................ 24
  9.2 AgentX Master Agent ...................................................................................................... 25
  9.3 Management Layer .......................................................................................................... 25

10. Related Research .................................................................................................................... 26
  10.1 Moving Toward Decentralized Management ................................................................. 26
  10.2 Using Mobile Agents as Proxy Agents ........................................................................ 27

11. Schedule for Completion ...................................................................................................... 27

12. Bibliography .......................................................................................................................... 28
1. Introduction

Agent systems are quickly becoming a popular means of implementing complex distributed systems. Their ability to communicate independently without a central control mechanism and their ease of implementation makes them an attractive software solution to many scenarios.

The need to manage these systems is already becoming increasingly important. Details of agent communication and overload conditions are difficult to find in such a distributed environment, and general software debugging tools fail to provide adequate context of the system as a whole. Agent-based systems have an inherent tradeoff in that, while being extremely flexible, the entire system is often difficult to manage.

This research will attempt to show that network management techniques, coupled with SNMP (Simple Network Management Protocol) [3], provide a solution to this problem. The flexibility of the protocol as well as the availability of powerful, off-the-shelf enterprise management packages that already support SNMP make it an ideal fit for managing agent-based systems.

2. Motivation

The motivation behind this research stems from the following areas:

- Complexity of agent platforms
- Lack of a formal management specification

2.1 Complexity of Agent Platforms

While software systems that utilize agent-based technology benefit from its flexibility, they quickly become extremely complex. The need to manage these systems both during development and use is a requirement.
The idea of formally managing an agent system is relatively new. Since agent-based systems are used most often in research (so far), management needs arise from the software developers themselves, who are interested in watching the interactions between agents as well as the system load. Developers often find that a traditional debugger does not have the context necessary to show them the information they need. The result is typically a custom management implementation that is tied closely to their specific agent system as well as the specific information they need to manage. While this might solve the individual developer’s needs, other developers will undoubtedly encounter the same scenario in other systems, but they won’t be able to easily re-use the same management software.

2.2 Lack of Formal Management Specification

FIPA (Foundation for Intelligent Physical Agents) does not define any formal external management scheme for an agent platform. In fact, FIPA specifies that issues of management are implementation-specific and thus are not addressed in the high-level architecture of the system. [1]

The Agent Management System (AMS), a central component of each agent platform, provides supervisory control over the system, providing methods for registering new agents, naming agents, etc. However, the AMS does not expose any management functionality to external clients. (For instance, there is no standardized means of querying a platform for its current state.) This has led to the development of platform-specific mechanisms to access this type of information. The Jade [17] agent platform implementation, for example, has developed an Introspector agent that provides feedback on platform events and agent communication. However, this type of functionality is only available via platform-specific additions to the AMS standard, which cannot be easily generalized to a standard management scheme.
While the FIPA architecture is correct in assuming that managing an agent platform is dependent on implementation details, the lack of a standard has led to inconsistent management techniques. This research will formalize the concept of managing an agent platform and attempt to provide a standard solution that will work for all FIPA implementations.

3. Management Requirements

In order to be effective, the research must meet the basic requirements of managing an agent system and be flexible enough to allow more specific management scenarios to be easily adapted. The following are proposed requirements that are essential to an agent management system, regardless of its implementation. Note that these are tied closely to the scenarios that will be tested in this research.

3.1 Standard Management Tasks

A manager should be able to perform some basic operations on agents within a platform:

- Obtain the configuration of a platform, including information about the number of containers that exist in the platform and a list of agents living in each container.
- Create new agents.
- Destroy agents.
- Change the state of an agent, such as pause execution.
- Add/remove containers from a platform.
- Add/remove behaviors from an agent.
- Move agents between containers.
Other basic operations are mentioned in the sections below as part of other management concepts. In the current FIPA specification, only the first operation listed above is supported directly by the AMS, and even then in very limited form. Other operations are implementation dependent, and most platforms do not provide a complete set of these operations to an outside software client.

3.2 Load Balancing
A management system needs to be able to identify agents within a platform that are degrading the performance of the platform and make changes to achieve balance within the system. This abstract idea is defined as load balancing in this research.

3.2.1 Defining Agent Load
Agents that fit the characteristic above are overloaded, meaning they are stressing some inherent limitations of the agent platform. These limitations can be partitioned into the following categories:

3.2.1.1 Physical Resources
Normally, the abstract design of an agent platform with multiple containers will be logically mapped to either one computer or a set of computers, one for each container. It is likely that at some point during the life of a container, agents within that container will begin to push the physical limits of their host machine in terms of memory usage, CPU usage, or network bandwidth.

