TECHNOLOGY WHITE PAPER

Overcoming Mobile Fragmentation:

Building Applications for Multiple Operating Systems

There is a mobile war brewing among industry heavyweights that is pitting Android (Google), against Windows (Microsoft), Symbian (Nokia) against iPhone (Apple). While on the surface this stiff competition appears to benefits consumers, for developers it's a growing nightmare to build applications that run on these different operating systems and try to maintain some degree of compatibility and interoperability. The problem compounds when you consider applications should ideally span not only phones, but MIDs, UMPCs, cars (telematics) and other embedded and wearable computing devices. Pervasive is the new Mobile. A single platform that spans all 'Screens of Life' would truly be something to behold.

This paper discusses some of the challenges and trade-offs developers must make and highlights one solution to overcoming them.

Sponsored by

RECURSION SOFTWARE, Inc.

Confronting Mobile Fragmentation

In the midst of iPhone mania, Android frenzy, and Symbian going open-source, there has been much talk about growing fractures and compatibility chaos within mobile landscape. This growing storm affects not only from a software developers, but also hardware manufacturers as well. With well over 30 operating systems for mobile and embedded devices, developers must carefully choose which devices they want to target and which devices, along with their customer base, they must leave behind. Just listing the leading Smartphone software stacks to consider only emphasizes the problem: Nokia's Symbian (JME and C++), RIM's Blackberry, Microsoft's Windows Mobile, LIMO's Embedded Linux, Apple's iPhone, Palm, and a host of legacy proprietary phone operating systems (Figure 1). Others of note include: Qualcomm's BREW, Open Handset Alliance's (Google) Android, Sprint's Titan, Sun's JavaFX, and Neo's OpenMoko.

This cross-platform headache holds true not only for user-interface development, but also for writing and accessing services and intelligent information on edge devices. These services, accessible on any node will open the door to advanced real-time, location-aware applications that are currently only envisioned, but not built.

Standards and performance also are key issues. Once you get to the mobile and embedded arena there are no multi-language standards for database access, rules integration, or location determination. Organizations would love to see a consolidation of these options, but the number of devices and platforms is only growing, not shrinking. Additionally, employees and customers don't want to be told what Smartphone and other devices they can and can't use. They also increasingly want to have access to the applications and information regardless of which 'screen' (phone, car, MID, PC, TV) they are using at any given moment in time.

Worldwide: Smartphone Sales to End Users, Shares by Operating System, 2Q08

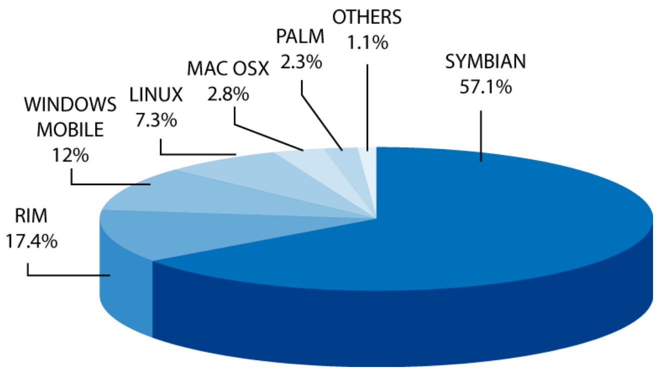


Figure 1
[Source: Gartner 2008[1]]

Friendly Partners Or Strange Bedfellows?

The intense competition that is occurring amongst handset manufacturers, mobile OS vendors, application companies, and network providers make the mobile environment even more complicated. Until recently these groups competed within their own industry, but the lines between these industries are beginning to blur.

The handset manufacturers must decide what mobile OS's to ship on their phones. They also have to find ways to cut prices and differentiate their devices from the myriad of outstanding choices in the market today, which is growing daily. Furthermore, most manufacturers have ancillary hardware lines, such as set-top boxes and MID devices that they would like to manage easily and unify via pervasive applications.

