Int. J. Communications, Network and System Sciences, **2010, 3, 311-320**
doi:10.4236/ijcns.2010.33040 Published Online March 2010 (http://www.SciRP.org/journal/ijcns/).

Scheduling Mobile Data Services in a Bluetooth Based Platform

Xiaoyu Liu, Kin Choong Yow
School of Computer Engineering, Nanyang Technological University, Singapore City, Singapore
Email: *liu.xiaoyu.mac@gmail.com, kcyow@ntu.edu.sg*
Received November 4, 2009; *revised December* 10, 2009; *accepted January* 8, 2010

Abstract

Public buses play an important role in public transportation in most parts of the world and it is still the dominant public transportation mode in some regions. Nowadays, as people switch to a mobile lifestyle, they spend significant amount of time on the traveling to work, back and forth. However, not much research has been done on how to provide some on-board service for those commuters in the public bus. This paper presents a Bluetooth-based system which is inexpensive, yet flexible, and scalable to serve commuters in a personalized manner using Bluetooth enabled mobile phones. However, from the Bluetooth specification, one Bluetooth dongle can connect to at most seven other Bluetooth devices. As we expect more than 7 users to use the services provided in the Bluetooth-based system (a full double-deck bus can carry around 100 passengers), we need to work out an effective scheduler to schedule all the private services on the Bluetooth servers in the bus. This paper also describes a scheduling algorithm that exploits the park mode feature of the Bluetooth specification to allow more users to have access to the Bluetooth services on the bus.

Keywords: Mobile Device, Bluetooth, Public Bus, Data Service

1. Introduction

We have transited into the Information era, and most people are eager to get various kinds of information in convenient ways. However, due to the increasingly mobile and busy lifestyle, people are having lesser time for some daily activities e.g. reading the newspaper, checking online news, enriching themselves by casual chat or relax themselves with games. Moreover, due to the large distances and the increasing of mobile lifestyle, people are using public transportation very frequently. After an observation on the public bus or MRT in Singapore, we found that many passengers are killing time by playing some casual game or reading some text content with their cell phones or PDAs during the journey. Quite few people will use GPRS to browse news, download some audio or video clips or go for online chatting due to the cost consideration. We can see that the on-board information and entertainment system in the public transportation system is required.

Several proposed systems exist in providing integrated support for commuters in using a public transportation system, such as e-ticketing and navigating through the complex metro system through mobile phones in Japan [1], and bus routes and schedules information display in bus stations in Mexico (EMI system [2]). Some others extend the public transportation routes and scheduling information to be included on a travel planner application on mobile phones, such as TramMate [3] in Australia and Buster [4] in Denmark. These systems, although helpful for commuters, do not address the 'idle-ty' of commuters while on the ride.

Some other systems address this issue by providing infotainment experience on-board the public transportation, such as the system proposed by Lin and Chang [5], ALSTOM [6], and the system deployed on French TGV trains [7]. These systems include a LAN on-board the public transportation. This LAN is connected to the outside world through either cellular networks or satellite connection. Although these systems are able to provide data live from the Internet while on the move, the requirement of direct connection to the cellular networks (3G/3.5G) or satellite network is still considered expensive nowadays.

This paper presents the BlueBus System, a system which is inexpensive, yet flexible, and scalable to serve commuters in a personalized manner using Bluetooth enabled mobile phones. However, from the Bluetooth specification, one Bluetooth dongle can connect to seven other Bluetooth devices at most. In order to serve more

than 7 users simultaneously, as a full double-deck bus can carry around 100 passengers, we need to work out an effective scheduler to schedule all the private services on the BlueBox side. This paper also proposes a scheduling algorithm that exploits the park mode feature of the Bluetooth specification to allow more users to have access to the BlueBus services.

2. System Design

2.1. Overview

We aim to design an inexpensive and flexible system to serve the passenger at a personalized level in the bus, so we would like to develop the client system on the Bluetooth enabled mobile phone. The project is named as "BlueBus" which indicate the bus is Bluetooth signal covered, passengers can use their Bluetooth enabled mobile phone as the interface to get the kinds of information. Beside the client side, BlueBus consists the other two parts that are base station on the bus and the administration server at the bus terminal. The overall system architecture is shown in **Figure 1**. The Mac mini will be installed on the bus as the base station, which should be able to support multiple users on the BlueBus platform. The Mac mini will communicate with mobile phone through Bluetooth while it will communicate with server via Wi-Fi.

Administration server is connected to the base station, BlueBox, through the Wireless LAN connection. BlueBus end user, mobile phones are connected to BlueBox via Bluetooth connection.

Function of the Administration server is to manage all BlueBox in the public buses parked in its located bus terminal. It is able to push updates and content synchronization to all BlueBox through Wi-Fi. Administration server located in the bus terminal office must have a stable broadband Internet connection, in order to function well.

