Comparative Study on AFH Techniques in Different Interference Environments

Bo-Zheng Pang[1], Tim Claeys[2], Davy Pissoort[2], Hans Hallez[1] and Jeroen Boydens[1]
[1]Department of Computer Science, M-Group, KU Leuven Bruges Campus
[2]Department of Electrical Engineering, M-Group, KU Leuven Bruges Campus
Spoorwegstraat 12, 8200 Bruges, Belgium
{Bozheng.Pang | Tim.Claeys | Davy.Pissoort | Hans.Hallez | Jeroen.Boydens}@kuleuven.be

Abstract – **In this paper the use of Adaptive Frequency Hopping (AFH) as a solution to interference problems caused by the proximity and simultaneous operation of radio systems in the 2.4 GHz band is discussed. The main algorithms for AFH that attempt to avoid frequency collisions are considered. A comparative analysis of their respective performance is conducted. The trends and trade-offs for different interference levels are discussed. Performance is analyzed in terms of collision frequency and channel usage distribution frequency.**

Keywords – **Bluetooth Low Energy, Frequency Hopping, Channel Selection, Interference, Robustness**

I. Introduction

In recent years, the concept of IoT (Internet of Things) has become more and more popular. It is expected to be a revolutionary update for information transfer in multiple dimensions. Smart devices would be allowed to connect, transfer and even make autonomous decisions on behalf of people. This new technology is called "connectivity for anything"[1], which is defined as to be able to connect anywhere, anytime and anything.

In order to achieve the connectivity for anything, wireless communication is a must for IoT systems. This will introduce a large number of wireless communication protocols belonging to different types, like WLAN and WPAN, with typical representatives Wi-Fi and Bluetooth/Bluetooth Low Energy [2]. Using different wireless communication protocols will cause interference problems in the same frequency band. With the increase of wireless communication devices, the management for frequency and time has become a priority. Coexistence of extensive wireless devices demands management in high quality and effectiveness [2].

Bluetooth Low Energy (BLE) is included in the Bluetooth standard since version 4.0. It operates as a short-range wireless system in the 2.4 GHz band. This band is shared by various types of radio systems, like Wi-Fi and ZigBee [3]. The sharing of frequency band makes it difficult to guarantee performance of devices across different technologies. Therefore, it is of vital importance to study their coexistence by considering mutual and cross-technology interferences. BLE employs AFH to improve its coexistence with other protocols. It hops over 40 channels to mitigate interference from other radio systems [4]. The channel decision algorithms used in standard BLE are CSA #1 (Channel Selection Algorithm #1) and CSA #2 (Channel Selection Algorithm #2), both of which help increase the performance of AFH [5]. Other frequency hopping algorithms, as shown in [6-8], developed by researchers from related domains also exist.

This paper makes a comparison between CSA #1 and #2 in different interferences. It shows how both algorithms react to different environments. Our results will provide more information to help developers make a decision between CSA #1 and #2. This paper will offer some ideas to help for the future improvement of algorithms.

The paper is organized as follows. Section II presents the channel selection algorithms in AFH technology in BLE. Section III illustrates the experimental setup for the algorithm simulation. In Section IV, the results of the simulation experiments are demonstrated and analyzed, and some discussion about them are performed. Section V concludes the paper. Section VI proposes future work.

II. Adaptive Frequency Hopping

In this section, two different AFH techniques, in use by BLE, are explained from a software algorithm point of view. Their results will be compared in section III.

A. Basic Concept and Logic

In BLE systems, the devices are generally divided into two types. These two types are the Master (Central) and Slave (Peripheral). BLE uses a total of 40 channels for selection between each pair of communication devices. These 40 channels include 37 data channels and 3 advertising channels. Master devices scan for other devices, and are usually smartphones and personal computers. Slaves advertise and wait for connections from Masters on the 3 advertising channels.

The master of the communication shall classify data channels into used and unused channels [9], which is called a channel map. The minimum number of used channels inside a channel map should be 2. This effectively prevents algorithm failure at the beginning of communication. Also, a master and a slave in the same communication process shall share the same channel map updated only by the master.

The basic logic for the two AFH algorithms is exactly same, as shown in Figure 1. Both algorithms need 4 inputs, varying on the algorithm. Input 3 is calculated from input 1 & 2. Input 4 is the number of channels classified as used channels.

This work was supported by VLAIO-Flanders (Technology Transfer / CORNET project "EEWISE, EMC for Emergent Wireless Interconnected Systems").

```
1. Input: Input1, Input2, Input3, Input4
2. unmappedChannel=Basic Algorithm (Input1, Input2)
3. IF unmappedChannel is a used channel:
4.     Channel Index=unmappedChannel
5. ELSE:
6.     remappingIndex=Remapping Algorithm (Input3, Input4)
7.     Channel Index=Channel Map [remappingIndex]
8. END IF
```

Figure 1. Basic Logic of CSA

B. Channel Selection Algorithm #1

CSA #1 can only be used in connection events for BLE for channel selection. The algorithm includes 2 stages for the channel selection, which are the unmapped channel index calculation and mapping this index to a data physical channel index from the set of used channels [9].

