This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2007 proceedings.

RIA: An RF Interference Avoidance Algorithm for Heterogeneous Wireless Networks

Manoj Pandey, Daniel Delorey, Qiuyi Duan,
Lei Wang, Charles Knutson and Daniel Zappala
Computer Science Department
Brigham Young University
Provo, Utah 84602
{manoj, pierce, qiuyi, lei, knutson, zappala}@cs.byu.edu

Ryan Woodings
MetaGeek, LLC
5465 Terra Linda Way Suite 96
Nampa, ID 83687
ryan@metageek.net

***Abstract*— Devices with multiple wireless interfaces are becoming increasingly popular. We envision that these devices will become the building block for future mesh networks, providing seamless connectivity across a range of heterogeneous devices. Although these devices typically implement frequency sharing, using either Direct Sequence Spread Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS), they may still interfere with one another. In this paper we provide a novel Radio Interference Avoidance (RIA) algorithm that solves the problem of interference between IEEE 802.11 and Bluetooth. We then extend this algorithm to other types of DSSS and FHSS combinations. Though the algorithm is limited to devices with both of these interfaces, this is a very common case. We analytically derive the expected value of the response time for RIA and run simulations to demonstrate its effectiveness. Our results indicate that RIA is able to eliminate interference with a very short response time. RIA also outperforms Adaptive Frequency Hopping, a solution proposed by the IEEE 802.15 Co-existence Working Group.**

I. Introduction

It is becoming increasingly common to have several co-located devices, each having multiple wireless technologies. This paradigm is supported by both industry and research. Today, many of the portable devices shipped, such as PDAs and laptop computers, are equipped with multiple wireless interfaces (for IEEE 802.11, Bluetooth, IrDA etc.). Research projects including BARWAN [13] and MosquitoNet [3] further extend this vision. These projects solve the problem of providing a route between two devices, separated by a network with different wireless technologies. These devices have multiple interfaces and switch dynamically to the best available technology from a set of all available wireless technologies.

Many of today's wireless technologies operate in the 2.4 GHz ISM band. The unlicensed nature of the band, combined with the propagation qualities of the frequencies it covers, makes it desirable for many short range wireless solutions. These technologies can be categorized into two types of spread spectrum: frequency hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS). In FHSS, devices hop across a set of frequencies, transmitting on each frequency for a predetermined length of time. Examples of FHSS devices include Bluetooth, 802.11b, HomeRF, and Cordless Phones. In DSSS, devices spread their transmissions over a set of frequencies, transmitting on all frequencies simultaneously. Examples of DSSS devices include 802.11b,g, ZigBee, WirelessUSB, and Cordless Phones.

Multiple interfaces located in the ISM band can cause interference to each other. Interference affects data transfer significantly by reducing throughput and by increasing latency; this is confirmed by our simulation results, which we discuss later. To the best of our knowledge, recent projects, including BARWAN [13] and MosquitoNet [3], do not address the problem of handling interference among these technologies.

In this paper, we address this problem in two steps. We begin by providing a novel algorithm, RIA, to solve the interference problem between Bluetooth and IEEE 802.11. RIA is not only efficient, but also has a faster response time. Next, we extend RIA to provide insight into solving the problem of interference between other DSSS and FHSS combinations. Our solution works in the common case when devices experiencing interference have interfaces for both technologies. This is a valid assumption, since most of the wireless devices today have multiple wireless interfaces including 802.11, Bluetooth, IrDA, and WirelessUSB.

RIA extends our Quality of Transport architecture (QoT) proposed by Knutson et al. [10]. QoT allows multiplexing when multiple transport interfaces, such as IrDA, Bluetooth, and 802.11 are available on a single device. The goal of QoT is to transparently and automatically use the best technology for data transfer as devices move in an out of range in a heterogeneous wireless network. RIA makes QoT more robust by letting QoT avoid interference between wireless technologies.

