Unambiguous Device Identification and Fast Connection Setup in Bluetooth

Axel Busboom†, Ian Herwono*, Marko Schuba†, Guido Zavagli†

† Ericsson Eurolab Deutschland GmbH, Ericsson Allee 1, 52134 Herzogenrath, Germany
Tel: +49 2407 575 0, Fax: +49 2407 575 400, E-mail: *firstname.lastname*@eed.ericsson.se

* Aachen University of Technology, Communication Networks, 52056 Aachen, Germany
Tel: +49 241 80 27248, Fax: +49 241 80 22242, E-mail: ian@comnets.rwth-aachen.de

ABSTRACT

The paper addresses two problems of Bluetooth which arise in situations where a user device performs transactions with a specific server device out of a number of similar devices in the proximity, such as ticket machines or point-of-sales terminals. Firstly, the server that a connection should be set up with, needs to be uniquely identified and selected. Secondly, with the current Bluetooth protocol specifications the connection setup may take a prohibitively long time. The paper revisits the Bluetooth inquiry and page procedures and their duration. We discuss two ways of uniquely identifying the target device relying on manual user interaction, and compare the times needed for connection establishment. We further propose to use a second short-range communication medium, such as an integrated infrared interface or an RFID transponder, to facilitate the connection setup. This eliminates the need for manual selection of the target device and further speeds up the connection establishment.

1. INTRODUCTION

Bluetooth is a short-range radio interface in the 2.45 GHz frequency band that has been designed as a cable replacement technology as well as for ad-hoc networking between a multitude of stationary and portable electronic devices [1]. Conceivable applications include wireless connections of mobile phones to headsets or laptop computers, short-range wireless local area networks, information and transaction services from local access points and many others.

This paper addresses application scenarios where a number of local servers, such as ticket or vending machines or point-of-sales terminals, offer information and transaction services. For example, vending machines could accept payment for a drink via Bluetooth from the customer's mobile phone or Personal Digital Assistant (PDA). Likewise, a ticket machine could offer information about departure schedules via Bluetooth and sell electronic transportation tickets. Or a customer uses their mobile device to authorize payment at a Bluetooth enabled cash register when checking out at a store. Obviously, it is important in these scenarios that the user can be sure they are communicating with the right machine if there are several similar or identical ones in the vicinity. Otherwise, they might be asked to authorize payment for a purchase that someone else has just made at another vending machine or cash register nearby. Furthermore, such a mobile transaction system is only useful if it is fast, i.e. not significantly slower than conventional systems using cash, checks or credit cards.

From a technical point of view, this poses two challenges: Firstly, since the range of the Bluetooth radio interface is around 10 m at 0 dBm (up to 100 m at +20 dBm), it is not unlikely that the user's device may simultaneously be within radio range of several local servers, e.g. ticket machines. The desired server needs to be uniquely identified before any transaction can take place. Secondly, connection establishment between two Bluetooth devices that have no prior knowledge about each other, takes 5.76 s in a typical case and as much as 23 s in the worst case [1]. To this, the time needed to actually perform the interactions and transactions needs to be added. The long connection setup time during which – from a user's point of view - nothing is happening, would severely degrade the usefulness and user acceptance of a Bluetooth based transaction system. This paper discusses a number of ways how these two issues – unambiguous target device identification and fast connection setup – can be resolved.

Section 2 gives some background on how device discovery and connection setup is handled according to the current Bluetooth specifications and what durations for connection establishment result. Section 3 proposes solutions for unambiguous device identification that only rely on Bluetooth as a communication medium. In Section 4, it is shown how the use of a complementary short-range communication medium such as Radio Frequency Identification (RFID) transponders or an infrared interface, integrated into the same client device as the Bluetooth unit, can resolve both the ambiguity and the connection setup time issues. A concluding comparison of connection setup times is made in Section 5. The findings of this paper are summarized in Section 6.

2. DEVICE DISCOVERY AND CONNECTION SETUP IN BLUETOOTH

In Bluetooth, the procedures "inquiry" and "page" are used in order to establish new connections [2]: Using the inquiry procedure, a Bluetooth unit can discover new devices within its range. The page procedure is used to actually set up a connection with a previously discovered device.

