Integration of Mobile agents with SNMP: Why and How

B. Pagurek, Y. Wang, and T. White
Department of Systems and Computer Engineering
Carleton University, Ottawa Ontario Canada
{bernie, ywang, tony}@sce.carleton.ca

Abstract

Mobile agents have been proposed as a solution to the problem of the management of increasingly heterogeneous networks. However, the proposed solutions often ignore the value of legacy solutions and protocols. This paper extends an existing mobile agent framework targeted at network management with an architecture and design for integration with an SNMP agent. Brief descriptions of the use of this solution for the provisioning of ATM permanent virtual circuits and providing a Mobile Agent MIB are also provided.

Keywords

SNMP, DPI, mobile agents

1. Introduction

In recent years, a number of papers involving mobile agents in network management have been written. These have considered, for example, tasks of network health monitoring [1], [2], fault diagnosis [3], and network configuration [4]. There are two parts to the "why" question of the title: "Why mobile agents?" and "Why integrate?" There are also two aspects to the "how" part of the title involving whether using pure SNMP is sufficient to accomplish the integration, or whether some other protocol is appropriate.

Although it is not the main focus of this paper, let us consider first, "Why mobile agents?" The main argument used for mobile agents is to get management intelligence out into the network where needed and preferably on the individual devices. That is, what is needed, where it is needed, when it is needed. Efficiency arguments concerning mobile agents depend very much on the problem to be solved, and whether or not you think computing power is increasing faster than bandwidth. There are very few examples in the literature that really present a commanding argument based on real data for real problems, although a model has been proposed [5]. No one has really come up with the killer mobile agent application yet, although it is clear that in large networks it is not possible to monitor many sub-networks over a WAN without causing a bottleneck at a centralized manager. This problem is typically handled by employing distributed mid-level managers or data collectors which usually are not mobile. SNMPv2 attempted to introduce the manager-to-manager MIB for such purposes and failed (but mainly for other reasons).

Recently also, the concept of Management by Delegation was expounded [6] and the IETF Distributed Management (DISMAN) Working group has been working since 1996 to formalize and standardize some of these ideas. Work initiated by Levy

and Case in 1993 on a script language and script management has evolved into the DISMAN proposals including a standard MIB for delegating management scripts [7] and a MIB for scheduling their execution [8]. Clearly, there is strong opinion that such techniques are useful. In fact, it is possible to delegate Java programs as scripts. There is however a difference between mobile code and mobile agents but the dividing line is a constant source of argument. Mobile agents tend to have some degree of intelligence and usually have a level of autonomy making them capable of adaptive migration from node to node and capable of communication with other agents. The script MIB appears to support mobile code with more limited capabilities. In any event mobile agents in network management seem to be here to stay, and are being actively investigated[1][2][20][21]. Delegation, among many other tasks[11], is easily and flexibly accomplished by employing a mobile agent platform based on Java, and Java Virtual Machines are now beginning to appear on routers etc. Such systems as described in [9] and [10] are typically built much like the one shown in Figure 1 below and suggest replacing a script with a mobile agent.

Now let us turn to the other part of the question, "Why integrate?" First of all SNMP is likely to be the network management protocol of choice for the foreseeable future and many network nodes will be equipped with an SNMP agent. Some of these agents may be older legacy agents and some may be enhanced with more modern additional extensibility instrumentation like the Distributed Protocol Interface [12] (DPI) or its successor the Agent Extensibility Protocol [13], [14] (AgentX protocol). These agents will already be gathering substantial node information and even if mobile agents were being employed for new tasks, it would be reasonable to take advantage of the existing capabilities of the resident SNMP agent. With this in mind, let us consider the kinds of interaction that might be likely:

(a) A mobile agent may arrive at a node and wish to access data, such as interface statistics, from the resident SNMP agent's MIB. That is, it wants to Get or perhaps even Set data in the SNMP MIB. One could equip the mobile agent with SNMP managerial capabilities and allow the mobile agent, acting as a manager, to issue SNMP request packets to the resident SNMP agent. Of course the mobile agent would not have to be co-resident to be able to do this, but for security reasons alone, it is not a good idea to allow a mobile agent to open a socket to communicate with the SNMP agent. A better way would be to provide a local SNMP service or a carefully controlled secure interface to the nodes resources for this purpose. There would still need to be an application program interface (API) or a mechanism that would give the mobile agent a way to make its wishes known to the intermediary service. Such an approach has been adopted in [1] and in [23] using readily available Java SNMP classes.

    The advantage of such an approach is that one does not have to modify the SNMP agent to provide access to its MIB. The cost of this is that the mobile agent would need to have full SNMP managerial capabilities and would have to handle such tasks as BER encoding. Another approach is to design a lightweight protocol that would take advantage of the co-residency of the mobile and SNMP agents. This approach is new and will be presented in section 4 of this paper where the RDPI protocol [15] is described.