We validate our solution primarily using simulations, where we study Bluetooth and 802.11 interference avoidance using RIA. In this way we are able to compare our solution to other proposals for Bluetooth and 802.11 interference avoidance. Our simulations show that RIA outperforms the Adaptive Frequency Hopping (AFH) mechanism significantly in terms of achievable throughput and response time. Furthermore, for most of these simulations, we also vary channel width of DSSS systems, which is one of the major properties of DSSS systems. By doing so we gain an insight into how RIA handles DSSS systems other than IEEE 802.11.

1525-3511/07/$25.00 ©2007 IEEE

II. Related Work – Interference Between 802.11 and Bluetooth

Most interference avoidance studies have focused on 802.11 and Bluetooth. Proposed solutions can be broadly classified as collaborative or non-collaborative. Collaborative solutions require shared awareness and cooperation between the interfering technologies. Non-collaborative solutions are generally preferred because they require no coordination between existing technologies or newly developed technologies. Our proposed algorithm, RIA, is a non-collaborative solution. Unlike most solutions, in addition to solving the interference between Bluetooth and 802.11, RIA can also be easily extended to most of the DSSS system.

Several collaborative mechanism have been proposed to solve Bluetooth and 802.11 interference. Shellhammer proposed a mechanism for devices with both 802.11 and Bluetooth interfaces [12] in which 802.11 devices schedule traffic for both Bluetooth and 802.11. RTS and CTS mechanisms are used to clear the channel for Bluetooth if it hops into a frequency used by the 802.11 interface. This proposal pushes complexity into the 802.11 device, requiring it to know about Bluetooth hopping behaviors. It is not easily extensible when new technologies are introduced into the ISM band. Van Dyck and Soltanian propose a simpler collaborative method for collocated interfaces [6]. In this method the 802.11 interface inserts a null value into part of the direct sequence spectrum whenever the Bluetooth interface transmits within the DSSS band. While simpler than the previous solution, this still requires a significant amount of complexity in the 802.11 device and is not easily extensible.

We describe two of the important non-collaborative solutions proposed. The first is AFH, which suggests that Bluetooth detect poor channels and adjust its hopping sequence dynamically [2]. While it does describe how to add AFH to Bluetooth without having to change the existing frequency selection kernel, it is still fairly incomplete. For example, It does not explain how the quality of a channel would be determined. It provides some suggestions, but it does not indicate which is preferred for AFH. Similar to AFH, Golmie proposed a non-collaborative MAC solution [7]. The proposed architecture requires Bluetooth to record used channels and avoid them. A used channel is one in which the Bluetooth device experiences a significant bit error rate over time. However, as compared to RIA, both these approaches are slow to respond to interference.

More recently, Chiasserini et al. [4] proposed a method that can be used in both collaborative and non-collaborative environments. Their method proposes that Bluetooth devices sense channel conditions and adjust packet scheduling accordingly to avoid interference. Nevertheless, RIA is able to achieve good performance with a simpler algorithm.

III. Radio Frequency Interference Avoidance algorithm: RIA

We briefly describe the RIA algorithm, provide a bound on its response time, compare its response time with AFH, and discuss its power implications. For a more extensive discussion of these topics, please see our extended report [11].

A. Algorithm Description

RIA uses a $CollisionTable$ to track the number of collisions that occur. This data structure is kept in the QoT Session layer [10], and entries are expired after some length of time. For each collision that occurs at the FHSS interface, RIA appends a new record to $CollisionTable$, which stores the frequency at which collision occurs. After appending a new entry, RIA counts the number of the recent records; if the count exceeds λ, RIA invokes its interference avoidance mechanism. RIA uses the distribution of frequencies of the recent λ collisions to find a channel, ξ, such that its midpoint is closest to the mean, ξ_{est}, of the λ frequencies. Formally,

$$\xi = \{\xi_o : |mid(\xi_o) - \xi_{est}| < |mid(\xi_j) - \xi_{est}|; o \neq j\}$$

Thus, RIA allows the FHSS device to avoid entire DSSS channel without having to experience collisions on every frequency in that channel. This reduces the response time for interference avoidance.

