Mobile Networks and Applications 9, 21–31, 2004

Bluetooth Dynamic Scheduling and Interference Mitigation

N. GOLMIE
National Institute of Standards and Technology, Gaithersburg, MD 20899, USA

Abstract. Bluetooth is a cable replacement technology for Wireless Personal Area Networks. It is designed to support a wide variety of applications such as voice, streamed audio and video, web browsing, printing, and file sharing, each imposing a number of quality of service constraints including packet loss, latency, delay variation, and throughput. In addition to QOS support, another challenge for Bluetooth stems from having to share the 2.4 GHz ISM band with other wireless devices such as IEEE 802.11. The main goal of this paper is to investigate the use of a dynamic scheduling algorithm that guarantees QoS while reducing the impact of interference. We propose a mapping between some common QoS parameters such as latency and bit rate and the parameters used in the algorithm. We study the algorithm's performance and obtain simulation results for selected scenarios and configurations of interest.

Keywords: WPANs, Bluetooth, interference, MAC scheduling

1. Introduction

Today most radio technologies considered by Wireless Personal Area Network (WPAN) industry consortia and standard groups including the Bluetooth Special Interest Group [1], HomeRF, and the IEEE 802.15, employ the 2.4 GHz ISM frequency band. This same frequency band is already in use by microwave ovens and the popular Wireless Local Area Network (WLAN) devices implementing the IEEE 802.11 standard specifications [8].

However, instead of competing with WLANs for spectrum and applications, WPANs are intented to augment many of the usage scenarios and operate in conjunction with WLANs, i.e., come together in the same laptop, or operate in proximity in an office or conference room environment. For example, Bluetooth can be used to connect a headset, or PDA to a desktop computer, that, in turn, may be using WLAN to connect to an Access Point placed several meters away.

Thus, an issue of growing concern is the coexistence of WLAN and WPAN in the same environment. Several techniques and algorithms aimed at reducing the impact of interference have been considered. These techniques range from collaborative schemes intended for Bluetooth and IEEE 802.11 protocols to be implemented in the same device to fully independent solutions that rely on interference detection and estimation. In particular:

- *Collaborative mechanisms.* Mechanisms for collaborative schemes have been proposed to the IEEE 802.15 Coexistence Task Group and are based on a Time Division Multiple Access (TDMA) solution that alternates the transmission of Bluetooth and WLAN packets (assuming both protocols are implemented in the same device and use a common transmitter) [9]. A priority of access is given to Bluetooth for transmitting voice packets, while WLAN is given priority for transmitting data.
- *Non-collaborative mechanisms.* The non-collaborative mechanisms range from adaptive frequency hopping [11] to packet scheduling and traffic control [4]. They all use similar techniques for detecting the presence of other devices in the band such as measuring the bit or frame error rate, the signal strength or the signal to interference ratio (often implemented as the Received Signal Indicator Strength (RSSI)). Frequency hopping devices may be able to detect that some frequencies are used by other devices and thus modify their frequency hopping pattern. They can also choose not to transmit on "bad" frequencies. The first technique is known as adaptive frequency hopping, while the second technique is known as MAC scheduling. The main advantage of scheduling is that it does not require changes to the Bluetooth specifications.

In this paper we present a Bluetooth Interference Aware Scheduling (BIAS) algorithm to deal with coexistence. This algorithm takes advantage of the fact that devices in the same piconet will not be subject to the same levels of interference on all channels of the band. The basic idea is to utilize the Bluetooth frequency hopping pattern and distribute channels to devices such that to maximize their throughput while ensuring fairness of access among users.

In this paper, we propose several extensions to a preliminary discussion of the algorithm [4] in order to address (1) priority scheduling, (2) dynamic changes in the environment, and (3) asymmetric scenarios where packet lengths and data rates are chosen differently in the upstream (slave to master transmission) and downstream (master to slave transmission) directions. In addition, we describe how to map commonly used QOS parameters, namely bit rate, and jitter and the parameters used in BIAS. Simulation results for scenarios and configurations of interest are presented and performance is measured in terms of packet loss and mean access delay.

The remainder of this paper is organized as follows. In section 2, we give some general insights on the Bluetooth interference environment. In section 3, we describe BIAS and discuss the mapping of QOS parameters. In section 4,

we present simulation results and offer concluding remarks in section 5.

2. Interference environment

Since Bluetooth operates in the 2.4 GHz band along with other wireless technologies such as 802.11, high and low rate WPAN (802.15.3 and 4), the resulting mutual interference leads to significant performance degradation.

In this paper, we assume that interference is caused by an 802.11 spread spectrum network operating in proximity of the Bluetooth piconet. This represents the worst case interference for Bluetooth. Golmie et al. [6] use a detailed MAC and PHY simulation framework to evaluate the impact of interference for a pair of WLAN devices and a pair of Bluetooth devices. The results indicate that Bluetooth performance may be severely impacted by interference with packet loss of 8% and 18% for voice and data traffic, respectively. In [6], the authors investigate the effect of several factors, such as transmitted power, offered load, packet size, hop rate, and error correction on performance. First, they note that power control may have limited benefits in an interference environment. Increasing the Bluetooth transmission power even ten times is not sufficient to reduce the Bluetooth packet loss. Second, using a shorter packet size leads to less packet loss for Bluetooth at the cost of causing more interference on WLAN. Overall, the results exhibit a strong dependence on the type and characteristics of the traffic distribution used.

Additional analytical [5,10] and experimentation [3,7] results confirm these findings.

3. Bluetooth Interference Aware Scheduling

In this section, we present a Bluetooth Interference Aware Scheduling (BIAS) algorithm that consists of several components, namely, (i) dynamic channel estimation, (ii) credit computation, and (iii) access priority. A preliminary discussion of BIAS appeared in [4].

In this sequel, we assume that traffic from slave S_i to the master (upstream) is characterized by a data rate, γ^i_{up}, equal to $(N^i_{\text{peak}} \cdot l^i_{\text{up}})/p^i$ where N^i_{peak} is the number of packets sent back-to-back within a poll interval, p^i, and l^i_{up} is the packet length (1, 3, or 5 slots depending on the packet type). Similarly, the data rate in the downstream (from the master to slave S_i) is characterized by γ^i_{dn} equal to $(N^i_{\text{peak}} \cdot l^i_{\text{dn}})/p^i$. Note that N^i_{peak} and p^i are the same in the upstream and downstream, since every packet in the upstream corresponds to one in the downstream. In addition, we assume the following transmission rules for the master and slave.

