Towards Securing Network Management Agent Distribution and Communication

Larry Korba
National Research Council of Canada
Rm. 286B, Bldg. M-50, Montreal Road
Ottawa, Ontario K1A 0R6
larry.korba@iit.nrc.ca

1.1 Abstract

Network Management applications using distributed software agents require secure techniques for agent distribution and communication. Although there are several popular mobile agent environments written in the Java programming language that contend to offer these functions, these packages are generally heavy weight and do not provide a framework for authenticating agent responsibilities within agent systems. This paper describes an agent distribution and communication framework for Java-based software agents. The framework has several key features: 1) a secure mechanism for dispatching agents to selected nodes from secured agent repositories. 2) an operating environment for each agent that depends upon its role-based authenticated security level. 3) object-based, peer-to-peer and peer-to-many communication. 4) provision for encrypted communication. 5) a low overhead distributed naming directory of agents populating the network. 6) proactive agent monitoring for secure operations. 7) tools for reflective monitoring of agent contents and communication. 8) a lightweight environment that is easy to use while keeping the agents small in size.

Keywords

Network Management, Mobile Agent, Distributed System, Software Agent, Authentication, Security

2. Introduction

A distributed approach for network management applications offers distinct advantages over a centralized paradigm. Key factors in favor of the distributed approach are the redistribution of network traffic load and the distribution of processing requirements. Collocation of network management processes or agents with the resources they manage may let agents communicate locally, rather than over the network. The agent obtains local information about node or network performance. At the same time it deploys local computing resources to process the gathered information. Putting diagnostic intelligence at managed nodes also increases the potential for network management process survivability. A node may be managed even when the connection to the computer hosting a monitoring function is not functional.

NRC Paper Number: 41616

The introduction of the web and more recently the Java programming language has resulted in an explosive push toward distributed agent-based Network Management applications. To further this movement, several organizations have developed agent systems in the Java Programming language for deployment in network management and other applications. Some basic requirements for a network management agent environment include the ability to distribute agent entities throughout the network, implementation of measures to ensure safe operations by the agents and the agent environment, and support for different types of intercommunication among the distributed agents.

Fundamental to successful deployment of mobile agent systems is meeting security requirements [1]. Some research within the network management community is focussed on full agent mobility. Unfortunately fully mobile agents significantly compound security issues [2]. A major concern when deploying these systems is the ability to prevent illicit operations by any of the distributed entities. Illicit or unwanted operations include:

- Agent corruption within their storage or operating environments. Agents need a secure repository for their storage or operating environment to ensure that they are not corrupted. For itinerant agents, each agent meeting place must be secured.
- Inappropriate agent transfer or corruption of an agent during transfer. The transfer of an agent must be secured and sources verified to ensure that only verified agents are allowed to operate in the agent environment and to ensure that the agent is not corrupted in transit. It is important to maintain the integrity and privacy of the agent and the information it may carry.
- Communication interception and corruption. Communication between distributed agents must be handled in a manner to guarantee that the sender and receiver are allowed to communicate, and the message exchanges are not easily deciphered or modified by a third party.
- Corruption or damage to network resources. Depending on the mandate of an agent, it may be restricted to accessing only certain network resources.
- Corruption or damage to host resources. Agents must be restricted from unauthorized access to resources on the host. The goal here is prevention of potentially harmful access or changes to host resources.

This paper first highlights the development of distributed approaches to network management. Several of the better known environments for mobile agent deployment in the Java programming language are also compared with a focus on their communication and security aspects. Finally, the paper describes an agent communication and distribution environment (ACDE) targeted for the development of Network Management applications.

3. Background

3.1 Agents and Network Management

Within the last decade software vendors and researchers have seriously considered a distributed approach to Network Management applications. Management by Delegation (MBD), for instance, is an attempt to provide improved operating characteristics over the centralized SNMP paradigm through decentralization [3]. The distributed approach of MBD

supports a means for creating new functions delegated by other processes. Appropriate decentralization reduces the impact on bandwidth for a network management system and partitions the application and resources. One application area suggested by Meyer as ideal for MBD is security, i.e., distributed intrusion detection [4]. Suzuki et al. demonstrated a delegation approach using scripted agents [5]. Their work demonstrates the effectiveness of distributing processes for network management depending on data transfer requirements. Although using common management information protocol (CMIP), the approach is applicable to the SNMP world.

