Enhancing Performance of the Bluetooth Wireless Channel Through Dynamic Segmentation*

Carlos M. Cordeiro and Dharma P. Agrawal
Center for Distributed and Mobile Computing, ECECS
University of Cincinnati, Cincinnati, OH 45221-0030 - USA
{cordeicm, dpa}@ececs.uc.edu

Abstract

The emergence of Bluetooth as a default radio interface has allowed handheld electronic devices to be instantly interconnected as ad hoc networks. Recent studies show that since the Bluetooth standard operates in the unlicensed 2.45 GHz ISM (Industrial-Scientific-Medical) frequency band, the presence of multiple piconets in the vicinity creates interference on signal reception which, in turn, degrades the overall throughput of the network. This type of interference generated by Bluetooth piconets themselves is called intermittent interference, and it is a representative source of error in the Bluetooth channel. In order to cope up with this interference source, this paper proposes a sophisticated interference aware Bluetooth segmentation algorithm, where Bluetooth packet types are selected depending on the packet success probability, which is calculated based on the interference generated by multiple piconets. Such a segmentation algorithm is based on a precise analytical model, which is then compared with extensive simulation results. Among other things, we show that when the number of bridge nodes are larger than five the propagation delay between piconets increases rapidly. We have also observed a drastic overhead in the current Bluetooth piconet switching procedure and conclude that future enhancements to this procedure are crucial to efficiency of Bluetooth-based ad hoc networking.

Keywords: Wireless Communication, Mobile Computing, Bluetooth, Ad hoc Networks, Performance Evaluation.

1. Introduction

Bluetooth [1] is wireless communication technology that provides short-range, semi-autonomous radio network connections in the 2.4GHz ISM (Industrial-Scientific-Medical) band, and can establish ad-hoc networks, called piconets. This reduces any need for putting wires between personal devices such as computers, keyboards, printers, mobile phones, LANs, etc, within a small distance (up to 10m), hence offering added potential for a new range of applications. It was also chosen to serve as the baseline of the IEEE 802.15.1 standard for wireless personal area networks (WPANs) [17], which can support both synchronous traffic such as voice, and asynchronous data communications. As of this writing, the development of the final standard is in its final stages.

The Bluetooth standard defines different packet types to adjust to different application requirements. Those range from single unprotected 1-slot packet to FEC (Forward Error Correction) encoded 5-slot packets. Ultimately, the application is responsible for selecting the

* This work has been supported by the Ohio Doctoral Enhancement Funds and the National Science Foundation under grant No. CCR-0113361.

packet type which is best suited for its requirements, for instance, specifying its quality of service parameters. In case of a real-time application FEC encoded packets could be chosen, whereas in a non real-time application unprotected packets are the best option since they provide higher effective throughput. It is the Bluetooth adaptation layer responsible to receive messages from upper layers and segment them into small pieces of data to fit into a Bluetooth standard packet for best utilization. Ideally, the adaptation layer should choose the best suitable packet for transmission, both based on the application requirements and the wireless channel condition. Furthermore, as per the current Bluetooth specification, this choice cannot be static for the entire message due to dynamic nature of a wireless channel.

Recent studies [2, 13, 19] show that the presence of multiple piconets in the vicinity creates interference on signal reception. Based on an analytical and simulation study, a mathematical model for the packet success probability per Bluetooth slot has been derived [2], which takes into account factors such as the network load, propagation-loss law and shadowing, and showed that interference is a limiting factor in Bluetooth piconet throughput. Furthermore, with increased transceiver sensitivity, it is likely that the number of piconets that may interfere with each other will increase dramatically. This kind of interference, which has on nearby piconets as its main source, is hereby defined as *intermittent interference*. Such a definition is due to the frequency-hoped nature of the Bluetooth radio that generates interference in an intermittent fashion.

This work draws on the results of [2] to elaborate and propose an algorithm called IBLUES (Interference aware BLUEtooth Segmentation). IBLUES dynamically switches between Bluetooth packet types as packet success probability, due to interference, increases or decreases. This algorithm relies on the fact that packet success probability is inversely proportional to interference levels in a short-rage wireless technology such as Bluetooth [2]. Moreover, as it will be shown later, it is the interference that is taken into account in deciding the best suitable packet type. The rationale behind this algorithm is that a large packet outperforms a small packet in a low error rate channel (low interference level) since it possesses low overhead. On the other hand, small packets are best suitable when in a high error rate channel (high interference level) once they are less likely to be corrupted. In order to devise an accurate switching mechanism, IBLUES also takes into consideration interference from an environment consisting of multiple piconets. As far as the authors know, this is the first study that proposes a solution to mitigate and improve performance in face of interference.

The remainder of this paper is organized as follows. Section two provides an introduction to the Bluetooth technology highlighting its adaptation layer functionally. Next, section three elaborates on the IBLUES algorithm through the development of an intermittent interference model. Evaluation of the proposed algorithm through simulation is then given in section four. Finally, this paper is concluded in section five.

2. Bluetooth Overview

The entire Bluetooth protocol architecture is defined in [1]. A brief summary will be given here with special emphasis on its adaptation layer design issues. Bluetooth operates in a Master-Slave concept whereby a device that is selected by the master may transmit whereas others must wait for their turn. A message transmitted by any device is segmented into small packets. The basic unit of segmentation is a 625σsec long slot (see Figure 1), and, with a 1Mbps Bluetooth symbol rate, a slot can carry up to 625 bits. A TDD (Time Division Duplex) scheme is employed where a Master and at most seven Slaves alternatively transmit, with the Master transmitting in even numbered of slots and the Slaves in odd numbered slot, where

slots are numbered according to the master clock. In Bluetooth, the adaptation layer is located at the L2CAP (Logical Link Control and Adaptation Protocol), which, in turn, resides in the data link layer. As long as L2CAP is transmitting packets, it adopts a channel communication model representing a data flow among L2CAP remote devices. Such channels may be connection oriented or connectionless. The Bluetooth specification defines two distinct types of links for the support of voice and data applications, namely, SCO (*Synchronous connection-oriented*) and ACL (*Asynchronous connectionless*). The first link type supports point to point voice switched circuits while the latter supports symmetric as well as asymmetric data transmission. This work mainly considers the use of ACL links since the L2CAP specification has been defined only for this link type [1]. Besides, most data applications are expected to use this kind of link.

