eAFH: Informed Exploration for Adaptive Frequency Hopping in Bluetooth Low Energy

Valentin Poirot
Kiel University, Germany &
Chalmers University of Technology, Sweden
vpo@informatik.uni-kiel.de

Olaf Landsiedel
Kiel University, Germany &
Chalmers University of Technology, Sweden
ol@informatik.uni-kiel.de

***Abstract*—With more than 4 billion devices produced in 2020, Bluetooth and Bluetooth Low Energy (BLE) have become the dominant solutions for short-range wireless communication in IoT. BLE mitigates interference via Adaptive Frequency Hopping (AFH), spreading communication over the entire spectrum. However, the ever-growing number of BLE devices and WiFi traffic in the already crowded 2.4 GHz band lead to situations where the quality of BLE connections dynamically changes with nearby wireless traffic, location, and time of day. These dynamic environments demand new approaches for channel management in AFH, by both dynamically excluding frequencies suffering from localized interference and adaptively re-including channels, thus providing sufficient channel diversity to survive the rise of new interference. We introduce *eAFH*, a new channel-management approach in BLE with a strong focus on efficient channel re-inclusion. eAFH introduces informed exploration as a driver for inclusion: using only past measurements, eAFH assesses which frequencies we are most likely to benefit from re-inclusion into the hopping sequence. As a result, eAFH adapts in dynamic scenarios where interference varies over time. We show that eAFH achieves 98-99.5% link-layer reliability in the presence of dynamic WiFi interference with 1% control overhead and 40% higher channel diversity than state-of-the-art approaches.**

***Index Terms*—Bluetooth Low Energy, BLE, Adaptive Frequency Hopping, AFH, exclusion, blacklisting, exploration**

I. Introduction

With more than 4 billion new devices shipped in 2020 alone [1], Bluetooth has become the most prominent enabler of short-range wireless communications, empowering IoT applications in smart health [2], smart homes [3], and smart cities [4]. Bluetooth Low Energy (BLE), introduced in 2010, extends Bluetooth's support to small devices with limited resources and stringent energy constraints. Yet, the rise of Bluetooth and BLE-capable equipment comes at a price: these new devices must all co-exist in a 2.4 GHz ISM band already crowded by Bluetooth and WiFi traffic. To survive interference caused by stronger WiFi signals or concurrent BLE transmissions, BLE employs Adaptive Frequency Hopping (AFH): a BLE connection constantly hops between frequency channels. Thereby, BLE spreads communication over the entire spectrum and never stays long on the same interfered frequency, thus avoiding consecutive losses. The eventual packet loss is then handled via link-layer retransmissions.

Yet, too many retransmissions will eventually degrade performance: scattered losses degrade sound quality in audio streaming and lead to higher response time when operating wireless mice and keyboards. BLE solves this problem by excluding channels from the hopping sequence. However, the BLE standard leaves the design of channel management to vendors and developers [5]. For example, the Silicon Labs BLE stack relies on the local noise-floor [6], while the iOS operating system further removes channels overlapping with any WiFi the smartphone uses. However, Spörk et al. show that the local noise-floor is unable to detect interference close to the peripheral and leads to degraded performance [7]. Rather, the link-layer Packet Delivery Ratio (PDR) is a better measure of performance and can drive efficient channel exclusion [8]. While channel exclusion has been extensively studied in the literature, the focus of this paper, i.e., the counterpart challenge of re-including channels especially important in dynamic environments, received little attention so far.

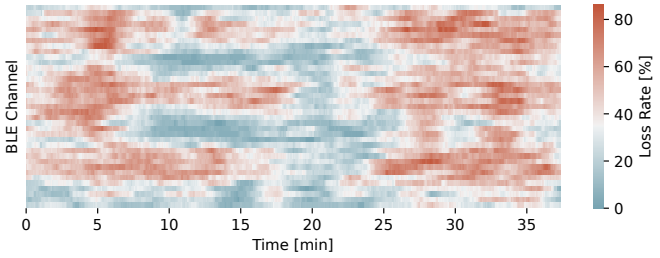

Fig. 1. Walking in a shopping mall. Two BLE devices located in the front pocket of a user's jacket and a backpack communicate every 100 ms at −4 dBm. The first 10 min are spent in the busy alleyway, after which we visit the bookstore at the 20 min mark, before going through the food stall from 25 min onward.