Consider, for example, an e-service provider that has an agent platform running on a web server. As new customers arrive to query the service for information (in the form of agents), e-service agents are dispatched to communicate with customers and answer questions as well as
to collect sales information. For ease of organization, the company has set up agent containers to logically map to geographic regions. For example, upon arrival, each customer agent is moved to the container that represents the geographic area where the customer lives.

At any given time, a large number of customers from a particular region may use the e-service, causing a sudden increase in the number of agents living in the same container. Together, these agents will undoubtedly exhaust the resources of the host machine.

3.2.1.2 Programmatic Resources

Despite their ease of implementation, distributed agents are difficult to debug in a distributed environment. Agents that are developed and tested in a small, controlled agent platform may function incorrectly when placed in a much larger, complex agent community. They may begin to exhibit much higher response times compared to the controlled environment due to internal data structures getting much larger. In this sense, agents can exhaust the programmatic resources allotted to them at the time of development.

As an example, consider a case where a relatively slow dictionary data structure was chosen during development of an agent to maintain state during the agent’s lifetime. The agent functioned perfectly during testing on a small agent platform, since the dictionary never grew very large. However, when placed in a complex agent community, the agent exhibited poor performance due to excessive usage of the dictionary. While the physical resources allocated for the agent might remain relatively small, the agent is programmatically stressing the system due to decisions made in the development process.
A management system needs to be able to identify either of the situations above on a per-agent basis and react to them in a way that balances out the system. This might involve redistributing agents to other containers and/or creating new agents to accommodate some of the load.

This research will assume the following characteristics are sufficient to identify overloaded agents within a platform:

1. Communication level of the agent (type/number/size of messages sent/received)
2. Resources being used on the host machine (CPU time, memory usage, etc.)
3. Agent response time (how long agents take to respond to requests, if applicable)

Knowledge by the management system is then necessary to combine these factors and determine if a particular agent is overloaded and what action should be taken. This research will provide access to the necessary data so that a manager can make this decision effectively.

3.3 Agent Mobility

Agents have the ability to move between containers and even between entire platforms. Managers must be knowledgeable about the location of agents at all times and must be informed when they change locations. New agents that arrive from other locations should be recognized by the management system and added to the set of managed agents.

In the e-service example above, customer agents that are moved to their corresponding regional e-service container should be immediately within the management system’s domain, allowing the manager to keep track of customer agents’ resource usage on the system and their interactions with e-service agents.
3.4 Agent Communication

An effective agent management system must have the ability to analyze communication between agents on a platform. This might mean watching individual messages passed between agents on a platform or merely analyzing the number of messages that have been processed by a particular agent.

Consider again the e-service example described above. A manager of this system might want to know characteristics of conversations that take place between e-service agents and arriving customers. For instance, the manager might want to know how many messages are required on average to finish a transaction with a customer, or how many customers leave without making use of the service (due to high prices, unavailable merchandise, etc.)

Note that analyzing agent communication in this context is a more involved form of agent communication analysis than what was necessary for load balancing discussed above. The intent here is to capture not only the level of communication and the response time of agents, but also the patterns underlying the communication.

3.5 Unresolved Requirements

The following aspects of agent system management are yet to be resolved and will not be attempted in this research.

3.5.1 Security

Should a manager have complete access to every agent on a platform? (i.e the ability to change agent state, destroy agents, etc.) In an e-service scenario, customers would probably not want a manager of the e-service to be able to inspect all aspects of their agent, especially if their agent contains sensitive information like an address, phone number, or credit card numbers.
3.5.2 Advanced Mobility (Agent Scope)

When an agent leaves a platform, is it out of the management domain? Does the manager need to have some record that the agent left the system? Does the manager need to “follow” the agent to its new location?

4. Network Management Background

Network management is a key analogy used during this research. Since the fundamentals of network management are very similar to agent management, it is important to understand some background of managing networks and the types of systems available.

Network management has been a topic of research for many years. Researchers in both academic projects and in industry are continually trying to find the most flexible, scalable, and easy-to-use network management system to match growing industry requirements.

The following sections describe some aspects of network management that provide background for this research. They describe the basic components of a typical network management system, all of which will have corresponding roles in this research.