Master – The master polls S_i every p^i slots in order to guarantee γ^i_{up} in the upstream direction. A poll message can be either a data or POLL packet. A data packet is sent if there is a packet in the queue destined for S_i. This packet contains the ACK of the previous packet received from S_i. In case there is no data to transmit and the master needs to ACK a previous slave transmission, it sends a NULL packet.

Slave S_i – Upon receipt of a packet from the master, the slave can transmit a data packet. This data packet contains the ACK information of the master to slave packet transmission. In case the slave does not have any data to send, it sends a NULL packet in order to ACK the previous packet reception from the master. No ACK is required for a NULL message from the master.

In a nutshell, we propose a method that allows the master device, which controls all data transmissions in the piconet, to avoid data transmission to a slave experiencing a "bad" frequency. Furthermore, since a slave transmission always follows a master transmission, using the same principle, the master avoids receiving data on a "bad" frequency, by avoiding a transmission on a frequency preceding a "bad" one in the hopping pattern.

This simple scheduling scheme illustrated in figure 1 needs only be implemented in the master device and translates into the following transmission rule. *The master transmits in a slot after it verifies that both the slave's receiving frequency, f_s,*

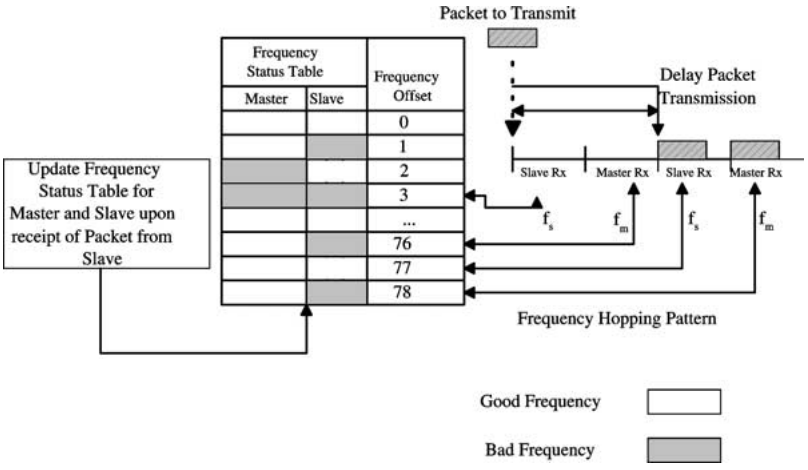


Figure 1. Interference Aware Scheduling.

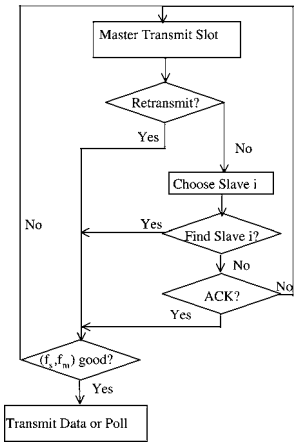


Figure 2. Master packet transmission flow diagram.

and its own receiving frequency, f_m, are "good". Otherwise, the master skips the current transmission slot and repeats the procedure over again in the next transmission opportunity.

Figure 2 describes the master's transmission flow diagram. In addition, to checking the slave's and the master's receiving frequencies pair, (f_s, f_m), the algorithm incorporates bandwidth requirements, and quality of service guarantees for each master/slave connection in the piconet. This bandwidth allocation is combined with the channel state information and mapped into transmission priorities given to each direction in the master/slave communication. It is shown in the "choose slave" routine in the flow diagram. Note that the master invokes the "choose" routine after serving the retransmission ACK queue for packets sent by the master requiring retransmission.

In the remainder of this section, we discuss (a) a dynamic channel estimation procedure, (b) a credit allocation function, and (c) a service priority routine that schedules packet transmissions to devices according to their service requirements and the state of the channel.

3.1. Dynamic channel estimation

Estimation is mainly based on measurements conducted on each frequency or channel in order to determine the presence of interference. Several methods are available ranging from BER, RSSI, packet loss rate, and negative ACKs. In this discussion, the estimation is based on negative ACKs, which belongs to the class of implicit methods that do not require messages to be exchanged between the master and the slave devices. First, we define two phases in the channel estimate procedure as illustrated in figure 3. During the *Estimation Window*, EW, packets are sent on all frequencies regardless of their classification.

Note that in case no data traffic is available for transmission, POLL/NULL packets could be exchanged between the master and the slave in order to probe the channel and collect measurements. This POLL/NULL exchanged is designed in most implementations to keep the connection alive and check the status of the slave. It comes at the expense of causing more interference on other systems. EW takes place at the

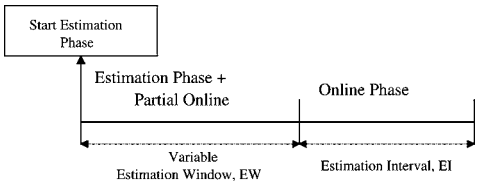


Figure 3. Implicit estimation.

beginning of every *Estimation Interval*, EI, and is followed by an *Online* phase where the master uses only "good" frequencies to selectively send data and POLL packets to slaves in the piconet. Next, we give a lower bound on the EW and describe how to adjust EI based on the environment's dynamics.

Estimation Window. The time to perform the channel estimation depends on the frequency hopping rate since the methods used to perform the classification depend on packet loss measurements per frequency visited. A lower bound calculation is as follows. First, we assume a hop rate of 1600 hops/s given single slot packets. For each receiver the hopping rate is $1600/2$ hops/s, or 800 hops/s since nodes receive on every other "frequency" or "hop" in the sequence. Next, we consider the Bluetooth frequency hopping algorithm. In a window of 32 frequencies, every frequency is selected once, then the window is advanced by 16 frequencies, and the process is repeated. Therefore, it takes 5 windows of 32 frequencies in order to visit each of the 79 frequencies twice. In other words, 160 hops visit each frequency twice. The time to visit each frequency four times per receiver is $160/800 \cdot 2 = 0.4$ seconds or 400 ms. In fact, 400 ms constitutes a lower bound assuming full load and single-slot packets.

