CSS497 Fall 2005

Redesigning and Enhancing the UWAgent Execution Engine

Status Report: Fri 10/21/2005

Summary

Dates: Sat 10/15/2005 – Fri 10/21/2005

This work period:

- Started inter-agent communication over a gateway
    - GatewayMessageTest agent
    - New registerAgentIpGateway socket method

Next work period:

- Finish inter-agent communication over a gateway
    - Use a child agent's location information to send a message from the parent over the list of gateways to the child's current location.

Notes

- Path and classpath conflicts on mnodeN and medusa
- When an agent migrates to a node over a gateway, it now needs to send to its parent/children both its current location, and the list of gateways it used to get there. The problem: the agent's parent is now on the other side of the gateway, so the registerAgentIp call fails because the parent is unreachable over a socket connection. I created a new SocketThread method called registerAgentIpGateway. This method traverses the list of gateways until it reaches the parent agent's node, and then communicates the child agent's location information.

Code Changes

- I created a test agent called GatewayMessageTest based on the MessageTest agent. When executed on uw1-320-xx, it does the following:
    - The parent agent spawns a child agent, waits, and then attempts to send it a message.
    - The child agent hops to mnodeN, using the medusa gateway. It then hops to mnodeN+1 and waits. Finally, it attempts to receive a message and print it out.

Here is the code for GatewayMessageTest:

```
import java.io.*;

public class GatewayMessageTest extends UWAgent implements Serializable {
    private String hopTo = "mnode1";            // first destination node
    private String hopToLast = "mnode2";        // last destination node
    private String firstGateway = "medusa";     // gateway protecting the destination node
    private String currentLocation = "";

    public GatewayMessageTest() {}
```

```
public GatewayMessageTest(String[] hopToArgs)
{
    System.out.println("hopToArgs[0] = " + hopToArgs[0]);
    if (hopToArgs.length > 0) {
        hopTo = hopToArgs[0];
    }

    int nodeNum = -1;
    if (hopTo.startsWith("mnode")) {
        String sNodeNum = hopTo.substring(5, hopTo.length());
        nodeNum = Integer.parseInt(sNodeNum);
        nodeNum++;  // get next node number in mnode cluster
        hopToLast = "mnode" + String.valueOf(nodeNum);
        System.out.println("hopToLast = " + hopToLast);
    }
}

private void printMsg(String message) {
    if (message == null || message.equals("")) {
        System.out.println();
    } else {
        System.out.println(getAgentId() + ": " + message);
    }
}

public void printMsg(String[] args) {
    printMsg(args[0]);
}

public void DoChildTask()
{
    if (currentLocation.equals(hopToLast)) {
        // We have reached the last node, so look for a message
        printMsg("waiting before receiving message...");
        try {
            Thread.sleep(12000);
        } catch (InterruptedException e) {}
        // Dequeue a new message from the queue
        UWMessage receivedMessage = retrieveNextMessage();
        // Retrieve the sender agent's ID from the message
        int senderAgentId = receivedMessage.getSendingAgentId();
        // Extract the message into a string array
        String[] message = receivedMessage.getMessageHeader();
        printMsg("Received the following message: " + message[0]);
    } else if (currentLocation.equals("")) {
        // We are at the starting node (on the public network), so hop to the next node
        currentLocation = hopTo;
        printMsg("Hopping to " + hopTo);
        String[] gateway = new String[1];
        gateway[0] = firstGateway;
        // Need the gateway argument to get from the public network to the private network
        hop(hopTo, gateway, "DoChildTask", null);
    } else if (currentLocation.equals(hopTo)) {
        // We are at the intermediate node, so hop to the last node
        currentLocation = hopToLast;
        printMsg("Hopping to " + hopToLast);
        // Don't need a gateway argument, since we're already behind the gateway
        hop(hopToLast, "DoChildTask", null);
    }
}

public void ContactAgent() {
    printMsg("waiting before sending message...");
    try {
        Thread.sleep(6000);
    } catch (InterruptedException e) {}
    printMsg("Sending message");
    UWMessage message = new UWMessage(this, "Hello from agent " + getAgentId());
    boolean success = talk(1, message);
    if (success) {
```

```
            printMsg("Succeeded");
        } else {
            printMsg("Failed");
        }
    }

    public void init() {
        printMsg("");
        printMsg("Hello from GatewayMessageTest");
        printMsg("My parent's ID = " + getParentId(getAgentId()));

        if (getAgentId() == 0) {
            // This is the parent, so spawn a child agent which will then hop to other nodes
            spawnChild("GatewayMessageTest");
            // Contact the child agent
            ContactAgent();
        } else {
            // This is a child, so hop to another node or look for a message
            DoChildTask();
        }

        printMsg("GatewayMessageTest exiting");
        printMsg("");
    }
}
```

Here is the registerAgentIpGateway handler in SocketThread. Notice that the agent information as well as the gateway information needs to be passed over the socket:

```
} else if (strFuncName.equals("registerAgentIpGateway")) {
    // Gateway version of registerAgentIp

    int ts = headerParam1;  // agent ID
    int dataLength = headerParam2;  // length of additional data

    // Read byte[] representation of address
    byte[] bytRawAddress = new byte[UWUtility.INT_SIZE];
    readBytes(in, bytRawAddress, UWUtility.INT_SIZE);

    // Read byte[] representation of gateway array
    byte[] bytGateways = new byte[dataLength - UWUtility.INT_SIZE];
    readBytes(in, bytGateways, dataLength - UWUtility.INT_SIZE);

    String strGateways = null;
    try {
        strGateways = new String(bytGateways, "UTF8");      // Decode using UTF8
        strGateways = strGateways.trim();
    } catch (UnsupportedEncodingException uee) {}
    System.out.println("strGateways = " + strGateways);

    String[] gateways = null;
    try {
        gateways = strGateways.split(" ");
    } catch (java.util.regex.PatternSyntaxException pse) {}
    System.out.println("gateways[0] = " + gateways[0]);

    // Read destination host name
    byte[] bytHostName = new byte[UWUtility.HOSTNAME_SIZE];
    readBytes(in, bytHostName, UWUtility.HOSTNAME_SIZE);

    String strHostName = null;
    try {
        strHostName = new String(bytHostName, "UTF8");      // Decode using UTF8
        strHostName = strHostName.trim();
    } catch (UnsupportedEncodingException uee) {}
    System.out.println("strHostName = " + strHostName);

    // Read position in list of gateways
```

```
byte[] bytGwpos = new byte[UWUtility.INT_SIZE];
readBytes(in, bytGwpos, UWUtility.INT_SIZE);
int gwPos = UWUtility.BytesToInt(bytGwpos);
System.out.println("gwPos = " + gwPos);

// Read agent id
byte[] bytFs = new byte[UWUtility.INT_SIZE];
readBytes(in, bytFs, UWUtility.INT_SIZE);
int fs = UWUtility.BytesToInt(bytFs);
System.out.println("fs = " + fs);

in.close();
skt.close();

gwPos--;
if (gwPos==0) {
    // We're done traversing gateways, so call registerAgentIp on the destination node
    OutputStream out =
        UWUtility.InitUWPSocket(UWUtility.MSG_TYPE_FUNC, strHostName,
            uwP.getPortNumber(), "registerAgentIp", ts, bytRawAddress.length);

    out.write(bytRawAddress);  // write the byte[] representation of the InetAddress
    out.write(bytFs);
} else {
    // We still have gateways to traverse, so recursively call
    // registerAgentIpGateway on the next gateway
    OutputStream out =
        UWUtility.InitUWPSocket(UWUtility.MSG_TYPE_FUNC, gateways[gwPos],
            uwP.getPortNumber(), "registerAgentIpGateway", ts,
            bytRawAddress.length + bytGateways.length + bytHostName.length);

    out.write(bytRawAddress);               // write address
    out.write(bytGateways);                 // write gateway array
    out.write(bytHostName);                 // write destination host name

    ByteBuffer gwPosBuffNew = ByteBuffer.allocate(UWUtility.INT_SIZE);
    gwPosBuffNew.putInt(gwPos);
    out.write(gwPosBuffNew.array());        // write gateway position

    out.write(bytFs);                       // write agent ID
}
```