Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (21): 3912-3916, 2009
ISSN 1812-5654

A Sample Chat Application Based on JXTA

R. Mekki and R. Fezza
Department of Computer Science USTO, BP. 1505 El-Mnaouer Oran, Algeria

Abstract: In this study, the application aims use Peer-to-Peer technology to perform bidirectional communication. As the web continues to grow in both content and the number of connected devices, Peer-to-Peer computing is becoming increasingly prevalent. JXTA is a set of open, generalized Peer-to-Peer (P2P) protocols that allow any networked device (sensors, cell phones, PDAs, laptops, workstations, servers and supercomputers) to communicate and collaborate mutually as peers. The JXTA protocols are programming language independent and multiple implementations, also known as bindings, exist for different environments. Their common use of the JXTA protocols means that they are all fully interoperable. Therefore, in this study we introduce the Peer-to-Peer architecture that distribute the resources responsibility among all the connecting peers, the JXTA concepts, protocols and structure, so finally we can be able to implement a chat with sharing files application in JXTA. Unlike other projects that exist in the literature, we use bidirectional pipe that offer a reliable and bidirectional communication between peers.

Key words: Peer to Peer, network, JXTA, protocol, messaging, sharing files

INTRODUCTION

Peer-to-Peer (Amoretti *et al*., 2008) has come to solve some of the client/server architecture problems.

It already exist in the market so many Peer-to-Peer technologies like JINI that promise to interconnect any type of device over any type of network but it's limited on the Java language; Gnutella that provide a communication between peers but it capable of traversing only a single barrier firewall.

Obviously, we needed a more powerful technology, JXTA has brought the solution. JXTA is an open network computing platform designed for Peer-to-Peer (P2P) computing by way of providing the basic building blocks and services required to enable anything anywhere application connectivity.

The JXTA is short hand for juxtapose. It is a recognition that P2P is juxtaposed to client-server or Web-based computing, which is today's traditional distributed computing model.

The JXTA provides a common set of open protocols backed with open source reference implementations for developing Peer-to-Peer applications. The JXTA protocols standardize the manner in which peers:

- Discover each other
- Self-organize into peer groups
- Advertise and discover network resources
- Communicate with each other
- Monitor each other

The JXTA protocols are designed to be independent of programming languages and transport protocols alike. The protocols can be implemented in the Java programming language, C/C++, .NET, Ruby and numerous other languages. Furthermore, they can be implemented on top of TCP/IP, HTTP, Bluetooth and other network transports all the while maintaining global interoperability.

Many projects were created by the JXTA community (Microsystems, 2007). JuMP a management framework for the JXTA platform, built over the manager/agent model. Jxse-rmi, provides the standard familiar RMI interface on top of JXTA, it makes use of JXTA pipes to allow RMI connections between authenticated peers. Jxtacast, broadcast files through propagation pipes. Jxse-cms, a single content management system for JXSE (Ismail *et al*., 2008). Myjxta, Peer-to-Peer instant messenger through unidirectional pipes, it also use the jxse-cms to allow file sharing, etc.

In this study, we created a chat and a sharing files application too, but unlike Myjxta we used bidirectional pipe that offer a reliable and bidirectional communication between peers.

The Peer-to-Peer: It is an architecture that enables any network-aware device to provide services to another network-aware device (Wilson, 2002).

The Peer-to-Peer it is an overlay network, in the Peer-to-Peer network:

Corresponding Author: Rachida Mekki, Department of Computer Science USTO, BP. 1505 El-Mnaouer Oran, Algeria

- A peer is a processor, an application; it can be any connected device
- A peer has a sensitization for the other peers
- The communication between the peers is by the TCP and the HTTP protocols

The most popular Peer-to-Peer applications are: collaboration, games, file sharing, content distribution, grid computing and chat applications in which the peers directly interact with each other we will talk about it more lately.

JXTA: It is an open network computing platform with a set of sex protocols that support Peer-to-Peer.

