Empirical Performance Evaluation of JSR-82 Oriented Bluetooth Piconet Systems

Sami Ibrahim Serhan                            Omar Yahya Tahboub

Department of Computer Science, University of Jordan

Abstract

This paper aims at evaluating the empirical performance of JSR-82 oriented systems that operate in Bluetooth piconet environments. A general purpose layered JSR-82 oriented software architecture together with a server-initiated communication model are proposed. Their performance was evaluated empirically. Results revealed that JSR-82 oriented systems achieve the required performance for indoor legacy Bluetooth applications, which justifies the fact that they are adopted by the leading wireless solution enterprises.

Keywords: Bluetooth piconet system, Performance evaluation, Communication model, Client/Server communication, Point-to-point communication

1. Introduction

Bluetooth is an emerging computing and telecommunication specification that describes how mobile phones, computers, Personal Digital Assistants (PDAs), and many other Bluetooth-enabled devices can be interconnected via wireless (radio) transmission [5,25]. This protocol has become an important promising standard for wireless integration of small devices due to its low-cost, low-power, and short range technology [9,16]. Since late nineties, this wireless technology has been realized and adopted by mobile and wireless technology vendors [2,16,17,18,21,22,24]. Bluetooth system software, which is still being developed, implements the actual communication model among the interconnecting parties [26] and manages the
interconnection between the application and Bluetooth radio hardware. Consequently, most of the Bluetooth vendors provide system and application software required to operate their Bluetooth-based products. This sort of software is either implemented in native languages, such as C/C++ or virtual machine-based languages, such as Java [1]. In addition, Java programming language is currently considered a strong and an optimal language for cross-platform wireless connectivity solutions. Therefore, providers [2,18,22,24] supply the Java-based Bluetooth system and application software that suits for mobile devices and desktop computers. The specification of Java-based Bluetooth was provided for a Java API for Bluetooth Wireless Technology (JABWT) by the Java Community Process. This specification is published under the title JSR-82 [19,20].

Modern Bluetooth-based wireless applications that employ Java have two main concerns: the utilized communication model [26], and the related performance issues.

In Bluetooth, there are two topologies: piconet and scatternet. When two Bluetooth units share the same channel they form a piconet [5]. A piconet is the basic Bluetooth network topology which ranges in size from two to eight Bluetooth-enabled connected devices [5,13,25]. One of them acts as the server of the piconet, and the remaining act as clients [5,25]. Multiple piconets with overlapping convergence areas form a scatternet [5,25].

2. Related Work

The interest in Bluetooth wireless technology started in 1994, when a special study was launched by Ericsson. This study aims at investigating the low power, and low cost radio interface between mobile phones and their accessories. In 1998, the first Bluetooth chip came out and in the same year the Bluetooth Special Interest Group (SIG) [5] was founded by Ericsson, IBM, Intel, Nokia, and Toshiba [1,16,17]. Moreover, Java has been
recently employed by significant wireless communication enterprises such as Nokia and Ericsson [18,24]. This adoption has led to formulating a special expert group called Java Specification Request 82 (JSR-82) [16,17]. The literature related to Java Bluetooth system software was reviewed in [6,7,11,15,19,23]. In [7,19,23], a JSR-82 compliant software architecture was designed and implemented.

Piconet performance can be measured via three different methodologies, the first of which is the analytical methodology [8,10,14,20] that aims to model the Piconet and evaluate its performance by utilizing a set of deterministic and validated mathematical equations. The second methodology is the simulation methodology [2,3,19], which expands the previous methodology by modeling the mathematical model of a given piconet into a discrete–time simulation environment in order to gain more realistic outcomes.

Finally, the empirical methodology [6,7,13,27] is concerned about evaluating the piconet performance in reality, by means of implementing it in terms of the required hardware, software, utilized topology, and performance. Moreover, this methodology achieves the nearest vision about the performance of any Bluetooth-based environment by measuring its concerned performance metrics as achieved throughput, connection times, and connection delays [12].

3. The Proposed System Software Architecture

In most Java-based networking systems, the system architecture is found to be either hierarchical or layered as observed in [1,22,24]. The architecture implemented follows the layered design, which guarantees a secure level of integrity and concurrency of multiple communication operations. For instance, it is observed in [1] that the process of
establishing a Bluetooth connection requires a remote device discovery, services discovery, and finally a connection setup. By dedicating a special layer for each operation, the application utilizing that architecture gains flexibility of implementing its specific operation concurrently with the previous Bluetooth-related operation, preserving the-up-to-date information about the surrounding. Moreover, it is also observed in [23] that a Bluetooth device can be connected up to seven remote devices. In order to gain a communication deadlock-free application, a special layer must be contained to impose scheduling constraints among the established connections.

