Bluetooth Park State Scheduling Algorithm for BlueBus, an On-Board Infotainment System for Public Buses in Singapore

KIN CHOONG YOW[1], JACKY TJOA[2], KEOK KEE LEE[3]
School of Computer Engineering, Nanyang Technological University,
Nanyang Avenue, Singapore 639798
SINGAPORE
[1]kcyow@ntu.edu.sg, [2]jack0005@ntu.edu.sg, [3]askklee@ntu.edu.sg

Abstract: - Singapore, a small island city state, has developed an extensive public transportation system. However, the information and entertainment (infotainment) system within has not received much attention. Several approaches for implementing infotainment system on-board public transportation exist, but they require considerable investment in the hardware. In this paper, we propose a low cost on-board entertainment system using Bluetooth and passengers' own Java ME-enabled mobile phone for delivering the infotainment content, called "BlueBus". Since BlueBus uses Bluetooth as the connection medium, it suffers from the Bluetooth limitation of only capable of supporting 7 active connections. Therefore, we developed a scheduling algorithm utilizing Bluetooth Park State capability and priority of services offered by BlueBus. We constructed a computer simulation to measure the performance of the algorithm. The result shows that even with large number of users, our algorithm performs well and incurs a first service delay that is three times lower than the Basic Bluetooth Connection configuration.

Key-Words: - BlueBus, Bluetooth, Scheduling algorithm, Park State, On-board infotainment system

1 Introduction

Singapore, a small city-state in Southeast Asia, has developed an extensive transportation system for the convenience of its citizens, with an average of over 4 million rides per day on the public transportation. Despite the convenience offered by the transportation system, the lack of an On-Board Infotainment System (OBIS) cause most commuters to stay inside idly during the travel until they reach their destination.

In order to address this issue, we have developed an OBIS which allows interactive and personalized experiences based on Bluetooth, called "BlueBus". Using BlueBus, the passengers of a bus are able to browse multimedia contents with their own mobile phones and remain entertained during their journey.

Our solution uses Bluetooth due to its ubiquity, low power consumption (which is an issue in mobile phones), and less-prone to interference. It does not require line-of-sight and moreover, it is free for its usage. However, Bluetooth has the limitation of only able to support up to 7 active connections on one device. Therefore, in order to support much more users in a bus, we develop a scheduling algorithm utilizing the Park State feature of Bluetooth, called "Park State Scheduling Algorithm – PSSA".

The organization of the rest of this paper is as follows: Section 2 provides overview of the related works in the area, Section 3 describes the BlueBus system in detail, Section 4 describes the scheduling algorithm developed in this project, Section 5 presents the experiments and simulations conducted and finally, Section 6 concludes this paper.

2 Related Systems

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 include 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

Other systems address this issue in providing infotainment experience on-board the public transportation, such as the system proposed by Lin and Chang [5], ALSTOM [6], and the system deployed by French TGV trains [7]. These systems include a LAN on-board the public transportation, which 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.

The straightforward approach for enabling personal and interactive experience is to install in-seat display screens. Through the screen, passengers are able to choose the contents stored in the on-board server(s), as deployed by Ho-Hsin Bus Traffic Company [8] in Taiwan and MyExpressBus [9] in Malaysia-Singapore route, for coaches (luxury buses) and iCabTV [10] for cabs/taxis in Singapore. Such systems, although enabling personalized experiences, require a display screen to be installed in every seat of the bus, which is prohibitively expensive to be deployed.

Other approaches consider the use of passengers' own mobile phones (or any handhelds) for delivering multimedia contents on-board the public transportation, such as Nokia Wi-Fi Zone [11] through Wi-Fi and Bluespot [12] through Bluetooth, both on public buses. The Nokia Wi-Fi Zone enables users to browse the Internet live, free of charge, while they are riding on the bus. However, the live connection is achieved through the 3G/3.5G network, in which the cost have to be borne by the bus company.

Our solution has advantages over these systems since BlueBus does not require direct access to the Internet, but is able to keep the content regularly up-to-date. Moreover, BlueBus is able to handle much more users than the limitation of Bluetooth by the use of the custom-developed scheduling algorithm, the PSSA. The mechanism to achieve these features will be described in the subsequent sections

3 BlueBus Solution Design

This section describes our proposed solution, the BlueBus. BlueBus aims to be the pioneer of the mass OBIS for the public transportation in Singapore. BlueBus comprises of two sub-systems: the front-end system on the buses and the back-end system on the bus terminals.