In order to avoid having to fix the EW, or compute it manually, we propose a simple technique to dynamically adjusts the window based on the number of times, N_f, each frequency in the band should be visited. For example, if N_f is equal to 2, then each receiving frequency in the band is visited at least twice, i.e., the estimation phase ends only when the last frequency in the band has been used twice for each device in the piconet. Note that, avoiding "bad" frequencies can start before EW ends, or as soon as frequency status information becomes available.

Estimation Interval. How often to update the channel estimation depends on the application and the dynamics of the scenario used. We propose an adaptive procedure to adjust EI, which is the interval between two consecutive estimation windows.

First, we let δ, be the percentage of frequencies that change classification status (from "good" to "bad" or vice versa) during the previous estimation phase. More formally, let $S(f,t)$ be the status of frequency f at time t.

$$S(f,t) = \begin{cases} 1 & \text{if } f \text{ is "good"}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Using the exclusive bit "OR" operation between $S(f,t)$ and $S(f,t+1)$ represents the change of status of frequency f from

time t to $t+1$. A change of status leads to a logic "1" while a no change yields a logic "0". Summing over all frequencies and dividing by the number of frequencies available, which is 79 in this case, is then equal to δ.

$$\delta_{t+1} = \frac{1}{79}\sum_{f}^{79}\big(S(f,t) \oplus S(f,t+1)\big). \tag{2}$$

Initially, EI is set to $EI_{\min}$. Then, EI is updated every interval, t, according to the rationale that if a change were to happen it is likely to happen again in the near future and therefore EI is set to $EI_{\min}$. Otherwise, the window is doubled.

$$EI_{t+1} = \begin{cases} \max(2 \cdot EI_t, EI_{\max}) & \text{if } \delta_{t+1} \leqslant 0.1, \\ EI_{\min} & \text{otherwise.} \end{cases} \tag{3}$$

3.2. Credit allocation

The credit system controls 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. Since the rate in the upstream can be different from the rate in the downstream, we define c^i_{up} and c^i_{dn} for both the upstream and downstream credits. Credits can be computed according to the upstream and downstream rates negotiated as follows:

$$c^i_{\text{up}} = \gamma^i_{\text{up}} \cdot N, \qquad c^i_{\text{dn}} = \gamma^i_{\text{dn}} \cdot N, \tag{4}$$

where N is the number of slots considered in the allocation and $\gamma^i_{\text{up/down}} = l^i_{\text{up/down}} \cdot N^i_{\text{peak}}/p^i$. Credits are decremented by the number of slots used in each data packet transmission. The transmission of POLL and NULL packets does not affect the credit count based on the rationale that credits are not required for the transmission of POLL and NULL messages. An interesting question is how to compute γ or derive it from application QOS parameters such as delay, peak bandwidth, and jitter. Let d (seconds), r (bits/s), σ (seconds) represent delay, peak bandwidth, and jitter, respectively. r is part of the L2CAP QOS parameters and for some applications is negotiated between the master and the slave at connection setup. r is equal to $(N_{\text{peak}} \cdot E_l \cdot 8)/(p \cdot 625 \cdot 10^{-6})$ and $\gamma = (r \cdot l \cdot 625 \cdot 10^{-6})/(E_l \cdot 8)$. Note that E_l is the number of information bytes contained in a packet of length l. Table 1 gives E_l corresponding to the various DH formats. The choice of l depends on the L2CAP packet size, k. When $k \leqslant E_5$, $N_{\text{peak}} = 1$ and l is such that:

$$l = \begin{cases} 1 & \text{if } 0 < k \leqslant 27, \\ 3 & \text{if } 27 < k \leqslant 183, \\ 5 & \text{if } 183 < k \leqslant 339. \end{cases} \tag{5}$$

Table 1
Packet encapsulation rate for DH packets.

Packet type	l	E_l (bytes)
DH1	1	27
DH3	3	183
DH5	5	339

However, when $k > E_5$, higher layer packets (L2CAP) are segmented into N_{peak} packets. The aim is to find N_{peak} equal to

$$N_{\text{peak}} = \left\lceil \frac{k}{E_l} \right\rceil \tag{6}$$

such as to minimize $N_{\text{peak}} \cdot l$, or the total number of slots needed. Furthermore, since master and slave transmission alternate, the end-to-end delay of a packet accounts for the segmentation and the transmission of packets in both directions. Therefore, the choice of l_{up} and l_{dn} are loosely constrained by the delay requirements as follows:

$$N_{\text{peak}} \cdot (l_{\text{up}} + l_{\text{dn}}) \leqslant \frac{d}{625 \cdot 10^{-6}}, \tag{7}$$

where $625 \cdot 10^{-6}$ is the length of a slot in seconds. Finally, the choice of p is determined by σ as follows:

$$2 \leqslant p \leqslant \frac{\sigma}{625 \cdot 10^{-6}}, \tag{8}$$

where 2 is the minimum value for the poll interval since every other slot is dedicated to a master (or slave) transmission.

In case r, d, and σ cannot be determined from the application QOS, γ can be set to $1 - \sum \gamma^i$, the leftover bandwidth after having calculated γ for all other applications with known service rates ($\sum \gamma$).

3.3. Service priority

The third component of the algorithm is to give an access priority to devices based on their channel conditions and their allocated credits.

We let u_i be the probability that a pair of master/slave transmission slots are "good". Thus, u_i represents the available spectrum to slave S_i, and we write:

$$u_i = \min\Bigg(\left(1 - \frac{1}{79}\right), P(\text{slave } i \text{ has a good receiving frequency}) \times P(\text{master has a good receiving frequency})\Bigg), \tag{9}$$

where

$$P(\text{device } i \text{ has a good receiving frequency}) = \frac{\text{Number of good channels}_i}{\text{Total number of channels}}. \tag{10}$$

We use a two-tier system with high and low priorities, denoted by A, and B, respectively. Priority A is used to support delay constrained applications such as voice, MP3, and video. On the other hand, priority B, is used to support best effort connections such as ftp, http, print, email. The scheduling routine services priority A devices first, and priority B devices second. Also, among same tier connections, we choose to give devices with fewer number of good channels the right of way over other devices that have more channels available. The priority access is determined according to a weight factor, w, that is the product of the credits and the probability of

Table 2
Definition of parameters used in the scheduling algorithm.