The basic idea of this algorithm is shown in Figure 1. The Input 1 & 2 here are `lastUnmappedChannel` and `hopIncrement`. The Input 3 and 4 here are `unmappedChannel` and `numUsedChannels`. The Basic Algorithm here is:

$$unmappedChannel = (lastUnmappedChannel + hopIncrement)\ mod\ 37$$

In the equation, operand 37 is used instead of 40. Because the 3 advertising channels are excluded. The `unmappedChannel` and `lastUnmappedChannel` are the unmapped channel indices of two consecutive connection events. When a connection event closes, the `lastUnmappedChannel` shall be set to the value of the `unmappedChannel`.

The Remapping Algorithm here is:

$$remappingIndex = unmappedChannel\ mod\ numUsedChannels$$

Where `numUsedChannels` is the number of used channels in the channel map. Then `remappingIndex` would be used as an index into the remapping table to obtain the channel index for the event [9].

C. Channel Selection Algorithm #2

This algorithm is named CSA #2. It can be used in both connection events and periodic advertising events of BLE. The algorithm includes 2 primary calculation modules, which are Unmapped Event Channel Selection and Event Mapping to Used Channel Index [9].

The basic idea of this algorithm is shown in Figure 1. But, in comparison to CSA #1, CSA #2 needs some other inputs or basic components, like a 6-bit number of channels classified as used channels, a 16-bit `counter` that changes for each event, and a 16-bit `channelIdentifier` that is fixed for any given connection or periodic advertising train. The 16-bit `counter` and `channelIdentifier` are Input 1 & 2 respectively. The `channelIdentifier` is calculated from the Access Address by:

$$channelIdentifier = (Access\ Address_{31-16})\ XOR\ (Access\ Address_{15-0}),$$

where Access Address is a 4-byte value used to identify the radio communication on the physical link.

The Basic Algorithm here is called Unmapped Event Channel Selection. It is a process to generate `unmappedChannel`. Two stages are inside this process. They are generating a pseudo random number `prn_e` and computing the `unmappedChannel` by `prn_e`. The `unmappedChannel` is calculated as `prn_e` modulo 37. It would provide more unpredictability for `unmappedChannel` or channel selection.

The Remapping Algorithm here is called Event Mapping to Used Channel Index. It is used when the calculated `unmappedChannel` is busy/unused. The Input 3 & 4 are used here, and they are called `prn_e` and `N` respectively. By pseudo random number `prn_e` and the number of used channels `N`, `remappingIndex` shall be calculated as follows:

$$remappingIndex = Floor\ (\frac{N * prn\_e}{2^{16}})$$

Then `remappingIndex` would be used as an index into the remapping table to obtain the channel index for the event [9].

III. Experimental Setup

Both channel selection algorithms are implemented in GNU Octave [10]. They are asked to perform 100000 connection events in 3 environments. The 3 different environments represent 1 normal environment and 2 extreme environments in reality. This section first describes the interference generation process and deployment used in our simulation experiments. Next, parametric analyses for comparison between these 2 algorithms are discussed.

A. Interference Generation Process

Based on the uncertainty of interferences in reality, a uniform distribution function is used generating a channel map to simulate a random communication environment for multiple devices with frequency hopping algorithms.

A total of 3 different communication environments are used in the simulation experiments.

The first one is a medium communication environment. It is based on a uniform distribution function. There are about 11 to 27 channels available in this environment. This would provide interference characteristics of a normal number of wireless communication devices. But it lacks demonstration for extreme communication environments.

The other two simulate extreme communication environments. They are high-interference environment and low-interference environment. High-interference environment means there are interferences in most channels. Because a large number of communication devices are in the same communication area. There are about 2 to 10 channels available in high-interference environment. Low-interference environment means most channels are available. Because a small amount of devices are in the area. There are about 28 to 37 channels available in low-interference environment.

These two environments are generated by random sequences with specified probability, 5/37 and 32/37 respectively.

B. Parametric Criteria

After validating CSA #1 and #2 in the three different environments two analyses are executed:
Number of Collision (NoC): The number of frequencies that `unmappedChannel` calculated by CSA as unusable according to channel map.
Channel Usage Distribution Frequency (CUDF): The number of times or frequency for each channel decided by CSA to be used.

IV. Results and Discussion

In this section, simulation results are shown. The evaluation concentrates on the influences and inferences under the parametric criteria.

