Improving Bluetooth Network Performance Through A Time-Slot Leasing Approach

Wensheng Zhang, Hao Zhu, and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802

***Abstract*— Bluetooth is a promising technology aiming at supporting short-range wireless communication. To achieve the advantage of simplicity and low-power, the master/slave model is used. However, this model has some drawbacks since no direct link exists between any two slaves in a piconet. Consequently, slave-to-slave communications must go through the master, and the master has to use extra bandwidth to forward the packets exchanged between slaves. In this paper, we propose a time-slot leasing (TSL) and an enhanced TSL (ETSL) approach to address these drawbacks. Simulation results demonstrate that the TSL approach, especially the ETSL approach, can significantly improve the system performance compared to the standard master/slave model.**

***Keywords*—Bluetooth, ad hoc networks, leasing, piconet.**

I. Introduction

Bluetooth [1], [2], [3] is a promising technology which is aimed at supporting wireless connectivity among mobile devices such as cellular phones, headsets, PDAs, digital cameras, laptop computers and their peripheries. The technology enables the design of low-power, small-sized, low-cost radios that can be embedded in existing portable devices. Initially, the technology will be used as a replacement for cables. Eventually, it will provide solutions for point-to-multipoint and multihop networking, lead toward ubiquitous connectivity and truly connect everything to everything.

In Bluetooth, a group of devices sharing a common channel is called a *piconet*. Multiple piconets can co-exist because each uses a different hopping sequence. Each piconet has a *master* unit, which controls the access to the channel, and at most seven *slaves* as group participants. Piconets can also be interconnected via *bridge* [4], [5] nodes to form a bigger ad hoc network known as a *scatternet*. Within a piconet, the channel is shared using a slotted time division duplex (TDD) protocol where the master uses a polling style protocol to allocate time-slots to slave nodes. The master should start its transmission in even-numbered time slots (*master-to-slave*) only, and the slaves should start its transmission in odd-numbered time slots (*slave-to-master*) only. To avoid collisions among slaves, a slave is permitted to transmit in the slot-to-master slot if and only if it has been addressed in the preceding master-to-slave slot. As a result, two slaves cannot communicate with each other directly, and their communication must be forwarded [6] by the master.

The major advantages of the *master/slave* model [7] in Bluetooth are *simplicity* and *low-power*. Since the master centrally controls the usage of the time slots, there is no collision and slaves are stateless. The TDD protocol can save power since slaves can sleep during the slave-to-master slots whenever it is not addressed in the previous master-to-slave slots. However, the master/slave model has some drawbacks. Since no direct link exists between any two slaves within the same piconet, the master has to forward the packets exchanged between them. For example, for each packet transmitted from a slave S_1 to another slave S_2, two transmissions are needed: $S_1 - to - master$ and $master - to - S_2$, which will double the bandwidth consumption and the communication delay.

The drawbacks with the master/slave model can be addressed by the solutions based on the idea of dynamic structure/role management [8], which reconfigures the scatternet/piconet structure or device roles based on the traffic pattern. When a slave needs to frequently communicate with other slaves, a simple solution called *master-slave switching (MSS)* [8] can be used. In this approach, the slave can switch its role with the master and become the new master. In another simple solution, called *piconet partition (PP)*, if there are frequent communications between a pair (group) of slaves, these slaves can form a new piconet and a new master is appointed. However, from the perspective of the scatternet, these approaches may change the scatternet topology, which brings new issues such as scatternet topology stability, and scatternet global optimization.

In this paper we propose a new approach, called *time-slot leasing (TSL)*, to address the problems associated with the master/slave model. This approach can improve the performance by establishing temporary piconets which will lease slots from the original piconet to support slave-slave communication. TSL approach can be further enhanced to allow each temporary piconet to have its own dedicated channel. Both TSL and the enhanced TSL (ETSL) approach do not change the scatternet/piconet structure permanently and have no negative effects on inter-piconet communications.