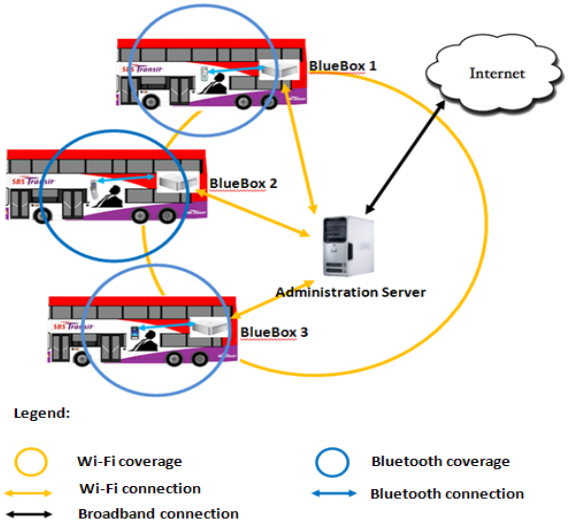

Fig. 1 Overview of the BlueBus operation

Fig. 1 provides an overview of the BlueBus operation. In a bus, a local server, called the "BlueBox", is mounted to serve the passengers while the bus is on its journey. This is the front-end system. The passengers, using their own handhelds, are able to browse the infotainment content stored in the BlueBox via Bluetooth connection. The BlueBox has no direct connection to the Internet; it serves as an offline data server for the users. The back-end system is in the bus terminal, consists of an Administration Server (AS) which is used as the central controller for managing the BlueBus operation. The AS is connected to the Internet via broadband connection. It serves to provide the updated data and synchronization for the BlueBoxes that enter the terminal via Wi-Fi connection. Several Wi-Fi access points are distributed across the terminal to accommodate the AS-BlueBoxes traffic. When a bus enters a terminal, the BlueBox in the bus will establish a Wi-Fi connection to the AS. The local content on the BlueBox is updated with the fresh content from the Internet via the AS. This refreshed data is the data used by the BlueBox in the subsequent bus travel.

The BlueBus system is designed in a modular and layered manner in which all of the operations are done through a request-reply mechanism. The layered approach enables subsequent updates to the components only when necessary without affecting the whole operation of the BlueBus system. Moreover, it also provides the flexibility to expand the system to account for more features or services in the future. Fig. 2 shows the BlueBus architecture.

The Administration Server (AS) manages the administrative tasks of the BlueBus system. A system administrator, via the web-based module of the AS, is able to view the list of all currently connected BlueBoxes and performs administration/maintanence of the BlueBoxes, such as retrieving error logs or event history of a particular BlueBox. The synchronization of the contents of the BlueBox and the AS can be done automatically, using the SyncML (Synchronization Markup Language) synchronization technology. SyncML is a platform-independent information synchronization standard that has been widely used in mobile synchronization solutions. Once the AS and the BlueBox is connected, the AS will synchronize its contents (data and file system) to the BlueBox, via the Wi-Fi connection. When a user requests for a service, for example, the chat application, the Java ME client software will notice that a service is being requested (Service Handler) and a transaction will be generated (Transaction Handler). The transaction is divided into messages (Message Handler) and subsequently to bit streams (RFcomm driver) in order to send the request to the BlueBox via the Bluetooth connection. Upon arrival of the request at the BlueBox, the message will go

through the reverse process. After the message reaches the Service Handler, it will be processed and passed to the intended service. In a similar way, the reply from the service is arranged by the Service Handler as transactions, to messages, and finally to stream data and sent back to the client. From the BlueBox, the reply message can be transmitted either in a one-to-one or a one-to-many scheme. We have implemented 5 BlueBus services: (1) bus stop alert, (2) the RSS News, (3) download service and (4) video streaming service which are examples of the one-to-one schemes, and (5) chat service, which is an example of the one-to-many scheme, since, when a user joins the chat service; all other currently online users should be notified that a new user is online.