Two values are essential in these procedures: the native clock and the device address. Each Bluetooth unit has a free-running native clock with a clock rate of 3.2 kHz. Further, each unit is uniquely identified by a

48-bit Bluetooth Device Address (BD_ADDR) whose addressing scheme is derived from the IEEE 802 standard.

Any Bluetooth unit can act as a master or as a slave. The device initiating a connection to one or more other devices by carrying out a page procedure and thus establishing a "piconet" becomes the master for that particular piconet. The timing and the frequency hopping within that piconet are determined by the device address and the internal clock of the master. Note that a Bluetooth device can simultaneously be part of several piconets.

2.1 Inquiry

If a device wants to discover new devices within its radio range, it needs to perform the "inquiry" procedure. The Bluetooth specifications define 32 dedicated hopping frequencies within the Bluetooth radio band. These 32 frequencies are divided into two partial sequences of 16 frequencies each, the so-called "frequency trains" A and B. During a 10 ms interval, the inquiring device sequentially transmits inquiry messages at all 16 frequencies of either train A or train B. The respective frequency train is repeated 256 times, i.e. for 2.56 s, then the other frequency train is used. After another 2.56 s, the device switches back to the first frequency train, and so forth. This is repeated for at least 4 frequency trains, i.e. 10.24 s.

A device allowing itself to be discovered, performs an "inquiry scan" at least every 2.56 s. It listens for inquiry messages at one of the 32 hopping frequencies. The phase within the sequence of hopping frequencies is determined by the native clock of the device and changes every 1.28 s. Once the device has detected an inquiry message, it waits for a random interval of time between 0 and 639.4 ms, and then starts transmitting inquiry response messages at random times and at different frequencies if it continues to detect inquiry messages. The random time delay has been introduced in order to minimize the risk of collisions between several devices attempting to respond to an inquiry message at the same frequency. The inquiry response message contains the responding device's address BD_ADDR, its native clock and some other device information. The message also indicates whether the device will continuously scan for page messages (scan repetition mode R0), or scan at least every 1.28 s (mode R1) or 2.56 s (mode R2). It should be noted that an inquiry procedure can be done either non-selectively, or selectively for a specific class of Bluetooth devices.

The duration of an inquiry procedure depends on a number of factors: It is increased by a factor of 2 or 3, respectively, if the inquiring device has to serve one or two Synchronous Connection-Oriented (SCO) links (using HV3 packets) at the same time while performing the inquiry procedure. The inquiry time may be increased further in an error-prone environment, i.e. if inquiry and inquiry response messages are not guaranteed to get through, or in the case of collisions between two devices sending an inquiry response at the same frequency and at the same time.

For the remainder of this paper, it will be assumed that no simultaneous SCO links need to be served, neither by the inquiring nor by the scanning device, no collisions occur, and transmission is always error-free. It needs to be remembered, though, that these are best case conditions. Further, we assume in the following that the inquired device operates in scan repetition mode R1.

The maximum duration $T_{i,max}$ of an inquiry procedure is 10.24 s as detailed above. To estimate a typical duration T_i of an inquiry procedure, we assume that with equal likelihood the discovered device will detect the inquiry either during its first or second inquiry scan, depending on whether the frequency it listens at is within the first or second frequency train of the inquiring device. The inquiry scan can occur at any time (with uniform distribution) during the respective 2.56 s interval, therefore its expectation is either 1.28 s or 3.84 s, i.e. 2.56 s in average. To this we must add the expectation of the time the discovered device waits until it responds to the inquiry scan, which is 0.64 s / 2 = 0.32 s. This yields an expectation for T_i of 2.88 s under very optimistic assumptions.