The rest of the paper is organized as follows. Section II describes the TSL approach. The ETSL approach is presented in Section III. In Section IV, we evaluate the performance of the proposed solutions. Section V concludes the paper.

II. The Time-Slot Leasing (TSL) Approach

A. Basic Ideas

Different from the MSS approach and the PP approach, TSL only temporarily changes the piconet structure. Specifically, the master can temporarily lease some time slots to the slaves,

which frequently communicate with each other, so that they [1] can form a *temp-piconet*. The master specifies one slave as the *temp-master* of the temp-piconet. This temp-piconet is time- and frequency-synchronized with the original master. Thus, the temp-master can only use the assigned time slots to communicate with slaves in the temp-piconet. At the same time, the master can also communicate with the slaves in the temp-piconet on any time slot except those assigned to the temp-master. The master or the slaves in the temp-piconet can terminate the lease sometime later. Before the lease terminates, slaves in the temp-piconet have the flexibility to use the leased time slots, and the original master cannot use these time slots. After the lease terminates, the temp-piconet is removed and the slaves return to their original role.

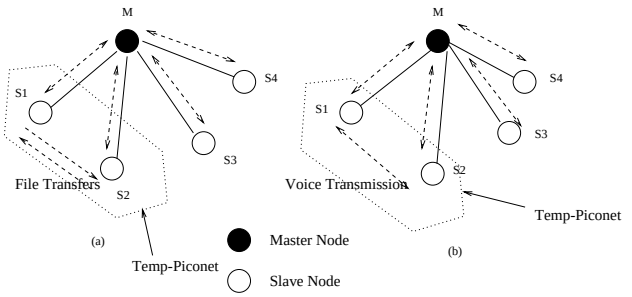

Fig. 1. Examples of slave-slave communication in piconets: (a) supporting slave-slave file transfer; (b) supporting slave-slave voice transmission.

Fig. 1 shows two application scenarios, where the solid lines represent the master-slave links, and the dashed lines represent the traffic flows. In the scenario shown in Fig. 1(a), large files are transferred between S_1 and S_2. In the scenario shown in Fig. 1(b), S_1 and S_2 need to exchange time-bounded information like voice. In both scenarios, the master has to exchange packets frequently with its slaves. The slave-slave communications in these scenarios can be supported efficiently using TSL. For example, when an application process in S_1 has a large file to send to another process in slave S_2 (as shown in Fig. 1(a)), the network layer entity in S_1 may be informed and S_1 will request the master to set up a temp-piconet consisting of S_1 and S_2, via the Bluetooth link management protocol (LMP). Similarly, a temp-piconet consisting of S_1 and S_2 can be set up in scenario (b).

B. The TSL Protocol

In this subsection, the protocol for implementing the TSL approach is described.

B.1 Notations

Before presenting the protocol, we introduce some notations as follows.

[1] Assume these slaves are within each other's direct communication range.

- *Leasing slot pattern* (D_l, T_l): The 2-tuple defines the set of slots leased to a temp-piconet. The slots are leased to a temp-piconet in the unit of *slot pair* (two consecutive slots). D_l specifies the first slot leased to the temp-piconet. T_l defines the intervals at which the slot pairs are leased to the temp-piconet. For the example shown in Fig.1(a), if the temp-piconet's leasing slot pattern is (2,8), then during the lifetime of the temp-piconet, the slots whose number n is equal to $8*i+2$ or $8*i+3$ where i is an integer, are leased to the temp-piconet. Certainly, the protocol can be extended to allow more general kinds of leasing. For example, leasing 4 or 6 slots in each T_l instead of 2 slots. For simplicity, we only show the case of using two consecutive slots.

- *Slot maps* $M_p[]$, $M_{s_i}[]$: In a piconet, the master has a slot map $M_p[]$ which indicates the allocation status for each slot of the channel, and each slave S_i has a slot map $M_{S_i}[]$ which indicate whether it is busy in each slot. $M_p[j] = 1$ if the j^{th} slot is already reserved for some activity; $M_{S_i}[j] = 1$ if S_i is busy in the j^{th} slot.