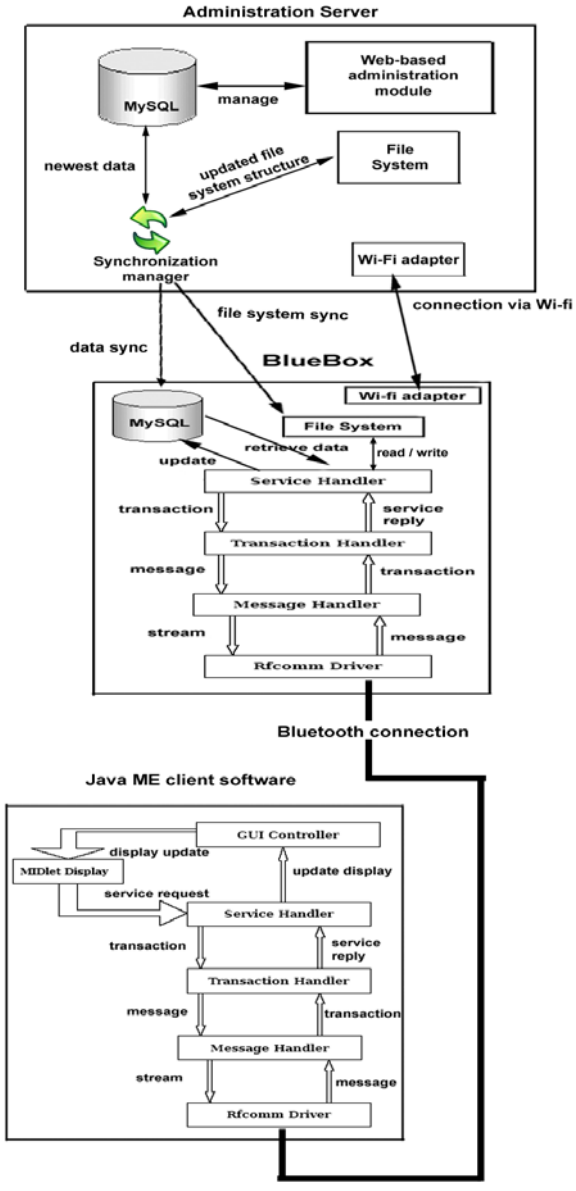

Fig. 2 BlueBus Architecture

4 BlueBox's Park State Scheduling Algorithm

This section describes the scheduling algorithm we developed for the BlueBus system to overcome the limited number of active connections that can be supported by one Bluetooth device. The scheduling algorithm make use of the Park State capability of the Bluetooth combined with the nature or characteristics of each service of BlueBus in order to allow the BlueBox in the bus to use limited number of Bluetooth servers to handle many users simultaneously.

When a connection is coming from a mobile client, it will first go through the dispatcher for registration with the BlueBox. The BlueBox keeps track of the load of each of the server dongles, and hence the dispatcher dongle will dispatch the new connection to the dongle that has the lowest load.. This is to ensure fair usage of the dongles and to avoid the overloading at a particular dongle or the under-utilization of a dongle.

Once a mobile node is connected with a server dongle, it will remain connected to that dongle. It will not be assigned to other server dongles during its lifetime. This is to avoid the need to connect and re-connect the mobile client, which will add complexity in the BlueBox's scheduling.

Since one Bluetooth dongle can only handle 7 active connections at a time, a further constraint is added to the Bluetooth server dongles such that only five out of seven available active connections can be used for download or video streaming traffic. This is to prevent the situation where all of the active links are occupied with bulky transfer in which this condition will lock up other request for the BlueBox's service. If a server dongle is currently occupied with bulky transfer on five of its active links, a further request for download or video streaming service will simply be rejected.

4.1 Global Park Queues

If a mobile client is currently in the Park State, the client will not be able to communicate with the master directly since there is no ACL (Asynchronous Connection-Less) link between them. Therefore, if a mobile client has a request, it must generate an un-park request first. The server will then un-park the client and the request of the client can be sent to the server. Since the client comes from the Park State, the request will join the 'global park queues', where there is one queue for each service. The mobile client is subsequently parked again. This is done to free up the links occupied which can be used if other clients request to be un-parked. The queues are processed based on the priority of each service as described in the previous section. On the un-parking operation, if there are slots available for active connections, the slave can be un-parked directly;

otherwise, the server dongle will seek for which active connection is idle and park the idle slave. Once the slave with the request is un-parked, its request is sent out to the server and it is put in the appropriate queue.

