Bluetooth Real-Time Mobile Auctions

F. J. González-Castaño, F. J. Gil-Castiñeira, J. García-Reinoso
Departamento de Ingeniería Telemática, Universidad de Vigo
ETSI Telecomunicación, Campus, 36310 Vigo, Spain
bDepartamento de Ingeniería Telemática, Universidad Carlos III de Madrid, Spain
{javier,xil}@det.uvigo.es

Abstract—A practical real-time mobile auction system must fulfill diverse constraints. In [1], we described a mobile auction system implemented with 433 MHz band modems. Nowadays, that system has become a standard in the Galician fish markets, which include the most important fishing port in Europe, Vigo (Spain). Although our system has been performing satisfactorily for four years, it is obvious that new technologies such as Bluetooth and IEEE 802.11 have shaped the wireless market and deserve our attention. In this paper, we evaluate Bluetooth in the context of real-time mobile auctions. As a result, we propose a novel approach to implement mobile auction systems by exploiting the park mode in Bluetooth specifications, which may also be useful for other remote-command applications with hard real-time constraints.

I. INTRODUCTION

Galicia (Spain), like other Atlantic regions, obtains a large percentage of its income from sea industries: In-shore and deep-sea fishing, canning industries and shipyards. One of the main Galician cities, Vigo, is the most important fishing port in Europe. In-shore Galician fishermen are organized in guilds (cofradías). These guilds have annual fish sales in the range of 100-200 million Euros/US$ [2].

According to European Union food handling regulations, nowadays it is compulsory to establish a physical barrier between food and dealers. In some places, such as Plymouth Fish Trade (UK) or Ostend Fish Auction (Belgium), dealers sit in desks with bid buttons, in dedicated auction rooms. An electronic clock displays price count-down [3].

In the case of Galicia, this solution is not desirable, for two reasons:

Tradition: For centuries, dealers have been allowed to move around. Mobility is, therefore, a critical issue.

Tourism: In-shore fish auctions are important tourist destinations at Galician sea ports. Any solution should have little impact on this secondary profit source, by introducing as few changes as possible in rooms.

As a result, the Galician government decided to fund a system to allow dealers make their bids from mobile terminals. The system does not introduce significant changes in the auction room. In addition, the following performance constraints must hold:

A. An average person can hardly distinguish two manipulations separated by 0.1 s. Therefore, elapsed time $T_s$ between any winner bid and auction stop must be $\sim 0.1$ s or less (in our implementations, this deadline determines price persistence time, 0.25 s).

B. Once auction stop is signaled, elapsed time $T_w$ between auction stop and winner determination can be as long as 1 s (even longer, if the stop alarm includes a “slow” message display on the auction screen).

The rest of this paper is organized as follows: section II presents the background of this research. Section III explains our Bluetooth mobile auction solution. Section IV describes the simulations we performed to validate our proposal. Finally, section V concludes.

II. BACKGROUND

In [1], we describe a cellular system that satisfies constraints A-B among others, jointly developed by Universidad de Vigo and SAEC-DATA S.A. (Spain). In that system, each auction room is a cell. Both the base station that defines the cell and the mobile terminals of the customers are microcontroller systems with 433 MHz modems, which implement a bidirectional automated auction protocol. The Media Access Control (MAC) layer relies on a combination of Time Division Multiplexing (TDM) and slotted Aloha.

For a review of some technologies related to automated auctions see [3]. Some application examples are the European fish auctions described in the previous section, the Dutch flower industry [4], the Nigerian coffee exchange [3] and diverse Internet auctions [5].

Figure 1 shows a view of the Bayona auction room (Spain), where one of the systems described in [1] is installed. Al-
though those systems have been operating satisfactorily for four years, it is obvious that new technologies such as Bluetooth and IEEE 802.11 Wireless Local Area Network (WLAN) have shaped the wireless market and deserve our attention. In this paper, we propose a novel approach to implement real-time mobile auction systems by exploiting the park mode in Bluetooth specifications.

We have chosen Bluetooth for three main reasons:

1. The terminals in [1] are too specialized and, as a consequence, considerably more limited than a general-purpose Personal Digital Assistant (PDA) or a mobile phone with Bluetooth Application Programming Interface (API), as the new devices implementing Java Specification Request JSR 82 [6]. Wireless PDAs and Bluetooth phones are still much more expensive than our terminal, but this situation will change in a close future. In addition, they may support a wide range of applications related to auction activities (management, personal records, etc).