The mobile OS providers have to decide whether or not to open-source their software, a trend that arguably Google/Open Handheld Alliance (OHA) put into high gear with the Android release. This move likely motivated Nokia's recent purchase and eminent open source release of the Symbian operating system. The mobile OS providers also must decide what additional features/capabilities to

[1] Data provided by *Gartner Says Worldwide Smartphone Sales Grew 16 Per Cent in Second Quarter of 2008*, September 8, 2008. Graphic created by Recursion Software.

provide.   This is a tough segment to generate meaningful revenue, which is why the movement to open-source and/or management of OS's by committee will continue. Network carriers have their own headaches. What phone manufacturers and subsequently mobile OSs do they allow to run on their Telco networks? How do they differentiate their Telco networks from the others? How will they compete against WiFi, WiMax and the recently auctioned wireless spectrums that must be free and open to any application?

The reality is that the traditional products that each of these groups provides are being commoditized and that trend will only continue. So what should they do to remain relevant? What can application developers do to future-proof their roadmap?

"It's The Applications, Stupid!"

It all gets back to differentiation, more specifically, consumers are demanding mobile, real-time, location-aware capabilities that can deliver relevant and useful content to the consumer and allow them to collaborate in a sophisticated manner with dynamically assembled groups of interest.   The money is in the applications and the revenue/advertising dollars directly or indirectly generated from them.

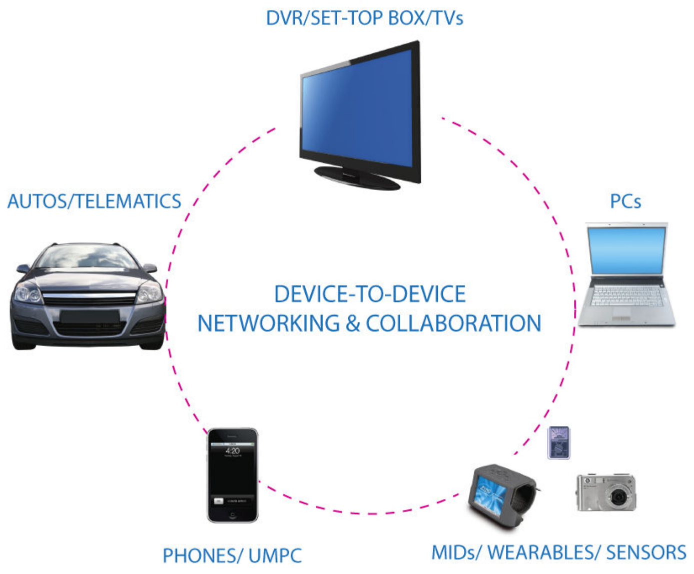
So if differentiation is in the applications, shouldn't a savvy product manager's goal be to get that application running on any device and on any network? For visionaries who have always had a pervasive mindset, the answer is an emphatic "Yes!" Yet they remain greatly challenged because of the mobile fragmentation previously discussed.

Nonetheless adopting this mentality, will provide enormous new business opportunities, and ultimately help offset the commoditization of their traditional product lines. Surely we will begin to see many of these players compete at the application level, on multiple devices, in particular with location-based services.

This mindset is quickly gaining industry support. Vishy Gopalakrishnan, founder and CTO at Mobility Partners, a wireless and mobility consultant to Fortune 1,000 companies, observed at a recent meeting of the New York Software Industry Association (NYSIA), "We see location as something that factors into decisions for building applications." He went on to say, "Look at what Nokia is doing and how they are moving from being a devices company to a software and services company and what Apple is doing with Apple MobileMe -- there is a lot of money to be made." Indeed, these applications will generate millions of dollars across every commercial, governmental and social industry and will change the way we live, work and play by seamlessly integrating with the "screens of life" that we use on a daily basis. The traditional 'Three Screens of Life' has expanded to other mobile devices, to include: phones, PCS/laptops, cars, MID/UMPCs, TV/DVR/set-top and others (Figure 2).

A UNIFYING PLATFORM
FOR THE '3+ SCREENS OF LIFE'

Figure 2



The Mobile World is Not Enough

Ironically, the company that has long had a pervasive mindset is Microsoft. Despite their aggressive tactics, they are correct in promoting the need for a unified platform to unify the various devices that we use in our daily lives. Bill Gates and Steve Ballmer have each echoed these sentiments in the past year, albeit from a strictly Microsoft OS perspective. Gates expressed in this final keynote at CES 2007, "The second digital decade will be more focused on connecting people. …[Applications] will run not only on the PC, they'll run up in the Internet, or in the cloud, as we say, on the phone, in the car, in the TV. The applications will use the best of rich platforms and those Internet services."