It is made of three layers, the first one is the core layer it provides essential elements used by the services the second layer that provide network services for the last layer application layer that provide the most common Peer-to-Peer application. (Arora *et al*., 2002).

The goals of JXTA are:

- Programming language independence
- Operating system independence
- Ubiquity

The six JXTA protocols are based on XML message, each protocol is semi-independent of the others and each protocol conversation is divided into two portions, one in the local peer that responsible for the generation and sending the messages, the other one is in the remote peer that responsible for handling the incoming messages and processing it to perform a specific task (Microsystems, 2007).

The six protocols are:

- **Peer Discovery Protocol (PDP):** Used by peers to advertise their own resources and to discover resources from the other peers
- **Peer Information Protocol (PIP):** Used especially by a monitoring peer to get status information about peers (Uptime, state, recent traffic, etc.)
- **Peer Resolver Protocol (PRP):** Used to standardize queries sent on the network, thus it is enable peers to send a generic query to one or more peer and receive it response. It allows peer to exchange any arbitrary information needed
- **Pipe Binding Protocol (PBP):** In the JXTA network peers use pipe for communication, so they can create a new pipe, bind to an existing pipe or unbind from a pipe for those cases we use the PBP
- **Endpoint Routing Protocol (ERP):** Used to rout messages to its destination, it helps peers to know about available routes for sending messages
- **Rendezvous Protocol (RVP):** Used by peers to propagate messages within a peer group

The JXTA peers do not have to implement all these protocols; they only need to implement the ones that they will use.

The JXTA concepts

Peer: Any device in the network that implement one or more JXTA protocol, it could be a PDA, Laptop, PC, cell phone, etc.

A peer is uniquely identify by a Peer ID, it publish one or more network interface peer endpoint to enable multi protocol use, it exist three types of peer: (Traversat *et al*., 2003):

- **The simple peer:** Usually exist inside an internal network, it has the least amount of responsibility in the Peer-to-Peer network, it provide services to the network and consume services providing by other peers
- **The Rendezvous peer:** Usually exist outside a private internal network's firewall, it help peers to discover other resources of the network (peers) either by caching peer advertisement or by forwarding discovery request
- **Router (Relay) peer:** Provides a mechanism to communicate with peers separated from the network by barriers firewall, NAT, peers that are not routers must determine a router peer to use to route their messages

Peer group: A set of peers formed to serve a common interest or goal dictated by the peers involved. It provides services that can be accessible to it members only, it is uniquely identify by the group Id. it's created for reasons of security, scoping or monitoring.

Endpoint: It's the peer network interface (Tcp port + IP address) used to send and receive data.

Advertisement: It describes the network resources such as peers, peer groups, pipes, etc. An advertisement is represented as an XML document; it is used by peers to discover the network resources.

Pipe: Provide an asynchronous, unidirectional, virtual communication channels between two or more endpoints.

JxtaSocket and JxtaBiDipipe: Built on top of pipes, endpoint messengers and the reliability library, provide a bidirectional and reliable communication channels.

Messages: An object that is sent from one endpoint to another over pipe. The peer package the data to transmitted into messages using the output pipe, the peer that receive the messages from an input pipe extract the transmitted data. A message can be an XML format or Binary.

Security: The use of XML messages five us the ability to ass supplement metadata such as credentials, certificates, digests and public keys to a JXTA message that provide it security (Daniel *et al*., 2002).

Implementation of a LAN chat application: At this point, we should be familiar with the basics of JXTA, now we will use some of it protocols and concepts to implement a simple chat application, it is similar to the other instant-messaging applications but it incorporate many of the JXTA protocols.

This application is designed in a LAN; it gives us the capability to register, sign in, search for the connecting peers, exchange messages and to sign out.

The implementation requires to install the Java, JDK in your machine and to download JXTA jxse-lib, tutorial, documents, src (Microsystems, 2007).

