

# Managing Agent Platforms with AgentSNMP

**Brian D. Remick**

University of Utah  
50 S. Central Campus Drive  
Salt Lake City, UT 84104  
(801) 588-1776

**remick@cs.utah.edu**

**Robert R. Kessler**

University of Utah  
50. S. Central Campus Drive  
Salt Lake City, UT 84104  
(801) 581-4653

**kessler@cs.utah.edu**

## ABSTRACT

Management of agent platforms is an inherently difficult problem given the flexibility and ambiguity associated with autonomous mobile agents. In addition, no standard mechanism is defined for managing agent platforms, which has led to ad hoc solutions throughout the industry.

The nature of agent platforms draws an analogy to distributed networks, an area in which management has been well researched. This research applies a network management solution using the Simple Network Management Protocol (SNMP) to managing agent platforms.

Utilizing the flexibility of industry-standard SNMP techniques, a formal interface is defined for management of a FIPA-compliant agent platform. A specific proxy agent implementation of this interface for the JADE agent platform implementation is also developed. The resulting management system, AgentSNMP, is integrated with an off-the-shelf enterprise management system, HP Openview, as a proof of concept. The result is an efficient, flexible means of managing agent platforms.

## Categories and Subject Descriptors

I.2.11[**Distributed Artificial Intelligence**]: multiagent systems

## General Terms

Management, Standardization

## Keywords

Agent platforms, management, SNMP, network management

## 1. INTRODUCTION

As agent platforms become an increasingly popular paradigm in distributed environment implementations, the need to formally manage these platforms is also becoming equally important.

Agent platforms are extremely complex runtime environments. Details of agent state, 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.

We present a solution to this problem with AgentSNMP, an agent management system utilizing the Simple Network Management Protocol (SNMP) [1] and proven network management techniques. The flexibility of SNMP, coupled with the availability of powerful, off-the-shelf network management software that supports it, provides a highly-effective means of managing agent platforms [2].

## 2. MOTIVATION

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 reuse the same management software.

In addition, FIPA (Foundation for Intelligent Physical Agents) [3] does not define any formal external management scheme for an agent platform, leaving the implementation to decide what type of management paradigm to support. Unfortunately, this has led to a wide variety of management schemes that are specific to the implementation used.

Most of these custom management tools are low-level and most helpful in agent development, but they tend to be less useful when agent populations become very large or communication levels are very high. For instance, most management tools provide access to individual messages sent on the platform, but they lack support for higher-level communication analysis. Other management requirements such as load balancing are generally left out altogether.

## 3. NETWORK MANAGEMENT

Fundamentally, managing a network of devices has strong similarities to managing a network of agents running on an agent platform.

### 3.1 Network Management Properties

In order to be effective, a network management system must have the following three properties. First, it must support a wide variety of resources in a standard way. In other words, one piece of management software must be able to effectively manage all resources on a network. Second, it must be scalable and flexible. The management software should handle modifications to the network topology gracefully, even if the changes are significant. Finally, the management software must be as non-intrusive as possible. Gathering information about network traffic should not significantly add to the network traffic itself.

### 3.2 Network Management Components

Generally, a network management system contains the following components:

- **Resource.** The network device to be managed. This is normally a hardware device, but software is normally supported as well.
- **Resource Interface.** The resource exposes a standard interface that management software uses to manipulate properties of the resource. This interface is coupled with the protocol used.
- **Proxy Implementation.** An implementation of the resource interface, called a *proxy*, is specific to the managed resource and hides the complexity of the resource from the rest of the system.
- **Protocol.** A standard protocol is used to facilitate communication between management software and the resource interface. This protocol provides a standard means of communication regardless of the type of resource.
- **Management Software.** Powerful management software is used to communicate with resources via a standard protocol, gather temporal data for resource attributes, provide threshold checking, and alert managers when errors occur. Most management systems also provide a high degree of customization to allow devices to be visualized in different ways.

### 3.3 SNMP

SNMP was one of the first network management protocols to be developed and continues to be the standard in industry today. Although very simple, nearly all enterprise management software natively supports SNMP, allowing any SNMP-compliant resource to be managed without any customization.

SNMP uses a structure called a MIB (Management Information Base) to define the interface for a resource. A MIB consists of a set of object definitions, each of which exposes some property of the resource to be managed. These definitions must be extremely flexible, since SNMP is designed to support any type of resource.

## 4. NETWORK AND AGENT MANAGEMENT

From the descriptions above, it is straightforward to see that both the agent management and network management domains need to solve similar problems. They both need to manage distributed resources of varying types, overload conditions, resource crashes/errors, etc.