2. Regarding spectrum management, 433 MHz modems are very limited, compared to advanced technologies like IEEE 802.11 or Bluetooth.

3. Ideally, we are interested in a Radio Frequency (RF) technology included in commercial devices. To understand this statement, consider the case of IR ports, which are included in most PDAs and are not considered an option any longer; they are simply there. This may be the case of Bluetooth in the close future. Diverse state-of-the-art terminals include Bluetooth modems [7], [8].

III. BLUETOOTH MOBILE AUCTIONS

A. Bluetooth

Bluetooth [9] is a short-range wireless networking technology. The typical range for class 2 modems is 10 m [10]. It has been designed for applications such as wireless headphones and computer-to-peripheral communications. A Bluetooth master can handle up to seven slaves in connected mode [9].

In principle, there are two ways to support the large number of users that may fit within the range of a class 2 modem (~300 m²):

- **Simultaneously connected**, by defining a scatternet. This means that many user terminals will be both masters and slaves. According to our experience [11], [12], the resulting configuration would be extremely unreliable.

- **No connections are established a priori**. In this scenario, user terminals know the Bluetooth address of the piconet master (auction base station). Whenever a user presses the bid button, the Bluetooth modem in its terminal waits to enter connected state (to obtain a piconet connection to transmit the bid). Previous authors have determined that the typical delay to establish a point-to-point Bluetooth connection is longer than 2 s [13]. Therefore, it is not possible to satisfy constraints A-B in section I.

On the other hand, Bluetooth has interesting advantages: low power consumption (~1 mA standby and ~60 mA peak [14]), small modem size (1 cm²) and low cost (5-10 US$) [15]. Also, as we said above, state-of-the-art commercial terminals include Bluetooth modems.

Therefore, we studied a way to avoid the seven-slave boundary, while still satisfying constraints A-B. Bluetooth specifications state that, when a slave does not need to participate in the piconet channel but wants to remain synchronized with the channel, it can enter the low-power park mode with minimum activity [9]. This is obviously interesting for battery-dependent systems (like ours). In addition, the park mode allows to handle more than seven communications between a single master and its slaves. The number of slaves that can be parked is only limited by Bluetooth addressing range. Kim et al have proposed to use this mode to implement an access point for Internet browsing from multiple terminals [16] (although for that particular application they needed several master modems). However, as far as we know, there are no previous proposals in the context of hard real-time remote-command applications, like ours. As a matter of fact, none of the most important Bluetooth research simulators (Bluehoc [17], Bluescat [17] and Blueware [18]) considered the park mode at the time this paper was written.

When one or more slaves are parked, the master establishes a beacon channel (figure 2) to:

1. transmit master-to-slave packets so that the parked slaves can re-synchronize.
2. transmit messages to change the beacon parameters in the parked slaves.
3. transmit general broadcast messages to the parked slaves.
4. unpark one/more parked slaves.

At this stage of Bluetooth specifications, only the polling access technique has been defined. Figure 3 shows an example of beacon channel access window (9), page 115), adapted to our application (there may be any number of access windows in a beacon slot, even none). Each unit in the access window is composed of a master-to-slave slot and two slave-to-master ones, potentially shared by several slaves. The master polls the slaves in master-to-slave slots. When a slave enters the park state, the master assigns a half-slot to its (shared) Access Request Address (AR_ADDR). A parked slave is allowed to send responses to the master in its half-slot, identified by AR_ADDR. Parked slaves can send requests to be unparked in their slave-to-master slots.

B. Bluetooth mobile auction solutions

We propose the following Bluetooth mobile auction protocol:

- **Before each new auction, the master collects the addresses of all Bluetooth modems present. If a given address is included in the list of valid users, the master activates the corresponding modem and parks it (a modem must be activated before it can be parked).**

- **When a new device is parked, the master must update the beacon parameters to add the device to the access window.**
Once the auction starts, the master does not generate more inquiry cycles. It assigns all transmission time available to the beacon channel, by setting $T_B = T_{ACCESS} + D_{ACCESS}$ (and, therefore $T_Q = 0$ s). Then, it waits for unpark requests in slave-to-master slots. When a user presses a button in its terminal to stop the auction, its modem sends an unpark request to the master in its half-slot. As soon as the master detects one of such requests, the auction stops.