The following are some LMP packets designed for the TSL protocol.

- *setup_req*, *setup_ack*, *setup_rej*: The *setup_req* packet is sent from a slave to the master to request for setting up a temp-piconet with another slave. The parameters of the packet include: the requested interval between leasing slot pairs (denoted as T_{req}), the slave's slot map. The master may send back a *setup_ack* packet as an acknowledge to the request, or a *setup_req* as a reject.

- *slots_inq*, *slots_ack*: The *slots_inq* packet is sent from the master to a slave for inquiring its slot map. The packet *slots_ack* is sent as a reply to *slots_inq*, and the slave's slot map is included in the packet.

- *terminate_req*, *terminate_ack*: The *terminate_req* packet is sent from a node to request for terminating an existing temp-piconet. The *terminate_req* packet is a reply to *terminate_req*.

B.2 The Protocol

There are three steps in the protocol: setting up a temp-piconet, transmitting packets using leased slots, and removing an existing temp-piconet.

Setting Up A Temp-Piconet

The procedure for slaves S_1 and S_2 to set up a temp-piconet is as follows:

S_1 sends a *setup_req* packet to the master. After receiving the request, the master sends S_2 a *slots_inq* packet. S_2 returns a *slots_ack* packet. When the master receives the *slots_ack* packet, the *find_slots* procedure (shown in Fig.2) is called to decide whether the temp-piconet can be set up and which slots will be leased to the temp-piconet. If the master doesn't allow

Procedure: find_slots

A: The master attempts to find a slot pattern (D_l, T_l) which satisfies the following conditions:

1: If the requested transmission requirement is hard, $T_l = T_{req}$; if the requirement is soft, $T_l \geq T_{req}$.

2: For any $j = (D_l + i \times T_l)$, $M_p[j] \vee M_{S_1}[j] \vee M_{S_2}[j] = 0$, where $i = 0, 1, 2, ...$, and S_1 and S_2 are the slaves to form a temp-piconet.

B: If the slot pattern can be found, the master agrees to set up the temp-piconet and the slots specified by the slot pattern (D_l, T_l) are leased to the temp-piconet. Otherwise, the request for setting up temp-piconet is rejected.

Fig. 2. The find_slots procedure

the temp-piconet to be set up, it sends a *setup_rej* to S_1 and S_2 respectively, and then the setup procedure fails. If the master decides to establish the temp-piconet, it sends a *setup_ack* to S_2 and waits for its reply. If the master receives a *setup_ack* from S_2 before it times out, it sends a *setup_ack* to S_1 and waits for its reply. If the master receives a *setup_ack* from S_1 before it times out, the establishment finishes successfully. Otherwise, the master sends *setup_rej* to both slaves, and the setup procedure fails. Fig.3 illustrates how S_1 and S_2 can successfully set up a temp-piconet.

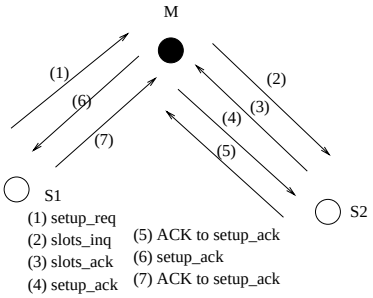

Fig. 3. A successful temp-piconet setup procedure

Transmitting Packets Using Leased Slots

During the leased slots, the slave assigned as the temp-master will transmit at the master-to-slave slots and the other slave will transmit at the slave-to-master slots. The two slaves are not allowed to communicate with nodes outside of the temp-piconet during the leased slots. When a slave has nothing to transmit to the other one in the temp-piconet, a default packet such as a NULL packet is sent. During these slots, the original master is not allowed to transmit.

