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

Formal Requirement Specification

Version 1.0

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

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

Formal Requirement Specification

1 Introduction

1.1 Purpose

The purpose of this document is to provide the formal requirement specification of the project "Applying Broadcast/Multicast/Secured communication to agentMom in Multi-Agent Systems". This specification uses the UML/OCL methodology as specified in the UML specification version 1.5. The Object Constraint Language (OCL) is a formal language used to express constraint and specify invariant for the system being model. It provides a precise and unambiguous specification of the system.

1.2 Scope

In the specification, we specify the pre and post condition of the interest properties to ensure that these properties are hold in our system model. These properties are:

1.) Unicast conversation
- 1.1) Only the specified address receives the unicast message.
- 1.2) Sent message is the same as received message

2.) Multicast conversation
- 2.1) Only the specified group receives the multicast message for that group
- 2.2) Sent message is the same as received message

3.) Broadcast conversation
- 3.1) Only the conversations holding the same broadcast address receive the broadcast message.
- 3.2) Sent message is the same as received message

4.) Secured unicast conversation
- 4.1) Only the specified address receives the unicast message
- 4.2) Sent message is the same as received message

The properties are based on the driving requirement as stated in the Software Requirement Specification version 1.0. Furthermore, we use the UML- based Specification Environment (USE) tool to check the type and syntax to ensure correctness of the specification. Please refer to Appendix A for a full specification of the model.

1.3 References

- Software Requirement Specification, Version 1.0, Kansas State University, 2003, (http://www.cis.ksu.edu/~cme6556/software_requirements_specification_1.0.pdf)
- OMG Unified Modeling Language Specification, Version 1.5, (http://www.omg.org)
- Architecture Design, Version 1.0, Kansas State University, 2003, (http://www.cis.ksu.edu/~cme6556/architecture_design.pdf)
- USE manual, University of Bremen (http:// www.db.informatik.uni-bremen.de/project/USE)

2 Formal Requirement Specification Descriptions

This section explains the unicast conversation and multicast conversation specification in detail. Because unicast conversation and secured unicast conversation specifications are almost

identical, only the unicast conversation specification is convered. Also, it is the same as multicast conversation and broadcast conversation specifications.

2.1 Unicast conversation

The unicast conversation is named “Conversation”. The attributes of this class are: m, localhost and connectionHost. The attribute m is a Message type, and it is used for storing a message sent to another agent. The attribute Localhost is a String type of Internet address of the agent. The attribute connectionHost is a String type of Internet address of the connecting agent (receiver agent). Furthermore, the class and association related to unicast conversation is shown below:

```
class Conversation
attributes
m: Message;
Localhost: String;
connectionHost: String;
connectionPort: Integer;
operations
sendMessage(m: Message)
receiveMessage(): Message
end

association Agent-Conversation between
      Agent[1] role agent
      Conversation[0..*] role unicastConversation
End

association ConstructUnicast between
      Conversation[0..1] role createdByUnicast;
      Message[0..1] role createdMessage;
end

association ReceiveUnicast between
      Conversation[0..1] role receivedByUnicast;
      Message[0..1] role receivedMessage;
end
```

Finally, the pre and post condition related to this class is described below:

2.1.1 Only the specified address receives the unicast message.

```
context Conversation::sendMessage(m: Message)
-- unicast conversation associates with the Message parameter
      pre cond_1: self.createdMessage = m
-- Message must be well defined before sending
      pre cond_2: m.isDefined
-- Only the destined address and port receive the message.
      post cond_3: Conversation.allInstances->
                              exists(c: Conversation|
                                    ((c.Localhost = self.connectionHost
                                    and
                                    c.agent.port = self.connectionPort)
                                    implies
```

```
                                    c.receivedMessage = m)
                                    and
                                    (c.receivedMessage = m
                                    implies
                                    (c.Localhost = self.connectionHost
                                    and
                                    c.agent.port = self.connectionPort)))
```

This part of specification defines pre and post condition of the operation sendMessage of the class Conversation. There are two pre-conditions and one post-condition. The pre-condition "cond_1" states that the Message object m must be created. The pre-condition "cond_2" states that the attributes of Message object m must be defined. Finally, the post-condition "cond_3" states that there exist a Conversation object that receives the Message object m, and the Internet address and port number of the receiver must be the same as the address that sender connects to. Therefore, only the specified address and port number receives the unicast message.