The ACL link allows the use of 1, 3, and 5-slot data packets with the optional use of FEC (Forward Error Correction). Figure 1 shows the 1, 3 and 5-slot packet transmission in Bluetooth. These ACL packet times are further distinguished whether they carry or not FEC coding. DMx (data medium-rate) packets provide a 2/3 FEC Hamming code and DHx (data high-rate) packets carry no FEC coding at all, where x = 1, 3, or 5, depending on the number of slots it occupies. On one hand, DMx packets trade data efficiency for a greater probability of successful transmission and can be useful in interference affected channels. On the other hand, for more efficient transmission, Bluetooth defines DHx packets. Among these, DH5 packets are the most data efficient ones since they carry the largest amount of data for the same amount of overhead (higher number of information bits per slot). Table 1 presents the possible ACL link packet types with their respective maximum user information, whether or not it carries FEC and the throughput achieved in both symmetric and asymmetric links.

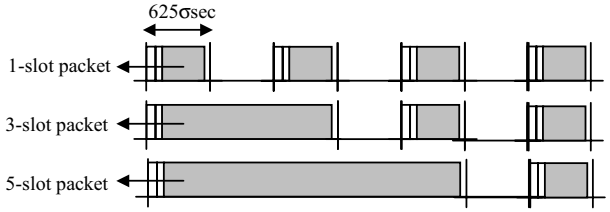

Figure 1 – Packet transmission in Bluetooth

Table 1. ACL Packet Overview

Type	User Payload (bytes)	FEC	Symmetric (Kbps)	Asymmetric (Kbps)	
DM1	0-17	Yes	108.0	108.8	108.8
DH1	0-27	No	172.8	172.8	172.8
DM3	0-121	Yes	256.0	384.0	54.4
DH3	0-183	No	384.0	576.0	86.4
DM5	0-224	Yes	286.7	477.8	36.3
DH5	0-339	No	432.6	721.0	57.6

In our prior work [2], we have proposed a model to determine the packet success probability between these packet types and the number of slot it occupies. Depending on the success probability of the wireless channel, different packets lead to different performance parameters. In a highly reliable channel (high success probability), we may start by using DH5 packets since they offer the best throughput for the least cost. As the success probability starts decreasing, DH3 packets might be more suitable once the packet error rate for DH5 packets increases faster than for DH3 packets. If the level of interference decreases, we may go back to using DH5 packets, whereas if it increases we have to consider using DH1 packets since they have a higher probability of success for higher error rates. In order to make this scheme feasible, we have to define thresholds where the adaptation layer would switch to

different packet types dynamically either according to the packet error rate or to the packet success probability. Section 3 elaborates on this issue, where we consider both the network load and the distance between the master and the slave of a piconet as parameters for our adaptation algorithm.

Two other schemes have been proposed to adjust packet types based on the channel condition. A mechanism is proposed [3] that chooses between FEC and non-FEC encoded packets. In this work, we consider not only the use of FEC, but a rather broader approach to switch between different slot packets according to error rate levels due to interference. A more general scheme introduced in [4] considers different packet types, with switching decisions based on only static parameters, and disregards interference influence on packet success probability [2]. Furthermore, the evaluation was conducted for a single piconet case only and multiple piconets environments were not considered. As we show later, our new algorithm takes into account interference effects on packet success probability and, therefore, provides an accurate and dynamic operation. The interference model we use takes into account environments where interference arises from multiple piconets operating in the vicinity. In accordance to that, the packet success probability is calculated based on the load and the distance between the transmitter (e.g., master of the piconet) and the receiver (e.g., slave), which has been shown [7] to critically affect packet error rates.

3. The IBLUES Algorithm

3.1 Intermittent Interference Model

In Bluetooth, packet transmissions depend on the master's clock and frequency hopping sequences regardless of the state of other piconets operating nearby. As we have mentioned earlier, our goal is to propose an interference aware algorithm, called IBLUES, which could provide an ideal switching mechanism between packet types as the channel conditions varies with time. The major goal of IBLUES is to optimize the number of slots used for the transmission of a given message, thereby efficiently utilize the scarce bandwidth available in the wireless channel. Large packets offer a high data rate but suffer from a higher error probability and may be inappropriate for use due to frequent retransmissions. On the other hand, small packets offer a lower data rate, however the success probability for these packets is better.

In [2], the Bluetooth wireless channel is modeled in terms of the intermittent interference effect on throughput. This model predicts the packet success probability per slot irrespective of FEC. In this work, the packet success probability has been found to be:

$$P_S(G, r_0) = \int_{-\infty}^{\infty} d\bullet \frac{e^{-\frac{\bullet^2}{2\omega^2}}}{\sqrt{2\phi}\omega} e^{-4GJ(\bullet, r_0)}, \qquad (1)$$

where

$$J(\bullet, r_0) = \int_{-\infty}^{\infty} dx \frac{e^{-\frac{x^2}{2\omega^2}}}{\sqrt{2\phi}\omega} \int_{r_0}^{\infty} \frac{2r dr}{1 + b^4 e^{\bullet - 4x \left(\frac{r}{r_0}\right)^{\gamma}}}. \qquad (2)$$