Another approach to the distribution of network management functions is the Managing Agents for Information Control system (MAGIC) from the Technical University of Munich [6]. Their architecture introduces an intermediate layer (MAGIC) that interacts directly with SNMP Agents and management stations. MAGIC agents have two SNMP ports. They are SNMP agents that implement their own Rule-MIB. The Rule-MIB provides an SNMP-standardized way for implementing rules that employ managed node MIB Objects, events and timers within expressions to monitor and interact with a managed NEs.

The Versatile Network Management System (VINES) also distributes sources of information, algorithms and services [7]. This system offers management client applets, which interact with network management Domain Servers (DS) for groupings of NEs. A DS is essentially the union of the MIBs for all NEs within a domain. The DS layer interacts with those MIBs under control of the management applet.

Mobile autonomous agents have also been proposed and used for Network Management. Hjalmtysson and Jain have proposed an agent-based architecture for service management and provisioning in the telecommunications field [8]. Agent mobility conceptually separates service abstraction from location of execution. An agent may be dispatched on an as needed basis to perform different service functions. This facilitates load balancing, service quality, resource allocation, provisioning and service maintenance. The notion of “Plug and Play networks” as put forward by Bieszczad et al. encapsulates some of these notions [9].

Although the ACDE environment may be used to implement a variety of distributed network management frameworks, our current security applications follow the management by delegation approach. Agents, equipped to interact with network resources, such as those available via SNMP, may monitor and respond proactively to the state of the resources. Agents may absorb new functions delegated by other agents or processes. The agent may create events to pass information to managing agents that in turn may delegate new agents to deal with those events or collate the information for centralized management or logging.

3.2 Security and Communication for Mobile Agents

There are currently many agent environments developed by universities or research institutions and by companies [10]. Kiniry and Zimmerman provide a more general overview of the more popular systems systems [11] [12]. This section briefly outlines the security and communication features for the following environments: Concordia [13], Voyager [14], Aglet’s Workbench [15], and Odyssey [16].

Currently, the four packages mentioned above implement extensions of the java.lang.SecurityManager to provide a sandbox in which mobile agents may operate. The only package that goes well beyond this approach is Mitsubishi's Concordia. Concordia provides the most in-depth security measures [17] with the inclusion of secure sockets layer V3, encrypted storage of agents and the use of a user identity. These more sophisticated security modules are only available with the licensed version of the software. One security feature Concordia does not provide is authentication of mobile agents.

In recognition of the need to bolster security facilities for Aglets, the developers have suggested a security model that defines an authorization language specifying the principals within an Aglet system and the responsibilities of each of them within the system [18]. The model also defines how the agents might migrate and access local resources. This work does not define the infrastructure for distribution of the security information nor does it address the problem of protection of the internal state of Aglets.

In terms of communication, Aglets provides a message-based, socket level communication. Concordia offers both distributed multicast events and collaborative communication based on Java's Remote Method Invocation (RMI) API. Objectspace Voyager uses CORBA services for communication. Odyssey uses RMI as its standard communication facility while providing DCOM and IIOP as options. No package provides fine grain monitoring of agent communication.

In contrast, ACDE uses a role-based security model for handling access to host and network resources. Role-based access control has been used by others to manage intranet resources for instance [19]. In a similar fashion to this approach, all network objects are given a unique identity, along with permissions, properties and access control lists. Before an agent may be started, it must first be authenticated through exchanges with a security server. It is only allowed to access resources permitted through its control lists based upon the agent's role within an agent system. ACDE also has a facility for fine grain monitoring of agent communication.

Agent communication in ACDE is object-based but lightweight. There are provisions for both multicast and peer-to-peer communication. While fine grain communication monitoring provides additional security benefits, it also provides an aid in system development. The next section provides more details of the security and communication provisions of ACDE.

4. System Overview

The initial target for ACDE is distributed, agent-based, security applications in the area of Network Management. Secured access to wireless local area networks is the first application of this system [20]. In this system, five different agent types interact with each other to form an agent system that adds a user authentication application layer for wireless LANs. Agents monitor network elements for unauthorized access to wireless services. The wireless LAN security system automatically configures itself by sending agents to operating nodes where they will be most effective. As network usage or configurations change so does the agent system configuration.