Challenges. Interference in the 2.4 GHz band is often dynamic: WiFi bursts, user mobility, and distance between connected devices affect the performance of a channel over time [7], [9], [10]. Therefore, the optimal set of channels to maintain in the hopping sequence changes continuously. Channel diversity is paramount in dynamic environments: it ensures that the hopping sequence is diverse enough to survive the rise of new interference. While current approaches efficiently drop channels suffering from interference, their focus on non-mobile devices lead to very naïve and inefficient re-inclusion strategies. Real-world environments are often complex: residential buildings feature many WiFi access points occupying the entire 2.4 GHz band and BLE connections

suffer from losses scattered across the entire spectrum, but should, for example, not exclude all frequencies as a result. Moreover, user mobility impacts greatly the performance of channels over time. For example, a user wearing headphones and carrying a smartphone while in a shopping mall encounters many WiFi access points operating on different frequencies; the headphones will suffer losses in different parts of the spectrum based on their current location. Fig. 1 depicts 40 min of a user walking through a mall on a Saturday afternoon. Many collisions happen across all WiFi channels at the busy mall entrance, but WiFi channels 4 and 8 quickly become free of disturbances once the user walks further along the alleyway. In the bookstore (20 min mark), almost all BLE channels are free, except for channels co-located with WiFi channel 1. Finally, interference sharply arises once moving closer to the food stalls (25 min mark and onward).

However, detecting the end of interference once a channel is excluded is non-trivial as the PDR can only be computed from packet transmissions on the channel. In this work, we employ channel exploration to re-assess channel performance. Exploration only induces minimal overhead when updating the hopping sequence and only the central device needs to implement exploration, thus working with any commercial peripherals already on the market. The challenge lies in selecting the most promising channels to re-include and explore.

Approach. We present *eAFH*, a channel management mechanism for Adaptive Frequency Hopping with a strong focus on efficient channel re-inclusion. eAFH introduces the notion of *informed exploration* as key building block to re-include channels from past performance measurements: eAFH combines the link-layer packet delivery ratio, nearby-channel's performance, and timeouts inspired by exponential backoffs to assess the exclusion and re-inclusion of BLE channels. As a result, eAFH adapts to dynamic scenarios in which interference varies over time, as well as ensures channel diversity even under scattered losses due to high path loss. Further, eAFH is standard-compliant and works with all commercial peripherals available today, only the central device needs to implement eAFH.

Contributions. This paper makes the following contributions:

1) We introduce the Uncertainty Measure U, a new heuristic allowing informed exploration when re-including channels for Adaptive Frequency Hopping;
2) We present eAFH, an AFH system featuring a reliability-based channel exclusion and an uncertainty-aware channel inclusion for Bluetooth Low Energy. eAFH only needs to be implemented by the central, and thus works with all commercial peripherals available today;
3) We implement and distribute eAFH[1] for the Zephyr RTOS and evaluate it against dynamic WiFi and BLE traffic. We show that eAFH achieves 98-99.5% link-layer reliability against dynamic WiFi interference, providing a 40% increase in channel diversity for a 1% control overhead cost.

[1]Available at: github.com/ds-kiel/eAFH

II. Background

Introduced as part of Bluetooth 4.0 in 2010 [5], Bluetooth Low Energy (BLE) is a short-range wireless technology in the 2.4 GHz ISM band. BLE targets direct, one-hop communication and provides data rates of up to 2 Mbit/s within 10-50 meters, typically. It uses 40 2-MHz wide frequency channels, 37 reserved for data traffic, and 3 for advertisements and broadcasts.

Connections. BLE has two operation modes: connected and non-connected mode. The non-connected mode is used to broadcast data, to disclose the presence of connectable devices, or to initialize a connection between a central device (e.g., smartphone) and a peripheral (e.g., headphones). Once a connection is established, the devices periodically communicate on the 37 dedicated data channels. The central device starts a *connection event* by switching to a new channel and sending a packet to the peripheral. Both devices then successively transmit until no new data is available or when they reach the maximum time allocated. The connection event then ends and communication is resumed on a new frequency after a pre-defined connection interval.

BLE provides reliable communication through the use of acknowledgments at the link layer. Each packet includes a 1-bit Sequence Number (SN) and a 1-bit Next Expected Sequence Number (NESN). Packets are repeated until acknowledged, even across multiple connection events.