RIA listens on each channel for a sampling period of τ_{sample}. While listening on channel ξ, reception of a valid DSSS frame confirms that DSSS is active on channel ξ. Hence, for FHSS interface, RIA blocks all the frequencies Π_ξ, which belong to the DSS channel ξ. This is done by taking intersection of the set of usable frequencies, Π and the complement of π_ξ. Formally,

$$\Pi = \Pi \cap \pi_\xi^c$$

RIA blocks channels on a *soft state* basis. After a certain time interval, frequencies belonging to a blocked channel are released. This is because due to mobility, the DSSS source might have moved away and the blocked channel should be released. If the DSSS source did not move way, RIA would be invoked and the channel will be blocked again.

Besides blocking the frequencies from the usable list of frequencies, RIA also drops those records from $CollisionTable$, which contain frequencies belonging to the set π_ξ. This is because records of collisions caused by the previously identified DSSS device may adversely affect subsequent estimations.

If no activity is detected on DSSS channel ξ, the channel is expanded outwards as long as the maximum (ξ_{max}) and minimum (ξ_{min}) channels are not reached. Formally,

$$\xi \ \pm = 1$$

as long as $\xi_{min} < \xi < \xi_{max}$

There are two reasons why RIA might fail to discover the the interfering DSSS source. First, the source of interference may not have the DSSS technology that RIA invokes. This is because, as per our assumption, RIA only identifies those DSSS technologies, which are installed on the device. Second, the source of interference might not transmit any frames during the channel sampling period τ_{sample}. In the latter case, we would observe interference again and hence reinvoke RIA.

B. Comparing RIA's Response Time and AFH

In this section, we estimate the advantage that Bluetooth could gain if it were to use RIA instead of AFH. We include our result derived in [11] to demonstrate the superiority of RIA over AFH. Our analysis parameterizes the interference model so that it may be applied to any DSSS and FHSS technology combination. Thus, it helps us demonstrate the general purpose nature of our solution.

For Bluetooth, if $E(t_{AFH})$ and $E(t_{RIA})$ are expected values of time needed to detect and eliminate interference from a generic DSSS source, for AFH and RIA respectively, then:

$$\frac{E(t_{AFH})}{E(t_{RIA})} \approx \frac{\omega_{dsss} * H_{\omega_{dsss}}}{\lambda} \quad (1)$$

where, H_n is the sum of harmonic series, $1, 1/2, 1/3, ..., 1/n$ and ω_{dsss} is the width of the DSSS channel.

Let us now calculate the above ratio, established by Theorem 1, for Bluetooth and IEEE 802.11. For 802.11, $\omega dsss$ equals 22 and $H_{\omega_{dsss}}$ is $19093197/5173168$[1]. Hence

$$\frac{E(t_{AFH})}{E(t_{RIA})} = \frac{81.19}{\lambda} \quad (2)$$

Based on our results in section V, even a small value of λ (like 3) is enough to detect and avoid interference. Using this value, AFH takes $81/3 = 27$ times longer to adapt as compared to RIA. AFH has slower response because as it detects more and more number of frequencies that belong to DSSS, the number of noisy frequencies reduces. This reduction in interference frequencies increases the time needed for the next collision to occur.

C. Power Implications on the FHSS device

Using an FHSS device to invoke RIA might appear to starve the FHSS device. However, the FHSS device invokes RIA only when it is experiencing interference and has a minimum of λ collisions in the collision table. Assuming a usage model of a Bluetooth device traveling in an ad hoc network, the FHSS would need to invoke RIA only once, as all the other 802.11 devices would use the same channel.

