Introduction to Bluetooth and J2ME

Part 1 - Introduction to Bluetooth and J2ME
by Jason Lam

Overview
This article goes through the introduction of what is Bluetooth and what you can do with Bluetooth. Both technical aspects of Bluetooth and development tools will be briefly introduced to you; however we will not go into great detail or specifically how to develop a Bluetooth application. This in itself will require several articles if not an entire book to demonstrate and explain. However, part 2 of this article we will go through in detail the basics of creating a J2ME Bluetooth server and client..

What is Bluetooth

Bluetooth is a wireless communication protocol mainly used for short distance and in devices with low power consumption. Because Bluetooth is capable of communicating in an omni-directional manner of up to 30 feet at 1 Mb/s it is far superior to infrared. Where infrared requires a distance of a few feet or less and requires a direct line of site for transmissions. Okay what about WiFi, which typical can transmit up to 300 feet at 11 Mb/s. Well the fact is these are really two different beasts; Bluetooth was developed for small data transfers and/or voice communications. Which makes it an excellent candidate for peripherals devices such as wireless microphones, headsets, mice, keyboards and of course mobile handsets. WiFi in general was developed to transmit large amounts of data and to serve as an extension of an existing network such as LAN. Not only does Bluetooth does away with wired cabled connections such as serial, parallel, USB and Fire; but also, it presents to us an unified standard that truly makes connecting to devices to each other ubiquitous. There are hundreds if not thousands ways Bluetooth and be used to enhance our daily lives. Aside from entertainment value of playing games head to head in multiplayer mode there are many business solutions for us to explore. Here are a couple of ideas:

- Efficient and easy way to update your PIM from home to office or where ever you go
- Easy to exchange information with others like mobile business cards
- Concurrent exchange of data, this comes in handy when a group of people are in meetings or at conferences
- Accessing devices such as printers and fax machines, this would definitely come in handy when visiting other offices of your company or client sites
- Monitoring systems, for example if you were a maintenance man doing routine system checks in a factory, it allows you to easily interface at each check point

- Going beyond the peer-to-peer use of Bluetooth there is what is called BlipNet used in enterprise scenarios, more about this later
- Profile Holder – This may be best explained with an example, say you are using your buddies gaming console that is Bluetooth enabled you can upload your saved games and download your current game. Another example, you visit your local drug store and beam your prescription and once it is filled out you get notified on your phone this allows you to continue shopping without the hassle of waiting inline or trying to decipher what is being said over the PA system.
- Provide entertainment during waiting periods, for example waiting in line to buy a ticker for movie you could play Bluetooth movie trivia games, look up facts/reviews to help you decide which movie you want to see and/or possibly opt-in for movie discounts and a chance to win a random price.

Who invented Bluetooth? Bluetooth was originally researched and developed by the Ericsson organization and were the ones who named the technology after King Harald Blatand (Bluetooth) of Denmark. Ericsson formed the Bluetooth Special Interest Group. More about Bluetooth is available at http://www.bluetooth.com/and http://www.bluetooth.org/.

