Voice Over Peer-to-Peer Telephony Service

Darren Yeung
Honor's Thesis 2005
Boston College
Computer Science Department
Professor Robert Signorile
May 13, 2005

Abstract.

The purpose of this project is to build a telephone service that operates over a peer-to-peer (P2P) network. That is, a service that does not require a centralized server to allow two or more clients to communicate with one another. Rather, each peer is an independent entity with "equal" power and capabilities in terms of using this application. P2P systems allow networks to sustain a lot more peers than current server-client models of networks. Each peer will have capabilities such as the ability to find users connected to the network, accept or reject calls, store peers in a phonebook for future reference, have an answering machine, and record messages during a call.

This has the potential to completely change the business, home, and mobile phone industries. Imagine you have a mobile phone or any mobile device that supports Wi-Fi technology. Whenever you are within range of a wireless hotspot, you are able to call any peer connected to the network, even peer in your phonebook halfway across the country (assuming they are connected to the network). Of course, all of this is free because the fantastic group at Sun Microsystems in charge of Project JXTA created a platform enabling P2P communication with no cost to us, the consumers. Using several protocols defined in JXTA, I was able to establish a completely decentralized telephone network with many functions that are available on current mobile phones. I decided to use JXTA because it provides the basic building blocks to produce peer-to-peer applications.

Introduction.

At the beginning of the semester, I asked myself, "What year-long project can I do that would be interesting, be worth pursuing, and have something to show for at the end of the year?" I knew I wanted to create some sort of application using what I have learned at Boston College and at my summer internships while also teaching myself something new. Increasing popularity with distributed computing and peer-to-peer (P2P) networking as well as Voice over Internet Protocol (VoIP) made me realize, "Why not put them both together to create a voice application using P2P?" That became my goal: to create a telephone service application using peer-to-peer networking. Having taken no prior courses in networking, I knew this was going to be a challenge as well as a learning experience.

First started as a JXTA project that seemed to have been abandoned two and a half years ago, I have adopted the techniques used in that project into my telephone service application called vop2p. The JXTA project stopped short of its intended goal (my current goal), which was to "explore the possibility of creating a completely decentralized telephone network. A phone network without a hierarchy of mega-switches and no requirement for fault-tolerant central offices." To do this, I used Java with the JXTA framework because it contains easily implemental protocols that enable the creation of a peer-to-peer network. After a year-long work, I can say that I have successfully accomplished my mission, and have created a telephone service application that does what many mobile phones do today.

Background.

Historically, telecommunications companies have been using circuit-switched technology to transport telephone calls from one user to another. This technology establishes a "permanent" connection between the calling and the called parties for the entire duration of the call. The problem with this technology is that a significant amount of bandwidth is needed for each call. Moreover, the hardware needed to run circuit-switched networks is very expensive, due in large part to the fact that voice and data services must be carried on different wires. This means that separate hardware is required to provide for two types of transfer. The cost of building and maintaining a circuit-switched network is passed to the consumers in the form of higher telephone service rates.

VoIP, or Voice over Internet Protocol, is a method for taking analog audio signals, like the kind you hear when you talk on the phone, and turning them into digital data that can be transmitted over the Internet. We have to digitize it with an ADC (analog to digital converter), transmit it, and at the end transform it again in analog format with DAC (digital to analog converter) to use it.

This is basically the way VoIP works, sending voice information in digital form in discrete packets or chunks rather than in the traditional circuit-committed protocols of the public switched telephone network (PSTN). Whether the call originated on a PC, a telephone, or an Integrated Access Device (IAD), and whether it is going to be terminated on a PC, telephone, or IAD, determines the type of software and hardware needed to initiate and complete the call. A major advantage of VoIP and Internet telephony is that it avoids the tolls charged by ordinary telephone service.

There are three different kinds of VoIP service used today:

- ATA (analog telephone adaptor) – This is the most simple and common way of using VoIP. The ATA is a box that is connected between a standard phone and your computer or your Internet. The ATA is an analog-to-digital converter, which means it takes the analog signal (i.e. your voice) from your traditional phone and converts it into digital data so that it can be transferred over the Internet to your callee. Vonage, Lingo, and AT&T are such providers that supply ATA boxes with their service.
- IP Phones - These specialized phones look just like ordinary landline phones with the exception of having an RJ-45 Ethernet connector rather than the standard RJ-11 phone connector. Everything is supplied inside the phone such as the hardware and software needed to handle IP calls. Wi-Fi IP phones will be available, allowing subscribing callers to make VoIP calls from any Wi-Fi hot spot, which is one of my future goals for this project. Figure 1 is an example of what these phones look like.

Figure 1.

- Computer-to-computer – This is the easiest way to use VoIP because the only requirements needed to make VoIP work is the software, microphone, speaker, and a high-speed Internet connection. The microphone and speakers usually come standard on today's computers or can be had for very cheap. Software can be downloaded for free or a minimal cost and can be easily installed. There are many advantages of using this technology. Except for the normal monthly ISP fees, there is usually no charge for computer-to-computer calls, regardless of the distance.

Flexibility:

With VoIP, you can make a call from anywhere with a broadband connection. Since the IP phones or ATAs broadcast their information over the Internet, the provider can administer them anywhere there is a connection. This means that anyone can take their phones or ATAs with them on trips and always have access to their home phone. Another alternative is the softphone. A softphone is client software that loads the VoIP service onto your desktop or laptop. As long as you have a headset/microphone, you can place calls from your computer anywhere with a broadband connection. Currently, Vonage and AT&T CallVantage are the only companies offering this capability.