Frequency hopping. BLE relies on channel hopping to avoid succumbing to WiFi interference; a connection hops to a new channel for each connection event. A central maintains and updates a channel map C_{map} for each established connection indicating all usable frequencies. A peripheral maintains a copy of C_{map}, but cannot modify it; since BLE 5.3 [5], the peripheral can indicate bad channels that should be removed, and the central is responsible for the change. The central adds or removes channels from C_{map} via channel map update procedures and the update is applied by both devices after a given delay greater than 6 connection intervals. The Channel Selection Algorithm (CSA), responsible to select the next channel, is run independently on both devices and defined in the standard.

Link performance. Different metrics can be extracted from BLE communication, such as the link-layer Packet Delivery Ratio (PDR), the per-packet received signal strength (RSSI), or the per-channel noise-floor [8]. Spörk et al. show that the link-layer PDR is an accurate estimator of a channel quality in BLE [8], and define the per-channel PDR as $PDR = \frac{\#ACK(P \to C)}{\#TX(C \to P)}$ where $\#ACK(P \to C)$ is the number of ACKs received by the central, and $\#TX(C \to P)$ the number of messages sent by the central.

III. Design: Exploration and eAFH

We begin by introducing the concept of channel inclusion via exploration in §III-A, derive our uncertainty heuristic U as

a driver for informed exploration in §III-B and present eAFH, our channel-management system for AFH in §III-C.

A. Channel Inclusion via Exploration

Channel inclusion is paramount in dynamic environments as it ensures that the hopping sequence has high channel diversity and can survive the rise of new interference. To avoid re-including previously excluded channels that still suffer from degraded conditions, we must, however, collect up-to-date measurements on the current performance of excluded frequencies. While PDR accurately models the performance of a channel, it must be computed over packet exchanges. Without new transmission, the PDR becomes *stale*: the last measurement might be seconds-, hours-, or even days-old.

Performance estimation. To update the PDR, we must either use channel probing or introduce active performance measurement over excluded channels. In active probing, the central sends probes on an excluded channel and listens for an acknowledgment. Active probing is however costly: it requires additional signaling, as the peripheral must be aware which channel will be probed and when, probing consumes energy, and it must not impede on other connections. Further, probing is not supported by the standard, and would work only with peripherals that implement the same probing algorithm.

In this paper, we instead rely on exploration as re-inclusion mechanism: eventually, the central re-includes the most promising excluded channels into the hopping sequence. By re-using these channels for data exchanges, we obtain new measurements and update the estimated performance. Channel exploration is simple and standard-compliant: all commercial peripherals can be used as only the central needs to implement the re-inclusion mechanism. While using data packets to measure the performance seems risky, the inherent link-layer retransmissions of BLE ensure that the data is never lost. Further, it is generally more energy-efficient than relying on active probing: active probing requires at least three packets (probing coordination, probe, and channel map update), while exploration only requires one (channel map update). Exploration immediately sends the C_{map} update and uses the next data packet as probe. If lost, the packet is retransmitted on the next frequency, and the channel re-excluded within the next update.

Exploring channels. Designing an efficient exploration mechanism poses the following challenges: in a scenario with mobility, it is important to explore channels frequently to maintain enough channel diversity, as new interference might arise in a different part of the spectrum (see Fig. 1). In a static scenario, for example an office with frequent traffic on the same WiFi channels, exploration at a constant rate is harmful as the same frequencies are continuously under interference. While we could simply re-include the entire spectrum once too few channels are left active, this method leads to high losses and delays until the channel map converges to interference-free frequencies: it implies that we even re-include channels that are under constant interference. In this paper, we introduce *informed exploration* to solve the problem of channel re-inclusion. We rely on PDR measurements, exponential backoff, and performance of nearby channels to estimate how promising exploring channels is.

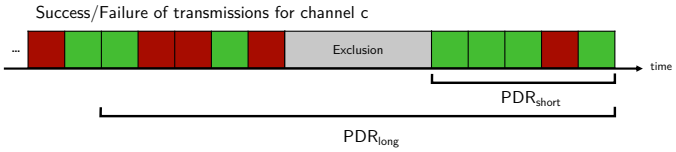

Fig. 2. Measures of performance. PDR_{short} (5 samples) represents the immediate performance, PDR_{long} (10 samples) the average performance of a channel, respectively.