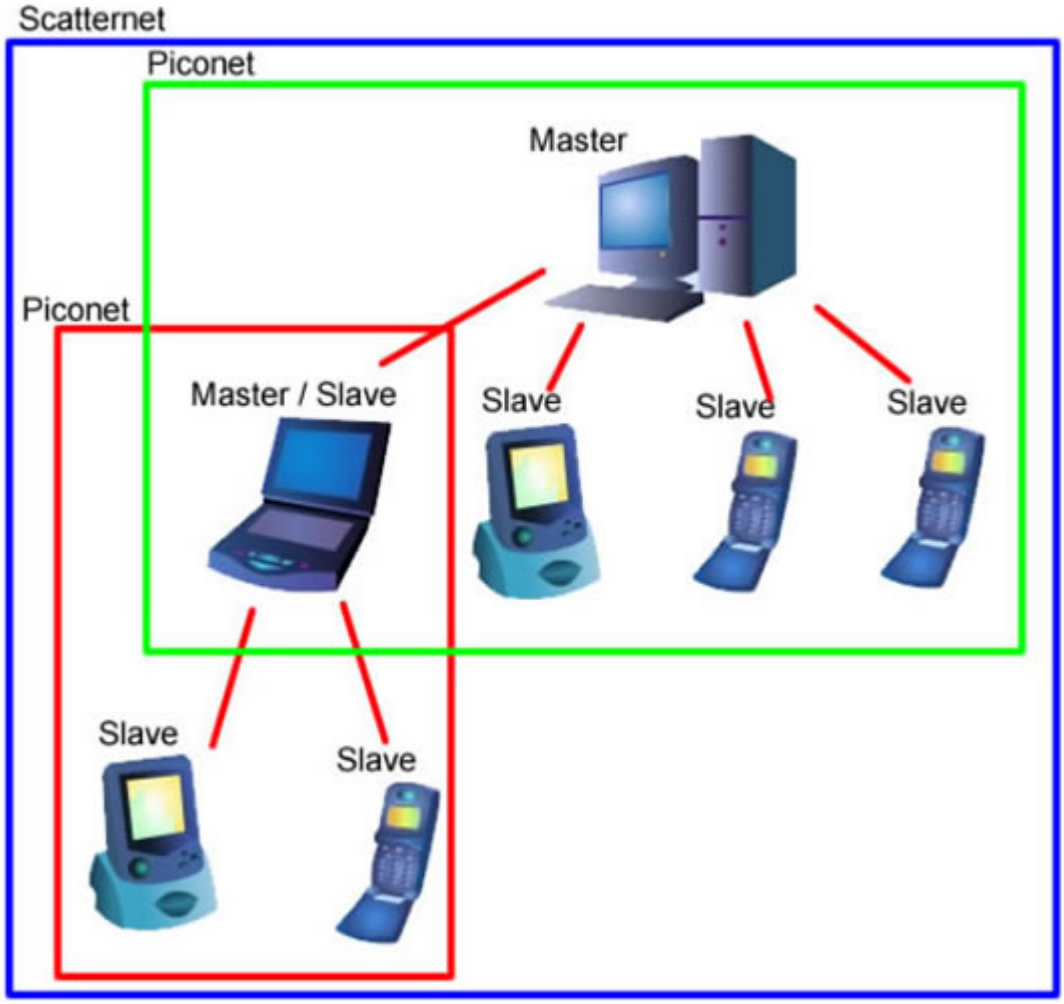
Definitely checkout all the products that are Bluetooth enabled, this definitely will if not already provide plenty of opportunity for us developers to make some innovative applications.

Technical Background

Okay now onto some the technical things you need to know about Bluetooth some of which was mentioned earlier:

- 30 Feet Range (Exception Bluetooth Class 1 has 300 feet range)
- 1 MB/s
- Capable of transferring voice
- Low power consumption
- Omni-directional radio signal
- 2.4 Ghz-2.482 Ghz
- 3 different classes (Class 1 100 meters, Class 2 20 meters and Class 3 10 meters)
- Bluetooth Protocol stack allows you to control the Bluetooth device programmatically, in this case J2ME optional package JSR 82. You will need to familiarize yourself with the Bluetooth stack and what each layers of the stack provide. For example, the following three protocols (sub-protocols under Bluetooth) RFCOMM (stream data), L2CAP (packet data) and OBEX (object data) provide you methods of transmitting data
- Bluetooth Profiles – defined functionality for Bluetooth such as Fax Profile that enables a Bluetooth device to send a fax via Bluetooth fax machine. These profiles may seem similar to the J2ME profiles but they aren't. It isn't an add-on to J2ME but rather an add-on to Bluetooth. Bluetooth profiles can be implemented in other languages like C/C++.

- The network between Bluetooth enabled devices is called a PAN, which stands for Personal Area Networks. A PAN can be a piconet or scatternet, where a piconet is when there is one master and several slaves. A scatternet consists of 2 or more masters and several slaves, in other words one of the Bluetooth devices is both a master and a slave, see illustration below:



Extra side note Bluetooth ranges can be extended/detected beyond the assigned standard, checkout http://www.wifi-toys.com/wi-fi.php?a=articles&id=21 and http://www.bluedriving.com/.