2.1.2) Received message is the same as sent message

```
-- Receive unicast pre-post condition
-- Received message is the same as sent message
context Conversation::receiveMessage(): Message
-- New received message is created
      post cond_1: self.receivedMessage.oclIsNew = true
-- New created received message is the same as sent Message
      post cond_2: Conversation.allInstances->
                              exists(c: Conversation|
                                    ((c.connectionHost = self.Localhost
                                    and
                                    c.connectionPort = self.agent.port)
                                    implies
                                    c.createdMessage = self.receivedMessage)
                                    and
                                    (c.createdMessage = self.receivedMessage
                                    implies
                                    (c.connectionHost = self.Localhost
                                    and
                                    c.connectionPort = self.agent.port)))
-- Result of receiveMessage()
      post cond_3: result = self.receivedMessage
```

This part of specification defines pre and post condition of the operation recevieMessage of the class Conversation. There are three post-conditions. The post-condition "cond_1" states that the received Message object is created during the operation receiveMessage (Message is received from another agent). The post-condition "cond_2" states that the new received Message object must be the same as the Message object that is sent by the sender. The post-condition "cond_3" specifies the return result of this operation. In this case, it is the received message is returned. Therefore, the sent message is the same as received message.

2.2 Multicast conversation

The multicast conversation is named "MulticastConversation". The attributes of this class are: multicastPort, m, join and multicastAddress. The attribute m is a Message type, and it is used for storing a message sent to another agent. The attribute multicastAddress is a String type of multicast address of the group that agent subscribes to. The attribute multicastPort is a Integer type

of the port that multicast listening. Furthermore, the association related to multicast conversation is shown below:

```
class MulticastConversation
attributes
multicastPort: Integer;
m: Message;
join: Boolean;
multicastAddress: String;
operations
sendMessage(m: Message)
sendJoin()
sendLeave()
receiveMessage(): Message
end

association Agent-MulticastConversation between
      Agent[1] role agent
      MulticastConversation[0..*] role multicastConversation
end

association ConstructMulticast between
      MulticastConversation[0..1] role createdByMulticast;
      Message[0..1] role createdMessage;
end

association ReceiveMulticast between
      MulticastConversation[0..1] role receivedByMulticast;
      Message[0..1] role receivedMessage;
end
```

2.2.1) Only the specified group receives the multicast message for that group

```
-- Send multicast pre-post condition
context MulticastConversation::sendMessage(m: Message)
-- Multicast conversation associates with the Message parameter
      pre cond_1: self.createdMessage = m
-- Message must be well defined before sending
      pre cond_2: m.isDefined
-- Need to subscribe to the multicast group first
      pre cond_3: self.join = true
-- All conversations that have the same multicast address and port receives the
-- message, including itself.
      post cond_4: MulticastConversation.allInstances->
                              forAll(c: MulticastConversation|
                                    ((c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)
                                    implies
                                    c.receivedMessage = m)
                                    and
                                    (c.receivedMessage = m
                                    implies
                                    (c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)))
```

This part of specification defines pre and post condition of the operation sendMessage of the class MulticastConversation. There are three pre-conditions and one post-condition. The pre-condition “cond_1” states that the Message object m must be associated with the sender. The pre-condition “cond_2” states that the attributes of Message object m must be defined. The pre-condition “cond_3” states that the join attribute of the agent must be true (agent must join in the group first). Finally, the post-condition “cond_4” states that all MulticastConversation objects that subscribes to the same multicast address and listening to the same port as the sender receive the Message object m. Therefore, all subscribers receive multicast message.