The next step is to create the frame using the swing classes for the GUI (Graphics User Interfaces) in which we will invoke the classes that we are going to create.

We create a class StartMe() it job is to create the peer that will be a Rendezvous peer if it's the first one in the NetPeerGroup or a simple peer if we already have a Rendezvous peer for the g roup, then we are going to check the existence of any network configuration to set the peer name, the peer ID and the protocol Tcp.

Once this step has succeeded we invoke the startNetwork() method to launch the JXTA network. As shown in Fig. 1.

Fig. 1: Starting JXTA interface

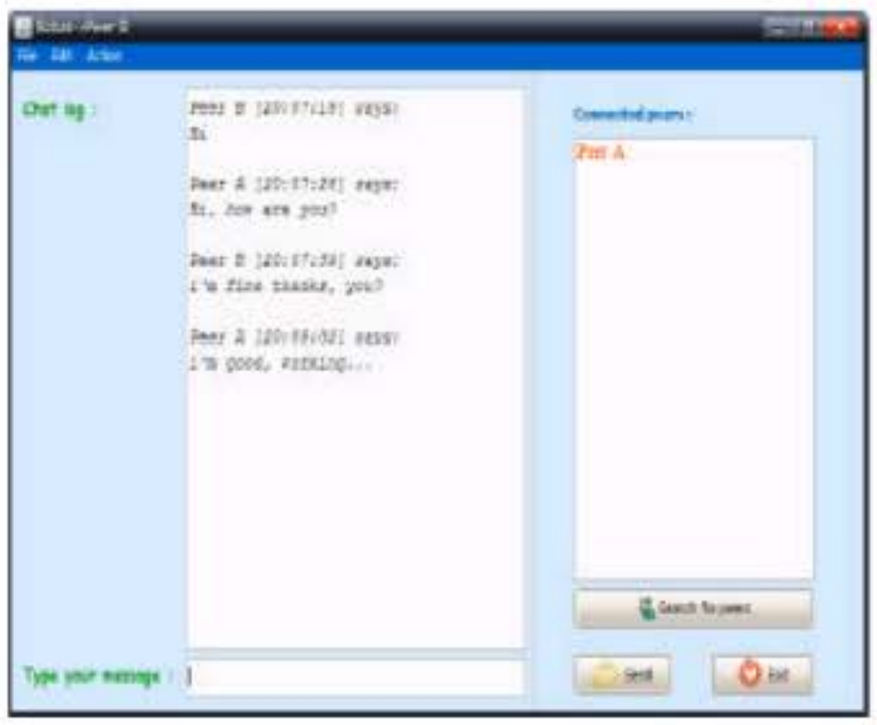

Fig. 2: Chatting interface

```
// Creation of the network manager
 NetworkManager MyNetworkManager = null;
MyNetworkManager = new NetworkManager(
      NetworkManager.ConfigMode.RENDEZVOUS,      Name,
ConfigurationFile.toURI());
// Checking for configuration existence
NetworkConfigurator MyNetworkConfigurator = null;
MyNetworkConfigurator = MyNetworkManager.getConfigurator();
// Setting more configuration
MyNetworkConfigurator.setTcpEnabled(true);
// Setting the Peer Name
MyNetworkConfigurator.setName(name);
// Setting the Peer ID
MyNetworkConfigurator.setPeerID(PID);
// Starting the JXTA network
NetPeerGroup = MyNetworkManager.startNetwork();
```

The class Rdiscovery() it responsibility is to generate the discovery queries and to receive the discovery responses.

```
// generating the discovery queries
TheDiscoveryService = NetPeerGroup.get DiscoveryService();
TheDiscoveryService.getRemoteAdvertisements(null,DiscoveryService.P
EER,null,null,5,this);
/ / We received an advertisement event
public void discoveryEvent(DiscoveryEvent TheDiscoveryEvent){
DiscoveryResponseMsg TheDiscoveryResponseMsg = TheDiscoveryEvent.
getResponse();
```