(b) A mobile agent may arrive at a node and have the capability of extending the existing SNMP MIB with certain meta-variables that it brings with it. That is the mobile agent may bring with it some state, some information from the outside network at large, that it was charged with to explore and analyze. A manager would have to know of the existence of such additional information to be able to take advantage of it, but presumably it delegated the mobile agent to acquire such information in the first place and would have defined it as a MIB sub-tree.

The SNMP DPI protocol [12] and its successor the AgentX protocol [13] were designed specifically to allow a sub-agent process to extend or enhance an existing SNMP agent by first registering with it, and then communicating with it using the specified protocol. Clearly the SNMP agent would have to have the protocol built in, but many agents today are DPI enabled and it appears that many future agents will have the improved AgentX protocol incorporated. Clearly a mobile agent could behave as an AgentX sub-agent, or better, would employ an API of a local service that would securely accomplish the registration and pass queries back and forth. In either case, the manager would need to have predefined the new MIB variable to be dynamically added to the local SNMP MIB. This approach was proposed earlier by the authors [9] and is developed further here. It has also recently been adopted by others [24].

(c) A mobile agent, having acquired information during its migratory activity, may arrive at a node that consequently implies a fault or degraded performance either at that node or elsewhere in the network, and wish to send off a defined SNMP trap to a remote SNMP manager. Both the AgentX and the DPI protocol are designed to enable the SNMP agent to receive such a trap from a sub-agent/mobile agent and transmit it to an SNMP manager.

(d) Finally, it is conceivable (but less likely) that some entity would like to send an SNMP trap to a mobile agent which is acting in the role of an SNMP manager. This assumes that the sender can actually locate the mobile "manager" using the mobile agent environment's agent location service. This feature could be part of the RDPI protocol mentioned in (a) above.

At this point, we should define the acronym RDPI. It stands for the Reverse Distributed Protocol Interface. If efficiency is an issue, and generally it is, the RDPI protocol has certain advantages over using SNMP directly. The RDPI protocol is a lightweight protocol that allows a process, usually running on the same node as the agent, to issue SNMP managerial commands to that agent without having to do all the work of BER encoding etc. In this respect, RDPI is related to SNMP the way DPI was related to the now obsolescent SMUX protocol. The RDPI messages are a subset of the DPI message set and they were designed to share the same structure as the corresponding DPI message. The main difference is that where the DPI protocol uses a group ID and an instance ID, the RDPI protocol uses just an object ID. RDPI and DPI fit conceptually into the functional architecture as shown below in Figure 2. The actual implementation employs an intermediary called the VMC between the mobile agent and the SNMP agent for security and abstraction reasons.

Following this introduction, section 2 briefly describes the mobile agent environment used and the Virtual Managed Component (VMC) which serves as the controlled interface to all of the nodes' resources. Section 3 then briefly describes the DPI protocol whose sub-agent side we implemented in Java for inclusion in mobile

agents. Section 4 describes the RDPI protocol in more detail while section 5 discusses the VMC architecture and the APIs used by mobile agents. Section 6 briefly describes a PVC configuration application that uses RDPI and this is followed by the conclusions and references.

2. The Mobile Agent Environment

In order to perform network management functionality using mobile agents, it is necessary to have an infrastructure that provides a framework for code mobility. An infrastructure that supports this functionality has been developed as part of the Perpetuum Mobile Procura Project in the Department of Systems and Computer Engineering at Carleton University [9]. The Java programming language was chosen for development of the infrastructure in order to take advantage of its inherent platform independence and portability, support for code mobility, security, and robustness.

As shown in Figure 1, every Network Component (NC) contains a Mobile Code Daemon (MCD) running within a Java Virtual Machine (JVM). The MCD provides a number of services that facilitate the execution of mobile agents. Included are: a Mobile Code Manager (MCM) that manages the life cycle of a mobile agent from its arrival and authentication at the network component to its migration or perhaps destruction, a Migration Facilitator to transport mobile agents between NCs, a Communication Facilitator for collaboration between local and remote mobile agents, and an interface called the Virtual Managed Component which provides for mobile agents accessing the NC's managed objects and resources in a controlled and secure way. The VMC is responsible for management of the mobile agents access rights and the allocation of resources to that agent.