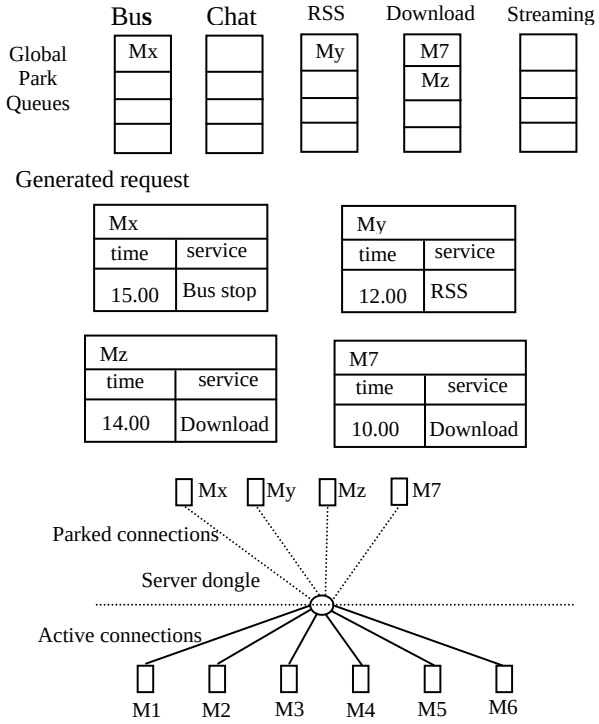

Fig. 3 Global Park Queues

Fig. 3 illustrates the scenario of the global park queues. At first, M7 comes in with a download request and the request is put to the global park queue for download service. Subsequently, My follows with an RSS News request. A while later Mz also puts in a download request, hence, its request is queued behind that of M7. Finally, a bus stop request comes from Mx. Since the Mx's request of bus stop has the highest priority of all, this request will be processed first despite Mx's latest arrival. This is followed by the RSS request and the two download request.

4.2 Local Reply Queues

Based on the characteristics of the services of BlueBus, the download of bulky transfer (either pure download or video streaming) is allowed to be interrupted. In order to enable such a mechanism, four local queues are assigned to the mobile client once it is connected to a server dongle. Hence, each mobile client has its own set of queues maintained by its server dongle. These queues are different from the global queues described before. The global park queues are used to hold the *request* from any mobile clients that come in from the Park State, hence, it is named 'global'. The local queues described here are used to hold any *reply* from the server to the mobile client which cannot be sent out immediately due to either the mobile client is in Park State or it is engaged in other activities. Therefore, these queues are referred to by the term 'local reply queues'. These queues are maintained for each of the mobile client that is connected to the server.

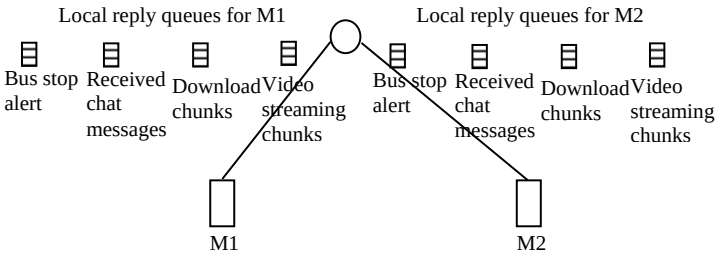

Fig. 4 Local Reply Queues

The four queues are filled up based on the service type, one for bus stop alert reply, one for chat message that comes in, one for download chunks and one for video streaming chunks. Fig. 4 illustrates these queues. The order of processing of the pending reply queues is as follows (from the highest to the lowest priority): Bus stop alert, received chat message, and download or streaming. The bus stop alert reply queue is assigned the highest priority, since the alert is urgent and should be sent out in a minimal time as possible. For the received chat message pending queue, it is possible that more than one chat messages are received and stacked on the queue. Hence, in order to minimize the waiting time for each chat message received, ten messages are processed at a time before the server proceeds to check the next priority queues (either download or video streaming pending queue). The download and video streaming request do not have priority over each other, since only one of them is allowed at a time by the server. Once the request for download or video streaming is received by the server dongle, the file requested is broken down to chunks of 100 Kilobytes each and these chunks are filled in the download or video streaming local reply queues respectively. The RSS News service has no pending queue, since the mobile client is locked in a session when the request is made, and the reply will be available immediately. Therefore, a reply queue for RSS News service is not needed. These local reply queues are checked during every transfer for any data that are pending for the mobile client, for all the connected mobile clients.

5 Experiments and Simulations

BlueBus has yet to be physically deployed on the public buses. However, we have built a prototype system for BlueBus. Here, we presents the experiments and simulation conducted to test the scheduling algorithm of BlueBus, the PSSA, and measure its performance.