The system to support this and other security applications was developed with the following goals:

- Provide a method for one-hop agent dispatch from secure agent storage,
- Authenticate agents on the basis of their roles within agent systems,
- Provide negotiation for and security management of network and host resources,
- Offer agent communication via remote object sharing and multicast events.
- Ensure secure Agent communication,
- Monitor and log resource usage and,
- Provide a lightweight, distributed naming service to support multiple authorized managers for the agent system monitoring.

All agents are digitally signed in ACDE. Single-hop agents simplify the security infrastructure. In the current version of ACDE agents are only allowed to move from a ”safe” agent storage place (an access-restricted directory of a web server) to an agent meeting place called the Agent Daemon. A three-tiered architecture supports extension of the agent environment to the sandbox of a web browser. All agents have predefined roles for operation within an agent system. The predefined roles of an agent include information concerning its resource requirements, and levels of interaction and authority in dealing with network and host objects (databases, computer resources, network resources, Agent Daemons, or other agents). A trusted security server stores information regarding the agent systems and agents. Agents are authenticated via the trusted server when they arrive at an Agent Daemon. Agents operating within Agent Daemons and within browsers must negotiate with their host Agent Daemon for network resources. Finally, in order to operate in the Agent Daemon environment, agents must implement an interface permitting Agent Daemon monitoring of its communications.

4.1 One-Hop Agents

For many network management applications, mobile, multi-hop agents are unnecessary and undesirable. It has been argued by many that the key benefit of agent mobility is dynamic resource usage or load balancing for distributed applications [21]. Balancing of resource usage may be easily accomplished with one-hop agents. Using one-hop rather than multi-hop agent mobility lowers the complexity of the resulting agent infrastructure and greatly reduces security exposures. The agent mobility of ACDE offers a means for populating a network with agents required for distributed network management systems and adapting the system for optimal operation. When initially deploying an agent system, mobility provides a means for populating the network appropriately, depending on network and computer resources. As the network characteristics change, the agent system may adapt to the changes by retracting or adding more agents.

It is much easier to establish trust for one-hop agents as opposed to multi-hop agents. A one-hop agent travels between known agent home servers and the execution environment, while a multi-hop agent may roam widely among many target environments. Trust must be established between the execution server (Agent Daemon) and the agent’s owner. The execution server must also evaluate the level of trust to be put in the agent home server. In an Intranet situation, where there is no exposure to the Internet, trust may be reasonably high, given that it is reasonable to assume that the access to the agent home servers will be limited to the network administrator. In the more general case, the agent system owner may digitally sign the

information comprising an agent. Within ACDE, agents are authenticated at the Agent Daemon.

Since agent distribution, in the one-hop case, requires mediation of the agent home server, trust may be assigned on the basis of server owner. Thus within an organization there may be many servers offering many agent system components. The execution server may have a security policy that provides a level of trust and access to resources dependent upon the agent home server owner. In the case of ACDE, the agent system owner digitally signs all agents. Not only are agents signed and authenticated to assure agent authorship and delivery, they are also registered with a trusted server to provide a system of security based upon the role of the agent in an agent system.

4.2 System Block Diagram

The ACDE deploys Agent Daemons throughout the network. Agent Daemons provide an environment at network nodes to which agents may be dispatched. They allocate resources based upon agent authentication, and agent requirements and authority within agent systems. Several managers within the Agent Daemon interact with agents to deliver agent services. Figure 1 depicts the functions supported by the Agent Daemon.

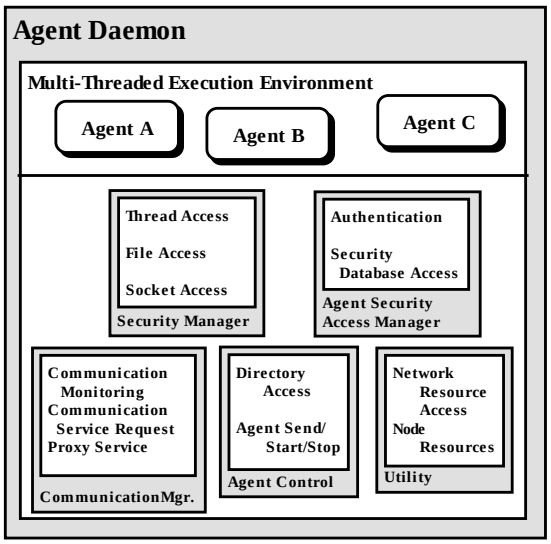

Figure 1: Block Diagram showing the elements of the Agent Communication and Distribution Environment.