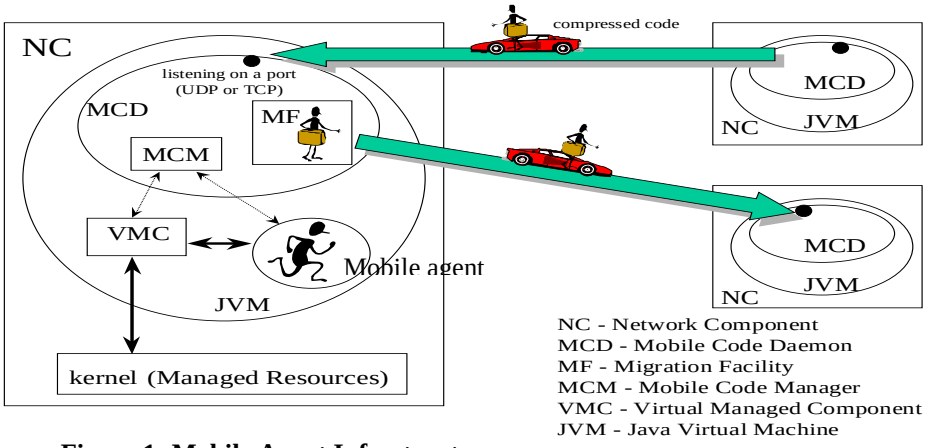

Figure 1: Mobile Agent Infrastructure

Once a mobile agent has been authenticated and accepted, it is instantiated as a thread within the same JVM as the MCD. The MCD maintains handles to all the instantiated mobile agents within via the MCM. Note that there can be more than one instance of VMC and also that the mobile agents can obtain handles to the VMCs from the MCM.

The Virtual Managed Component shown in Figure 1 is a special subclass of the mobile agent class that provides kernel access services to the mobile agents. After analysis, a decision was made that mobile agents should not have direct access to sockets or have the ability to make direct connections with the SNMP agent. This provides security to the mobile environment while also allowing lightweight SNMP ignorant mobile agents. A decision to move complexity out of the mobile agents and into the VMCs is based on an analysis of their lifecycles. Typically, mobile agents will be migrating off and on throughout their lives while VMCs will be stationary on their network component.

The VMCs will communicate with the SNMP agent on the mobile agents' behalf using protocols such as DPI/RDPI. The DPI protocol provides dynamic links between a number of SNMP sub-agents and the SNMP agent. The sub-agents can then attach their extended MIBs to the standard MIB supported by the SNMP agent. RDPI handles local mobile agent requests to the SNMP agent and is a lightweight variation of SNMP itself. With DPI and RDPI, mobile agents can perform a full set of management functions as well as provide new information to the central network management station, such as the number of mobile agents on the component and the services they are providing.

VMCs providing these MIB extension and MIB access services could be communicating with the SNMP agent using DPI/RDPI, AgentX or some other local protocol. Conceivably, multiple protocols could be active on the same component at the same time. It has been decided that a set of interfaces should be designed and developed to provide an abstraction of the communication between mobile agent and VMC. This would provide a constant, unchanging interface to the mobile agents regardless of the underlying VMC-SNMP communication implementation.

A dual VMC implementation was proposed after analyzing the alternatives described in section 5. One VMC would provide MIB access, using RDPI in this implementation. Another VMC would provide MIB extension capability to the mobile agents. Interfaces are proposed for each VMC that encapsulate this functionality. This solution provides the required capability while keeping the complexity to a minimum. It should be noted that there is no reason that the two VMCs could not be combined. The decision to design their interfaces separately reflects a preference for flexibility, in recognition of the independence of the two processes (MIB extension versus MIB access). The next few sections provide some more detail about DPI, RDPI and the VMC design.

3. The SNMP DPI Protocol

The Simple Network Management Protocol Distributed Protocol Interface [12] (DPI) is an extension to SNMP agents that permits the dynamic addition, deletion or replacement of management variables in the network component's SNMP MIB without requiring recompilation of the SNMP agent. This extension of the SNMP MIB is achieved by writing so-called sub-agents that communicates with the agent via the SNMP-DPI. Sub-agents that communicate with the SNMP agent via DPI do not need to know SNMP messaging, ASN.1 details and BER encoding rules. It is also necessary to modify the SNMP agent to make it DPI enabled. The SNMP agent must keep track of all attached sub-agents and the MIB sub-trees that each of these agents support. It is responsible for de-multiplexing a managers request using the

OIDs of variables included in the variable binding list, distributing them to the appropriate sub-agents, and then collecting the responses into an overall response to the original request. The work required to modify an SNMP agent to DPI enable it is substantial but the resulting protocol is lightweight and efficient.

A sub-agent wishing to extend the SNMP agent's MIB must go through a few steps to do so. First, the sub-agent must open a connection to the agent. The sub-agent then registers the roots of any sub-trees that it wishes to provide. Once registered, the SNMP agent will handle requests for those objects from network management stations. The sub-agents are invisible to these stations, only the expanded SNMP MIB is known.

The DPI enabled SNMP agent is a standard agent modified to recognize requests initiated by a sub-agent and those initiated by a manager. For our project, we chose to modify the source of the publicly available UCD SNMPv1 agent [16] written in C.