Removing The Temp-Piconet

A temp-piconet can be terminated by the master, the temp-master or the slave in the temp-piconet. The procedure of a temp-piconet termination initiated by the temp-master is presented as follows: When the temp-master (S_1) wants to terminate the temp-piconet, it sends a *terminate_req* to the master. The master forwards the packet to the other slave in the temp-piconet (S_2). If S_2 does not have any packet to send to S_1, it replies a *terminate_ack* to the master. When the master receives a *terminate_ack* from S_2, it forwards the packet to S_1. From now on, the master can use the leased slots. When S_1 receives the *terminate_ack*, it returns to the original role before setting up the temp-piconet.

III. The Enhanced TSL (ETSL) Approach

The TSL approach can improve the performance of a Bluetooth network without permanently modifying the scatternet/piconet structure. However, there is a drawback in this approach. Since all the temp-piconets share the same channel with the original piconet, once a slot is leased to a temp-piconet, any other node outside of the temp-piconet is not allowed to transmit during this slot unless it can switch to another piconet of which it is also a member. Consequently, the system throughput improvement is limited by the available bandwidth of a single channel. To address the problem, we propose an enhanced TSL (ETSL) approach. The ETSL approach allows each temp-piconet to use its dedicated channel defined by its temp-master, and all the temp-piconets are time-synchronized to the original piconet for easy coordination. With the ETSL approach, slots can be leased to multiple temp-piconets which have no common member. During the leased slots, both the temp-masters and the original master are allowed to transmit simultaneously, since they use different frequencies. In this way, the actual bandwidth of the piconet is increased. The ETSL approach is different from the PP approach in which a piconet is broken into two or more interconnected independent piconets. In the ETSL approach, the temp-piconets are not independent or permanent, they exist only during the leased slots specified by the original master. For other time slots, the members of the temp-piconet act as slaves of the original piconet.

To implement the ETSL approach, the following enhancements should be done to the TSL protocol presented in the previous section:

- The *setup_ack* packet should include the temp-master's Bluetooth address, so that the slave of the temp-piconet is able to frequency-synchronize itself with the temp-master based on the frequency hopping sequence deduced from the address.
- Step **A.2** of the procedure *find_slots* is changed as follows. For any $j = (D_l + i \times T_l)$, $M_{S_1}[j] \vee M_{S_2}[j] = 0$, where $i = 0, 1, 2, ...$, and S_1 and S_2 are the slaves to form a temp-piconet. The slot usage information about other slaves is irrelevant, since each temp-piconet will have its own dedicated channel and the slots leased to different temp-piconets can be

overlapped.

- A slave in a temp-piconet switches between multiple channels like a bridge node. Specifically, during the slots leased to the temp-piconet, the slave synchronizes its frequency hopping sequence to the temp-piconet; during other slots, its frequency hopping sequence is synchronized to the original piconet. However, they keep time-synchronized to the original piconet all the time.
- The original master is allowed to transmit during the slots which has been leased to a temp-piconet.

IV. Performance Evaluations

In this section, we evaluate the performance of the proposed approaches.

A. Simulation Model

We study the performance of the proposed approaches through discrete event simulations. Our simulation uses the piconet structure illustrated in Fig. 1. For simplicity, the packets transmitted in the piconet are assumed to have the same size of one slot, and packet transmission errors are not considered. We evaluate the performance of our approaches based on two kinds of communications. One has real-time requirements, such as voice. The other does not have real-time requirements, such as file transfer. For voice, which is also referred to as the constant bit rate (CBR) transmission, we use the average packet transmission delay as the performance metric. For file transfer, the performance metric used is the file transfer delay, which is defined as the delay for transmitting the whole file.

B. Simulation Results

B.1 Performance Evaluation (with Slave-Slave CBR Transmission)

We first compare the performance of the standard master/slave model and the proposed approaches when slave-slave CBR transmission is supported. Suppose some voice transmission is requested between S_1 and S_2 in two directions simultaneously. The constant transmission rate of the traffic is $64kb/s$ for each direction. The voice information is sent using HV3 packets [1], which have a payload length of 240bits and can carry $3.75ms$ of speech at a rate of $64kb/s$. With simple math, the transmission rate can also be represented as $267\ packets/s$ for each direction.

