Concordia

This Java mobile agent technology offers security and persistence while maintaining a record of an agent's travels.

Reuven Koblick

The scale of applications now being considered for network environments requires security and reliability previously available only in large transaction processing systems. To be viewed as a realistic development alternative for such applications, mobile agent systems have to provide highly secure and reliable environments. Mitsubishi's Concordia, introduced in 1998, is designed for such complex, secure, reliable, real-world enterprise applications.

Concordia offers features specially suited for these applications, including extensive security, reliable transmission of agents, access to legacy and native applications, remote administration, and agent debugging. It also includes several forms of interagent communication (for more detail, go to www.meitca.com/HSL/Products/Concordia).

Concordia provides a rich security model for protecting servers, agents, and Concordia itself from attack or unauthorized access. Agent protection is the process of protecting an agent's contents from tampering or inspection during transmission across a network connection or when stored on disk. Such protection ensures the privacy and integrity of the agent and the potentially sensitive information it carries.

Agent users need assurance that sensitive data carried by an agent cannot be compromised and that the agent cannot be redirected to perform unwanted actions. So, prior to transmission, Concordia encrypts an agent's bytecodes, member data, and state information through a combination of symmetric and public-key cryptography. Concordia servers also authenticate each other by exchanging digital certificates.

Concordia encrypts an agent's on-disk representation. For added reliability, it uses a persistent object store to periodically checkpoint an agent; in case of system failure and restart, the agent executes from its last checkpoint. Since the object store saves an agent and its state information, it could also be a potential security risk. So Concordia further secures this on-disk representation through encryption.

Agent authorization. Server resource protection ensures that an agent performs only the server tasks for which it is authorized and for no others. Concordia's server resource protection follows two design concepts: agent identification and resource permission. An agent's user identity uniquely represents the user who launched the agent. It consists of a user name identifying a particular individual, a group name identifying a group of individuals, and a password. Within the user identity, the password is always stored in a secure form and is never represented in clear text.

An agent roaming the network carries its own identity. At each stop in its travels, the agent's identity is verified against a list of the system's valid users. Each server includes a list of users as well as the corresponding resource-access permissions allowed for that user. Default permissions may also be configured and assigned to unknown users.

Resource permissions can be used to allow or deny fine-grain access to machine resources. For example, a resource can be constructed to allow read-access to a machine's file system. Another resource can deny such access. And a third can specify read-access only to a particular file on the machine. Concordia's resource permission mechanism is built atop the standard Java security classes, ensuring that agents use only the server resources to which they are granted access.

If a system's source code can be tampered with, no security policies can guarantee agent or server protection. Hence, class protection ensures that Concordia code is not compromised. Concordia's bytecodes are digitally signed. When an agent executes, Concordia guarantees the agent has not been altered in any way by verifying its digital signature.

Concordia's reliability features include transactional message queuing for guaranteed delivery of agents to remote systems, proxies to shield agents from the effects of system and network failures, and the object store.

Transmission across the network. The Concordia infrastructure provides reliable transmission of agents across the network by way of an underlying message-queuing subsystem. Concordia's queuing support is a natural fit for the disconnected operational mode of the mobile agent paradigm, providing a store-and-forward mechanism. Prior to transmission, an agent is stored in the local system's message queue and remains there until it has been received by the remote host. Agents can also be stored on the message queue of a local system while a remote host undergoes repair or is merely being moved to a different physical location. When the remote server comes back online, the local server then forwards the agent to the server that was offline.

The message queuing subsystem provides additional reliability by maintaining a copy of the agent to be transmitted in an on-disk queue until the recipient of this agent transmission acknowledges receipt via the two-phase commit protocol.

Proxies increase Concordia reliability by shielding agents and other objects from the effects of server and system failures. Concordia provides proxies for components supporting potentially long-lived connections. Proxies transparently attempt to reestablish connections when they are unable to communicate with their original counterparts.

Concordia includes an extensive remote administration facility that starts up, shuts down, and configures Concordia nodes, or the places where agents execute. It also manages changes in the security profile of agents, as well as servers. The administration facility also monitors the progress of agents throughout the network and maintains agent and system statistics.

Concordia's "service bridge" component gives agents controlled access to native applications (such as legacy databases). It uses the system's security features to ensure that agents do not exceed the permissions granted them by the administration component. Instances of a service bridge can be located through lookup in Concordia's directory service.

A notable difficulty in agent development is tracking the progress of an agent through the network. Concordia's agent debugger monitors, controls, and modifies an agent as it travels and executes throughout the network. The agent debugger helps track an agent at all times. C

Reuven Koblick (reuven@meitca.com) is assistant laboratory director at the Mitsubishi Electric Information Technology Center America in Waltham, Mass.

requests the agent manager transport it to the correct location.

The security manager protects the host and the mobile agents against unauthorized access. All other mobile agent system components interact with it to authenticate and authorize mobile agents. The security manager also may protect agents by encrypting them before transmission and before they are saved to persistent storage. In highly secure systems, the security manager may digitally sign agents, and mobile agent systems may authenticate each other through an exchange of certificates. The security manager also allows authorized agents to pass through firewalls.

The reliability manager ensures the robustness of the mobile agent system. In highly reliable systems, it shields agents from the effects of server and system crashes. One of its main tasks is to guarantee the persistence of state associated with agents as well as with the mobile agent system. In addition, the reliability manager may use transactional queuing, possibly with a two-phase commit, to ensure agents reach their destination, even during system crashes.

The interagent communications manager in Java and other systems facilitates communication between mobile agents dispersed throughout a network. All but the simplest of applications use multiple agents to perform their computations, and the existence of multiple associated agents mandates interagent communication. Mobile agent systems typically offer messaging or distributed events. Some systems include more sophisticated forms of interagent communication, such as Concordia, which enables affiliated agents to cooperatively solve a complex problem that can be partitioned into smaller subtasks.

The application gateway serves as a secure entry point through which agents can interact with application servers (such as legacy databases). Agents may use the JNDI-based directory manager to identify the location of an application server and then migrate to the host on which the server is located. An arriving agent accesses resident servers through this gateway. The security manager has to authorize the agent's use of the gateway and the application server.

Although this generic architecture is sufficient for most application domains, certain extensions to it and improvements in basic distributed computing technology would make mobile agents more efficient and practical for e-commerce applications. The current generation of agent frameworks implements abstractions supporting