The Advantages:

VoIP technology uses the Internet's packet-switching capabilities to provide phone service. VoIP has several advantages over circuit switching. For example, packet switching allows several telephone calls to occupy the amount of space occupied by only one in a circuit-switched network. Using PSTN, a 10-minute phone call consumes 10 full minutes of transmission time at a cost of 128 Kbps. With VoIP, that same call may have

occupied only 3.5 minutes of transmission time at a cost of 64 Kbps, leaving another 64 Kbps free for those 3.5 minutes, plus an additional 128 Kbps for the remaining 6.5 minutes.[1] Data compression further decreases the size of each call, so it is possible to fit more calls meaning that more peers can connect with you at the same time.

For users who choose to use VoIP with analog telephone adaptors, their ATAs can travel with them if he/she decides to leave home. Say you live in California and you have a subscription with Vonage. If you decide to travel to Boston for school, business, or vacation, you can simply take your ATA with you and plug it into your school or hotel's broadband connection to make and receive calls using your home phone number. This is equivalent to having a cellular phone, but with more power and features.

The Disadvantages:

One of the major drawbacks of current VoIP services is the issue of public safety. The 911 telephone infrastructure is simply not supported by current VoIP providers. It is difficult to locate the caller because since the IP phone is transportable, the phone number does not supply 911 with sufficient information about where the caller is. Here we see a clash between old and new technologies. VoIP is growing too fast for other parts of the industry to catch up. The need for police to know the contact number and location of the caller is essential. The advertisement above from Vonage is a way around the problem. The problem is placed on the 911 callers by

[1] Tyson, Pg. 7.

requiring them to give their physical location. Better hardware and software implementations of the system could solve the problem.

Traditional telephone technology still beats VoIP in emergency situations. When you dial 911 from a standard landline, your call travels to a local switching station, which then bounces to a dedicated network reserved only for 911 calls. This special emergency circuit also links the call to the "Automatic Number Identification/Automatic Location Identification" database of phone numbers, names, and addresses. Plugging VoIP into this system is not easy by any means.

Another major disadvantage is the inability to use your IP phone or ATA during a power outage. Traditional phones get their power from the telephone line, so even during a power outage, it is still possible to make and receive calls. Power outages or any natural disaster that disrupts power is when the need to make calls is necessary. The only way around this is to have an uninterruptible power supply (UPS), but that still is only temporary and it won't be implemented by many people due to its cost.

Trends:

Canada and US Combined Residential VoIP Subscribers and Revenues			
Year	2003	2004	2005
Subscribers	100,000	1,300,000	4,500,000
Revenues	$34 million	$390 million	$1.35 billion

VoIP has gained wide scale entry into mainstream network applications. Enterprises have begun to trust vendor claims that VoIP offers advanced voice and data convergence features, in addition to dirt-cheap calling rates. According to Distributed Networking Associates, for example, this year marks the point at which more enterprises in the U.S. will use VoIP versus traditional telephony. Over 85% of enterprises either

already have IP telephony or plan to install a VoIP system, Distributed Network says. By 2007, about 19% of all business calls will be made through a VoIP network, according to In-Stat/MDR.

For cable operators, the VoIP sector will account for 4.2 million new subscribers per year, which will generate $5.4 billion in sales for cable system operators by the end of 2008, according to analyst firm Kagan World Media. The table above shows the trends of VoIP during the previous two years. Traditional telephony providers and even DSL service providers are also looking at more than 50% VoIP growth rates per year, several analyst firms have reported.[2]

What is JXTA?

As more and more computers are connecting to the Internet through high-speed means, servers in a server-client model of a network simply cannot provide enough resources for the demands of its increasing client membership. This will overload the server with requests and will eventually shut down the network due to the traffic.

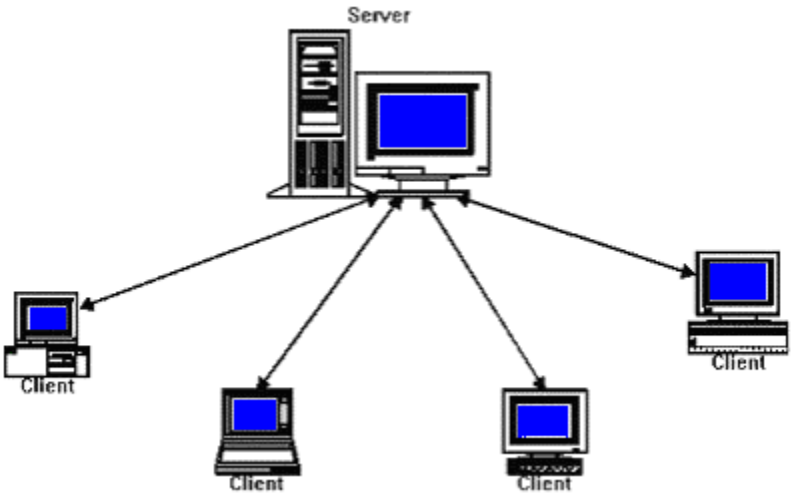

Figure 2.

[2] Gain, pg. 1.