And Ballmer noted at CTIA WIRELESS & IT last October: "We need to bring together four powerful computing phenomena that exist today: The desktop PC, enterprise computing, mobile services running in the cloud and phone devices… The other thing which I think our industry needs, so that all of our innovations can add up where the whole is bigger than the sum of the parts, is really a rich platform that supports work style and lifestyle innovation on the phone."

Microsoft's success rests on the belief that their rigidity will ultimately win the pervasive race. With so much fragmentation in the mobile space, the problem only increases when you add cars, TVs and other devices to the mix. Will developers ultimately buy into the Microsoft story? To date, no major competitor has yet to tackle the pervasive problem convincingly, even Microsoft. The Linux Community has shown a pervasive focus of late, but with many different flavors of Linux from many alliances and vendors that don't universally span everything from enterprise to desktop to embedded to phone.

Leave No Developer (Or Their Customers) Behind

Now the obvious question remains, what alternatives do the software architects and engineers have to build the pervasive applications that run on all or various subsets of the devices previously mentioned? What platform will convincingly abstract networks and protocols to finally solve the "write once, run everywhere" conundrum that Java alone failed to deliver? Advances in next-generation mobile middleware are addressing many of these problems. Such tools will be an integral part of achieving complete interoperability.

When one stops to think about the pervasive world awaiting us, there is no limit to the possibilities. And with the introduction of cross-platform pervasive middleware like VOYAGER ONE (discussed in the following section), excuses about fragmentation and technology limitations are becoming harder to justify. Getting developers, as well as business managers, to leave their silos of Symbian, Android and .NET strategies and to think pervasively is the first step.

The remainder of this paper discusses VOYAGER ONE features and code examples.