2.2.2) Multicast sent message is the same as received message

```
-- Receive multicast pre-post condition
context MulticastConversation::receiveMessage(): Message
      pre cond_1: self.join = true
-- New received message is created
      post cond_2: self.receivedMessage.oclIsNew = true
-- New created received message is the same as sent
      post cond_3: MulticastConversation.allInstances->
                              exists(c: MulticastConversation|
                                    ((c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)
                                    implies
                                    c.createdMessage = self.receivedMessage)
                                    and
                                    (c.createdMessage = self.receivedMessage
                                    implies
                                    (c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)))
-- Result of receiveMessage()
      post cond_4: result = self.receivedMessage
```

This part of specification defines pre and post condition of the operation receiveMessage of the class MulticastConversation. There are one pre-condition and three post-conditions. The pre-condition “cond_1” states that the join attribute must be true. The post-condition “cond_2” states that the received Message object is created during the operation receiveMessage (Message is received from another agent). The post-condition “cond_3” states that the received Message object must be the same as the sent Message object. That is there exists a sending conversation that subscribe to same group and listen to the same port as the receiving conversation, then sent message is the same as received message. The post-condition “cond_4” specifies the return result of this operation. In this case, it is the received message is returned. Therefore, the sent message is the same as received message.

Appendix A

```
AgentMom_ocl.use
-- Description: Formal Requirement Specification based on agentMom's
-- Architecture design using UML/OCL methodology.
-- We want to formalize to show that our model holds the following properties by
-- defining the pre and post conditions:
-- 1.) Unicast conversation
```

```
-- 1.1) Only the specified address receives the unicast message
-- 1.2) Sent message is the same as received message
-- 2.) Multicast conversation
-- 2.1) Only the specified group receives the multicast message for that group
-- 2.2) Sent message is the same as received message
-- 3.) Broadcast conversation
-- 3.1) Only the conversations holding the same broadcast address receive the
-- broadcast message
-- 3.2) Sent message is the same as received message
-- In this model we assume that the underlying physical communication is
-- reliable.
-- Project: Applying Broadcast/Multicast/Secured Communication to agentMom in
-- Multiagent Systems
-- Author: Chairoj Mekprasertvit
-- File: agentMom_ocl.use
-- Course: CIS895 MSE Project 2003
-- Project Advisor: Dr. Scott A. DeLoach
-- Department of Computing and Information Sciences
-- Kansas State University
-- version 1.1 11-23-2003
model agentMom

class MomObject
attributes
name: String;
port: Integer;

broadcast_port: Integer;
secure_unicast_port: Integer;
operations
end

class Agent < MomObject
attributes
operations
end

class Component < MomObject
attributes
operations
end

class MessageHandler
attributes
operations
end

class Message
attributes
content: String;
force: String;
host: String;
inreplyto: String;
language: String;
ontology: String;
performative: String;
port: Integer;
```

```
receiver: String;
replywith: String;
sender: String;
end

class Conversation
attributes
m: Message;
Localhost: String;
connectionHost: String;
connectionPort: Integer;
operations
sendMessage(m: Message)
receiveMessage(): Message
end

class MulticastConversation
attributes
multicastPort: Integer;
m: Message;
join: Boolean;
multicastAddress: String;
operations
sendMessage(m: Message)
sendJoin()
sendLeave()
receiveMessage(): Message
end

class BroadcastConversation
attributes
broadcastPort: Integer;
m: Message;
broadcastAddress: String;
operations
sendMessage(m: Message)
receiveMessage(): Message
end

class SecureUnicastConversation
attributes
Localhost: String;
connectionHost: String;
connectionPort: Integer;
m: Message;
operations
sendMessage(m: Message)
receiveMessage(): Message
end


-- Associations

association Agent-Conversation between
      Agent[1] role agent
      Conversation[0..*] role unicastConversation
end
```

```
association Agent-MulticastConversation between
      Agent[1] role agent
      MulticastConversation[0..*] role multicastConversation
end

association Agent-BroadcastConversation between
      Agent[1] role agent
      BroadcastConversation[0..*] role broadcastConversation
end

association Agent-SecureUnicasttConversation between
      Agent[1] role agent
      SecureUnicastConversation[0..*] role secureUnicastConversation
end

association ConstructUnicast between
      Conversation[0..1] role createdByUnicast;
      Message[0..1] role createdMessage;
end

association ReceiveUnicast between
      Conversation[0..1] role receivedByUnicast;
      Message[0..1] role receivedMessage;
end

association ConstructMulticast between
      MulticastConversation[0..1] role createdByMulticast;
      Message[0..1] role createdMessage;
end

association ReceiveMulticast between
      MulticastConversation[0..1] role receivedByMulticast;
      Message[0..1] role receivedMessage;
end

association ConstructSecureUnicast between
      SecureUnicastConversation[0..1] role createdBySecured;
      Message[0..1] role createdMessage;
end

association ReceiveSecureUnicast between
      SecureUnicastConversation[0..1] role receivedBySecured;
      Message[0..1] role receivedMessage;
end

association ConstructBroadcast between