The last class Rchating() is going to set up a bidirectional and reliable communication between the peers using the JxtaServerPipe that expose a input pipe to process connection requests, whereby the JxtaBiDiPipe bind to respectively to establish private dedicated pipes independent of the connection of the request pipe (Fig. 2). The JxtaBiDiPipe use a messenger instead of an output pipe.

```
// Creating the bidirectional pipe server
MyBiDiPipeServer = new JxtaServerPipe(NetPeerGroup, GetPipeAdvertisement());
MyBiDiPipe = MyBiDiPipeServer .accept();
if (MyBiDiPipe != null){ MyBiDiPipe. setMessageListener(this);}
//Creating a Pipe Advertisement
public static PipeAdvertisement GetPipeAdvertisement() {
PipeAdvertisement MyPipeAdvertisement = (PipeAdvertisement) AdvertisementFactory.newAdvertisement(PipeAdvertisement.getAdvertisementType());
PipeID MyPipeID = IDFactory.newPipeID(PeerGroupID .defaultNetPeerGroupID, Name.getBytes());
return MyPipeAdvertisement; }
/ / Receiving a message
public void pipeMsgEvent(PipeMsgEvent PME) {
Message ReceivedMessage = null;
ReceivedMessage = PME.getMessage();
MessageElement me = ReceivedMessage.getMessageElement ("NameSpace","TheTextElement");}
//Sending a message
public void sendMsg() {
Message MyMessage = null;
MyMessage = new Message();
StringMessageElement sme = new StringMessageElement ("TheTextElement", msg, null);
MyMessage.addMessageElement("NameSpace", sme);
MyBiDiPipe.sendMessage(MyMessage);}
// Creating the Bidirectional pipe in the edge peer
MyBiDiPipe = new JxtaBiDiPipe(NetPeerGroup ,(PipeAdvertisement) Rchating.GetPipeAdvertisement(), 30000, this);
if (MyBiDiPipe.isBound()) {
log.setText(log.getText() + "Bidirectional pipe created!\n");}
```

The class Rsharing() is going to create an instance of CMS (Contreras *et al*., 2003) for the peer group to start the sharing and to process the incoming request for shared files. It will also stop the CMS in case we want to stop sharing our files by calling the method stopApp().

```
//This class will share contents through peers in Group
public class Rsharing extends Thread implements SearchListener
{
/** Creates a new instance of CMS */
public Rsharing(JTextArea log, File givenPath) {
launchCMS();
}
private void launchCMS()
{
//This method will initializie the CMS
cms = new CMS();
cms.init(NetPeerGroup,null,null);
//sharing all files
ContentManager contentManager = null;
contentManager = cms.getContentManager();
File [] list = myPath.listFiles();
Rchecksum checkSum = new Rchecksum();
for(int i=0;i<list.length;i++){
if(list[i].isFile())
{//Sharing Files and check sums in network
contentManager.share(list[i],checkSum.getFileSum(list[i]));
}
}
//viewing the shared contents
Content [] content = cms.getContentManager().getContent();
log.append("[+]All Content are Successfully Shared\n");
}
public void stopCMS()//this method will stop CMS
{
cms.stopApp();
File temp = new File(myPath.getAbsolutePath()+ File.separator +"shares.ser");
if(temp.delete())
{ //also deletes the CMS data file
log.append("[+]File \""+ myPath.getAbsolutePath()+ File.separator + "shares.ser\" successfully deleted.\n"); }
}
//Listener to shows requested queries
public void queryReceived(String query){
log.append("[Query Received]: " + query);
}
```

