CSS497 Summer 2005

Redesigning and Enhancing the UWAgent Execution Engine

Status Report: Fri 08/05/2005

Summary

Dates: Sat 07/23/2005 – Fri 08/05/2005

This work period:

- Fixed the intermittent failure in socket communication. The problem occurred when the socket reader did not receive the expected number of bytes, but program execution continued. The read operation would then get out of sync, producing a corrupted object during deserialization. This is fixed using the following function:

```
// Read len bytes from InputStream in into byte array buffer;
// blocks until at least len bytes are available.
private void readBytes(InputStream in, byte[] buffer, int len) {
      // Wait for bytes to become available
      try {
            do { Thread.yield(); } while (in.available() < len);
            in.read(buffer);
      } catch (IOException ioe) {}
}
```

- Implemented socket versions of registerAgentLocation, verifyAgentLocation, and receiveMessage. Tested using a modified version of CascadeTest.java.
- Added a UWUtility class for utility functions specific to the UWAgent execution engine.

Next work period: Implement the remaining RMI methods using the socket code. Update documentation.

Code Changes

The new UWUtility class:

```
public class UWUtility {
        // Constants
        // Sizes (in bytes); used for declaring byte arrays for agent navigation
        // and communication
        public static final int HEADER_SIZE = 40;       // total header size
        public static final int INT_SIZE = 4;           // one integer

        // maximum function name length
        public static final int FUNC_NAME_SIZE = HEADER_SIZE-INT_SIZE*3;

        public static final int MSG_TYPE_FUNC = 1;      // message type = function

        // Read len bytes from InputStream in into byte array buffer;
        // blocks until at least len bytes are available.
        private static void readBytes(InputStream in, byte[] buffer, int len) {
                // Wait for bytes to become available
                try {
                        do { Thread.yield(); } while (in.available() < len);
                        in.read(buffer);
                } catch (IOException ioe) {}
        }

        // Initialize a Socket to send a command to a UWPlace
        // Sends a header through the socket, and returns an OutputStream for sending
        // more data.
        public static OutputStream InitUWPSocket(int messageType, String hostName,
                        String portNum, String funcName, int headerParam1,
                           int headerParam2)
        {
                // Set up a socket to send the message header and body
                // The server socket should be set up to listen on this port in main()
                Socket skt = new Socket(hostName, Integer.parseInt(portNum));
                    OutputStream out = skt.getOutputStream();

                    // Set up a 40-byte header with the following structure
                    //  message type (int) | function name (string)
                    //  | header parameter 1 (int)
                    //          | header parameter 2 (int)
                    ByteBuffer headerBuff = ByteBuffer.allocate(HEADER_SIZE);
                    headerBuff.putInt(messageType);             // message type

                    // Function name
                    byte[] bytFuncName = funcName.getBytes("UTF8");
                    headerBuff.put(bytFuncName);
                    for (int i=bytFuncName.length; i<FUNC_NAME_SIZE; i++)
                    {
                        // pad with spaces to end of func name field
                        headerBuff.put((byte)' ');
                    }

                    headerBuff.putInt(headerParam1);
                    headerBuff.putInt(headerParam2);

                    out.write(headerBuff.array());              // write the header

                    return out;
        }
}
```

Here is how the new InitUWPSocket function is used to send a UWMessage between agents:

```
public boolean sendMessage(UWMessage message, InetAddress ip, long time) {
        InetAddress receivingIpAddr = message.getReceivingIp();

        // Write the byte representation of the message to the output
        // stream and send to the new place.
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        oos.writeObject(message);
        oos.close();
        byte[] byteArrayObj =  baos.toByteArray();
        baos.close();

        OutputStream out =
        UWUtility.InitUWPSocket(UWPlace.MSG_TYPE_FUNC,
                        receivingIpAddr.getHostName(),
                        getPortNumber(), "receiveMessage",
                        message.getReceivingAgentId(),
                        byteArrayObj.length);
        out.write(byteArrayObj);

        // Indicates message went to new UWAgentMailbox successfully
        System.out.println(message.getSendingAgentId() + " sends message to "
                + message.getReceivingAgentId() + " =) ");
}
```