A Security Manager maintains the parameters of the Java sandbox for each agent operating within the Agent Daemon. The Security Manager extends the java.lang.SecurityManager (JDK 1.1x) class to monitor activities within agent threads. It handles issues such as the level of access to the host file system (open, read write, delete), and socket access, among many others [22].

The Agent Security Access Manager handles access to the secure server. Whenever an agent or user requires authentication, this manager provides the channel for those negotiations. Any

agent wishing to access information from the secure server regarding agent systems for instance does so through the Agent Security Access Manager.

The communication manager handles requests from agents for communication services. It also monitors agent communication activity, allowing only authorized communication for the agent. The communication services controlled by the Agent Daemon include: publishing objects, requesting objects from other agents, requesting network-based Agent Daemon services such as multicast services and use of sockets. To allow an applet to communicate with an agent system, the communication manager also offers a proxy service. An Agent Daemon running on the same node as the web server hosting the applet provides an authorized applets with communication access to the agent system via the communication manager.

When an agent wishes to publish an object, it must negotiate with the communication manager for available resources. The communication manager determines from the trusted server whether or not the agent is authorized to use communication resources. The trusted server contains reference information about the agent and the agent system(s) with which it is associated. Among that information is a list of the allowed types and targets of interactions for agent communication.

Agents or applets wishing to control the details of the other applets must use the Agent Control Manager. Agents authorized for agent control may send agents to destination Agent Daemons, start and stop agents, and access directory services.

A set of utility classes provides access to methods that are useful in assessing or diagnosing computing or network performance. These utilities may be employed to determine the best physical locations for agents to operate as part of an agent system within the network. For instance, an agent may require extensive computing resources. By querying performance assessment methods within this class at various Agent Daemons, the agent may be sent to the agent with the ideal set of free resources to optimize agent performance.

4.3 Communication

To operate in an ensemble, distributed entities must be able to communicate with each other. The ACDE provides an easy-to-use, versatile, yet lightweight communication infrastructure. Many mobile agent packages offer object communication using either CORBA or Java's RMI. These communication techniques abstract away the communication interface for object exchange and remote method invocation. A local method invocation may have its arguments or references serialized and transferred to the target method in a remote class. Any objects returned from the remote method are serialized and returned to the entity that remotely invoked the method.

CORBA is designed to work across languages. To do this CORBA uses Object Management Group's Interface Definition Language to provide the communication interface. Although IDL provides "the best standard notation language available for defining component boundaries" [23], mapping IDL constructs into programming languages adds a significant level of overhead.

Java RMI (supported within JavaSoft's Development Kit (JDK) Ver. 1.1 and above) takes advantage of Java's Object model. It is a much simpler architecture than CORBA and easier to use. There are some drawbacks to the approach. A number of steps and rules are required to

make a Java class remotely enabled [24]. It is reasonably heavy weight, especially that the number of classes and interfaces representing the API for the RMI amount to twenty-six (JDK Version 1.1.3). In addition, an RMI Registry program must also be running for each node with remotely enabled objects.

ACDE builds upon the communication techniques developed by Ennio Grasso at CSELT (Centre Studi e Laboratori Telecomunicazioni S.p.A., Italy). Known as the Java Reflection Broker [25], the package uses several of Java's APIs: Sockets, Object Serialization and Reflection to deploy a dynamic invocation model similar to the dynamic invocation interface in CORBA. In contrast with CORBA or RMI, remotely enabling a Java class requires no modifications to the class. No stubs and skeletons need be developed for remote enabling. The communication package works with JDK 1.1x compliant browsers. To publish an object, the agent or applet creates for itself something akin to a mini ORB. The remote method invocations for JRB provide true peer-to-peer communication. The package also provides an event service for asynchronous, multicast, connectionless communication. In this one-to-many communication model, a sender's message may be received by a number of recipients, registered for the message type event.. Not only is the package easy to use, it is also compact, requiring a class file size of 14K bytes for both client and server sides.

The communication services have been extended for robust communication for multiple agents operating within a single Agent Daemon and to include a "back door" monitoring facility for agent communications. To operate within an Agent Daemon, agents must maintain objects reflecting the identity, nature and contents of communications with its published class. An Agent Daemon monitors information within the agent regarding recent data exchanges. The information includes communication streams, identities of entities connecting for remote invocation, multicast events requested and received.

