Journal of Computational Information Systems 6:14 (2010) 4617-4625
Available at http://www.Jofcis.com

Mobile Bluetooth-Based Multi-player Game Development in Ubiquitous Computing

Andy S.Y. LAI[1,†], A.J. BEAUMONT[2]

[1]*Department of Information and Communications Technology, Institute of Vocational Education, Hong Kong*

[2]*Department of Computer Science, Aston University, Aston Triangle, Birmingham, United Kingdom*

Abstract

This paper describes the use of Bluetooth and Java-Based technologies in developing a multi-player mobile game in ubiquitous computing, which strongly depends on automatic contextual reconfiguration and context-triggered actions. Our investigation focuses on an extended form of ubiquitous computing which game software developers utilize to develop games for players. We have developed an experimental ubiquitous computing application that provides context-aware services to game server and game players in a mobile distributed computing system. Obviously, contextual services provide useful information in a context-aware system. However, designing a context-aware game is still a daunting task and much theoretical and practical research remains to be done to reach the ubiquitous computing era. In this paper, we present the overall architecture and discuss, in detail, the implementation steps taken to create a Bluetooth and Java based context-aware game. We develop a multi-player game server and prepare the client and server codes in ubiquitous computing, providing adaptive routines to handle connection information requests, logging and context formatting and delivery for automatic contextual reconfiguration and context-triggered actions.

Keywords: Mobile; Ubiquitous Computing; Bluetooth; Multi-player Game; Context Aware System

1. Introduction

One significant aspect of the merging mode of ubiquitous computing is the constantly changing execution environment. This work presents an experimental context-aware application that provides context-aware information to game server and game players in a mobile distributed computing environment. Game players might access their computing resources from wireless portable machines and also through stationary devices and computers connected to local area networks. The hardware configuration is continually changing. Similarly, wireless portable machines may move from one location to another, requiring users to join or leave games, frequently interacting with other machines while changing location. One challenge for game development using mobile Bluetooth-based distributed computing is to exploit the changing environment with a class of game servers that are aware of the context in which they are run [1, 2]. Such context-aware software adapts according to the location, the collection of nearby players, hosts, and accessible devices, as well as to change in the runtime environment over time.

Our aim is to construct an adaptive distributed computing application which uses Bluetooth and

† Corresponding author.
Email addresses: andylai@vtc.edu.hk (Andy S.Y. LAI).

December, 2010

Java-Based technologies to implement a multi-player mobile game in ubiquitous computing [3][4][5]. This prototype context-aware application can be viewed as a concrete example of automatic contextual reconfiguration and context-triggered actions in ubiquitous computing. The implementation steps required to develop this application are micro-architectural elements of a ubiquitous computing framework that documents and motivates the semantics of ubiquitous computing in an effective way [6]. The rest of this paper is organized as follows: Section 2 describes the end application and the overall system architecture. Section 3 presents the details of implementation of a prototype game server providing automatic context reconfiguration of its discoverable device and services in ubiquitous computing environment. Section 4 illustrates the details of implementation of game players for context-triggered actions occurred in ubiquitous computing. Finally, in Section 5, we give some concluding remarks.

2. System Architecture in Context-Awareness Mobile Bluetooth-Based Game

The mobile Bluetooth-based game we developed is deployed using client–server system architecture. The software that generates the game world runs continuously on a server, and players connect to it via client software. The client software may provide access to the entire playing world. The Bluetooth standard allows up to seven slaves devices and a master to form a centralized ad hoc network called a piconet. In our mobile Bluetooth-based game, the master can be allocated by all its clients. When there is a high density of clients around, we have to interconnect the multiple close-by piconets to form a scatternet [7][8]. The Bluetooth standard does not specify a specific scatternet formation algorithm. We adopt the scatterent topology similar to the one in Fig. 1. The overall network architecture is based on the cooperation of Bluetooth technology wireless network.

