MobileAgents

(see also Tony White's thesis and Danny Lange's slides)

(see also powerpoint version of this lecture [1])

An Introduction to Mobile Agents

A Mobile Agent is an independent program which executes on behals of the a user which moves in the network to perform its function.

- They are autonomous.
- They have identity.
- They have ability to travel in a network.

Justification

- Reduction of network load - Mobile agents consume fewer network resources since they move the computation to the data rather than the data to the computation.
- Overcome network latency - Mobile agents do not require a continuous connection between machines.
- Asynchronous interaction. - Mobile agents can replace asynchronous communications, i.e. mobile documents.
- Convenient Programming Paradigm - Mobile agents hide the communication channels but not the location of the computation and facilitate deployment of distributed applications.

Types of Applications

- Dynamic load balancing.
- Dynamic service deployment.
- Intermittently connected systems.

Background

- Process Migration.
- Agent Systems.

Topics
Mobile Agents

- Mobile Agent Concepts (MobileAgentConcepts)
- Mobile Agent Components (MobileAgentComponents)
- Mobile Agent Platforms (MobileAgentPlatforms)
- Mobile Agents R&D Issues (MobileAgentResearch)

(last edited November 26, 2002)
Find Page by browsing or searching
MobileAgentConcepts

Mobility

- Strong Mobility - Code and data is transferred to host, i.e. control is transferred to host. Associated strongly with scripting capability rather than simply object transfers.
- Weak Mobility - Only data is transferred. Associated with the transfer of Java code to a virtual machine. Primarily does not allow for mid-execution code transfer.
- Proxy server - Maintains state of the system by having a repository for all files and therefore a 'system state'.
- No proxy server - There exists no universal location for files.

Languages

- Java class libraries - Take advantage of Java's universal virtual machine (Aglet, concordia, Voyager, Grasshopper, D'Agents).
- Scripting Languages - Take advantage of interpreted languages (D'Agents, ARA, Tacoma).

Security

- It is not clear that the security problems are as an important issue as researchers tend to claim. There is a requirement for basic security that in some circumstances can be controlled through the design of systems in controlled company intranets.
- There are 3 common types of threats:
  - An agent can be malicious to the host.
  - Host can be malicious to the agent.
  - Agent can cause havoc to the network.
- Two approaches for security:
  - Authentication using X509 certificates.
  - Encryption using SSL.

Naming Scheme

Name addressing for an agent is an important concept. The naming scheme has to keep track of <B, "TR", 17>:

- B - A machine name.
- TR - An agent's symbolic name.
- 17 - An 'ID' that the host gives to the agent.
MobileAgentComponents

Each mobile 'Agency' (not an agent) will generally be composed of the following components:

- Communication
- Management
- Persistence
- Registration
- Security
- Mobility

Communication Service

Responsible for all remote interactions among the agents. Manages the networking and transport layer of communications. Ideally this should support CORBA IIOP, Java RMI, or plain socket communications. It should support asynchronous, synchronous, and multicast communication.

Management Service

Allows the management and monitoring of the agents. A user should be able to deploy, start or stop execution, and monitor the state of an agent. Ideally this could be using SNMP or CMIP but realistically no few if no agent platforms support this.

Persistence Service

A storage of agent state and places in the agent system. Not only should the agents be recoverable but the 'Agency' as well. A must for mobile agents to support intermittent connections.

Registration Service

Agents must be able to register with the 'Agency'. Beyond these requirements there is also a need for a region wide registry to maintain information about other 'Agencies'.

Security Service

The 'Agency' must support several models of security. There are 2 security issues to be concerned about: External security and Internal security.

- External security - Deals with authentication and protection of the information between agents and agencies.
- Internal security - Deals with intrusion into the agency by un-authorized agents.

Mobility Service

Handles the mechanism for mobility of an agent. Current systems use code serialization for this. Certain code is not serializable (i.e. JESS uses java class HashTableEnumerator that is not serializable).
MobileAgentPlatforms

There are a great number of mobile/static agent platforms. The more renowned systems are:

- Aglet (IBM Research)[1]
- D'Agents (Dartmouth College) [2]
- Concordia (Mitsubishi Electric ITA) [3]
- Voyager (ObjectSpace?) [4]
- Grasshopper (IKV++) [5]
- Agents for Remote Access ARA (Un. of Kaiserslautern) [6]

Aglet

Resembles an applet model in Java. Relatively simple and programmer overrides pre-defined methods to add functionality. Java code runs in an 'Aglet-enabled' host. Uses a message class to encapsulate message exchange and a proxy object to relay messages. Group oriented messaging is not available. Uses an event-driven programming model (dispatch, onDispatching, onArrival, ...).

D'Agents

Appears to be a new 'Agent Tcl' from Dartmouth College. Advantages are that it is primarily a program that can be written in any language and mobility is supported by a common service package implemented as a server. All functionalities are available in the server, i.e. code is rather small but server is ever powerfull. can support multiple languages, Tcl, Script, Java.

Concordia

Also a Java-based system where the agency supports a number of services that manage mobility, security, persistence, communication, administration, and resources. Programmed using an itinerary concept that encapsulates the destination address and method to execute when arriving there. When a method completes the agent moves to the next destination and calls the appropriate method. Also contains a service specification interface known as a 'service builder'.

Voyager

Base on Java as a programming language. It built on top of CORBA and has an agent-enhanced object request broker. Takes advantage of the Java’s reflection mechanism. Has 'Agent' class that can be sub-classed. Primarily a platform for object mobility. Contains a persistent store database along with a federated directory service and group communication (multicast) facility.

GrassHopper?