Obviously, some differences exist. Agent platforms are generally much more dynamic than networks. Agents are created and destroyed at very high rates, whereas nodes on a network tend to remain static. The idea of dynamic device (agent) mobility is an idea completely foreign to network management.

Despite these differences, however, it is reasonable to draw an analogy between these two domains. AgentSNMP provides a solution to managing agent platforms using a network management paradigm, showing that the principles and techniques of managing a distributed network are effective in managing an agent platform. Figure 1 illustrates this idea.

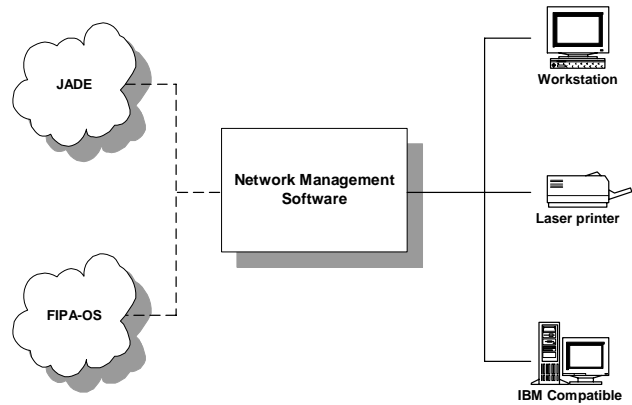


Figure 1. A network management paradigm can be applied to managing agent platforms.

## 5. AGENTSMP

The AgentSNMP system uses the concepts and techniques described above in network management to manage FIPA-compliant agent platforms.<sup>1</sup>

The agent platform itself is the “managed resource” in AgentSNMP. By exposing a standard interface to the platform, it can be managed using off-the-shelf network management software just like any other network device.

In addition, the customization capabilities of most enterprise management packages allows the agents on the platform to be visualized just like nodes on a network.

### 5.1 Architecture

The architecture of AgentSNMP closely follows that of any other managed resource in network management.

1. A standard interface (MIB) provides access to attributes of the resource.
2. A proxy specific to the resource implements the interface to provide the necessary information.
3. An enterprise management plugin is provided to take advantage of the customization available and visualize the resource appropriately.

Figure 2 shows the basic architecture of AgentSNMP.

#### 5.1.1 FIPA MIB

The FIPA MIB [4] is the SNMP interface that exposes properties of agent platforms that are essential to effective management. The MIB allows management software access to low-level platform information, such as attributes of individual agents living on the platform and messages sent between agents. The following is a representative list of the contents of the MIB:

- **Hosts.** A list of the host machines that comprise the agent platform. Hosts can be added and removed from the platform remotely.
- **Agents.** A list of agents running on the platform. Attributes are exposed for each agent, including the agent’s host machine, start time, response time, state, etc. Various levels of analysis can be enabled on a per-agent basis, which

<sup>1</sup> The FIPA agent platform specification, now a de facto standard, was chosen as a baseline for management.

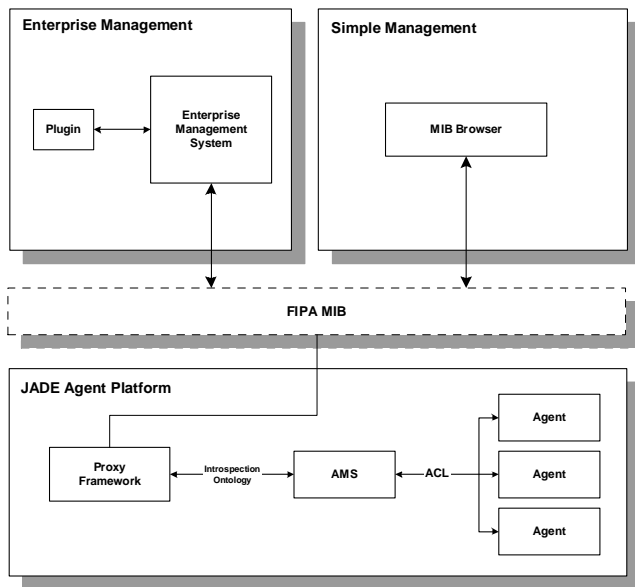


Figure 2. AgentSNMP architecture.

avoids unnecessary processing of agents that are not being managed.

- **Messages.** A list of the most recent messages sent on the platform. Each entry in the list includes information about the sender/recipient agents, the type of performative used, message content size, and even the message content. Messages are tracked both between agents on different platforms and between agents on the same platform.
- **Channels.** To provide access to higher level messaging information, a list of the channels active between agents is provided. A *channel* is an abstraction of messages sent between two agents. Channels provide information about the rate of communication between agents, the types of messages sent between agents, etc.
- **Notifications.** A list of notifications is provided that management software can receive asynchronously from an agent platform. These events include agent create/destroy notifications, message send/receive notifications, etc.

The FIPA MIB closely follows SNMP standards so that network management software can interpret the interface directly with no additional customization.

### 5.1.2 Proxy Implementation

Given the FIPA MIB definition described above, software must be written that provides an implementation for this interface and gathers the required information on the agent platform itself. In the case of the JADE [5] agent platform used in AgentSNMP, the proxy is actually a FIPA-compliant agent living on the platform. This design is necessary in JADE to gain access to platform-level events.

The proxy is responsible for translating generic SNMP requests from outside the platform to JADE-specific platform requests. It must also keep track of the data structures defined in the FIPA MIB so that it can pass information back quickly to management software.

The JADE platform provides the Introspection ontology that facilitates communication between agents living on the platform

and the AMS. The Introspection ontology gives access to agent create/destroy, message send/receive, and other platform-level events. By using the Introspection ontology, the proxy can update the MIB data structures as needed and provide information through SNMP.

### 5.1.3 Enterprise Management Plugin

Because the FIPA MIB is defined according to standard SNMP structures, no additional work is required to manipulate the interface to the agent platform via SNMP-compliant management software. Thus, all of the polling, graphing, and threshold-checking capabilities common in network management software are available with no customization. For example, see Figure 4 for an example of a graph produced directly from the FIPA MIB.

However, one added benefit of using enterprise management software is the visualization of managed devices that is available. Due to the variety of resources that the software is capable of managing, a high degree of flexibility and customization is often provided to tailor the visualization of specific devices to the needs of the user. This functionality is commonly implemented as a plugin interface, where users can write customized software with a vendor-provided SDK that is loaded at runtime by the management software.

AgentSNMP takes this approach with HP Openview [6] as a proof of concept. A plugin is provided with AgentSNMP that interfaces with Openview's GUI and provides customized management capability of an agent platform directly through Openview.

The plugin essentially performs the following tasks:

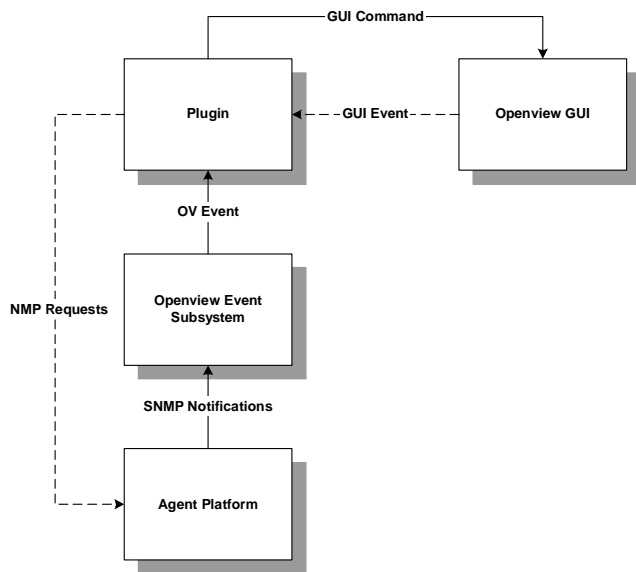
- Translates SNMP notifications from the agent platform into Openview topological changes. For instance, when an agent is created on the platform, a new symbol is created in Openview to represent the agent.
- Translates Openview GUI events to SNMP requests for the agent platform. For instance, if a user drags an agent symbol in Openview from one host machine to another, the plugin detects this action and makes a request through the FIPA MIB to migrate the agent from one host to another.

These translation functions are illustrated in Figure 3. As an example, when an *agentCreate* notification is generated by the agent platform, the event passes through the Openview event subsystem and arrives at the plugin as an Openview event. The plugin responds by requesting the Openview GUI to create a symbol to represent the agent. The opposite sequence occurs when a user initiates a GUI event, such as dragging an agent from one host machine to another. The plugin intercepts this event and sends a request to the agent platform via the FIPA MIB to migrate the agent.

Like most network management packages, Openview shows the nodes of the network visually, drawing lines between symbols to represent physical connections on the network. The plugin uses this functionality by logically connecting agent symbols when messages are sent between agents on the platform. This allows platform administrators and developers to easily analyze agent communication patterns on the platform.