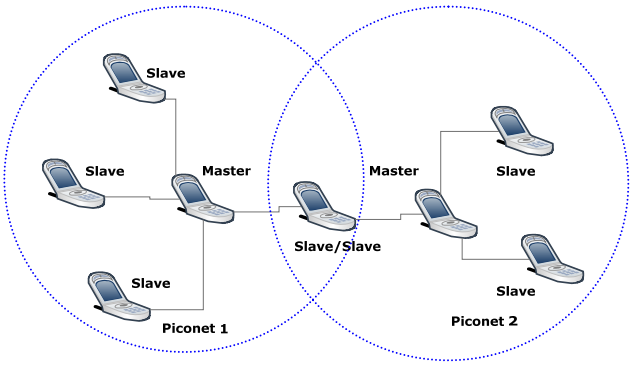

Fig.1 Bluetooth Topology Where Two Piconets are Connected to Form a Scatternet Mobile Game Network.

Two piconets can communicate by sharing one or more bridge devices. Each bridge may act as a master in one piconet and as a slave in the other or as a slave in both piconets, but not as a master in both.

The system considers two types of software entities: Bluetooth game players (slaves in the scatternet) and the central Bluetooth game server (a master in the scatternet). A mobile game player only requires a Bluetooth enabled mobile phone device. A mobile client, while wondering around the game server, will continuously search for new game servers using the Bluetooth inquiry process. When a server is found, it is checked to see if it has the required Universally Unique Identifier (UUID) and Service Discovery Protocol (SDP). The search for game servers takes place automatically, although the search can also be done on demand. The game players can have automatic context reconfiguration by making a selection online from

the game servers list appearing on their mobile phone devices. The selected game server will then be used by game players to perform the context-triggered actions in the ubiquitous computing environment [9].

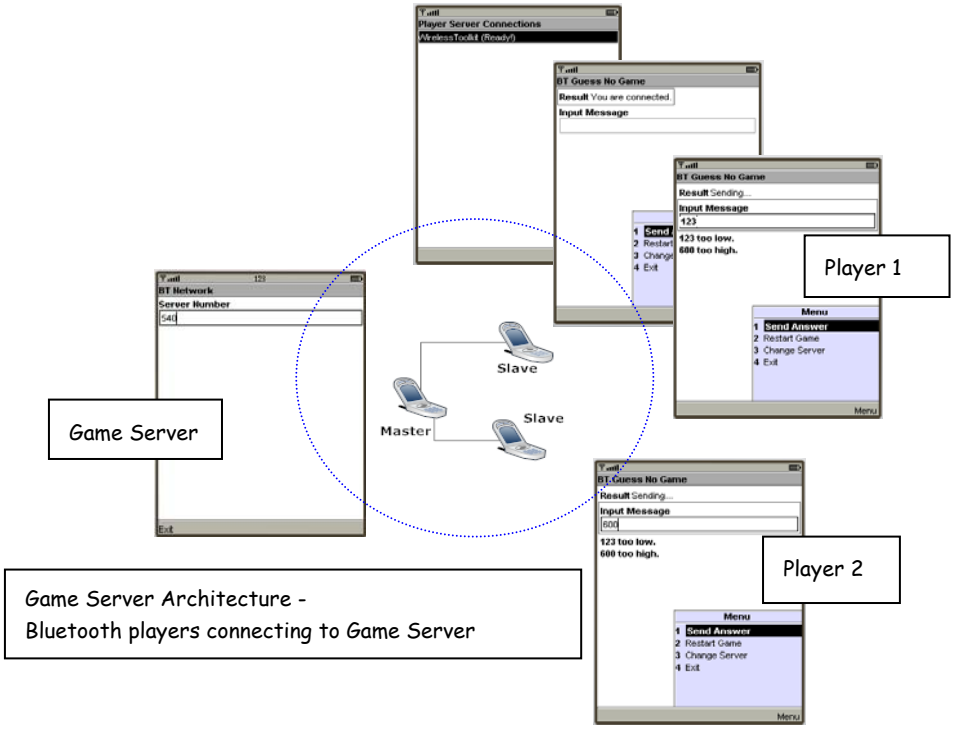

Fig. 2 Mobile Bluetooth-Based Game System Architecture.

