Location-Based Advertising Using Bluetooth

Mohamed Al-Arag and Jalal Kawash
American University of Sharjah, UAE
{b00013254|kawash}@aus.edu

ABSTRACT

Despite the huge adoption of wireless handheld devices that are Java- and Bluetooth- enabled, this potential E-Commerce sector has not been fully exploited, with the possible exception of gaming industry. This paper presents an E-Commerce system that implements advertising in shopping malls using Bluetooth, the short-range wireless communication standard. The system offers businesses a cheap yet effective alternative for electronic advertising. Furthermore, the system is solely *pull*-based, giving the users a harassment-free advertising experience. Our system makes use of a number of state-of-the-art technologies, including Java 2 Micro Edition (J2ME) and .Net.

Key Words: E-Commerce, mobile applications, Bluetooth, J2ME

1. Introduction

There is a constant increase in the adoption of wireless handheld devices, and this is paralleled with an increase in their capabilities. About half a billion users currently carry handheld phones that can run J2ME, and there are 150 mobile operators supporting Java [1]. Most currently available mobile phones and PDAs support Bluetooth. On November 14th of 2006 the number of shipped Bluetooth-enabled devices reached one billion [2]. However, except for the industry of mobile games, the E-Commerce sector is not fully exploiting this huge device availability [1], nor is it taking full advantage of the wide adoption of short-range wireless communication, such as Bluetooth.

Companies allocate substantial budgets for advertising, whether electronic or otherwise. Our aim is to provide a low-cost yet user acceptable advertisement model for shopping malls. Following the North American model, shopping in Europe, the Middle East, and other parts of the world is increasingly depending on the "hyper-shopping mall" model. Outlets in a shopping mall can place their promotions on a dedicated centralized server. Bluetooth access points can be deployed throughout the shopping mall, which gives customers the opportunity to pull promotions on their Bluetooth-enabled devices while in the mall.

Since Bluetooth covers a maximum range of 100 meters, the use of sufficient access points is a must. Users should have two options. The first is to select the offers that are in the same area as the user at the time of request; the location of the user can be simply determined through the access point being used. That is, the position can be approximated without resorting to expensive location-determination technologies, some of which do not work in closed areas anyway. The second is to allow the user to retrieve existing promotions pertaining to all the shops in the mall.

Similar applications are rare [3,4]. We came across extremely few such applications which are deployed in real life. However, our approach is fundamentally different than the existing ones. For instance, a representative 'model' application is deployed in Sultan Mall in Kuwait [3,4]. Unlike our approach which is *pull*-based, access points *push* advertisements to the customer devices. If a user rejects an offer, the user will be

deprived of receiving more promotions for one week. In addition, each user cannot receive more than three offers per week.

Realizing the limitations of similar existing solutions, the advantages of our approach are as follows:

- Users are solely in control of whether they would like to receive any promotions, when to receive them, and the number of promotions they would like to consider.
- Users do not have to switch their Bluetooth off to avoid receiving any offers.

Bluetooth is a standard for wireless personal area networks. Bluetooth connections are secure and unlicensed. Hence, Bluetooth communication is free of charge. While Wi-Fi (Wireless LAN) provides a stronger connection and wider spatial coverage, Bluetooth has several advantages over Wi-Fi. These include easier service discovery, less power consumption, and wider adoption in wireless handheld devices. Table 1 makes more comparisons with Wi-Fi and InfraRed standards.

The rest of the paper is organized as follows. In Section 2, we give the high-level design of our 4-tier system. These are the Bluetooth client, Bluetooth access point, application server, and data server. Section 3 highlights some of the implementation details. Section 4 concludes the paper.

Property	**IR**	**Bluetooth**	*WLAN*
Range	1m	10-100m	300m
Bandwidth	4Mbps	1.1 Mbps	11-54 Mbps
Power Consumption	Low	Low	High
Device's size	Small	Small	Large
Circumference of transmission.	30 ^{o}	Omni-Directional	Omni-Directional
Device discovery	-	Yes	-
Service discovery	-	Yes	-
Security	-	Yes	Yes
Adoption in mobile devices	High	High	Low
price	$1 USD	$20 USD	N/A