The class Rsearching() is going to create a listener to process the results, a ListRequestor to initiate the search, a method notify More Results()whenever a peer respond with a list of results and show them in a table (Fig. 3). It will also stop the search process by calling the method cancel() of the ListRequestor.

```
//This class will search for a particle piece of content in the peer group
public class Rsearching extends Thread
{
public Rsearching(String searchKey, JTable table, JLabel label) {
}
public void run() // initiate the search
{ while(true) {
reqestor = new ListRequestor(NetPeerGroup,searchValue, table, label);
reqestor.activateRequest();
}
}
public ContentAdvertisement [] getContentAdvs() // shows found contents of the searching process
{
return reqestor.searchResult;
}
}
class ListRequestor extends CachedListContentRequest
{
public ListRequestor(PeerGroup NetPeerGroup , String SubStr, JTable table, JLabel label){
super(NetPeerGroup,SubStr);}
public void notifyMoreResults() //this method will notify user when new contents are found
{searchResult = getResults();
//showing the results
String [] titles = {"File" , "Length Bytes","Status", "User"};
//add new contents to Search table
DefaultTableModel TableModel1 = new DefaultTableModel(titles, searchResult.length);
table.setModel(TableModel1);
for(int i=0; i < searchResult.length;i++){
table.setValueAt(searchResult[i].getName(),i,0);
table.setValueAt(searchResult[i].getLength(),i,1);
table.setValueAt("Waiting",i,2);
table.setValueAt(pname,i,3);
}
}
public ContentAdvertisement [] getContentAdvs() //return contents
{
return searchResult;
}
```

The class Rdownloading() extends the GetContentRequest is going to create a GetRemoteFile

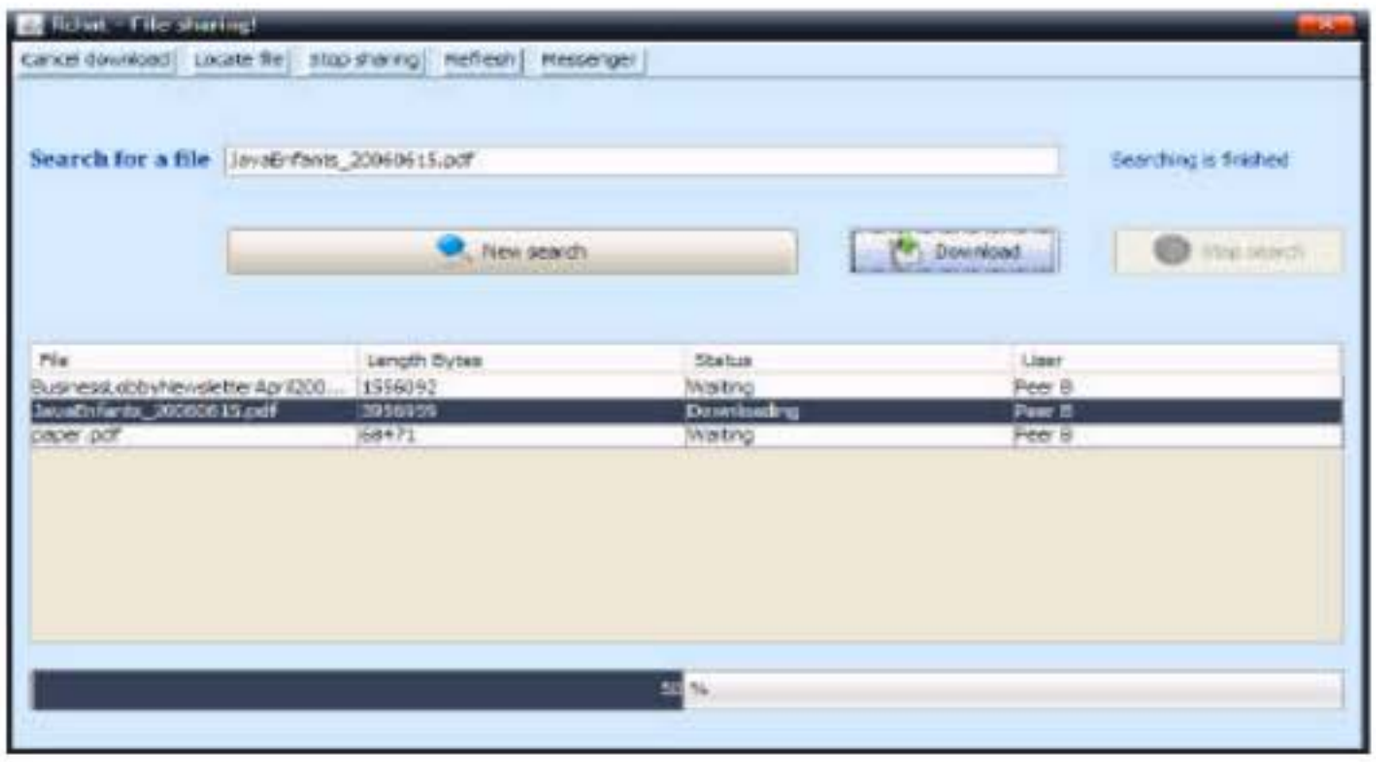

Fig. 3: Sharing interface

class that process the incoming list and handle the process of reconstituting a document from another peer, the method GetRemoteFile() will download the desire file and show us the download progress.

```
//This class will download the selected file and show the download progress in the //progress bar
public class Rdownloading extends Thread
{
public Rdownloading(ContentAdvertisement contentAdv, File destination JTable table, JProgressBar progress)
{
this.NetPeerGroup = StartMe2.getNetPeerGroup();
myDonwloader = new GetRemoteFile(NetPeerGroup, contentAdv, destination, this.table, progress);
}
class GetRemoteFile extends GetContentRequest
{
public GetRemoteFile(PeerGroup group, ContentAdvertisement contentAdv, File destination , JTable table, JProgressBar progress) {
super(group, contentAdv, destination);
}
public void notifyUpdate(int percentage) { //this method will show the download percentage
 progressBar.setValue(percentage);
}
public void notifyDone(){//this method will return message about download process
table.setValueAt("Completed",table.getSelectedRow(),2);
}
public void notifyFailure(){//this method will return message if download failed
table.setValueAt("Failed",table.getSelectedRow(),2);
}
}
```

CONCLUSION

This article has introduced to the Peer-to-Peer world and what has the JXTA platform brought to it, as we showed, an implementation of an instant-messaging application that can be further extended. As long as you go deep in this platform you will find how rich it is.

REFERENCES

Amoretti, M., M. Bisi, M.C. Laghi, F. Zanichelli and G. Conte, 2008. Reputation management service for peer-to-peer enterprise architectures. Proceedings of the E-Business and Telecommunication Networks Third International Conference, Aug. 7-10, Springe, Berlin, pp: 52-63.

Arora, A., C. Haywood and K.S. Pabla, 2002. JXTA for J2ME-Extending the Reach of Wireless with JXTA Technology. Sun Microsystems, UK.

Contreras, P., S. Johnstone, F. Murtag and K. Englmeier, 2003. Distributed multimedia content with P2P JXTA technology. http://www.cs.rhul.ac.uk/home/pedro/papers/HCI-2003.pdf.

Daniel, B., G. Darren, K. Navaneeth and J.C. Soto, 2002. JXTA Java™ P2P Programming. 1st Edn., Sams Publishing, UK.

Ismail, A., M. Merabti, D.L. Jones and S. Sudirman, 2008. The 9th annual postgraduate symposium. Liverpool John Moores University, June 2008.

Microsystems, S., 2007. JXTA java ™ standard edition v2.5: Programmers guide. September 10th, 2007. https://jxta-guide.dev.java.net/files/documents/7150/131364/JXSE_ProgGuide_v2.5.pdf.

Traversat, B., M. Abdelaziz, M. Duigou, J.C. Hugly, E. Pouyoul and B. Yeager, 2003. Project JXTA Virtual Network. Sun Microsystems, UK.

Wilson, B.J., 2002. JXTA. 1st Edn., New Riders Publishing, USA.