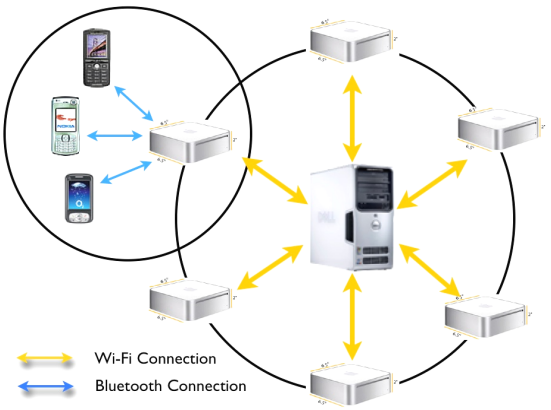

Figure 1. Overall system architecture.

BlueBox serves as the bridge in the overall system. It retrieves all contents from the Administration server and stores all contents in its local storage waiting for BlueBus commuters to browse or download. It also has to handheld all service requests from client application. The BlueBox must communicate with the Administration server periodically to ensure that it always carries the most recent content and services. In the scenario of the bus, the synchronization takes place when the bus arrives at the bus interchange terminal. The BlueBox will detect the availability of Administration server network and automatically start the service or content synchronization with Administration server.

Client application is a Java application installed in commuters' mobile phones or PDAs. Users in the bus will use this application to choose and enjoy the services in the system.

Figure 2 is the flow-chart on how to distribute the content to every single end user.

2.2. Hardware Selection

2.2.1. Administration Server

For Administration server, it must able to create a wireless network or join the wireless network in the bus terminal in order to synchronize the content and do some administrative control to all BlueBox. It also needs a broadband Internet connection to download new contents

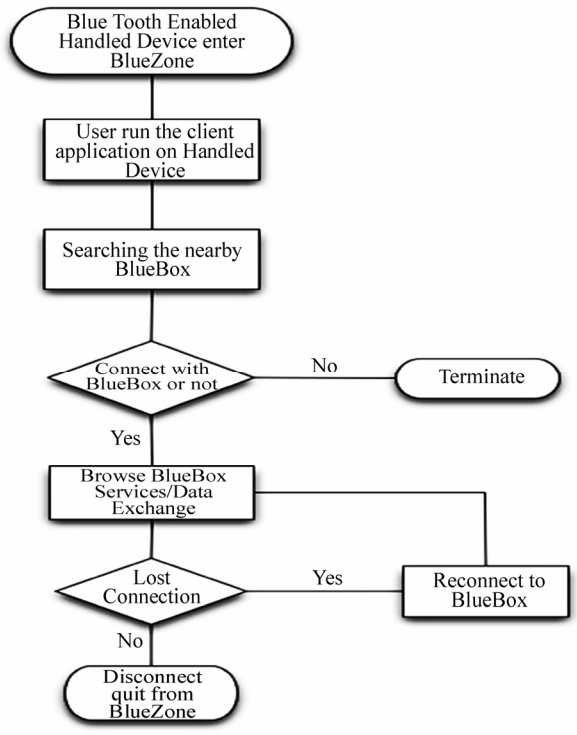

Figure 2. Flow chat of overall system.

and synchronize end user's upload content with all Administration servers in different bus terminals. It must be preinstalled Windows OS because Server side is developed using SyncML under Windows platform, Windows XP would be preferred. Any hard disk which capacity is larger than 20GB should cater for this system.

2.2.2. BlueBox

The BlueBox serves as the access point to this BlueBus system. The domain of each of these BlueBox is the range of Bluetooth i.e. a circle radius 10–100 meters, depending on the power of Bluetooth dongle. So the BlueBox should have little processing power, certain storage capacity and both Bluetooth and wireless LAN capability, certainly, small physical size will be a bonus point. Based on a preliminary analysis, we chose Mac mini as the BlueBox whose specifications are shown in **Table 2** below.

The Mac mini has a built in Mac OS X which provides a stable Bluetooth framework for system implementation. **Figure 3** shows the Bluetooth profiles available in Mac OS X [8]. We will use OBEX and RFCOMM for system development.

Table 1. Recommended specifications for administration server.

Specifications for a Administration Server	
Processor	1.5 GHz
Memory	512 MB
Hard Disk	20 GB(min)
Local Area Network	802.11b/g (11 Mbps/54 Mbps)

Table 2. Recommended specifications for BlueBox.