There are two reasons that motivate us to allow FHSS device to invoke RIA instead of than the DSSS device. First, even if the DSSS device was to change the channel, it would continue to occupy the same number of frequencies within the spectrum and the FHSS device will continue to register the same degree of interference. This is because FHSS technologies randomly select frequencies from the entire spectrum. Second, given the usage model where 802.11 devices could be in the form of wireless ad hoc networks, changing the channel for one DSSS device would require changing the channel for all the other 802.11 devices in the entire network.

[1]For large n, $H_n = ln(n) + \gamma_{euler} + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \epsilon$, where $\gamma_{euler}(= 0.57721)$ is the *Euler's constant* and $0 < \epsilon < \frac{1}{252n^6}$ [9]

IV. Simulation Methodology

In this section, we present an overview of some of the DSSS/FHSS technologies and our simulation methodology.

A. DSSS Technologies: IEEE 802.11, Zigbee, WirelessUSB

IEEE 802.11 is one of the most popular DSSS technologies being used today [5]. 802.11 uses CSMA/CA (Collision Sense Multiple Access/Collision Avoidance) by using the sequence RTS/CTS/DATA/ACK to send data. 802.11 primarily uses DSSS. The DHSS spectrum for 802.11 consists of multiple channels (eleven in United States) each of which is 22 MHz wide with overlap among consecutive channels.

Besides 802.11, there are several other DSSS technologies. We briefly describe two of them – Zigbee and WirelessUSB. Based on IEEE 802.15.4, ZigBee is aimed towards low data rate wireless personal area networks (WPAN) [1]. Like 802.11, Zigbee uses a CSMA/CA mechanism at the MAC layer. The DSSS channel for ZigBee is 5MHz wide. Developed at Cypress Semiconductor, WirelessUSB is a low power, low data rate, low latency, low cost solution for small range wireless applications. WirelessUSB uses DSSS with 1 MHz wide channels [14].

B. FHSS Technology: Bluetooth

In our simulations, we use Bluetooth as the representative technology for FHSS mechanisms. A Bluetooth network consists of a master node and up to seven slave nodes. A Bluetooth device operates on 79 channels spaced 1 MHz apart starting at 2.4 GHz. Each frame is sent by hopping to different frequencies in a pseudo-random manner as per the FHSS algorithm. The nominal hop rate is 1600 hops/s and hence the slot length, τ_{slot} is 625 μs. For reliability, Bluetooth uses an automatic retransmission request (ARQ) scheme, whereby a lost packet is retransmitted by the sender. In our simulation, we implement retransmission for lost packets.

C. Simulation Methodology

We use GloMoSim-2.03 for our simulations [15]. We extend GloMoSim-2.03 with an abstract model of Bluetooth. Our Bluetooth implementation uses a class 2 Bluetooth FHSS radio with 2.4 mW power with radio sensitivity of -70 dBm. We implement a simple ARQ scheme, whereby a lost packet is retransmitted a given number of times (4 for our implementation). For 802.11 based ad hoc network flows we use the Dynamic Source Routing, (DSR) [8] protocol at the routing layer and 802.11 at the MAC layer. Our DSSS radio uses 6.1 mW power and its sensitivity is -81 dBm.

Our topology contains one piconet comprising of one master and four slaves along with four 802.11 ad hoc nodes (Figure 1). The four 802.11 nodes are within the radio range of each other. All the devices can have RIA, but at the very least, Bluetooth devices should have RIA. We keep two ad hoc 802.11 flows in the same radio range because it creates intra-802.11 channel contention besides contention between 802.11 and Bluetooth. Also the sender of the first flow is kept inside the piconet. With the second flow, the receiver sits inside the

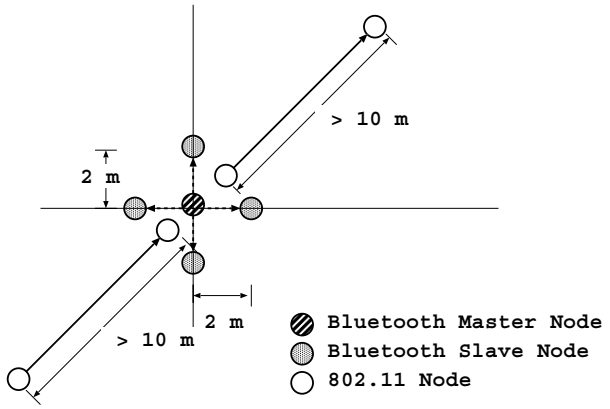

Fig. 1. Field Topology - 5 Bluetooth nodes, 4 802.11 nodes. Bluetooth devices have RIA.