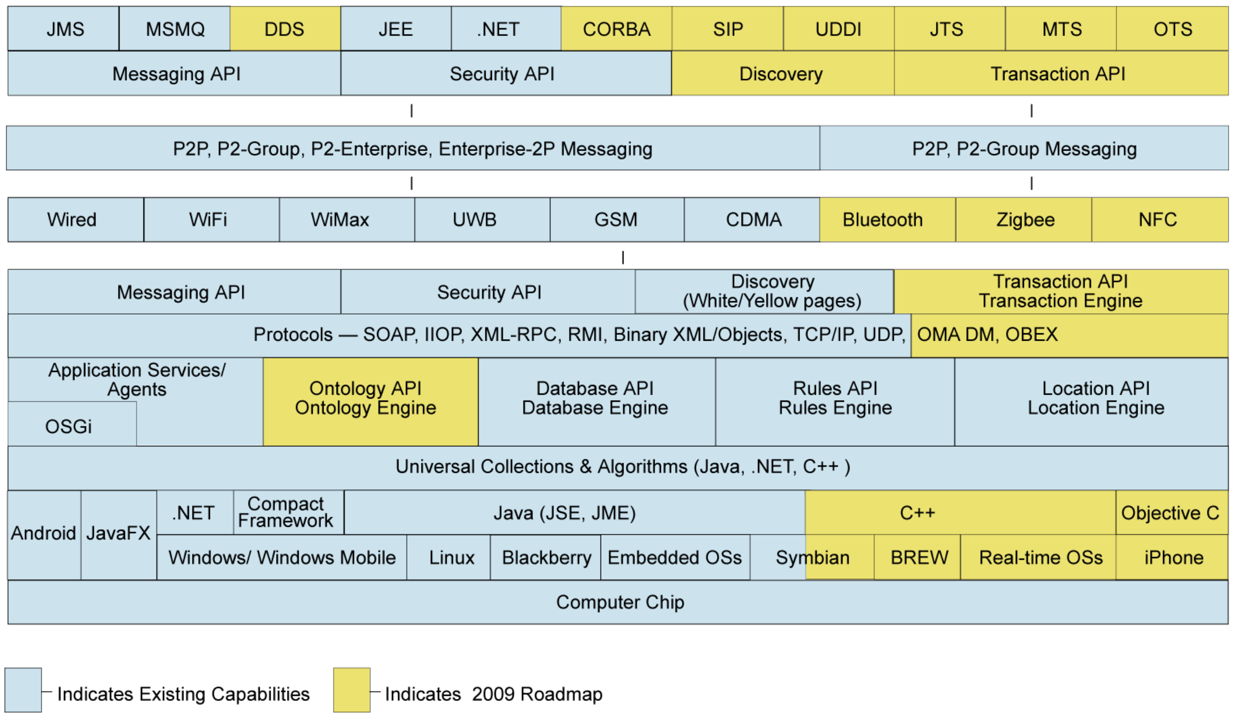
Introducing Voyager ONE

Achieving "once write, run many" is a holy grail of mobile application development. VOYAGER ONE extends multiplatform interoperability for messaging and communications to all mobile and pervasive applications (Figure 3). Other unique capabilities include real-time collaboration and discovery. VOYAGER ONE handily achieves all of these long-awaited capabilities through the most robust communications engine available in mobile middleware today. As the demand for pervasive applications continues to skyrocket, the easy community engine that VOYAGER ONE provides will quickly become a must-have tool for developers.

The VOYAGER ONE platform transcends the limitations of the Internet by enabling apps to autonomously deliver services in real-time based on context and extremely targeted marketing. It turns both phones and non-Internet based devices (cameras, sensors, RFID readers, etc.) into intelligent servers that know the user's behaviors and habits, and can predict the user's needs based on their context (location, work or play, time/day, etc.). These devices will be able to deduce the user's needs and proactively push information without requiring the user to 'Google' for, or even think about it. This framework will enable a whole new world of micro commerce through location-based, extreme personalization, intelligence, and cognitive analysis. Participation in any behavior community or distributed knowledge network will be entirely opt-in, customizable and editable in respect for the user's privacy concerns.

Figure 3

Voyager™ Software Abstraction Layer
Existing Capabilities & Roadmap



Voyager Features

VOYAGER ONE is an extension and expansion of Recursion Software's existing Voyager™ Intelligent Communications Platform to Compact Framework, JME and Android. The combination of these technologies offers exciting new capabilities to every multi-platform application that has been, or will ever be built. The following is a brief list of its capabilities:

A. VOYAGER ONE is a Survivable, Ad-Hoc, and Multi-protocol Communications Platform

- Eliminates single point of failure, a significant weakness of client-server
- Handles network interruptions and limited bandwidth networks
- Uses multiple standard protocols dependent upon target/network
- Pervasive - small, light, fast - OS & software language independent

B. VOYAGER ONE services/agents communicate w/ANY Enterprise System/Database via Universal Business Objects

- Accomplished via Integration with Enterprise Software Stacks
- Runs natively on Java (EE/SE/ME) and .NET (CLR and CF) VM's
- Provides a Universal Business Object Layer between Java and .NET virtual machines
- Supports all protocols (SOAP, Binary XML, IIOP, XML-RPC, RMI) & Messaging (JMS, MSMQ)
    - Microsoft .NET enterprise Systems
    - Java & CORBA based enterprise Systems,
    - Mainframe based systems can expose any existing Java or .NET class as a remote, movable agent with only two lines of code

C. VOYAGER ONE apps enable peer-to-peer heterogeneous device communication

- Directly to another edge device/participant
- To a group of nodes/devices /participants
- To Semi-centralized Systems/Databases
- To Centralized Systems/Databases
- To Smartphones, PDA's, MIDs/UMPCs, Laptops, Servers, Routers, Telematics and Wearable Devices, Set Tops, etc.