Specifications for BlueBox	
Processor	Core Solo 1.5 GHz
Memory	512 MB
Hard Disk	60 GB
Personal Area Network	Bluetooth v2.0+EDR
Wireless LAN Network	802.11g (54 Mbps)
Battery life	1 hour (min)
Physical dimensions	6.5" × 6.5" × 2" (L*W*H)

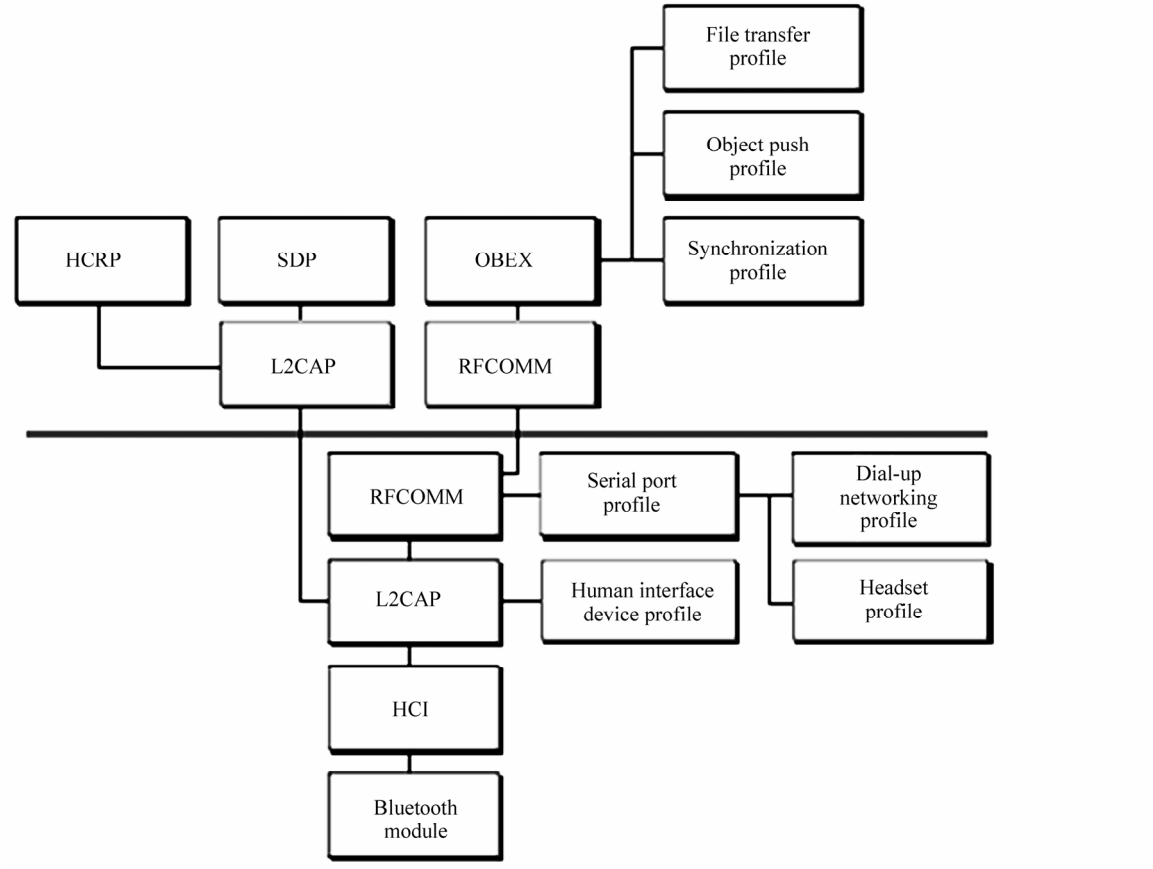

Figure 3. Mac OS X Bluetooth profiles.

2.2.3. J2ME Client Application

The J2ME client application can run on any handheld device which is Java enabled and built in JSR82 Bluetooth API.

According to our study in the market, there are more than 90 mobile phone models satisfied the hardware requirement that indicates that we really got a large potential user base.

2.3. System Requirements

As described in the previous section, BlueBus serves as the platform for mobile data services that can be delivered to users over Bluetooth. The requirements of the system are as follows:

The system must be able to detect new users who are running the client application with Bluetooth enabled handhelds.

- The BlueBus client application must provide a friendly, interactive interface for users to connect to the system and use the services provided.
- The Administration server must provide an effective interface for content providers to add new content to existing services.
- The system must be scalable and must provide an easy method for the system administrator to add new BlueBox to the system.
- The BlueBox must be able to simultaneously support several users accessing the services provided.
- The BlueBox must be easily deployable in static as well as mobile application scenarios, with occasional as well as continuous connectivity.
- The Administration server must ensure automatic synchronization of content and services on all connected BlueBox. If a BlueBox is temporarily disconnected, it must be updated as soon as connectivity restored.

2.4. BlueBox Design

BlueBox is developed in Apple Mac OS X platform. The programming language we used is Cocoa which is a famous programming language in Mac OS. Because we expect an automatic working style of BlueBox (no user interference involved) it is implemented using IOBluetooth framework in Cocoa.

Table 3. Recommended specifications for handheld device.

Specifications for Handheld Device	
Display Resolution	176*208 pixels
Free Device Memory	2 MB (min)
Phone External Storage	16 MB
Personal Area Network	Bluetooth v1.1(min)
JAVA	JAVA MIDP 1.0 or 2.0

When the server started, it will register a new Bluetooth service provided to Bluetooth stack and start accepting a new connection. The system will automatically create the input stream and output stream when there is a new client wants to connect to the service.