The requests that can be initiated by a sub-agent are OPEN, REGISTER, UNREGISTER, ARE_YOU_THERE and CLOSE, all of which are for DPI control purposes, and a TRAP. The agent responds to OPEN, REGISTER, UNREGISTER and ARE_YOU_THERE with a RESPONSE packet. The CLOSE packet is just accepted by the agent which then closes the physical connection. The TRAP packet is also just accepted and then forwarded by the agent to a manager without returning any information to the sub-agent, but it is a true management information packet.

The requests that can be initiated by a manager directed towards a DPI sub-agents are: GET, GETNEXT, and SET. These are requests directed to the extended part of the MIB and it is the agent's duty to direct the request to the appropriate sub-agent based on the OIDs involved. The agent responds with a RESPONSE packet. See the bottom of Figure 2 for an overview of the DPI packet flow. In this figure, the XMS-SNMP Agent (the eXstensible Mobility Supported SNMP Agent) represents an SNMP agent enabled with both the DPI and RDPI protocols.

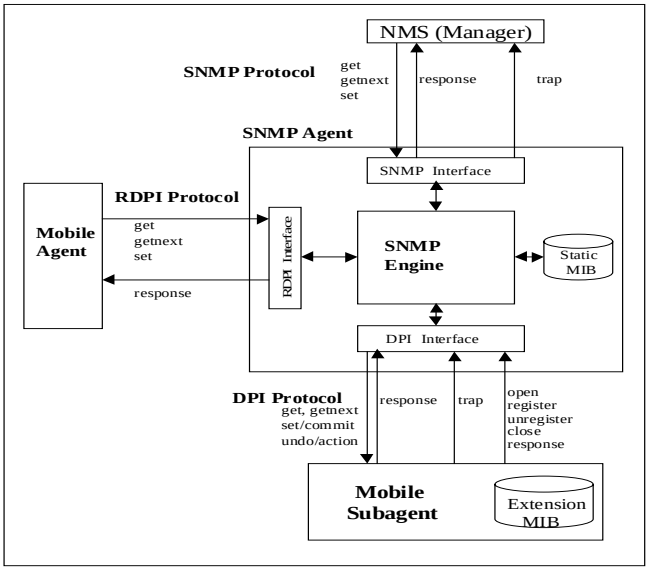

Figure 2: Functional Architecture of DPI-RDPI Agent

The Reverse Distributed Protocol Interface is new and was designed to reverse the direction of requests and responses. It will be discussed more fully in the next section. RDPI defines a method for accessing the SNMP Agent MIB without the necessity of creating full SNMP messages in SNMP Protocol Data Units (PDU), or encoding the data with the ASN.1 Basic Encoding Rules. This allows mobile agents to efficiently access the SNMP Agent MIB including any extensions provided by other mobile agents via DPI. This efficiency is realized on both ends of the protocol; the mobile agent saves while creating requests and processing responses; and the SNMP agent saves while processing requests and creating responses. The left side of Figure 2 also shows the RDPI packet flow.

4. The RDPI Protocol

The purpose of the RDPI protocol is to give a registered sub-agent access to the SNMP MIB of an RDPI enabled SNMP agent without having to undertake the overhead of BER encoding and decoding. This lightweight protocol is modeled on the DPI protocol and, in fact, reuses the PDU format for the Get and Set operations. The main difference is in the agent where the incoming request, say a SET, must fit into the agent processing stream further down the line than an SNMP Set. Modifying an SNMP agent to RDPI enable it, is in fact easier than enabling it for the DPI protocol. The savings in both message overhead and message processing make the effort worthwhile.

Table 1 below shows the Set packet format and indicates the overhead involved.

Offset (Octets)	Field	Overhead (Octets)
0	Packet length to follow (MSB to LSB)	2
2	Protocol version (=2)	
3	Protocol minor version (=2)	
4	Protocol release (=2)	
5	Packet ID (MSB to LSB)	
7	Packet type = SNMP_RDPI_SET (=3)	
8	Community name length (MSB to LSB)	2
10	Community name (if any)	
10 + L1	Object ID with two NULLs	2
10 + L2	Variable type	1
10 + L2 + 1	Length of value (2 octets, MSB to LSB)	2
10 + L2 + 3	Value	
10 + L3	Optionally more variable-bindings (Object ID, type, length, value)	
Note:	L1 = length of community name L2 = L1 + strlen(object ID) + 2 L3 = L2 + 3 + length of value	

Table 1. RDPI SET Packet and Overhead Calculation

Here we take overhead to mean the extra octets needed for encoding and decoding purposes only. For this example with a single varbind variable, RDPI has nine octets of overhead whereas an equivalent SNMPv1 Set packet would use 23 octets of overhead. For additional varbinds, SNMP uses six octets of overhead while RDPI uses five for each.