D. VOYAGER ONE apps bring real-time intelligence to the edge

- By gathering/filtering/analyzing and turning data into knowledge on the phone via multi-language simple APIs
- Integration with Embeddable Rules/AI EngineIntegration with Embeddable Database(s)
- Integration with Location Engines(s)

E. VOYAGER ONE enables next-generation applications

- Location-based real-time advertising
- Mobile, real-time, location-aware and advanced collaboration applications
- Mobilizing and extending enterprise or social applications to any of the 3+ 'Screens of Life'
- Multi-node Gaming, Simulation and Augmented Reality Applications
- Many pervasive apps including those forming Distributed Knowledge Networks and Intelligent Sensor Networks. VOYAGER ONE can enable self-monitoring and self-healing mobile/embedded networks
- Agents can determine the connectivity, CPU, battery, memory constraints of current device, and move to a nearby device for continued communication

VOYAGER CHAT APPLICATION DEMO

Instant messaging is used worldwide by both businesses and individuals, therefore chat makes an easy model for demonstrating many of Voyager capabilities. The basic concept is simple: two people sending messages to each other. The ChatDemo application outlined below is a distributed application that touts exposing Android services as remote services that are accessible from any node in the heterogeneous network

Note: This demo runs on multiple heterogeneous platform (e.g. Windows, Linux, etc.). Binaries for additional platforms are available for download at ftp://ftp.recursionsw.com/voyager/chatdemo/.

A. ChatDemo Capability Overview

The ChatDemo application has basic features necessary for an instant messaging application.

A ChatDemo user can:

- View a list of online users
- Send messages to another user
- Receive messages from other users

B. ChatDemo Architecture Overview

The server runs under .NET and uses the .NET version of the Voyager Edge platform. ChatDemo clients are provided for Android, Java SE, and .NET. All communication is client/server (Note that Voyager features not covered in this particular demo demonstrate Voyager's peer-to-peer/group capabilities which are not covered here.). Each client registers with the server to make its presence known, and polls the server for messages. Figure 4 is a diagram depicting the ChatDemo architecture:

ChatDemo requirements are as follows:

- The ChatDemo .NET Server and Client require Microsoft .NET 2.0.
- The ChatDemo Android client requires Android M5 RC15.
- The ChatDemo Java client requires Java 1.4.2 or better.

C. Installing and Running ChatDemo

ChatDemo ships as a ZIP file. Unzip ChatDemo.zip. Following is a description of the file and directories:

```
VOYAGER ONEChatDemo/android
```
(for Android binaries)
```
VOYAGER ONEChatDemo/dotnet
```
(for .NET binaries)
```
VOYAGER ONEChatDemo/java
```
(for Java binaries)

The .NET binaries and Java binaries will run on the host environment. The Android binaries (.apk) must be installed on the Android emulator. Assuming an emulator is running, the following command will install the ChatDemo Android binaries:

```
adb install VOYAGER
ONEChatDemo\android\VOYAGER
ONEChatDemo.apk
```

The ChatDemo Activity is called VOYAGER ONEChat.

Starting the Server

The ChatDemo server must be started before running any clients. From Windows Explorer, navigate to VOYAGER ONEChatDemo/dotnet and run ChatDemoServer.exe. The server GUI will be displayed. Change the Startup URL to "//127.0.0.1:18000". Click Startup to start the server. You should see the following (Figure 5):