Bluetooth and J2ME / JSR 82

Like other extended APIs/libraries to the J2ME world we have one for Bluetooth the JSR 82, http://jcp.org/aboutJava/communityprocess/final/jsr082/. To start developing J2ME application/games you can download the Wireless Toolkit 2.2 (currently still Beta at the time of this writing), which includes the JSR 82 and two Bluetooth demos with source code. One of the demos is a picture sharing program, there isn't anything special you need to do aside from enabling the JSR 82 API, in the WTK go under the settings and make sure the check box for Bluetooth/OBEX for J2ME (JSR82) is

checked off. Afterwards start to instances of the program (one as a slave/client and the other as a master/server) the reset of it is self-explanatory. You will see an image being passed from one device to the other.

Beyond Peer-to-Peer

Going beyond the peer-to-peer / piconet environment and into the realm of enterprise Bluetooth applications. Enterprise Bluetooth environment major advantages over non-enterprise are:

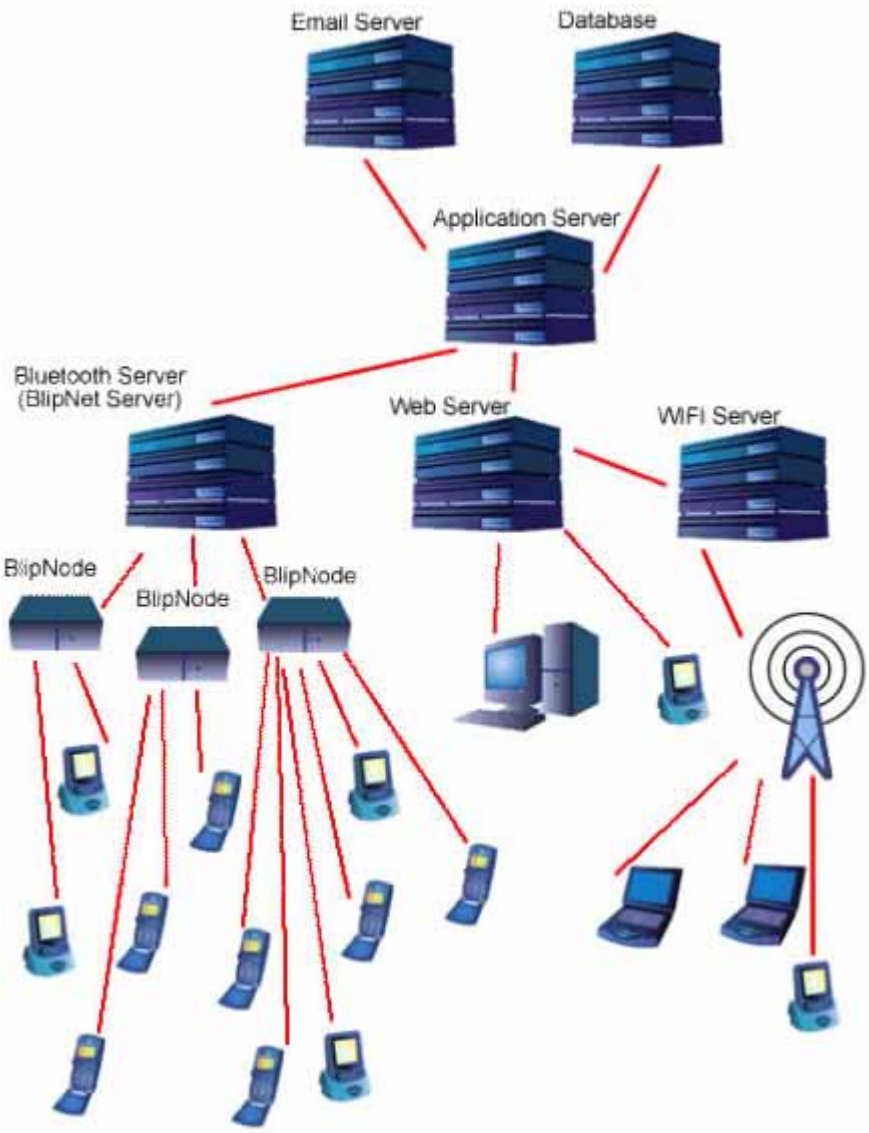
- Ability to handle more then 7 connections
- Larger range, class 1, 300 feet
- Session Management
- Centralized server