To investigate the performance profile achievable using our Bluetooth and Java-Based technologies we have developed a simple multi-player game for mobile devices in ubiquitous computing, a mobile game in which players guess numbers. We use this game as our basic benchmark and as the backbone of the following sections to facilitate our discussion in this paper. Although the example game given here is relatively simple, it is useful to illustrate many typical issues encountered in the ubiquitous computing. Fig. 2 shows a pictorial representation of the system architecture of our Guessing Number game.

3. Implementation of Mobile Bluetooth-Based Context Awareness Game Server

Ubiquitous computing aims to enable service delivery through proactively adapting use and access with respect to available context information. This section describes the technical details of implementation of the prototype mobile game server for discoverable devices and the services it provides that implement the context-aware environment in ubiquitous computing.

3.1. Dynamic Service Discovery Registration in Game Server

A Game Server is a service in Bluetooth that has a persistence presence on the mobile device, performing the function of context-awareness, often in the background. The way to create a Game Server service is by packaging the service in a mobile application, which responds to Bluetooth Service Notification. The service sits in an *uninitialized* state and the notification is basically an instruction to that server to initialize

itself. The *initialization* step includes allocating globals, creating listener server socket and registering the socket with Service Discovery Protocol (SDP). The States Diagram in Fig.3 presents the Service States in Game Server. The *Session Start* state accepts the game player's connection which actually is the incoming player's inquiry and contains the Bluetooth device address and the Class of Device, whereas the Bluetooth game player decides what service in Game Server to connect to and find out how to connect to it. Lately, the game player can send SDP requests and will receive SDP responses from the SDP server in the Game Server. The SDP layer responses to requests autonomously, where the context-awareness takes place here.

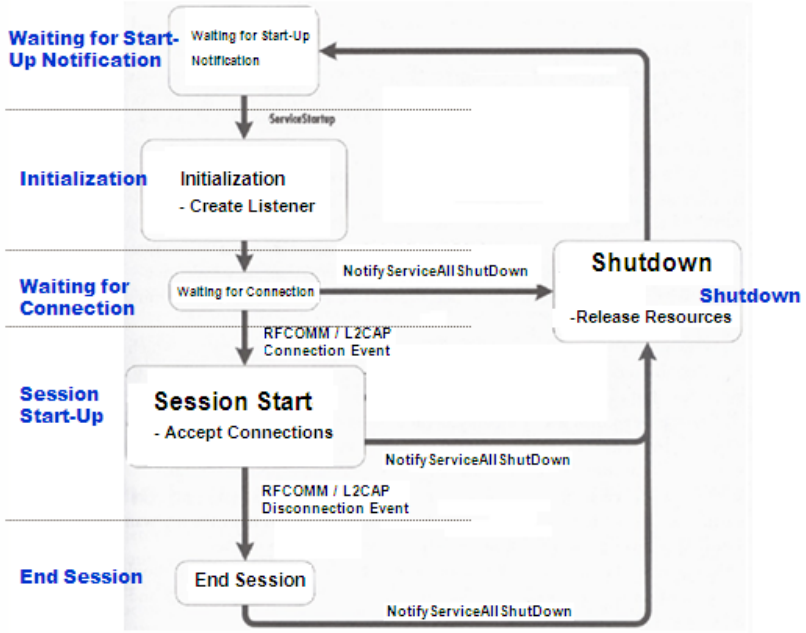

Fig.3 Service States of Bluetooth-Based Game Server

The sequence diagram in Fig. 4 shows how the connection is established using RFCOMM socket in Server. The TCP Server Socket programming is applied with socket listening and creation [10]. A socket from game player must be connected before performing any I/O data manipulation. The procedure for doing this depends on whether the server socket (`notifier`) is accepting incoming connections (game player sockets).

The game server socket programming involves three essential steps as follows:

Step1. After the notifier is created by the entity in server, Connector, it will be bound to a local Bluetooth adapter and port number (URL);

Step2. The server then accepts incoming connections (sockets) from game players.

Step3. Once a game player finishes the play operations with its connected socket, it will call close on the socket to disconnect and free the system resources used by the connection.