Communication is mostly built in request-reply model where client send the request to server first then server response to the client accordingly.

When server receives a request message, the function associated with that service is called. Each service would have separate function to handle data from client. After some processing, BlueBox will send the reply to client.

3. Bluebus Scheduling Design

3.1. Bluetooth Network Topology

Bluetooth enabled devices are organized in groups called Piconet [9]. The Piconet consists of a master and up to seven active slaves. A master and a single slave use point-to-point communication; if there are multiple slaves, point-to-multipoint communication is used. The master unit is the device that initiates the communication. A device in one Piconet can communicate to another device in another Piconet, forming a scatternet, as depicted in **Figure 4**. Notice that a master in one Piconet maybe a slave in another Piconet:

3.2. Constraints on Bluetooth

From the Bluetooth specifications we know that one Bluetooth dongle can connect to seven other Bluetooth devices at most. However, we expect more users can use the services provided in BlueBus because a full double-deck bus can carry around 100 passengers. We consider add more Bluetooth dongles on the Mac mini, but after

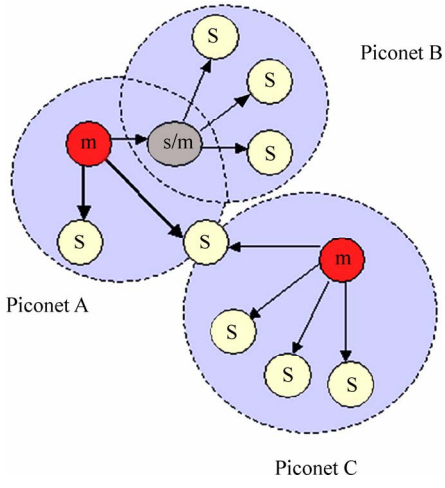

Figure 4. Scatternet comprising three piconets.

trying, we found that the seven slaves limitation cannot be solved by simply adding USB powered Bluetooth dongles. This is because the Bluetooth stack currently does not support multiple Bluetooth dongle connections. Although multiple Bluetooth dongle connection is possible in theory, so far, there is no Bluetooth stack in any other platform that support this feature.

3.3. Scheduling Algorithm Design

We want to create the Piconet which can serve all the Bluetooth end users. However, the Bluetooth master only can support seven active channels simultaneously which means only seven passengers can enjoy the services we provided in the BlueBus system. The addition of the Bluetooth dongle will not help in solving this issue as previous section discussed.

In order to serve more users simultaneously, we need to work out an effective scheduler to schedule all the private services on the BlueBox side.

3.3.1. Power Mode of Bluetooth Device

Every Bluetooth device has four different power modes [10] shown below:

- ***Active Mode***: The slave actively participates in the Piconet by listening, transmitting and receiving packets. The master periodically transmits to the slaves to maintain synchronization.
- ***Sniff Mode***: The slave listens on specified slots for its messages. It can operate in a reduced-power status the rest of the time. The master designates a reduced number of time slots for transmitting to a specific slave.
- ***Hold Mode***: The device can participate only in SCO packet exchanging and runs in reduced-power status. While it is not active, the device can participate in another Piconet.
- ***Park Mode***: It is a low power mode with very little activity. Used when a slave does not need to participate in a Piconet, but still is retain as part of it. The device is changing Active Mode Address (AM_ADDR) to Park Mode Address (PM_ADDR). With this mode, a Piconet may have more than seven slaves.

As we see from the properties of the four different power modes, we need to switch the slaves between Active mode and Park mode, in order to achieve our goal supporting more than seven devices simultaneously.

3.3.2. How to Switch between Active Mode and Park Mode

The Bluetooth enabled device has an Active Mode Address (AM_ADDR) when it is in active status while it has a Park Mode Address (PM_ADDR) when it is in park/inactive status. The length of AM_ADDR is 3 bits while the length of PM_ADDR is 8 bits. So theoretically one Bluetooth master can support 2^3-1 = 7 active Bluetooth devices and 2^8=256 parked devices.

Before a Bluetooth device entering Park mode, the master will give the device a PM_ADDR. Then this Bluetooth slave will give up the active member address, AM_ADDR if it already got one and listen to the broadcast traffic in regular intervals. The park mode timing parameters should be negotiated with the master using Link Manager Protocol (LMP). Master also can send message to parked device to wake up the parked the device and reassign the AM_ADDR to it [11].

(*In present devices on the market, the master will do the mode switching automatically instead of user controlling. So we don't know how to control the mode switching in the API level or Stack Level. If we can do that, the following will be the scheduling algorithm.*)

3.3.3. Scheduling Algorithm – First Version

Define Service Priority

The Schedule Algorithm design is divided into two parts, one is the active Bluetooth devices scheduling (the number of Bluetooth devices is less than or equal to 7) and the other part is overall scheduling (the number of Bluetooth devices is greater than 7).