4.1 Proxy Agents

Proxy agents\(^1\) are commonly used in network management as monitors for resources. These “agents” are typically pieces of software distributed around the network to act as representatives for the resources that need to be managed. (Managed resources are devices such as printers, routers, hubs, or machines, but they can also be other software applications.)

\(^1\) Network management uses the term “agent” to describe a software process that monitors a particular resource and provides information to a network manager. To avoid confusion with agent-based systems that are the focus of this research, the term “proxy agent” will be used where possible to describe these agents.
Proxy agents hide the sometimes complex details of the device being monitored and expose a standard interface to it. This interface can then be used by management systems to query and change attributes of the device. Proxy agents are also capable of broadcasting events, called traps, to any listening management systems.

For instance, a printer is a common network resource that needs to be managed. A common interface for a printer that a proxy agent might expose would look something like the following (written as a pseudo C-style API):

```
[methods]
int getPaperCount();  // returns paper remaining in printer
int getTonerLevel();   // returns the amount of toner left
int getJobQueueSize(); // returns the number of jobs waiting
void purgeJob(int job); // purges a job from the queue
void reset();          // resets the printer to its default state

[traps]
OutOfPaperError       // thrown when the printer is out of paper
OutOfTonerError       // thrown when the printer is out of toner
```

Using this interface, a management system could manipulate attributes for almost any type of printer without knowing the details of the printer itself. It can also receive well-defined traps from the proxy agent when errors occur. Intuitively, proxy agent interfaces provide the same level of abstraction as any software API; the means of calling the functions in the API are somewhat different, however.

### 4.2 Protocols

Protocols provide the core communication between managers and proxy agents. These protocols are simply the language that the management devices use to communicate in a precise and meaningful way. SNMP (Simple Network Management Protocol) was one of the first of these to be developed and continues to be the standard protocol in network management in the
industry. The details of the protocol itself are not critical to understand. However, the following properties make it especially important in the world of network management:

1. *Easy to Use* – SNMP is, by definition, simple (relative to other network management protocols). It is fairly intuitive and does not provide a lot of sophisticated management support in its base implementation. By combining this with the flexibility mentioned below, this makes SNMP an intuitive and scalable protocol.

2. *Flexible* – SNMP can be easily modified to communicate with new proxy agent interfaces and can be used to manage almost any device.

3. *Widely-used* – Because SNMP was one of the first protocols developed for network management, it tends to be widely accepted and thus readily available in many different tools on the market.

### 4.3 Management Techniques

With proxy agents and a standard protocol in place, management systems are ready to begin analyzing the information available to them. Network management systems (sometimes called enterprise management systems) range anywhere from a single application to a complete software suite - from shareware to tens of thousands of dollars in software and consulting fees. Despite the wide range of choices, however, there are two basic ideas that pervade all of these systems.

#### 4.3.1 Centralized Management

Centralized network management has long been the standard technique for managing large, distributed network systems consisting of a variety of components. As discussed above, proxy agents are responsible for providing information via a standard protocol about the devices on
Managers typically poll proxy agents at some regular interval for data and then analyze and display the data based on user preferences. This scheme is central because the manager is the center of all management information for the network. The manager is responsible for all data collection, analysis, and reporting in the system. Sophisticated managers allow thresholds to be set based on the data, which can trigger alarms when values go out of range. (For instance, an alarm can be triggered when a resource is not responding.)

Figure 1. A standard centralized network management scheme. Managers poll proxy agents at regular intervals to retrieve information. This information is then 1) analyzed and displayed to the administrator or 2) used to trigger an alarm based on some administrator-defined threshold.
4.3.2 Decentralized Management

Recently, research has been conducted to improve on some of the issues that arise with centralized network management. Network bandwidth issues that arise with constant polling for values on proxy agents, as well as a lack of support for sophisticated data analysis, has led network administrators to look to decentralized management schemes, where the proxy agents themselves actually perform some level of analysis.

This decentralized method has inherent tradeoffs. On one hand, the proxy agents have the context necessary to analyze the data they are retrieving from the resource more effectively than a generic manager application. However, they lack the context of the entire system to make decisions on a global scale. (A management decision made by a proxy agent based on an isolated node in the network might let an underlying widespread problem continue.)

The result of these different approaches is that either or both work depending on the application. For instance, bandwidth-constrained networks that require some level of centralization may benefit from a “partially centralized” management scheme. [5]

5. Comparing Agent Systems to Networks

The above description of network management used the context of a system of hardware connections, but thinking of a “network” in the abstract sense as just a connection of nodes does not break the management paradigms described above. This abstraction of managing networks is the fundamental motivation behind this research.

Because agent-systems are distributed, they inherently have some similarities to traditional computer networks. If we consider agents to be nodes in a network, some of the problems common to network management become very similar as well. For instance, an agent being
overloaded with messages is analogous to a server being overloaded with requests by clients. Agents can get caught in infinite loops or even crash due to defective software, causing them to hang a piece of the system or leave other agents waiting for a response. This is analogous to a particular node in a network going down due to a power failure, server crash, etc. Other problems, such as the load of agents on the platform itself (which eventually translates to a machine), apply directly to network management problems.

Obviously, the analogy can only go so far. Concepts like agent mobility do not correlate well to computer networks, where the network configuration itself is not changing dynamically. However, enough basic similarities exist to argue that applying a network management solution to this problem is certainly reasonable.

Since SNMP is a standard protocol for network management, sophisticated enterprise management products like Hewlett Packard’s Openview software, provide off-the-shelf, customizable, SNMP-compliant solutions. At a minimum, these products provide the ability to browse through the interface of a proxy agent and manipulate the values associated with it, as well as visualize temporal views of the data. More complex systems provide the ability to set thresholds on values which trigger events when crossed. These events can be received by customizable plugins that provide custom behavior.

While many of these SNMP products are targeted for computer networks, they must remain flexible, given the nature of SNMP and the variety of network components that exist. Most products advertise the ability for customers to write custom proxy agents that manage their own resources in the way they choose. They also provide a means to customize the
management system itself, giving customers complete control over how the system looks and reacts to events.

This flexibility makes SNMP and products like OpenView an ideal fit for agent management. Most of the underlying ideas of managing resources are the same for network management as they are for computer networks, and these similarities can be exploited for managing agent systems.

6. Research

Given the above motivating factors, the goal of this research is to provide a formal method of managing agent platforms using a network management paradigm. This solution can then be used in currently available industry-standard enterprise management products to test its viability.

With the similarities between networks and agent platforms described above, a centralized network management scheme can be applied to manage the agent platform. In this case, the managed resource will be the agent platform itself, and proxy agent(s) will provide a standardized view of the agent platform to management systems. SNMP will be used as the management protocol in the system.

While decentralized management has some advantages, centralization is still the industry standard and thus many sophisticated management systems are available that support it.

The following steps will be taken to perform this research:

- Definition of a standard management interface that meets the basic requirements listed earlier for managing an agent platform.
• Implementation of a proxy agent that exposes this interface and can communicate via SNMP.

• Integration of the proxy agent with a number of management systems, including customization where necessary to take advantage of application-specific management capabilities.

• Testing the system with generalized, pattern-based agent scenarios.

The result of this research will be an agent-platform independent, extensible agent management system that can be used with currently existing, widely-used management software.

7. Approach

Each step to be taken in the research is described below in brief detail.

7.1 Management Interface Definition

An interface must be defined which will control access to information in the agent platform through SNMP. While all possible information about the platform cannot be provided, the interface will serve as a baseline for common management information. Specific implementations can add more platform or agent specific information as needed.

Initially, this interface might contain some of the following types of data:

• Platform configuration information (lists of containers, agents, etc.)

• Agent message information

• Agent manipulation access, such as creating, destroying, moving agents, etc.

• Platform manipulation access, such as adding, removing containers, etc.
7.2 Proxy Agent Implementation

A proxy agent will be implemented that will interact with the agent platform to retrieve information requested through the interface above. The proxy agent will communicate with managers via the SNMP protocol.

Since this research is not meant to be an exercise in implementing SNMP from scratch, the AgentX architecture [7] will be used to provide SNMP support. AgentX acts as a server for SNMP requests from managers. Sub-agents can register with a master agent (or the AgentX server) and communicate through a standard API, thus relieving the developer from knowing the details of SNMP. In some sense, the protocol itself is not that important to this research. The fact that it is widely used and flexible is a much more important issue.

The SNMP agent that will be developed for this system will actually be a sub-agent of the AgentX architecture.

7.3 Management System Integration

The most appealing aspect of SNMP is its compatibility with industry-standard management packages. Integrating an agent platform management scheme with some of these packages will be compelling evidence that the research is a success. The following types of management software will be integrated with this system:

7.3.1 MIB Browser

A MIB browser is a simple piece of software that can be used to exercise the fundamental operations of an SNMP interface, called a MIB (Management Information Base). A MIB browser will be used to get/set variables exposed through the MIB and poll for various values. Traps can also be caught in most MIB browser implementations. Integrating the agent management
system with a MIB browser will be a straightforward method of showing basic functionality and testing.

7.3.2 Enterprise Management Systems

While MIB browsers are simple, easy-to-use diagnostic tools, they are too small for network management tasks. Enterprise management systems provide a standard solution for more sophisticated management needs. These systems normally provide the following:

- Polling support to gather temporal data from proxy agents.
- User-defined thresholds that can trigger events based on trends in the data.
- Hierarchical maps of the management domain as well as connections between components, allowing managers to visualize devices relative to their physical or logical locations.
- Full customization of event handling and visual display through an API.

These capabilities are just as useful in agent platform management as they are in network management. By using the visual display capabilities, agent platforms can be visualized hierarchically, showing the containers within a platform as well as the agents in each container. Communicating agents can be identified by a connection between the agent symbols on the map.

7.4 Managing Patterns of Agent Interaction

To fully exercise the system, some generalized patterns of agent communication and interaction will be developed and executed on an agent platform. Management systems can then be used to recognize problems exposed by these patterns. This allows the entire system to be used in a more realistic environment. The patterns described below are designed to expose common
agent interactions that pose challenges to a management system (and thus an agent-platform administrator). They closely follow the requirements of managing an agent system that were described earlier.

Each pattern is characterized by an action and a desired response from the management system.

7.4.1 Platform Load Pattern

**Action:** Agents are spawned in a container until an overload condition is reached.

**Reaction:** The management system must react by moving agents to different containers until stability is achieved.

7.4.2 Client/Server Pattern

**Action:** One agent acts as a server for requests made from other agents. The server agent becomes gradually overloaded with requests from clients.

**Reaction:** As the server communicates with more and more agents simultaneously, the management system must recognize and react to overloading a particular agent with messages. Another server agent will be created to service some of the client agents and balance out the system.

7.4.3 Agent Mobility Pattern

**Action:** Agents move continuously between containers.

**Response:** The management system must observe the movement, update visual displays accordingly, and respond quickly to the dynamic nature of the pattern.

8. Assumptions and Limitations

The following assumptions and limitations will affect the approach and scope of the research.
8.1 FIPA Platform Specification

The research will focus on the FIPA agent platform specification as a baseline target. FIPA has a solid, widely implemented specification for agent platforms that is the standard throughout the industry.

8.2 Access to Managed Data

When proxy agents are implemented, the resource that the proxy is monitoring must provide access to the data that is exposed through the interface. Otherwise, the proxy agent would not be able to provide any useful information to the manager. For instance, in order for a printer agent to implement the interface described earlier, the printer itself must provide a device driver (or some other low-level mechanism) that allows access to such information.

The same is true for agent platform implementations. While a standard interface can be developed that provides useful information for management software, the ability to retrieve this information is dependent upon the agent implementation itself. The FIPA specification does not provide any standard means for obtaining information about an agent platform, beyond a very high level description.

Despite the lack of a standard, most agent platforms provide some level of internal access to platform-specific information. For instance, the Jade implementation has a number of ways to access management information at all levels in the system, from platform events to specialized ontologies that can be used specifically for “sniffing” messages that are sent to/from agents.

This research assumes that it is up to the agent platform implementation that wishes to be managed to expose the necessary information in some manner.
9. Technical Architecture

The following diagram shows the architecture for the system. Note that this is preliminary and is subject to change as the research continues.

Figure 2. Architecture of the agent management system. The system can be logically partitioned into three distinct layers.
The management system can be partitioned into the following three layers, each of which refers to the above diagram.

### 9.1 Agent Platform Layer

Beginning with the lowest layer in the diagram above, this layer provides the core management functionality of the system. The main component is the proxy agent, which provides the central communication mechanism between the agent platform and any listening managers. The proxy agent will be an actual FIPA-compliant agent living in the platform, allowing it to have full access to platform events. Agents in the Jade platform can register themselves as “tools” and are thus given access to more diagnostic information than other agents. Subscribing as a tool agent as well as using the introspection ontology allows most important system events to be received.

The proxy agent will implement the interface described in the MIB for the system. This is typically accomplished by compiling the MIB file into a set of classes with default method implementations. Developers then derive from the generated classes and override the methods to provide correct implementations. The classes must make the necessary communication to return information about the agent platform when requests come in from managers. MIB compilers are a standard component of proxy agent toolkits, as they relieve the burden of manually maintaining the MIB hierarchy.