5.1 BlueBox Simulation

We developed a computer simulation to measure the performance of the scheduling algorithm. We use ns-2 network simulator with UCBT (University of Cincinnati - Bluetooth) Bluetooth extension for simulating Bluetooth. The simulation of BlueBox operations in a bus is simulated in an area of 12 meters x 2.5 meters, which is the typical dimension of a bus. Five static nodes are used as the BlueBox's dongles. They are the dispatcher and the four server dongles. The static nodes are arranged based on the layout of Fig. 5. Aside from these nodes, there are configurable number of mobile nodes which act as the mobile clients (the users). The mobile nodes are randomly moved within the simulation area following the Random Waypoint Model (RWM).

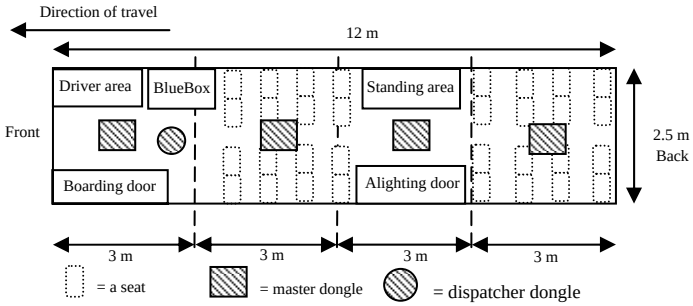

Fig. 5 Typical bus layout in Singapore with BlueBox system overlaid

An incoming connection from a mobile node will go through the dispatcher first and is subsequently assigned to a server dongle. Once connected, the mobile node generates its requests based on the Poisson arrival process. Ten requests are generated per mobile node. The bus stop alert request is always the first request and it is only generated once, since it is not possible for a user to drop at multiple destinations from the same bus. Subsequent requests can be of any type, randomly ordered. A constraint is put on the download and video streaming service, in which a mobile node is allowed to only generate either one of this bulky traffic. For other traffic, such as the chat and RSS News service, it can be generated multiple times.

All the services, except the video streaming service, use FTP traffic over TCP link, in which an acknowledgment is sent if the receiver successfully receives the transferred packet. For the video streaming service, CBR traffic is used over a UDP link. The parameters used for the simulation are shown in table 1:

Table 1. Simulation Parameters

General:	
Simulation area	12 meters x 2.5 meters
Simulation time	3,500 seconds
Number of server dongles	5 (1 for the dispatcher, 4 for the server dongles)
Number of mobile nodes	100
Arrival of the mobile nodes	Poisson arrival process with average inter-arrival time of 15 seconds
Arrival of traffic request per mobile node	Poisson arrival process with average inter-arrival time of 20 seconds
Mobile node movement model	Random Waypoint Model
Number of requests per mobile node	10

We compare the result of the simulation of PSSA with another system which only handles active connections. We call it 'Basic Bluetooth Connection (BBC)'. The BBC system is used as the base comparison with the PSSA in order to assess how well the PSSA performs.

5.2 First Service Delay

Since the connection time differs by a large factor, it will affect how long the first service will get executed since the arrival of the mobile node. The first service can be any of the 5 services. The same service is grouped together and the average is taken.

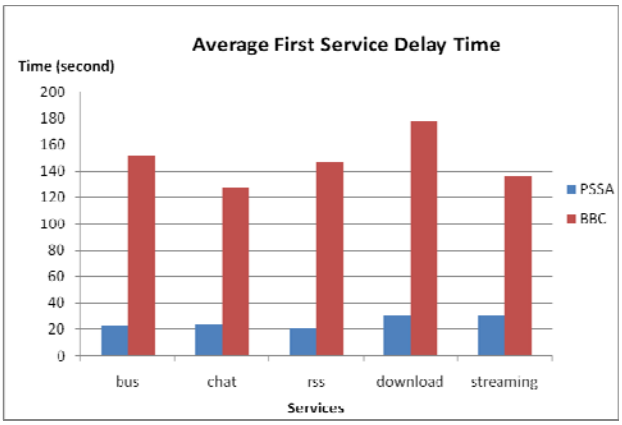

Fig. 6 First Service Delay

Fig. 6 shows the result of the first service delay. Without the use of parking on the BBC, the first service is served quite long after the arrival of the mobile clients, due to the waiting time the mobile clients experienced in connecting to the server. For PSSA, the download and video streaming request experience delay of 30 seconds on average, which is longer than other services, which experience 20 seconds delay on average. This is caused by the constraint that is imposed on PSSA that only 5 links out of 7 available active links on a dongle are used for bulky traffic. Hence, only 20 links for all 4 servers of PSSA can be used for bulky traffic. The next bulky traffic request will be put on the global park queues until one of the links finishes the bulky traffic transfers.