Another problem with centralized servers is that whenever there is a problem with the server (shutdown, freeze), the network directed by that server suddenly stops operating. This dependence in a server is a bottleneck in the network yielding an unreliable network. A peer-to-peer network, however, allows peers to connect with each other in a decentralized fashion without having a server direct the connections between clients.

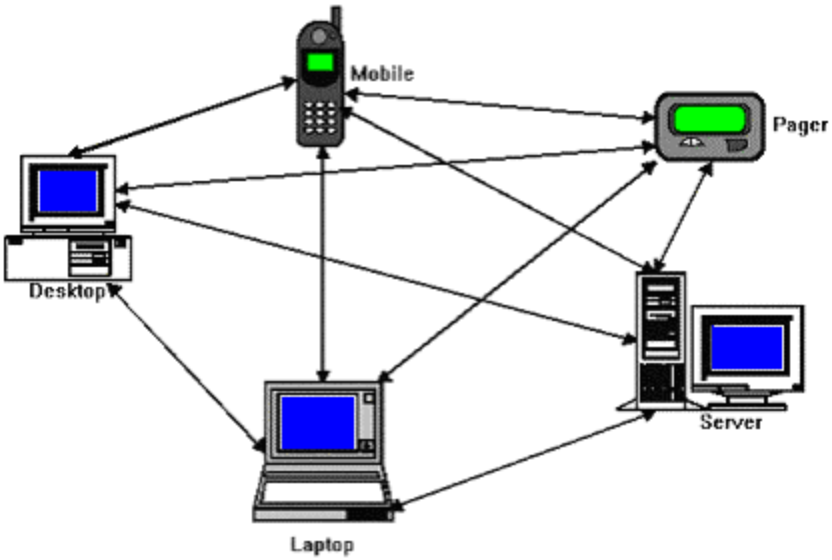

Figure 3.

In Figure 3, suppose the mobile phone is communicating with the laptop through the desktop. If the desktop to laptop connection suddenly fails, then another route can be found such as the desktop redirecting the call though the server and finally to the laptop. Searching is done non-deterministically, so a route may not be the best route. But in order for this network to work on a global scale, a common, general framework must be developed so that applications can communicate with other applications through common protocols and utilities. JXTA is a framework that enables peer-to-peer networks to exist.

JXTA has a set of standards for many different programming languages to support peer-to-peer applications. It provides an infrastructure and specific services that cater to the creation of P2P applications. Having to write one's own protocols (in a specific

language) to support a peer-to-peer network considerably restricts the range of its users. Only peers that utilize the protocols and are written in the same programming language can communicate each other. JXTA is a common platform that allows easy communication between different programming languages and applications.

JXTA is made up of three distinct layers: core, services, and applications. Figure 4 shows how the layers interact with one another.

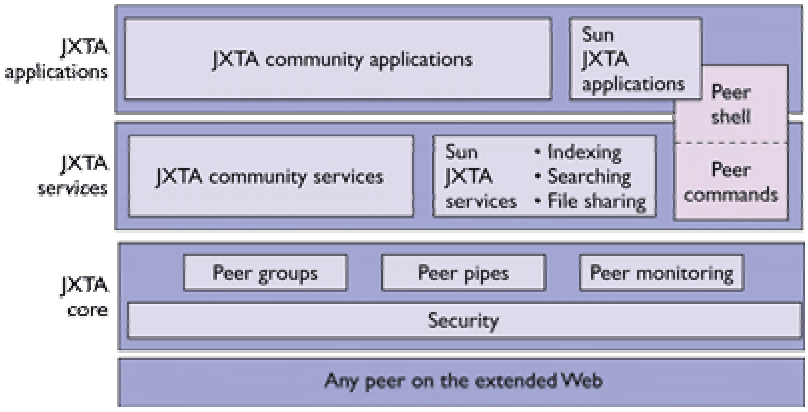

Figure 4.

The core layer provides primitive functions for peer groups, peer pipes, and monitoring peers. The peers can be any devices (computer, PDA, mobile phone) that can setup communication with other devices for sending messages back and forth. The peer groups contain mechanisms for joining peers to groups or removing them. The peer pipes allow communication between peers for transferring data, content, and code in a protocol-independent manner. The monitoring peers control the behavior and activities for each peer. Besides support for the primitive functions, the core layer provides mechanisms for security. Its security policies are defined in the higher level layers depending on the communication environment.

The service layer provides higher-level functionality using the primitive functions from the core layer. In the JXTA architecture, all peers automatically join a default

group called the NetPeerGroup. This peer group provides basic services, but it is not required for all peers to be a part of this large group. Peers can join smaller groups, and interact with only the peers who are members of that group. Also, custom services can be created to support specific functionalities. The applications layer uses components in the core and service layers to develop P2P applications.

The application layer uses components from the core and service layers to employ P2P applications such as Gnutella and Napster. This is the layer where individual peers are able to interact with one another.

In addition to the layered architecture, JXTA defines protocols, messages, pipes, advertisements, and peer groups. The JXTA platform defines a set of XML protocols designed to address the common functionality required to enable peers to form dynamic networks, independent of the operating system, development language, and network transport.[3] They are a standard format for representing and exchanging data between peers. The protocols can be implemented using many languages (Java, C/C++, Perl, etc.) allowing independent devices to exist and communicate with one another in a peer-to-peer system. The protocols are:

- Peer Resolver Protocol – used to send a search query to peers with the response (e.g. found or not found) as the result.
- Peer Discovery Protocol – allows a peer to discover peer, group, service, or pipe advertisements in the network. It is used to locate content.
- Peer Information Protocol – used to obtain peer status information such as its state, uptime, and capabilities.