In a piconet using the standard master/slave model, the transmission task has to be supported by two SCO links: the $master-S_1$ link and the $master-S_2$ link. Each link reserves two slots every six slots, and the master polls the slaves in the Round Robin manner during other slots. When TSL or ETSL is used, a temp-piconet consisting of S_1 and S_2 will be set up, in which slave-slave packets are transmitted directly using leased slots. For TSL, $T_l = 6\ slots$; that is, two slots in each six slots are leased to transmit the voice packets, and the master polls the slaves in a Round Robin manner during other slots. For ETSL, the master polls the slaves in a Round Robin mode all the time, and the temp-piconet can use the slots that are not used by the $master - S_1$ transmission or the $master - S_2$ transmission. We assume that there are always packets to send when the master polls a slave and when the slave replies the master.

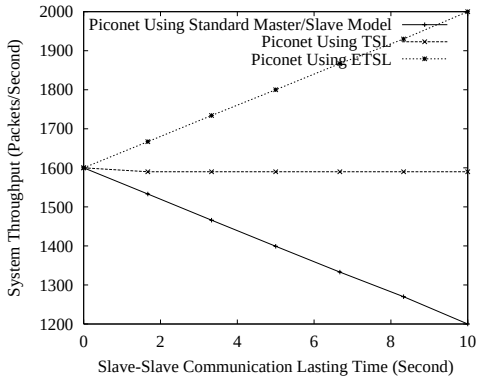

Fig. 4. A comparison of the system throughput (Total simulation time = $10\ seconds$)

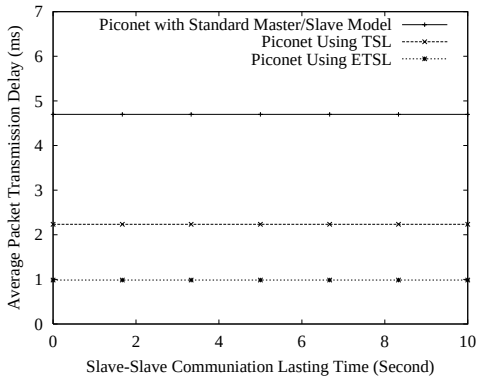

Fig. 5. A comparison of the average packet transmission delay (Total simulation time = $10\ seconds$)

Fig.4 shows the throughput of the piconets as a function of the slave-slave communication lasting time. As can be seen, the ETSL approach outperforms the TSL approach, which outperforms the standard master/slave model. The result can be explained as follows. When the standard master/slave model is used, for each packet transmitted between slaves, an extra slot is wasted to forward the packet. However, this kind of bandwidth waste can be avoided when the TSL approach is used, since the TSL approach changes a slave-slave transmission in the original piconet into a master-slave transmission

in a temp-piconet. Furthermore, the ETSL approach relies on extra channels temporarily to transmit slave-slave packets, and allows the original piconet to transmit more master-slave packets during the slots originally assigned to transmit slave-slave packets.

Fig.5 compares the average packet delay when these three approaches are used. As can be seen, the TSL approach and the ETSL approach can significantly reduce the average packet delay compared to the standard master/slave model, since they do not waste bandwidth to forward slave-slave packets. The ETSL approach outperforms the TSL approach since it allocates extra bandwidth to transmit the voice packets.

B.2 Performance Evaluation (with Slave-Slave File Transfer)

In this subsection, we evaluate the performance when slave-slave file transfer is supported. The simulation settings are similar to those described in the previous subsection except the follows. There is no CBR transmission between S_1 and S_2; instead, there are two files transferred from S_1 to S_2 and from S_2 to S_1 respectively, and the files are assumed to be transferred from the same time.

