Applying Broadcasting/Multicasting/Secured Communication to agentMom in Multi-Agent Systems

Project Overview

Version 1.1

This document is submitted in partial fulfillment of the requirements for the degree MSE.

Chairoj Mekprasertvit
CIS 895 – MSE Project
Kansas State University
Spring 2003

Table of Contents

1. Background

1.1 Motivation

Communication is one of the critical parts in multi-agent systems because it enables the agents in multi-agent systems to exchange information and cooperate with each other. This raises the question of what communication technique to be used in multi-agent systems. An earlier implementation of agentMom developed by Dr. Scott A. DeLoach, allows only one-to-one communication using the TCP/IP protocol. However, in the situation where a single message is destined for many recipients, multiple copies of the same message have to be sent by using TCP/IP protocol. This is not very efficient when there are a large number of agents in the system. Broadcast/Multicast communication on the other hand prevents network overloading by generating a single message destined for multiple recipients. This technique also reduces the amount of conversations that an agent has to handle. Depending on the situations, an agent may choose to unicast over broadcast/multicast messages. For example, unicast message is more appropriate when there are very few recipients. Thus, by integrating broadcast/multicast communication capability, agentMom can provide a more flexible way for an agent to communicate with other agents

1.2 Multi-Agent Systems

Multi-Agent Systems are one of the most recent contributions in the field of software engineering in building distributed intelligent systems. It is described as a further abstraction of the object-oriented paradigm where agents are a specialization of objects [1]. Agents are similar to objects; however, they have traits such as autonomy, cooperation, perception, and pro-activeness that imply characteristics that objects generally do not have. Basically, there are two differences:

1. Objects are passive. They react to external stimuli, but do not exhibit goal directed behavior.
2. Agents typically use a common messaging language between all agents whereas object messages are usually class dependent [3].

Therefore, an object is a logical combination of data structures and its corresponding methods while agents additionally support structures for representing mental state components such as attitudes, beliefs and goals.

1.3 Example of software agents

There are wide ranges of application domains that are making use of agent-oriented systems engineering. Software agents are being developed for fields as varied as entertainment, electronic commerce, user assistance, and information systems.

For example,

1. The animated paperclip agent in Microsoft Office
2. Computer viruses (destructive agents)
3. Artificial players or actors in computer games and simulations (e.g. Quake)
4. Trading and negotiation agents (e.g. the auction agent at Ebay)
5. Web spiders (collecting data to build indexes to used by a search engine, i.e. Google) [5]

1.4 agentMom

Agents in Multi-Agent systems environment have to communicate with other agents to be able to cooperate and achieve the assigned goals. However, it is not an easy task for developers to manage all communications.

agentMom is a communication framework for multi-agent systems implemented in Java. It provides a framework for building agents, conversations between agents and messages passed in the conversations. Currently in agentMom version 1.2, it consists of four important classes: Agent, Conversation, Message and MessageHandler. The Agent class is an abstract class that defines the minimum set of requirements for an agent to use agentMom. The Conversation class is an abstract class that is basically used for sending and receiving message using the TCP/IP protocol. The Message class defines the field used in the message passing between agents such as host, port, sender and receiver. The The MessageHandler class is used to start a socket on the indicated port and wait for connection from another agent. Basically, it monitors the local port for establishing connection.