Table 1. Comparison of IrDA, Bluetooth and WLAN

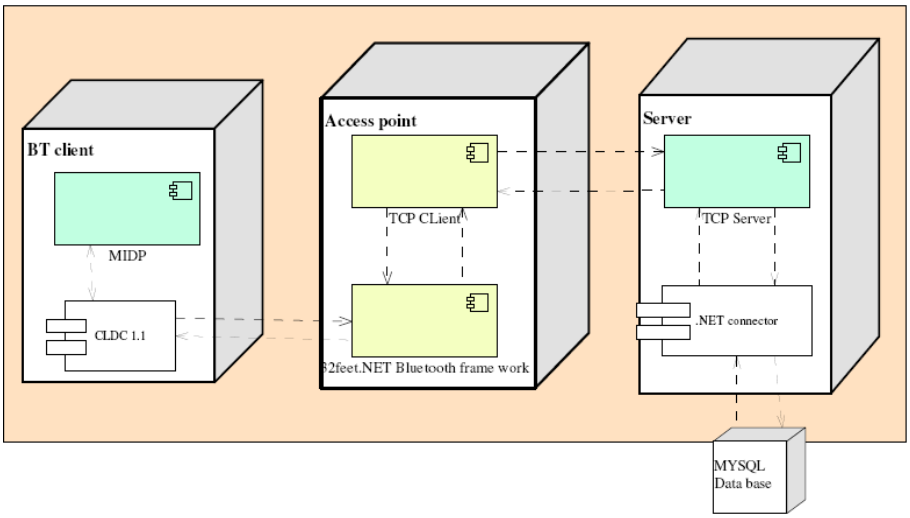

Figure 1. Architecture for Location-Based Advertising with Bluetooth

2. Design

Our system is developed using 2 platforms. J2ME is used in developing the mobile application (the Bluetooth client) and the .Net framework is used to implement the access points and the main server. A high-level view of the architecture is given in Figure 1.

2.1 Bluetooth Client

The Bluetooth client is developed as a J2ME client. The architecture of the client is given in Figure 2.

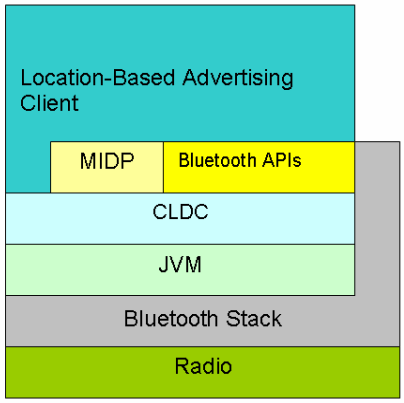

Figure 2. High-level Architecture of the J2ME Bluetooth client (adapted from [2])

J2ME: The J2ME architecture [5] involves *configurations*, *profiles*, and *optional packages*, which can be used by developers to construct a complete Java runtime environment that closely fits the requirements of a particular range of devices. The basic runtime environment is defined as a configuration, combining a virtual machine and a collection of core classes.

J2ME supports two configurations the Connected Device Configuration (CDC), version 1.1 known as JSR 218, and the Connection Limited Devices Configuration (CLDC), version 1.1 known as JSR 139. CLDC is targeted towards resource constrained mobile devices. The virtual machine HotSpot is used in CLDC 1.1. Configurations are complemented by profiles in order to provide support for application development and execution. Profiles provide Application Programmer Interfaces (APIs) for devices that share some capabilities.

In this work, the Mobile Information Device Profile (MIDP) is used. It creates the interface between the application and the Application Management System (AMS), under the CLDC configuration. MIDP is a set of Java APIs that provide a complete J2ME application environment defining user interfaces and communication methods.

CLDC: CLDC is intended for devices with limited hardware resources (CPU and memory). Typically, such devices run on either a 16- or 32-bit CPU with a

maximum of 512 Kbytes of memory available for client applications and the Java platform itself [6]. Most mobiles and smart phones fall into the CLDC category [5].

Bluetooth in J2ME: For the client to communicate with the access points using Bluetooth, the JSR82 v1.1 is required [7]. This API is designed to operate on top of the CLDC.

2.2 Bluetooth Access Points

The access point is implemented using C#.NET and therefore a framework is needed to have access to the Bluetooth stack through the .NET framework. The *32feet* open source project is used [10].

2.3 Application Server

The application server maintains the business logic for the durability of offers. It is backed by a MYSQL data server to store the relevant information. The application server communicates with the Bluetooth access points using (wired) TCP/IP.

3. Implementation

Figure 3 shows the steps involved in a successful interaction between the Bluetooth client and the remaining system components.