3.1. Design

The general design of the proposed JSR-82 oriented software architecture is presented in Figure 1. The layers of this architecture fall into three categories: the Bluetooth system Java API [5,25], the Bluetooth JSR-82 compliant stack [23], and the Bluetooth native MSI Bluetooth USB adapters.

![Figure 1: The Proposed Software Architecture](image-url)
3.2 Implementation

The software depicted in Figure 1 is intended to operate as a complete class library above the JSR-82 compliant (JABWT) stack. This system is implemented in Java under Linux environment (Red hat 9). In addition this system is implemented as a property of KASIT at the University of Jordan, and the JSR-82 complaint stack is licensed from Rococo Soft as a special release for research purposes. The implemented system API is composed of the following classes arranged according to the deployed package directories.

- **edu.uj.kasit.cs.bluetooth.piconet.physical**: This package directory contains a class called *PhysicalConnection*. This class is concerned with activating and deactivating the native HCI layer, and the native Service Discover Protocol (SDP) layer.

- **edu.uj.kasit.cs.bluetooth.piconet.device**: This package directory contains a single class called *LocalDeviceManager*, which manages the local Bluetooth device, discovery of remote devices, discovery of services on discovered remote device, and Bluetooth connection management.

- **edu.uj.kasit.cs.bluetooth.piconet.discovery**: This package directory contains the classes concerned with discovery.

- **RemoteDeviceDiscovery_Engine** is concerned with accessing remote device discovery agent of the (JABWT) layer in order to detect accessible remote Bluetooth devices in the surrounding.

- **RemoteServiceDiscovery_Engine** is concerned with remote service discovery on the discovered remote devices.
• **edu.uj.kasit.cs.bluetooth.piconet.connection**: This package directory contains the classes that deal with the actual Bluetooth connection using the JSR-82 complaint GAP and SPP profiles. This directory contains three classes: **BluetoothConnection_JSR82** class, which invokes the remote device discovery and remote service discovery engines, and manages the initiated GAP and SPP connections (sending and receiving data). **GAPConnection_JSR82** class, used for initiating and managing a GAP connection with a connected remote device. And **SPPConnection_JSR82** class, is similar to the previous class, but it is dedicated for SPP connections.

• **edu.uj.kasit.cs.bluetooth.piconet.routing**: This package directory contains the most essential class. **MessageRouter_JSR82** class is the message routing layer which initiates, manages, and schedules GAP or SPP connections using different non-preemptive scheduling policies like FIFO, LIFO, priority-based and Round Robin scheduling techniques.

**4. The Proposed Communication Model**

The proposed communication model is based on the client/server model [1,17,24], thus any application using this model employs a single server and up to seven clients (endpoints). Moreover, this communication model is implied into the proposed software architecture, therefore this communication model is achieved through it. Recalling back the denoted communication scheme is server-initiated, where the server side is responsible of several tasks including polling clients for communication, sending and
receiving messages, and routing messages among the connected clients. According to the communication scheme and the JSR-82 specification, the server side does the following operations: Remote Device Discovery, Remote Service Discovery, Communication Setup (including GAP and SPP connections), and Message Routing. On the other hand, the client side (endpoint) does two operations: Opening a connection (by registering a communication service), and maintaining the connection with the server side.

4.1. Communication

A finite state machine for the operation of each side operation is provided based on the JSR-82 specification and the proposed system architecture.

4.1.1 The Client Side

Figure 2 presents the complete state machine diagram for the client side operation. This diagram for the client side consists of six states, ordered according to the proposed software architecture and Bluetooth communication scheme described in the JSR-82 specification [17].

4.1.2 The Server Side

The server side state machine consists of seven states, by which the established piconet is managed on the basis of the JSR-82 specifications. These states are depicted in Figure 3.

4.2 Illustration of Complete Client/Server Communication Scenario

To clarify the overall vision of the proposed model, a complete communication scenario for a Bluetooth piconet of two nodes: a single client (C1) and a single server (S) is demonstrated in Figure 4. Moreover, the utilized topology is demonstrated along with the communication transactions based on the operation state machine shown in Figures 2 and 3. The complete point-to-point communication scenario is illustrated through figure 4 (a)
to (f). In step (a), both of \((C1)\) and \((S)\) are remotely visible, which implies that both sides

Figure 2: The Client Side Operation State Machine

Figure 3: The Server Side Operation State Machine
have activated their Bluetooth physical connections. In step (b), \((S)\) activated its own remote device discovery to be in the Remote Device Discovery state, while \((C1)\) registered a communication service to be in the Connectable state.

