CSS497 Summer 2005

Redesigning and Enhancing the UWAgent Execution Engine

Status Report: Fri 07/22/2005

Summary

Dates: Sat 07/16/2005 – Fri 07/22/2005

This work period:

- Fixed the BufferUnderflowException problem in receiveAgent(). The SpawnTest test now executes correctly, and UWPlace.main() uses a separate thread to read socket data.
- Started research on converting messaging from RMI to sockets.

Next work period: Extend the receiveAgent() code to work with UWMessagingSystem. The first step will be to upgrade notifyAgentLocation(), which is called at the beginning of a hop() operation. It uses RMI to look up the remote mailbox destination for message passing.

Code Changes

The modified socket code is in UWPlace.main():

```
public static void main( String[] args ) {
    try {
            ...
        // Listen on the UWPlace port number using a server socket
        int portNum = Integer.parseInt(uwplace.getPortNumber());

        ServerSocket srvr = new ServerSocket(portNum);

        // Busy waiting for UWAgent to come
        while (true) {
            SocketThread socketT = new SocketThread(uwplace, srvr);
            uwplace.engine(uwplace);
        }
        ...
}

class SocketThread implements Runnable
{
    Thread thread;
    UWPlace uwP;
    ServerSocket srvr;

    SocketThread(UWPlace uwplace, ServerSocket srvrSock) {
        // Instantiate thread
        thread = new Thread(this, "socketThread");

        // Initialize
        uwP = uwplace;
        srvr = srvrSock;

        thread.start();
```

```
}

public void run() {
    try {
        // Next statement blocks until client connects
        Socket skt = srvr.accept();
        InputStream in = skt.getInputStream();

        // Indicates this UWPlace is ready to accept UWAgent
        System.out.println("UWPlace is ready at localhost:"
                  + uwP.getPortNumber() + "/"
                  + uwP.getPlaceName() + ".");

        byte[] header = new byte[40];
        in.read(header);
        ByteBuffer bb = ByteBuffer.allocate(header.length);
        bb.put(header);
        bb.rewind();
        int type = bb.getInt();
        byte [] b = new byte[28];
        bb.get(b);
        String t = new String(b, "UTF8");
        t = t.trim();
        // size of byte array representation of object
        int byteArraySize = bb.getInt();
        int numClasses = bb.getInt();

        byte[] byteArrayObj = new byte[byteArraySize];
        in.read(byteArrayObj);

        int hashSize = 0;
        byte[] bytHashSize;
        HashMap classesHash = new HashMap();
        for (int i=0; i<numClasses; i++) {
            bytHashSize = new byte[4];
            in.read(bytHashSize);
            ByteBuffer bbh = ByteBuffer.allocate(bytHashSize.length);
            bbh.put(bytHashSize);
            bbh.rewind();
            int intHashSize = bbh.getInt();
            byte [] bytClassHash = new byte[intHashSize];
            in.read(bytClassHash);
            ByteBuffer bbch = ByteBuffer.allocate(bytClassHash.length);
            bbch.put(bytClassHash);
            bbch.rewind();
            int stringLength = bbch.getInt();
            int byteArrayLength = bbch.getInt();
            byte [] bytClassName = new byte[stringLength];
            bbch.get(bytClassName);
            String className = new String(bytClassName, "UTF8");
            byte [] bytClassFile = new byte[byteArrayLength];
            bbch.get(bytClassFile);
            classesHash.put(className, bytClassFile);
        }
        in.close();
        skt.close();

        // This will push the agent on to the stack
        uwP.receiveAgent(byteArrayObj, classesHash, null);
```

Output

Here is the output on mnode0/mnode1.
The source code is in `/home/uwagent/MA/UWAgent.new`.

