Investigating Simultaneous Wireless Connections for a Quiz Management System-A Case Study

Michael Hosein

Received: 15 December 2019 Accepted: 4 January 2020 Published: 15 January 2020

Abstract
Near Field Communication is a set of communication protocols for communication between two electronic devices over a distance of 4 cm or less and Bluetooth is a wireless technology standard used for exchanging data between fixed and mobile devices over short distances using UHF radio waves in the industrial, scientific and medical radio bands, from 2.402 GHz to 2.480 GHz, and building personal area networks (PANs). Both these protocols facilitate wireless/internet less communication between devices that have the capabilities. The paper titled "Establishing Simultaneous Server-Side Connections for NFC/Bluetooth enabled Quiz Management Systems" further expanded the concept and usage of these protocols via the examination of modifications made to a Quiz Management System. It highlighted the QMS which was further developed to address a key limitation that was observed in the prior system.

Index terms—quiz management system, smart classroom, protocol, bluetooth, NFC, UHF.

1 Introduction

The developed Quiz Management System (QMS) of this thesis can be seen as an iterative step from the former QMS model known as BlueQ, which was a Bluetooth Based Quizzing application. The quizzing application was able to allow communication between Server and Client devices via the Bluetooth protocol to allow the transferring of data from one device to another. The generalized functionality of the Client (student) device being, the submission of completed quiz material for marking and the receipt of same from the server device. Moreover, the functionality of the Server (teacher) device was primarily for distributing quiz material to students, receiving quiz material from students, performing data analysis on the information received and subsequent distribution of marked material.

The paper "Establishing Simultaneous Server-Side Connections for NFC/Bluetooth enabled Quiz Management Systems", elaborated on the additional functionality of the implemented QMS system of this thesis. It further shed light on the 2 approaches adopted by the QMS to facilitate more simultaneous server-side connections from a high-level perspective. This paper is aimed specifically at providing the technical analysis and testing details that the improved QMS would have entailed. A breakdown of results from each testing iteration would be discussed whereby the most efficient approach would be deduced based on empirical evidence.

2 II.

3 Literature Review

This Chapter would have already been sufficiently covered in the paper "Establishing Simultaneous Server-Side Connections for NFC/Bluetooth enabled Quiz Management Systems". As this paper’s main objective is to compliment the work of the aforementioned conference paper, the Literature Review would Cover all approaches undertaken in the Thesis QMS Solution. Details of this can be viewed in the below section.
6 Approach 1-Version 2 (multi-channel -Identical UUIDs)

This approach explored the route of adding additional RFCOMM channels that are associated with the same UUID. It mimicked the concepts of Client/Server TCP/UDP communications (Lei Wang et al., 2000) having the server socket wait for connection from a client socket, whilst a listening socket is activated for receiving new connections and mapping unto the server socket. In this concept, if we were to assign maximum of 2 channels per UUID, we would accomplish simultaneous connections for data transfer to 2 x 8 client devices at a time. The below Sequence Diagram Shows the flow of events in Approach 1-Version 2: By merit of how RFCOMM works it is stated that it would support a total of 60 simultaneous connections based on the UUIDs assigned to each of those channels. This UUID is what the client uses to firstly identify the channel it wishes to connect to via its SDP call, then a connection is established, but since RFCOMM is a serial communication protocol, it would only allow 1 connection at a time per channel, there is no multiplexing unless you switch from a serial protocol to a parallel protocol. Going further into this, the proposed modification could see the sharing of UUIDs for 2 channels at a time, 2 channels being the upper limit. Given that all the necessary provisions are made available, and the user has the optimum storage and processing power to host 60 simultaneous connections, this method of dual channeling per UUID could possibly push the amount of simultaneous connections to "120" total. Now this is of course the perceived amount of simultaneous connections, however it would still be recorded as 60, since half the figure represents Unique (UUIDs) and the other a replication.

But that is observing from a holistic point of view in accordance with the protocol’s specifications. As we apply this logic to the presented solution, we can now seek to address the inefficiency of having limited amount of channels used at any given time for material to be distributed by the server device or submitted by the client device. Consider the current application’s limit of 8 channels per server device. That alludes to the point that there are 8 UUIDs that are hardcoded into the lowlevel code of the server device by which the RFCOMM channels can be accessed by. The solution being proposed now assigns a total of 2 channels per UUID. This therefore raises the amount of allowable connections to the server by +8 giving a total of 16 connections at a time per server device. The below figure shows the results of sharing UUIDs as more than the maximum limit of 8 devices were able to connect. If in its strictest sense the server device must maintain 8 channels irrespective of UUIDs. Then a queue for each channel with an allocation of 1 allotment for a client device would be established, since the UUID would refer to the channel number in which the service is being provided, the queue would be able to guarantee that a connection has already been made with the channel, however it is currently in use and thus, once completed it will become available.

This guarantee stands on the basis that based on the UUID both the client devices would be treated as one device attempting to access the same service. However, it would all come down to which device connects first. Connecting to the server first would be based on which of the 2 devices is nearer, with the device closest, being successful in connecting to the channel and being able to access the channel and the device furthest from the server being successful in connecting to the server but placed into queue. The below figure shows the results of queuing clients with the same UUID. This functionality ensures that the client will receive the quiz content without having to re-attempt connecting to the server multiple times. Thus, this method is a viable option for improving the efficiency of Approach 1.