[3] Flenner, pg. 93.

- Peer Membership Protocol – allows a peer to join or leave a peer group. It also supports the authentication and authorization of peers into peer groups. The protocol has three key advertisements for authorization, and the credential. The credential created in this protocol is used as proof that the peer is a valid member of the group.[4]
- Peer Endpoint Protocol – used to find a route from one peer to another. It creates routes using routers, rendezvous, and gateways to route messages to another peer.
- Pipe Binding Protocol – used to bind a communication path between endpoint peers, i.e. a physical pipe endpoint to a physical peer. It is primarily concerned with connecting peers via the route(s) supplied by the Peer Endpoint Protocol.
- Rendezvous Protocol – used to propagate messages in the network. It defines a base protocol for peers to send and receive messages within the group of peers and to control how messages are propagated.

Most of the above protocols are independent of one another (the pipe binding protocol works in conjunction with the peer endpoint protocol), and a peer does not have to implement all of the protocols. It depends on what purpose the peer serves. For example, if a peer decides to just share files, it can implement the peer discovery, pipe binding, and peer endpoint protocols. These protocols are sufficient for peers to exist in a decentralized peer-to-peer environment.

A pipe is a virtual connection between peers. They are the main transport for messages that flow through a JXTA application. A few types of pipes currently implemented are:

- Uni-directional asynchronous – one-way pipe.

[4] Brookshier, pg. 57.

- Unicast reliable secure pipe – all messages sent will receive a return message of acknowledgement. The data is encrypted both ways.
- Bidirectional – essentially two uni-directional pipes.

Pipes can be addressed in two different types of ways: point-to-point or propagate pipes. Point-to-point pipes connect two different peers while propagate pipes connect a peer to any number of endpoints in the communication.

JXTA uses a new concept in peer-to-peer communication called advertisements. Advertisements are structured XML documents that describe services and information available in the JXTA network. It names, describes, and publishes the existence of a resource, such as a peer, peer group, pipe, or service. It generally consists of a resource's unique JXTA ID, name, type (UnicastType, UnicastSecureType, and PropagateType), and credentials. The JXTA network uses advertisements to locate resources for the desired search query. A peer that decides to advertise himself creates an advertisement and places it in the network allowing other peers to communicate with or find him. Advertisements are found using the peer discovery protocol.

The following is an example of an XML pipe advertisement:

```
<?xml version="1.0" encoding=UTF-8"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta=http://jxta.org>
        <Id>
                urn:jxta:uuid-59616261646162614E5047205032503357EF9360DC8147
                        D4A99EFD364A364ACD04
        </Id>
        <Type>
                JxtaUnicast
        </Type>
        <Name>
                yeungd
        </Name>
</jxta:PipeAdvertisement>
```

Figure 5.

From the advertisement in Figure 5, you can determine that the name of the user is yeungd, the type of the pipe is unicast, and its unique user ID (UUID). From this information, it is possible to connect with yeungd and begin exchanging data.

There are three different kinds of peers: rendezvous, router, and gateway. A rendezvous peer is a peer that processes queries from other peers. It can also send queries to other rendezvous peers. The reason for a rendezvous peer is to allow for the searching of advertisements beyond a peer's local area network. These peers must be able to store large amounts of information about the peers it can react with. That is because it must be able to cache each search query so that when it sees the same query in the future, it can easily find the peer who has or knows another peer who has the required data. A rendezvous peer can also act as a relay for searches if it does not have the requested information by forwarding discovery requests to other rendezvous peers.

A typical search for some data in a JXTA network is illustrated in Figure 6 below. Peer 1 requests data, which is located at Peer 5. The search starts first with Peer 1 sending search queries to his local peers (2 and 3) on the LAN. If these peers do not have the requested data, Peer 1 searches for rendezvous peers. Using a gateway (not shown), Peer 1 finds Peer 4 and propagates the requested message to him who continues to propagate the message to his peers. The data is finally found in Peer 5 and the route is routed back to Peer 1 with a "found" response.

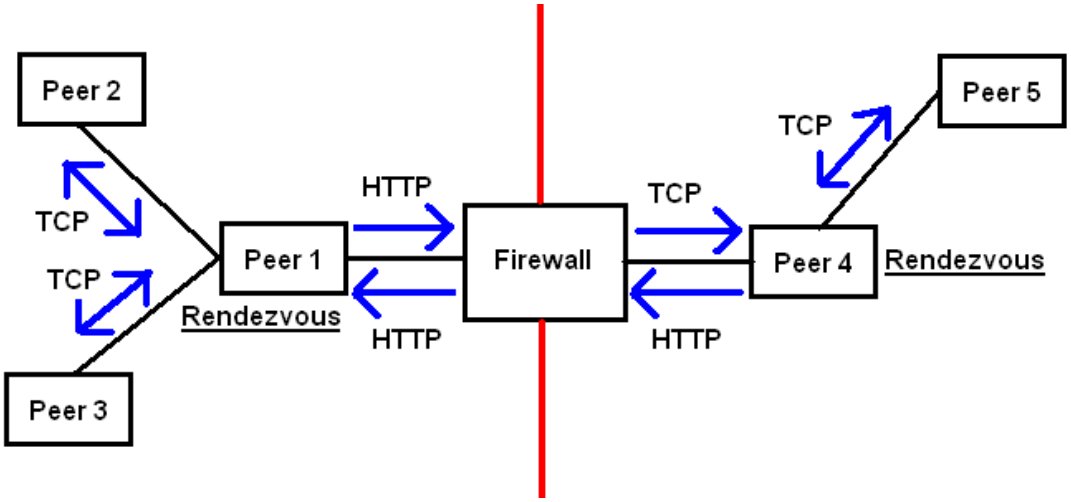

Figure 6.