Bluetooth Client: The application starts by displaying to the user a welcome message asking to choose whether the user wants to receive offers in the range of the access point(s) he/she is close to, or all the offers that are available in the mall.

Device Management: The user interface is backed by a command listener. Once the user chooses an option, the command listener in the MainClient class detects the chosen command and acts accordingly. Regardless of the option chosen by the user, an object of type ServerFinder will be created and a dedicated thread takes care of it. The first thing that the server finder does is to have a reference to the MainClient Midlet (parent) so that it can update the user with the appropriate display.

The ServerFinder makes use of the needed class from the Java Bluetooth specification to manage the Bluetooth device, represented by the Java class LocalDevice. After getting the local device object, a reference to a discovery agent is created (`agent=local.getDiscoveryAgent()`); it is responsible for searching for Bluetooth devices in the client's range. The server finder implements the runnable interface; a new thread searches for Bluetooth enabled devices in the range (`agent.startInquiry(DiscoveryAgent.GIAC, this)`).

The ServerFinder class implements the DiscoveryListner interface. Each time the inquiry finds a device, the method DeviceDiscovered is called. In order to avoid being overwhelmed with other Bluetooth devices (customer devices), the application only reports the access point devices and ignores other devices. Once searching for devices is done, the application calls the method SearchForServices, passing to it a list of devices found in the search inquiry and the server UUID that are required. Finally, shops and offers are retrieved based on the business logic implemented in the application server. Since Bluetooth access points can support a maximum of seven concurrent connections, the connection is closed as soon as possible.

Access Point: An access point is implemented through the use of a Bluetooth server that communicates with the application server using TCP/IP connections. A Bluetooth server publishes services to clients so that they can look them up and use them. For a client to have access to that service, the Bluetooth server must register the service in the service record and then start listening for connections to this service.