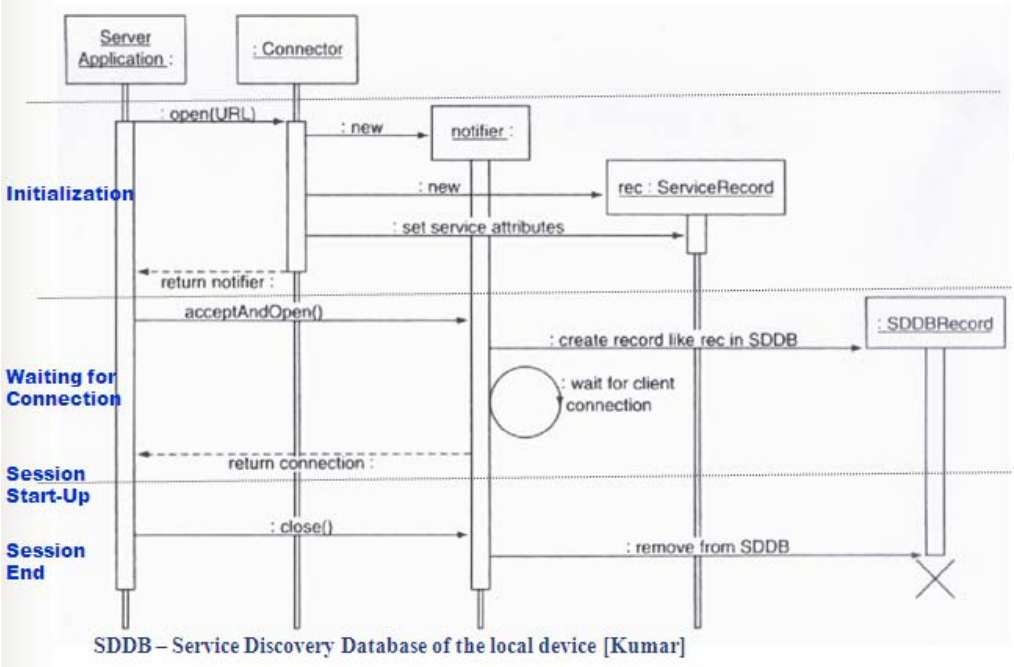

Fig.4 Sequence Diagram of Game Server to Accept Connections from Game Players.

3.2. Implementation of Dynamic Service Discovery in Game Server

The game was developed using the standard Java API's for Bluetooth wireless technology (JABWT) [11] proposed by the Java Expert Group JSR-82. The JABWT standard provides the socket API to BTSPP. The Connector entity in JABWT employs the `notifier` to accept socket and returns the connection. Once the registration has been done successfully, the game server can be discovered by its hostname and UUID. As a result, the context-awareness in the service discovery in server happens at runtime environment.

Fig. 5a and 5b shows the technical details in implementing the context-awareness in Bluetooth game server with Java coding.

The socket layer waits for a connection coming from the Bluetooth interface. Fig.5a shows the Java coding of the Dynamic Service Discovery and the Manipulation of I/O data. This latter process is repeated in the game server every 0.1 seconds.

While a game player connecting to the game server, the server will spawn a child process in a thread to deal with the mobile client's request. Fig.5b presents the implementation of the internal thread in Service Discovery, which will pause until the transmission of the game player occurs. The game server accepts and stores the connections and, lately, uses them to read the command or data from game players. You may have observed that the connections maintained by the game server can be used to broadcast messages to all Bluetooth game players as well.

```
UUID RFCOMM_UUID = new UUID(0x1234);
serverUrl = "btspp://localhost:" +
   RFCOMM_UUID + ";name= rfcomm;authorize=true";
localDevice = LocalDevice.getLocalDevice();
//set server discoverable by clients
localDevice.setDiscoverable (DiscoveryAgent.GIAC);
notifier = (StreamConnectionNotifier)
Connector.open(serverUrl);
//accept connections
System.out.println("\n\n Server Running...");
serverAcceptThread = new Thread(
   new ServerAcceptThread(notifier, this));
serverAcceptThread.start();
//reads requests and sends its responses
while (true)
{
   readMessage(); //read data from a client
   sendMessage(); //broadcast message to clients
   try
   {
   Thread.sleep(100); //pauses every 100 ms
   } catch (Exception e){}
}
```