A router peer is any peer that supports the Peer Endpoint Protocol. It is not required for all peers to implement the endpoint protocol because only a few are needed to support a large network. Figure 7 is an example of how a route is created. The request for a route starts at Peer 1 with a request for a route passed to available router peers until a complete path to Peer 5 is built. A router can also be a gateway, so Peer 6 is included in the route. Like rendezvous peers, router peers can support caching or complex algorithms for searching the network (depending on what the topology is). Peers are searched non-deterministically, so the created route may not be the best route.

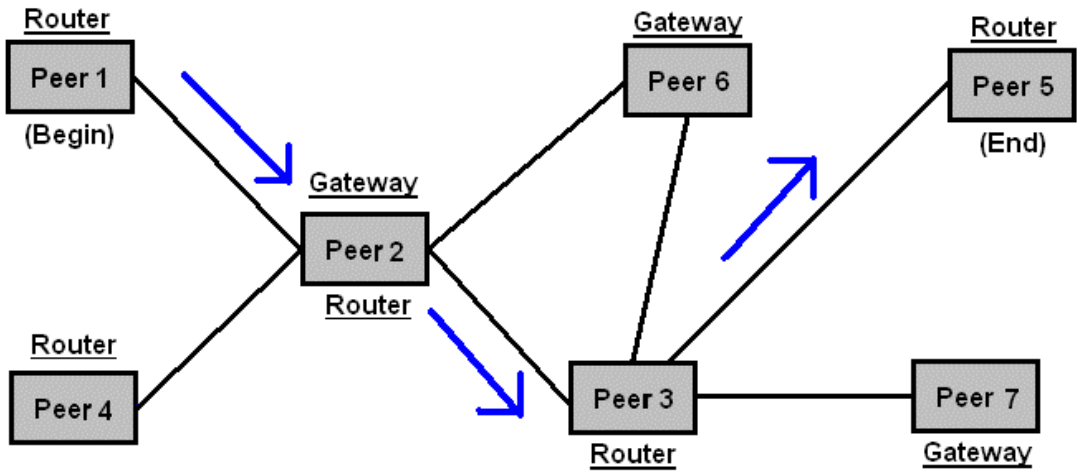

Figure 7.

A gateway peer acts as a communications relay. It is used to relay messages between peers. It does not do any requests like rendezvous peers, but is configured to act as a router or gateway for peers that use NAT addresses, are behind a firewall, or use proxy servers. They are essential to getting around most of the security on the Internet. Once executed, the gateway peer will start the appropriate gateway services and contact the destination rendezvous peer to make sure it exists on the network. A new pipe is bound, so a rendezvous peer inside a firewall is able to query other rendezvous peers outside the firewall. Figure 8 below shows how the gateway Peer 2 is used to interface between Peer 1 and Peer 3. The gateway translates HTTP messages from Peer 1 to TCP for delivery to Peer 3. When messages are sent from Peer 3, they are sent via TCP to Peer 2, which holds the message until Peer 1 makes an HTTP request to retrieve the data.[5]

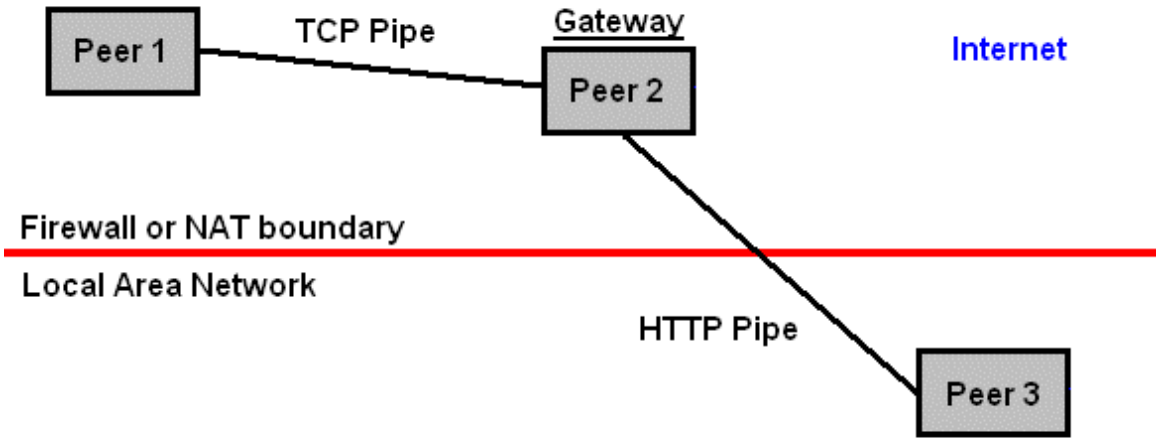

Figure 8.