In step (c), after \((S)\) had completed the process of remote device discovery and discovered \((C1)\), it immediately entered the Remote Service Discovery state. However \((C1)\) still in the Connectable state. In step (d), after \((S)\) completed the remote service discovery process on \((C1)\), it transited to Receive Only state, then \((S)\) directly polled \((C1)\) for communication by sending it a JOIN message.

Once \((C1)\) received the join message, it immediately responded by an acknowledgement message. In step (e), the server side added \((C1)\) connection to its message router, then entered the Piconet Communication Loop, and lastly sent \((C1)\) the send token in order to allow it to send data message to \((S)\). Finally, in step (f), after \((C1)\) received the send token from \((S)\), \((C1)\) decided if it has data to send it would send it, otherwise it would send an acknowledgement message to inform \((S)\) that it had no data to send.

5. Empirical Performance Evaluation

The empirical performance of a JSR-82 oriented system provides a crucial indication about its quality and applicability for Bluetooth-based applications. The performance of a Bluetooth-based system is evaluated empirically by: connection times, throughput, and interference with other wireless technologies like Wi-Fi IEEE 802.11b [16,25]. Since a JSR-82 system is a Bluetooth system [16,17,18,22,23,24], its performance is evaluated consequently by the similar performance metrics mentioned. Moreover, the usage model of JSR-82 oriented Bluetooth applications falls into: cable replacement usage as Dial-up networking, LAN access, and File transfer; and wireless audio usage as Intercom, and
Figure 4: (a)-(f): An illustrated scenario of Bluetooth piconet formulation and data communication using the proposed model.

Cordless telephony [1,16,17,25]. The performance evaluation scope in this paper is motivated to the cable replacement usage model of JSR-82 oriented Bluetooth applications through the proposed software architecture and communication model. Two main performance metrics are crucial for evaluating their performance: the relation of the average throughput with connection period, separation distance, and the effect of the proposed software architecture on the achieved average throughput. By referring to the nature of the wireless cable replacement applications, it can be observed that the separation distance between the two interconnected point plays a significant role. The relation of the average throughput with separation distance is obviously the achieved
average throughput at a determined separation distance. In order to determine their efficiency and reliability, the effect of the proposed software architecture is defined to be the effect of the difference between the average throughput achieved with the absence and the presence of the proposed software architecture.

5.1 Performance Evaluation Scenario

The cable replacement applications are naturally uses point to point connection, so a special JSR-82 oriented echo point-to-point application was implemented for the link throughput evaluation purposes.

A closed 4-by-4 meters research room was utilized to serve as an experimental Private Area Network (PAN) topology as shown in Figure 4, and two IBM Pentium 4 Netvista PCs plugged with MSI Bluetooth USB adapters were used for implementing the Bluetooth wireless connectivity. In addition, the native stack determined the Bluetooth wireless connectivity type to be ACL connections, and also determine the default packet type to be DM1 and DM3 [4].

In order to evaluate the throughput relationship with separation distance this test scenario was decomposed into three categories: one, two, and four meters separation categories. In each category, test application was run ten times; where in the first time lasted for 200 seconds tell the tenth which lasted for 2000 seconds. In each two seconds a complete communicational iteration is performed.

5.2 Point-to-Point Throughput versus Operation Time

In this section, the relation of point-to-point throughput versus operation time is investigated. It was previously stated that in each category, the test application was executed for 10 times, each for 200, 400, 600… 2000 seconds respectively. The
throughput for each side was empirically evaluated in terms of the income and the outcome data traffic: server to client data traffic, and client to server data traffic. It is observed that the proposed software architecture together with the proposed communication model are implemented as a class library mounted over the JSR-82 core class libraries, this is clearly seen in Figure 1. As a consequence, this class library logically may add processing time overhead to the applications using it. Therefore, in order to determine the reliability and the efficiency of both of the proposed software architecture and communication model, the throughput was evaluated under two assumptions: the presence of the proposed software architecture, and the absence of it. The former is considered to be the actual throughput achieved, and the latter is considered to be the optimal that can be achieved. The difference between the two findings states is the overhead resulting from them.