As we can see from (1), the Bluetooth packet success probability (P_S) depends on the interfering traffic load *G,* and on the distance between the transmitter and the receiver r_0,

hereafter referred simply as *r*. The other parameters mainly model multipath fading, shadowing and the effect of interference into P_S. In this context, • is a Gaussian random variable with zero mean and variance ω^2, ξ is the propagation loss exponent, and *b* is the system dependent capture threshold, where both the Gaussian (log-normal) attenuation variable • and the shadowing parameter ω are given in dB. For this work, we will consider the Bluetooth common values for these parameters [2], namely: ξ = 4, b = 6 dB, and ω = 0 dB. Moreover, we will also redefine P_S(G, r) as equal to P_S(X, G, r) where *X* is the Bluetooth packet type (see Table 1). As defined in [2], P_S(G, r) = P_S(DH1, G, r) which turns out to be our base case. This result of P_S(X, G, r) assists us in defining a precise switching mechanism between different packet types, according to the packet success probability. Hence, given a Bluetooth packet *X* we can define

$$efficiency(X) \mid \frac{payloadinbits(X)}{(slotperpacket(X) 2 1)\Delta slotsize} \qquad (3)$$

where *payloadinbits(X)* is the maximum amount of information in bits for packet type *X*, for *X* = DM1, DH1, and so on (see Table 1), (*slotsperpacket(X)* + 1), as the name suggests, is the number of slots occupied by *X* plus one for the acknowledgment, and *slotsize* is the size of a slot. In other words, *efficiency(X)* measures the number of user bits per slot in comparison to the size of the slot in bits for a given Bluetooth packet *X*. Table 2 lists the values, obtained through (3), for the efficiency of different Bluetooth packet types. Equations (1) and (3) enable us to define UB(X, G, r), the useful bandwidth for a given packet type *X*, as being equal to:

$$UB(X,G,r) \mid P_S(X,G,r)\Delta efficiency(X)\Delta nominalbandwidth \qquad (4)$$

Table 2. Bluetooth Packet Efficiencies

Packet Type	Efficiency
DM1	0.10
DH1	0.17
DM3	0.39
DH3	0.59
DM5	0.48
DH5	0.72

From (4), *nominalbandwidth* is roughly equal to 1Mbps for Bluetooth [1], and *efficiency(X)* is obtained from (3). Note that the values for *efficiency(X)* can be easily calculated, however we still need to obtain P_S(X, G, r) for all Bluetooth packets *X*. For example, if, for a given device *i*, $P_S(DH5, G_i, r_i) = 0.7$ and $P_S(DH3, G_i, r_i) = 0.8$, then we have that $UB(DH5, G_i, r_i) = 0.50$ Mbps and $UB(DH3, G_i, r_i) = 0.47$ Mbps considering a bandwidth of 1 Mbps. In this case, using DH5 packets is a better choice since they offer a higher useful bandwidth. In order to calculate P_S(X, G, r) for different packet types, let us define a refined view of it by expressing it in terms of the channel bit error rate (BER). We can define P_S(X, G, r) as:

$$P_S(X, G, r) = (1 - BER(X, G, r))^{payloadinbits(X)} \qquad (5)$$

where *payloadinbits(X)* is the payload field length in bits of packet *X*. If we consider a long period of time, we can assume that BER(X, G, r) is uniform for any Bluetooth packet *X* under the same load *G* and distance *r*. With this assumption, we can interpolate (5) to compute the

packet success probability for a different packet *Z* in terms of the BER resulted. Thus, for packet *Z* we have that:

$$\begin{aligned} P_S(Z, G, r) &= (1- BER(Z, G, r))^{payloadinbits(Z)} \\ &= (1- BER(X, G, r))^{payloadinbits(Z)} \\ &= (1- (1 - P_S(X, G, r)^{1/payloadinbits(X)})^{payloadinbits(Z)} \\ &= P_S(X, G, r)^{payloadinbits(Z)/payloadinbits(X)} \end{aligned} \quad (6)$$

Using (6), we can calculate the packet success probability of any Bluetooth packet type based on our base case $P_S(G, r) = P_S(DH1, G, r)$. This enables us to precisely define IBLUES switching algorithm. Simply stating, IBLUES dynamically evaluates $P_S(X, G, r)$, easily implemented in software, and then UB(X, G, r) through (6) and (4) respectively, and finally decides whether to switch between packet types as it reaches a threshold. Our task now is to find the adequate thresholds. One simple and efficient approach would be to define the thresholds where UB(X, G, r) = UB(Y, G, r), for X ∏Y. Figure 2 presents the UB curves for suitable values of G, having the distance r = 5m. From this figure we observe that under low interference level G (e.g., scatternet comprised of only two piconets), UB(DH5, G, r) has a higher value compared to all the other packets types and, thus, DH5 packets should be used. However, once G becomes higher than 0.26 and, as a consequence, the error probability increases, the algorithm should switch to DH3 packets once they offer better performance under such interfering traffic load. DM3 packets do offer higher useful bandwidth for *certain* load and distance values.

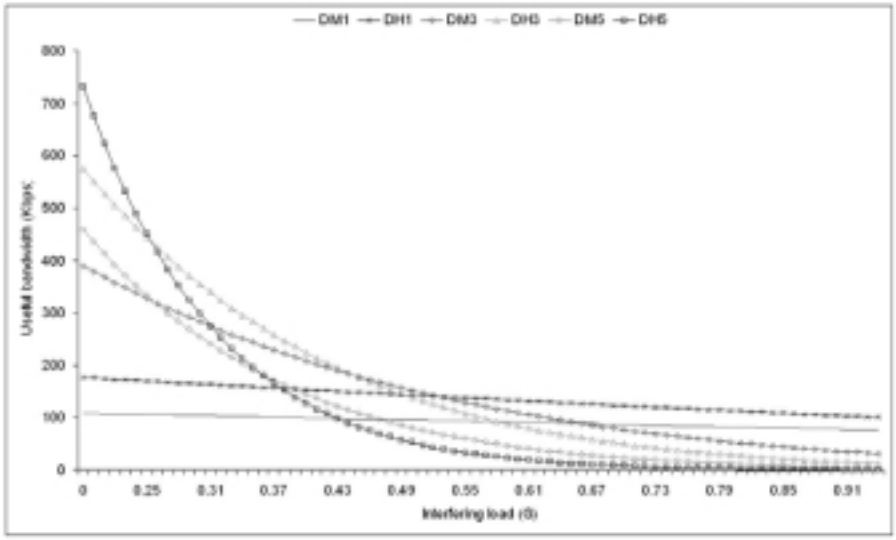

Figure 2 – Useful bandwidth for different packet types under increasing interfering load

Figure 3 depicts the state transition diagram built from the data of Figure 2 in terms of the packet success probability $P_S(X, G, r)$. As Figure 2 illustrates, packets DM5 and DM1 never offer higher useful bandwidth as compared to DHx packets, and that is why they are not included in the state transition diagram of Figure 3. On the other hand, DM3 packets showed to offer the highest useful bandwidth when the $P_S(DH3, 0.61, 5) < 0.80$. This result for DM3 packets is of importance since other studies [4, 5, 6], that do not take interference effects on performance into consideration, assume they do not offer higher throughput at any instant in time as compared to DHx packets. According to our analysis, this assertion is only true for DM1 and DM5 packets.