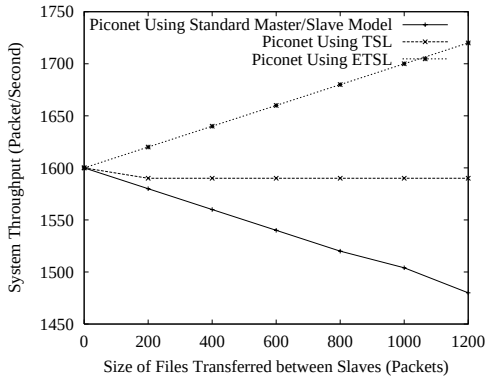

Fig. 6. Comparing the system throughput (The file transferred from S_1 to S_2 has the same size as the file transferred from S_2 to S_1.)

Fig. 6 and 7 show the throughput and of the piconets and the file transfer delay as functions of the aggregated size of the two files transferred between S_1 and S_2. The files transferred in different directions are assumed to have the same size. Due to the same reasons explained in the previous subsection, the ETSL outperforms the the TSL approach, and the later outperforms the standard master/slave model.

V. Conclusions

Ubiquitous computing has created a strong demand for a low-power, low-cost, and small size wireless communication technology. This demand has led to the development of the Bluetooth technology. Bluetooth uses a simple master/slave model to achieve media access control. Although the master/slave model has advantages such as simplicity and low-power, it also has some drawbacks since no direct link exists between any two slaves in a piconet. Consequently, slave-to-slave communications have to go through the master, and the master has to use extra bandwidth to forward the packets exchanged between slaves. In this paper, we proposed the TSL approach to address these drawbacks. With limited modifications to the master/slave model and without permanently changing the network topology, TSL can achieve good system performance. TSL can be further enhanced to allow each temporarily established piconet to have its dedicated channel, and improve the system performance further. Simulation results demonstrated that the proposed TSL and ETSL approach, can significantly improve the system throughput and reduce the packet transmission delay compared to the standard master/slave model.

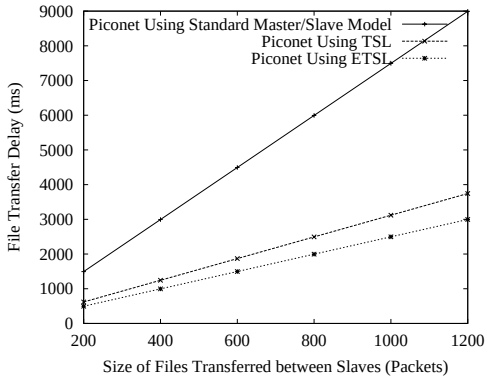

Fig. 7. A comparison of the file transfer delay

References

[1] Bluetooth Special Interest Group, "Specification of the Bluetooth System 1.0b, Volume 1: Core," *http://www.bluetooth.com*, Dec. 1999.

[2] J. Haartsen, "The Bluetooth Radio System," *IEEE Personal Communications*, vol. 7, no. 1, pp. 28–36, Feb. 2000.

[3] P. Johansson, N. Johansson, U. Korner, J. Elg, and G. Svennarp, "Short Range Radio Based Ad-hoc Networking: Performance and Properties," *Proc. of IEEE ICC'99*, pp. 1414–1420, 1999.

[4] J. Bray and C. Sturman, "Bluetooth: Connect Without Cables," *Prentice-Hall, Englewood Cliffs*, 2000.

[5] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks," *IEEE INFOCOM'01*, April 2001.

[6] P. Bhagwat and A. Segall, "A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets," *The Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99)*, 1999.

[7] B Miller and C. Bisdikian, "Bluetooth Revealed," *Prentice Hall*, 2000.

[8] W. Zhang, H. Zhu, and G. Cao, "On Improving the Performance of Bluetooth Networks Through Dynamic Role Management," *Technical Report, CSE-01-018 Pennsylvania State University*, also available at http://www.cse.psu.edu/∼gcao/paper/bluetooth.ps, May 2001.