Active Devices only Scheduling Algorithm

We had done a Bluetooth data transmitting speed test. First, connect individual Bluetooth enabled cell phone to Mac mini which served as the Bluetooth master in the Piconet and trying to download a 3.3MB size file from Mac mini. Second, connect 6 cell phones to Mac mini and try to download the same 3.3MB size file from Mac mini simultaneously. We took the records of the data transmitting speed and consolidate into the table shown below.

From the table, we can easily see that the data transmitting speed drop a lot when there are multiple connections existing. Generally speaking, the data transmission rate in single connection is two to three times of the rate in multiple connections in the cell phone.

Table 4. Bluetooth data transmission rate of different mobile phones.

Cell Phone Model	Single connection (4-5m)	6 connections (4-5m)
Nokia N-Gage QD	5 Kbps	5 Kbps
Nokia 6280	104 Kbps	20-30 Kbps
Nokia 3230	41 Kbps	15-18 Kbps
O2 Xphone-IIm	16 Kbps	10 Kbps
SonyErricson K750i	40 Kbps	15 Kbps
SonyErricson P910i	20 Kbps	10 Kbps
Motorola L7	3-10 Kbps	N/A (always disconnect)

In the BlueBus, we provide four basic services which are listed based on the priority, **1**) indicates the highest priority.

1) ***Bus Stop Alert***
2) ***Blue Bubble Chatting***
3) ***RSS News***
4) ***Entertainment Download***

Only the Entertainment download will involve a large size file which requires fast data transmitting speed. In this case, we will schedule the download process into serial order instead of parallel order because it can save the total data transmitting time.

Active Queue
A0 Chatting Service
A1 Chatting Service
A2 RSS News Service
A3 Download Service
A4 Bus Stop Alert Service
A5 Download Service 2
A6 Bus Stop Alert Service

After the Scheduling
A0 Bus Stop Alert Service → A4
A1 Bus Stop Alert Service → A6
A2 Chatting Service → A0
A3 Chatting Service → A1
A4 RSS News Service → A2
A5 Download Service → A3
A6 Download Service2 → A5

The actual service order is:

A0 Bus Stop Alert Service
A1 Bus Stop Alert Service
A2 Chatting Service
A3 Chatting Service
A4 RSS News Service
A5 Download Service
} Been processed simultaneously

⇩

A6 Download Service 2

In the Active queue, if there are no download services or only one download service be found, all the rest services will being processed simultaneously due to the low requirement of the data rate. However, if there are more than two download services in the Active queue, considering them as d1 and d2, we will get the files' size of each download service and current data transmitting rate of each channel. We can calculate the consuming time left to download the specific file. The less the downloadding time the higher the processing priority. We assume d1 got the higher priority, so d2 will start once the d1 finished the downloading. During d1 is in downloading status, d2 is in waiting status, it is possible that d2 will be swap into Park queue if there is another device parked with higher priority. We will talk about it in later section.

Active and Parked Devices Scheduling

There is a case which more than seven Bluetooth enabled devices are willing to connect to the Bluetooth master. We will put the first seven devices into the Active queue, the eighth device and followings will be put into Park queue, PM_ADDR of them will be set.

```
sort Active queue 0-6 based on the priority level
   Priority high → low
sort Park queue 0-n (n<256) based on the priority level
   Priority high → low
compare (AQ [6], PQ [0])
   if priority of AQ [6] >= PQ [0]
      Return;
   else if priority of AQ [6] < PQ [0]
      //swap (AQ [6], PQ [0]);
      set AQ [6] PM_ADDR;
      set PQ [0] AM_ADDR;
      add AQ [6] to Park queue, resort the Park queue;
      add PQ [0] to Active queue, resort the Active queue;
      compare( AQ[6], PQ [0]);
```

3.3.4. Improved Version of Scheduling Algorithm

3.3.4.1. Reserve Channel for a Certain Service

Bus Stop Alert service is not a time consuming service, we just need to record the user's registration and save the destination in the BlueBox database. After that, we are able to put it into the Park queue. We only need to wake up this user one stop before the customized destination by sending an alert message.

There is an alternative way to implement the Chat service. The design is shown below.

Figure 5 shows the data structure of a chat service user. It got two buffers which is used to send message while the other buffer is used to receive message from chat server.

We reserve one active channel for the Chat service, each time only one sender can send the message to chat server (BlueBox), after it deposit message to the server, server begin to broadcast this new message to every connected user who are online by swapping them one by

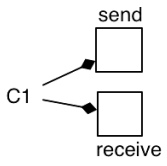

Figure 5. Send and receive buffer of C1.

one from Park queue to Active Queue. After that, the second sender will deposit its composed message to chat server, then server will do the same thing to broadcast this new message to all online users. Certainly, the server is also able to send the message to the specific recipient. We will limit the message length to 160 English characters which is as the same as the normal SMS length. Assume the average Bluetooth data transmitting rate is 5Kbps, then the average time to transmit one message will cost: **160*bytes* / 5*Kbps* = 0.03*s***

If there are 20 users using the Chat service, the server can finish the "swap broadcasting" in around 0.6s, within one second which should be acceptable for the users.