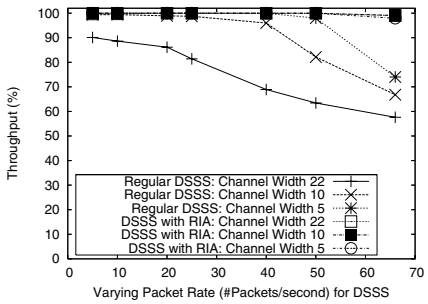

Fig. 2. DSSS Throughput versus DSSS Traffic Rate

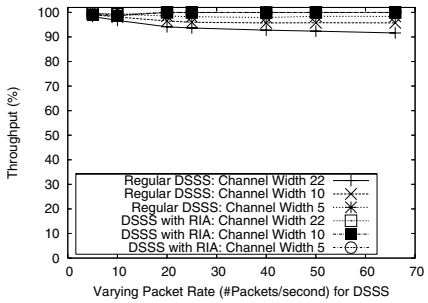

Fig. 3. Bluetooth Throughput versus DSSS Traffic Rate

piconet. It is worth mentioning that even though a receiver may not be sending any data from an "application layer" perspective, it does send CTS and ACK at the MAC layer. Hence, all the participating nodes, whether sender or receiver, transmit frames at the MAC layer. Since our scenarios are static, we do not implement the soft-state nature of RIA. If a channel is blocked during simulation, it remains blocked until the end of the simulation. Interference is generated by varying the traffic rate. The traffic rate is kept less than the capacity of that medium so that if the flows were to run individually, they would each obtain a throughput of 100%. For further simulation details, please refer to [11]

Our evaluation metrics consist of various flow related parameters collected as traffic is varied. These consist of throughput (packet delivery ratio), delay, number of retransmissions and number of collisions. We also vary λ to understand variations in RIA's response time.

V. Results

We perform three set of simulations to demonstrate the effectiveness of RIA. For the first set of simulations we vary the channel width of DSSS systems to understand the likely behavior of RIA on different DSSS systems. For these simulations, we keep the traffic rate for Bluetooth interfaces near its maximum and vary the DSSS traffic rate. The goal of this scenario is that, once RIA is invoked, both flows should achieve their maximum throughput. For the second set we study RIA's response time by varying λ. The third set of simulations compares RIA with the AFH approach.

A. Evaluating RIA by varying DSSS channel width

For the first set of simulations, we vary channel width and data rate for 802.11. From an interference perspective, variation of channel size is one of the most important characteristics of DSSS system. For DSSS, a large channel width not only makes it more vulnerable to interference but also generates more interference for other devices. A larger channel width provides higher sending data rate and hence we vary the data rate as well. Even though various DSSS technologies use different protocol stacks, varying the channel width along with the data rate allows us see the likely effect of RIA on other DSSS technologies. We use three different DSSS channel widths – 22, 10, and 5. Channel widths 22 and 5 represent IEEE 802.11 and Zigbee respectively. A channel width of 10 is used to increase the granularity of variation of channel widths. In this project, one of our future work is to explore effects of RIA on more accurate implementations of other DSSS technologies.

1) Throughput and Delay: Without RIA, both DSSS and FHSS flows suffer in terms of throughput (Figure 2). DSSS flows with larger channel width suffer more, since this increases overlap of frequencies, leading to more interference. Thus, as compared to 802.11, ZigBee is affected less by Bluetooth. This presents an interesting cautionary note. Although increasing DSSS bandwidth leads to higher data rates, sufficient care must be given either to increasing the robustness of the data transferred or to avoiding interference itself.