A very comprehensive platform focussed on object mobility. It has tried to conform to the OMG MASIF specifications and recently to the FIPA ACL proposed standards. Implemented in Java and supports several transport protocols like CORBA IIOP, MAF IIOP, TCP/IP with SSL, and RMI with SSL. Has comprehensive support for security models based on object authentication.
ARA

The Ara core can run agents concurrently using a fast thread package, with each interpreted agent running in a separate address space. Agents are scheduled under a time slicing policy. All core services work uniformly for all agents without regard to their implementation languages. The core offers a concept of agent interaction whereby client agents can meet local server agents at "service points" under symbolic names and synchronously exchange arbitrary requests and replies. Mobile agents can migrate to remote nodes at any point in their execution and continue from the exact state they left in, even between heterogeneous machine architectures. Mobile agents are transmitted using TCP (one data packet sent, one acknowledgement received). Supports Tcl and C/C++ by means of precompilation to MACE, an interpretable byte code.
What are the current R&D Directions in Mobile Agents?

- Performance
- Choosing when to move
- Choosing where to move
- Scaling to small platforms

**Performance**

Tests can easily demonstrate that there can be a reduction of up to 50% in the time it takes for a multi-agent system to resolve a problem using the concept of agent meeting places. In a wireless network this savings can be up to 75%.

This savings comes primarily from the overhead in TCP/IP communications and network performance.

**Choosing when to move**

An agent should jump when:

- the local environment has become suboptimal;
- the local environment will cease to exist.

An agent will jump to a new machine, M, if:

- M has unique resources required by the agent;
- M has the best environment for the agent’s next task(s).

In a dynamic environment, this requires "planning" or "control". Solving some sort of optimization problem. Planning, in turn, requires knowledge of "state."

- NOT agents for planning BUT planning for agents…..

Problem: Formulate the decision to jump and the decision where to jump as an optimization problem. This is generically NP-Hard because it involves TSP type problems, uncertainty about future environments, decisions about whether to clone or not, etc.
Where to go?

Depends on:

- locations of candidate services;
- network latencies;
- network bandwidths;
- machine loads.

This requires an infrastructure for supplying this type of information.
AgentFrameworks

- BDI/PRS
- Layered (see White, see Kendall)
- Subsumption (Brooks, see White)
- Touring Machines, Interrap (see White again)
- Kendall's layered pattern (see thesis on policies)
- demo with Madkit

list of existing frameworks:

- PRS, dMARS, Jack [8], Bee-gents [9]
- AGENT0 (Shoham) see BdiArchitecture
- DESIRE (Treur)
- FIPA compliant:
  - Nortel's framework [1]
  - Zeus (BT) [4]
- Hive (MIT) [2]
- BOND: http://bond.cs.purdue.edu/
- Madkit [5]
- Jess - JessAndAgents
- Abe [7], JKqml (IBM)
- Magenta: http://magenta.sf.net

more refs: [3]

back to SoftwareAgentCourse

(currently edited by 65.39.186.83?)
Find Page by browsing or searching
Mobile Agents

Babak Esfandiari
Types of Applications

- Dynamic load balancing.
- Dynamic service deployment.
- Intermittently connected systems.
Types of Mobility

• **Strong Mobility**: an agent can interrupt and resume execution elsewhere at any time
  – Need low level info such as stack state

• **Weak Mobility**: an agent needs to finish task before moving
What Language for Mobility?

• Java class libraries - Take advantage of Java's universal virtual machine (Aglet, Concordia, Voyager, Grasshopper)

• Scripting Languages - Take advantage of interpreted languages (D'Agents/AgentTCL, ARA, Tacoma).
Security Issues

• There are 3 common types of threats:
  – An agent can be malicious to the host.
  – Host can be malicious to the agent.
  – Agent can cause havoc to the network.

• Two approaches for security:
  – Authentication using X509 certificates.
  – Encryption using SSL.
Platform Services

- Communication
- Management
- Persistence
- Registration
- Security
- Mobility
Platforms: Aglet

- Resembles an applet model in Java. Relatively simple and programmer overrides pre-defined methods to add functionality.
- Java code runs in an 'Aglet-enabled' host.
- Uses a message class to encapsulate message exchange and a proxy object to relay messages. Group oriented messaging is not available.
- Uses an event-driven programming model (dispatch, onDispatching, onArrival, …)
Platforms: Voyager

- Based on Java, built on top of CORBA and has an agent-enhanced object request broker.
- Takes advantage of the Java's reflection mechanism. Has 'Agent' class that can be subclassed.
- Primarily a platform for object mobility.
- Contains a persistent store database along with a federated directory service and group communication (multicast)facility.
Platforms: Grasshopper

- A very comprehensive platform focused on object mobility. It has tried to conform to the OMG MASIF specifications and recently to the FIPA ACL proposed standards.
- Implemented in Java and supports several transport protocols like CORBA IIOP, MAF IIOP, TCP/IP with SSL, and RMI with SSL.
- Has comprehensive support for security models based on object authentication.
Research Challenges

• Performance
  – Use agent meeting places
• Choosing when to move
• Choosing where to move
Choosing When to Move

- An agent should jump when:
  - the local environment has become suboptimal
  - the local environment will cease to exist!

- An agent will jump to a new machine, M, if:
  - M has unique resources required by the agent;
  - M has the best environment for the agent’s next task(s).

- In a dynamic environment, this requires "planning" or "control". Solving some sort of optimization problem. Planning, in turn, requires knowledge of "state."
  - NOT agents for planning BUT planning for agents.....
Choosing Where to Move

• Depends on:
  – locations of candidate services;
  – network latencies;
  – network bandwidths;
  – machine loads.

• This requires an infrastructure for supplying this type of information.