Distributed Protocol Factory 2.0b Requirements Specification

1. Scope

1.1 Overview

This enhancement adds a Bridge Node capability. A bridge Node allows the creation of more complex Node networks by allowing communications between separate Node Groups. A Bridge Node bridges two or more otherwise unconnected groups. A bridge node will receive messages from its primary group, and retransmit the messages to the nodes belonging to its sub-group.

1.2 Logic Requirements

1.2.1 Bridge Node

The Bridge Node will bridge message communications between two or more Node Groups.
A Node Group is a logical grouping of nodes identified as belonging together. All nodes in a Node Group will be on the same physical subnet. A physical subnet may have multiple Node Groups on it.
The designation primary and secondary are based on which Group originates the message traffic. The Group from which the initial message is generated (including a message initiated by the bridge node itself) is designated the Primary Group. The Secondary Group(s) is(are) the Groups from where the message did not originate from.
For any one message flow there can be only one Primary Group.
There may be one or more Secondary Groups.

1.2.2 Masked Mode Node

In Masked Mode, nodes in separate Groups are not aware of each other. The Bridge Node will acknowledge its own message receipt on the Primary Group.
The nodes on each Secondary Group will act as a separate node network for message acknowledgement purposes.
Assuming all Nodes on the Primary Group acknowledge, then the Bridge Node must acknowledge the message, regardless of whether any Secondary Groups do not acknowledge the message.
If the Nodes on the Primary Group do not acknowledge the message, but one or more Secondary Groups all acknowledge the message, then the Bridge Node must acknowledge the message.
It will be up to the application to determine course of action when a message is acknowledged in one Group but not the other.

1.2.3 UnMasked Mode Node

In Unmasked Mode, nodes on separate Groups are not directly aware of each other.
A Bridge Node will only acknowledge receipt of a message from the Primary Group if all nodes in its Secondary Groups also acknowledge receipt of the message.

1.2.4 Multiple Mode subnets

The bridge and non-bridge nodes must be able to operate in a highly complex, Multiple Mode Group environments.
A Multiple Mode Group environment is where a Group may have a Bridge Node working in Masked Mode with one Group, and another Bridge Node working in Unmasked Mode with another Group. Groups may be chained (A to B to C to D etc.). Section 1.2.5 provides rules to, in most cases, prevent infinite transmission loops; ultimately, however, it is up to the application owner and not the component to ensure infinite loops do not occur.
A Node may participate in multiple Groups.
Message traffic sequence integrity will be maintained at the subnet level only.

1.2.5 Multiplicity

A Node may not appear more than once in a Node Group.
A Bridge Node will not retransmit a received message it itself originated.
A non-bridge Node that belongs to more than on Node Group and that originates a message must do so separately and independently for each Node Group it belongs to.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

When the node network grows very large it may become inefficient and time consuming to communicate with all nodes at once. A Bridge Node allows breaking down a large node network into multiple smaller node networks to better manage message flow. A Masked Mode Bride Node further allows semi-independent connected node networks to share messages but operate independently.

1.5 Future Component Direction

None identified.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.4
- Compile target: Java 1.4
- Multiple runtime environments
  - WebLogic
  - JBoss
  - JVM 1.4
- It is not guaranteed that the component will be running inside a J2EE container, but the J2EE jar will be accessible.

2.1.4 Package Structure

com.topcoder.network.synchronization.bridged

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the component need to be configurable?

The protocol implementation should be configurable programmatically.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

As determined by design.

3.2.2 TopCoder Software Component Dependencies:

None. (Do not use configuration manager, directly or indirectly.)

Component does not need to use base exception (but can at designers discretion) because Java 1.4 supports chained exceptions natively.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of Poseidon.