Throughput for Bluetooth is better than that of DSSS flows (Figure 3). This is because the frequency hopping nature of Bluetooth makes it more robust to interference than DSSS systems. DSSS systems occupy the same channel and upon seeing a collision, simply backoff. On the other hand, Bluetooth simply uses the next frequency in sequence to retransmit the packet, when collisions occur.

Besides throughput, RIA helps both 802.11 and Bluetooth to reduce delay. The results for this can be found in our technical report [11]. A reduced delay is better, since often these mobile devices (especially Bluetooth) are used for voice communications, which are delay sensitive in nature.

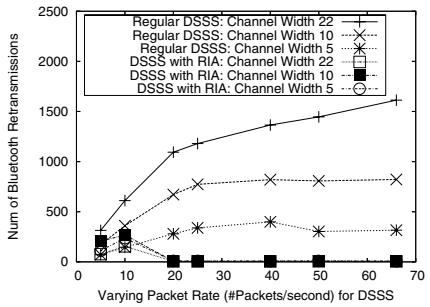

Fig. 4. Bluetooth Retransmissions versus DSSS Traffic Rate

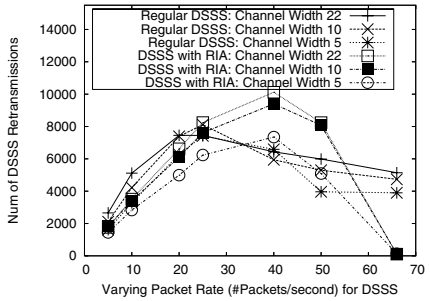

Fig. 5. DSSS Retransmissions versus DSSS Traffic Rate

2) Number of Retransmissions: Both DSSS and FHSS use retransmissions to provide reliability when a frame is lost. In Figure 5, we plot the number of retransmissions for both technologies as DSSS traffic is increased. Achieving high throughput is desirable, but it should be done by using the least number of retransmissions.

RIA reduces the number of retransmissions for Bluetooth significantly (Figure 4). This is valuable from the perspective of power consumption, since large numbers of retransmissions would increase power consumption. Bluetooth is often criticized for having a heavier protocol stack, which leads to higher power consumption; in cases of interference, RIA can help Bluetooth conserve its battery.

Surprisingly, for DSSS systems the number of retransmissions seem to remain identical with or without RIA (Figure 5). In order to understand this behavior, we provide a breakup of retransmissions (Figure 6). In 802.11, retransmission occurs due to two reasons – when a CTS is not received after having