2.2 Page

Once a device has discovered one or more Bluetooth units in its proximity, it may use the "page" procedure in order to actually set up a connection with these units. Similarly to the inquiry procedure, the master transmits page messages at 16 different hopping frequencies (train A) during a 10 ms interval. This frequency train is repeated a number of times, depending on the scan repetition mode of the slave device. In mode R0, where the slave is continuously scanning for page messages, the frequency train does not need to be repeated. In modes R1 and R2, the train needs to be repeated at least 128 and 256 times, respectively, in order to ensure that the slave will detect at least one of the page messages. After the repetitions, the master switches to frequency train B.

The page message contains the BD_ADDR of the slave device to be scanned, and the slave device listens only to its own device address. Independent of the scan repetition mode, the slave device switches the phase of the scanning frequency every 1.28 s.

From the inquiry response message, the master will usually have a reasonably good estimate of the slave's native clock, unless a long time has passed between the inquiry and the page. Therefore, it chooses the frequency train A in such a way that the estimated phase of the hopping sequence (i.e. the frequency at which the master expects the slave to scan) lies in the center of the frequency train. Since the scanning frequency changes every 1.28 s, the estimate of the slave's native clock can be wrong by as much as approx. 10.24 s (= 8 x 1.28 s), and still the master can be sure that frequency train A contains the frequency that the slave is actually listening at. Only if the estimate of the native clock is completely wrong, or if the master does not have any estimate, then the master may have to wait until frequency train B before the slave responds to a page message. Unlike the inquiry procedure, the slave immediately responds to a page message that it receives since the page message uniquely addresses a single Bluetooth device, so that there is no danger of collision.

Similarly to the inquiry procedure, the duration of the setup procedure depends on a number of factors, including error-free vs. error-prone transmission, number of simultaneous SCO links that the paging device is serving and the scan repetition mode of the slave device.

Again assuming error-free transmission, no SCO links, scan repetition mode R1 and a reasonable estimate of the slave's native clock, the duration T_p of the paging procedure is uniformly distributed between 0 s and 1.28 s, i.e. the expectation $T_{p,exp}$ is 0.64 s and the worst-case value $T_{p,max}$ equals 1.28. With one and two SCO links present, both the typical and the worst-case value must be multiplied by 2 and 3, respectively.

Without any estimate of the slave's native clock, it is equally likely that the slave is listening at a frequency within train A or train B. Therefore, the duration is uniformly distributed between 0 s and 2.56 s (without SCO links), yielding an expectation of $2T_{p,exp}$ = 1.28 s and a maximum of $2T_{p,max}$ = 2.56 s.

3. UNAMBIGUOUS DEVICE IDENTIFICATION USING BLUETOOTH ONLY

We first discuss ways of eliminating the ambiguity problem using only Bluetooth as a communication medium [3]. These methods all rely on a manual selection of the desired target device by the user. It is assumed that the physical machines the Bluetooth servers are associated with, are visibly and uniquely labeled, e.g. by names or numbers. Then either a list of servers in range to choose from is presented to the user on their device, or alternatively the user manually enters the desired server ID.

It should be noted that solutions have been proposed that rely on measuring the strength of the received Bluetooth signal in order to determine which target device is the closest [4]. These approaches would require modifications to the implementations of the standard Bluetooth system. In an environment of multipath propagation and fading it is not guaranteed that the strongest signal always corresponds to the closest transmitter. Further, different devices may transmit at different levels. Therefore, unless all Bluetooth servers are coordinated in such a way that their transmit signal strengths are always identical, the physical distance of the transmitter cannot be concluded from the level of the received signal.

3.1 Standalone Solution

In the standalone scenario, there is no connection between the different servers. The user's device performs an inquiry procedure to discover all servers in range and then executes a separate page procedure for each server that is in range. Upon successful connection setup, it requests a (human-readable) device description from each server. Note that the connections to all servers are kept up until the user has made their selection, so that no additional page procedure is required. Bluetooth supports connections with up to seven slaves in active mode. This number can be increased, however, by setting the slave devices into park mode. Parked devices can be reactivated very quickly (2 ms approx.), so that there is virtually no additional loss of time.