The proxy agent is responsible for communicating with both the AgentX master agent as well as the agent platform itself (mainly the AMS). Since it is registered as a tool, it will receive events from the AMS when various actions occur within the platform. The proxy agent must maintain some amount of state to be able to return information to managers asynchronously from these
events. This agent will also send traps to the master agent when agents are created, destroyed, etc.

Note that this part of the implementation is extremely platform-specific. The means by which a proxy agent obtains information from its resource depend almost completely on how the resource makes that information available. A key element to this design is the layering structure that allows completely different internal implementations, provided that they expose the interface defined by the MIB.

9.2 AgentX Master Agent

This layer of the architecture is very thin and requires little effort. It is meant to hide the SNMP details from the proxy agent and is essentially an off-the-shelf solution. The AgentX architecture is well-specified and used throughout the industry as an SNMP technique for writing custom SNMP-based proxy agents.

9.3 Management Layer

The top layer in the system is the management system. In the simple case, the management system will be a standard MIB browser that can be used to manipulate variables exposed through the proxy agent.

As discussed earlier, an enterprise management system provides more sophisticated management support as well as customization capabilities. The management system connects to the AgentX master agent and polls for values over time, checking user-defined thresholds. A plugin component will be written for the management system that allows more effective visualization of the agent platform when these thresholds occur.
10. Related Research

Research directly related to agent platform management is fairly scarce. Aside from implementation-specific methods of inspecting an agent platform, general-purpose management paradigms are relatively unexplored. This research will provide one of the first general solutions for managing an agent platform.

Since this research is closely tied to network management, it is appropriate to mention some of the relevant network management research issues that apply to managing agent platforms.

10.1 Moving Toward Decentralized Management

As mentioned earlier, decentralized management has obvious advantages over a purely centralized scheme. Research has shown that centralized methods, while extremely successful for smaller networks, are difficult to scale to large, distributed domains. More managed components lead to more polling, which increases network bandwidth. Under failure conditions, increased polling can easily create artificial bottlenecks caused in part by the management system itself [14].

Decentralization pushes more intelligence into proxy agents themselves, thus relieving the manager of the burden of constantly polling agents for information. The introduction of such techniques as Management by Delegation (MbD) gave rise to extremely flexible decentralized management methodologies [16].

Unfortunately, this trend toward decentralizing management lacks industrial support. All but a few of real-world management systems remain completely centralized, normally based on SNMP. Since this research intends to show the immediate application of agent platform management using currently existing tools, it will focus on centralized management. However,
it is reasonable to expect that as more decentralized systems become available, this research may need to adapt to a more decentralized scheme.

10.2 Using Mobile Agents as Proxy Agents

In conjunction with decentralization, many researchers are investigating the use of intelligent mobile agents to monitor resources [15]. These agents could not only collect and analyze data from the resource at a local level (instead of propagating all information back to central manager), but also make management decisions on behalf of a central manager, given some set of rules.

11. Schedule for Completion

The following schedule enumerates the tasks that must be completed and the approximate dates that correlate with them. Note that this schedule may change as the research continues.

<table>
<thead>
<tr>
<th>Task</th>
<th>Completion Date</th>
<th>Percent Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposal document</td>
<td>10/1/2001</td>
<td>95%</td>
</tr>
<tr>
<td>Proposal defense</td>
<td>10/29/2001</td>
<td></td>
</tr>
<tr>
<td>Interface definition/MIB</td>
<td>11/10/2001</td>
<td>40%</td>
</tr>
<tr>
<td>Proxy agent implementation</td>
<td>11/20/2001</td>
<td>40%</td>
</tr>
<tr>
<td>AgentX integration</td>
<td>12/5/2001</td>
<td>30%</td>
</tr>
<tr>
<td>MIB browser integration</td>
<td>12/15/2001</td>
<td></td>
</tr>
<tr>
<td>HP Openview plugin</td>
<td>1/20/2002</td>
<td></td>
</tr>
<tr>
<td>Pattern implementation</td>
<td>2/1/2002</td>
<td></td>
</tr>
<tr>
<td>Thesis defense</td>
<td>4/2002</td>
<td></td>
</tr>
</tbody>
</table>
12. Bibliography


6. Perkins, D., *Inter-Table Indexing in SNMP MIBs*, June 1998, SNMPInfo