Secure Sockets Layer version 3 (SSL v 3) provides a means for securing communication interchanges between agents within agent systems operating under ACDE. This protocol authenticates and encrypts Transmission Control Protocol streams. The protocol is popular and appears to have become a standard for secure network communication. There are a number of third party implementations available in Java, for instance [26].

4.4 Security Management

As discussed above, the approach of using single-hop agents eases some of the security requirements from the distribution point of view. With ACDE, agents may also pass state information to other agents. Authorized agents may destroy or deploy other agents. This process is intended for populating a network with an agent system or for dynamic load balancing, i.e. load balancing for an agent system while in operation.

Within web browsers, Java's sandbox approach to security management severely limits an untrusted applet's access to host and network resources. Using digitally signed applets can relax the sandbox restrictions [27]. After creation, classes are signed with a digital certificate based upon the class's author and class contents. Successful authentication of delivered classes assures first of all that a class has not been modified during transfer. Second, systems can be configured to allow different levels of privileges depending on the author of the class.

In ACDE all agents are digitally signed. This assures that the classes received by Agent Daemons are the same as the classes created by the specified author and stored at the server.

Rather than allowing the verified, signed agent immediate access to network or host resources, the Agent Daemon authenticates the identity and role of the Agent through interchanges with a trusted server. This approach authenticates the agent as a bona fide member of an agent system scheduled to operate within the network. The interchanges with the trusted server also provide an added level of assurance against agent corruption and provide a means for restricting or relaxing agent operations on the basis of agent role within an agent system. When agents and agent systems are created, their names, roles, required resources, relationships between agents and agent systems, and lifetimes of agent systems are stored in the trusted server. Figure 2 provides a schematic diagram of typical communication interchanges for ACDE operation.

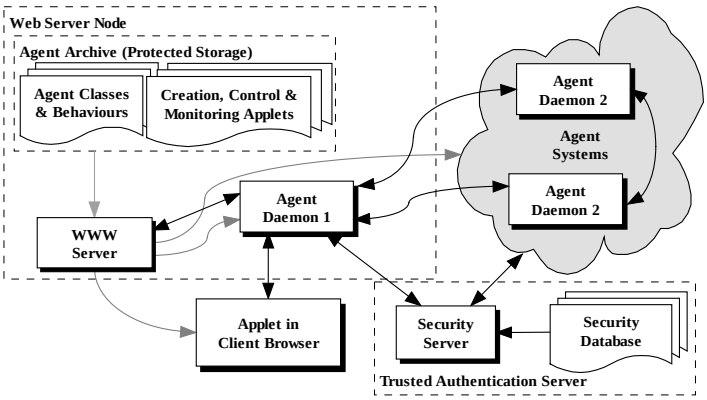

Figure 2: This diagram illustrates the some communication operations of ACDE. The gray lines indicate agent transfers. Solid lines indicate communication exchanges. Applets communicate with agent systems using a three tiered arrangement.

The Trusted Security Server contains sensitive information needed for authentication exchanges, key distribution, security audit. Figure 3 illustrates the organization of Agent System information while Figure 4 shows how the Agent object is organized. A developer uses a Security Management program to interact with the Security Server and edit these objects when creating an Agent and an Agent System. These objects also relate Agents to the communication relationships with other Agents, Agent Daemon services, and network services.

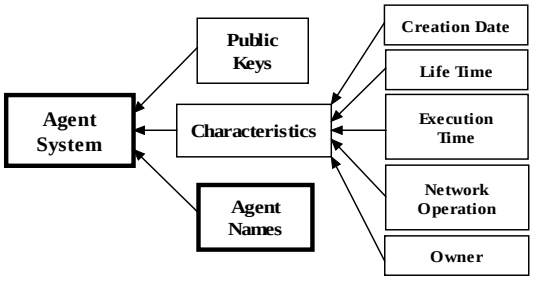

Figure 3: Some of the information defined in the Agent System Object.

The Security Server provides two functions: 1) generation of ANSI X.9.17 keys for authenticated agents and users and 2) management of a log of security events. Data Encryption Standard (DES) is the basis for encryption used by the Security Server. Although 56 bit encryption keys are considered weak by today's standards [28], DES provides reasonable software implementation speeds and adequate protection against attacks from individuals or organizations other than those with massive resources.