Similarly to equation (4), we can obtain the aggregate useful throughput of a network comprised of *N* piconets as:

$$AggUB(X,G,r) \mid UB(X,G,r)\Delta N\Delta\left\{14\frac{2\Delta(payloadinb\ its(X)2\ ackinbits\ (X))}{(slotsperpa\ cket(X)2\ 1)\Delta slotsize\ \Delta C}\right\}^{N41} \quad (7)$$

In equation (7), *ackinbits(X)* represents the size of the acknowledgment packet *X* in bits, and C is the number of available frequency channels which, in most countries, is equal to 79 [1]. In Bluetooth, a transmission of a DMx and DHx packet is followed by an acknowledgement from the recipient in the opposite direction. The acknowledgement information is included in the header of the return packet, so called piggy-backing. In our analysis, we consider the common case where acknowledgements do not carry payload information, therefore making *ackinbits(X)* in equation (7) always equal to 126 bits [1]. Figure 4 illustrates the aggregate useful bandwidth for an increasing number of piconets, under load G = 0.3 (30%) and average distance r = 5m. From Figure 4 we see that each packet type achieves its maximum aggregate useful bandwidth (MAUB) for a different number of piconets, what suggests and confirms our approach that different packets are best suitable under different scenarios. For instance, we can see that DH3 packets achieve its MAUB of approximately 8.17 Mbps when there are 61 piconets in the network. Thus, equation (7) is an important result that enables us to obtain the aggregate useful bandwidth under any combination of packet type, load and distance.

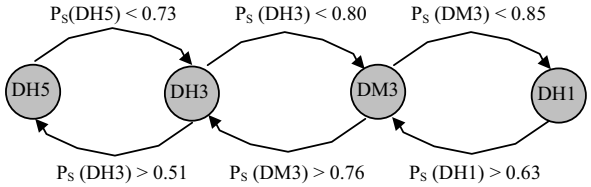

Figure 3 – State transition diagram for Figure 2 data

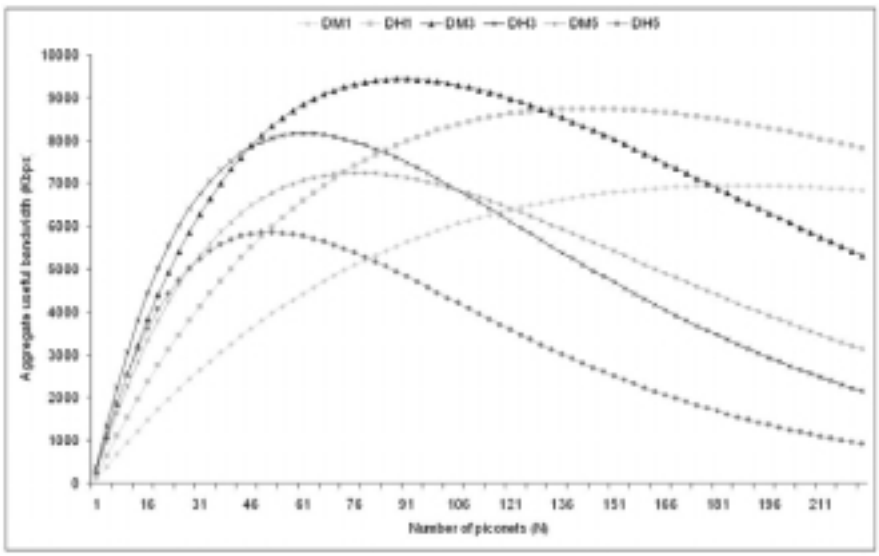

Figure 4 – Aggregate useful bandwidth under load of 30%
for an increasing number of piconets

3.1.1 Load and Distance Determination

We now have to address the issue of dynamically determining the interfering load *G* from neighboring piconets in a given scatternet, and the distance *r* between the receiver (e.g., slave) and transmitter (e.g., master), which are both used as input to the packet success probability $P_S(X, G, r)$. In IBLUES, the master of the piconet is responsible for acquiring the neighboring load and the distance information between itself and the selected slave, calculating the most suitable packet type according to our model, and sequentially propagating the resulting packet type to the slave as soon as it is polled. Two main questions arise here: first, what is the metric used by the master to determine the load and its distance from the polled slave; and second, how this information is relayed to the slave so that it can adhere to the adaptation algorithm requirements.

To measure the load, the most obvious choice is the queue length. Thus, the interfering load G in a piconet is computed by summing up the queue lengths from the devices of the piconets which are its neighbor, and have a common bridge node interconnecting them. IBLUES assumes that all devices have equal maximum queue lengths. In order to compute the load, we have created a new Bluetooth link manager packet called LOAD_UPDATE. This special packet is exchanged between piconet masters through bridge nodes. These LOAD_UPDATE packets have a logical two-hop lifetime, that is, one hop from the piconet master to the bridge node, and one last hop from the bridge node the respective master in the other piconet. To illustrate this better, consider the typical scatternet of Figure 5 composed of four piconets, where each piconet has several slaves (indicated by the letter $S_{i,j}$) and one master (indicated by the letter M_i). For example, in piconet 1, the master M_1 keeps track of its current piconet load by maintaining a weighted moving average of the past transmission loads for each of its slaves. At each inter load-update (ILU) time (measured in seconds), it schedules an event to send this load information encapsulated in a LOAD_UPDATE packet to its bridge nodes $S_{1,2}$ and $S_{1,3}$. This event is scheduled only if there are no pending events in the queue for $S_{1,2}$ and $S_{1,3}$, and is fired as soon as these slaves are polled. If this is the first transmission of a LOAD_UPDATE packet, meaning that M_1 does not yet know its bridge nodes, it simply broadcasts the packet. Upon receiving the packet, the bridge nodes $S_{1,2}$ and $S_{1,3}$ switch to piconets 2 and 4 respectively and convey the LOAD_UPDATE packet to the corresponding masters M_2 and M_4. As soon as M_2 and M_4 receive the LOAD_UPDATE packet, they first record the incoming address of the slaves as being bridge nodes addresses and next retrieve the information within the packet to update its own average interfering load G, sequentially discarding the LOAD_UPDATE packet. In other words, only direct neighbors of a piconet receive load information. Therefore, loops are prevented and LOAD_UPDATE packets do not roam forever, thus keeping a low overhead. Masters M_2 and M_4 will eventually schedule its own LOAD_UPDATE packet which when received by M_1 and M_3, will enable them to figure out who their bridge nodes are and update their load G. In case a master does not hear LOAD_UPDATE packets from its slaves, it assumes that no scatternet is formed and stops generating LOAD_UPDATE packets for a random time, currently of about 3 and 5 minutes. Last, but not least, an important remark has to be made about the reason behind opting for a static ILU time instead of some dynamic mechanism. As pointed out in [18], a device's queue length might change rapidly and, thus, the overhead introduced by some kind of scheme to dynamically adapt the ILU time is not worth it. Our design choice is to make IBLUES as simple as possible by adopting a static ILU parameter, which further helps in increasing a device's battery life.