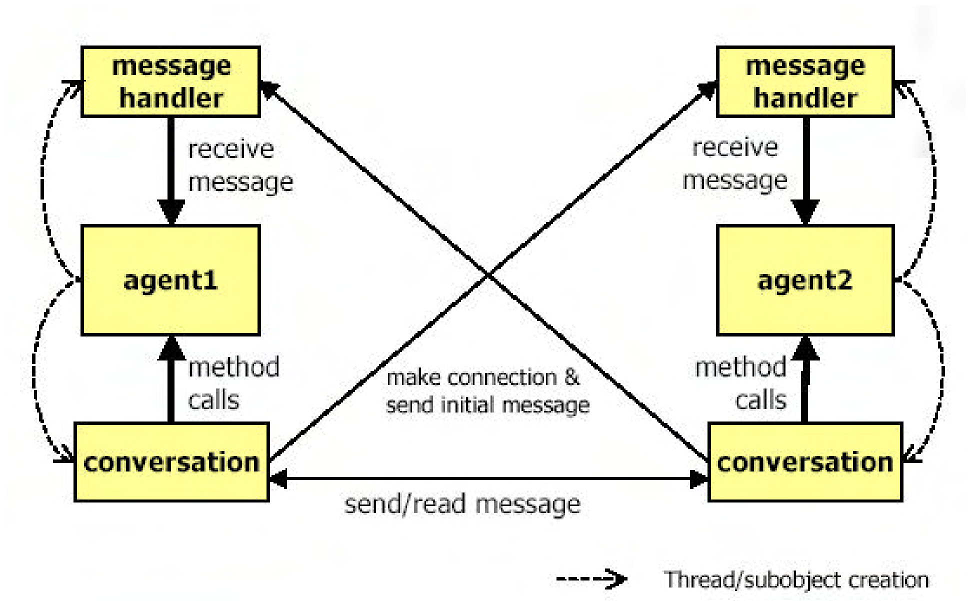
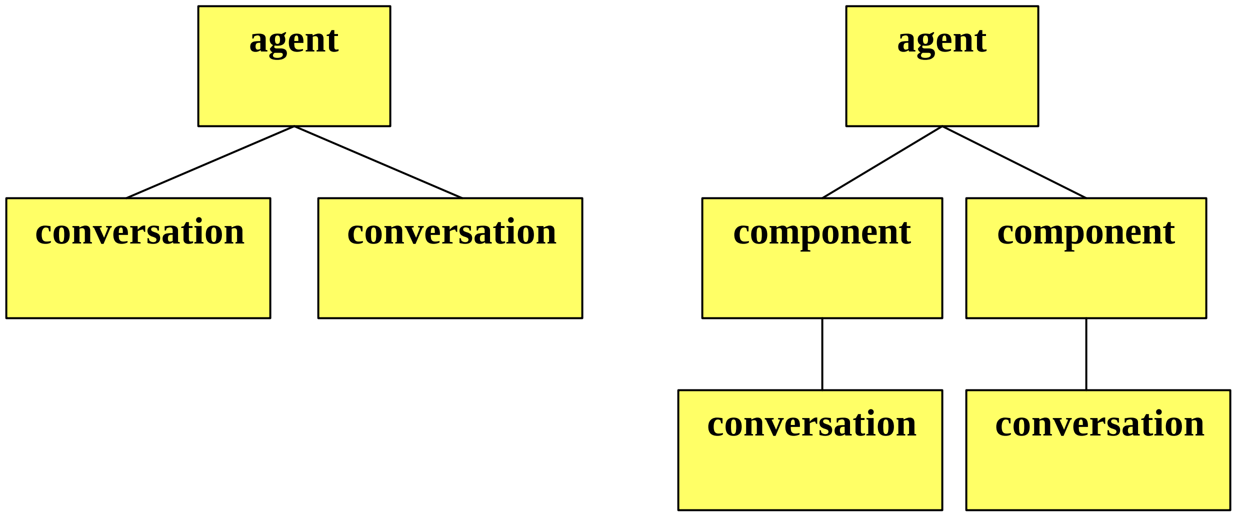


Figure 2. agentMom [4]

An overview of how agentMom works is shown in Figure 2. When an agent wants to start a conversation with other agents, that agent starts the MessageHandler that opens the indicated port, and then waits for the connection to establish with the other agent's MessageHandler. An agent can start the conversation by establishing socket connection with another agent's MessageHandler. When connection has been established and the initial message has been validated, they can then start sending and receiving messages. For more information and source code, please refer to [4].



a. Agent directly controls conversations b. Component controls conversations

Figure 3. agentMom's architectures

Furthermore, there are two architectures that can be applied to agentMom. The first architecture is shown in Figure 3a. In the first architecture, agent directly controls the conversations. This architecture is very straightforward since conversations belong to agent. In the second architecture as shown in Figure 3b, an agent consists of one or more components, and the conversations belong to components, not directly to agents. Also, an agent can have multiple components and components can have multiple conversation. The difference from the first architecture is that component is responsible for making conversation with other agents. In the first architecture, agents are directly responsible for controlling the conversation. Having components separately from agent allows developers to map the agent role's tasks to the component. From now, we will refer to the first architecture as agent-based architecture and the second architecture as component-based architecture.

In this project, we will consider these two architectures in applying multicast, broadcast and security into agentMom.

2. Project Overview

2.1 Terms and Definitions

Unicast refers to one-to-one communication in such a way that a packet originates from a single Internet host, and it is destined to a unique location of another Internet host.

Multicast refers to one-to-many communication in such a way that a packet originates from a single Internet host, and it is destined to multiple receivers within the same multicast address.

Broadcast refers to one-to-many communication in such a way that a packet originates from a single Internet host, and it is destined to all receivers within the same local network.

Organization refers to a set of agents.

Institution refers to a set of the basic element required to build a particular type of organization consisting of goals, roles, rules and protocols.

Reorganization refers to a situation where the previous organization structure is not efficient to succeed the mission.
Group refers to a set of agents who agree to use the same multicast address to subscribe group message.
Time-To-Live (TTL) refers to the number of hops that multicast message is allowed to remain in the network before it is discarded by the router.

