Interference In Bluetooth And Methods To Avoid It

Aravind Kilaru
University of Texas at Arlington
Department Of Computer Science & Engineering
akilaru@cse.uta.edu

Abstract

Bluetooth is a wireless technology that enables devices to communicate in the 2.4 GHz industrial, scientific, and medical (ISM) license free frequency band. Bluetooth avoid the interference by using a frequency hopping (FH) spread spectrum technology with hop rate of 1600 hops per second. This paper studies schemes to avoid local interference observed by the devices. It tries to come up with a logical solution for the problem in discussion.

Keywords: Channel, Frequency Look-ahead Unit, Frequency Selection Unit, Link State History, Piconet, Scatternet.

1. Introduction

Ever since the first personal computers were furnished with a printer, a cable was needed not just for the electrical power to it but also for the data to be printed from the computer. When modems became popular there had to be another cable to connect the modem to the serial port on the computer. The list goes on and on. Every time a computer has to be moved from one room to another, or just to another table, there is a lot of struggle with all the cables.

However, with new technologies an idea of how to make things better by removing cables and replace them with wireless communication has grown from just ideas to reality. Bluetooth is one of those ideas that are developing into a solution by reality.

The technology is based on a low-cost short-range radio link, built into a 9x9 mm microchip, facilitating protected ad hoc connections for stationary and mobile communication environments. It allows for the replacement of the many propriety cables that connect one device to another with one universal short-range radio link. Bluetooth radio technology provides a universal connection to existing data networks and a mechanism to form small private ad hoc groupings of connected devices away from fixed network infrastructures.

Bluetooth is a wireless technology that enables any electrical device to wirelessly communicate in the 2.4 GHz industrial, scientific, and medical (ISM) license free frequency band at a link range of 10 meters. This range can be increased up to 100 meters by improving the transmission power and receiving sensitivity. It has been specifically designed as a low cost, low power, radio technology, which is particularly suited to the short range Personal Area Network (PAN) application. And this distinguishes it from the IEEE 802.11 wireless LAN technology.

2. Overview of Bluetooth [2,10,11]

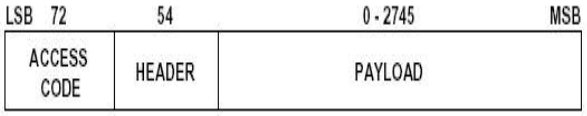
The Bluetooth system provides a point-to-point connection (only two Bluetooth units involved), or a point-to-multipoint connection. In the point-to-multipoint connection, the channel is shared among several Bluetooth units. Two or more units sharing the same channel form a *piconet*. One Bluetooth unit acts as the master of the piconet, whereas the other unit acts as slave.

The interference in Bluetooth is avoided by using a frequency hopping (FH) spread

spectrum technology. The Bluetooth specification defines a high hop rate of 1600 hops per second instead of just a few hops per second used in other implementations. The frequency band is divided into a number of hop channels with every channel being just a fraction of the total frequency band. In Bluetooth every channel is used for 625 µs (one slot) followed by a hop in a pseudo-random order to another channel for another 625 µs transmission repeated constantly. That way the Bluetooth traffic is spread over the entire ISM band and a very good interference protection is achieved. If one of the transmissions is jammed, the probability of interference on the next hop channel is very low. The master uses even numbered slots while odd numbered slots are reserved for slave transmissions. Furthermore, error correction algorithms are used to correct the fault caused by jammed transmissions. The 79 hop carriers have been defined for the Bluetooth wireless technology except for Japan, France and Spain where 23 hop carriers have been defined, because the ISM-band is narrower there. When Bluetooth units are communicating, one unit acts as master and the rest act as slaves. The master's unit system clock and the master identity are the central parts in the frequency hopping technology. The hop channel is determined by the hop sequence and by the phase in this sequence. The identity of the master determines the sequence and the master unit's system clock determines the phase. In the slave unit, an offset may be added to its system clock to create a copy of the master's clock. In this way every unit in the Bluetooth connection holds synchronized clocks and the master identity, that uniquely identifies the connection.

2.1 Packet Format

The Bluetooth packets have a fixed format (Figure 1). A 72-bit access code comes first in the packet. The access code is based on the master's identity and the master's system clock; for example, it provides the means for synchronization. This code is unique for the channel and used by all packets transmitting on a specific channel. The 54-bit header following the access code contains error correction, retransmission and flow control information. The error correction information can be used for correcting faults in the payload and in the header itself. Finally, the payload field can be up to 2,745 bits.