mnode1

```
[uwagent@mnode1 UWAgent.new]$ java UWInject mnode0 SpawnTest -m 3 6 5
File : /home/uwagent/MA/UWAgent.new
URL : file:/home/uwagent/MA/UWAgent.new/
URL : null
ip = mnode1/10.1.0.1, time = 1122053331758, ID = 0
file = ./SpawnTest.class
byteArrayClass.length = 1612
Setting up a socket in sendAgent()
Write header and agent to socket
Write serialized classesHash to socket (start)
Write serialized classesHash to socket (end)
end of UWInject (main)
[uwagent@mnode1 UWAgent.new]$
```

mnode0

```
[uwagent@mnode0 UWAgent.new]$ java UWPlace &
[1] 12250
[uwagent@mnode0 UWAgent.new]$ All backup files are deleted.
Creating server socket.

In SocketThread
Calling uwplace.engine()
Popping agent from stack
Accepting connections.

Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Done reading header
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1224
Number of classes: 1
Class name: SpawnTest
Closing socket
Calling receiveAgent()
UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registerd id= 0
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Accepting connections.

Agentthread#Agentthread: before activateMailbox(), myId = 0
In SocketThread

1: Hello from SpawnTest
1: My ID = 0
1: My parent's ID = -1
1: My class name = SpawnTest
1: My parent's class name = null
1: SpawnTest spawning SpawnTest2
Calling uwplace.engine()
Popping agent from stack
Accepting connections.
```

```
Setting up a socket in sendAgent()
Write header and agent to socket
Write serialized classesHash to socket (start)
Write serialized classesHash to socket (end)
1: Number of children I have spawned so far = 1
1: SpawnTest spawning SpawnTest2
Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Done reading header
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1332
Number of classes: 1
Class name: SpawnTest2
Closing socket
Calling receiveAgent()
UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registerd id= 1
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 1
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Accepting connections.


2: Hello from SpawnTest2
2: My ID = 1
2: My parent's ID = 0
2: My class name = SpawnTest2
2: My parent's class name = SpawnTest
2: SpawnTest2 spawning SpawnTest3
Accepting connections.

Setting up a socket in sendAgent()
Write header and agent to socket
Write serialized classesHash to socket (start)
Write serialized classesHash to socket (end)
2: Number of children I have spawned so far = 1
2: SpawnTest2 spawning SpawnTest3
Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Done reading header
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1322
Number of classes: 1
Class name: SpawnTest3
Closing socket
Calling receiveAgent()
Setting up a socket in sendAgent()
Write header and agent to socket
Write serialized classesHash to socket (start)
Write serialized classesHash to socket (end)
2: Number of children I have spawned so far = 2
2: SpawnTest2 spawning SpawnTest3
Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Done reading header
```

```
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1322
Number of classes: 1
Class name: SpawnTest3
Closing socket
Calling receiveAgent()
Setting up a socket in sendAgent()
Write header and agent to socket
Write serialized classesHash to socket (start)
Write serialized classesHash to socket (end)
2: Number of children I have spawned so far = 3
2: SpawnTest2 spawning SpawnTest3
UWAgent#spawnChild: Cannot spawn more than 3 children
2: Number of children I have spawned so far = 3
2: SpawnTest2 spawning SpawnTest3
UWAgent#spawnChild: Cannot spawn more than 3 children
2: Number of children I have spawned so far = 3
2: SpawnTest2 exiting

UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registerd id= 3
UWPlace#receiveAgent: before activateMailbox()
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 3
Accepting connections.

Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Done reading header
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1322
Number of classes: 1
Class name: SpawnTest3
Closing socket
Calling receiveAgent()
UWPlace#receiveAgent: before activateMailbox()
Setting up a socket in sendAgent()
Write header and agent to socket
Write serialized classesHash to socket (start)
Write serialized classesHash to socket (end)
1: Number of children I have spawned so far = 2
1: SpawnTest spawning SpawnTest2
UWAgent#spawnChild: Cannot spawn more than 2 children
1: Number of children I have spawned so far = 2
1: SpawnTest spawning SpawnTest2
UWAgent#spawnChild: Cannot spawn more than 2 children
1: Number of children I have spawned so far = 2