Situations where enterprise Bluetooth would come in handy are:

- Opt-in advertising campaign in highly dense communities/populations such as a mall, sporting venue or conference. This leads us to the vast world of CRM and and customer loyalty.
- Tracking employees and security monitoring large events such as large amusement park or airport
- Providing useful information in large condensed areas like amusements parks, provide information such as which restroom/restaurant/amusement ride is available with the least occupancy. Where is the nearest gift shop or where is the medical center? Possible provide advance bookings similar to what you find in Disneyland the “Fast-Pass” concept.

Bluetooth enterprise is really just an extensions or interface to larger a system, just like a person would approach a kiosk or terminal that is hooked up to a larger

computer system you would have Bluetooth nodes dispersed throughout the desired area; where each node has the ability to connect to a hundred or more Bluetooth clients. The nodes themselves are connected to a Bluetooth server (not really running Bluetooth connectivity but instead either hard wired connection or wireless WI-FI connection), this Bluetooth server itself is connected to the enterprise system. Below is a diagram of example of this:



Challenges

- Not all devices have Bluetooth and even if the device is Bluetooth enable it may not have JSR 82 profile implementation. As well you need to beware of the which Bluetooth profiles themselves are implemented such Fax, Serial, OBEX …etc
- Some wireless operates/carriers do not see the value in Bluetooth because they see it as non gain in revenue since data packets are not delivered across their network, for example Verizon has apparently crippled the use of Bluetooth on some the devices, see the following link for more information http://www.nuclearelephant.com/papers/v710.html

- Referring to the last point on Verizon's claims for blocking the use of Bluetooth is for security purposes. This does have some merit considering a lot users are not aware that their devices have Bluetooth let alone users turning off or filtering their Bluetooth options. This in turn opens doors for possibly having your mobile handset hacked into and gaining access to sensitive data. Another hack scenario is similar but may not be as affected as WiFi because of limitation in distance but a person with a Bluetooth device maybe able to steal sensitive data from a company if the proper security pre-cautions are not in place. But probably more then likely you can be spammed with advertisements this is sometimes called being BlueJacked. Now that the threat of mobile viruses is real (http://www.cnn.com/2003/TECH/10/15/itu.security/) this is or will soon become another concern in the wireless world.
- Consumers – this maybe hard to believe as most of us technically inclined people have heard or/and own Bluetooth devices and yet there seems to be a lot people who don't know what Bluetooth is. Even consumers who have heard about Bluetooth they are not sure what it really does or they think its just another way to synch their data via PC and mobile device. Telling consumers your application is Bluetooth aware/enabled may not help you it may just confuse the consumer, further education of consumers maybe needed for successfull sales of your application/game.
- As of lately some people of gained some uncertainty of where Bluetooth is heading towards with the recent new that Ericsson has discontinued the development of Bluetooth. There are arguments for and against if Bluetooth will it be around in the 10 years. You will definitely need to do some research here! Personally I don't think its dead, take for instance this headline that reads “Bluetooth® Wireless Technology Reaches Three Million Shipments per Week” (http://www.bluetooth.com/news/sigreleases.asp?A=2&PID=1352&ARC=1&ofs=), there are still are plenty of supporters and manufacturers of Bluetooth devices.

Summary

Well hopefully you have a better grasp of what Bluetooth is and all the potential opportunities there are in developing applications using Bluetooth. Stay tuned for the up coming article where we dive into the code of a simple Bluetooth J2ME application. Where we dive into the detiasl of how to do the following: device and service discovery, service registration and the actual communcation between Bluetooth devices.