Parameters	Definition
$\gamma^i_{\text{up,dn}}$	Rate allocated for device i in the upstream anddownstream
$w^i_{\text{up,dn}}$	Weight for device i
$c^i_{\text{up,dn}}$	Credit for device i
N	Number of slots considered in the allocation
u^i	Available frequency usage for device i

experiencing a bad frequency. w^i_{up} and w^i_{dn} are computed as follows:

$$w^i_{\text{up}} = c^i_{\text{up}} \cdot (1 - u_i), \qquad w^i_{\text{dn}} = c^i_{\text{dn}} \cdot (1 - u_i). \tag{11}$$

The master schedules a data transmission for slave i such as to maximize the product of the weights in the up and downstreams:

$$i = \max^f_S \left(w^i_{\text{up}} \cdot w^i_{\text{dn}}\right). \tag{12}$$

To transmit a POLL packet, the master looks only at the weight function in the upstream:

$$i = \max^f_S \left(w^i_{\text{up}}\right). \tag{13}$$

The selection of a slave is restricted over the set of slaves S that can receive on the master's current transmission frequency, f. Thus, any slave that experiences a "bad" channel on the current transmission frequency is not considered. Four sets of slaves are formed, A^f_{data}, A^f_{poll}, B^f_{data}, and B^f_{poll}. A_{data} and A_{poll} represent the set of high priority connections requiring data and POLL packet transmissions, respectively. Similarly, B_{data} and B_{poll} represent low priority connections. First, the algorithm tries to schedule a packet to high priority slaves in group A, then a POLL packet, before it moves to group B. The credit counters and weights are updated accordingly after every master's transmission. Table 2 summarizes the parameters used in the algorithm and their definition. The algorithm's pseudocode is given in table 11.

4. Performance evaluation

In this section, we present simulation results to evaluate the performance of BIAS. The experiments illustrate the algorithm's responsiveness to changes in the environment and the support of QOS. The results obtained are compared with Round Robin (RR) scheduling. Our simulation environment is based on a detailed MAC, PHY and channel models for Bluetooth and IEEE 802.11 (WLAN) as described in [6]. The parameters used in the setup vary according to the experiment. The common simulation parameters are summarized in table 3. The simulations are run for 900 seconds of simulated time unless specified otherwise. We run 10 trials using a different random seed for each trial. In addition, to plotting the mean value, we verify that that the statistical variation around the mean values are very small (less than 1%).

The performance metrics include the *packet loss*, the *mean access delay*, and the channel *estimation transient time*. The packet loss is the percentage of packets dropped due to interference over the total number of packets received. The access delay measures the time it takes to transmit a packet from the time it is passed to the MAC layer until it is successfully received at the destination. The delay is measured at the L2CAP layer. The estimation transient time measures the time it takes a Bluetooth device to detect the presence of a "bad" frequency, i.e., from the time a packet loss occurs until the frequency is classified "bad". This average is provided on a per frequency basis.

Table 3
Common simulation parameters.

Bluetooth parameters	Values
ACL baseband packet encapsulation	DH5
Transmitted power	1 mW
WLAN parameters	**Values**
Packet interarrival time	2.172 ms
Offered load	60% of channel capacity
Transmitted power	25 mW
Data rate	11 Mbit/s
PLCP header	192 bits
Packet header	224 bits
Payload size	12000 bits

4.1. Experiment 1: base case

This experiment includes Bluetooth performance results for the reference scenario when no interference is present. It represents a base case since the effects of BIAS are quantified and compared against the reference scenario. It also covers different levels of interference caused by WLAN systems operating in close proximity. Thus, we examine Bluetooth's performance when 1, 2, and 3 WLAN interfering systems are operational and compare that to the ideal performance when no interference is present. Note that, the maximum number of non-overlapping channels for WLAN systems is 3, i.e., there could be up to 3 WLAN networks operating simultaneously using different non-overlapping channels. In each case, results are obtained with BIAS and RR scheduling. The benefits of using BIAS are discussed in terms of packet loss and access delay.

Topology. We use the topology illustrated in figure 4 that consists of 3 WLAN systems (source–sink pairs), and one Bluetooth piconet with one master and one slave device. In a first step, we record the results of Bluetooth when no WLAN system is present. Then, we add one WLAN system at a time starting with WLAN (Source/Sink) 1, followed by WLAN (Source/Sink) 2, and 3.

Traffic. For Bluetooth, a generic source that generates DH5 packets is considered. The packet interarrival mean time in seconds, t_{B}, is exponentially distributed and is computed according to

$$t_{\text{B}} = 2 \cdot l \cdot 0.000625 \cdot \left(\frac{1}{\lambda} - 1\right), \tag{14}$$

where l is the packet length in slots and λ is the offered load. We assume that WLAN is operating in the Direct Sequence Spread Spectrum (DSSS) mode. The WLAN source is transmitting data packets to the sink which is responding with ACKs. The WLAN packet payload is set to 12000 bits transmitted at 11 Mbit/s, while the PLCP header of 192 bits is transmitted at 1 Mbit/s. The packet interarrival time in seconds, t_W, is exponentially distributed and its mean is computed according to

$$t_W = \left(\frac{192}{1000000} + \frac{12224}{11000000}\right)\frac{1}{\lambda}. \quad (15)$$

Results. Figure 5 gives the packet loss (a) and the mean access delay (b) measured at the slave for a variable Bluetooth offered load (5–80%). Observe that when no WLAN system is present, the packet loss is zero and the access delay remains flat at around 4 ms. This represents a reference measure for the Bluetooth performance when there is no interference. Each WLAN system addition an increase of 15% in packet loss as shown in figure 5(a). The packet loss is around 15%, 30% and 45% when one, two, and three WLAN systems are present, respectively. Repeating the same experiments using BIAS, brings the packet loss down to zero for any number of WLAN systems. The delay trends captured in figure 5(b) are consistent with the packet loss results. Using BIAS yields lower delays than when RR is used. When one WLAN system is present, the delay curve with BIAS is flat at 5 ms (a 1 ms increase compared to the reference case when no interference is present). When 2 WLAN systems are present, the delay curve takes off at 35% with RR, while the curve remains flat until 60% with BIAS. When 3 WLAN systems are present, the delay curve takes off sharply at 15% with RR, while the knee of the curve remains lower with BIAS (shifted to the right).

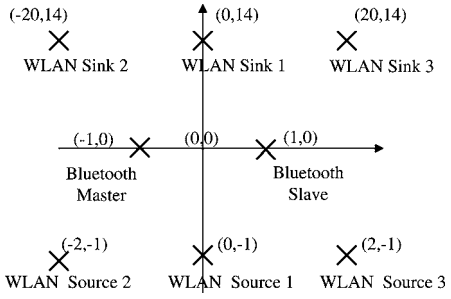


Figure 4. Topology for experiments 1 and 2.