      BroadcastConversation[0..1] role createdByBroadcast;
      Message[0..1] role createdMessage;
end

association ReceiveBroadcast between
      BroadcastConversation[0..1] role receivedByBroadcast;
      Message[0..1] role receivedMessage;
end
```

```
-- Constraints

constraints

-- Pre - Post Conditions
-- Send unicast pre-post condition
-- Only Specified agent receives message
context Conversation::sendMessage(m: Message)
-- unicast conversation associates with the Message parameter
      pre cond_1: self.createdMessage = m
-- Message must be well defined before sending
      pre cond_2: m.isDefined
-- Only the destined address and port receive the message.
      post cond_3: Conversation.allInstances->
                              exists(c: Conversation|
                                    ((c.Localhost = self.connectionHost
                                    and
                                    c.agent.port = self.connectionPort)
                                    implies
                                    c.receivedMessage = m)
                                    and
                                    (c.receivedMessage = m
                                    implies
                                    (c.Localhost = self.connectionHost
                                    and
                                    c.agent.port = self.connectionPort)))

-- Receive unicast pre-post condition
-- Received message is the same as sent message
context Conversation::receiveMessage(): Message
-- New received message is created
      post cond_1: self.receivedMessage.oclIsNew = true
-- New created received message is the same as sent Message
      post cond_2: Conversation.allInstances->
                              exists(c: Conversation|
                                    ((c.connectionHost = self.Localhost
                                    and
                                    c.connectionPort = self.agent.port)
                                    implies
                                    c.createdMessage = self.receivedMessage)
                                    and
                                    (c.createdMessage = self.receivedMessage
                                    implies
                                    (c.connectionHost = self.Localhost
                                    and
                                    c.connectionPort = self.agent.port)))
-- Result of receiveMessage()
      post cond_3: result = self.receivedMessage

-- Send secured unicast pre-post condition

context SecureUnicastConversation::sendMessage(m: Message)
-- secured unicast conversation associates with the Message parameter
      pre cond_1: self.createdMessage = m
-- Message must be well defined before sending
      pre cond_2: m.isDefined
-- Only the address that the message is destined to receives the message.
```

```
      post cond_3: SecureUnicastConversation.allInstances->
                              exists(c: SecureUnicastConversation |
                                    ((c.Localhost = self.connectionHost
                                    and
                                    c.agent.secure_unicast_port =
                                          self.connectionPort)
                                    implies
                                    c.receivedMessage = m)
                                    and
                                    (c.receivedMessage = m
                                    implies
                                    (c.Localhost = self.connectionHost
                                    and
                                    c.agent.parent.secure_unicast_port =
                                          self.connectionPort)))

-- Receive secured unicast pre-post condition
context SecureUnicastConversation::receiveMessage(): Message
-- New received message is created
      post cond_1: self.receivedMessage.oclIsNew = true
-- New created received message is the same as sent Message
      post cond_2: SecureUnicastConversation.allInstances ->
                              exists(c: SecureUnicastConversation |
                              ((c.connectionHost = self.Localhost
                                    and
                                    c.connectionPort =
                                          self.agent.secure_unicast_port)
                                    implies
                                    c.createdMessage = self.receivedMessage)
                                    and
                                    (c.createdMessage = self.receivedMessage
                                    implies
                                    (c.connectionHost = self.Localhost
                                    and
                                    c.connectionPort =
                                          self.agent.secure_unicast_port)))
-- Result of receiveMessage()
      post cond_3: result = self.receivedMessage

-- Send multicast pre-post condition
context MulticastConversation::sendMessage(m: Message)
-- Multicast conversation associates with the Message parameter
      pre cond_1: self.createdMessage = m
-- Message must be well defined before sending
      pre cond_2: m.isDefined
-- Need to subscribe to the multicast group first
      pre cond_3: self.join = true
-- All conversations that have the same multicast address and port receives the
-- message, including itself.
      post cond_4: MulticastConversation.allInstances->
                              forAll(c: MulticastConversation|
                                    ((c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)
                                    implies
                                    c.receivedMessage = m)
                                    and
```

```
                                    (c.receivedMessage = m