A. Number of Collision (NoC)

It can be seen from Figure 2 that the NoC for both algorithms are almost the same, which means that CSA #1 and CSA #2 provide similar performances in all environments.

In general, NoC is inversely proportional to the number of channels available, that is, the NoC decreases as the number of available channels increases, which is obviously reasonable. Also, on the condition of similar or even equal NoC, CSA #2 needs more calculations, compared with CSA #1, to compute the final channel index, so CSA #2 would consume more computation overhead and run-time in the procedure. Table 1 shows mean and standard deviation of algorithm simulation run-time. The ratio of run-time, as CSA #1 over CSA #2, is around 1/20 when facing low interference. And it is around 1/10 when facing high or medium interference. The huge difference between 1/20 and 1/10 is from a lower mean of CSA #1 in low interference. This also illustrates CSA #2 has a more stable computational time/overhead.

Table 1. Mean And Standard Deviation Based On 100 Sets Of Test Data Of Running Time For 1000 Connection Events

Algorithm	Interference	Mean (s)	Standard Deviation
CSA #1	High	0.35848	0.016692
	Medium	0.32532	0.020519
	Low	0.21756	0.011449
CSA #2	High	3.9696	0.21090
	Medium	4.0100	0.29027
	Low	4.0099	0.18327

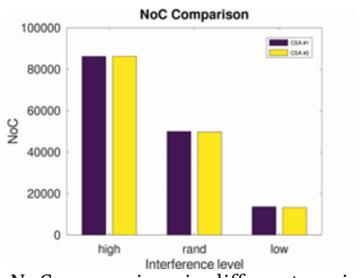

Figure 2. NoC comparison in different environments

B. Channel Usage Distribution Frequency (CUDF)

Figure 3 and 4 show how the used channels distribute in different environments for both channel selection algorithms.

It can be seen that CSA #2, compared to CSA #1, has a more uniform distribution for channel usage frequency, which means, in CSA #2, the chances of each channel being used are more equal, no matter what interference level CSA #2 is facing. Although the NoC for CSA #1 and CSA #2 are almost the same, the results from Figure 4 show that more computation overhead and procedure run-time provide a more uniform distribution on channel usage frequency.

Figure 3 clearly illustrates that chance of using channel 2 to 10 is much higher than other channels for a medium or low interference environment. While channel 0 and 1 have a rather low chance of being used. When CSA #1 is used in a high interference environment the chances more evenly spread across the channels. Future experiments will be carried out to investigate these results.

Another phenomenon from the distribution of used channels in CSA #1 under the environment of medium interference is the trend of channel usage frequency drops gradually after channel 3. This phenomenon has been analyzed in detail in [11,12]. It is caused by the inherent characteristics of CSA #1, more specifically the ununiform distribution from `remappingIndex`.

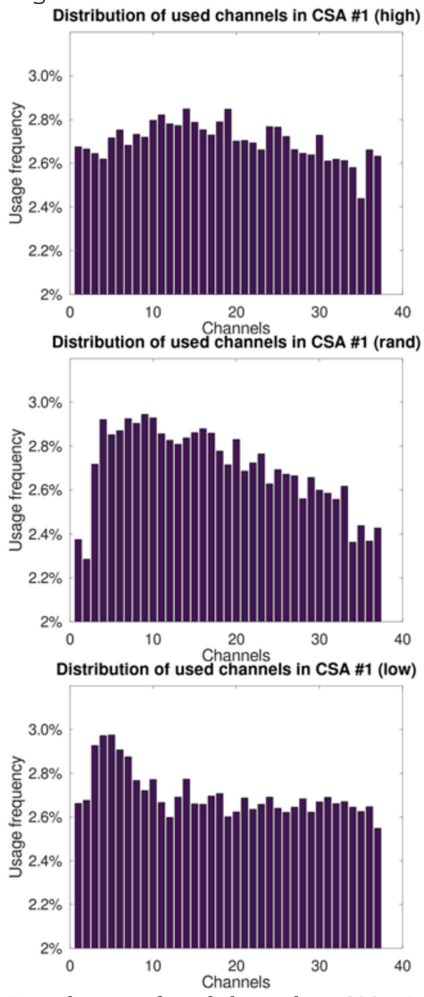

Figure 3. Distribution of used channels in CSA #1 in different environments, following the order of high, medium and low interference