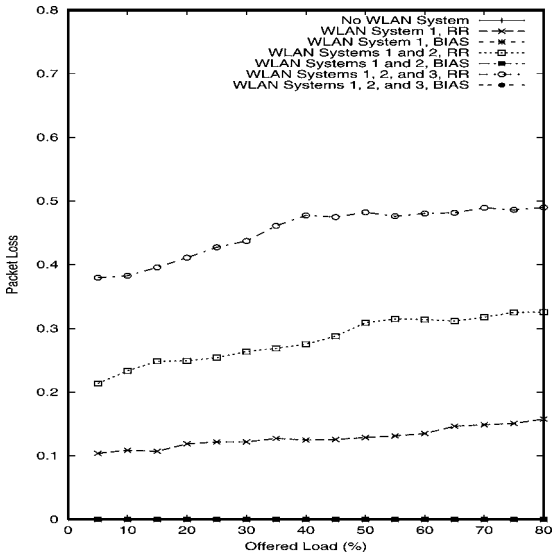
4.2. Experiment 2: dynamic behavior

In this experiment, we focus on BIAS's responsiveness to transient effects and sudden changes in the environment. We measure the channel estimation transient time per frequency and over the entire spectrum. We design an experiment where the WLAN traffic is turned on and off several times during each simulation run (about 30 times).

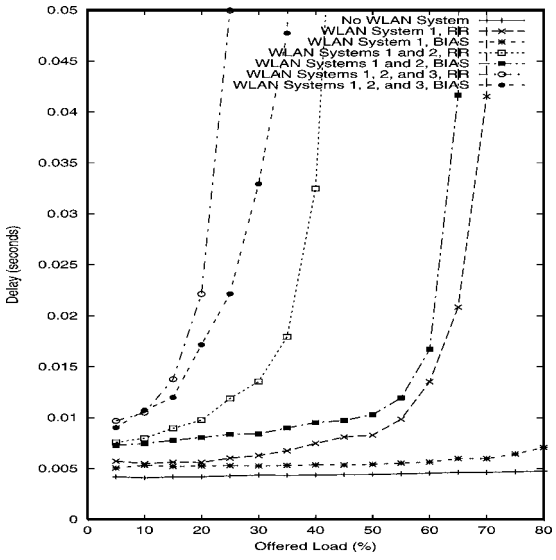
Topology. We use the topology of figure 4 with one WLAN system (Source/Sink 1) and the Bluetooth master/slave pair.

Traffic. The traffic is based on bulk data. The offered load for Bluetooth is varied between 10 and 100%, while for WLAN the offered load is set to 60%. For Bluetooth, both DH1 (1 slot) and DH5 (5 slots) packets are used in order to compare the difference in transient times. The time the WLAN connection is ON, T_{ON}, is exponentially distributed with a mean equal to 10 seconds, while the time the WLAN connection is OFF, T_{OFF}, is also exponentially distributed with mean equal to 20 seconds. Each simulation is run for 900 seconds. Unless specified otherwise, we set $EI_{min} = 2$ seconds, $EI_{max} = 100$ seconds, $N_f = 1$.

Results. Figures 6(a) and 6(b) give the packet loss and access delay, respectively, measured at the Bluetooth slave de-



(a)



(b)

Figure 5. Experiment 1. Variable number of WLAN interfering systems. (a) Probability of packet loss. (b) Mean access delay.

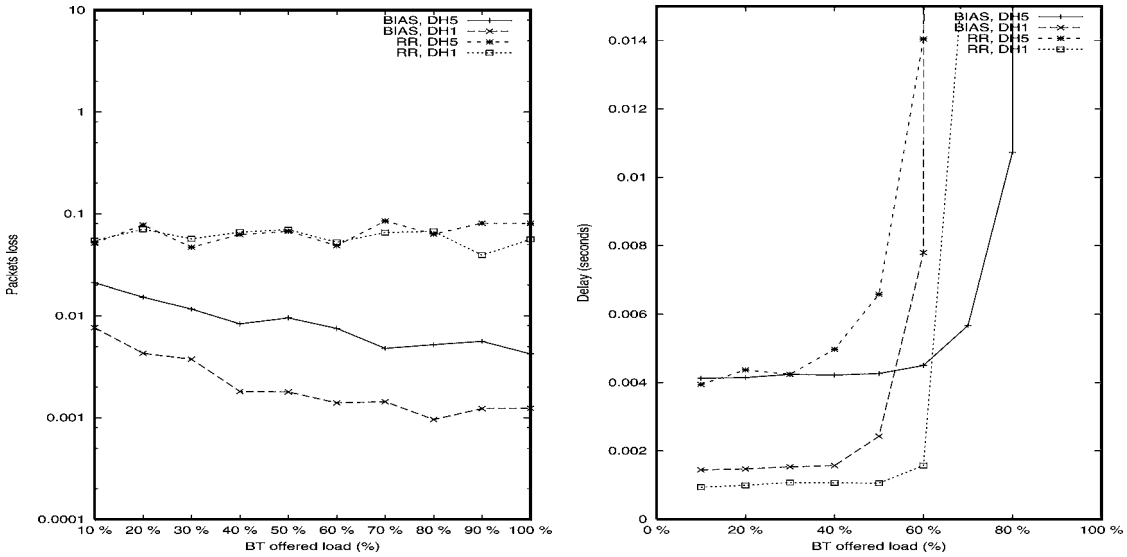


Figure 6. Experiment 2. Variable Bluetooth offered load. (a) Probability of packet loss. (b) Mean access delay.

vice. The packet loss obtained with BIAS is negligible (less than 2%) for both DH1 and DH5 packets. On the other hand the packet loss with Round Robin (RR) is close to 10%. The access delay obtained with BIAS for DH1 packets is lower than the delay for DH5 packets for offered loads under 70% (it is around 1.5 ms for DH1 packets, and 4 ms for DH5 packets). The knee of the curve for DH5 packets is located around 80% of the offered load while it is at 60% for DH1 packets. Observe that BIAS gives lower access delays than RR for DH5 packets (between 40% and 80% offered load). However, the same does not apply to DH1 packets, in which we observe a slight increase in access delay (0.5 ms) with BIAS compared to RR. For short packets (DH1) retransmissions due to packet loss (RR), and delay in transmission due to "bad" frequency avoidance (BIAS), yields comparable delays. Furthermore, given that the probability of packet loss (and retransmission) is small for short packets, RR gives lower access delays on average. Figure 7 gives the time it takes to estimate a "bad" frequency using DH1 and DH5 packets. The use of DH5 packets leads to a higher round trip transmission time, and therefore increases the transient time, up to 1.5 ms while it is around 0 μs for DH1 packets.