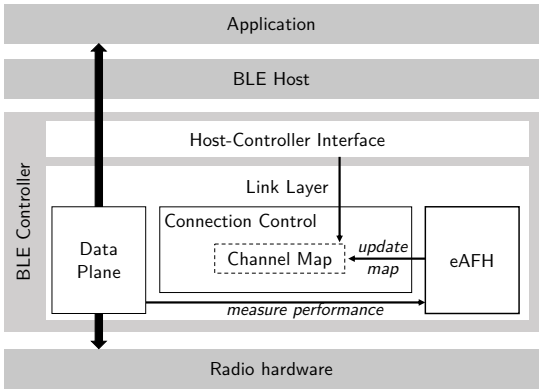

Fig. 3. eAFH System Integration. eAFH collects channel performance during data transmissions, computes the channels to exclude and explore, and updates the channel map used by the connection.

B. Estimating Uncertainty

Goal. To drive an efficient exploration of excluded channels, we require a heuristic able to inform us of the possible new channel performance. We introduce the *Uncertainty Heuristic* U as our heuristic: we predict how likely the actual performance of a channel has changed since we last used it. We rely on the time elapsed since we last used a channel (the staleness) and its past performance to drive our uncertainty estimation: U does not require new PDR measurements. Further, since WiFi affects several BLE channels at once, we take into account how nearby channels perform to further refine our heuristic.

Measuring performance. We extend the Packet Delivery Ratio metric (PDR, see §II) into two distinct metrics: the short-term and long-term performance of the channel c, $PDR_{short}(c)$ and $PDR_{long}(c)$, respectively, see Fig. 2. $PDR_{short}(c)$ represents the immediate state of the channel c, while $PDR_{long}(c)$ represents its long-term, average performance. We compute PDR_{short} over a sliding-window of the last transmissions and truncate everything that happened prior to a channel exclusion by resetting it at each exclusion. In contrast, $PDR_{long}(c)$ keeps statistics obtained across exclusions until the sliding-window is full, i.e., it is never reset. As a result, PDR_{short} represents the immediate performance of a channel and quickly evolves, while PDR_{long} represents the average long-term performance and we use it to estimate if a channel frequently suffer from interference.

Staleness is not uncertainty. While it is easy to determine the staleness (i.e., the age) of measurements, it is not, by itself, a determining proxy for channel re-inclusion. For example, in a static setup with continuous interference from WiFi channel 1, re-including channels after a static timeout driven by the age of past measurements will always fall victim to the same interference. We must instead combine the staleness with a measure of past losses, i.e., PDR_{long}.

Staleness and long-term performance. We define the uncertainty due to staleness and past losses U_{stale}. We define $(t_{now} - t_{exclusion}(c))$, the time elapsed since a channel is excluded as its staleness. To penalize channels with lower long-term average performance (PDR_{long}), we take inspiration from the exponential backoff mechanism: the higher the number of losses a channel has suffered, the exponentially longer it takes to explore it again. Once excluded, a channel remains inactive for $D \times 2^{losses}$ seconds, where D is the default exclusion duration (D = 2s) and $losses$ is obtained from PDR_{long}. Finally, we express the uncertainty due to staleness and long-term performance as:

$$U_{stale}(c) = \frac{t - t_{exclusion}(c)}{D \times 2^{losses(c)}} \quad (1)$$

i.e., the ratio between elapsed time since exclusion and the required exclusion duration.

Correlated performance. Some interference affects more than one BLE channel simultaneously: for example, WiFi transmissions overlap with several channels. From the current performance of direct neighboring channels, we infer how likely a channel is suffering from wideband disturbances. We define $U_{near}(c)$ as the uncertainty caused by correlated interference, such as:

$$U_{near}(c) = -\left(\frac{LossRate(c-1) + LossRate(c+1)}{2}\right) \quad (2)$$

As the loss rate of nearby channels increases, our certainty that channel c is also affected increases; our uncertainty of the channel's current performance therefore decreases.

Uncertainty heuristic. We therefore define our Uncertainty heuristic $U(c)$ such as:

$$U(c) = U_{stale}(c) + U_{near}(c) \quad (3)$$

where $U_{stale}(c)$ and $U_{near}(c)$ are given by Eq. 1 and Eq. 2.

Bounded convergence. By relying on the heuristic U, all excluded channels are eventually included again; it is an important property for AFH in dynamic environments. Thus, if our environment changes, our system eventually converges to a list of channels without excessive or outdated exclusions.