LSB – Least Significant Bit, MSB – Most Significant Bit

Figure 1: The Bluetooth Packet Format [10]

2.2 Piconet and Scatternet

Any two Bluetooth devices that come within range of each other can set up an ad-hoc connection, which is called a piconet. Every piconet consists of up to eight units. There is always a master unit in a piconet and the rest of the units act as slaves. The unit that establishes the piconet becomes the master unit. The master unit can change later but there can never be more than one master. Several piconets can exist in the same area. This is called scatternet. Within one scatternet all units share the same frequency range, but each piconet uses different hop sequences and transmits on different 1 MHz hop channels. All piconets share the 80 MHz band, thus, as long as the piconets pick different hop frequencies, no sharing of hop channels occurs.

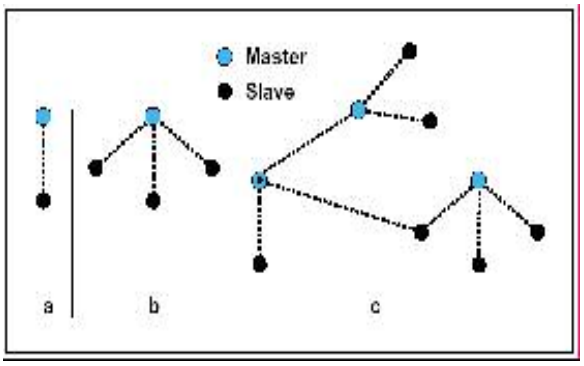


Figure 2: Piconets (a) single slave, (b) multi-slave and (c) scatternet [10]

2.3 Link Types

The Bluetooth specification defines two link types, Asynchronous Connectionless (ACL) and Synchronous Connection Oriented (SCO). Different master-slave pairs in the same piconet can use different link types. The link type may be changed during a session.

The SCO links are primarily used for voice traffic and their data rate is 64 kbps. ACL links are used mainly for data traffic and support broadcast messages (i.e. from the master to all slaves to the piconet). Multi-slot packets use the ACL link type and can reach the maximum data rate of 721 kbps in one direction and 57.6 kbps in the other direction if no error correction is used.

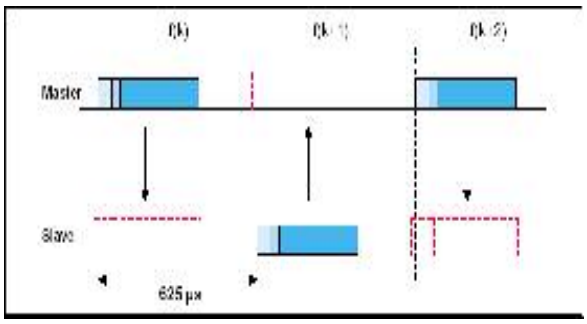


Figure 3: TDD and timing, Single Slot Packet [10]

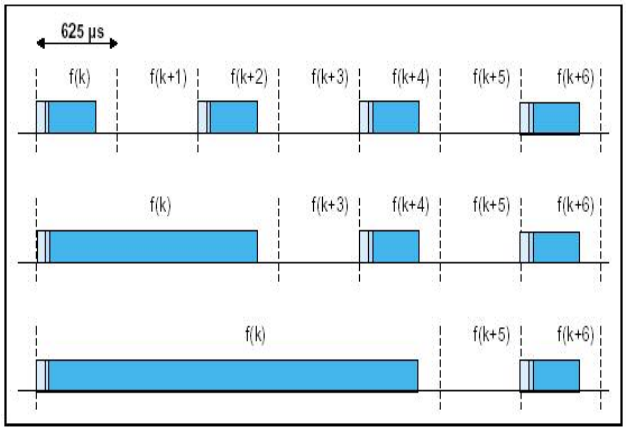


Figure 4: Multi-slot packets [10]

2.4 Security

Authentication is used in Bluetooth to prevent unwanted access to data and to prevent falsifying the message originator. Encryption is used to prevent eavesdropping. These two techniques combined with the frequency hopping technique and the limited transmission range for a Bluetooth unit (usually 10 m) gives the technology higher protection against eavesdropping. Three modes of security are defined in Bluetooth. The need for a particular mode is dependent on what kind of application is executed.

