```
import java.io.IOException;
import java.io.InputStream;
import java.util.Enumeration;
import java.util.Vector;

import net.jxta.discovery.DiscoveryService;
import net.jxta.document.AdvertisementFactory;
import net.jxta.document.Element;
import net.jxta.document.MimeMediaType;
import net.jxta.document.StructuredDocument;
import net.jxta.document.StructuredDocumentFactory;
import net.jxta.endpoint.Message;
import net.jxta.endpoint.StringMessageElement;
import net.jxta.exception.PeerGroupException;
import net.jxta.id.IDFactory;
import net.jxta.peergroup.PeerGroup;
import net.jxta.peergroup.PeerGroupFactory;
import net.jxta.pipe.InputPipe;
import net.jxta.pipe.OutputPipe;
import net.jxta.pipe.PipeService;
import net.jxta.protocol.PeerGroupAdvertisement;
import net.jxta.protocol.PipeAdvertisement;


// The HungryPeer joins the RestoNet PeerGroup and searches for
// RestoPeers. The HungryPeer then establishes a pipe connection to
// all the RestoPeers that it discovered. The HungryPeer sends
// auction requests for French fries to RestoPeers and then waits for
// auction bids from RestoPeers

public class HungryPeer {

    private PeerGroup netpg = null;     // NetPeergroup
    private PeerGroup restoNet = null;  // Resto Peergroup

    // Services within the RestoNet Peergroup
    private DiscoveryService disco;     // Discovery Service
    private PipeService pipes;          // Pipe Service
    private PipeAdvertisement myAdv;    // Hungry peer pipe advertisement
    private InputPipe myPipe;           // Input pipe to talk to hungry pe
er
    private MimeMediaType mimeType = new MimeMediaType("text", "xml");

    private int timeout = 3000;         // Discovery timeout
    private int rtimeout = 30000;       // Pipe Resolver Timeout
    // All RestoPeers found
    private Vector restoPeerAdvs = new Vector();
    private Vector restoPeerPipes = new Vector();

    private String myIdentity = "Bill Joy";  // Identity of this HungryPeer
    private String friesRequest ="medium";   // Fries Auction request

    public static void main(String args[]) {
        HungryPeer myapp = new HungryPeer();
        myapp.startJxta();
        System.exit(0);
    }

    private void startJxta() {
        try {
            // Discover (or create) and join the default jxta NetPeerGroup
            netpg = PeerGroupFactory.newNetPeerGroup();
        } catch (PeerGroupException e) {
            //Couldn't initialize; can't continue
```



```
            System.out.println("Fatal error : creating the NetPeerGroup");
            System.exit(1);
        }

        // Discover and join the RestoNet Peergroup
        try {
            if (!joinRestoNet()) {
                System.out.println("Sorry could not find the RestoNet Peergroup");
                System.exit(2);
            }
        } catch (Exception e) {
            System.out.println("Can't join RestoNet group");
            System.exit(1);
        }

        // Set our HungryPeer communication pipe so RestoPeers
        // can talk to us
        if (!setHungryPeerPipe()) {
            System.out.println(
                "Aborting due to failure to create our HungryPeer pipe");
            System.exit(1);
        }

        // Attempt to locate RestoPeers in RestoNet
        discoverRestoPeers();

        // Connect to RestoPeers that have been discovered
        connectToRestoPeers();

        // I am hungry. Send an auction request for French Fries
        // to the connected RestoPeers.
        sendFriesAuctionRequests();

        //Process incoming bids from RestoPeers
        receiveFriesBids();
    }

    // This method is used to discover the RestoNet Peergroup.
    // If found the peer will join the peergroup
    private boolean joinRestoNet() {

        int count = 3; // maximum number of attempts to discover

        System.out.println("Attempting to discover the RestoNet Peergroup");

        // Get the Discovery service handle from the NetPeerGroup
        DiscoveryService hdisco = netpg.getDiscoveryService();

        // All discovered RestoNet Peers
        Enumeration ae = null;

        // Loop until we find the "RestoNet" Peergroup advertisement
        // or we've exhausted the desired number of attempts
        while (count-- > 0) {
            try {
                // Check if we have the advertisement in the local
                // peer cache
                ae = hdisco.getLocalAdvertisements(DiscoveryService.GROUP,
                                                   "Name", "RestoNet");

                // If we found the RestoNet advertisement, we are done
                if ((ae != null) && ae.hasMoreElements())
                    break;
```

```
                // The RestoNet advertisement is not in the local
                // cache . Send a discovery request to search for it.
                hdisco.getRemoteAdvertisements(null,
                           DiscoveryService.GROUP, "Name", "RestoNet", 1, n
ull);

                // Wait to give peers a chance to respond
                try {
                    Thread.sleep(timeout);
                } catch (InterruptedException ie) {}
            } catch (IOException e) {
                // Found nothing! Move on.