                                    implies
                                    (c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)))

context MulticastConversation::sendJoin()
-- Not in the group
      pre cond_1: self.join = false
-- New received message is created
      post cond_2: self.receivedMessage.oclIsNew = true
-- All conversations that have the same multicast address receives the join
-- groupmessage, including itself.
      post cond_3: MulticastConversation.allInstances->
                              forAll(c: MulticastConversation|
                                    ((c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)
                                    implies
                                    c.receivedMessage = self.receivedMessage)
                                    and
                                    (c.receivedMessage = self.receivedMessage
                                    implies
                                    (c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)))
-- Now join the group
      post cond_4: self.join = true

context MulticastConversation::sendLeave()
-- Already in the group
      pre cond_1: self.join = true
-- New received message is created
      post cond_2: self.receivedMessage.oclIsNew = true
-- All conversations that have the same multicast address receives the leave
-- groupmessage, including itself.
      post cond_3: MulticastConversation.allInstances->
                              forAll(c: MulticastConversation|
                                    ((c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)
                                    implies
                                    c.receivedMessage = self.receivedMessage)
                                    and
                                    (c.receivedMessage = self.receivedMessage
                                    implies
                                    (c.multicastAddress = self.multicastAddress
                                    and
                                    c.multicastPort = self.multicastPort)))
-- Not in the group
      post cond_4: self.join = false

-- Receive multicast pre-post condition
context MulticastConversation::receiveMessage(): Message
      pre cond_1: self.join = true
-- New received message is created
      post cond_2: self.receivedMessage.oclIsNew = true
```

```
-- New created received message is the same as sent
      post cond_3: MulticastConversation.allInstances->
                                 exists(c: MulticastConversation|
                                       ((c.multicastAddress = self.multicastAddress
                                       and
                                       c.multicastPort = self.multicastPort)
                                       implies
                                       c.createdMessage = self.receivedMessage)
                                       and
                                       (c.createdMessage = self.receivedMessage
                                       implies
                                       (c.multicastAddress = self.multicastAddress
                                       and
                                       c.multicastPort = self.multicastPort)))
-- Result of receiveMessage()
      post cond_4: result = self.receivedMessage


-- Broadcast message is received by all broadcast conversation that has the same
-- broadcast address, which is the same local network.
context BroadcastConversation::sendMessage(m: Message)
-- Broadcast conversation associates with the Message parameter
      pre cond_1: self.createdMessage= m
-- Message must be well defined before sending
      pre cond_2: m.isDefined
-- All conversations that have the same broadcast address and port receive the
-- message, including itself.
      post cond_3: BroadcastConversation.allInstances->
                                 forAll(c: BroadcastConversation|
                                       ((c.broadcastAddress = self.broadcastAddress
                                       and
                                       c.broadcastPort = self.broadcastPort)
                                       implies
                                       c.receivedMessage = m)
                                       and
                                       (c.receivedMessage = m
                                       implies
                                       (c.broadcastAddress = self.broadcastAddress
                                       and
                                       c.broadcastPort = self.broadcastPort)))

-- Received broadcast message is the same as sent message
context BroadcastConversation::receiveMessage(): Message
-- New received message is created
      post cond_1: self.receivedMessage.oclIsNew = true
-- New received message is created
      post cond_2: self.receivedMessage.oclIsNew = true
-- New created received message is the same as sent
post cond_3: MulticastConversation.allInstances->
                                 exists(c: BroadcastConversation|
                                       ((c.broadcastAddress = self.broadcastAddress
                                       and
                                       c.broadcastPort = self.broadcast Port)
                                       implies
                                       c.createdededMessage = self.receivedMessage)
                                       and
                                       (c.createdMessage = self.receivedMessage
```

```
                                    implies
                                    (c.broadcastAddress = self.broadcastAddress
                                    and
                                    c.broadcastPort = self.broadcastPort)))
-- Result of receiveMessage()
      post cond_3: result = self.receivedMessage
```