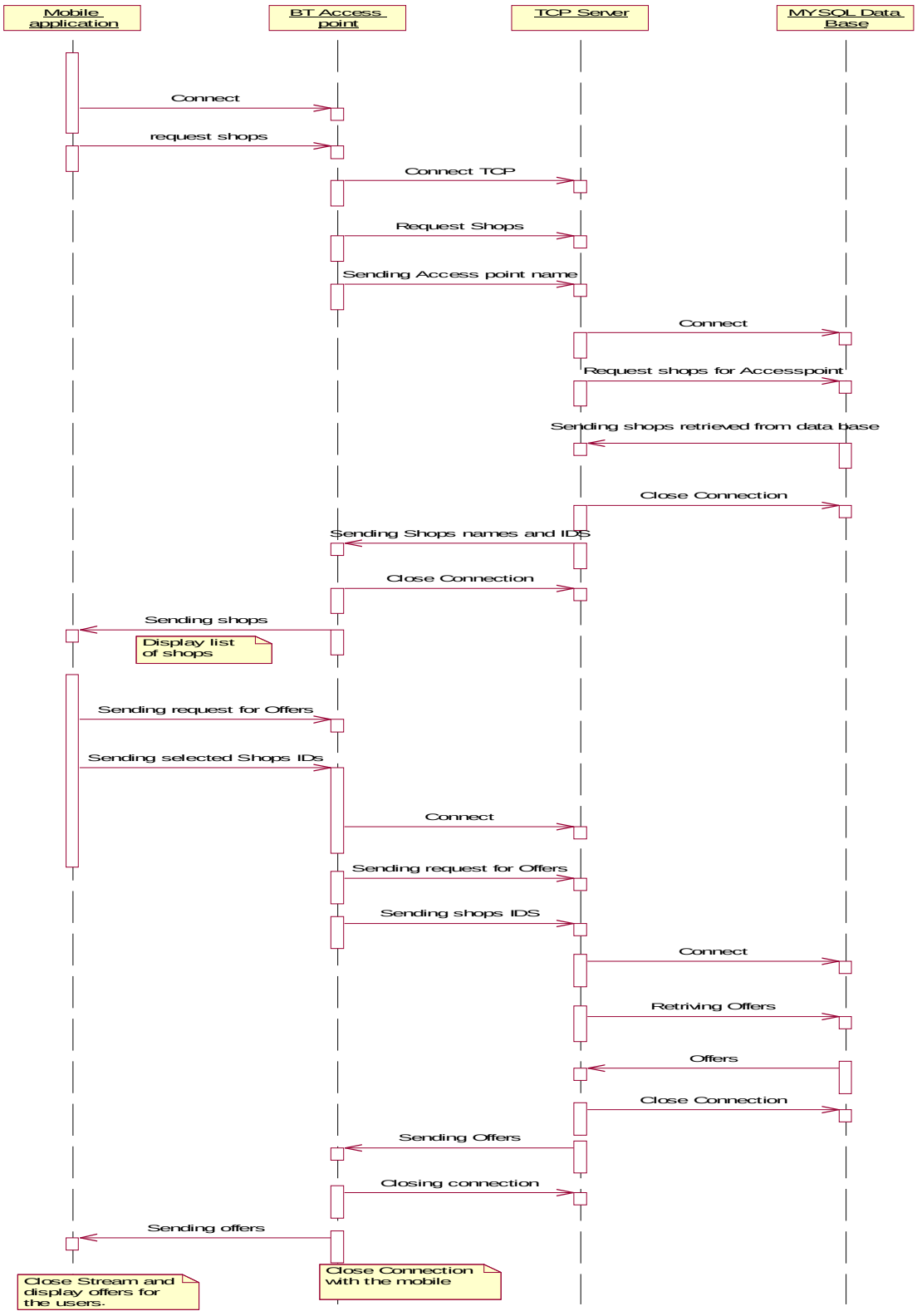

Figure 3. Sequencing Diagram for a successful attempt to retrieve promotions

Service Registration: Before a Bluetooth client device can use service discovery on a Bluetooth server device, the Bluetooth server needs to register its services internally in the Service Discovery Database (SDDB). This process is called service registration. To set up a Bluetooth service to be available to clients, the following steps are followed:

1. Creating a *service record* for the service That is to be made available .
2. Adding the new service record to the SDDB.
3. Registering the service.
4. Waiting for incoming client connections.

Once a client connects to the access point, a thread will be assigned to the client for handling his/her requests. The thread instantiates an object of type RequestHandlerThread and passes to it the Bluetooth client object along with the name of the access point itself. Access points are differentiated by assigning to them unique IDs (such as a MAC address). A connection to the main server is created through an object of type ClientToMainServer which handles dealing with the database, and then responding to the access point which relays the response to client. This thread waits for client requests, which are limited to four kinds:

1. Request for shops in the range of that access point.
2. Request for all shops in the mall that have offers.
3. Request for offers from shops selected by the customer in any of the above steps.
4. Request to close the connection.

The access point relays the client's requests to the application server.

Application Server: The server is responsible for satisfying client's requests through the access points. Each time an access point connects to the server, a thread takes care of the request, which is an object of type RequestHandlerThread. The server is responsible for accessing the database and retrieving the required information from it to reply back to the access points, which in turn pass the response to the Bluetooth client.

4. Conclusion

Our preliminary testing for the system shows good performance results. The system spends 30 seconds at most retrieving promotions from 15 shops. Most of this time is consumed by device and service discovery. The simplicity of design is crucial for the success of such applications. We believe that Bluetooth-based applications offer great potential for profitable business models and this area of E-Commerce is barely explored.

A future extension of this work is to take advantage of the peer-to-peer (P2P) potential [8], in order to create virtual market places using Bluetooth. Users can advertise the items they wish to sell to other Bluetooth users. With current technology, such an implementation depends on the use of powerful servers. As the capabilities of handheld devices improve, such E-Commerce applications are expected to become widely available.

References:

[1] Mobile Monday Web Site, Apr. 2006. http://www.mobilemonday.com

[2] Bluetooth Web Site, Dec. 2006, http://www.bluetooth.com/Bluetooth/SIG/Billion.htm

[3] Two Forty Eight AM Web Site, Dec. 2006, http://www.248am.com/mark/kuwait/bluetooth-advertising/

[4] Two Forty Eight AM Web Site, Dec. 2006, http://www.248am.com/mark/interesting/about-bluetooth-advertising-in-kuwait/

[5] Java 2 Platform Micro Edition. http://java.sun.com/j2me/index.jsp.

[6] Sam's Publishing, Dec. 2006, http://www.samspublishing.com/articles/article.asp?p=25082&rl=1

[7] Sun Microsystems, Dec. 2006, http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth2/

[8] Daniel KÄappeli, "JXTA over Bluetooth," in Information and Communication Systems Research Group . pp. 10, 16, 2003.

[9] Kumar, Paul J.Kline and Timothy J.Thompson, *Bluetooth application programming with the Java APIs*, Oxford, England: Morgan Kaufmann publishers, 2004.

[10] 32feet, http://www.32feet.net