            }
        }

        // Check if we found the RestoNet advertisement
        if (ae == null || !ae.hasMoreElements()) {
            return false;
        }

        System.out.println("Found the RestoNet PeerGroup Advertisement");
        // Get the advertisement
        PeerGroupAdvertisement adv =
            (PeerGroupAdvertisement) ae.nextElement();

        try {
            // Call the PeerGroup Factory to instantiate a new
            // peergroup instance
            restoNet = netpg.newGroup(adv);

            // Get the Discovery and Pipe services to
            // be used within the RestoNet Peergroup
            disco = restoNet.getDiscoveryService();
            pipes = restoNet.getPipeService();
        } catch (Exception e) {
          System.out.println("Could not create RestoPeerGroup");
          return false;
        }

        System.out.println("The HungryPeer joined the restoNet PeerGroup");
        return true;
    }

    // Create the HungryPeer pipe to receive bid responses
    // from RestoPeers. The advertisement of this pipe is sent as part
    // of the auction request for RestoPeers to respond.
    private boolean setHungryPeerPipe() {
        try {
            // Create a pipe advertisement for our hungry peer. This
            // pipe will be used within the RestoNet peergroup for other
            // peers to talk to our hungry peer
            myAdv = (PipeAdvertisement)
                AdvertisementFactory.newAdvertisement(
                PipeAdvertisement.getAdvertisementType());

            // Initialize the advertisement with unique peer information
            // So we can communicate
            myAdv.setPipeID(IDFactory.newPipeID(restoNet.getPeerGroupID())
);
            myAdv.setName("restoNet:HungryPipe:" + myIdentity);

            // Set the pipe type to be unicast unidirectional
            myAdv.setType(PipeService.UnicastType);
```



```
            // Create the input pipe
            myPipe = pipes.createInputPipe(myAdv);
        } catch (Exception e) {
            System.out.println("Could not create the HungryPeer pipe");
            return false;
        }
        return true;
    }

    // Discover RestoPeers that have joined RestoNet.
    // RestoPeers are discovered via their published pipe advertisement.
    private void discoverRestoPeers() {
        int found = 0;          // Count of RestoPeers found
        int count = 10;         // Discovery retries

        System.out.println("Locating RestoPeers in the RestoNet Peergroup");

        // Try to find at least two RestoPeers (an arbitrary number)
        // RestoPeers are found by their pipe advertisements
        while (count-- >0) {
            try {
                // Check if we already have restaurant advertisements
                // in our local peer cache
                Enumeration ae =
                    disco.getLocalAdvertisements(DiscoveryService.ADV,
                                  "name", "RestoNet:RestoPipe:*");

                // If we found some advertisements in the cache,
                // add them to the list
                if (ae != null && ae.hasMoreElements()) {
                    // Reset count and RestoPeerAdvs as we have
                    // just retrieved all advertisements, including
                    // previously cached ones
                    found = 0;
                    restoPeerAdvs.removeAllElements();
                    while (ae.hasMoreElements()) {
                        restoPeerAdvs.addElement(ae.nextElement());
                        ++found;
                    }
                    if (found > 1)
                        break;     // Want to find at least two
                }

                // Did not find enough advertisement in the cache.
                // Send a remote discovery request to search for
                // more RestoPeer advertisements
                disco.getRemoteAdvertisements(null,
                            DiscoveryService.ADV,
                            "name", "RestoNet:RestoPipe:*", 5, null);

                // Give the peers a chance to respond
                try {
                    Thread.sleep(timeout);
                } catch (InterruptedException e) {}
            } catch (IOException e){
                // Found nothing! Move on
            }
        }
        // Completed RestoPeer Discovery
        System.out.println("Found " + found + " RestoPeers(s)");
    }

    // Method to connect and open output pipes to all the
    // RestoPeers that we have discovered. Each RestoPeer is
```

```
    // identified by its unique RestoPeer pipe advertisement.