4.1 Performance Analysis

Aside from the overhead, RDPI is somewhat faster in decoding packets. For example, SNMP BER encodes every field in a generic way including header fields. RDPI, where possible, fixes the size and offset of fields. RDPI also eliminates the need for decoding "SEQUENCE OF" varbinds and n occurrences of SEQUENCE{name, value} in the variable-binding list. RDPI does not encode object identifiers in the same way as does SNMP. They are just strings with dots in them. And so on. Clearly the SNMP agent can decode an RDPI packet much faster than an SNMP packet. The larger the number of varbinds, the more prominent the efficiency of using RDPI. Furthermore, RDPI doesn't limit the packet size as SNMP does, and this is important when many SNMP packets would be involved.

The system described herein was implemented and performance measurement indicates that RDPI is about three times faster than SNMP in packet processing. The question arises as to whether such an improvement warrants a new protocol like RDPI. First of all RDPI is not really a new protocol but a subset of DPI used in a different way. (Agent-X is very similar and could be used). Anyone implementing an extensibility protocol like DPI can easily include RDPI, and the symmetry is attractive. Second, there are situations where the improved performance could be critical because large tables may need to be acquired by a mobile agent. In the high speed ATM domain for example[22], there are cases where SNMP agents already have to resort to statistical sampling because of the volume of data involved.

5. The VMC Interface Design

As indicated earlier, there are several possible designs for Mobile Agent – SNMP Agent interaction. The first is one in which the mobile agent interacts directly with the SNMP agent. This approach was discarded for a number of reasons. It requires mobile agents to be able to open sockets, which is a security issue. It also leads to larger, more complex mobile code because the overhead of communicating with the SNMP agent is not trivial, whether using SNMP or DPI/RDPI.

The second approach is to place a single intermediary VMC as an SNMP service between the mobile agents and the SNMP agent. While this approach satisfies the design requirements and does not have the problems mentioned above, it involves grouping two separate functions into a single class. Object oriented design principles would suggest that the interfaces should not be grafted together in an unnatural fashion. The approach selected for this project was to design the two functions, MIB extension and MIB access, as separate and unrelated.

Figure 3 shows the architecture of the selected approach. The entire network component is depicted, with two sub-components identified. The SNMP Agent and the Mobile Code Environment (in the JVM) contain the network management functionality of the network component.

MF: Migration Facilitator
SF: Security Facility
CF: Communication Facilitator

NC: Network Component
JVM: Java Virtual Machine
MCE: Mobile Code Environment
MCD: Mobile Code Daemon
MCM: Mobile Code Manager
MA: Mobile Agent
VMC: Virtual Managed Component
MIB: Management Information Base
VMCAccess: VMC providing MAs access to MIB
VMCExtend: VMC allowing Mas to extend the SNMP MIB

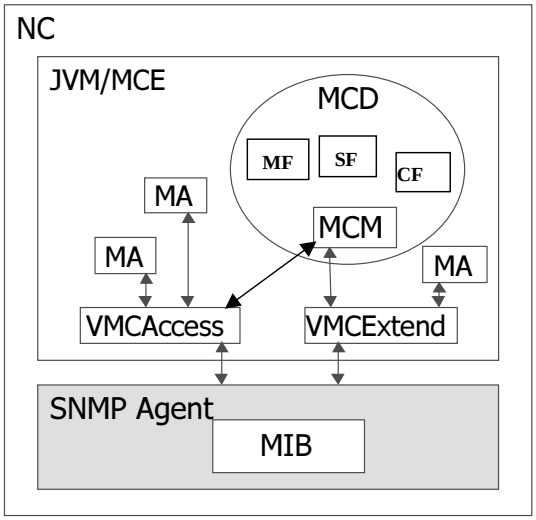

Figure 3: Dual VMC Design

Interactions between the mobile agents and VMCs are depicted as bi-directional in all cases where, in practice, the mobile agent calls to VMCAccess merely block until the response is ready. VMCExtend interactions are more complex. A VMCMIBExtend interface has been defined that provides a lightweight set of functionality modeled on that provided by the DPI protocol. Another interface, called MIBExtender, has been defined for any mobile agent wishing to extend the SNMP MIB. This interface is used by the VMC to notify the mobile agent of requests, from the SNMP agent, within the sub-tree(s) that it is providing.

The link between the Mobile Code Manager and the VMCExtend object represents the fact that the VMCExtend object will register with the Mobile Code Manager for any significant mobile agent events that might affect the provision of MIB extensions. If a mobile agent that had registered to extend the MIB becomes unavailable, through either impolite migration or premature death, the VMCExtend must deregister the mobile agent to prevent future unserviceable requests.

