CTS: Children Tracking System
Table of Contents

Introduction .................................................................................................................................................. 3
Work Done .................................................................................................................................................. 4
Conclusion .................................................................................................................................................. 7
References .................................................................................................................................................. 8
**Introduction**

In this progress report, as group 10, we will mention about what we have done up to this point about our project, the CTS (Children Tracking System).

After the submission of our initial progress report, we have focused on the actual implementation of the tracking system. We have researched about possible methods, libraries and scripts that we can use in the implementation. Since we will be using the Java ME platform we've gone through the documentation and found several libraries that could be of use to us.

The following is a list of such libraries and explains our plans on how they'll be used.


**Work Done**

Our approach for implementing this project requires us to make use of the JavaME platform. As explained above, we are going to use following java libraries:

```java
//For Exceptions
java.io.IOException;
javax.bluetooth.BluetoothStateException;

//For Closer Agents’ list
java.util.Enumeration;
java.util.Vector;
javax.bluetooth.DeviceClass;
javax.bluetooth.DiscoveryAgent;
javax.bluetooth.DiscoveryListener;
javax.bluetooth.L2CAPConnection;

//Control Local Mobile Device
javax.bluetooth.LocalDevice;

//Control Remote Mobile Device
javax.bluetooth.RemoteDevice;

//Connection Variables and Keys
javax.bluetooth.ServiceRecord;
javax.bluetooth.UUID;
javax.microedition.io.Connector;

// For Midlet (GUI) design
javax.microedition.midlet.*;
javax.microedition.lcdui.*;
```

Basically, we are going to set one-by-one communication for each device. Throughout our approach we will ask to the client if it is there and if there are any more clients around it. Hence we can transmit to other nodes and other devices around the clients.

We are going to explain main classes and methods that we are going to using with the explanations.
For One-to-One communication as a prototype, we will be using two different sides, server and client. These two classes will be called via main program, so server can establish the main communication.

For the server side, we are going to initialize communication protocol and other required protocols. As a todo list we will work on following steps:

a. Define local device
b. Set the local device discoverable
c. Define UUID value for
d. Get the Client’s Name
e. Reunion and create the URL to communicate
f. Open and accept the communication.

Here it will be important to mention about UUID. A Universally Unique Identifier (UUID) is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). The intent of UUIDs is to enable distributed systems to uniquely identify information without significant central coordination. Thus, anyone can create a UUID and use it to identify something with reasonable confidence that the identifier will never be unintentionally used by anyone for anything else. Information labeled with UUIDs can therefore be later combined into a single database without needing to resolve name conflicts.\(^1\)

Right after the server handle the handshaking protocol, we need to wait the client’s response. Then client sends its response. After server obtained the related information server closes the connection and try to connect another connection to obtain other information.
As a client side application; mostly the communication protocols works same. Here for the client side, the most important issue is synchronization. The rest of the connection methods are the same. The critical point is UUID address should be the same for a specific connection. Hence it won’t be a random value, which should be a known value for two different sides. The UUID works like a channel name.

The main code leads two different functions, sending and receiving data. Moreover, it controls the GUI of the system. Therefore, we are going to separate the whole system as three parts which are Client, Server, and Gui Part that are working under the Main system.
Conclusion

Up until now, we have done lots of research about concept, technologies, implementations etc. Based on that research we've looked for resources that could be utilized in the implementation stage. We've located such resources in terms of libraries and scripts.

In the upcoming weeks, we are plan on utilizing those libraries and start coding the actual implementation.
References

1) http://en.wikipedia.org/wiki/Universally_Unique_Identifier