    private void connectToRestoPeers() {
        // Enumerate all the RestoPeer pipe advertisments we have discover
ed
        // and attempt to connect a pipe which each of them
        for (Enumeration en = restoPeerAdvs.elements();
             en.hasMoreElements();) {

            PipeAdvertisement padv = (PipeAdvertisement) en.nextElement();
            try {
                System.out.println(
                    "Attempting to connect to discovered RestoPeer");

                // Create an output pipe connection to the RestoPeer
                OutputPipe pipeOut = pipes.createOutputPipe(padv,
                                                            rtimeout);

                // Check if we have a connected pipe
                if (pipeOut == null) {
                    // Failed; go to next RestoPeer
                    System.out.println(
                        "Failure to connect to RestoPeer Pipe:" +
                        padv.getName());
                    continue;
                }

                // Save the output pipe
                restoPeerPipes.addElement(pipeOut);
                System.out.println("Connected pipe to " + padv.getName());
            } catch (Exception e) {
                // Error during connection go to next RestoPeer
                System.out.println("RestoPeer may not be there anymore:" +
                                   padv.getName());
                 continue;
            }
        }
    }

    // Send an auction request for French Fries to all the RestoPeer
    // pipes we have successfully connected
    private void sendFriesAuctionRequests() {
        // Enumerate all the RestoPeer pipe connections we have successful
ly
        // connected to
        for (Enumeration en = restoPeerPipes.elements();
            en.hasMoreElements();) {
            OutputPipe op = (OutputPipe) en.nextElement();
            try {
                // Construct the request document
                StructuredDocument request  =
                    StructuredDocumentFactory.newStructuredDocument(mimeTy
pe,
                    "RestoNet:Request");

                // Fill up the Fries auction request argument
                Element re;
                re = request.createElement("Name", myIdentity);
                request.appendChild(re);
                re = request.createElement("Fries", friesRequest);
                request.appendChild(re);

                // Create the pipe message to send
             //   Message msg = pipes.createMessage();
                Message msg = new Message();
```



```
                // Fill the first message element which is the HungryPeer
                // pipe advertisement return address. We need this
                // so RestoPeers can respond to us
                StringMessageElement hpp = new StringMessageElement("Hungry
PeerPipe",myAdv.toString(),null);
              //    msg.addElement(msg.newMessageElement("HungryPeerPipe",
                //           mimeType, myAdv.getDocument(mimeType).getStream
()));
                msg.addMessageElement(hpp);
                // Fill the second message element, which is
                // the fries request. Insert the document
                // in the message
                StringMessageElement req = new StringMessageElement("Request
",request.toString(),null);

              //    msg.addElement(msg.newMessageElement("Request",
                //           mimeType, request.getStream()));
                msg.addMessageElement(req);
                // Send the auction message to the RestoPeer
                op.send(msg);
                System.out.println("Sent Fries Auction Request ("
                        + friesRequest + ") to connected peers");
            } catch (Exception ex) {
                // Error sending auction request
                System.out.println(
                    "Failed to send auction request to RestoPeer");
            }
        }
    }

    // Receive bid requests from RestoPeers on the
    // HungryPeer listening pipe
    private void receiveFriesBids() {
        // Continue until we get all answers
        while (true) {
            Message msg = null;      // Pipe message received
            String price = null;     // Fries price bid
            String brand = null;     // RestoPeer name which offers the bi
d
            String specials = null;  // Specials offer bid
            InputStream ip = null;   // Input stream to read message eleme
nt
            StructuredDocument bid = null; //Bid document received

            try {
                // Wait for a bid message to arrive from a RestoPeer
                // Will block until a message arrive
                msg = myPipe.waitForMessage();

                // Check if the message is valid
                if (msg == null) {
                    if (Thread.interrupted()) {
                        // We have been asked to stop
                        System.out.println(
                            "Abort Receiving bid loop interrupted");
                        myPipe.close(); // Close the Pipe
                        return;
                    }
                }
            } catch (Exception ex) {
                // Error in receiving message
                myPipe.close();
                System.out.println("Abort Receiving Error receiving bids");
                return;
```

```
            }

            // We got a message from a RestoPeer.
            // Extract and display infomation about the bid received.
            try {
                // Extract the Bid document from the message
                ip = msg.getMessageElement("Bid").getStream();
                bid = StructuredDocumentFactory.newStructuredDocument(
                            mimeType, ip);

                // Parse the document to extract bid information
                Enumeration enumeration = bid.getChildren();
                while (enumeration.hasMoreElements()) {
                    Element element = (Element) enumeration.nextElement();
                    String attr = (String) element.getKey();
                    String value = (String) element.getValue();
                    if (attr.equals("Price")) {
                        price = value;
                        continue;
                    }
                    if (attr.equals("Brand")) {
                        brand = value;
                        continue;
                    }
                    if (attr.equals("Specials")) {
                        specials = value;
                        continue;
                    }
                }

                // We got a valid bid. Print it.
                System.out.println("Received Fries Bid from RestoPeers (" +
                           brand + ") at a Price ($" + price +
                           ") \nRestoPeers Special (" + specials + ")");
            } catch (Exception e) {
                // Broken content
                System.out.println("Error extracting bid from the message");
                continue;
            }
        }
    }
}
```