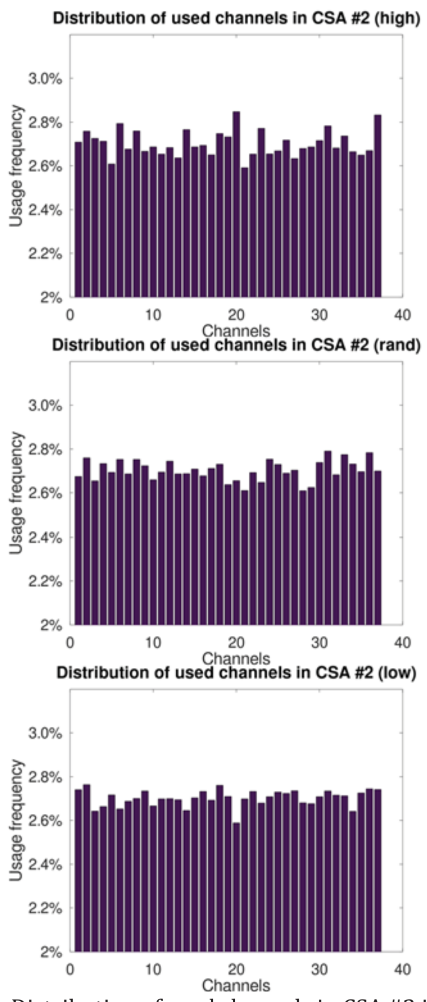

Figure 4. Distribution of used channels in CSA #2 in different environments, following the order of high, medium and low interference

V. Conclusion

The paper compares two channel selection algorithms which are applied in the BLE standard. Analyzing the simulation results after applying both algorithms, it could be concluded that if BLE devices are in a high interference environment, it is better to use CSA #1, because it provides a better computation overhead performance. If BLE devices are in a low or medium interference environment, it is recommended to use CSA #2, because of a more uniform distribution of channel map, but with the drawback of a higher computation overhead.

VI. Future Work

Theoretical simulation for CSA #1 and #2 has been performed. Although 3 types of interferences are used in simulation experiments, interferences in reality are difficult to simulate yet.

Hence, more experiments that are more similar to a real environment to see whether the same conclusions can be made. The reason why CSA #1 has a relatively uniform distribution, when facing high interference, will be further investigated. The cause why CSA #2 is robust on computational time/overhead will need for more experiments.

References

[1] S. Al-Sarawi, M. Anbar, K. Alieyan and M. Alzubaidi, "Internet of Things (IoT) communication protocols: Review," 2017 8th International Conference on Information Technology (ICIT), Amman, 2017, pp. 685-690.

[2] N. Golmie, N. Chevrollier and O. Rebala, "Bluetooth and WLAN coexistence: challenges and solutions," in IEEE Wireless Communications, vol. 10, no. 6, pp. 22-29, Dec. 2003.

[3] S. -. Lee and Y. -. Lee, "Adaptive frequency hopping for bluetooth robust to WLAN interference," in IEEE Communications Letters, vol. 13, no. 9, pp. 628-630, Sept. 2009.

[4] Q. D. La, D. Nguyen-Nam, M. V. Ngo and T. Q. S. Quek, "Coexistence Evaluation of Densely Deployed BLE-Based Body Area Networks," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, 2017, pp. 1-6

[5] M. Roy and J. H. S, "Comparative Study of Adaptive Frequency Hopping with Power Control to Avoid WLAN Interference in WPAN Systems like Bluetooth," 2010 7th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, 2010, pp. 1-5.

[6] R. A. Abbas, A. Al-Sherbaz, A. Bennecer and P. Picton, "A new channel selection algorithm for the weightless-n frequency hopping with lower collision probability," 2017 8th International Conference on the Network of the Future (NOF), London, 2017, pp. 171-175.

[7] J. So and Y. Kim, "Interference-aware frequency hopping for Bluetooth in crowded Wi-Fi networks," in Electronics Letters, vol. 52, no. 17, pp. 1503-1505, 18 8 2016.

[8] M. Zheng, B. Yang, W. Liang, H. Yu and L. Chen, "Adaptive frequency hopping in industrial Wireless Sensor Networks: A decision-theoretic framework," 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Shenyang, 2015, pp. 88-92.

[9] Bluetooth Core Specification v5.1, 2019

[10] John W. Eaton, David Bateman, Søren Hauberg, Rik Wehbring (2019). GNU Octave version 5.1.0 manual: a high-level interactive language for numerical computations. URL https://www.gnu.org/software/octave/doc/v5.1.0/

[11] M. O. A. Kalaa and H. H. Refai, "Bluetooth standard v4.1: Simulating the Bluetooth low energy data channel selection algorithm," 2014 IEEE Globecom Workshops (GC Wkshps), Austin, TX, 2014, pp. 729-733.

[12] M. O. Al Kalaa and H. H. Refai, "Selection probability of data channels in Bluetooth Low Energy," 2015 International Wireless Communications and Mobile Computing Conference (IWCMC), Dubrovnik, 2015, pp. 148-152.