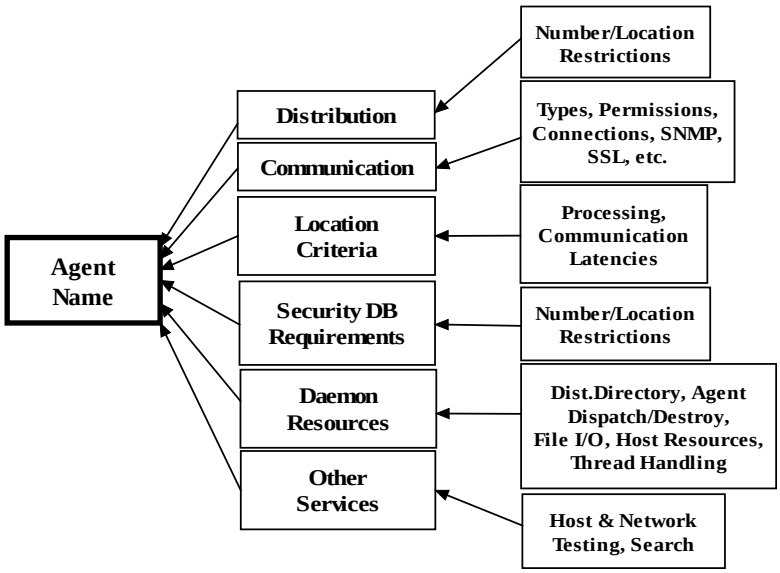

Figure 4: Some of the information defining the Agent object.

The process of Agent Daemon and agent authentication, and agent resource negotiation proceeds as follows:

- Whenever a user starts an Agent Daemon, its digital signature is checked through interchanges with the Trusted Authentication Server. A user name, password authentication exchange permits an Agent Daemon to access agents stored in a protected area of a web server and to act as a client for authentication exchanges with the Trusted Server on behalf of deployed agents.
- The Agent Daemon loads agent classes from the remote agent storage area. The Agent Daemon's class loader loads the agent as a signed class. After the agent' digital signature has been verified, it is still not yet considered by the Agent Daemon to be trusted. Using Java's Security Manager mechanism, the agent is not allowed access to resources, other than communication with the Agent Daemon security and communication managers until step 3 is completed.
- The Agent Daemon interacts with the Trusted Authentication Server to authenticate the agent as a member of an agent system scheduled for network deployment. The agent name is passed in plain text, the hash value derived from the Agent certificate provides a simple means for securing a channel for exchanging key information. Agent security

information and agent system information is exchanged using a key generated for the agent by the Trusted Authentication Server.

- The Agent Daemon and the trusted server perform a nonce exchange periodically to ensure the viability of certificates for agents running at the Agent Daemon node [28].
- If the agent has authority to publish classes for communication, it must negotiate with the Agent Daemon for allocation of communication resources for this purpose immediately. If the Agent does not do so, the Agent Daemon kills the Agent and adds a security violation to the event log.
- The Trusted Authentication Server monitors, time stamps and logs several different events including:
  - Successful and failed attempts of access to the security database.
  - Breaches within the agent daemon security manager. For instance, the Agent Daemon notifies the Trusted Server if an agent attempts to overstep its bounds in terms of host or network resources, or contact with other entities.
  - Creation and destruction of agents.

The Trusted server notifies the system administrator whenever any of several settable trigger events occur. One such trigger event is a repeated authentication failure for an Agent trying to operate at an Agent Daemon. If this should happen, the Trusted Server sends an electronic message to the system administrator. As well, subsequent requests from the source address for further authentication are rejected until the system administrator investigates. These events typically would be evidence of a brute force attack of an alien agent or agent daemon.

4.5 Agent Monitoring and Control

For multiple agents to negotiate share or allocate tasks make arrangements with each other, etc. they must communicate. While vital for agents to do their job, agent systems with many communicating agents can be very difficult to build without appropriate communication monitoring tools. A communication monitoring class with ACDE permits monitoring of agent communications by Agent Daemons. An Agent Daemon or an authorized agent or applet may remotely set environment settings within agents causing them to queue communication exchanges they have with other entities. The authorized agent may also inspect the internal methods of the published agent class.

Figure 5 illustrates the typical functions available within ACDE for manual monitoring and control of agents.