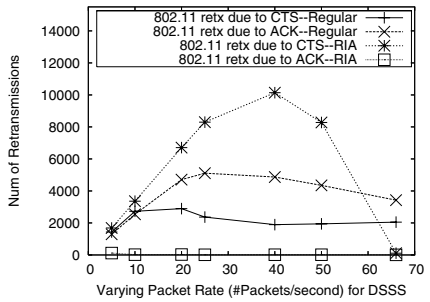

Fig. 6. Breakup of Number of Retransmissions for 802.11

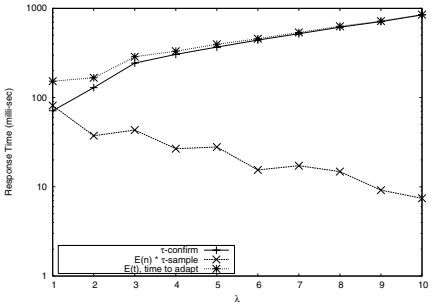

Fig. 7. Response time of RIA versus λ

sent an RTS or when an ACK is not received after having sent the DATA. Since both these reasons contribute to total retransmissions, we analyze retransmission with respect to these two components. Without RIA there is more backoff and hence sending an RTS is relatively easy. However due to a busy medium, more packets are lost, leading to a larger number of retransmissions due to loss of ACK frames. Absence of ACKs can be either due to lost DATA at the receiver or due to a lost ACK at the sender. When the medium is busy, both of the above can happen. On the other hand, RIA frees the medium from interference and reduces backoff. As a result, more RTS packets are sent. This leads to increased collisions at the CTS level due to another nearby 802.11 flow. This is a welcome sign since it reflects that the medium has less backoff and more and more attempts are being made to transmit frames. If a CTS is sent, channel reservation is honored, which is demonstrated by the lesser number of retransmissions due to lost ACK. In other words, RIA increases the probability of a successful reception of data.

B. Response Time for RIA

For the second set of simulations, we study RIA's response time. One of the strengths of RIA is its fast response time, since it avoids an entire DSSS channel rather than avoiding individual frequencies used by the DSSS channel. For this section we use the same topology as that of Figure 1, but keep the sending rate for both 802.11 and Bluetooth close to the maximum (1400 kbps for 802.11 and 360 kbps for Bluetooth).

Figure 7 shows variation of $\tau_{confirm}$ and $E(n) * \tau_{sample}$ with λ. $\tau_{confirm}$ is the time needed to acquire λ collisions (and hence confirms that it is time to invoke RIA), increases when λ is increased. With a larger number of λ collisions, the probability that the initial guess of the DSSS channel would be accurate increases. Hence $E(n) * \tau_{sample}$, which is the time needed to discover the DSSS channel number, decreases with increasing λ. With smaller values of λ, the initial guess is less accurate and RIA needs to listen on more channels before arriving at the correct one; this increases the value of $E(n) * \tau_{sample}$. Fortunately, a value of 3 for λ, seems to be a sufficiently satisfactory.

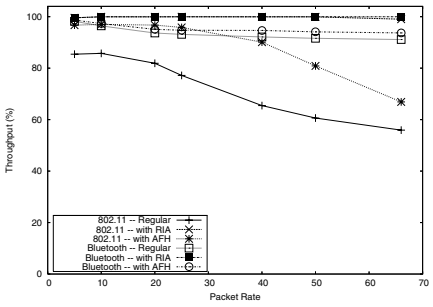

Fig. 8. Comparison of RIA and AFH

C. Comparing RIA with AFH

In our third set of simulations, we compare RIA to AFH. We chose AFH since it is one of the intuitive solutions for a frequency hopping system – detect and avoid individual frequencies that are noisy ([2],[7]). For these simulations we vary the traffic rate of 802.11 and keep the DSSS channel width to 22 (which represents 802.11). We compare Bluetooth and 802.11 with RIA, AFH, and without either one of these.

As shown in Figure 8, RIA is able to outperform AFH. This is due to a simple reason, as explained in III-B, AFH takes longer (27 times longer) to detect all the noisy frequencies that are being used by the current 802.11 channel. On the other hand, RIA's fast response allows it to guess the channel early. Upon confirmation it avoids the entire channel altogether, leading to better throughput.

VI. Conclusion and Future Work

In this paper, we designed and evaluated an RF interference avoidance algorithm for DSSS and FHSS radios. We began by solving the more common case of interference between Bluetooth and 802.11. In the next step, we extended this solution to a common case when the DSSS and FHSS technologies (that interfere with each other) are installed on the same device.

Using both simulation and analysis, we demonstrate the superiority of RIA. We present an analysis for RIA's expected time to detect interference and avoid interference. Our simulation results indicate that RIA can help DSSS and FHSS co-exist with each other by eliminating mutual interference. In our comparison with AFH, RIA is able to outperform it due to its ability to detect DSSS sources much more quickly. RIA is also capable of helping FHSS avoid interference from multiple nearby DSSS sources.

For our future work, we are planning to make RIA adapt to the traffic rate. When traffic rates are lower, RIA needs to be adjusted so that it can increase the τ_{sample} (currently 40 ms) to a higher value. This is required since with lower traffic rate, RIA could listen on a channel and then move to the next channel without hearing any valid frames on the correct channel. This would lead RIA to believe that the source of interference is not sitting on the correct DSSS channel. Though, after a short period of time, due to soft-state nature of RIA, it would reinvoke channel sampling algorithm, this can unnecessarily add to the latency of the algorithm. Likewise, when traffic rate increases, a smaller value of τ_{sample} should be used. We would also like to demonstrate that RIA is effective in solving interference even when there are multiple DSSS sources in the neighborhood.

Next, we are exploring the use of RIA to help distinguish the correct reason for a packet loss. In mobile wireless devices with random topologies, packets can be lost due to interference, besides mobility and congestion. It is essential that the true nature of loss is identified before reacting to the loss. For example, reacting to an interference loss by considering it as a mobility loss could be harmful. the device will wrongly keep sending mobility based link repair messages in the network as long as the interference continues.

Finally, we plan to extend RIA to avoid interference even from unknown DSSS sources. Instead of a valid DSSS frame, RIA can use *Received Signal Strength Indication* (RSSI) measurements provided by the physical layer instead of reception of a valid frame. These mechanisms can use RSSI values that can be returned by the radio layer. The radio layer can keep track of the signal being received along with the background noise. Using RSSI values would not require any additional hardware besides what is available for the standard DSSS interface, such as 802.11.

References

[1] Zigbee Alliance. http://www.zigbee.org.

[2] A. Batra et. al. B. Treister et. al., KC Chen et. al. Clause 14.3 adaptive frequency hopping. IEEE P802.15 Working Group Contribution, IEEE P802.15-TG2_366r1, April 2005.

[3] M. Baker, X. Zhao, and J. Stone. Supporting mobility in mosquitonet. In *USENIX Technical Conference*, 1996.

[4] C. Chiasserini and R.R. Rao. Coexistence mechanisms for interference mitigation in the 2.4-ghz ism band. *IEEE Transactions on Wireless Communications*, 2(5):964–275, September 2003.

[5] B.P. Crow, I. Widjaja, J.G. Kim, and P.T. Sakai. IEEE 802.11 Wireless Local Area Networks. *IEEE Communications Magazine, pp. 116–126*, September 1997.

[6] R. E. Van Dyck and A. Soltanian. Collaborative co-located coexistence mechanism. IEEE P802.15 Working Group Contribution, July 2001.

[7] N. Golmie. Non-collaborative mac mechanisms. IEEE P802.15 Working Group Contribution, IEEE P802.15-01/316r0, June 2001.

[8] D.B. Johnson and D.A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[9] D.E. Knuth. *The Art of Computer Programming. Vol 1.* Addison-Wesley Publishing House, 1997.

[10] C.D. Knutson, R.W. Woodings, S.B. Barnes, H.R. Duffin, and J.M. Brown. Dynamic autonomous transport selection in heterogeneous wireless environments. In *IEEE Wireless Communication and Networking Conference (WCNC)*, 2004.

[11] M. Pandey, D. Delorey, L. Wang Q. Duan, C. Knutson, D. Zappala, and R. Woodings. RIA: An RF Interference Avoidance Algorithm for Heterogeneous Wireless Networks. Technical Report TR-BYU-ILAB-2006-01, Internet Research Lab, BYU CS Department, Feb 2006.

[12] S. Shellhammer. Ieee 802.15.2 clause 14.1 - collaborative coexistence mechanism. IEEE P802.15 Working Group Contribution, IEEE P802.15-01/340r0, July 2001.

[13] M. Stemm and R. H. Katz. Vertical handoffs in wireless overlay networks. *Mobile Networks and Applications*, 1999.

[14] R. Woodings and M. Pandey. WirelessUSB: A Low Power, Low Latency and Interference Immune Wireless Standard. In *IEEE WCNC, Las Vegas*, April 2006.

[15] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Workshop on Parallel and Distributed Simulation*, May 1998.