5.1 VMC Design

Two VMC interfaces were introduced in the previous section. The VMCMIBAccess interface is implemented by the VMCRDPI class, using the RDPI protocol to communicate with the SNMP agent. The VMCAccess component in the VMC design shown in Figure 3 is an instance of the class VMCRDPI. The VMCMIBExtend interface is implemented by the VMCDPI class, using the DPI protocol to communicate with the SNMP agent. The VMCExtend component in the VMC design shown in Figure 3 is an instance of the class VMCMIBExtend.

5.1.1 VMCAccess

Requests from any mobile agent are converted by the VMC to RDPI and forwarded to the SNMP agent. As such, the mobile agents' thread is used to process the RDPI request. The mobile agent blocks while awaiting a response. The response is processed to strip away any RDPI elements and the mobile agent receives the results

as a return value. While this serializes access to the MIB, it significantly reduces the complexity of the interaction between mobile agent and MIB. It avoids the issue of having responses arriving out of order when multiple requests can be processed simultaneously. A multi-threaded design was considered but rejected on the grounds of implementation complexity and marginal added utility.

5.1.2 VMCExtend

This VMC has a more complex set of requirements. It must act as an intermediary between the mobile agent and the SNMP agent for requests by the mobile agent to open connections, register sub-trees, de-register sub-trees and close connections. It must also handle SNMP agent request for extended MIB variables and SNMP instructions to de-register sub-trees, or close the connection.

The two event types (SNMP driven and mobile agent driven) lead to some significant design challenges. This VMC must be capable of listening for requests from the SNMP agent via the DPI port, and thus, it must have at least one thread. The question of how many threads to have is a complex engineering issue. The options that were considered were a single thread or one thread per mobile agent. A pool-based solution was also considered. A single thread would have to be complex to handle requests for multiple mobile agents. It could become a performance bottleneck because after calling the appropriate MIB function of the mobile agent (get, getnext or set) it could be blocked. While waiting, no other SNMP DPI requests could be processed, even though there are no significant resource sharing or critical section issues. This approach was discarded as slow and unnecessarily complex.

A related variation is one by which a single agent that neither blocks when listening on the DPI port for an SNMP message nor when awaiting a mobile agent extended MIB response to a previous SNMP DPI request. The VMC must be able to map the MIB sub-tree requested with the appropriate MIB extending mobile agent. This internal table could grow quite complex and would need to be protected from multiple accesses that could violate its critical sections. Thus, this design would appear to merely move the bottleneck deeper into the VMC to the sub-tree / mobile agent linked data structure. This problem surfaces again in the opposite direction. Any request forwarded to a mobile agent must be remembered by the VMC. Upon completion, the return message must carry the appropriate DPI packet number to let the SNMP agent know to which request the response is associated.

A third approach would be to create an individual thread for each MIB extending mobile agent. The requests originating from the mobile agent (open, register, deregister and close) would be handled within the thread of the mobile agent itself. Requests originating from the SNMP agent would be handled by a devoted listening thread that would then call the appropriate interface functions of the MIB extending mobile agent.

This scheme has obvious design advantages in that there is no need for a complicated central registry that must be accessed for each event. The implementation of a class that is concerned only with one mobile agent and one DPI connection would obviously be more straightforward than those mentioned previously.