C. eAFH System Integration

Controller extension. eAFH acts as an extension to the link-layer of the BLE controller and does not replace any existing mechanism, see Fig. 3. The central device initializes a new eAFH instance for each active connection as different peripherals may be physically far away from each other and suffer localized interference.

Algorithm 1: eAFH Procedure

Input: channel map C_{map}, last used channel c_{t-1}, last measurement m
Update $PDR_{short}(c_{t-1})$, $PDR_{long}(c_{t-1})$ with m
1. Exclusion
foreach *channel c* **do**
  if $PDR_{short} \leq 95\%$ **then**
    $C_{map} \leftarrow C_{map} \setminus \{c\}$ ▷ Exclude c
    Reset $PDR_{short}(c)$
  end
end
2. Exploration
foreach *channel c* **do**
  if *c in* C_{map} **then**
    $staleness(c) \leftarrow 0$
  else
    $staleness(c) \leftarrow staleness(c) + 1$
    $U_{stale}(c) \leftarrow \frac{staleness(c)}{D \times 2^{losses(c)}}$
    $U_{near}(c) \leftarrow -\frac{LossRate(c-1)+LossRate(c+1)}{2}$
    $U(c) \leftarrow U_{stale}(c) + U_{near}(c)$
    if $U(c) \geq 1.0$ **then**
      $C_{map} \leftarrow C_{map} \cup \{c\}$ ▷ Explore c
    end
  end
end
if $|C_{map}| < C_{min}$ **then**
  Include $C_{min} - |C_{map}|$ channels with the highest PDR_{long}
end
`ChannelMapUpdateProcedure` (C_{map})

Execution. eAFH collects performance measurements after each connection sub-event (e.g., a transmission or reception), and executes its channel-management strategy once the connection event and all time-critical operations have been executed. Algorithm 1 depicts eAFH's procedure.

1. Exclusion: eAFH removes all channels with insufficient short-term reliability PDR_{short} by excluding channel c if $PDR_{short}(c) < T_{exclu}$ ($T_{exclu} = 95\%$ computed over 15 samples,see §IV-B). After exclusion, $PDR_{short}(c)$ is resetted.

2. Exploration: After exclusion, eAFH proceeds to the exploration mechanism. We update the uncertainty heuristic U following Eq. 3 and re-include channel c if $U(c) \geq T_{incl}$ ($T_{incl} = 1.0$), i.e., our uncertainty of this channel's performance has grown significantly.

Ensuring channel diversity. According to the BLE standard, the channel map must contain at least C_{min} active channels ($2 \leq C_{min} \leq 37$), where C_{min} is set by the peripheral when the connection starts. There may be less than C_{min} channels active after the exclusion and exploration steps. eAFH fulfills the condition by including additional channels until C_{min} is reached. Specifically, eAFH selects the channels with the highest long-term performance PDR_{long} recorded so far, thus maximizing reliability while ensuring channel diversity.

eAFH then executes the channel map update procedure of the controller's link-layer to propagate the new information to the peripheral. The devices apply the new hopping sequence after 6 connection events, as per the standard. If a channel map update is already underway, eAFH does not start a new update procedure.

Implementation. We implement eAFH in C for the Zephyr RTOS BLE stack [11], and make our implementation open-source (see §I). eAFH's implementation is platform-independent, uses up to 4 kB of memory, and takes 700 μsec

to compute a new channel map on a 64 MHz CPU. Moreover, eAFH is modular and easy to extend with new channel management algorithms.

IV. Evaluation

A. Setup

We experimentally evaluate eAFH using two nRF52840DK (nRF52). The nRF52 features a 64 MHz CPU, 256 KB of RAM, and a BLE 5 radio. The devices are spaced 15 cm apart, with a 0 dBm transmit power, and transmit every 20 ms. The boards are in close proximity to attenuate the impact of uncontrolled interference (e.g., nearby WiFi access-points, in-use BLE devices). We set two Raspberry Pi 4 at a distance of 20 cm away from the nRF52 boards, and use them to inject strong WiFi interference. We set TX Power to +20 dBm and use iperf to generate heavy TCP traffic, similar to a user downloading files or watching HD videos online.