We take an example of three users are using the public chat service. We use C1, C2 and C3 to indicate the three users.

From **Figure 6**, we can see that C1 and C3 want to send a message to others. They take the active channel and communicate with server in a serial manner. C1 deposits the message to server, then the server will broadcast the message to C2 and C3 when they are taking the channel. C3 will not send its own message to server in this server swap broadcasting round. So after 3), everyone got the message from C1. Message 1 will be displayed on every users' screens, and the message one will be cleared from receive buffer.

Then it goes to next message deposit round, it is same as round 1. Chat server will keep all the messages. Chat server will broadcast the message based on the time stamp of message itself which will ensure the accuracy of the message received by users.

As for the RSS News service, we also can use the similar mechanism, reserve one channel for RSS News service use only. We can do a calculation that, normally the size of one RSS News thread should be within 2KB which equal to 300 to 400 English words. It will cost the user less than 2 minutes to finish the reading on 176*208 sized cell phone screen, one minute is needed at least. Within 1 minute the server can send at 60 RSS News threads to users that mean it can support up to 60 users to read RSS News with one channel used (**Figure 7**).

Download service must be processed one by one unless the download file size is small.

With the above statement, we can reserve the 7 active channels for different services. Download and Bus Stop Alert services only reserve one channel for each while both Chatting and RSS News service can reserve 2 channels for each, because they are more timing issue is more critical for them.

3.3.4.2. Random channel allocation and Sub Queue Algorithm

We also can split the Park Queue to several Park queues based on how many services we provided. In current stage, we can reform 4 Park queues, which are Bus Stop

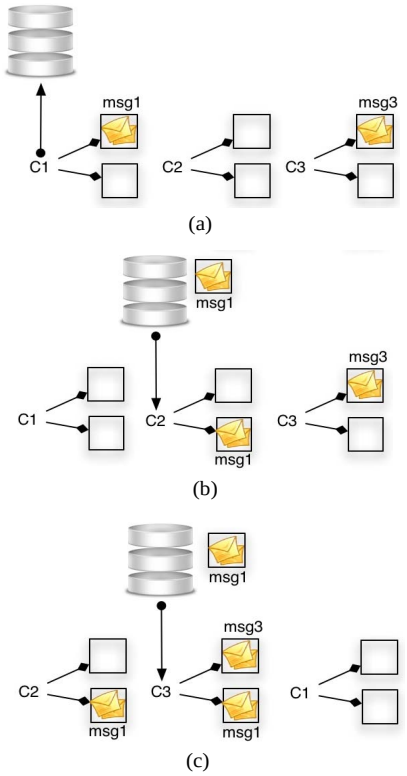

Figure 6. Swap-broadcast message from C1.

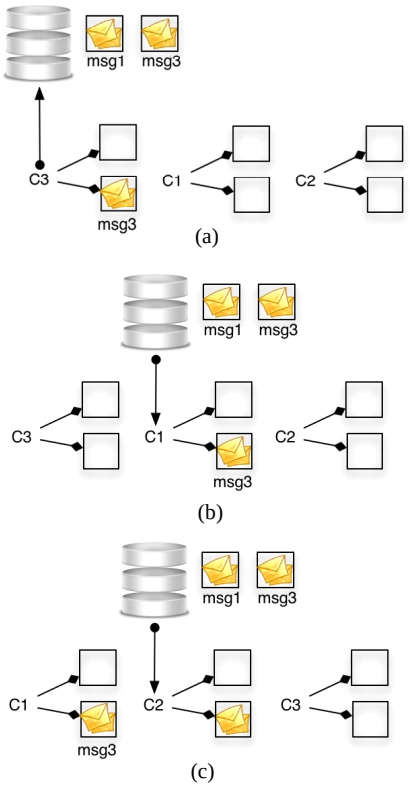

Figure 7. Swap-broadcast message from C3.

Alert Queue, Chat Queue, RSS News Queue and Download Queue, listed based on the priority from high to low. Once there is an active channel is free, we will process the service queue based on its priority level. If there are more active channels are free, more service queues will be processed by server.

Because there are 7 active channels while there are only 4 services currently, some service park queue, such as Chat service and RSS News service, can split its park queue equally into 2 sub-park queues. Each sub-park queue will take a channel for processing which may reduce the time consuming for one round "swap broadcasting".

4. System Evaluation

For the system evaluation, eight Bluetooth 2.0 Class 1 + EDR dongles are used. The experiments were conducted to measure the transmission rate and interference that may affect the transmission rate due to co-existing piconets and the effect of obstacles in-between the Bluetooth devices to the transmission rate. Therefore, three test sets were conducted as follows: 1. Single piconet Test, 2. Multiple piconets test, and 3. Obstacle test. For every test, the experiment is conducted five times and the average is taken and plotted to a graph.