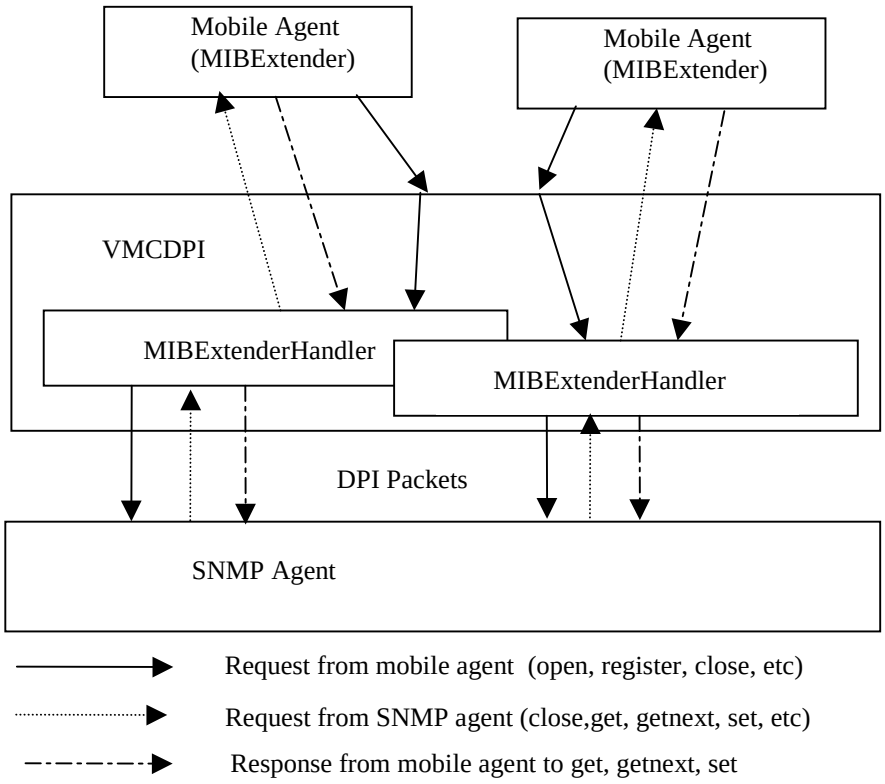

Figure 4: MIBExtender Handlers

Thus, the implementation chosen was to create a one to one mapping between mobile agents (MIBExtenders) and associated VMC sub-components, called MIBExtenderHandlers. These handlers are active objects that handle both directions of communication between the SNMP agent and mobile agent. This is shown in Figure 4.

In this design, the VMC class itself is a passive component that handles the creation of the extender-handler mapping, and holds records of the mappings and sub-tree registration lists. It mainly acts as an initial point of access for MIBExtenders and a shared resource for the active MIBExtenderHandlers. While there is a potential bottleneck in the central registry of mobile agent/extender handler mappings, this is negated by the fact that access to this database is only required when a mobile agent requests a new connection. All of a mobile agent's open

connections are handled via a single VMC thread to ease any critical section complexity during implementation. Since the thread blocks within the mobile agent's thread during a request, any subsequent DPI requests are queued lower down in the IP protocol stack.

In this implementation, a central (within the VMC) registry of registered sub-trees is also maintained. The DPI protocol itself provides for multiple overlapping registrations. The protocol calls for the DPI enhanced SNMP agent to only forward SNMP requests to the MIB extender with the highest priority. This concept could lead to inconsistent situations where a central manager does not truly know which MIB extender has serviced the request. If a MIB was merely a read-only database, that would be bad enough, but SNMP agents and mobile agents can have their states changed via a set request. No self-respecting central manager will issue such an instruction when it cannot know which SNMP DPI sub-agent will be registered with the highest priority.

To provide deterministic runtime behavior, the capability has been included, in our implementation, to have the VMC reject overlapping sub-tree registration attempts.

6. Application to PVC Configuration and a Mobile Agent MIB

Consider a network of heterogeneous ATM switches. No standard protocol exists for the configuration of permanent virtual circuits and so users resort to fax, telephone etc. to establish permanent connections. The approach we developed based on mobile agents [17], [18] utilizes a mobile configuration agent to serially visit the nodes or delegation agents to go in parallel, and in addition handle resource allocation exceptions.

In any event, a mobile agent arrives at a switch with a resident SNMP agent and needs to access its ATM MIB to check on VPI/VCI usage and bandwidth availability. It also needs to set values in a MIB table in order to establish a cross-connection. In this scenario, the mobile configuration agent looks up the VMCAccess VMC by contacting the Mobile Code Manager and then using the VMCMIBAccess interface implemented by it in order to interact with the resident SNMP agent. In case of failure to acquire resources like VPI/VCI indicators or bandwidth, the serial mobile agent needs to backtrack to the previous switch. This approach was compared with parallel delegation agents communicating with each other for exception handling[18] and in some circumstances can perform better. The main motivation for the mobile agent approach however was to provide the intelligence imbedded to handle the heterogeneity of the individual switch agents.

For the purpose of managing mobile agents, each MCD needs to keep track of the number of visiting mobile agents, their history, the resources they consume, and other pertinent information. With the above design, a convenient way of implementing this local mobile agent MIB is by VMCExtend extending the SNMP MIB using DPI. If there is no resident SNMP agent, then VMCExtend can provide the same information via the abstract interface VMCMIBExtend in a transparent way, one of the advantages of the design.

7. Conclusions

This paper clearly identifies the need for mobile agent solutions to network management problems that acknowledge the utility of legacy solutions and protocols. SNMP is arguably the most important network management protocol in service today; the research reported here acknowledging this argument by extending an existing mobile agent framework already used within the network management domain [19], [20]. The implemented design exploits the standard DPI protocol and proposes an extended RDPI protocol in order to enhance the interaction of mobile agents with SNMP agents. The dual VMC approach implemented has been used to solve the ATM PVC configuration problem [18], with interesting results. At the time of implementation, DPI code was most readily available but this has evolved into the Agent-X protocol which is very similar. Conversion to Agent-X would be very straightforward.

Acknowledgements

We would like to acknowledge the support of Communications and Information Technology Ontario (CITO) and the Natural Science and Engineering Research Council (NSERC) for their financial support of this work. We would also like to thank Peter Drake and Patricia Cuesta Rivalta for their contributions to the work reported here.

References

[1] Zapf M., Hermann K., and Geihs K., "Decentralized SNMP Management with Mobile Agents." In Proceedings of IM '99, Boston, May 1999.