• Non-secure mode – No authentication or encryption is used.

• Service-level security mode –This security mode provides the possibility to define trust levels for the services and units used respectively.

• Link-level security mode –This mode is based on the use of link keys that are secret 128-bit random numbers stored individually for each pair of devices in a Bluetooth connection. Each time two Bluetooth units communicate; the link key is used for authentication and encryption.

3. Schemes to Avoid Interference:

Frequency Hopping and Adaptive Frequency Hopping provide good immunity against randomly occurring interference signals of short duration. However, in many applications persistent errors of much larger duration are the likely source of problems. Such errors are caused by electromagnetic emissions from other co-located devices like copy machines, printers, microwave ovens, baby monitors etc. operating in the same band as the communicating devices. These devices stay operational for durations ranging from 10-20 minutes to longer like 2-3 hours and are expected to cause errors due to interference for the entire duration. This interference degrades the throughput of the system considerably even when FHSS (Frequency Hopping Spread Spectrum) techniques are employed. Apart from this a link is susceptible to randomly occurring wireless errors due to interference, fading etc. These errors are bursty by nature due to which retransmissions also get corrupted and the overall channel utilization falls considerably [3]. There are many source of interference in any typical application scenarios. It could be cordless phones and devices compliant with Home-RF, Wireless LAN IEEE 802.11 standards; these operate in the same band. Water molecules resonate at microwave frequencies around 2.4GHz; this results in absorption of the ISM- band radiation by the water carrying entities in the vicinity of Bluetooth units reducing the signal strength [4].

To avoid the above flaw scheduling algorithms and error avoidance mechanisms have been proposed. These mechanisms take in to account the specific requirements of the Bluetooth and provide an efficient way overcome these flaws.

At any scheduling instant, the Master should communicate with only that subset of its Slaves that are in the good state. Preferably, the Master should stop transmission on a channel as soon as its state goes bad and resume transmission as soon it comes back to the good state. A problem arises because the connectivity of the devices changes unpredictably with time and both the ‘bad’ and ‘good’ periods last for intervals of random duration. The task is to devise a scheme to identify the switch from the good to the bad state at the earliest and also to figure out that the channel has become good at the end of the duration of the bad period. Such a scheme can then be used by the Master to pick the subset of good connectivity devices at each scheduling instant. If such a scheme recognizes the bad period later than the actual time it starts, then channel utilization will fall, and if it recognizes the start of a good period too late, then the corresponding Slave device will get starved of data even during the good period. So the accuracy of the scheme is crucial in maintaining a good level of channel utilization.

3.1 The Link State History (LSH) Scheme: [3]

In this approach the interference is avoided by using the history of errors seen on the channel by previous packet exchanges. The Link State History (LSH) is used to predict the state of the wireless channel between the Master and each of its Slaves at every scheduling instant. This information is used to choose whether or not to transmit to a particular Slave on a particular frequency.

The LSH scheme is used by the Master to identify devices having a high probability of seeing bad connectivity and avoids communication with them while they are in that state. With the right tuning of thresholds, the scheme can achieves good accuracy in recognizing both the beginning and the duration of the bad period. This scheme has been discussed for a Master driven frequency hopping system derived from the Bluetooth specification.

The Link State History (LSH) table (figure 5) has one counter corresponding to each channel (Slave-frequency pair) on which the Master can potentially transmit. The counters record recent history of errors experienced on a particular channel. Two thresholds $T_{TRANSMIT}$ and T_{RESET} are defined for the counter values. $T_{TRANSMIT}$ represent the threshold value at which the transmission is allowed if the counter is less than that. T_{RESET} represent the threshold value at which the counter is reset.

The Frequency Look-ahead Unit (FLU) is a unit to look-ahead during any arbitrary future time slot and gives the frequency that would be used and it is maintained with the Master. The Frequency Look-ahead Unit **(**FLU) will obtain the frequencies to be used for transmission from either the same or another Frequency Selection Unit (FSU) in the Master unit by providing to the FSU, clock bits corresponding to that particular time slot.