1: SpawnTest spawning SpawnTest2
UWAgent#spawnChild: Cannot spawn more than 2 children
1: Number of children I have spawned so far = 2
1: SpawnTest spawning SpawnTest2
UWAgent#spawnChild: Cannot spawn more than 2 children
1: Number of children I have spawned so far = 2
1: SpawnTest exiting

Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Done reading header
```

```
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1332
Number of classes: 1
Class name: SpawnTest2
Closing socket
Calling receiveAgent()
UWPlace#receiveAgent: before activateMailbox()
In SocketThread
UWPlace#receiveAgent: registerd id= 5

3: Hello from SpawnTest3
3: My ID = 3
3: My parent's ID = 1
3: My class name = SpawnTest3
3: My parent's class name = SpawnTest2
3: SpawnTest3 exiting

UWPlace#receiveAgent: registerd id= 4
UWPlace#receiveAgent: registerd id= 2
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 5
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 4
In SocketThread
Accepting connections.


3: Hello from SpawnTest3
3: My ID = 5
3: My parent's ID = 1
3: My class name = SpawnTest3
3: My parent's class name = SpawnTest2
3: SpawnTest3 exiting

Accepting connections.

Calling uwplace.engine()

3: Hello from SpawnTest3
3: My ID = 4
3: My parent's ID = 1
3: My class name = SpawnTest3
3: My parent's class name = SpawnTest2
3: SpawnTest3 exiting

Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 2
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Accepting connections.


2: Hello from SpawnTest2
2: My ID = 2
2: My parent's ID = 0
2: My class name = SpawnTest2
2: My parent's class name = SpawnTest
2: SpawnTest2 spawning SpawnTest3
```

```
Accepting connections.

Setting up a socket in sendAgent()
Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Write header and agent to socket
Done reading header
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1322
Number of classes: 1
Write serialized classesHash to socket (start)
Class name: SpawnTest3
Closing socket
Calling receiveAgent()
Write serialized classesHash to socket (end)
2: Number of children I have spawned so far = 1
2: SpawnTest2 spawning SpawnTest3
Setting up a socket in sendAgent()
Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Write header and agent to socket
Done reading header
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1322
Number of classes: 1
Write serialized classesHash to socket (start)
Class name: SpawnTest3
Closing socket
Calling receiveAgent()
UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registerd id= 7
Write serialized classesHash to socket (end)
2: Number of children I have spawned so far = 2
2: SpawnTest2 spawning SpawnTest3
Setting up a socket in sendAgent()
Write header and agent to socket
Write serialized classesHash to socket (start)
Write serialized classesHash to socket (end)
2: Number of children I have spawned so far = 3
2: SpawnTest2 spawning SpawnTest3
UWAgent#spawnChild: Cannot spawn more than 3 children
2: Number of children I have spawned so far = 3
2: SpawnTest2 spawning SpawnTest3
UWAgent#spawnChild: Cannot spawn more than 3 children
2: Number of children I have spawned so far = 3
2: SpawnTest2 exiting

Server socket connected.

UWPlace is ready at localhost:35354/UWPlace.
Done reading header
UWPlace main() received a header.
Type field is: 1
Function name is: receiveAgent
Size of byte array: 1322
Number of classes: 1
Class name: SpawnTest3
Closing socket
Calling receiveAgent()
UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registerd id= 6
```

```
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 7
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 6
Accepting connections.


3: Hello from SpawnTest3
3: My ID = 7
3: My parent's ID = 2
3: My class name = SpawnTest3
3: My parent's class name = SpawnTest2
3: SpawnTest3 exiting

Accepting connections.

In SocketThread
Calling uwplace.engine()
Popping agent from stack

3: Hello from SpawnTest3
3: My ID = 6
3: My parent's ID = 2
3: My class name = SpawnTest3
3: My parent's class name = SpawnTest2
3: SpawnTest3 exiting

Accepting connections.

UWPlace#receiveAgent: before activateMailbox()
UWPlace#receiveAgent: registerd id= 8
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Creating a new AgentThread
Agentthread#Agentthread: before activateMailbox(), myId = 8
In SocketThread
Calling uwplace.engine()
Popping agent from stack
Accepting connections.


3: Hello from SpawnTest3
3: My ID = 8
3: My parent's ID = 2
3: My class name = SpawnTest3
3: My parent's class name = SpawnTest2
3: SpawnTest3 exiting

Accepting connections.
```