Fig. 5a Dynamic Service Discovery and Manilpulating Requests and Reponses in Game Server

```
public class ServerAcceptThread implements
Runnable …
{
StreamConnectionNotifier notifier;
public void run()
{
 while (true) {
  try{
 //Pauses thread until Transmission occurs
 StreamConnection conn =
notifier.acceptAndOpen();
 Server.addConnection(conn);
 }
 catch (Exception ex){
  System.err.println("Server Error..");}
}}
```

Fig.5b Internal Thread for Connections Acceptance in Game Server

4. Implementation of Mobile Bluetooth-Based Context Awareness Game Players

Bluetooth devices need a way to exchange a limited of data, allowing them to find and connect to each other before synchronizing on a common click and Bluetooth address. This section details the implementation of a prototype mobile game player, which includes two main functions: (a) to discover the server devices and their services in ubiquitous environment, and (b) to select a server from the list of available servers. The game player can connect and perform the game I/O in the form of requests and responses with the Bluetooth-based game server.

4.1. Discovery of Server Device and Services with Context Awareness in Game Players

The first step for each mobile game player is initialization with the proper searching properties for Bluetooth-Based Game Server such as the Service Discovery Protocol (`btspp://`)[12][13]. The game client starts his search through a Bluetooth inquiry process. If a server device is found, it will proceed to do SDP services searching. The application may have found more than one server device; in which case it will create a "Player Server Connections" list of the discovered server objects with their names.

In Fig. 6, the implementation of the inquiry process in game player with the Java coding is shown on the left of the figure and the "Player Server Connections" list is shown on the right. Once the player has chosen a server from the list on his mobile device, the player application will then connect to the game server device and keep stable throughput with server.

```
// Client Device
LocalDevice localDevice =
    LocalDevice.getLocalDevice();
discoveryAgent = localDevice.getDiscoveryAgent();

// Client Display servers on List
serverList = new List("Player Server Connections",
List.IMPLICIT);
serverList.setCommandListener(this);
Display.getDisplay(midlet).setCurrent(serverList);

// Start Searching for Bluetooth Devices
discoveryAgent.startInquiry(
   DiscoveryAgent.GIAC, this);
try{
   createMainForm(); // Display player screen
   // process the commands
   ClientRequestThread thread =
     new ClientRequestThread(GamePlayer.this);
   thread.start();
} catch (Exception e){}
```

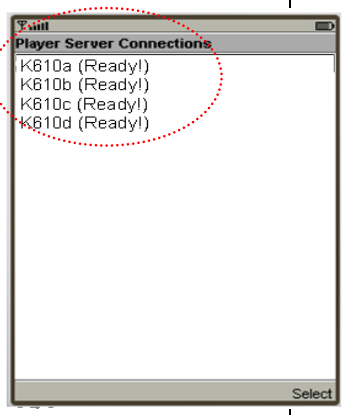

Fig.6 Inquiry Process in Game Player produces List of Available Bluetooth-Based Game Servers

4.2. Requests and Responses with Context Awareness in Game Players

Once connected, the Mobile Bluetooth-Based players will send the game data to the server, guessing numbers and receiving the responses from the Bluetooth game server. The entire process of sending and receiving game data is handled by a separate internal thread in game player program. The thread resumes its execution every 0.1 seconds. Fig. 7 shows the implementation of that internal thread to handle requests and responses in the game player program.

```
public class ClientRequestThread extends Thread
{
conn = (StreamConnection)  Connector.open(serviceUrl);
dis = conn.openDataInputStream();
dos = conn.openDataOutputStream();

public void run ()
{
 while (isRunning){
  if (message != null){//message available
     byte[] data = message.getBytes();
  dos.writeInt(data.length);
  dos.write(data);
  dos.flush();
  message = null;
}
readMessage();
if (dis.available() > 0){//input available
   int length = dis.readInt();
   byte[] data = new byte[length];
   dis.readFully(data);
}
try{
      Thread.sleep(100); //pauses every 100 ms
} catch (Exception e){}
}}
```

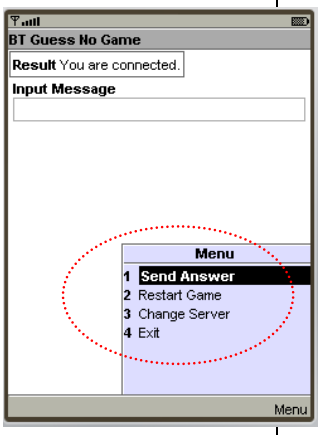

Fig.7 Internal Thread in the Game Player Program to Handle Requests and Responses.