JXTA is a good choice for creating a heterogeneous network between computers since XML is the primary means for locating services. Whether a peer is behind a

[5] Brookshier, pg. 53.

firewall or a NAT address, the JXTA protocol allows interaction between any peer(s) implementing the framework. Peers in a P2P system are considered equals in terms of functional capabilities. This equality means the need for an intermediary to help a user participate in a network is not necessary. Once a peer is connected to a network, he/she can get involved. Peers in a peer-to-peer system have the capability to provide and consume services; therefore a peer can be considered a client when it is requesting a service, and can be considered a server when it is providing a service.

The Program.

Test Environment:

Two laptops acting as peers on the same subnet in 66 Commonwealth Avenue. One laptop named "yeungd" is an IBM Thinkpad X20 running on Windows 2000 Pro. The other named "bamboo-w" is a Sony Vaio R505 running on Windows XP Pro. Both machines are running Java Development Kit (JDK) 1.4.2 and JXTA J2SE 2.3.1. Fulton Hall became another testing ground for testing peers on different subnets since the wireless network and the LAN connection run on different subnets.

Vop2p is a telephone application that allows communication between peers through a peers-to-peer network using the JXTA framework, which has protocols that allow P2P networking. Voice is encoded in GSM format (the same format used in modern-day cellular phones on the GSM network) and is sent from one peer to another peer using JXTA unicast pipes. Once the package arrives at its destination, it is decoded and played on the peer's speakers.

Voice encoding:

GSM 06.10 encoding is used in this project because it provides a digital encoding of voice, so it can be easily transferred between computers. It compresses 160 13-bit samples into 264 GSM frame (or 33 bytes), i.e. 1650 bytes/sec (at 8000 samples/sec) since the data rate is 13 kbps. It results in very good compression with good quality output but is very costly in terms of performance.[6] Samples are taken every second of compression in this program and are stored in an array of size 1650, which is then sent into the JXTA pipe to be decoded by the peer.

GSM is a "lossy compression," which means that it eliminates "unnecessary" bits of information making the file smaller. In this case, the original recorded voice cannot be recreated after the compression. More sophisticated hardware and software will be needed to recreate the sound of real voice, which is inherently analog.

Once the vop2p program is run, a daemon thread is created, which starts the peer in the idle state. During this state, the user is able to make calls, receive calls, set his availability status, and listen to his voice messages. Figure 9 shows what the idle state looks like. The status bar is located at the bottom of the GUI and the peer's username is located in the title.

[6] Bagwell, pg. 8.

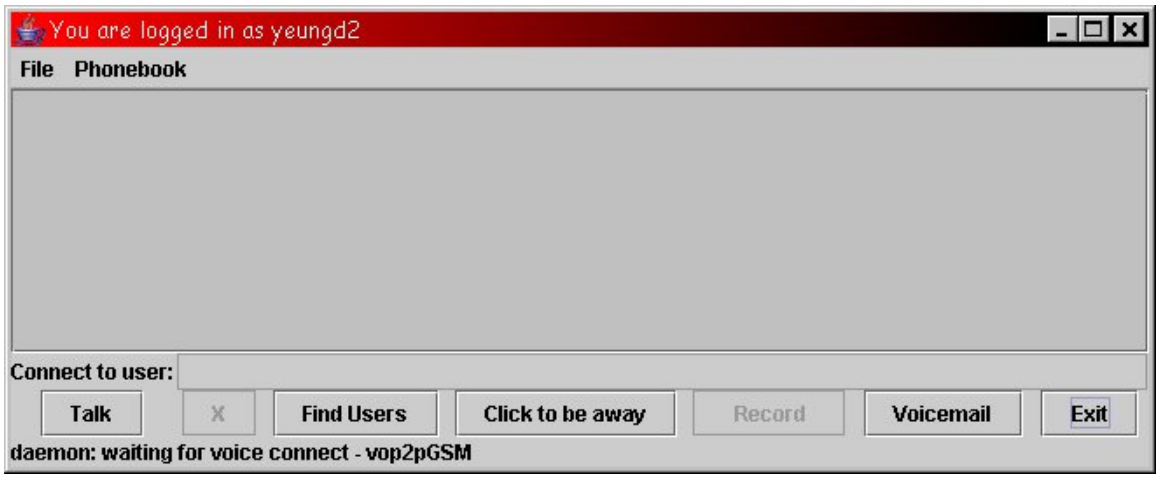

Figure 9.

If a peer would like to make a call, she has two options for finding a peer: 1) on the network using the "Find Users" button and selecting a peer from the list, or 2) connect with a peer already saved in her phonebook.

If Peer 1 want to talk to Peer 2, he needs to click on the name of the peer and then click on the "Talk" button. When the "Talk" button is clicked, the Peer 2's pipe advertisement is retrieved and a thread is created. The new thread opens a record line to record the voice coming from the microphone to a buffer of bytes. It creates an output pipe, which binds to the input pipe of the Peer 2 to send the GSM compressed voice through the pipe to the destination peer. Inside the first buffer of data that Peer 1 sends is the caller's username, which the receiving peer will use to automatically connect with the caller. In essence, a "bi-directional" pipe is being created. Figure 10 is the pseudo code for sending data and Figure 11 is the pseudo code for receiving data.

```
while( current thread not interrupted and
        complete = false ) {
    data = recorded audio bytes;
    numBytesRead = # of byte read;
    if( numBytesRead = = -1 )
        complete = true;
    else
        write data to output stream;
}
```

Figure 10.

```
while( current thread not interrupted and
        complete = false ) {
    data = read input stream bytes;
    numBytesRead = # of bytes read;
    if( numBytesRead = = -1 )
        complete = true;
    else
        write data to playback line;
}
```

Figure 11

Figure 10 resides in Peer 1's sending thread. It stores the recorded audio bytes to data, which is an array of 1650 bytes. If there are still bytes being read, it continues to write the buffer into the output pipe. Figure 11 is run inside Peer 2 when she has detected an incoming voice (and has accepted it). The buffers in the input stream are the buffers that Peer 1 is sending. As long as Peer 2 is still receiving bytes, it writes the data to the computer's speakers.