	FREQ1	FREQ2	FREQ3	...	FREQ78	FREQ79
SLAVE1	C(1,1)	C(1,2)	C(1,3)	...	C(1,78)	C(1,79)
SLAVE2	C(2,1)	C(2,2)	C(2,3)	...	C(2,78)	C(2,79)
SLAVE3	C(3,1)	C(3,2)	C(3,3)	...	C(3,78)	C(3,79)
SLAVE4	C(4,1)	C(4,2)	C(4,3)	...	C(4,78)	C(4,79)
SLAVE5	C(5,1)	C(5,2)	C(5,3)	...	C(5,78)	C(5,79)
SLAVE6	C(6,1)	C(6,2)	C(6,3)	...	C(6,78)	C(6,79)
SLAVE7	C(7,1)	C(7,2)	C(7,3)	...	C(7,78)	C(7,79)

C(i,j) = Counter for Link State of SLAVE_i for FREQ_j

Figure 5: The LSH Table

3.1.1 Updateting and Maintaining LSH table:

LSH is updated based on packets received by the Master. The LSH counters make a note of the most recently seen number of consecutive NAKs (negative-acknowledgements) received by the Master on the corresponding channel. The explanation of how this table is maintained is given below.

Suppose the Master is communicating with Slave '*i*'. Let f_j be the frequency of transmission by the Master and f_k be the frequency on which slave replies. If the counter C(i, j) is below $T_{TRANSMIT}$ the Master allows transmission or reception to take place from Slave-'*i*' on frequency f_j and changes counter value depending on the success or failure of the transmission. If the transmission was not successful, as indicated by the receipt of a NAK or a garbled packet, the corresponding counter is incremented by 1. If the transmission was indeed successful, as indicated by an ACK, the counter is reset to zero.

When the Master receives no packet from the Slave it assumes that the Slave did not reply because it never received the transmission due to channel *i-j* being bad, so it increments C(i, j) by 1. If the Master receives a garbled packet from the Slave, it indicates that channel *i-k* was bad. Then, the Master assumes that the packet was carrying a NAK and therefore, increments both C(i, j) and C(i, k).

When the Master receives NAK from the slave, it indicates that the transmission on channel *i-j* was not received correctly, but the channel *i-k* on which the Slave sent the NAK is good. Therefore C(i, j) is incremented, and C(i, k) is reset. The receipt of an ACK from the Slave sent on frequency f_k reports successful receipt of the data sent by the Master on frequency f_j , and indicates that the channels, *i-j* and *i-k* are good. Therefore both C(i, j) and C(i, k) are reset to zero. If the counter value is equal to or above $T_{TRANSMIT}$ then the Master does not allow transmission or reception from Slave-'*i*' on frequency f_j or f_k. In this case the Master simply increments C(i, j) by 1. Each

time the Master increments the counter, it then checks if the counter value now equals T_{RESET}, and if it does, the Master resets the counter to zero.

3.1.2 Use of LSH table for selecting a Slave:

The LSH-scheme requires that before transmission, the Master consult the LSH table to determine whether the channel (i, j) is in good state or not.

Let us consider the case when the Master is communicating with Slave-'*i*' on frequency f_j. If the LSH counter C(i, j) is below a threshold, $T_{TRANSMIT}$, then the Master can go ahead with the transmission. Contrary if the counter value is greater than or equal to $T_{TRANSMIT}$, then the Master is required to not transmit to that Slave and find some other Slave, to which it can transmit at the same time and same frequency, and not wasting the current slot on the channel. The Master checks the LSH table for the next Slave, and so on, till it finds a Slave, for which the corresponding counter for f_j is below $T_{TRANSMIT}$. In the event that it can find no Slave with a counter below $T_{TRANSMIT}$, it chooses the Slave whose LSH counter has the highest value. The loss of service by the intended Slave and gain of service by another Slave can be recorded to address the issue of service fairness as per any suitable policy.

For wireless systems in which communication happens in slot pairs, transmission is considered unsuccessful if even one of the packets sent on either of the 'to' or 'from' slots gets corrupted. For such systems, the Master is required to use the FLU to obtain the frequency, say f_k, on which the Slave will send its reply, and check both the channels, *i-j* and *i-k*, at each scheduling instant, and carry on with the transmission only if both the counters, C(i, j) and C(i, k), are below $T_{TRANSMIT}$. The Master does not ever change the hopping sequence of the wireless system, and hence this scheme can be used for systems with a fixed hopping sequence.