There are two possible strategies depending on our optimization goal. We can choose to minimize elapsed time between any winner bid and auction stop ($T_s$), or to minimize elapsed time between auction stop and winner determination ($T_w$).

1) Best-$T_w$ solution: In this solution, for 250 users\(^1\), the access window has 125 master-to-slave slots to poll the slaves and 125 slave-to-master slots (with two unique AR_ADDR addresses per slot). In the worst case, since $T_Q = 0$ s and $D_{ACCESS}$ is negligible, $T_s = 250 \times 625 \mu s = 0.15625$ s, and $T_w = 0$ s for the primary winner. Although $T_s$ is worse than the result in [1], it satisfies constraint A and it is possible to poll all users (and, as a consequence, define any set of winners).

2) Best-$T_s$ solution: It is possible to improve $T_s$. We have previously said that AR_ADDR addresses are not always unique. In the best-$T_w$ solution, we assign a different AR_ADDR address to each parked slave. Thus, when the master receives an unpark request, it immediately knows the identity of the corresponding user.

If several devices share the same AR_ADDR address and they wish to stop the auction, the master will receive a single unpark request (all transmissions are synchronized [9]). Obviously, this improves $T_s$, because the master polls *modem groups* instead of individual modems. On the other hand, it is necessary to establish the identity of the winner(s) within the group that raises the unpark request, which increases $T_w$. However, note that a considerably long $T_w$ value ($\approx 1$ s) is affordable according to constraint B.

Once the auction has stopped, it is necessary to activate all the slaves in the group and interrogate them to determine the winner. In the best-$T_s$ solution, each group can have seven slaves at most. With 250 users, 36 groups are necessary. Therefore, $T_s \approx 36 \times 625 \mu s = 22.5$ ms. This value is similar to the result achieved in [1] for 433 MHz band modems.

The master sends as many Link Manager Protocol (LMP) messages as slave Active Member Addresses (AM_ADDR) in the winner group. Then, the master must check the success of the unparking procedure by sending POLL packets to the slaves. According to the specifications, unparked slaves should answer with a LMP_accepted packet. Then, the master must check all slaves in the winner group. If the group is composed of seven slaves, this check takes the time to receive LMP_accepted packets from every device in the group and the time to receive a packet from each polled device indicating if it generated a bid or not. Consequently, the total time is $T_w = (625 \mu s \times 7 \times 2) \times 2 = 17.5$ ms (we are not considering the time to send the UNPARK packet, which is comparatively irrelevant).

**Remark:** If we were allowed to modify the LMP layer in the slaves, we could order them to transmit LMP_accepted packets only if the user had pushed the bid button. Then, in the worst case, $T_w$ would be the total time to receive seven LMP_accepted packets: $T_w = 625 \mu s \times 7 \times 2 = 8.75$ ms.

We conclude that both solutions -best-$T_w$ and best-$T_s$- are feasible from the point of view of constraints A-B, and we are free to choose one or another according to the relative importances of $T_s$ and $T_w$ in a specific scenario.

**IV. Simulations**

**A. Bluehoc modifications**

Bluehoc is an extension of the well-known ns network simulator [19]. Bluehoc includes the following Bluetooth features: Baseband, Logical Link Control Adaptation Protocol (L2CAP) and LMP. To validate the theoretical calculations in the previous section, we added park mode features to Bluehoc 3.0, based on ns-2.1b8a. The code is available from the authors upon request by e-mail.

Bluehoc implements each layer of the Bluetooth protocol stack as a C++ class. The classes required are integrated in an oTCL program. Basically, there exist the following classes:

- **Baseband**: Implements the baseband layer, as well as clock management and data transmission (radio and physical medium). There is an instance per node.
- **BT.DRRScheduler**: Management algorithm for the data transmission queue. As in the case of the previous class, there is an instance per node.
• **LinkController:** Link control. There is an instance per active node connection. There is also a leaky bucket filter to manage each queue.

• **LMP:** Implements some LMP commands.

• **L2CAP:** Basic L2CAP functionality for data transmission.

• **BTHost:** Simulates a Bluetooth host that would be associated to the physical device. There is an instance per node.

In order to implement the auction system, it was necessary to modify the following classes:

• **BTHost.** To implement the base station, we added mechanisms to:
  - Find all terminals present.
  - Request the baseband layer to set connections with all terminals.
  - Request the LMP layer to park all terminals.
  - Request the LMP layer to enter park mode.
  - Collect terminal bids.