7 Approach 2 -Delegate Function()

Approach 2 -Given that the server device is mainly responsible for the distribution of quizzing materials and can only supply material to up to 8 devices at a time. The proposed solution would see the modification of the existing system to include the 'delegation' functionality. The delegation functionality works as such: Server Device connects to 8/100 devices at a time, given that the assumed sample/classroom size is 100 students. Then in an effort to increase the efficiency of the system, the Server device can select one of the connected client devices and elevate their privileges or rather give/delegate new functionality to allow the distribution of quiz material. Assuming that the selected client device is able to support the same amount of connections as the server device, this therefore means that a total of 16/100 devices can connect to receive quiz material at a time. If the number of client devices 'n' that is given Server privileges increases, then the rate in which quiz material can be distributed would take on the form of an exponential curve, thereby increasing the efficiency of the Bluetooth quizzing system. This approach when compared to the first approach has some immediate advantages: It does not affect the performance of the original hosting device but rather acts as an extension of its functions. What must be taken into consideration is the UUID generator function in which random UUIDs are generated and are hardcoded as the UUID’s assigned to the available channels for clients to access their services. This method must ensure that the function delegate() is triggered upon the original Server Device selection of the option to Delegate. As it is an extension of Blueq the interface would consist of the below options. In theory this additional functionality
can be likened unto a wi-fi peer to peer network as was postulated by developer.android.com, October 2020. It is important to recall that the core concept of a peer-to-peer network is to partition tasks or workloads between pairs. In this concept peers are equally privileged, similar to the work done by (Sewook Jung et al., 2007). However, in this modification of the system, the server device, determines the functions to send to the client device and thus plays an integral role in determining the amount of privileges a selected client device is afforded. The below Sequence Diagram Shows the flow of events in Approach 1-Version 2:

8 Modeling and Testing
In this section the model or methodology for determining the efficiency of each system as well as which one performs the best comes about upon analyzing the test results for each Approach and comparing the top performing approach’s performance, with that of the former BlueQ system.

9 Results Summary:
Based on all previous tests that were conducted, Approach 2 performed with the highest level of efficiency and scalability when compared to the previously developed systems and the other approaches tested. The overall performance can be viewed in the below figure 13.

10 Conclusion
In Conclusion, three solutions/approaches were developed and implemented to solve the problem of lack of popularity or usage of Bluetooth/NFC Quiz systems within the context of a University Ambient. These three solutions, saw the usage of the concepts of sharing UUIDs to achieve a greater level of connectivity to the server device as well as the concept of delegating responsibilities from the server device to the client device in an effort to distribute channel sharing load, thus making even more RFCOMM channels available and elevating the privileges of selected clients. Extensive Testing was carried out to analyze the performance of each approach. Additionally, subsequent testing was conducted on the most efficient solution method against the previous BlueQ model to obtain essential comparative analysis data on the advantages and shortcomings of both systems.
Figure 1: Figure 1:
Figure 2: Connected Devices - (9)

- SM-G970F Channel 1
- VIBE K5 Note Channel 2
- Galaxy J2 Core Channel 3
- Galaxy A5 (2017) Channel 4
- SM-A102U1 Channel 5
- ASUS_1001D Channel 6
- Galaxy A5 (2017) Channel 7

Connected

10.34257/GJCSTEVOL20IS4PG1
Figure 3: Figure 3:
Figure 4: Investigating

| Approaches          | Total Amount devices that can be supported in an instance | Average Device Connect Time/s | Total Connection Time/s | Total Amount devices that can be supported |
|---------------------|----------------------------------------------------------|-------------------------------|-------------------------|-----------------------------------------|
| Approach 1 Version 1 - Shared UUIDs and Multi RFCOMM | 16                                                        | 2.403                         | 18.45                   | 16                                      |
| Approach 1 Version 3 - Shared UUIDs and RFCOMM Channel | 16                                                        | 2.574                         | 111.78                  | 16                                      |
| Approach 2 - Delegate | 64                                                        | 2.326                         | 10                      | 64                                      |
| Previous System - Blue Q | 5                                                         | 2.38                          | 37.3                    | 5                                       |

Figure 5: Investigating
Figure 9: Figure 8:

![Graph showing device connection times](image)

**Figure 10: Figure 9:**

| Server Device Results |
|-----------------------|
| **Server** | **Device** |
| ULD No. | Channel No. | Device No. | Device Connect Time/s | Disconnects |
| 1 | 1 | 1 | 1.1 | 0 |
| 2 | 1 | 2 | 2.1 | 0 |
| 3 | 2 | 3 | 3.2 | 0 |
| 4 | 3 | 4 | 4.1 | 0 |
| 5 | 4 | 5 | 5.2 | 1 |
| 6 | 5 | 6 | 6.3 | 1 |
| 7 | 6 | 7 | 7.3 | 0 |
| 8 | 7 | 8 | 8.4 | 0 |

| | **Total Connection Times/s** | **Average Device Connect Time/s** | **Average Time in Queue/s** | **Total Disconnects** |
|---|---|---|---|---|
| | 11.6 | 2.38 | 5.5 | 2 |

[Not available] not available
Figure 11: Figure 10:

Figure 12:
Additional testing was conducted with Approach2 and the Previous System to further analyze performance, whereby the below results show the substantial advantage the implemented Approach2 had over the previous system known as BlueQ.

A series of four (4) experiments were conducted on both the BlueQ system and Approach2 of BlueQ2. Whereby, each experiment saw an increase in the amount of client devices requiring connection to the server device. As shown in the table above, Experiment 4 highlighted the considerable improvement Approach2 provided, as it did not require multiple cycles to facilitate the client devices but rather, only needed to delegate responsibilities to already connected client devices which an upper limit of 8 was given, this meant that a total of 8 x 8 client devices could have been simultaneously connected, once registered with the system.

The shortcoming of the previous BlueQ system was that it only allowed a total of 5 devices at a time to connect simultaneously, but for the purpose of the comparative analysis its upper limit of 8 devices were