## 6. USING AGENTSMP

In order to illustrate the use of AgentSNMP under realistic conditions, a number of patterns of agent behavior were



**Figure 3. The AgentSNMP plugin for HP Openview translates between SNMP and the Openview GUI.**

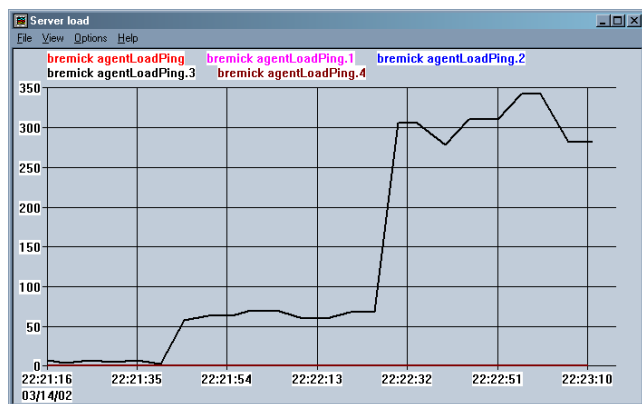
developed. These patterns simulate complex agent environments that would be difficult to manage with existing platform management tools.

### 6.1 Client/Server Agent Pattern

Among these patterns is a client/server simulation, where many client agents make requests to a server agent. Management software can be used to track the response time of the server as well as recognize the client/server communication pattern on the platform.

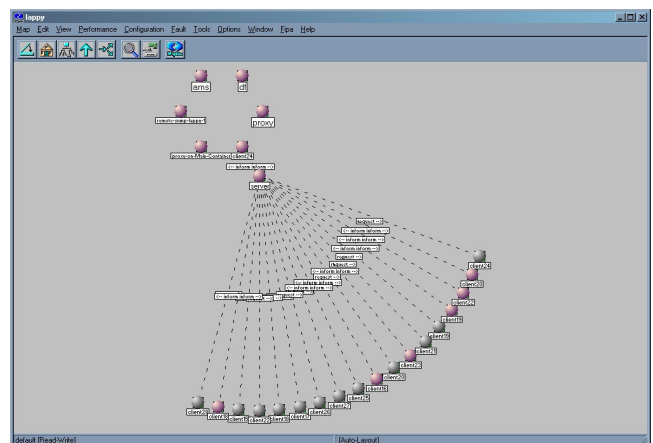
Figure 4 shows the server response time increasing as the number of client agents is increased. The graph shown was generated using Openview’s real-time graphing capabilities. Note that this functionality requires no additional customization (no plugin is required). Openview is polling values directly from the FIPA MIB.

The simulation begins with no client agents, accounting for the very fast server response time. At 22:21:40, 15 client agents are created, resulting in a small increase in the response time. At 22:22:25, 15 more client agents are created, resulting in a large increase in response time.



**Figure 4. Server response time (in ms) as the client agent count increases.**

Figure 5 illustrates Openview’s ability to visualize this pattern, in this case with the help of the AgentSNMP plugin. Agent symbols shown in gray are actually running on different host machines – they are shown on the same host as the server so that communication between them can be seen. Dashed lines between agent symbols represent messages between the clients and the server, where the performative of each message is shown on the connection label. Note that Openview has automatically oriented the agent symbols so that the topology is clear, which is extremely useful in recognizing the patterns underlying the communication between agents.



**Figure 5. Openview visualization of client/server pattern.**

## 7. SUMMARY

Management of agent platforms is an extremely difficult problem. It combines the requirements of managing a distributed network environment with the additional complexity of agent mobility.

AgentSNMP shows that network management techniques can be used to manage agent platforms. The flexibility of SNMP, as well as the availability of powerful, off-the-shelf management software that supports it natively, makes network management a viable solution to managing agent platforms.

## 8. REFERENCES

- [1] J. Case, *et al.*, “A Simple Network Management Protocol (SNMP),” Request for Comments 1157, May 1990
- [2] B. Remick, “Managing Agent Platforms with the Simple Network Management Protocol,” May 2002, <http://www.cs.utah.edu/~remick/research/research.html>
- [3] Foundation for Intelligent Physical Agents, “FIPA Agent Management Specification,” Geneva, Switzerland, 2000
- [4] B. Remick, *FIPA Management Information Base*, available online, <http://www.cs.utah.edu/~remick/research/fipa.mib>
- [5] F. Bellifemine, A. Poggi, G. Rimassa, “JADE – A FIPA-compliant Agent Framework,” in *Proceedings of PAAM '99*, London, England, April 1999
- [6] Hewlett-Packard Openview Network Node Manager, <http://www.openview.hp.com>