The time between the start of the inquiry procedure and a list of all available servers being available for the user is $T_{i,max} + nT_p$ where n denotes the number of servers within range. The time for the inquiry procedure in this case is $T_{i,max}$ because the inquiring device does not know how many inquiry responses to expect. It therefore has to wait for the maximum possible inquiry time to be sure that it has not missed any responses.

Note that it is possible to slightly speed up the connection establishment by always displaying a list of devices to the user that have already been successfully paged. As soon as the user sees the desired server on this list, they can select it and the remaining devices do not need to be paged any more.

3.2 Name Server Solution

In this scenario, each server has access to information about all other servers in the vicinity. For example, in a cluster with ten vending machines, each one has access to a list of all ten machines, including their human-readable device information and their Bluetooth device addresses BD_ADDR. This information can be either retrieved from a central name server, or can be replicated in a peer-to-peer network among all servers. The servers (and name server, if applicable) can either communicate via a separate piconet in Bluetooth, or via a different communication medium, such as an Ethernet LAN.

The user's device performs an inquiry procedure which can be aborted as soon as any one device of the suitable class has responded. It then pages this device and requests a list of available servers from it. As soon as the user has made their choice, it pages the selected device whose BD_ADDR it knows from the list. Note that for this page procedure the master device has no knowledge of the slave's native clock. Of course, it is also conceivable that all servers coarsely keep track of each other's native clocks by periodically exchanging clock information via Bluetooth or via a wired LAN.

In this case the connection setup takes $T_i + 3T_p$ (without clock synchronization) or $T_i + 2T_p$ (with clock synchronization) independent of the number of servers where T_i denotes the time actually needed for the inquiry procedure. It may be less than the maximum time $T_{i,max}$ since the inquiry can be aborted as soon as any server has responded.

Clearly, this solution is preferable over the standalone scenario in terms of connection setup time. One drawback is the increased complexity of the servers that need to keep track of a server list, either by replicating the list or by accessing a name server. Another issue is that it is not guaranteed that the selected target device actually is within radio range of the user's device. This is unlikely though, because the desired server is typically the one closest to the user and should be more likely to be within radio range than the others.

4. UNAMBIGUOUS DEVICE IDENTIFICATION USING A SECOND SHORT-RANGE COMMUNICATION MEDIUM

The above proposals solve the problem of uniquely identifying a Bluetooth server within a cluster of similar machines. The second proposal in addition signifi-

cantly speeds up the connection setup time by eliminating the need to page all devices within range and by aborting the inquiry procedure as soon as any device has responded, thus reducing the inquiry time.

However, both solutions rely on inconvenient manual interaction by the user. Also, connection establishment even in the second proposal may still require a prohibitively long time.

In order to completely avoid any manual interaction and to further accelerate connection setup, we propose to use a second short-range communication medium [3], independent of Bluetooth, for example an infrared interface or an RFID transponder. Both on the client and on the server side, this second interface, i.e. infrared interface, RFID transponder or reader, is integrated into the same physical device as the Bluetooth unit. It is only used to facilitate the unique identification of the desired server and the fast setup of a Bluetooth connection. The actual data interchange and transactions are still performed via the Bluetooth link. Obviously, the second communication medium needs to have a significantly shorter range than Bluetooth (< 1 m) and/or a requirement for a line-of-sight between the two devices. By this means, it is ensured that there is no ambiguity problem: the user's device can only communicate with a single server device which is the closest one and/or the one within line-of-sight. Further, connection setup needs to be significantly faster than in Bluetooth.

Via this second communication medium, the server device communicates its Bluetooth device address as well as a human readable device description to the user's device. The user's device therefore does not need to carry out an inquiry procedure and can immediately page the server Bluetooth unit in order to set up a connection. Unless the native clocks of the two Bluetooth devices are also synchronized via this second communication medium, the connection setup will take $2T_p$.

Note that the actual communication between the two devices is envisioned to be performed only via Bluetooth. A connection using the second communication medium, if one has been established at all, does not need to be kept up, nor do any restrictions apply regarding range or line-of-sight once the Bluetooth connection has been set up.