[2] Ferudin M., Kasteleijn W., and Krause W., "Distributed Management with Mobile Components." In Proceedings of IM '99, the 6th IFIP/IEEE International Symposium on Integrated Network management, pp857-870, Boston, 1999.

[3] White T., Bieszczad A., and Pagurek B., "Distributed Fault location in Networks Using Mobile Agents" In Proceedings of the Second International Workshop on Intelligent Agents in Telecommunications Applications (IATA '98), Springer Verlag Pub., Paris July 4th - 7th, 1998.

[4] Pagurek B., Li Y., Bieszczad A., and Susilo G., "Configuration Management in Heterogeneous ATM Environments using Mobile Agents." In Proceedings of the Second International Workshop on Intelligent Agents in Telecommunications Applications (IATA '98), Springer Verlag Pub., Paris July 4th - 7th, 1998.

[5] Baldi M., Gai S., and Picco G. P., "Exploiting Code Mobility in Decentralized and Flexible Network Management", First International Workshop on Mobile Agents Mobile Agents'97, pp 13-26, Berlin, Germany, April 7-8, 1997.

[6] Yemini Y., Goldszmidt G., and Yemini S., "Network Management by Delegation." Proceedings of ISINM '91, Integrated Management II (Krishnan and Zimmer Eds.) pp. 95-97, North Holland Pub., April 1991.

[7] Levi D., Schönwälder J., "Definitions of Managed Objects for the Delegation of Management Scripts. RFC 2592, May 1999.

[8] Levi D., Schönwälder J., "Definitions of Managed Objects for Scheduling Management Operations. RFC 2591, May 1999.

[9] Susilo G., Bieszczad A., Pagurek B., “Infrastructure for Advanced Network Management Based on Mobile Code.” In Proceedings of NOMS '98, New Orleans February 1998. See also http://www.sce.carleton.ca/netmanage/publications.html
[10] GRASSHOPPER - The OMG-MASIF conformant Mobile Agent Platform, http://www.ikv.de/products/grasshopper/index.html.
[11] Bieszczad A., White T., and Pagurek B., "Mobile Agents for Network Management." In IEEE Communications Surveys, September 1998.
[12] Wijnen B., Carpenter G., Curran K., Sehgal A., Waters G. "The SNMP Distributed Protocol Interface", Version 2.0, RFC 1592, March 1994.
[13] Daniele M., Wijnen B., and Francisco D., "Agent Extensibility Protocol Version 1", RFC2257, January 1998.
[14] Agent Extensibility Issue, The Simple Times, Volume 4, Number 2. April 1996, http:// www.simple-times.org
[15] Wang Y., "Integration of a Mobile Agent Environment with Legacy SNMP." M.Eng. Thesis, Carleton University, Dept. of Systems and Computer Engineering, Ottawa Canada, August 1998.
[16] UCD-SNMP Implementation, ftp://ftp.ece.ucdavis.edu/pub/snmp/ucd-snmp.tar.gz
[17] Pagurek B., Li Y., Bieszczad A., Susilo G., "Network Configuration Management In Heterogeneous ATM Environments." In Proceedings of the Second International Workshop on Agents in Telecommunications Applications (IATA'98), Agent World' 98, Paris, France, July 4th-7th, 1998.
[18] Boyer, J., Pagurek, B., White, T., "Methodologies for PVC Configuration in Heterogeneous ATM Environments Using Intelligent Mobile Agents." Proceedings of MATA '99, Ottawa, Oct. 99. World Scientific Publishing Co., Singapore. pp211-228.
[19] White T., Pagurek B., and Bieszczad A., "Network Modeling For Management Applications Using Intelligent Mobile Agents," accepted for publication in a special issue on Mobile Agents of the Journal of Network and Systems Management to be published in September, 1999.
[20] Knight G., and Hazemi R., "Mobile agent based management in the INSERT project," accepted for publication in a special issue on Mobile Agents of the Journal of Network and Systems Management to be published in September, 1999.
[21] Gavalas D., Greenwood D., et al "Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology" to appear in Computer Communications, Special Issue on Mobile Agents in Communications, 2000.
[22] Bierman, A., and McCloghrie, K., "ATM-RMON MIB: Remote Network Monitoring MIB Extensions for ATM Networks", IETF draft-bierman-rmon-atmrmon-01.txt, May 1996.
[23] Simoes P., Silva L., Boavida F., "Integrating SNMP into a Mobile Agents Infrastructure" Proc. of IEEE/IFIP DSOM '99, Zurich Oct.1999.
[24] Simoes P., Reis R., Silva L., and Boavida F., "Enabling Mobile Agent Technology for Legacy Network Management Frameworks" Proceedings of IEEE Softcomm '99, Oct. 1999.