Part 2 - Introduction to Bluetooth and J2ME

by Jason Lam

Overview

In Part 1 we went over the basics of Bluetooth technology and some possible development opportunities Bluetooth gives us. Part 2 of this article goes into a little more detail how to create a Bluetooth server and client.

About the Example

In keeping things simplistic as possible proper handling of multiple slaves/clients is not implemented essentially it only works with one server and one client. As well the sample is just that sample code, it is coded from the perspective of getting the two nodes to communicate with each without any real effort in "proper" coding technique/design. Remember the focus is to get you started with the actual Java Bluetooth API.

Before we continue you should download the JSR 82 Java Doc API from http://jcp.org/aboutJava/communityprocess/final/jsr082/. Familiarize yourself with the available classes and get feel where things are. We won't go into every single class or explain every single detail mainly because the Java Doc does a good job of this already.

This demo includes three source files / classes:

- BluetoothEchoDemo.java – The main driver that allows you to start a client or server
- EchoClient – The client class
- EchoServer – The server class

This example uses RFCOMM for transferring data the other two options are L2CAP and OBEX. In short RFCOMM is used for streaming data, L2CAP is for packet data and OBEX is for object data.

Client Overview

Let us start with the client, the first thing we need to do is to determine what Bluetooth devices and services are there around us. First get the local device and obtain the discovery agent.

```
LocalDevice localDevice = LocalDevice.getLocalDevice();

discoveryAgent = localDevice.getDiscoveryAgent();
```

Now to start searching for Bluetooth devices within the area use the startInquiry method from the DiscoveryAgent class.

```
discoveryAgent.startInquiry(DiscoveryAgent.GIAC,this);
```

DiscorveryAgent.GIAC stands for “The inquiry access code for General/Unlimited Inquiry Access Code” refer to the Java Docs for the other inquiry access code definitions.

Now with the use of the Discovery Listener you have available to you 4 call back methods. As defined in the Java Doc they are:

Method Name	Description
void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod)	Called when a device is found during an inquiry.
void inquiryCompleted(int discType)	Called when an inquiry is completed.
void servicesDiscovered(int transID, ServiceRecord[] servRecord)	Called when service(s) are found during a service search.
void serviceSearchCompleted(int transID, int respCode)	Called when a service search is completed or was terminated because of an error.

That is it for the discovery part! It really is that simple. Now with these calls being triggered you can obtain device information, service information and with that make the appropriate data transmission/communication with the Bluetooth server. (see source code)

In the deviceDiscovered method we obtain some device info which are more or less self-explanatory. Except for the Major and Minor classification, these two attributes let you know what type of device it is. The following is an incomplete sample of the Major and Minor definitions, for a complete list visit http://www.bluetooth.org/. When you run the program you will notice it outputs Major 512 and Minor 4, according to the chart it is a Phone / Celluar.

Major Class	Minor Class	Major Class Description	Minor Class Description
256	4	Computer	Desktop
256	8	Computer	Server
256	12	Computer	Laptop
256	20	Computer	PDA
512	4	Phone	Cellular
512	8	Phone	Household cordless
512	12	Phone	Smart phone
1536	32	Imaging device	Camera
1536	64	Imaging device	Scanner
1536	128	Imaging device	Printer

In the servicesDeiscovered method you can obtain the respective URL needed to open a connection to the available service(s).

```
for(int i=0;i<servRecord.length;i++) {

serviceUrl = servRecord[i].getConnectionURL(0,false);
}
```

Now that you have the url, you can send data through the standard Generic Connector.

```
String msg = "Say Hello World";
conn = (StreamConnection)Connector.open(serviceUrl);
OutputStream output = conn.openOutputStream();
output.write(msg.length());
output.write(msg.getBytes());
output.close();
```