Features:

VOP2P is jam-packed with features. Many of the features found in present-day cellular phones are in this program. There's no point of having a telephone service without any features. They are:

Voicemail/Answering Machine:

The answering machine message is stored in a file which is accessed and sent over the pipe to the calling peer whenever you are away or reject a call. After the message is sent, the callee waits 20 seconds for the caller to leave a message. Afterwards the pipe is disconnected and both applications (from caller and callee) are sent back to the idle state. To access or listen to these messages, click on the "Voicemail" button.

Voice record:

During a call, a peer is able to record as many messages as he/she chooses. A peer simply clicks on the "Record" button to start recording, then clicks the "Stop Recording" button to stop recording. All recordings are saved as *.dat, which can only be read and interpreted by VOP2P. Clicking on the "Voicemail" button allows you to listen to these recordings since answering machine and recorded messages are all stored in the same directory.

Away/busy:

This allows a peer to set its status so that whenever another peer tries to call her, the caller is directed immediately to the answering machine. To be away, click on the "Click to be away" button. To return, click on the "Click to return" button.

Incoming call notice:

This allows a peer to decide whether he wants to accept or reject the peer trying to call him. If he rejects, then the caller is sent to the answering machine. There are no messages sent back to the caller except that the callee is not available. This enables users to screen their calls and to choose who he/she want to talk to.

Find users:

This feature allows peers to search the local network to find peers to connect to in JXTA network. After this button is pressed, the JXTA Discovery Service searches for local advertisements and returns a list of available peers.

Phonebook:

Rather than searching for peers each time you join the network, which takes about 10 seconds, you are able to simply call the peer from your phonebook. The phonebook is

accessed through the menu bar. You are only allowed to add peers from the list generated by clicking on the "Find Users" button.

Timing is pertinent for VOP2P to hang-up properly (at least in its current version). During a conversation between two peers, four threads exist between the two peers: two to receive data and another two to send data (see Figure 12).

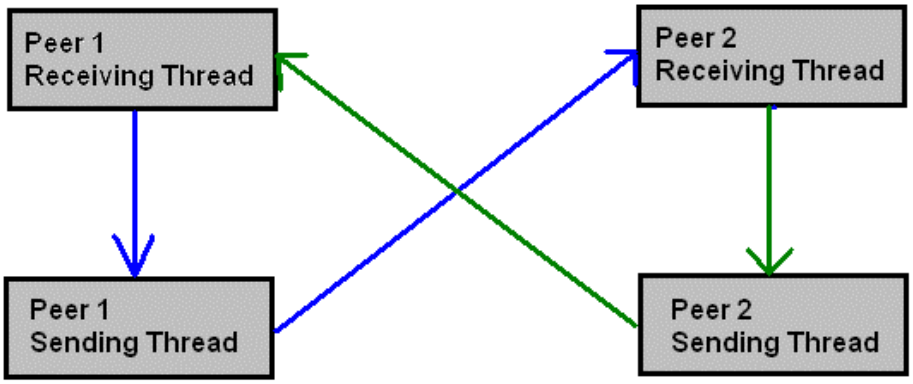

Figure 12.

Say Peer 1 and Peer 2 are interacting with each other and exchanging a conversation. In a perfect world if Peer 1 wants to end the call, he would simply click on the hang-up button; both peers will close their sending threads and be in their idle state. But with lag time, Peer 1 cannot return to the idle state until Peer 2 does; otherwise Peer 1 will detect an incoming call since the sending thread of Peer 2 is still sending data to Peer 1. This makes sense because Peer 2 does not know that she disconnected with Peer 1 when the "hang-up" button is clicked by Peer 1.

This synchronization issue became a big problem in this project because the voice delay ranged from 6 to 10 seconds, which led to severe consequences on hang-up. My solution was to make Peer 1 (who ended the call) wait for about 15 seconds giving Peer 2 time to find out that Peer 1 hung up, close his sending thread, and return to the idle state.

A better alternative would be to be independent of timing. I attempted to pass the hang-up notice on the pipe once the "hang-up" button has been clicked, but somehow the message was not getting across to the receiving peer. I tried putting the string "QUIT NOW" (converted to bytes) as the first ten bytes of the sending buffer, but when I had the receiving peer print out the first ten bytes of its buffers (converted to a string, of course), all I got was garbage. If I got this to work, the hang-up would be a lot safer.