The image is a screen shot of the control window for an Agent Daemon. Fields within this window list:

- The IP address of the Agent Daemon.
- The IP address of the Agent Server, in this case a web server located at the same IP address as the Agent Daemon
- A list of Agents available for distribution from the agent repository (Agents for Distribution Box). The contents of agent archives (e.g. MyBlink.zip) may be examined remotely.
- A list of agents running at the Agent Daemon with their assigned names in the box named Running Agents.

- A list of methods available for remote execution for the highlighted agent in the Running Agents box.
- A list of the methods available in the selected agent. This is a reflectiive service for examining details of the methods available for remote invocation.
- A list of the latest invocation events. The list shows the invoking method, the method invoked and the time of the invocation.
- A list of the multicast events received by the Agent Daemon.

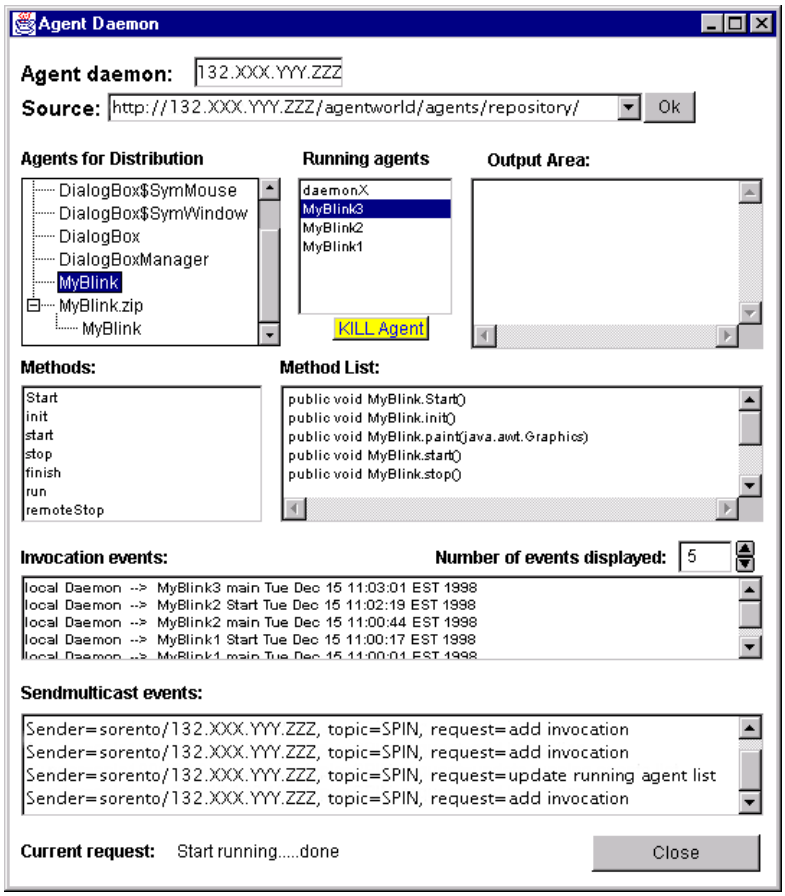

Figure 5: A screen snapshot of the control and monitoring window of an Agent Daemon.

5. Discussion

This paper introduces an environment for secure distribution and communication for software agents. It was designed to accommodate development of Network Management security applications. The ACDE offers a lightweight environment, deploying one-hop agents that present the potential for increasing the level of trust between agents and agent operating

environments. Agents communicate using a package that provides both multicast and peer-to-peer object communications. These communication streams may be monitored. Monitoring provides automated fine-grained censorship of agent communication and manual monitoring of agent communications during development or for analysis. Central to the system is a mechanism for agent authentication and for authorized agent access to various network and computing resources, on the basis of the roles of agents within agent systems.

While providing advantages for rapid prototyping and testing of distributed systems created in the Java programming language, development in other languages is not precluded. Java's native language interface provides one facility for bridging the language gap. CORBA, although a more heavyweight solution, is yet another approach for legacy or multi-language systems.

Using a centralized security server approach for authentication provides a rapid technique for updating and distributing agent and agent system information. For instance, if there is a change in an agent system configuration, or its security requirements, only the information in the security server needs to be changed. The key drawback to this approach is the fact that the system requires that the security server is operational at all times. A successful denial of service attack on the security server would effectively disable ACDE operation. A common technique for countering this type of attack is to use a backup server or a series of security servers.