4.1 Combination with IrDA

The Infrared Data Association (IrDA) has defined a set of protocols for infrared light communications with a range of up to 10 m, requiring a line-of-sight between transmitter and receiver. In particular the IrDA IrMC specification [5] defines the rules for utilization of infrared communication in wireless communications equipment, such as mobile phones, PDA's, pagers and notebook computers. Typical ranges are up to 20 cm (IrMC to IrMC) or up to 30 cm (IrMC to standard IrDA). Due to these short ranges and the line-of-sight requirement, IrMC is well suited to eliminate the ambiguity problem with Bluetooth. During connection setup, the user needs to point their device directly at the target server device with a short distance. Note, however, that once the connection establishment has taken place, the user is free to move within the much larger range of the Bluetooth communications.

IrOBEX [6] is a very lean session level protocol and application framework which is part of – but not confined to – the IrDA protocol family. It has also been adopted by the Bluetooth Special Interest Group as part of the Bluetooth protocol family. OBEX supports both connectionless and connection-oriented transfer of "objects" of any kind between two devices.

OBEX can operate either on top of the "Tiny Transport Protocol" (TinyTP) or on top of the "Ultra" transport protocol [7], which are both part of the IrDA protocol family. If it is used on top of TinyTP, then similar problems as in Bluetooth occur: the device wishing to set up a connection needs to find out the device address of the target device. Therefore, a discovery procedure (a service of the IrDA Serial Infrared Link Access Protocol IrLAP, below TinyTP) needs to be carried out. This can take similarly long as a Bluetooth inquiry (in lab experiments with a laptop computer and a PDA we have found it to take around 6 s).

Ultra is an extremely lightweight transport protocol which offers neither discovery, sniffing or connection-oriented services nor reliable data transport or device addressing. These limitations, however, make it very simple to implement and very fast. The only OBEX mechanism available over Ultra is the OBEX PUT command that allows to push a single object of no more than 255 bytes from one device to another. No connection between the two devices is being established and no device selection via addressing is performed, i.e. every device within IrDA range will receive the pushed object. Since data transport is not reliable, it must be verified by out-of-band mechanisms, e.g. a beep or message box on the receiving device, that the object has been successfully received.

Due to its speed and ease of implementation, we propose to use OBEX over Ultra to transfer a BD_ADDR between the two Bluetooth devices. The time required for the Bluetooth connection setup is then $2T_p$ (without Bluetooth clock synchronization), plus the time for the OBEX object transfer (several 100 ms) which is much lower than the Bluetooth paging time. Two solutions are conceivable: (a) The user's device pushes its BD_ADDR to the Bluetooth server upon initiation by the user, e.g. by pushing a button. The Bluetooth server then pages the user's device via Bluetooth. (b) The Bluetooth server periodically pushes out its BD_ADDR. As soon as a user device comes within IrDA range and receives the address, it pages the Bluetooth server. If the transmission of the BD_ADDR is repeated very fast, there should almost no time delay. The second solution has the advantage that no user interaction is required.

4.2 Combination with RFID Tags

Radio Frequency Identification (RFID) stands for a broad range of technologies for identifying physical objects using RF communication between a reader in the proximity of the object and a "transponder" or "RFID tag" on the object itself [8]. Typically, a transponder consists of a single integrated circuit and an external antenna. "Passive" transponders are available that do not require a battery or other power source, but utilize the power from the electromagnetic field of the

reader to charge a small capacitor. The capacitor is then discharged in order to send a short response signal to the reader. RFID solutions operate at a broad range of radio frequencies, from 60 kHz all the way to 5.8 GHz, depending on the application. Ranges are typically between a few cm and some meters. Depending on the frequency, line-of-sight between the reader antenna and the transponder may or may not be required. RFID tags can typically store small pieces of information (some bits up to a few kbit), such as an identification code for the object.