5. Conclusions and Future Work

In this paper, we presented an experimental Mobile Bluetooth-Based game development in ubiquitous Computing. The JSR-82 standard has been employed as a programming interface to mobile client and server code. We demonstrated that Bluetooth is a candidate wireless networking technology to support ubiquitous applications in a mobile game distributed system environment. The implementation steps in the application can be viewed as micro-architectural elements of a ubiquitous computing framework that documents and motivates the semantics in ubiquitous computing in an effective way. Our investigation focused on an extended form of ubiquitous computing which game software developers utilize to develop games for players. We call this study an experimental context-aware application which provides context-aware information in a ubiquitous computing environment.

We observed that Bluetooth offered a relatively steady throughput. As a matter of fact, we found that the delay incurred during service inquiry remains constant for distances up to 20 meters in a room. The key future advancement in mobile Bluetooth-Based game technology should be the design of fast inquiry service discovery in an environment which offers a vast selection of available game servers. In this paper, we have demonstrated the practicality of applying Bluetooth and Java-based technologies to develop a multi-player mobile game in ubiquitous computing.

Acknowledgement

This work was partially supported by Hong Kong Vocational Training Council, Hong Kong, under Grant (SDS) 000163.

References

[1] B. Schilit, N. Adams, R. Want: Context-Aware Computing Applications, Proceedings of Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994. Pages 85-90, IEEE Computer Society.

[2] T. Gu, H.K. Pung, D.Q. Shang: A service-oriented middleware for building context-aware services, Journal of Network ad Computer Applications 20 1-18, Elsevier, 2005.

[3] A.S.Y. Lai, A.J. Beaumont: Meta-based Distributed Computing Framework, Lecture Notes of Computer Science, Parallel and Distributed Processing and Applications, ISPA2004, LNCS 3358, p85-90, Springer Verlag, December 2004.

[4] A.S.Y. Lai, A.J. Beaumont: A Metalevel Component-Based Framework for Distributed Computing Applications, Fourth Annual ACIS International Conference on Compute and Information Science (ICIS'05), IEEE, Journal of Computer Society, p268-273, April 2005.

[5] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire: Distributed Topology Construction of Bluetooth Personal Area Networks, Proceddings of IEEE INFOCOMM, Anchorage, Alaska, USA 2001.

[6] A.S.Y. Lai: Meta Level Component-Based Framework for Distributed Computing Application, PhD's Dissertation, Computer Science, Engineering and Applied Science, Aston University, Birmingham, UK, 2008.

[7] J. Cano, P. Manzoni, C.K. Toh: UbiqMuseum: A Bluetooth and Java Based Context-Aware System for Ubiquitous Computing, Journal of Wireless Personal Communications, Springer Verlag, 2006.

[8] C. Law, K.Y. Siu: A Bluetooth Scatternet Formation Algorithm, Proceedings of the IEEE Symposium on Ad Hoc Wireless Networks 2001.

[9] A. Soylu, P. Causmaecker, P. Desmet: Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering, Journal of Software, Vol 4, No 9 (2009), 992-1013, Nov 2009.

[10] C.B. Kumar, P.J. Kline, T.J. Thompson: Blutooth Application Programming with JAVA APIs, Morgan Kaufmann Publishers, 2004.

[11] B. Kumar: JSR-82: Java APIs for Bluetooth. Available at: http://www.jcp.org/en/jsr/detail?id=82.
[12] Promoter Members of Bluetooth SIG: Specification of the Bluetooth System-Core. Version 1.1, Bluetooth SIG, Inc.
[13] S. Basagni, C.Petrioli:A Scatternet Formation Protocol for ad hoc Networks of Bluetooth Devices, Proceedings of IEEE VTC Spring 2002.