3.1.3 Packet size determination:

The Master has to decide which packet size to transmit depending on the good or bad connectivity with the Slave on different channels. The slot on which the Master transmit is slot x say. If the Master sends a 5-slot packet, the Slave will respond on the $(x+5)^{th}$ slot, if it sends a 3-slot packet, the Slave will respond on the $(x+3)^{th}$ slot and if it sends a 1-slot packet, the Slave will respond on the $(x+1)^{th}$ slot. The frequencies corresponding to these slots are f_{k_5} , f_{k_3} and f_{k_1} say. The Master had already got these frequencies by using the FLU. The Master is required to first check counter C(i,k_5). If C(i,k_5)< $T_{TRANSMIT}$ then the Master sends a 5-slot packet to Slave-'*i*', else the Master checks counter C(i,k_3). If C(i, k_3)< $T_{TRANSMIT}$, then the Master sends a 3-slot packet to Slave-'*i*', else the Master checks counter C(i,k_1). If C(i,k_1) $\geq$ $T_{TRANSMIT}$ then the Master tries to find another Slave in a similar fashion. If all the counters for all the seven Slaves indicate bad channels, the Master defaults to the original Slave-'*i*' and a 1-slot packet.

3.2 Link State History (LSH) and Goodness Counter Link State History (GC–LCH) Scheme: [1]

This scheme is similar to the previous scheme the only difference is that it uses another table called Goodness Counter Link state history (GC - LCH). In this method the slave monitors the wireless channel using the Goodness Counters.

In this every active slave is required to monitor the master - slave transmission. Each slave "*i*" has to maintain the Goodness Counter GC(i, j) for all the 79 hopping frequencies. The GC(i, j) is incremented every time a slaves "*i*" receives a valid packet successfully from the master transmission on frequency f_j. But, if the

slave detects an error in the receiving packet header, the goodness counter is left unaltered, because some slave to continue transmission of a 3-slot or a 5-slot packet may use even slot. The value of this counter is sent to Master unit where it forms the i^{th} row of the history table that master unit maintains.

The slave send this short term link state history information stored in Goodness Counters to the master at regular intervals and then resets their counter to zero to monitor the link state for next period of communication. The number of bits in these counters is small in order to minimize the overhead of transmitting this information to the master device. Thus the GC(i, j) is allowed to count up to a maximum value and then stay at that value until it is reset for the next monitoring. The value of the Goodness Counter received by the master indicates the relative goodness of the link between the master and slave "*i*" on the frequency f_j; higher the count better the link.

The Master unit stores the link state information received from the slave, in tabular forms as the "Goodness Counter Link State History" (GC–LCH) table. The i^{th} row of this table holds the counters GC(i, j) sent by the slave "i" to Master. A MIN_GOODNESS threshold is defined for the Goodness Counters. The Master uses the GC–LCH table in the similar fashion as described in the previous scheme of LCH done from the master side.

3.2.1 Packet size determination:

The Master has to decide which packet size to transmit depending on the good/bad connectivity with the Slave on different channels. The slot on which the Master transmit is slot x say. If the Master sends a 5-slot packet, the Slave will respond on the $(x+5)^{th}$ slot, if it sends a 3-slot packet, the Slave will respond on the $(x+3)^{th}$ slot and if it sends a 1-slot packet, the Slave will respond on the $(x+1)^{th}$ slot. The frequencies corresponding to these slots are f_{k_5} , f_{k_3} and f_{k_1} say. The Master had already got these frequencies by using the FLU. The Master is required to first check counter GC(i,k_5). If GC(i,k_5) > MIN_GOODNESS then the Master sends a 5-slot packet to Slave-'*i*', else the Master checks counter GC(i,k_3). If GC(i,k_3) > MIN_GOODNESS, then the Master sends a 3-slot packet to Slave-'*i*', else the Master checks counter GC(i,k_1). If GC(i,k_1) < MIN_GOODNESS then the Master tries to find another Slave and packet-size in a similar fashion. If all the counters for all the seven Slaves indicate bad channels, the Master defaults to the original Slave-'*i*' and a 1-slot packet.

3.3 Bluetooth Interference Aware Scheduling (BIAS): [4]

The main objective of BIAS is to alleviate the impact of interference while maintaining fairness and supporting different Quality of Service (QoS).

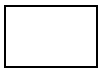
Suppose traffic from slave S_i to the master is data rate, r_i, equal to l_i or p_i where l_i is the packet length in slots (1, 3 or 5 slots depending on the packet type), and p_i is the poll interval in (master/slave) slot pairs.