All masters in a scatternet follow the same procedure and exchange LOAD_UPDATE packets periodically (at each ILU time), resulting in every master having the knowledge of the

load in its neighboring piconets. Although the load may change rapidly in a piconet before LOAD_UPDATE packets be exchanged [18], the results to be present soon show that maintaining a weighted moving average of past loads is a good enough estimate for a variety of applications. The next and final issue is the distance *r* between a master and its slave. For simplicity, we will assume that the slaves are uniformly distributed throughout the piconet. Once the common Bluetooth range is 10m, we will generate *r* according to a uniformly distributed random variable between 0.5m and 10m. Although this assumption introduces some error, it is generally acceptable [16], especially in (unpartitioned) office buildings. Nevertheless, more precise approach for determining *r* is a subject of ongoing research.

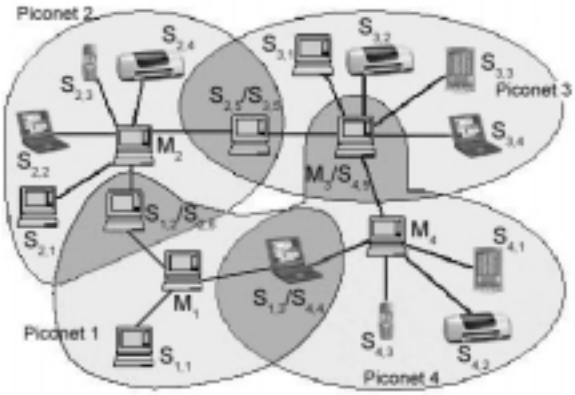

Figure 5 – Load information distribution within a scatternet

Next comes the question of how the master would dynamically forward to each polled slave the information about the packet type is to be used in its subsequent transmissions, as the network state is constantly changing. This is accomplished through the use of the same LOAD_UPDATE packet with a special flag in its payload field. This packet is transmitted as soon as the master polls the slave and indicates the type of packet the slave should use in its transmission in order to increase its transmission success probability. In case the packet success probability within the piconet increases/decreases rapidly, the master sends the slave another LOAD_UPDATE packet. In our simulations, this event never occurred once the time slice assigned for each device did not allow such scenario.

4. Simulation

To evaluate IBLUES, we have implemented it in the Network Simulator (ns-2) [10] and BlueHoc [11], an open-source Bluetooth simulator provided by IBM. Since BlueHoc only provides the basic functionality of Bluetooth, we have made considerable extensions to this simulator in order to conduct our experiments.

4.1 Bluetooth Scatternet Environment Simulation

We have carried out an initial experiment by employing an environment comprised of only Bluetooth devices connected in the form of a scatternet, that is, without external sources of interference. In order to show the viability of IBLUES, our first evaluation computes the overhead in the network associated with the exchange of LOAD_UPDATE messages by piconet masters. As we have shown earlier, this overhead is intrinsically related to the ILU time of transmission of LOAD_UPDATE messages, as well as the number of bridge nodes a piconet possesses once a LOAD_UPDATE is sent to every bridge node. We have used the

scatternet topology of Figure 6 for this evaluation. In this topology, within a total area of 50m x 50m, we have defined a core piconet, piconet 1, with eight devices where we collect measurement data. According to the Bluetooth specification, all devices of a piconet can function as bridge nodes, but in this simulation the master of the piconet does not assume this role since the BlueHoc simulator does not allow masters to work as bridges. Therefore, we have executed our simulations for up to 7 (seven) bridge nodes in piconet 1 and with different values for the ILU time. Each simulation lasted for 300 seconds and we used exponential sources as traffic generators. For each and every slave (eventually bridge) node in piconet 1, there is a corresponding piconet in the scatternet of Figure 6. Figure 7 shows the relative overhead of LOAD_UPDATE packets for ILU = 1, 5, 10, and 15 seconds, and for different number of bridge nodes.

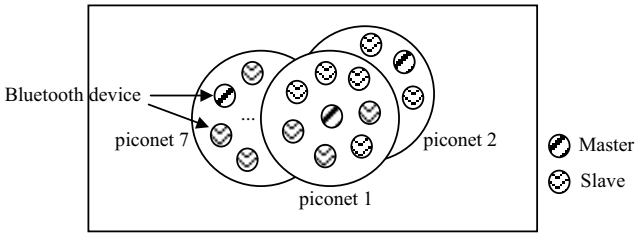

Figure 6 – Scatternet topology for LOAD_UPDATE overhead evaluation

As we can see from Figure 7, ILU = 1 second generates the largest overhead of up to 24% of the total traffic within the scatternet when there are a total of 7 (seven) devices in the piconet assuming the role of a bridge node. This is unacceptable for a wireless technology such as Bluetooth with maximum total bandwidth of roughly 1 Mbps. Furthermore, the energy consumption involved in the process of these large number of messages can be excessively high. On the other hand, ILU = 1 second provides IBLUES with the ideal situation where it has the optimal view of the network load since masters are continuously exchanging load information. The overhead is drastically reduced when we employ ILU = 5, 10 and 15 seconds. More specifically, regardless of the number of bridge nodes, it decreases by more than 5 (five) times, and remains below 5%. Among the values of ILU = 5, 10, and 15, there is still a significant difference in overhead, but such differences are negligible as compared to ILU = 1. Furthermore, as it will be shown shortly, our simulations indicate that the performance drop between ILU = 1 and ILU = 5 is not significant, however it becomes excessive for ILU = 10 and ILU = 15.

The propagation delay analysis reveals a potential drawback in the Bluetooth design. From Figure 8(a) we can see a slightly higher delay when ILU = 1 second as compared to the other values of ILU since the queues are filled up with both control and data packets. However, for other values of ILU, there is not much difference between these curves. As Figure 8(a) also shows, the average propagation delay of packets between piconet masters remains steady when the number of bridge nodes is less than or equal to 5. However, when the number of bridge nodes is higher than 5 the delay increases rapidly. In all cases, the time taken to execute the Bluetooth switching procedure for a slave to leave momentarily one piconet and join its neighboring piconet [1] contributes most to the total delay as depicted in Figure 8(b). More specifically, the switching procedure consumes 67, 69, 67, and 65 percent of the total delay for ILU = 1, 5, 10, and 15, respectively, and is excessive when the number of bridge nodes is higher than 5. Future improvements in the Bluetooth specification [1] are necessary to optimize this procedure in case Bluetooth is to run well-known ad hoc routing protocols such as AODV [14] and DSR [15], and real time applications which possess strict

quality of service requirements. Nevertheless, as we shall see later, the simulation results of IBLUES proved it to effectively improve the overall piconet performance even when of this overhead is present.

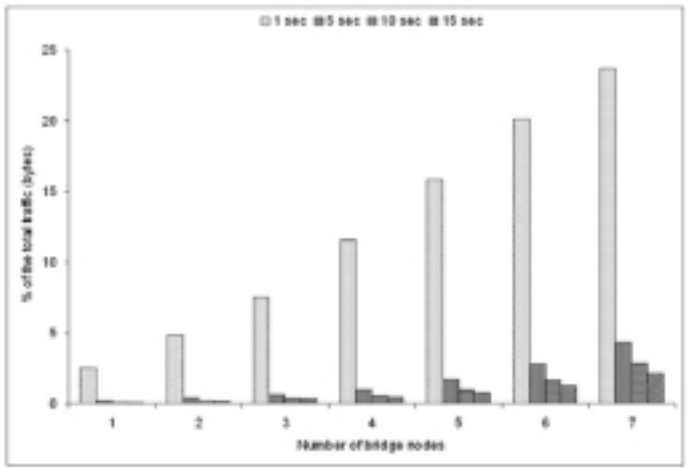

Figure 7 – LOAD_UPDATE overhead

As per the previous analysis, we will consider only 1 and 5 seconds as values for the ILU time. The choice of ILU = 1 second is because it is useful for benchmarking purposes, and ILU = 5 seconds because it greatly reduces the overhead and does not perform much inferior than ILU = 1. Moreover, the experiments to be conducted all assume less than 6 slave devices working as bridge nodes, since this is a likely scenario of a typical office environment [9].

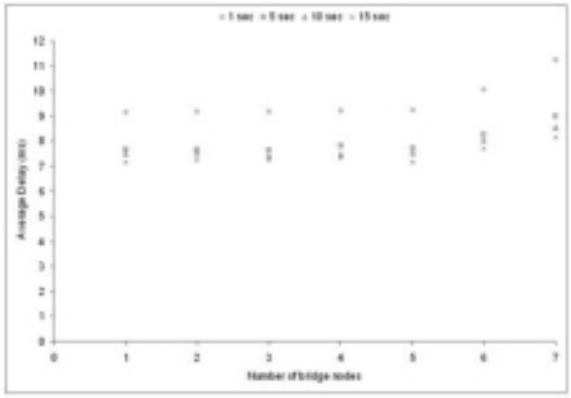

Figure 7(a) – Master-to-Master propagation delay

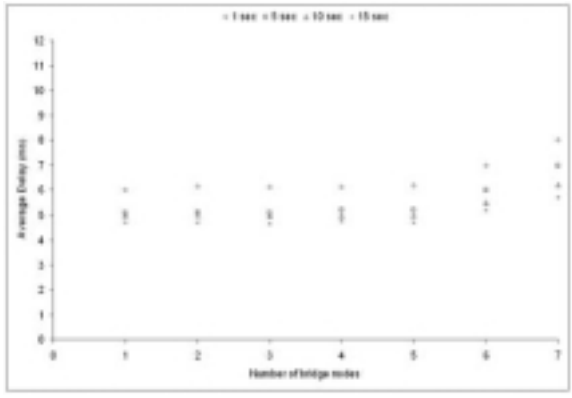

Figure 7(b) – Bluetooth switching procedure delay

To demonstrate the performance of both configurations of the ILU time, we perform our study with the evaluation of IBLUES under these two scenarios, namely ILU = 1 and 5 seconds. We have considered a network topology similar to the one depicted in Figure 6, but with two important differences. First, for this study we have considered a scatternet composed of 30 piconets distributed in an area of 200m x 200m, reflecting a typical office environment [9]. Each piconet possesses 4 (four) devices and exactly two devices of each piconet are connected to some other piconet [9]. Note that part of this non-trivial topology has been built with the aid of a graphical interface available with the BlueHoc package that, even if it does not support scatternet construction, at least enables the definition and code generation of basic piconet code. The second major change is in the application layer. Each master of a piconet establishes a CBR (Constant Bit Rate) connection with each of its slaves with a MTU

(Maximum Transfer Unit) of 500 bytes, and the transmission lasts for the entire length of 300 seconds. A total of five iterations are executed per experiment and is averaged to give the final results.

Figures 9 and 10 show the results for both ILU = 1 and ILU = 5 seconds respectively, for increasing interfering network load. It is instructive to compare such results with the one obtained analytically in Figure 2 and notice that they have a similar behavior, thereby validate each other. When ILU = 1 second, piconet masters have precise network load information and thus can make the right switching decision at the right time. From this figure we can see that IBLUES always seeks to attain the highest possible useful bandwidth by switching between packet types as the load increases. In particular, we have highlighted some specific spots in IBLUES curve that are observed to be nearly equal to the actual measured throughput for each packet type. We found that the almost negligible difference between the measured IBLUES throughput curve and the corresponding curves for the Bluetooth packet types are due to the random variable we use to generate the distance between the transmitter and the receiver. As Figure 8 shows, the propagation delay between piconet masters does not have a major impact in our configuration of two bridge nodes. On the one hand, although the distance has some effect on the throughput, when ILU = 1 the master continuously receives LOAD_UPDATE packets that rapidly converges IBLUES to the optimum packet. On the other hand, from the highlighted spots in Figure 10, we can see that the distance has a larger impact on the measured throughput of IBLUES when ILU = 5 seconds. Nevertheless, its performance is still nearly close to the optimal case. From the simulation results we see that ILU = 5 seconds provided the ideal time for scheduling LOAD_UPDATE messages. In other words, when IBLUES attempts to schedule a new LOAD_UPDATE message, the previous one has already left the output queue shortly before the new arrival. The low overhead generated by the ILU = 5 seconds and the nearly optimal performance encouraged us to select it as the default value for the ILU parameter in the final IBLUES implementation. In addition, our simulations for ILU = 5 seconds offers the best trade-off between efficiency and limited overhead.

Finally, we performed another simulation to obtain the aggregate useful bandwidth under an offered load of 30%, 4 devices per piconet, and a total of 115 piconets. Figure 11 illustrates the collected aggregate useful bandwidth, contrasting it with the one obtained through the analytical model. As we can see, the aggregate useful throughput increases to a certain limit of piconets, at which time it begins to decrease. IBLUES switching mechanism seeks to guarantee that at any instant in time, and for any configuration of interfering load, distance, and number of piconets, the Bluetooth piconet is always operating at its maximum useful bandwidth, and the scatternet is always operating at its maximum aggregate useful bandwidth.

5. Conclusions and Future Work

Bluetooth devices themselves are likely to be the significant interferers to the Bluetooth technology in the very near future. This problem must be addressed by the Bluetooth technology which operates in the ISM unlicensed band. This paper proposes a dynamic segmentation algorithm called IBLUES for the Bluetooth technology, with piconet masters having approximate load information from the whole scatternet. This offered load is encapsulated into a new LOAD_UPDATE packet defined in the LMP (Link Manager Protocol) layer of the Bluetooth protocol stack. Next, the load is used as input to a mathematical model which indicates the packet type for the highest useful bandwidth. This algorithm is further explored to show very satisfactory under increasing load, and under a variety of applications. Moreover, we observed an enormous overhead in the current

Bluetooth piconet switching procedure and future improvements are crucial to enable efficient ad hoc networking.

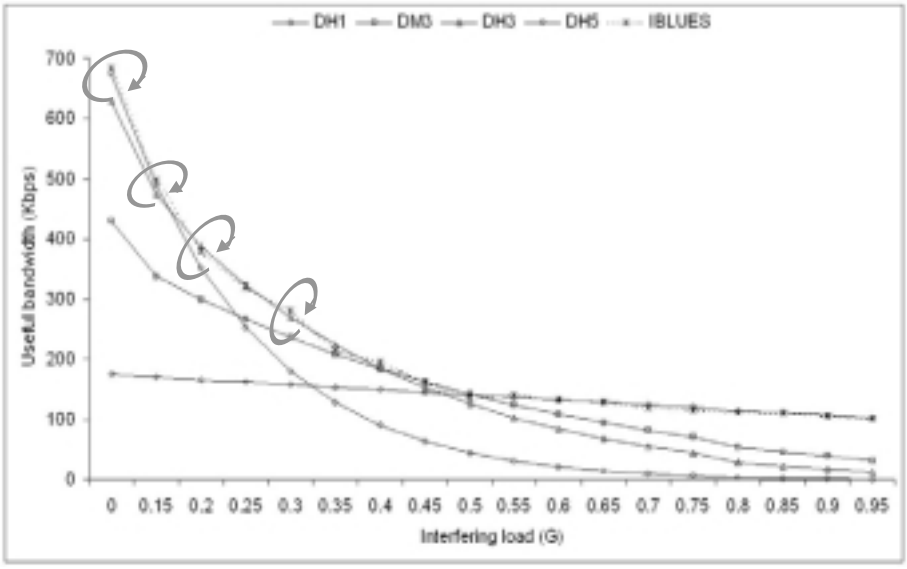

Figure 9 – Simulation results of the useful bandwidth for different packet types under increasing interfering load for ILU = 1 second

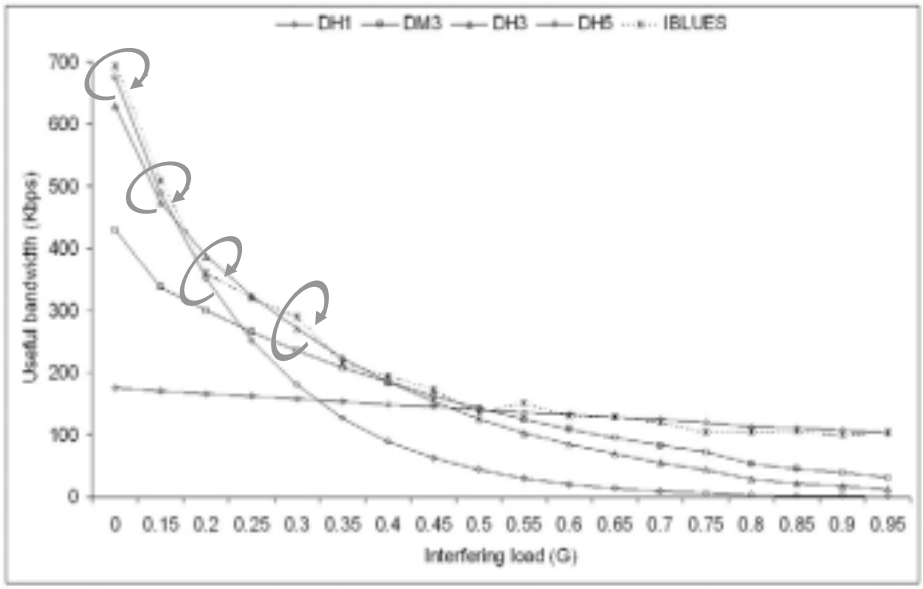

Figure 10 – Simulation results of the useful bandwidth for different packet types under increasing interfering load for ILU = 5 seconds

As our future work, we plan to devise a better way of estimating the distance between the transmitter and the receiver Bluetooth devices. Moreover, we need to adapt our model to consider burst errors, as well as study ways to reduce overheads and consider quality of service parameters. We also intend to extend IBLUES to take into consideration interference generated by IEEE 802.11 devices, since they operate in the same unlicensed ISM frequency band as Bluetooth, as well as study the impact of mobility of Bluetooth devices on signal reception, both at the perspective of Bluetooth itself and IEEE 802.11.