2.2 Overview

This MSE project is part of the research project "Autonomous Reorganization of Cooperative Robotic Teams for Robust Performance" supervised by Dr. Scott DeLoach. The main focus of this research is to provide autonomous cooperative robotic teams with enough knowledge of their team goals and organizational structure to allow them to autonomously organize and reorganize to achieve their team goal in the face of changing environmental conditions and individual team member failures [2].

The main focus of this MSE project is on extending agentMom capability to manage unicast, multicast and broadcast communication and to provide secured communication such as message encryption and decryption. Currently, agentMom1.2 only supports unicast communication without multicast broadcast and security. There are many advantages in using broadcast/multicast communication in multi-agent systems environment. First of all, when there are many agents in the system and the use of network bandwidth is critical, then sending multiple copies of the same message to each receiving agent may not desirable. Broadcast or multicast communication can save network bandwidth by sending a single message destined for multiple receiving agents. Secondly, when an agent has to send the same message to hundreds of agent, then this agent may not be able to do anything else, but sending and receiving messages. Broadcast and multicast techniques can reduce agent's workload. Furthermore, in the situation where the sender does not know the address of all agents in the system, sender can choose to multicast or broadcast the message to find agents on service. In the bidding/marketing-based technique, agents may broadcast or multicast messages to other agents for some services. The recipients of these messages evaluate those requests, and then submit bids with directed message to the originating agents. The originating agents use this information to choose the appropriate agent to do the jobs, and then send directed message back to the desired agents. Lastly, there are many situations that need to divide agents into different groups such as search group and rescue group. Agents may want to receive messages only from the group they belong to. Multicast communication supports this implementation.

In a multi-agent system, security is also an important issue when message is sent over a public network such as Internet. It is undesirable if someone who is not specified to receive message can see the content of it. Message encryption can prevent this situation.

Therefore, integrating broadcast/multicast communication and security features to agentMom can provide a more flexible way for communication in multi-agent systems.

2.3 Goal

Integrate multicasting, broadcasting and secured communication capability in agentMom in order to provide more efficient and effective way for communication in multi-agent environment.

2.4 Purpose

1. Enable agents to broadcast a message to all the agents within the same local network.
2. Enable agents to multicast a message to all the agents within the same multicast address.
3. Allow agents to choose among unicast, multicast and broadcast communication.
4. Allow agents to join and leave multicast group
5. Reduce network bandwidth from multiple copies of the same message by using multicast and broadcast communication.
6. Reduce agent's workload by reducing the number of sending and receiving messages.
7. Provide message encryption and decryption techniques.

2.5 Feature

1. Support unicast, multicast and broadcast communication.
2. Allow agent to choose which communication method to be used (unicast/multicast/broadcast) to fit the needs.
3. Allow agent to join and leave multicast group.
4. Allow agent to choose to encrypt or not to encrypt message.

2.6 Risk

1. Reliable message delivery – multicast/broadcast packets are delivered with best effort. Thus, a packet may be delivered to all specified agents or none.
2. Security – we provide some basic mechanisms for security such as message encryption. However, there is no guarantee that the others cannot decrypt the encrypted messages.

2.7 Direction

1. Reliable message delivery – scalable reliable message delivery is an important issue in multicast and broadcast communication. It is a hot research area in communication network and there is no single solution to this problem. Thus, this can be further in the future work.
2. FIPA Agent Communication Language (ACL) – this project can be further to conform to FIPA ACL specification, including FIPA ACL messages represented in XML.

2.8 Environment

1. This software package will be compiled using Java 1.4.2.
2. Rational Rose 2000 will be used for creating various object diagrams..
3. Eclipse IDE 2.1 will be used for coding the software package.
4. This software package will be tested under Microsoft Windows XP/2000, Linux Debian and Unix Solaris 9.
5. USE 2.0 will be used for modeling the formal specifications, using UML/OCL methodology.

3 References

[1] DeLoach, S. A. Wood, M. F., and Sparkman, C. H.: Multiagent Systems Engineering, *The International Journal of Software Engineering and Knowledge Engineering* 11(3):231-258.

[2] DeLoach, S. A., Matson, E. T., and Li, Y.: Applying Agent Oriented Software Engineering to Cooperative Robotics, *Proceedings of the 15th International FLAIRS Conference (FLAIRS 2002)* pp. 391 – 396 Pensacola, Florida. May 16-18, 2002.

[3] DeLoach, S. A.: Multiagent Systems Engineering. http://www.cis.ksu.edu/~sdeloach/ai/mase.htm.

[4] DeLoach, S. A.: agentMom User's Manual, *Air Force Institute of Technology*, July 2000.

[5] Tveit, A.: A survey of agent-oriented software engineering, *NTNU Computer Science Graduate Student Conference,* Norwegian University of Science and Technology, Norway, May 2001.