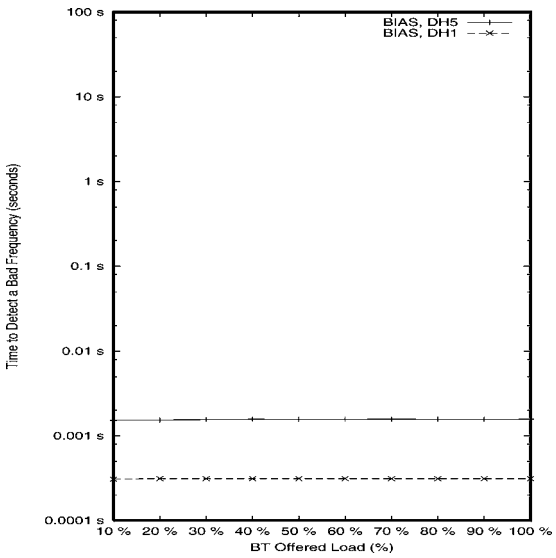


Figure 7. Experiment 2. Variable Bluetooth offered load. Time to estimate a "bad" channel.

4.3. Experiment 3: QOS support

This experiment highlights the support of QOS in an environment where devices experience different levels of interference and connections have a range of service requirements.

Topology. We use the topology illustrated in figure 8. Slaves 1 and 2 experience the same level of interference, while slave 3 does not experience any interference. The y-coordinate of the WLAN FTP server is varied along the y-axis in order to vary the level of interference on the Bluetooth piconet.

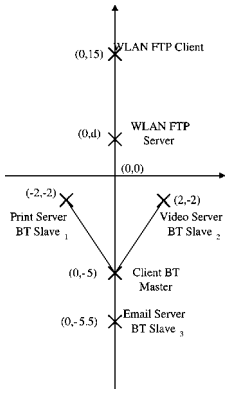


Figure 8. Topology for experiment 3.

Traffic. For Bluetooth, we consider three application profiles, namely, Print, Video, and Email. We use print, video, and email traffic between slaves 1, 2, 3 and the master, respectively. Note that the master is the client process in all three connections. The profile parameters are given in table 4. The WLAN uses the FTP profile described in table 5.

Since the video application generates roughly around 93 and 58 packets in the upstream and downstream directions, respectively, and since it is often difficult to predict the exact traffic distributions, the rate is divided evenly between both directions. Thus, we set $\gamma_{\text{up}}^2 \gamma_{\text{dn}}^2 = 0.25$. The two other applications, share the leftover bandwidth ($\gamma_{\text{up,dn}}^{1,3} = (1 - 0.5)/4 = 0.125$).

Table 4
Bluetooth application profile parameters.

Parameters	Distribution	Value
Email		
Send interarrival time (sec)	Exponential	120
Send group	Constant	3
Receive interarrival time (sec)	Exponential	60
Receive group	Constant	3
Email size (bytes)	Exponential	1024
Print		
Print requests interarrival time (sec)	Exponential	30
File size	Normal	(30 K, 9 M)
Video		
Frame rate	Constant	1 frame/s
Frame size (bytes)	Constant	17280 (128 × 120 pixels)

Table 5
WLAN application profile parameters.

Parameters	Distribution	Value
FTP		
File interarrival time (sec)	Exponential	5
File size (bytes)	Exponential	5 M
Percentage of get		100%

Results. Figure 9 depicts the results when the WLAN y-coordinate is varied between 0 and 10 meters. In figure 9(a), the packet loss with BIAS is below 0.1% for all three slaves and the master. With RR, slave 1 (Print) and slave 2 (Video) vary between 15% and 3% of packet loss between 0 and 10 meters, respectively. While the packet loss for the master is above 20%. Slave 3 (Email) has a low packet loss with both BIAS and RR since it is far from the WLAN server.

The access delay for slave 2 (Video) in figure 9(b) is 0.3 seconds with BIAS, while it is almost double with RR (0.6 seconds). For Print, delays with BIAS are half the delays with RR (0.01 seconds as opposed to 0.02 seconds). The delays for Email are also reduced by half with BIAS.

4.4. Experiment 4: WLAN and multi-Bluetooth piconets interference

When two or more Bluetooth piconets are proximally located, one expects few collisions when the packets happen to be transmitted on the same frequency. However, the probability of such collisions is low as discussed in [2] since each piconet has a unique frequency sequence. Given that these packet collisions are random in nature and are already mitigated by frequency hopping, we do not expect significant performance improvements when BIAS is used since the packet loss is already very low. Furthermore, the fact that frequencies are eliminated due to other Bluetooth piconet interference may even cause delay increases. We illustrate this particular issue using the following scenario.

Topology. We use the topology illustrated in figure 10 representing a conference hall environment. It consists of one WLAN AP located at $(0, 15)$ meters, and one WLAN mobile at $(0, 0)$ meters. The WLAN mobile is the server device,

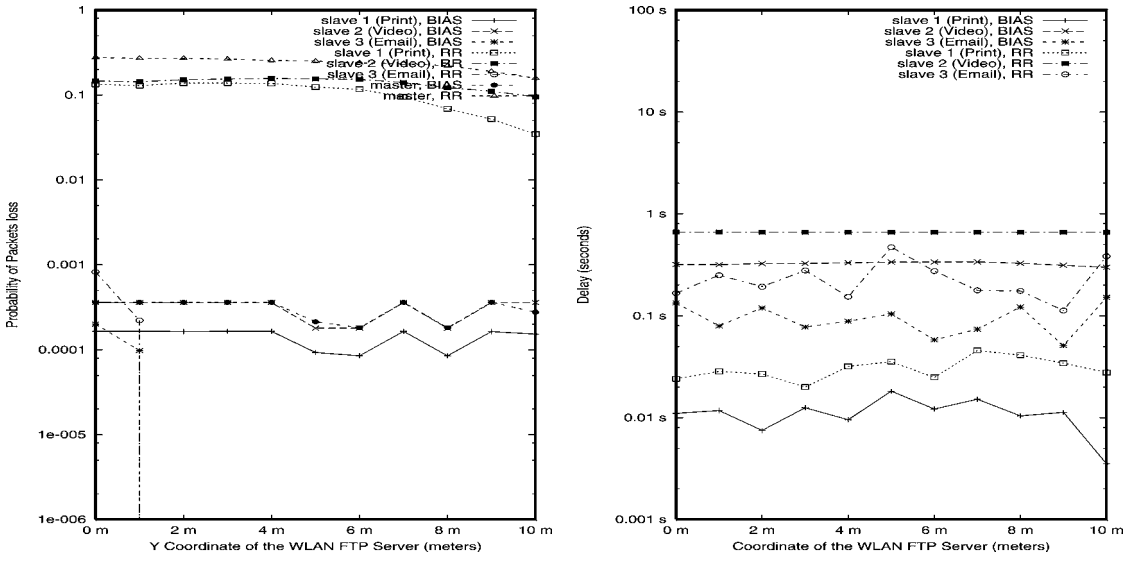


Figure 9. Experiment 3. Variable distance. (a) Probability of packet loss. (b) Access delay.

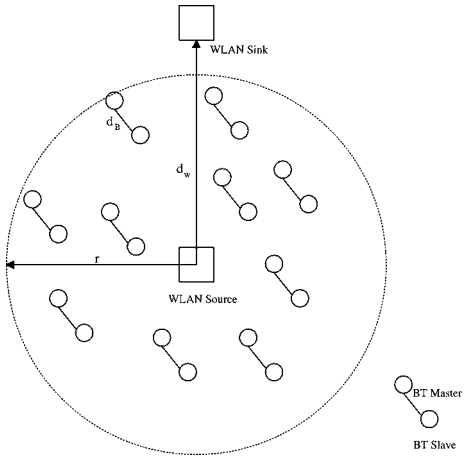


Figure 10. Topology for experiment 4.

Table 6
Profile parameters.

Parameters	Distribution	Value
Bluetooth FTP		
Percentage of put/get		100%
Inter-request time (sec)	Exponential	5
File size (bytes)	Exponential	250 K
HTTP		
Page interarrival time (sec)	Exponential	30
Number of objects per page	Constant	2
Object 1 size (bytes)	Constant	1 K
Object 2 size (bytes)	Uniform	(2 K, 100 K)

while the AP is the client. The distance between the WLAN AP and mobile is $d_W = 15$ meters. There are ten Bluetooth piconets randomly placed, covering a disk. The center of the disk is located at (0, 0) and its radius is $r = 10$ meters. We define d_B as the distance between a Bluetooth master and slave pair. $d_B = 1$ meter for half of the master and slave pairs, while $d_B = 2$ meters for the other half of the master and slave pairs.

Traffic. We run four experiments with different combinations of WLAN and Bluetooth applications, namely, HTTP and FTP. We use the application profiles available in the OPNET library and configure the parameters according to table 6. The WLAN FTP profile parameters are given in table 5.

Results. The results for the Bluetooth packet loss and access delay are given in tables 7 and 8, respectively. The results are grouped by application category (FTP, HTTP), and d_B, for each of the WLAN profiles. Overall, the packet loss results with BIAS are comparable to the packet loss obtained with RR. In some instances, the packet loss with BIAS is slightly lower than with RR, however the difference remains less than 2%. The access delays for Bluetooth is given in table 8. The results with BIAS and RR are comparable. However, there are no significant advantages in using BIAS.

Tables 9 and 10 give the packet loss and the access delay respectively for the WLAN FTP and HTTP profiles. Observe a significant reduction in packet loss with BIAS for both WLAN applications, in which the packet loss drops from 30% and 34% to 15% and 25% for the FTP and HTTP application, respectively. The access delay shown in table 10 is consistent with the packet loss results and shows slight improvements with BIAS. In summary, the use of BIAS in a multi-Bluetooth and WLAN environment leads to performance improvements for WLAN, while it has little benefits on the Bluetooth performance.

Table 7
Bluetooth packet loss probability for experiment 4.

BT traffic		WLAN traffic			
		FTP		HTTP	
		BIAS	RR	BIAS	RR
FTP	$d_B = 1$ m	0.0103	0.0158	0.0064	0.0356
	$d_B = 2$ m	0.1079	0.1210	0.0379	0.0393
HTTP	$d_B = 1$ m	0.0012	0.0034	0.0003	0.0002
	$d_B = 2$ m	0.0425	0.0614	0.0265	0.0071

Table 8
Bluetooth MAC delay (sec) for experiment 4.

BT traffic		WLAN traffic			
		FTP		HTTP	
		BIAS	RR	BIAS	RR
FTP	$d_B = 1$ m	0.1805	0.1749	0.1912	0.1739
	$d_B = 2$ m	0.3753	0.4574	0.2444	0.2378
HTTP	$d_B = 1$ m	0.0840	0.0861	0.0836	0.0835
	$d_B = 2$ m	0.0945	0.1121	0.0963	0.0952

Table 9
WLAN probability of packet loss for experiment 4.

BT traffic	WLAN traffic			
	FTP		HTTP	
	BIAS	RR	BIAS	RR
FTP	0.1534	0.303	0.2510	0.3481
HTTP	0.0192	0.0961	0.0721	0.1534

Table 10
WLAN MAC delay (sec) for experiment 4.

BT traffic	WLAN traffic			
	FTP		HTTP	
	BIAS	RR	BIAS	RR
FTP	0.0017	0.0022	0.0010	0.0011
HTTP	0.0011	0.0018	0.0009	0.0012

5. Concluding remarks

In this paper we propose a scheduling technique, BIAS, aimed at eliminating interference on WLAN and alleviating the impact of interference on the Bluetooth performance. This work addresses the need to adjust to changes in the environment, support asymmetric traffic in the upstream and downstream, in addition to the use of different scheduling priorities.