There are three main procedures in this scheduling, (i) *estimate channel()* is used to detect the presence of interference in the frequency band. (ii) *compute weights()* computes the weights of channel for a particular device (to give a particular channel more weight for a device having less number of channels than others). In other words it gives the "right of the way" or priority access to certain devices. (iii) *compute credits()* computes the credit used to control and allocates the bandwidth used so that all the device gets fair share.

Each Bluetooth receiver maintains a *Frequency Usage Table* where a bit error

rate measurement, BER_f , is associated to each frequency as shown in Figure 6.

Use	Frequency Offset	BER_f
	0	10^{-3}
	1	10^{-1}
	2	10^{-2}
	3	10^{-1}
	...	
	76	10^{-4}
	77	10^{-3}
	78	10^{-3}



Clear (Can be sent)

Used (Can't be Sent)

Figure 6: Frequency Usage Table

Frequencies are classified according to a criteria that measure the level of interference in the channel and are marked *used* if the bit error rate BER is more than bit error rate threshold value, BER^T or else *clear*. Other criteria such as frame error rate, packet loss, or the received signal strength can be used in addition to the bit error measurement to detect a high level of interference in a specific frequency band.

Since the master device controls all transmissions in the piconet, the slaves need to send their *Frequency Usage Table* in the form of status update messages. The scheduler at the master uses the measurements collected during the *Channel Estimation* phase to optimize the frequency allocation on each time slot and avoid a packet transmission in a receiving channel with a high level of interference. Figure 7 illustrates the frequency allocation that occurs at the master. As shown in figure 7, frequency 78 is used to communicate with slave S_i, while frequencies 76, 1 and 0 are assigned to slaves S_i, S_{i+1} and S_{i+1} respectively. M in figure 7 marks a receiving slot for the master device while an S is a receiving frequency for a slave device. Although, the master scheduler attempts to maximize channel utilization, it intentionally leaves certain slot pairs empty if either the master or the slave receiving frequency is *used*. As shown in Figure 7, frequencies 16, 2, 7 and 77 are not used since frequencies 2 and 77 are not *clear* for the master.

The basic idea in the credit system is to control the bandwidth allocated to each device in order to ensure that no device gets more than its fair share of the available bandwidth. Thus, devices with a positive credit counter, c_i, are allowed to send data. One of the ways to compute credits is to use the max-min fairness criteria. The method to calculate the credit is given in [4] and can be referred for more details. The credit is calculated based on the probability of getting the channel free and taking average of the available channels.

The remaining component of the algorithm is to actually give the priority access to certain devices. The devices with less number of good channels are given higher priority over other devices that have more channels available.

4. Discussion:

In [1] which describes the Goodness Counter Link state history (GC - LCH) does not give the analytical modeling or simulation result. The other methods discussed like Link State History (LSH) [3] and Bluetooth Interference Aware Scheduling (BIAS) [4] present analytical

Receiving Frequencies			Frequency Offset
Master	$Slave_{i+1}$	$Slave_i$	
			0
			1
			2
			3
			…
			76
			77
			78

Channel estimation Procedure

Set of Receiving Devices per (Clear) Frequency

$A_0 = \{M, S_{i+1}\}$

$A_1 = \{M, S_{i+1}\}$

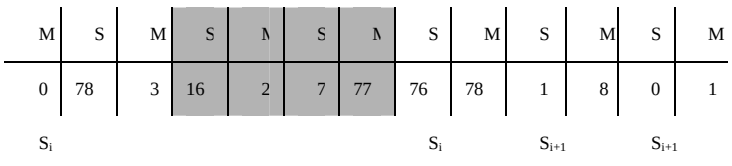
$A_2 = \{\}$

$A_3 = \{M, S_i, S_{i+1}\}$

$A_{76} = \{M, S_i, S_{i+1}\}$

$A_{77} = \{S_i, S_{i+1}\}$

$A_{78} = \{M, S_i\}$



Bluetooth Receiving Frequency Pattern

Figure 7: Master Frequency Allocation Scheduling

modeling and simulation results.