Server Overview

In the server class we need to initialize once again but this time instead of searching for devices we need to setup a server

```
private static String serverUrl = "btspp://localhost:" +
BluetoothEchoDemo.RFCOMM_UUID + ";name=rfcommtest;authorize=true";
.
.
.
conn = null;
localDevice = LocalDevice.getLocalDevice();
localDevice.setDiscoverable( DiscoveryAgent.GIAC );
notifier = (StreamConnectionNotifier)Connector.open(serverUrl);
.
.
.
```

Now make a connection with the same Generic Connector you use when making HTTP calls. The url definition is as follows:

```
scheme://host:port;parameters
```

Name	Description
scheme	Connection type such as http, in the case of Bluetooth you use btspp for RFCOMM or btl2cap for L2CAP
host	The address to connect to or if you are setting up a server then you put in localhost
port	The port of the client connections and for servers it describes the UUID more about this later.
parameters	Specify option parameters ie: like given the service name name=rfcommtest

Now wait for a client response, this pauses the thread until it receives something.

```
conn = notifier.acceptAndOpen();
```

Once a client responds read the data in and send back a message

```
// Read Data Transmission
String msg = BluetoothEchoDemo.readData(conn);
System.out.println("Received Message from Client: " + msg);
// Send Back a Message
msg = "Hello Back from Server";
output = conn.openOutputStream();
output.write(msg.length()); // length is 1 byte
output.write(msg.getBytes());
output.close();
```

Running the Example

Download the source and compile from here. You will need to invoke the Sun WTK twice because the outputs are displayed in the consoles. Start one instance and select Server, then start a second instance and select Client (Yes you need to start 2 instances of the WTK not just simply hit the "run" button twice on the same instance because the console of the first instance will not show). Like HTTP calls you may need to answer yes for connections being made.

In the end you will see the following in the server console:

```
Starting Echo Server
Server Running...
Received Message from Client: Say Hello World
```

In the client console you will see:

```
Device Discovered
Major Device Class: 512 Minor Device Class: 4
Bluetooth Address: 000060854FBF
Bluetooth Friendly Name: WirelessToolkit
InquiryCompleted
ServicesDiscovered
SERVICE_SEARCH_COMPLETED
Service URL:
btspp://000060854FBF:1;master=false;encrypt=false;authenticate=false
Hello Back from Server
```

Summary and What is Missing

You should be able to make a basic RFCOMM communication between a Bluetooth client and server. However, even though we went through a lot detail there are still details that were left out or not mentioned. Now that you have a better understanding of Bluetooth with J2ME you need to dive deeper into the following:

- An understanding of Service Discovery Database (SDDB)

- A more in depth look at UUIDs and other Data Elements
- Understand what a Service Record is and what it does
- Understanding the relationship between SDDB, Service Record and Data Elements

As a recap the example in this article the primary goal is to give you an introduction to setting up the communication between a master and slave. Rather then jumping into a full example and getting over whelmed this sample code layouts out the bare bones. Along with more research into the APIs and review of more complete samples of Bluetooth you will be on your way to making some exciting new Bluetooth games/aplications. More complete examples are available with developments kits such as:

- BluetoothDemo comes with the WTK 2.2 Beta (Sun)
- BluePad comes with the SDK from SonyEricsson
- BlueToothCar comes with the SDK from SonyEricsson
- Samples that comes with the Rococo Software SDK

Source Code
Download Source to Bluetooth Example (Click Here)

Development Kits
http://java.sun.com/(Wireless Toolkit 2.2 Beta includes JSR 82)
http://forum.nokia.com/(Nokia Developer Suite 2.2 includes JSR 82)
http://developer.sonyericsson.com/(J2ME SDK 2.1.4 Beta includes JSR 82)
http://www.atinav.com/
http://www.rococosoft.com/
http://www.esmertec.com/
http://www.smartnd.com/
http://www.oi-us.com/

Resources
http://jcp.org/aboutJava/communityprocess/final/jsr082/
http://www.bluetooth.com/
http://www.bluetooth.org/
http://opensource.nus.edu.sg/projects/bluetooth/
http://bluez.sourceforge.net/
http://benhui.net/