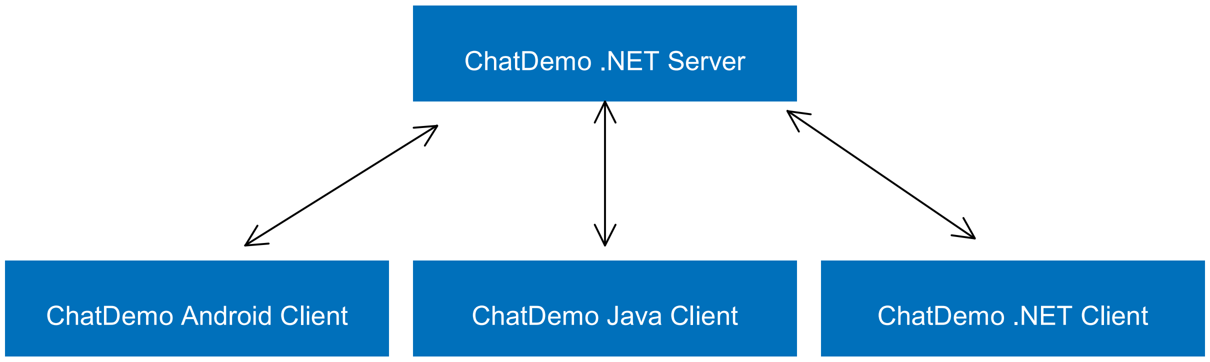
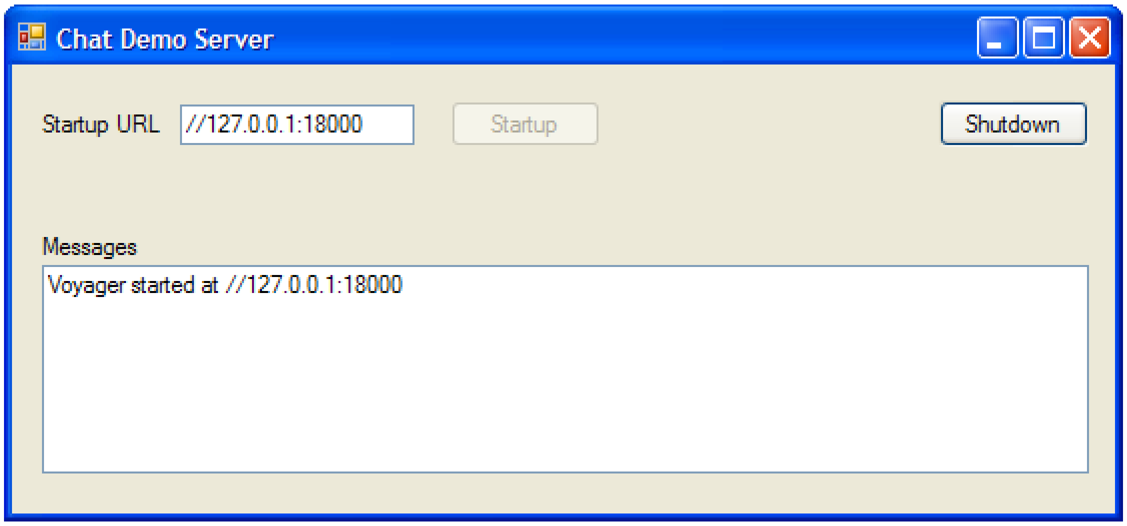


Figure 4



Voyager started at //127.0.0.1:18000

Figure 5

Starting and Using the Clients

For the purposes of this brief demo overview, we have included Java client screen shots. For Android and .NET clients, see full documentation included with the download of the binaries at ftp://ftp.recursionsw.com/voyager/chatdemo/.

The ChatDemo Java client is started by extracting

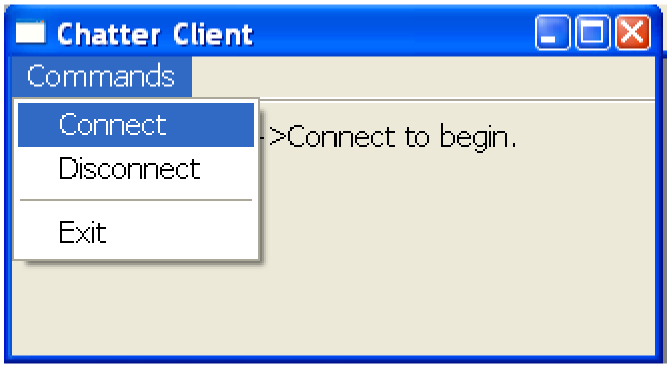
```
"jar xvf ChatDemoSwtWin32.jar"
```

and then in the same directory running

```
"java -cp
.;org.eclipse.core.commands_3.3.0.
I20
070605-
0010.jar;org.eclipse.equinox.common_3.3.0.v
20070426.jar;org.eclipse.jfVoyager
ONE_3.3.2.M20080207-
0800.jar;org.eclipse.sw
t.win32.win32.x86_3.3.3.v3349.jar;RSIcommo
n.jar;ve-core.jar Main"
```

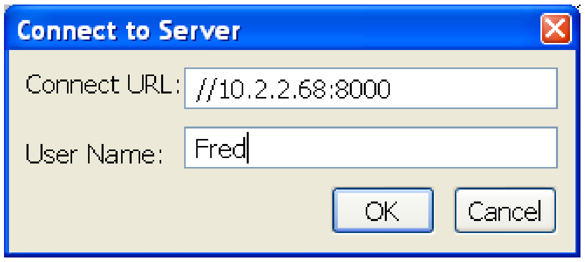
See Figure 6.

