

DaAgent: A Dependable Mobile Agent System

Shivakant Mishra, Yanjun Huang and Harshavardhan Kuntur
Department of Computer Science
University of Wyoming, P.O. Box 3682
Laramie, WY 82071-3682, USA.

A mobile agent is an active entity that performs a computation distributed over several nodes in a distributed system. While performing a computation, an agent moves independently and asynchronously from one node to another. It differs from a traditional operating system process in two important characteristics: (1) *Location Awareness*: an agent *knows* where it is executing in a distributed system at any point in time, and (2) *Communication Awareness*: an agent is aware of the communication network and makes an informed move from one node to another during execution.

An agent-based computing model is a very attractive model for computing over the Internet. In this work, we address one key problem associated with deployment of agent-based computing model over the Internet: computing over the Internet is not dependable. Nodes connected via the Internet constantly fail and recover, and the communication links go down at any time. Furthermore, information transferred over the Internet is insecure and the security of an agent or the node on which an agent runs is not guaranteed.

We are currently implementing an agent-based computing system called DaAgent. This system has been explicitly designed to experiment with providing dependability support to an agent-based computing system. DaAgent is being implemented in Java and allows users to construct dependable Java agents.

1 Agent Structure

Each agent in the DaAgent system has a node called *agentHome* associated with it. An agent is uniquely identified by its *agentId* (a pair of integers) assigned by its *agentHome*. The *agentHome* of an agent also maintains some control information about its agent and keeps track of the movement of agent in the Internet.

An agent consists of two parts: *agentHead* and *agentCode*. The *agentHead* includes some control information such as *agentId*, *agentHome id*, maximum time to live, and the *agentCode* is the actual code executed by the agent. There are three special func-

tions that a DaAgent programmer needs to be aware of: *agentInit()*, *agentGo()*, and *agentTerminate()*. The *agentInit()* function invoked at the beginning allocates appropriate resources to launch an agent. Whenever an agent intends to migrate to another node, it invokes the function *agentGo()* that has a node address as a parameter. When an agent decides to terminate, it invokes the *agentTerminate()* function.

2 Architecture

The main focus in the design of DaAgent has been on modularization to facilitate experimentation. It is possible to redesign and reimplement a single module within DaAgent without affecting any other module.

The DaAgent system consists of five modules (See Figure 1): DaAgent server, agent watchdog, agent consultant, agent, and communication module. A DaAgent server runs on every node in the network which might be visited by an agent or from which an agent may be launched. A DaAgent server running on a node services requests from local clients on that node to launch their agents. In addition, it also services an agent that migrates from some other node to that node. When it receives a request from a local client to launch an agent, or when an agent arrives from some other node, the DaAgent server starts a new process called the agent watchdog. This agent watchdog oversees the functioning of this agent on its node and its migration to the next node. There is one agent watchdog for every agent running locally on a node.

An agent watchdog first consults a local agent consultant to determine if the new agent meets all the admissibility criteria, and if so, starts a new process to run the agent locally. After starting the agent, its agent watchdog simply waits for the agent to terminate. When an agent decides to migrate to another node, it creates two files. In one file, it writes the address of the next node it wants to migrate to, and in the other file, it writes the code needed to start this agent on the next node along with a state. After creating these two

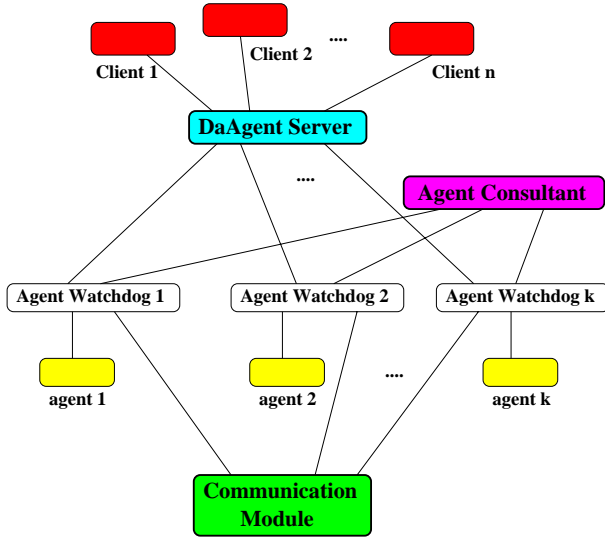


Figure 1: Architecture of DaAgent.

files, the agent terminates. After an agent terminates, its agent watchdog transfers its code and state to the next node the agent needs to migrate to. After this the agent watchdog also terminates.

There is one agent consultant on every node that provides DaAgent services in the network. An agent consultant on a node determines if a locally launched agent or an agent migrated from some other node can be run locally. Finally, the communication module provides the communication support for transferring an agent from one node to another.

3 Dependability

3.1 Fault Tolerance

There are three important problems caused by communication or node failures in the Internet: (1) A communication failure may prevent the transfer of an agent from one node to the next. (2) Failure of a node to which an agent intends to visit will force an agent to terminate prematurely. (3) Failure of a node will result in a premature termination of all agents running on that node.

Agent watchdog and agentHome of an agent are used to detect communication and node failures and ensure that agents do not terminate prematurely: (1) An agent watchdog is responsible for (i) ensuring a successful transfer of its agent to the next node that the agent intends to visit (via positive acknowledgements), (ii) continuously monitoring the execution of its agent on this next node (via periodic acknowledgements), and

(iii) recovering the agent by migrating it to an alternate node if the node to which the agent has moved fails. (2) agentHome of an agent also monitors the execution of its agent by requiring the agent to send periodic acknowledgements. The time period of this periodic acknowledgement is set at the time the agent is launched.

3.2 Security

Since an agent is written independently by a client, migrates over the network, and runs on remote nodes, there are three important security problems that may arise: (1) A malicious agent may jeopardize the integrity of a remote node on which it runs. (2) A malicious node may jeopardize the integrity of an agent that runs on it. (3) An agent may be interfered with during migration over the network.

Agent consultant, communication module, and agentHome of an agent are used to deal with these potential security problems: (1) Agent consultant at a node runs an admissibility protocol to determine if a newly arrived agent is authorized to run locally. The admissibility criteria can include techniques such as digital signatures for authentication, trusted node relationships, and authenticity certifications. (2) The communication module uses appropriate cryptographic algorithms to ensure that an agent cannot be interfered with during transmission, and that if an agent has been interfered, the agent consultant on a node can detect this interference. (3) agentHead of an agent carries a list of nodes that the agent should avoid visiting and another list of nodes that the agent has visited. The first list can be dynamically updated by the agentHome of the agent. The agent watchdog ensures that the agent never visits a node that is included in this list. The second list is used by agentHome to determine if a portion of the network is safe for future agents to visit.

3.3 Runaway Agents

A runaway agent is an agent that continues to run visiting various nodes in the network even when it is no longer needed. In an asynchronous distributed system such as the Internet, runaway agents can be created due to node or communication failures. Since runaway agents consume network bandwidth and processing power, it is important that such agents be caught and terminated. In DaAgent, DaAgent server and agentHome are used to detect and terminate runaway agents. Each DaAgent server requires each locally running agent to be acknowledged periodically from its agentHome. A runaway agent is not acknowledged, and is simply terminated by the DaAgent server.