CSS497 Fall 2005

Redesigning and Enhancing the UWAgent Execution Engine

Status Report: Fri 10/07/2005

Summary

Dates: Sat 10/01/2005 – Fri 10/07/2005

This work period:

- Completed navigation over a gateway, both from a private node to a public node (e.g., mnode8 to uw1-320-20) and from a public node to a private node (e.g., uw1-320-20 to mnode8).
- Started modifying UWAgent.java as specified in ~/orange/temp_AgentUtil/UWAgent.java.

Next work period:

- Implement inter-agent communication over a gateway.
- Complete UWAgent code cleanup.

Code Changes

The major changes required for navigation over a gateway were in UWAgent.java:

- Modify the original hop() method to use the gateway specified in the -g option
- Add a new hop() method that accepts a gateway[] array
- Add a new method called hopGateway() to hop across each gateway in the array

The modified hop() method retrieves the -g option from the current UWPlace. If the -g option is not specified, then the method works as it did previously: the agent is sent directly to the destination. If the -g option is specified, then hop() first tries to contact the destination directly (in case it is not on the other side of the gateway). If this fails, then the gateway version of hop() is used.

In the UWB environment, uw1-320-20 is visible to mnode8 (e.g., using ping). However, an attempted socket connection times out. Therefore, SocketTimeoutException is used to indicate that mnode8 is behind a gateway.

The new hop() method that accepts a gateway array works as follows:

- Convert the gateway[] array into a Stack that is stored as a member variable in the UWAgent object.
- Save the host name, function name, and function arguments in UWAgent member variables
- Call hopGateway()

The hopGateway() method works as follows:

- If the gateway stack is empty, hop to the destination.
- Otherwise, pop the next gateway off the stack and hop to it, calling hopGateway() again on arrival

```
public final void hop(String hostName, String funcName, String[] funcArgs) {
    boolean socketError = false;
    String uwpGateway = place.getGateway();     // get -g option from current UWPlace
    if (!("".equals(uwpGateway))) {             // gateway specified
        // Try to contact hostName directly using a socket connection
        try {
            OutputStream out =
                UWUtility.InitUWPSocket(UWUtility.MSG_TYPE_FUNC, hostName,
                    place.getPortNumber(), "detectHost", 0, 0);
            // OutputStream out is unused since there is no data to send
        } catch (SocketTimeoutException e) {
            // Host name found, but isn't responding
            socketError = true;
        } catch (UnknownHostException e) {
            // Host name not found
            socketError = true;
        } catch (Exception e) {
            UWUtility.Log(e.toString());
            UWUtility.Log("Cause: " + e.getCause());
        }

        if (socketError) {
            // No response, so try the gateway
            String[] gateway = new String[1];
            gateway[0] = uwpGateway;
            hop(hostName, gateway, funcName, funcArgs);
            return;
        }
    }
    // Socket connection succeeded (or no gateway specified), so just go
    // directly to destination
    setNextFunc(funcName);
    setFuncArgs(funcArgs);
    try {
        notifyRelativesOfLocation(InetAddress.getByName(hostName));
    } catch (java.net.UnknownHostException e) {}

    getPlace().sendAgent(this, hostName);

    UWUtility.LogExit();
}

// Hop across each gateway
// Must be public so it is visible to AgentThread.run()
public final void hopGateway()
{
    UWUtility.LogEnter();
    try {
        if (destGateway == null || destGateway.empty()) {
            hop(destHostName, destFuncName, destFuncArgs);
        } else {
            String nextHostName = (String)destGateway.pop();
            String[] funcArgs = null;
            hop(nextHostName, "hopGateway", funcArgs);
        }
    } catch (Exception e) {
        UWUtility.Log(e.toString());
        UWUtility.Log("Cause: " + e.getCause());
    }
    UWUtility.LogExit();
}

// hop through gateway list to destination host
```

```
public final void hop(String hostName, String[] gateway, String funcName, String[] funcArgs) {
    UWUtility.LogEnter("gateway version; hostName = " + hostName + ", funcName = " + funcName);

    try {
        // Initialize destination variables, so that the current agent always knows
        // its destination, and which gateways to use to get there. These member
        // variables will travel with the agent as it moves from host to host.
        destHostName = hostName;

        if (gateway == null) {
            destGateway = null;
        } else {
            // Create gateway Stack from gateway array
            destGateway = new Stack();
            for (int i=gateway.length-1; i>=0; i--) {
                destGateway.push(gateway[i]);
            }
        }

        destFuncName = funcName;

        if (funcArgs == null) {
            destFuncArgs = null;
        } else {
            destFuncArgs = new String[funcArgs.length];
            for (int i=0; i<=funcArgs.length-1; i++) {
                destFuncArgs[i] = funcArgs[i];
            }
        }

        // Start hopping through the gateway list
        hopGateway();
    } catch (Exception e) {
        UWUtility.Log(e.toString());
        UWUtility.Log("Cause: " + e.getCause());
    }
    UWUtility.LogExit();
}
```

Output

The source code is in `/home/uwagent/MA/UWAgent.new`.

Private to public with **no** gateway specified. Operation fails because public node is not reachable.

mnode8

```
[uwagent@mnode8 UWAgent.new]$ java UWPlace &
[1] 4671
[uwagent@mnode8 UWAgent.new]$ java UWInject localhost GatewayHopTest uw1-320-20
[uwagent@mnode8 UWAgent.new]$
0: Hello from GatewayHopTest
0: Hopping to uw1-320-20
0: Coming FROM medusa cluster, so use regular hop method
UWPlace#sendAgent: java.net.SocketTimeoutException: connect timed out
UWPlace#sendAgent: Cause: null
0: GatewayHopTest exiting


[uwagent@mnode8 UWAgent.new]$
```

medusa

```
[uwagent@medusa UWAgent.new]$ java UWPlace
```

uw1-320-20

```
[duncans@uw1-320-20 UWAgent.new]$ java UWPlace
```

Private to public **with** gateway specified. Operation succeeds by using the gateway machine as an intermediate hop.

mnode8

```
[uwagent@mnode8 UWAgent.new]$ java UWPlace -g medusa &
[1] 4708
[uwagent@mnode8 UWAgent.new]$ java UWInject localhost GatewayHopTest uw1-320-20
[uwagent@mnode8 UWAgent.new]$
0: Hello from GatewayHopTest
0: Hopping to uw1-320-20
0: Coming FROM medusa cluster, so use regular hop method
0: GatewayHopTest exiting

[uwagent@mnode8 UWAgent.new]$
```

medusa

```
[uwagent@medusa UWAgent.new]$ java UWPlace
```

uw1-320-20

```
[duncans@uw1-320-20 UWAgent.new]$ java UWPlace
0: *** Hop to uw1-320-20 complete ***
```

Public to private **with** gateway specified. Operation succeeds by using the gateway machine as an intermediate hop.

uw1-320-20

```
[duncans@uw1-320-20 UWAgent.new]$ java UWPlace &
[1] 28769
[duncans@uw1-320-20 UWAgent.new]$ java UWInject localhost GatewayHopTest mnode8 medusa
[duncans@uw1-320-20 UWAgent.new]$
0: Hello from GatewayHopTest
0: Hopping to mnode8
0: Going TO medusa cluster, so hop to destination using a gateway
0: GatewayHopTest exiting

[duncans@uw1-320-20 UWAgent.new]$
```

medusa

```
[uwagent@medusa UWAgent.new]$ java UWPlace
```

mnode8

```
[uwagent@mnode8 UWAgent.new]$ java UWPlace
0: *** Hop to mnode8 complete ***
```