Possible causes of the delay include voice compression and decompression or the overhead of sending and receiving data (including sending and receiving across multiple peers). Since I am taking a sample or duration of compressed voice every second and sending it, there is already the initial time to compress the required 1650 bytes, which adds to the delay. Also, the buffer size used after the compression is 1650 bytes, which is huge. By making it smaller to around $1/20^{th}$ of 1650, the delay should improve. Also, the distance the chunks must travel contributes to the lag time. If the chunks must travel across many peers, the delay time would increase. Therefore the topology of the network is important, which is dependent on the number of rendezvous and router peers to direct transmissions.

Future.

Unfortunately, I was not able to test my program with one peer inside and another outside a firewall simply because the only resources I had lie within the confines of Boston College. So the essence of peer-to-peer networking was not fully taken advantage of. With unlimited time to work on this project, I would create a gateway peer within the Boston College firewall and attempt to connect with it through HTTP tunneling from the outside.

If I had more time to work on this project, I would attempt to diagnose and fix the lag problem because one of the biggest appeal of online telephone services is the amount of (or lack of) delay between messages. The more "real-time" the conversation can be, the more it will be accepted and used. I also plan to add several more features, expand existing ones, and increase the overall stability of the program. Not having to rely on timing will considerably stabilize the program.

Some features I would add include call-waiting and conference calling. In order for call-waiting to work, it would be necessary to put a peer on hold. That is, temporarily stop the transmission between one peer and begin another transmission with another peer. An approach would be instead of sending recorded voice, send "silence" or music from a special file for this purpose. Timing would be a problem here because if not timed correctly, a call-waiting can become a conference call. If timing were not an issue, then the called peer who is already in a conversation can send to his connected peer a hold message. Then the called peer can communicate with the calling peer without worrying about the holding peer listening in on the conversation.

The voicemail/answering machine feature would be expanded allowing custom away messages as well as the ability to manage one's mailbox. Once a peer begins listening to his/her voicemail, I imagine a pop-up window letting the user know information about the current message such as the time of call and the peer who left the message with the option of saving or deleting the message.

My future goal is to implement this application on a Wi-Fi or 802.11 enabled device with a subscription to T-Mobile HotSpot. For those who are not familiar with T-Mobile HotSpot, it provides high-speed Wi-Fi wireless Internet service in convenient

public locations. The most popular locations are in the airport and Starbucks. It would be very interesting to be able to connect with peers in a remote location to really put my program to the test. Another feat would be to actually put this application into a device using J2ME.

Conclusion.

Overall, I would consider my project a success. Not only was I able to implement a peer-to-peer network using JXTA and send voice between two computers, I was able to add workable features. While there are still many improvements that can be made to replicate a telephone service (i.e. voice latency across the pipe), the basics have been done. My goal of creating a completely decentralized telephone network has been achieved.

The first thing I noticed when I was able to send chunks or voice between two peers was how clear the sound came through the speakers once it was decoded. It did not have a lot of static or fuzziness, which was what I expected. Even though I used GSM 06.10 encoding, I thought the hardware differences would decrease the quality of the voice, but this wasn't the case. Sound was clear and crisp. The GSM encoding was powerful enough to capture many of the background noises clearly. Although this may or may not be desirable, it shows the effectiveness of the encoding algorithm.

Seeing how easy voice was being sent through the pipe to a peer, I discovered how easy it will be to send messages as well. Since voice is simply stored in an array of bytes, I can simply do the same thing with text. The problem with this is determining how the receiving peer will know whether the data received is text or voice. I made it

explicit in my program that the first 10 bits of the first chunk be the name of the caller. This makes it possible for the receiving peer to know who the caller is.

In order for two peers on different subnets to connect with each other, it is necessary to change the parameters in the JXTA configurator. This requires adding the peers IP address into the "Public address" field in the Advanced tab. The "Enable Incoming Connections" checkbox must be checked.

The Java programming experience I gathered here at Boston College as well as my programming projects in my summer internships contributed greatly to the success of this program. I would like to thank all of my Computer Science teachers who have taught and gave me the skills needed to turn an idea into a reality. They are: Prof. Sergio Alvarez, Prof. William Ames, Prof. Robert Muller, Prof. Amitabha Roy, Prof. Howard Straubing, and finally, my advisor, Prof. Robert Signorile, who contributed most to my project. Special thanks go to my parents who have supported and believed in me to succeed and do well.

References.

1. Brookshier, Govoni, Krishnan. JXTA: Java P2P Programming. Indianapolis, IN: Sams Publishing, 2002.

2. Bagwell, Chris. "GSM 06.10 lossy speech compression." November 14, 1998. May 3, 2005. <http://kbs.cs.tu-berlin.de/~jutta/toast.html>.

3. Cognigen Networks, Inc. "PC to Phone: VOIP: Internet Telephony." Mountlake Terrace, WA: Cognigen Networks, Inc., 2005. 29 April 2005. < http://ld.net/products/?cogid=la102014&page=Technology&product=delta3>.

4. CollabNet, Inc. "VoP2P Project Home." JXTA: Get Connected. 2003. May 4, 2005. <http://vop2p.jxta.org>.

5. Flenner, Robert, et al. Java P2P: Unleashed. Indianapolis, IN: Sams Publishing, 2003.

6. Gain, Bruce. "The Year of VoIP." Tom's Hardware Guide. 6 May 2004. 4 May 2005 <http://www4.tomshardware.com/business/20040506/voip-01.html>.

7. Tyson, Jeff, and Robert Valdes. How VoIP Works. HowStuffWorks, Inc. 20 Apr. 2005 <http://computer.howstuffworks.com/ip-telephony.htm>.