Figures 5, 6, and 7 illustrate the relationship between the average throughput with operation time for 1 meter, 2 meters, and 4 meters separation distance respectively for both data traffic directions in terms of the optimal and actual average throughput achieved, and the defined overhead.

It was noted that in general in all presented categories that the average throughput of server to client data traffic is higher than the client to the server data traffic, which indicated the asymmetry of the link. It was also observed that the achieved average throughput converges to the data rate achieved DM1 and DM3 packets under native Bluetooth environments [4,5,12].

It is also noticed that the average throughput degraded gradually with operation time as well as the separation distance is doubled in each presented category. it is obviously
shown that the overall average throughput decreased by 20% for the client to server traffic and 25% for the server to client traffic in Figure 5, 17% and 25% in Figure 6, and 17% and 11% in Figure 7. The overall overhead on throughput is 2.8% for the first category in Figure 5, 4.8% for the second category in Figure 6, and 2.4% for the third category in Figure 7. Results revealed that the overhead resulted from both of the proposed software architecture and the communication model on the achieved average throughput is almost negligible if considered.

5.3. Point-to-Point Throughput versus Separation Distance

In this section, the overall relationship between the average throughput and separation distance is stated. Similarly to the previous section, the same measurements of average throughput were evaluated. The overall relationship between the average throughputs versus separation distance is shown in Figure 8. Figure 8 (a) describes the client to server traffic average throughputs for all stated categories, while figure 8 (b) describes the server to client traffic. The overall relationship between the average throughputs versus separation distance was evaluated by executing the echo application five times each of 2000 seconds iterations per category, then the average of each category was taken.

Figure 8 (a) shows that the average data throughput falls as the separation distance doubles especially when the distance was doubled from 1 to 2 meters. However, the same effect is also noticed in Figure 8 (b), but with less decrease, which again states the link asymmetry achieved through DM1 and DM3 packet used by the Bluetooth radio. The observed average throughput degradation resulted from doubling separation distance is due to two main factors: Bluetooth is short-range low power wireless technology so as a consequence, the signal strength is affected by distance; and the fact that JSR-82 oriented
systems are naturally operate under Java virtual machine, which consumes higher amount of memory than native languages as C/C++. Finally, it is also noticed the insignificant overhead of the proposed software architecture together with the proposed communication on the achieved average throughput for both data traffic directions.

Figure 5: Average Throughput versus Operation Time of one-meter category (a) client to server traffic, (b) server to client traffic.

Figure 6: Average Throughput versus Operation Time of two-meter category (a) client to server traffic, (b) server to client traffic.
Figure 7: Throughput versus Operation Time of four-meter category (a) client to server traffic, (b) server to client traffic.

Figure 8: The overall relationship between the average throughput versus separation distance: (a) client to server traffic, (b) server to client traffic.

6. Summary and Conclusions

The performance of JSR-82 oriented systems was evaluated empirically through the proposed software architecture and the proposed communication model. The performance was evaluated in terms of the relationship between the average throughput versus operation time and versus separation distance for both data traffic directions. It was
observed from the empirical results shown in section 5 that the proposed JSR-82 oriented software architecture and communication model suit for Bluetooth cable replacement applications, especially when the client side receives more than it sends. Typical examples include: FTP, Multimedia, and also Bluetooth-based legacy applications, which operate at short ranges of 1 to 4 meters, and short connectivity time periods of 30 minutes. In addition to its suitability and applicability, JSR-82 oriented systems also provide the suitable infrastructure for standard Java-based Bluetooth applications for small and medium size computers due to its extendibility of building application-specific software architecture as well as communication models.

Finally, depending on the observed applicability and suitability of JSR-82 oriented systems, it is concluded that the JSR-82 verifies its adoption by leading wireless solution enterprises, and promises to provide a wider future for Java-based Bluetooth systems operating on more extensible topologies as scatternets.

7. References


http://www.iks.inf.ethz.ch/publications/files/jxtabt03.pdf


[16] Java Community Process, JSR-000082 Java APIs for Bluetooth
http://www.jcp.org/aboutJava/communityprocess/final/jsr082

[17] Motorola. Java APIs for Bluetooth Wireless Technology (JSR-82) version 1.0a. Austin, 2002. Received from:
http://www.jcp.org/aboutJava/communityprocess/review/jsr082/.

http://www.forum.nokia.com/main/1,6566,015_291,00.html


[20] Perkinen, J. ‘Group awareness in Bluetooth’ Department of Mechanical Engineering, Helsinki University of Technology, Finland, 2000. Received from:


