QoS Based Scheduling for Incorporating Variable Rate Coded Voice in BLUETOOTH

Shuchi Chawla
shuchi@cs.cmu.edu

Huzur Saran
saran@cse.iitd.ernet.in

Mitali Singh
mitalisi@usc.edu

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi, India

Abstract— **Bluetooth is an emerging standard for low-power, low-cost indoor pico-cellular wireless systems. It is a Master driven Time Division Duplex (TDD) system. Real time services such as voice are given 64Kbps bandwidth in Bluetooth. However most other wireless networks use compressed voice, which requires much lesser bandwidth, leading to a substantial increase in system capacity. Bandwidth can be further conserved by using Voice Activity Detection (VAD) techniques and variable rate voice codecs In this paper we propose and analyse modifications to be made to Bluetooth for incorporating Variable Rate Coded Voice. Current mechanisms in Bluetooth use synchronous channels with fixed slot allocation for voice and a best effort service for data. We propose and study two scheduling strategies which optimise bandwidth consumption by using variable rate coded voice. In the first scheme, Adaptive T_{SCO} Scheduling, we modify the conventional scheduling policy to change the time period of scheduling a voice channel depending upon its activity. In the Voice over ACL Scheduling, we schedule voice asynchronously like data using a QoS based scheduling scheme with maximum scheduling delay tolerable by packetsas the QoS parameter. This scheme can also be used to schedule other multimedia applications with varying QoS requirements. We observe from simulations that the Voice over ACL scheme gives more than 115% increase in bandwidth over the currently used scheduling in the presence of two voice connections.**

I. Introduction

Bluetooth is an upcoming standard for indoor wireless enabling simple, spontaneous wireless connectivity to a wide range of devices. The Bluetooth specification [1], [2], [3] is for a 1 Mbps, low-cost radio solution that can provide links between portable handheld devices alongwith connectivity to the internet. Owing to features such as robustness, low cost, longer battery life and seamless roaming, it is expected to be used widely for indoor wireless.

Bluetooth supports both voice and data traffic. The two kinds of traffic are treated differently. Voice is given a guaranteed service while data is given a best effort service. In the current Bluetooth specifications, toll quality speech (with the same quality as speech on a telephone line) is supported and 64Kbps bandwidth is alloted for each voice channel.

Many wireless systems on the other hand use compressed voice, which occupies less than half the bandwidth needed for toll quality speech. These use algorithms based on a high temporal correlation in speech [9]. Additionally, bandwidth can be further saved by detecting silence periods in speech (known as *Voice Activity Detection 'VAD'*) and reducing the coding rate during silence.

Voice connections are generally two-way with separate bandwidth being allocated for each direction. During a call, atleast one of the two speakers is silent most of the times and sometimes, both the speakers are silent. As a result more than half the time, bandwidth allocated to the call for one direction remains unused, as during silence, bandwidth requirements are lesser, because, only background noise needs to be transmit. The period of activity is known as 'talk-spurt'. We can exploit this feature of voice connections by using a variable resource allocation scheme for voice. The bandwidth conserved in this manner can be used for other voice connections or data transfer.

Based on this idea, many voice codecs have been designed using Voice Activity Detection (VAD) algorithms [9] and code speech at two or more different rates. For example, Qualcomm's Q4401 [11] can code speech at rates of 4Kbps, 4.8 Kbps, 8Kbps or 9.6Kbps. Another codec designed by Kari Jrvinen et al [10] codes speech at two rates, 11.4Kbps and 22.8Kbps. Both these codecs change the rate of coding every 20 msecs. Accordingly, a wireless standard can also choose to transmit voice at two or more distinct rates depending on activity of the call.

Incorporation of variable rate coded voice over Bluetooth would necessitate a substantial change to the scheduling policies used in Bluetooth which treat voice and data traffic differently and intrinsically allocate 64Kbps bandwidth to voice. In this paper, we propose and analyse two alternative scheduling strategies to incorporate variable rate coded voice in Bluetooth.

Bluetooth is a Time Division Duplex (TDD) system. Voice channels are scheduled over slots reserved a priori for each voice connection. In order to decrease the amount of bandwidth allocated to voice in this existing framework, we could decrease the frequency of slot reservation, or in other words, increase the time period between consecutive voice slots. This forms the basis of our first scheduling strategy, namely, Adaptive T_{SCO} scheduling.

Alternatively, we can revamp the entire framework of reserving slots for voice. In this case, slots can be allocated to voice on a need basis, that is depending on whether there are voice packets in the transmission queue or not. Here we have to make sure that voice is given priority over data and that scheduling delays are within the required QoS for voice. This idea forms the basis of our Voice over ACL scheme. To ensure that QoS for voice is met, we use a sophisticated EDD based scheduling scheme with maximum tolerable scheduling delay as a QoS measure for voice.

In order to test the performance of our scheduling schemes, we implemented them using the Bluetooth extension to Network Simulator [12]. Since Bluetooth is an indoor wireless standard with maximum 10m radio range [1], it is reasonable to expect not more than three voice calls in a single cell. Hence, we simulated our schemes in the presence of upto 3 voice connections in one cell of Bluetooth.

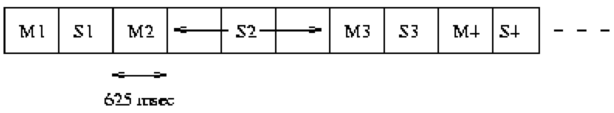
This paper is organized as follows. In Section 2 we study the Bluetooth medium access protocol and some features such as the synchronous and asynchronous services offered. In Section 3 we discuss a few scheduling strategies studied previously for Bluetooth and other TDMA systems. In Section 4 we propose two variations of QoS scheduling for incorporating variable rate coded voice in Bluetooth. These are Adaptive T_{SCO} and Voice over ACL. We also discuss the issues involved in implementation of each. Section 5 describes the simulation scenario and criteria for comparison of various schemes, followed by the simulation results. Finally in Section 6, we present the conclusions.

II. Medium Access in Bluetooth

Bluetooth is a master-slave based pico-cellular system. It uses a Master driven TDD for medium access control. Several Bluetooth devices sharing the same channel form a *Piconet*. One Bluetooth unit acts as the master of the piconet, whereas the other units act as slaves. Upto seven slaves can be active in a single piconet. In addition, more slaves can remain locked to the master in a *parked* state.

The channel is divided into time slots each 625μs in length [1], [2], [3]. The master and slaves transmit packets alternatively. (ref Figure 1) Packets can be of lengths 1, 3 or 5 slots. The master starts transmission only in even-numbered slots, and in the following slot, only the slave which has been addressed in the previous slot, can send the next packet. Medium Access control lies with the master.

Fig. 1. TDD in Bluetooth



Bluetooth supports both voice and data traffic [1], [2], [3]. Depending on the kind of data, two types of links are supported between any two members of the piconet forming a Master-Slave pair. These are the (i) *Synchronous Connection Oriented (SCO)* links, and the (ii) *Asynchronous Connection Less (ACL)* links. The *SCO* link is typically used for a voice connection which occupies fixed slots that are assigned a priori by the Master [1]. The *ACL* link is intended for asynchronous data traffic and occupies the non-voice slots.

The Bluetooth protocol stack is different for the *ACL* and *SCO* links. For an *ACL* connection, data from the network layer is first passed to the Logical Link Control and Adaptation Protocol (L2CAP) layer. This layer performs the segmentation and reassembly operation. *SCO* links on the other hand go around this layer, directly to the physical or Baseband layer.

III. Related Work

SCO connections in Bluetooth occupy fixed slots that are assigned a priori by the master. For the remaining slots, which are to be allotted to ACL connections, the master follows a scheduling policy. Current Bluetooth implementations follow a Round Robin approach for MAC scheduling for ACL connections, in which slaves sending data are addressed sequentially. This clearly leads to bandwidth wastage when one or more slaves do not have data to transmit.

Several improvements to this scheme, exploiting the symmetry of the wireless link in Bluetooth, have been studied by Manish Kalia et al. One such proposal [4] is to prioritize bandwidth allocation to a channel based on the presence of a packet at both the master and slave queues. Another scheduling scheme, the HOL Priority policy [5], proposed by them is based on comparison of lengths of the Head of Line packet at each master and slave transmission queue. These solve the problem of bandwidth usage in ACL links to quite a large extent. However, they do not modify SCO connections at all and as before extra bandwidth used for voice still continues to be wasted. Moreover they do not deal with the issue of QoS for individual connections, and hence, cannot be extended to incorporate voice over ACL links.

In other similar TDMA systems such as GSM, the incorporation of variable rate coded voice has been studied in detail. J Y Jeng et al [6] have proposed several approaches for dealing with services having variable bandwidth requirement during the course of a connection. One such proposal is to allocate more than one slot per time frame whenever the bandwidth requirement of a connection increases. In another proposal by P Lin et al [7], they define two kinds of service for connections - half rate and full rate, and study various schemes of scheduling them to achieve maximum possible bandwidth usage. Since Bluetooth does not have a clearly defined time frame, these schemes cannot be used directly in Bluetooth.

We will now present our scheduling schemes, which are suited specifically to Bluetooth.

IV. Scheduling for Variable Rate Coded Voice

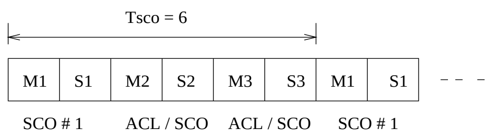
Keeping in view the fact that for variable rate coded voice the bandwidth required is much lower than the bandwidth offered by the *SCO* links we can follow two approaches for conserving bandwidth. The first is to modify the structure of SCO links to decrease the bandwidth occupied by them, while the second is to do away with SCO links and schedule both voice and data over ACL links.

We assume that a variable rate voice codec connected to any Bluetooth device will output at two distinct rates, corresponding to talkspurt and silence. In order to communicate the state of the call (i.e. silence or talkspurt) to the MAC or Baseband layer, the application layer can use a signaling packet periodically. This information needs to be transmit once every 20msec (as most codecs [10], [11] change rate of coding once every 20 msecs). This introduces a minor overhead of 4.16% which is very less compared to the improvement in bandwidth utilisation due to the two schemes discussed below.

A. Adaptive T_{SCO}

The Synchronous Connection Oriented *SCO* links, which carry voice, receive a guaranteed service. Slots are allocated to *SCO* links periodically. The time period of slot allocation is known as T_{SCO} (ref figure 2). For voice channels with no error coding, T_{SCO} is typically 6 TDD slots to account for 64Kbps bandwidth.

Fig. 2. Scheduling of an SCO connection



As mentioned earlier, the coding rate of variable rate coders is much less than 64Kbps even during talkspurt. In order to adapt to this coding rate, the time period of scheduling SCO connections can be increased. For instance, $T_{SCO} = 16$ would correspond to a bandwidth usage of 24Kbps, which is approximately the bandwidth allocated to voice in GSM during talkspurt. Similarly, T_{SCO} can be increased further during silence, to adapt to an even lower coding rate.

In the Adaptive T_{SCO} scheme, we dynamically adjust T_{SCO} depending on the activity of the call. Accordingly, T_{SCO} toggles between two distinct values with change in activity of voice. As discussed above, $T_{SCO} = 16$ during talkspurt and 32 during silence would correspond to the bandwidth allocated to variable rate coded voice in GSM [10]. In our simulations, however, we use the values 6 and 12 slots for T_{SCO}, in order to make minimal change to the standard, which currently specifies $T_{SCO} = 6$. These values reflect the half-rate full-rate scheduling scheme used by P Lin et al [7] for GSM.

B. Voice over ACL

In this scheduling policy, instead of providing a guaranteed service to voice, we schedule voice over a best effort service link i.e. the *ACL* link. We need to ensure however that the QoS for voice is still met. Hence, we use a QoS based scheduling scheme, which is described below.

Latency based scheduling

In this scheduling scheme, we define QoS requirements of a connection in the form of the maximum scheduling delay each packet of the connection can tolerate, which is termed as the *latency* of the connection. Accordingly, if a connection has latency of ‘n’ slots, then each packet of that connection should be scheduled within 'n' slots of its arrival. Based on this we define a 'deadline' for each packet as the number of the slot by which it should be scheduled keeping in mind its latency requirements.

To schedule packets, we use a greedy *Earliest Due Deadline (EDD)* scheme. We schedule that connection, for which, its HOL packet has the earliest deadline. EDD is known to give an optimal schedule in the sense that if it is possible to satisfy QoS requirements of every connection, they are satisfied. However, if all connections have a low latency, it may not be possible to satisfy the requirements of all connections. Thus, this defines the system capacity and necessitates admission control for the system.

In order to use Latency based scheduling for voice, a suitable value for latency of voice needs to be determined. For a voice connection, end-to-end latency of less than 100 msec is acceptable as it is not noticeable to the human ear [8]. Keeping in mind that the connection may go through many hops, latency over a single hop should be much lesser than 100 msec. In order to avoid large end to end delay and jitter, we try to avoid queueing of more than one voice packet at any transmission queue in the system. Accordingly, we choose maximum tolerable scheduling delay of a voice connection to be the time gap between arrival of two packets. Thus, the latency of voice for a 22.8Kbps connection should be 18 time slots. This corresponds to a latency of approximately 11 msec. Similarly, for a 11.4Kbps connection, latency should be 36 time slots.

As stated before the Bluetooth Protocol Stack differs for SCO and ACL connections. When we schedule voice over ACL, we have to take care that voice packets are not modified at the L2CAP layer, they should bypass this layer. For this purpose, voice packets are still passed down the protocol stack as SCO packets. Only at the MAC layer during scheduling, they are scheduled as ACL packets.

V. Relative Performance of the Scheduling Schemes

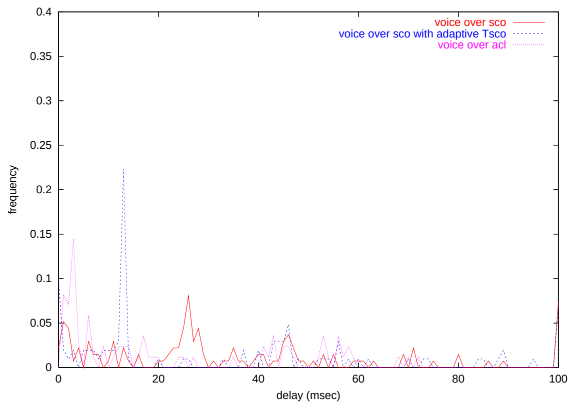
A. Simulation Scenario

We implemented our algorithms using the Bluetooth extension to 'ns' (Network Simulator) [12]. Simulations were performed for a piconet having seven slaves and one master. For the sake of uniformity and fair comparison, we used the Latency based QoS scheme (ref Section 4.2) for scheduling ACL connections in all our simulations.

We used a two state Markov chain model to simulate talkspurt and silence periods for voice. Voice traffic was generated in bursts of 20 msecs length. Coding rates for silence and talkspurt were chosen in accordance with the rates typical for voice over GSM [10], i.e. 22.8Kbps and 11.4Kbps respectively.

Latency for voice connections was taken to be 18 slots during activity and 36 during silence, as explained in the previous section. Latency for data connections varied between 50 and 200 slots. We did not study admission control in the system at this stage, hence such high values of latency were chosen for data in order to be able to meet QoS requirements of all connections.

Fig. 3. Scheduling delays for data in presence of 1 voice connection



For Adaptive T_{SCO} scheduling, the values of T_{SCO} during activity & silence were chosen to be 6 and 12 as discussed in Section 4.1.

In order to compare the different scheduling schemes as above, we measured the average data throughput of the system, which is reflective of the fraction of timeslots being used for data connections while satisfying QoS of voice connections. We also measured the scheduling delay suffered by data packets in each scheme.

B. Simulation Results

From the simulations we observed that both our scheduling schemes perform better than the currently used SCO scheduling with fixed T_{SCO}. Figure 3 displays scheduling delays for data packets for the three schemes in the presence of one voice connection. It can be seen that scheduling delays for data packets in the case of adaptive T_{SCO} are lesser on average than those in the case of fixed T_{SCO}. However the improve-

Fig. 4. Scheduling delays as a fraction of latency in the presence of 1 voice connection

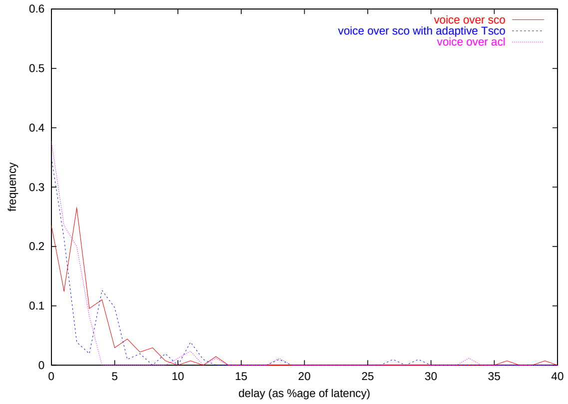
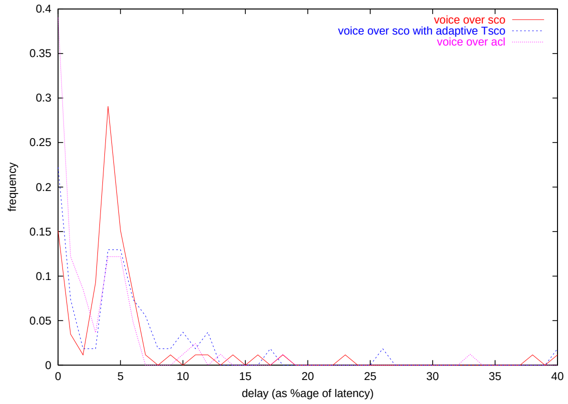


Fig. 5. Scheduling delays for data in presence of 2 voice connections



ment is more marked in the case of scheduling voice over ACL links.

Since we use different values of latency for different connections, comparison of (delay of a packet)/(Latency of the connection) is more informative than simply the delay.

The delay/latency graph (ref Figure 4) for the same set of data as above shows similar results as Figure 3. Even in the presence of two voice connections (ref Figure 5), both our scheduling schemes perform better than voice over SCO and voice over ACL performs best.

The trend is similar in the case of the data throughput achieved. We observe that introducing a voice connection in the system leads to a drop of almost 33.3% in data throughput (Table I). This improves marginally by 7.6% if the adaptive T_{SCO} scheme is used. However the improvement is substantial, almost 15%, if voice is scheduled over ACL links. When there are two voice connections in the systems, the proposed

TABLE I
DATA THROUGHPUT & PACKET DELAYS IN BLUETOOTH WITH ONE VOICE CONNECTION

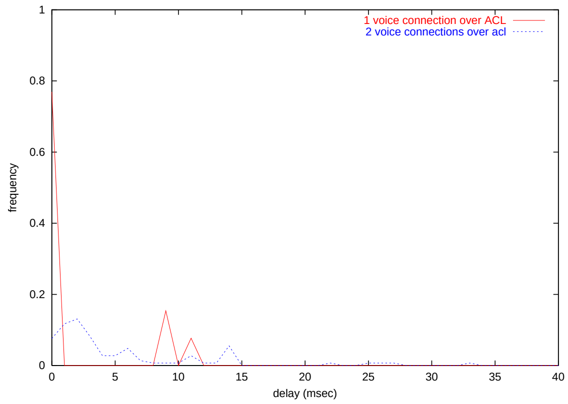
Scheme	Data Throughput (Kbps)	Average Delay(msec)
Without voice	358.4	9.98
Voice over SCO	238.93	21.72
Adaptive T_{SCO}	257.18	18.09
Voice over ACL	274.24	11.78

TABLE II
DATA THROUGHPUT & PACKET DELAYS IN BLUETOOTH WITH TWO VOICE CONNECTIONS

Scheme	Data Throughput (Kbps)	Average Delay(msec)
Without voice	358.4	9.98
Voice over SCO	119.4	36.47
Adaptive T_{SCO}	172.2	31.05
Voice over ACL	258.8	15.25

algorithms work even better. The adaptive T_{SCO} scheme performs better than traditional scheduling by almost 44.2% (Table II), while scheduling over ACL links leads to a substantial improvement of 116.7%.

Fig. 6. Scheduling delays for voice packets in Voice over ACL scheduling



Simulating 3 voice connections gives interesting results. As is already known, currently used Voice over SCO scheme does not allow any data throughput in the presence of 3 voice connections as SCO links in this case hog the entire bandwidth. Using voice over SCO with adaptive T_{SCO} on the other hand, leaves some slots free for data and an average data throughput of 94.2Kbps is achieved (Table III). Similarly voice over ACL scheme also gives a high throughput

TABLE III
DATA THROUGHPUT IN BLUETOOTH WITH THREE VOICE CONNECTIONS

Scheme	Data Throughput (Kbps)
Voice over SCO	0.0
Adaptive T_{SCO}	94.2
Voice over ACL	102.6

of 102.6Kbps.

We also observe that scheduling delays for voice packets in Voice over ACL scheduling remain within 20 msec (ref Figure 6), which is an acceptable limit. Hence scheduling voice over ACL links performs the best among all the schemes we have studied.

VI. Conclusions

We find that incorporation of variable rate coded voice in Bluetooth can lead to a large improvement in bandwidth utilisation. A substantial increase in data throughput and corresponding reduction in packet scheduling delay is achieved if we use SCO links with adaptive T_{SCO} rather than standard SCO links in Bluetooth. A much larger improvement can be achieved by using voice over ACL links at the cost of added complexity.

Improvements to our work can be made by using a better model of voice instead of the Markov chain model used by us. However, we believe that this will not make a significant difference to the nature of our results.

References

[1] Specification of the Bluetooth System. Volume 1, v1.05, December 1999.
[2] http://www.bluetooth.net
[3] http://www.cellular.co.za/bluetooth.htm
[4] Manish Kalia, Deepak Bansal, Rajeev Shorey, *Scheduling policies for Bluetooth MAC*, IEEE VTC-Spring'2000, Japan, May, 2000.
[5] Manish Kalia, Deepak Bansal, Rajeev Shorey, *MAC Scheduling and SAR policies for BlueTooth: A master Driven TDD Pico-Cellular Wireless System*, IEEE Momuc'99, San Diego, California, November, 1999.
[6] J Y Jeng, C W Lin, Y B Lin, *Dynamic Scheduling for GSM Data Services*, IEICE Transactions on Communication, Vol.E80-B No.2 p.296.
[7] Phone Lin, Yi-Bing Lin, *Channel assignment for GSM Half-Rate and Full-Rate traffic*, Computer Communication, NCTU 1997.
[8] S. Keshav, *An Engineering Approach to Computer Networking*. Addison-Wesley, 1999.
[9] Speech Coding Tutorial. http://wwwdsp.ucd.ie/speech_tut.htm
[10] Kari Jrvinen et al, *GSM Enhanced Full Rate Speech Codec*, ICASSP'97, Munich, Germany, April 1997.
[11] Variable Rate Vocoders. http://www.qualcomm.com/ProdTech/asic/vlsi/q4401.html
[12] UCB/LBNL/VINT Network Simulator NS. http://www-mash.CS.Berkeley.EDU/ns/