Output

To test sending a UWMessage between agents, I used a test called CascadeTestLocal, which sends a UWMessage between two UWAgents on the local host. The source code is in `/home/uwagent/MA/UWAgent.new.`

mnode1

```
[uwagent@mnode1 UWAgent.new]$ java UWInject localhost CascadeTestLocal -m 10
File : /home/uwagent/MA/UWAgent.new
URL : file:/home/uwagent/MA/UWAgent.new/
URL : null
ip = mnode1/10.1.0.1, time = 1123258992656, ID = 0
file = ./CascadeTestLocal.class
byteArrayClass.length = 2061
UWPlace is ready at localhost:35354/UWPlace.
end of UWInject (main)
UWPlace main() received a header.
Function name is: receiveAgent
receiveAgent#Added class name: CascadeTestLocal
UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registered id= 0
agentsList.size = 1
ag = CascadeTestLocal@1efb836
ag.Name = CascadeTestLocal

0: Hello from CascadeTestLocal
0: My parent's ID = -1
REGISTER agent location:

agentId: 1
location: mnode1/10.1.0.1
UWAgent#registerAgentLocation
        myId = 0, agentId = 1, location = mnode1/10.1.0.1
REGISTER agent location:

agentId: 0
location: mnode1/10.1.0.1
UWAgent#registerAgentLocation
        myId = 1, agentId = 0, location = mnode1/10.1.0.1
0: waiting before sending message...
UWPlace is ready at localhost:35354/UWPlace.
UWPlace main() received a header.
Function name is: receiveAgent
receiveAgent#Added class name: CascadeTestLocal
UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registered id= 1
agentsList.size = 2
ag = CascadeTestLocal@1efb836
ag.Name = CascadeTestLocal
ag = CascadeTestLocal@19209ea
ag.Name = CascadeTestLocal

1: Hello from CascadeTestLocal
1: My parent's ID = 0
1: Hopping to localhost
```

```
UWAgent:notifyRelativesOfLocation
        reciptId = 0, recipAddr = mnode1/10.1.0.1, agentId = 1, addr = localhost
/127.0.0.1
Before getMailbox()
After getMailbox()
In notifyRelativesOfLocation:
Recipient Address: 1
Recipient id (UWAgentMailbox): localhost/127.0.0.1
Sender Address: 0
Sender id (UWAgentMailbox): mnode1/10.1.0.1
In notifyAgentLocation
Recipient Address: localhost/127.0.0.1
Recipient id (UWAgentMailbox): 1
Sender Address: mnode1/10.1.0.1
Sender id (UWAgentMailbox): 0
Writing to output stream
success = true
returning from notifyAgentLocation
        success = true
1: CascadeTestLocal exiting

UWPlace is ready at localhost:35354/UWPlace.
UWPlace is ready at localhost:35354/UWPlace.
UWPlace main() received a header.
Function name is: registerAgentLocation
UWPlace main() received a header.
Function name is: receiveAgent
REGISTER agent location:

agentId: 0
location: /10.1.0.1
UWAgent#registerAgentLocation
        myId = 0, agentId = 0, location = /10.1.0.1
receiveAgent#Added class name: CascadeTestLocal
UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registered id= 1
agentsList.size = 2
ag = CascadeTestLocal@1efb836
ag.Name = CascadeTestLocal
ag = CascadeTestLocal@1b8e059
ag.Name = CascadeTestLocal
1: waiting before receiving message...
[uwagent@mnode1 UWAgent.new]$ 0: Sending message
hostName: mnode1
talk to 1, mnode1/10.1.0.1, Hello from agent 0
0 sends message to 1 =)
0: Succeeded
0: CascadeTestLocal exiting

UWPlace is ready at localhost:35354/UWPlace.
UWPlace is ready at localhost:35354/UWPlace.
UWPlace main() received a header.
Function name is: verifyAgentLocation
UWPlace main() received a header.
Function name is: receiveMessage
1: Received the following message: Hello from agent 0

[uwagent@mnode1 UWAgent.new]$
```