Enter the URL for the server and select your username (Figure 7).



>Connect to begin.

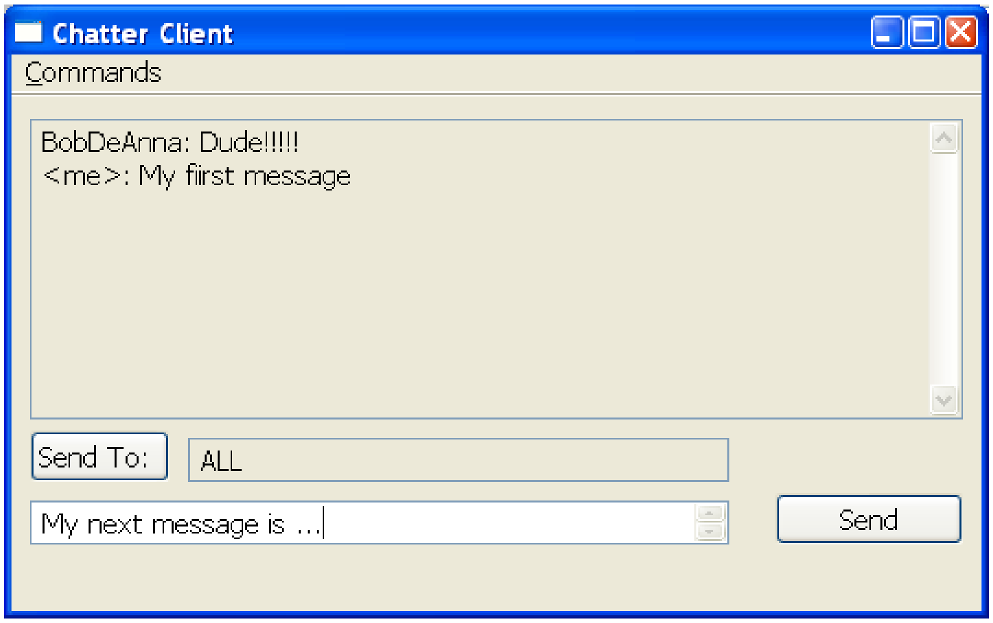
Figure 6



//10.2.2.68:8000

Fred

Figure 7



BobDeAnna: Dude!!!!!
<me>: My first message

ALL

My next message is ...

Figure 8

Once connected, you can view the list of connected users and send messages to either a specific user or all users. Type a message in the chat textbox and click “Send” to send the message (Figure 8).

D. Sample Code

The ChatDemo server is written in C#. It starts Voyager Edge on the specified URL and waits for clients to sign in, send messages, and sign out. The following two lines of code start the ChatDemo service:

```
      Voyager.startup(url);

      Namespace.bind("/ChatServer",
      chatServer);
```

The first line starts Voyager Edge on the requested URL. The second line binds the chat server implementation class into Voyager Edge’s naming service. This makes it available for lookup by a client. The server implements the following interface:

```
   public interface IChatServer
    {
        /*
* Register username and get back
userId for this chat session
         */
 int register(string name);
        /*
* Get user data on currently
registered users
         */
        UserData[] getUsers();

        /*
         * Send a message to a user
         */
        void
sendMessage(ChatMessage message);

        /*
         * Get new messages for me
         */
        ChatMessage[]
getMessages(int userId);

        /*
         * Deregister client
         */
        void deregister(int
userId);
    }
```

These methods are available to ChatDemo .NET, Java and Android clients.

The clients acquire a proxy to the chat server using the following code:

```
        Voyager.startup();
        IChatServer server =
(IChatServer)
Namespace.lookup(serverURL +
"/ChatServer");
```

The first line, again, starts the Voyager Edge client. The second line performs a naming lookup to get a proxy to the chat server running in the .NET ChatDemo server. Once the client has a proxy to the chat server it can call remote methods on it:

```
int myId = server.register("Joe");
        server.sendMessage(new
ChatMessage(myId, -1, "Greetings to
everyone!"));
```

These lines register ("sign in") a user and send a message to all connected users using the "magic" userId of -1.

[END CODE SAMPLE]

About Recursion Software, Inc.