For the scenario considered in this paper, we propose to attach a passive transponder to the user's mobile device which stores the BD_ADDR of the Bluetooth unit in the user's device. An RFID reader should be integrated into the target device, e.g. vending machine. Obviously, the RFID reader should operate outside the 2.45 GHz ISM band lest the Bluetooth and RFID systems interfere with each other. Since typical passive RFID systems have ranges of some 10 cm approximately, the user needs to bring their device in close proximity to the target device in order to establish a connection. Once the RFID reader in the target device has been able to read the transponder, it pages the Bluetooth unit of the user's device whose BD_ADDR it knows from the RFID tag. The time required, again, is $2T_p$ since no Bluetooth clock synchronization takes place. The time for reading the transponder is much lower than the Bluetooth paging time – typically in the range between 10 ms and 100 ms.

This proposal solves both the unique identification problem and drastically reduces the connection setup time. It must be noted, however, that it adds costs and complexity to the system. While the costs for simple passive transponders are rather low today (less than $1), the costs for the reader infrastructure in the target devices may not be negligible.

5. COMPARISON OF CONNECTION SETUP TIMES

As detailed above, the actual connection setup times for each of the considered methods depend on a number of assumptions, e.g. the scan repetition mode of the target devices, the likelihood of transmission errors, clock synchronization, the delays caused by IrDA or RFID systems, etc. However, trends and orders of magnitudes can be indicated that allow a comparison between the three considered systems. Table 1 summarized the connection setup times based on the assumptions stated in the text above. For each method, a worst case and a best case value are given, as well as a "typical" value which is essentially the expectation under the discussed typical assumptions. We have further assumed that the IrDA or RFID communication cycle requires 500 ms which is a rather pessimistic assumption.

standalone Bluetooth	worst case	10.24 + *n* 1.28
	typical case	10.24 + *n* 0.64
	best case	10.24
Bluetooth with name server	worst case	14.08
	typical case	4.80
	best case	0.00
Bluetooth with IrDA or RFID	worst case	3.06
	typical case	1.78
	best case	0.50

Table 1: Connection setup times in s for the three systems. *n* is the number of target devices within range.

6. SUMMARY AND CONCLUSIONS

We have considered application scenarios for Bluetooth where a user device communicates and performs transactions with a target device such as a ticket or vending machine or point-of-sales terminal. Challenges for this type of applications are the long inquiry and page time in Bluetooth as well as the unique identification of the right target device among a cluster of similar or identical target devices within Bluetooth range.

We have shown that solutions relying on Bluetooth only as a communication medium always require a manual selection of the target device by the user. Also, connection setup times can be prohibitively long. They can be reduced by introducing a name server or directory service that all server devices have access to. Further, when using a name server, the connection setup times become independent of the number of devices within range.

We have then proposed to utilize a second communication medium, integrated into the same client device as the Bluetooth unit, to facilitate the connection establishment. While this obviously adds costs and complexity to the system, it eliminates the need for manual selection of the target device and accelerates the Bluetooth connection setup time to a reasonable level (approx. 3 s max., 1.5 s typical). As two examples, we have shown how this can be implemented using infrared communication or RFID transponders.

REFERENCES

[1] J. Haartsen: "The Universal Radio Interface for Ad Hoc, Wireless Connectivity", Ericsson Review, No. 3, 1998, pp. 110-117.

[2] Bluetooth SIG, "Specification of the Bluetooth System, version 1.1", chapter 10.6-10.7, http://www.bluetooth.com, February 2001, pp. 96-109.

[3] I. Herwono, "Solutions for Selective and Non-Ambiguous Identification of Bluetooth Devices", Technical Report, Communication Networks, Aachen University of Technology, October 2000.

[4] H. Hall, A. Aquilon, S. Willehadson, "Method and Device for Wireless Telecommunication", International Patent Application WO 00/51293, August 2000.

[5] Infrared Data Association, "Specifications for Ir Mobile Communications (IrMC), version 1.1", March 1999.

[6] Infrared Data Association, "IrDA Object Exchange Protocol (IrOBEX), version 1.2", March 1999.

[7] Infrared Data Association, "Guidelines for Ultra Protocols, version 1.0", October 1997.

[8] Klaus Finkenzeller, RFID Handbook, Wiley, New York, 1999.