5.3 Request-Response Delay

Fig. 7 shows the result for the request response time, which is the measure of the responsiveness of the server by service type. For PSSA, the graph reflects the priority mechanism of the scheduling algorithm.

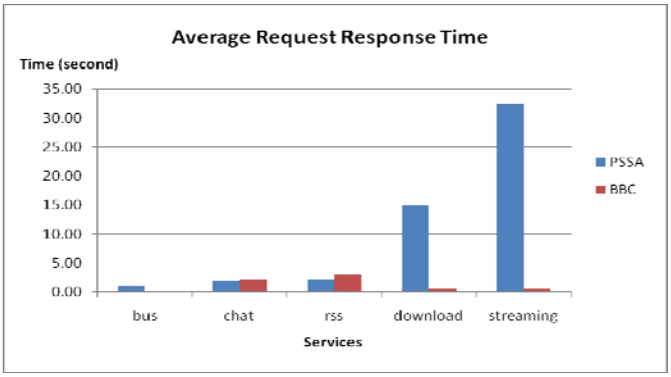

Fig.7 Average request response time

It shows that the service delay time follows the order of the priority assigned to the global park queues of PSSA. The bus stop alert is the highest priority service, hence, the delay time for the response is the least of all services. Contrary, the video streaming service is the lowest priority, and therefore, it will wait longer in the global park queues before it got executed. On the other hand, with BBC, the services can be executed immediately once it is requested, since, once a mobile node is connected to a server dongle, the established link is dedicated entirely to the mobile node until it finishes all of its requests and get disconnected from the system. Therefore the services experience considerably shorter delay in response to the mobile node's request.

6 Conclusion

This paper discusses the design and implementation of a Bluetooth Park State Scheduling Algorithm (PSSA) for a system called BlueBus, which is an On-board Infrotainment System, for public buses in Singapore. The Park State Scheduling Algorithm (PSSA) enables fast connection registration to the BlueBox, and allows the BlueBox to overcome the limited number of active connections that can be supported by one Bluetooth device. Compared to the Basic Bluetooth Connection (BBC), the PSSA registers a first service time that is is roughly 7 times better compared to BBC. For the service responsiveness, although PSSA does not perform as well as BBC, the PSSA is able to allow higher priority services to be served first before the lower priority services. For the BBC, even though the responsiveness is very good, links may be fully occupied by traffic and cause the new incoming connections to wait for a long time before getting connected.

References:

[1] K. Goto and Y. Kambayashi, "Integration of Electronic Tickets and Personal Guide System for Public Transport using Mobile Terminals," in *Proc. of the 2003 ACM SIGMOD international conference of management of data*, 2003, pp. 642-646.

[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," in *Conference on Human Factors in Computing Systems, CHI'07 extended abstracts on Human factors in computing systems*, 2007, pp. 2037-2042.

[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," in *Proc. of the 2003 Conf. on Designing for user experiences*, 2003, pp. 1-4.

[4] J. Kjeldskov, E. Andersen, and L. Hedegaard, "Designing and Evaluating Buster: an Indexical Mobile Travel Planner for Public Transportation," in *Proc. of 19th Australian conference on Computer-Human Interaction: Entertaining User Interfaces*, 2007, pp. 25-28.

[5] K.D. Lin and J.F. Chang, "Communications and Entertainment Onboard a High-Speed Public Transport System," *IEEE Wireless Communications*, vol. 9, pp. 84–89, Feb. 2002.

[6] V. Scinteie, "Implementing Passenger Information, Entertainment, and Security Systems in Light Rail Transit," in *Transportation Research Circular E-C058: 9th National Light Rail Transit Conference*, 2003, pp. 528-533

[7] Appear, the context company. (2007) "French Railways launch their on-board internet and multimedia connectivity services on TGV high-speed train". [Online]. Available: http://www.appearnetworks.com/French-Railways-launch-their-on.html

[8] Ho-Hsin Bus Traffic Company Co., Ltd. website. (2008) [Online]. Available: http://www.ebus.com.tw/en/about.htm

[9] MyExpressBus website. (2009) [Online]. Available: http://www.myexpressbus.com/

[10] UZoneMedia Pte. Ltd. website. (2007) [Online]. Available: http://www.uzonemedia.com/

[11] Channel News Asia. (2007) The Mobile Traveller. [Online]. Available: http://www.channelnewsasia.com/wifi/feature_131107.htm

[12] J. LeBrun and C.N. Chuah, "Bluetooth content distribution stations on public transit," in *Proc. of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, 2006, pp. 63-65