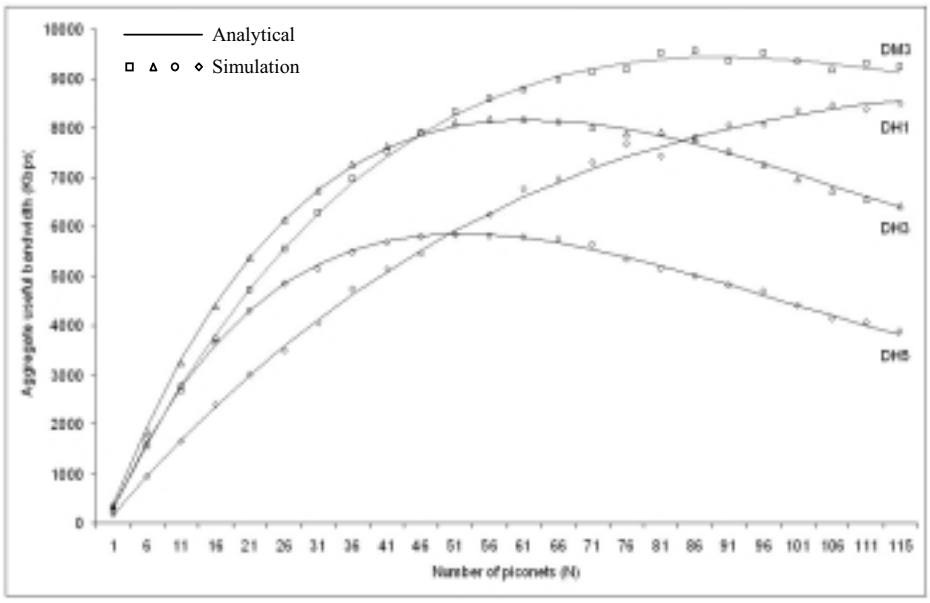

Figure 11 – Simulation result of the aggregate useful bandwidth

References

[1] Bluetooth SIG, "Bluetooth Specification", http://www.bluetooth.com.
[2] C. Cordeiro, D. Sadok and D. Agrawal, "Piconet Interference Modeling and Performance Evaluation of Bluetooth MAC Protocol", *in Proceedings of IEEE GLOBECOM*, San Antonio, USA, 2001.
[3] A. Das, A. Ghose, V. Gupta, A. Razdan, H. Saran, and R. Shorey, "An adaptive link-level error recovery mechanisms in Bluetooth", *in Proceedings of IEEE International Conference on Personal Wireless Communication*, pp. 85-89, 2000.
[4] J. Kim, Y. Lim, Y. Kim, and J. Ma, "An adaptive segmentation scheme for the Bluetooth-based wireless channel", *in Proceedings of IEEE IC3N*, pp. 440-445, 2001.
[5] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey, "Enhancing the performance of asynchronous data traffic over bluetooth wireless ad hoc networks", in *Proceedings of the IEEE INFOCOM*, vol. 1, pp. 591-600, 2001.
[6] S. Zurbes, W. Stahl, K. Matheus, and J. Haarsten, "Radio network performance of bluetooth", *in Proceedings of the IEEE ICC*, pp. 1563-1567, 2000.
[7] D. Duchamp and N. Reynold, "Measured performance of a wireless LAN", 17th Conference on Local Computer Networks, Meneapolis, September 1992.
[8] N. Golmie and F. Mouveaux, "Interference in the 2.4 GHz ISM band: Impact on the Bluetooth access control performance", in *Proceedings of the IEEE ICC'01*, 2001.
[9] M. Fainberg, and D. Goodman, "Analysis of the Interference Between IEEE 802.11b and Bluetooth Systems", *in the IEEE VTC Fall 2001*, October 2001.
[10] The Network Sumulator (ns-2), *http://www.isi.edu/nsnam/ns/*.
[11] BlueHoc, IBM Bluetooth Simulator, *http://oss.software.ibm.com/developerworks/opensource/bluehoc/*.
[12] N. Golmie, R.E. Dyck, and A. Soltanian, "Bluetooth and 802.11b interference: simulation model and system result", IEEE 802.15/195r0, 2001.
[13] Y. Kim, B. Zhen, and K. Jang, "The hybrid of Listen-Before-Talk and Adaptive Frequency Hopping for coexistence of Bluetooth and IEEE 802.11 WLAN", Samsung Advanced Institute of Technology Report, 2001.

[14] C. E. Perkins and E. M. Royer, “Ad-hoc on demand distance vector routing," in IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), February 1999.
[15] D. Johnson, and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks," in Mobile Computing (T. Imielinski and H. Korth, eds.), Kluwere Academic Publishers, 1994.
[16] T. Rappaport, S. Seidel, K. Takamizawa, “Statistical channel impulse response models for factory and open plan building radio communicate system design”, *IEEE Trans. on Comm.*, Vol. 39, Number 5, May 1991, pp. 794-807.
[17] C. Bisdikian, “An Overview of the Bluetooth Wireless Technology“, IEEE Communications Magazine, December 2001, pp.: 86-94.
[18] A. Capone, M. Gerla, R. Kapoor, “Efficient Polling Schemes for Bluetooth Piconets”, IEEE ICC’01, Helsinki, Finland, June 2001.
[19] Y. Lim, J. Kim, S. Min, and J. Ma, "Performance evaluation of the Bluetooth-based public Internet access point", Proceedings of 15th International Conference on Information Networking, 2001, pp.: 643-648.