The LHS scheme is a good method to avoid the interference in Bluetooth device. It mainly aims to communicate with only those devices that are not having interference in their channel. It is observed that modeling of the channel as a two state process is done. Stochastic processes were used to analyze the working of the LSH scheme to establish the limits the start and end of the bad period. But the simulations were not exhaustive, it was mentioned that only three cases were considered to show the correctness of the analytical results. The performance enhancement of the throughput by 15% and the goodput[1] by 30% was shown on these three cases. The algorithm modeled was good. But an error in the algorithm could be witnessed. It was mentioned that when the Master cannot find any slave with whom it can communicate, it (Master) chooses the slave with highest LSH counter. It implies that the Master tries to connect the slave having interference. This will result in further increase of the counter again and again (since the counter keeps on incrementing due to interference). It is better if the master tries to communicate with slave having least Link State counter.

In [4], specific distinction between *upstream* and *downstream* traffic was not made. Unlike [1] and [3] it focus on a scheduling policy for polling slaves in order to allow them to access the channel. It assumed that the source of interference to the Bluetooth system is another wireless system working in the same frequency as Bluetooth. It emphasized that the technique can be adapted to any other interference environment as well but this cannot be acknowledged.

[1] Goodput is the ratio of the number of error free message-response pair packets in a given time interval to the maximum number of message-response pair packets that can be exchanged in the same time interval.

5. Conclusion:

In Bluetooth, Frequency Hopping and Adaptive Frequency Hopping provide good protection against randomly occurring interference signals of short duration. However, there are many cases when the interference may last longer reducing the throughput drastically. The algorithm discussed in [1] and [3] proposes methods to alleviate this by avoiding communication with the slave device facing interference. But the device facing interference is left out of the network if communication with it is avoided. In [4] an algorithm was proposed which takes care of this problem by allocating the slave another frequency if it is facing interference at a particular frequency.

It would be a good idea to use both schemes together. Since one scheme is good for detecting the interferences and other is good for scheduling. If both of the schemes were combined then there would be no starvation of any slave and would provide a better solution for the problem in discussion.

References:

1. Anvekar, D.K.; Kapoor, M**. Frequency look-ahead and link state history based interference avoidance in wireless picocellular networks.** Personal Wireless Communications, 2000 IEEE International Conference on , 2000 Page(s): 434 –438.
2. Charles D. Knutson, David K. Vawdrey and Eric S. Hall. **Bluetooth.** IEEE Potentials , Volume: 21 Issue: 4, Oct./Nov. 2002 Page(s): 28 –34
3. Deb, S.; Kapoor, M.; Sarkar, A. **Error avoidance in wireless networks using link state history.** INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , Volume: 2 , 2001 Page(s): 786 -795 vol.2
4. Golmie, N. Chevrollier, N. ElBakkouri, I. **Interference aware Bluetooth packet scheduling.** Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE page(s): 2857 - 2863 vol.5 25-29 Nov. 2001
5. Haartsen, J.C. **The Bluetooth radio system**. IEEE Personal Communications , Volume: 7 Issue: 1 , Feb. 2000 Page(s): 28 -36
6. Niklas Johansson, Fredrik Alriksson, Ulf Jönsson**. JUMP mode---a dynamic window-based scheduling framework for Bluetooth scatternets**. SIGMOBILE: ACM Special Interest Group on Mobility of Systems.
7. Natarajan, B.; Nassar, C.R.; Shattil, S. **Enhanced Bluetooth and IEEE 802.11 (FH) via multi-carrier implementation of the physical layer.** IEEE Emerging Technologies Symposium on , 2001 Page(s): 129 –133.
8. Paul T. M. van Zeijl **One-chip bluetooth Asic challenges.** EDAC : Electronic Design Automation Consortium IEEE-CAS : Circuits & Systems , SIGDA : ACM Special Interest Group on Design Automation.
9. Sand K., **Bluetooth**, http://www.tcm.hut.fi/Opinnot/Tik111.550/1999/Esitelmat/Bluetooth/bluetooth.html
10. The Bluetooth Specification, http://www.bluetooth.com
11. W. S. Wang. **Bluetooth: a new era of connectivity.** IEEE Microwave Magazine Sept. 2002 Volume: 3 Issue: 3 page(s): 38 - 42
12. White paper: Bluetooth And Bluetooth Internet Access Points. http://www.pico.net/download/whitepaper.pdf
13. White paper: http://www.mobilebluetooth.com/whitepaper.pdf
14. White paper: http://www.pico.net/download/Bluetooth_Networking.pdf
15. White paper: http://www.pico.net/download/whitepaper.pdf
16. http://www.thewirelessdirectory.com
17. Motorola's Bluetooth site http://www.motorola.com/bluetooth
18. http://www.palowireless.com/infotooth/