4.1. Single Piconet Test

This test was conducted to measure the effect of increasing number of slaves on a Bluetooth master to its transmission rate.

Figure 8 shows the test result. For the transmission to only one slave on a master dongle, it achieved the data rate of 1.6 Mbps, which is fairly close to the theoretical transmission rate of Bluetooth, i.e., 2.0 Mbps. However, as the number of slave increases, the total data rate drops quite significantly. With only one additional slave, the total data rate drops by 400 kbps to 1.2 Mbps. According to the Bluetooth Specification [12], this drop is due to the fact that when there is only one slave, the master is able to transport L2CAP (Logical Link Control and Adaptation Protocol) broadcasts over the ACL–U (Asynchronous ConnectionLess – User) logical link rather than over the ASB–U (Active Slave Broadcast – User) or PSB–U (Parked Slave Broadcast – User) logical links. This allows the transmission to be more efficient in terms of bandwidth (if the physical channel quality is not too degraded) for the case of only one slave.

Also, as only one device is allowed to transmit/receive in each 625 μsec time slot of a Bluetooth transmission, the data rate of one of the slaves (in the two-slave case) drops to only 600 kbps (from 1.6 Mbps when there is only one slave) – a very significant reduction. However, on subsequent addition of slaves, the total data rate varies only slightly from the two-slave case, where the total data rate decreases as the number of slaves increases. Although there is a drop in the total data rate, each slave still gets an equal share of bandwidth. At the limit of 7 slaves, each slave achieves a data rate of around 100 Kbps which is considered acceptable for the BlueBus services.

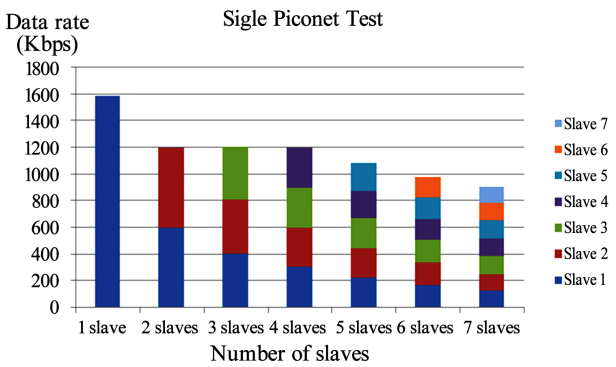

Figure 8. Effect of the number of slave dongles on the data rate in one piconet.

4.2. Multiple Piconets Test

The Bluetooth Class 1 device could reach the range of 100 m, and this is the Bluetooth device planned to be used on the BlueBus. Since the interior area of the bus is less than 100 m, having four server dongles (the dispatcher is not involved in the experiments) may introduce interference problem. Therefore, this test is intended to see the effect of the increasing number of co-existing piconets on the transmission rate.

Figure 9 shows the data rates achieved when multiple piconets co-exist. Here, 'm' denotes the master dongle and 's' denotes the slave dongle. In the base experiment of one server and one client (which is essentially the same as the case of only one slave in **Figure 8**), it can achieve a data rate of 1.6 Mbps. However, if multiple

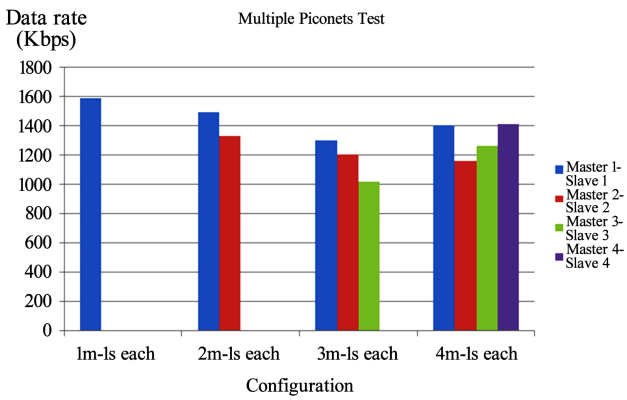

Figure 9. Effect of the number of master dongles on the data rate.

masters exist (i.e. multiple piconets), the performance drops. It can be seen from **Figure 9** that the drop is not very significant, and it is concluded that the multiple piconets interference does not affect the data rate much.

4.3. Obstacle Test

Bluetooth is a wireless technology, and as such, its transmission rate may be subjected to the objects that hinder its transmission path. Inside a bus, the obstacles can be the seats or the metal poles, including the people packed inside the bus with the various objects they may carry. Therefore, this test is used to find out the influence of various typical objects in between the Bluetooth transmission path on the Bluetooth's transmission rate. In this test, the master and the slave dongle are separated 1.5 meters apart. The separation is based on the derivation from **Figure 7**. All the obstacles are placed within close proximity to the receiving dongle (around 10 cm). If a user uses his mobile device on a bus, it will be close to the objects that he is currently holding (books, newspapers, etc) or the seat in front of him. In this test, only one master and one slave are used.