Table 11
BIAS pseudocode.

```
1:  Every N slots
2:     estimate_channel();
3:     compute_credits();
4:  Every even TS_f                                   // Master transmission slot
5:     if TS_f + l_dn is clear                        // Master can receive in next slot
6:        {
7:        A^f_data = {set of high priority slaves s.t. ((f "good") and (qsize > 0) and (c_dn > 0)}
8:        A^f_poll = {set of high priority slaves s.t. ((f "good") and (c_up > 0))}
9:        B^f_data = {set of low priority slaves s.t. ((f "good") and (qsize > 0))}
10:       B^f_poll = {set of low priority slaves s.t. ((f "good") and (c_up · c_dn > 0))}
11:       // Service high priority slaves first
12:       if (A^f_data ≠ ∅)                           // transmit data packets
13:          {
14:          i = max_{A^f_data} (w^i_up · w^i_dn)     // select device i with the largest weight
15:          transmit data packet of size l_dn to slave i
16:          c^i_dn,up = c^i_dn,up − l^i_dn,up;       // decrement credit counter
17:          w^i_dn,up = (1 − u_i) · c^i_dn,up;       // update weights
18:          }
19:       else if (A^f_poll ≠ ∅)                      // transmit polls
20:          {
21:          i = max_{A^f_poll} (w^i_up)              // select device i with the largest weight
22:          transmit poll to slave i
23:          c^i_up = c^i_up − l^i_up;                // decrement credit counter
24:          w^i_up = (1 − u_i) · c^i_up;             // update weights
25:          }
26:       // Then service low priority slaves
27:       else if (B^f_data ≠ ∅)
28:          {
29:          i = max_{B^f_data} (w^i_up · w^i_dn)     // select device i with the largest weight
30:          transmit data packet of size l_dn to slave i
31:          if (c^i_dn > 0) c^i_dn = c^i_dn − l^i_dn; // decrement credit counter
32:          else c^i_up = c^i_up − l^i_dn;           // decrement credit counter
33:          w^i_dn,up = (1 − u_i) · c^i_dn,up;       // update weights
34:          }
35:       else if (B^f_poll ≠ ∅)                      // transmit polls
36:          {
37:          i = max_{B^f_poll} (w^i_up)              // select device i with the largest weight
38:          transmit poll to slave i
39:          if (c_up > 0) c^i_up = c^i_up − l^i_up;  // decrement credit counter
40:          else c^i_dn = c^i_dn − l^i_up;           // decrement credit counter
41:          w^i_dn,up = (1 − u_i) · c^i_dn,up;       // update weights
42:          }
43:       }
```

The performance results obtained are summarized as follows. First, BIAS eliminates packet loss even in the worst interference case when more than $3/4$ of the spectrum are occupied by other devices. Delay is slightly increased over the reference scenario (when no interference is present). This increase varies between 1 to 5 ms on average. Furthermore, BIAS is able to rapidly adjusts to changes in the channel. The channel estimation transient time can be as low as 1.5 ms and 250 μs for DH5 and DH1 packets, respectively. In addition, BIAS supports QOS and maintains a low access delay for delay-sensitive traffic such as video applications. Finally, we observe that the use of BIAS is not as effective to mitigate interference caused by other Bluetooth piconets. In this case, we note no improvements in access delay and packet loss results, which are comparable to results obtained with Round Robin (RR).

An immediate next step for our work consists of developing a channel estimation procedure that is able to differentiate between different types of interference, namely, WLAN and Bluetooth interference. Our preliminary results indicate that this may be helpful in a multi-Bluetooth and WLAN environment.

Acknowledgements

The author would like to thank O. Rebala and A. Tonnerre for their help in developing the simulation models and compiling the results.

References

[1] Bluetooth Special Interest Group, Specifications of the Bluetooth System, Vol. 1, v. 1.0B "Core" and Vol. 2, v. 1.0B "Profiles" (December 1999).

[2] A. El-Hoiydi, Interference between Bluetooth networks – upper bound on the packet error rate, IEEE Communications Letters 5 (June 2001) 245–247.

[3] D. Fumolari, Link performance of an embedded Bluetooth personal area network, in: *Proceedings of IEEE ICC'01*, Vol. 8, Helsinki, Finland (June 2001) pp. 2573–2577.

[4] N. Golmie, N. Chevrollier and I. Elbakkouri, Interference aware Bluetooth packet scheduling, in: *Proceedings of GLOBECOM'01*, Vol. 5, San Antonio, TX (November 2001) pp. 2857–2863.

[5] N. Golmie and F. Mouveaux, Interference in the 2.4 GHz ISM band: Impact on the Bluetooth access control performance, in: *Proceedings of IEEE ICC'01*, Vol. 8, Helsinki, Finland (June 2001) pp. 2540–2545.

[6] N. Golmie, R.E. Van Dyck and A. Soltanian, Interference of Bluetooth and IEEE 802.11: Simulation modeling and performance evaluation, in: *Proceedings of the Fourth ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, MSWIM'01*, Rome, Italy (July 2001) pp. 11–18. Extended version appeared in ACM Wireless Networks 9(3) (2003) 201–211.

[7] I. Howitt, V. Mitter and J. Gutierrez, Empirical study for IEEE 802.11 and Bluetooth interoperability, in: *Proceedings of IEEE Vehicular Technology Conference (VTC)*, Vol. 2 (Spring 2001) pp. 1103–1113.

[8] IEEE Std. 802-11, IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification (June 1997).

[9] J. Lansford, R. Nevo and E. Zehavi, MEHTA: A method for coexistence between co-located 802.11b and Bluetooth systems, IEEE P802.11 Working Group Contribution, IEEE P802.15-00/360r0 (November 2000).

[10] S. Shellhammer, Packet error rate of an IEEE 802.11 WLAN in the presence of Bluetooth, IEEE P802.15 Working Group Contribution, IEEE P802.15-00/133r0, Seattle, WA (May 2000).

[11] B. Treister, A. Batra, K.C. Chen and O. Eliezer, Adapative frequency hopping: A non-collaborative coexistence mechanism, IEEE P802.11 Working Group Contribution, IEEE P802.15-01/252r0, Orlando, FL (May 2001).

Nada Golmie received the M.S.E. degree in computer engineering from Syracuse University, Syracuse, NY, and the Ph.D. degree in computer science from the University of Maryland, College Park, MD. Since 1993, she has been a member of the Advanced Network Technologies Division of the National Institute of Standards and Technology (NIST). Her research in traffic management and flow control led to several papers presented at professional conferences, journals and numerous contributions to international standard organizations and industry led consortia. Her current work is focused on the performance evaluation of protocols for Wireless Personal Area Networks. Her research interests include modeling and performance analysis of network protocols, media access control, and quality of service for IP and wireless network technologies. She is the vice-chair of the IEEE 802.15 Coexistence Task Group.
E-mail: nada.golmie@nist.gov