Currently the system is being deployed and extended while new security applications are under development. The initial applications include secured access for wireless local area networks and a distributed intrusion detection system.

6. References

[1] C. G. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: Are they a good idea? Technical report, IBM Research Report, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY, March 1995. http://www.research.ibm.com/massive.

[2] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for mobile agents: Authentication and state appraisal. In Proceedings of the Fourth European Symposium on Research in Computer Security, pages 118-130, Rome, Italy, September 1996. Springer-Verlag Lecture Notes in Computer Science No. 1146.

[3] Y. Yemini, G. Goldszmit, S. Yemini, Network Management by Delegation. Proceedings of the 2nd Int. Symp. On Integrated Network Management, pp. 95-97, Washington, DC, April, 1991.

[4] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldsmidt, Y. Yemini, Decentralizing Control and Intelligence in Network Management. Proceedings of the 4th Int. Symp. On Integrated Network Management, Santa Barbara, CA, May 1995.

[5] M. Suzuki, Y. Kihira, S. Nakai, Delegation Agent Implementation for Network Management. IEICE Transactions on Communications, 1997, V.80, No.6, pp. 900-907.

[6] M. Trommer, R. Konopka, Distributed Network Management with Dynamic Rule-Based Managing Agent. Proceedings of the 5th Int. Symp. On Integrated Network Management, San Diego, CA, 12-16 May 1997.

[7] G. Mansfield, E.P. Duarte Jr., M. Kitahashi, S. Noguchi, VINES: Distributed Algorithms for a Web-Based Distributed Network Management System. Proceedings of the 5th Int. Symp. On Integrated Network Management, 12-16 May 1997, San Diego, CA, pp. 281-294.

[8] G. Hjalmtysson, A. Jain, Agent-Based Approach to Service Management- Towards Service Independent Network Architecture, Proceedings of the 5th Int. Symp. On Integrated Network Management, 12-16 May 1997, San Diego, CA, pp. 715-729.

[9] Bieszczad, A. and Pagurek, B. (1997), Towards plug-and-play networks with mobile code, in Proceedings of the International Conference for Computer Communications ICCC'97, November 19-21, 1997, Cannes, France.

[10] Agent Tools List http://www.agentbuilder.com/AgentTools/

[11] Joseph Kiniry and Daniel Zimmerman. A Hands-on Look at Java Mobile Agents. IEEE Internet Computing, July-August, 1997.

[12] Joseph Kiniry and Daniel Zimmerman. Addendum to "A Hands-On Look at Java Mobile Agents, A Look at Mitsubishi's Concordia", http://computer.org/internet /v1n4/kiniry.htm

[13] Concordia http://www.meitca.com/HSL/Projects/Concordia/Welcome.html

[14] Objectspace Voyager http://www.objectspace.com/

[15] IBM Aglets http://www.trl.ibm.co.jp/aglets/

[16] General Magic http://www.genmagic.com/technology/mobile_agent.html

[17] Tom Walsh, Noemi Paciorek, David Wong. Security and Reliability in Concordia, Hawaii International Conference on Software Systems, HICSS 31, January 6-9, 1998, Hawaii.

[18] Gunter Karjoth, Danny B. Lange, and Mitsuru Oshima. A Security Model for Aglets, IEEE Internet Computing, July-August, 1997, http://computer.org/internet

[19] Zahir Tari and Shun-Wu Chan. A Role-Based Access Control for Intranet Security, IEEE Internet Computing, September-October, 1997, http://computer.org/internet

[20] Larry Korba. Securing Wireless LAN Access: A Network Management Approach. IEEE Int. Symp. on Wireless Communication, Montreal, P.Q., May 22-23, 1998.

[21] UMBC Agent email list. http://www.cs.umbc.edu/agentslist/

[22] Scott Oaks, Java Security. O'Reilly & Associates.

[23] Tom Mowbray. The Essential CORBA, John Wiley & Sons, 1995.

[24] Java RMI http://java.sun.com:80/products/jdk/rmi/index.html

[25] Java Reflection Broker http://andromeda.cselt.it/users/g/grasso/free.htm

[26] Phaos SSLava http://www.phaos.com /main.htm

[27] Java Security1 http://java.sun.com/marketing/collateral/security.html

[28] B. Schneier, "Applied Cryptography." John Wiley & Sons, 1996.