Figure 10 shows the result of putting different obstacles in the Bluetooth transmission path. The obstacles used are representative of the objects that are typically in the bus or the objects that people may carry. The obstacle of human body is obvious, since the bus will be packed by passengers inside. The 3.5 cm thick book used in this test represent any types of printed media of pages bound together such as text books, newspapers, magazines, etc. A thick book is used, since if the transmission rate is good on the thick book, the Bluetooth will perform equally well on thinner books or printed media. The bottle of water represents any liquid that a person may carry. The plastic glass represents materials made up of plastic such as the plastic bottles, and finally the glass represents the windows of the bus.

As **Figure 10** shows, different obstacles of different materials affect the data rate to different extents. However, the figure still shows acceptable performance on all the obstacles tested. Nevertheless, to minimize the impact of obstacles inside the bus, the master dongles should be mounted along the ceiling of the bus and connected to the BlueBox by USB cables. As the distance limit of a USB cable is 5 m, additional USB hubs can be used to extend this distance limit.

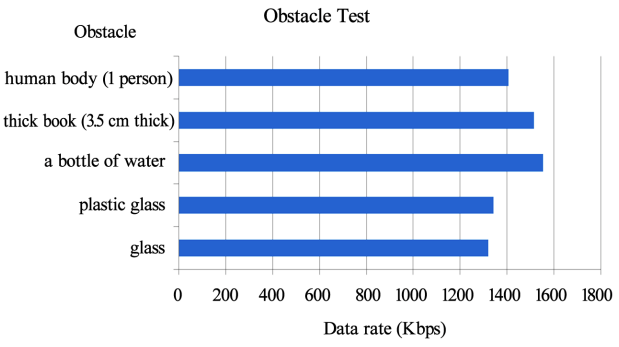

Figure 10. Influence of different obstacles on the data rate.

5. Conclusions and Future Development

We had designed and developed the BlueBus system which is able to provide variety information and entertainment services to passengers in a personalized level and give them a good user experience. We have shown that, based on the test on real Bluetooth dongles that even when a Bluetooth device is full with active connections, it can still maintain relatively good data rate for each slave on 100 Kbps. The experiments conducted also shows that the interference of co-existing Bluetooth piconets does not affect the data rate much. This is also true for the obstacle experiment where the Bluetooth data rate does not vary widely when it is presented with different materials.

We are planning to provide more interesting services such as send birthday or greeting cards, share personal Mobile Log (similar with Blog) etc. Apart from this, we also aim to find a way in the Bluetooth stack level or scheduling algorithm to improve the multiple user support feature.

6. References

[1] K. Goto and Y. Kambayashi, "Integration of electronic tickets and personal guide system for public transport using mobile terminals," Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 642–646, 2003.

[2] T. Y. Banos, E. Aquino, F. D. Sernas, Y. R. Lopez, and R. C. Mendoza, "EMI: A system to improve and promote the use of public transportation," Conference on Human Factors in Computing Systems, CHI'07 Extended Abstracts on Human Factors in Computing Systems, pp. 2037–2042, 2007.

[3] J. Kjeldskov, S. Howard, J. Murphy, J. Carrol, F. Vetere, and C. Graham, "Designing TramMateña context-aware mobile system supporting use of public transportation," Proceedings of the 2003 Conference on Designing for User Experiences, pp. 1–4, 2003.

[4] J. Kjeldskov, E. Andersen, and L. Hedegaard, "Designing and evaluating buster: An indexical mobile travel planner for public transportation," Proceedings of 19th Australian Conference on Computer-Human Interaction: Entertaining User Interfaces, pp. 25–28, 2007.

[5] K. D. Lin and J. F. Chang, "Communications and enter-

tainment onboard a high-speed public transport system," IEEE Wireless Communications, Vol. 9, pp. 84–89, February 2002.

[6] V. Scinteie, "Implementing passenger information, entertainment, and security systems in light rail transit," Transportation Research Circular E–C058: 9th National Light Rail Transit Conference, pp. 528–533, 2003.

[7] "French railways launch their on-board internet and multimedia connectivity services on TGV high-speed train," Appear, the Context Company, 2007. http://www.appear-networks.com/French-Railways-launch-their-on.html.

[8] "Bluetooth device access guide in Mac OS X." http://developer.apple.com/documentation/DeviceDrivers/Conceptual/Bluetooth/index.html.

[9] "Bluetooth network topology." http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth1.

[10] "Bluetooth power mode overview." http://72.14.235.104/search?q=cache:4LenKxr14d0J:www.ece.ualberta.ca/~sbates/downloads/Mint702_PartSix.ppt+how+to+assign+the+PM_ADDR&hl=en&gl=sg&ct=clnk&cd=3.

[11] "Simple differences between Sniff mode and Park mode." http://www.palowireless.com/infotooth/knowbase/baseband/78.asp.

[12] "Bluetooth core specification V2.1 + EDR," Bluetooth SIG, 2007.