To support user terminal operation, we added mechanisms to admit connection requests from the base station.

• **LMP.** We have implemented the LMP.park.req command. The base station calls it once per user terminal. We have also implemented its reception at user terminals, so that BTHost can park them with any parameter choice.

• **Baseband.** We have added the park mode, with all the associated actions. We have also implemented the transition to park mode from connected state.

The elements added in Baseband and LMP are part of Bluetooth specifications. The modifications in BTHost correspond to the application level of the auction system. The major effort was devoted to Baseband. Baseband operation involves clock handlers (simulating the clock tick), timers, and emission/reception functions depending on the current state, which is determined by a state variable. We have implemented a new park state that we consider in every called function.

The new LMP functions control park mode parameter passing between master and slaves and the transition from active to park states.

### B. System initialization

At system initialization, it is necessary to set connections with all terminals present and park them. Note that, although only seven simultaneous active connections are possible, as soon as a connection is established the corresponding terminal can be parked and it quits connected state. Even assuming the largest number of competing users recorded in Galician auctions so far, 150, a single inquiry cycle (10.24 s) is not enough to detect all their terminals (although it detects most of them). Figure 5 shows the number of terminals detected in an inquiry cycle, for up to 150 users present. Even assuming the need of several inquiry cycles, this "massive" user terminal detection only has to be performed once, and it is valid for all subsequent auctions.

**Remark:** Alternatively, we could force user terminals to set connections with the master prior to parking. To implement this approach, user terminals should know the master address (fixed) and they should include a start routine to be manually activated before the auction starts.

### C. Average \( T_s \) estimation

Since \( T_w \) is not subject to a hard deadline, we have focused on the evaluation of \( T_s \). In sections III-B.1 and III-B.2 we developed worst-case bounds for \( T_s \), but it is interesting to estimate its average value. In our simulations we used the modified version of Bluehoc described in section IV-A. Given a common starting time (for a certain price), all users press their bid button with an exponentially distributed delay with 0.1-s average. Figure 6 shows the average value of \( T_s \) across 10 trials, for the best-\( T_w \) case (the best-\( T_w \) case is an upper bound for the best-\( T_s \) case, since \( T_{w_{\text{best}}}^s(n) = T_{s_{\text{best}}}^w(n) \)), where \( n \) is the number of users present). As we expected, the error bars lie below the theoretical upper bound in section III-B.1. They almost encompass the entire range between 0 and the theoretical maximum due to the fact that a bid may appear anytime during the park mode query cycle.
D. Implementation considerations

The implementation of our system with a Bluetooth Integrated Circuit (IC) equipped with park mode is feasible. Most current ICs employ this low power mode. For example, CSR is the key player in the Bluetooth market nowadays [15], and its current Bluetooth ICs are park-mode capable. Let us consider the BlueCore2-External BC02 case. This is a single-chip baseband+radio IC [22]. It is Bluetooth 1.1 and 1.2 compliant and it only requires a few external components to be configured as a class 1 (10-m range) or class 2 device.

BC02 contains a microcontroller core. Besides of Bluetooth connections, the BlueLab Software Development Kit (SDK) supports the park mode through command DM_HCI_PARK.MODE. A bid would consist of the transmission of a DM_HCI_EXIT_PARK.MODE to the server (notified with a DM_HCI_MODE_CHANGE_EVENT_T primitive).

V. CONCLUSIONS

In this paper, we propose a Bluetooth implementation for the mobile auction system in [1], which is fully operational in Galician inshore-gold facilities. Our strategy is based on the park mode, which, as far as we know, is a novel approach in the context of hard real-time remote-command applications. Our results can be applied to scenarios with similar requirements, like real-time active Radio Frequency Identification (RFID).

We close the paper with a comment on security. Bluetooth is considerably secure, since it uses a hop selection mechanism with up to 1600 hops/sec. This makes attacks difficult, because the attacker must be synchronized. In addition, we can take advantage of the three security modes defined in the specifications. In the third one, the Bluetooth device initiates security procedures at link level before the channel is established. If we use this method, the auction procedure is protected even against synchronized attackers, because they would not know park mode parameters. Previous work has shown that Bluetooth is better protected than IEEE 802.11b [20], despite of some weaknesses that are not important in practice [21].

Remark: it would be possible to add extra protection by altering the specifications: we could add a payload to BT_ID packets [9], to send a prearranged code or a signature.

References