Scenarios. We evaluate the following scenarios:

- Static setup: We generate TCP traffic on WiFi channel 1, all remaining channels are left free.
- Bluetooth co-existence: We use a Bluetooth headset connected to a laptop to play music. The headset is located close to the nRF52 peripheral and communicates every 11.25 ms with the laptop. The laptop is placed 20 cm away from central.
- Slow dynamics: We generate TCP traffic on a random WiFi channel for 30 seconds. We then stop jamming for 30 additional seconds. The access-point (AP) switches to a new random WiFi channel and starts over. This represents a scenario where a user walks around a street and encounters different WiFi APs, or the library in Fig. 1.
- Fast dynamics: Every 30 seconds, the AP changes to a new random WiFi channel and immediately starts transmitting (no free periods). This represents mobility in denser deployment such as the food stalls in Fig. 1.

Metrics. We evaluate the following set of metrics: **1.** Channel diversity: number of active channels. **2.** Link-layer PDR: Number of frames correctly received (App-layer reliability is always 100% due to retransmissions). **3.** Overhead: Packets exchanged to update C_{map} and retransmit failed transmissions. Unless stated otherwise, we depict the average result over multiple experiments, while bars represent the standard deviation.

Baselines. We compare eAFH to the default Zephyr implementation that does not implement channel management (referred to as No AFH), the state-of-the-art PDR-based channel exclusion by Spörk et al. (referred to as PDR-Exclusion) [8], and a static re-inclusion mechanism (referred to as Static timeout) similar to Mast et al. [12]. PDR-Exclusion exclude channels once the PDR drops below 95% (computed over 20 samples, as described in the original work) and the channel map and all statistics reset once less than C_{min} channels are available. Static timeout re-includes channels after a static timeout.

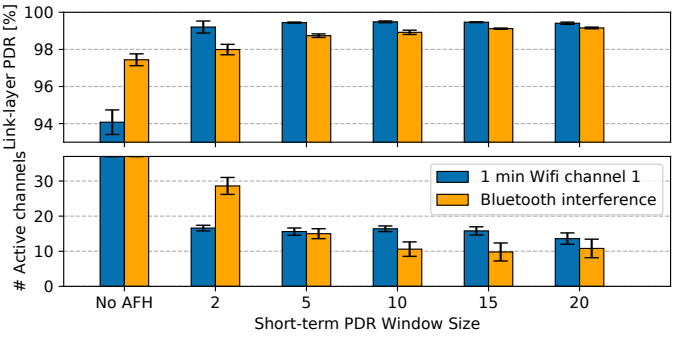

Fig. 4. Effect of the sliding window size on channel exclusion. The more samples are used to compute PDR_{short}, the better reliability becomes, at the cost of many excluded channels.

B. Channel Exclusion

Scenario. We experimentally study the aggressiveness of exclusion mechanisms by varying the number of samples PDR_{short} is computed from (referred to as the window size). We run two scenarios: 1. 1-min long jamming on WiFi channel 1 before leaving the spectrum undisturbed, and 2. in the presence of a concurrent BLE communication. In these experiments, channels are never re-included to see the impact of scattered losses over the exclusion mechanism. We run 5× 5-min experiments for each scenario and parameter value.

Results. Fig. 4 depicts the link-layer PDR and the number of channels kept active by the exclusion mechanism. The baseline (no exclusion) achieves 94% link-layer PDR against the strong 1-min WiFi interference while the PDR-based exclusion mechanism achieves from 99.2% up to 99.4% based on its window size and by discards 20 channels for 2 samples up to 24 channels for 20 samples, i.e., it can easily detect nearby WiFi interference with a small window size. In contrast to WiFi, concurrent BLE communication creates losses scattered across the entire spectrum and should not trigger exclusions. The baseline achieves 97.4% PDR, while the exclusion mechanism achieves from 98% (2 samples, 8 excluded channels), up to 99.2% (20 samples, 26 excluded channels). The larger the window size, the more likely multiple losses will occur within its span and cause an exclusion, potentially leading up to all channels being excluded. Computing the PDR over many samples increases the aggressiveness of the exclusion mechanism and leads to system collapse (no channel diversity) in the presence of scattered losses. For the remainder of the evaluation, we compute PDR_{short} with a window size of 15, providing a good trade-off between fast exclusion of WiFi interference but not being overly aggressive against random losses.

C. Channel Inclusion

Scenario. We study the effectiveness of eAFH's re-inclusion mechanism. We investigate four inclusion techniques: 1. Static timeout, set to re-include a channel after 2 seconds (conceptually similar to Mast et al. but much shorter [12]); 2. Static timeout set to 30 seconds; 3. Dynamic timeout using

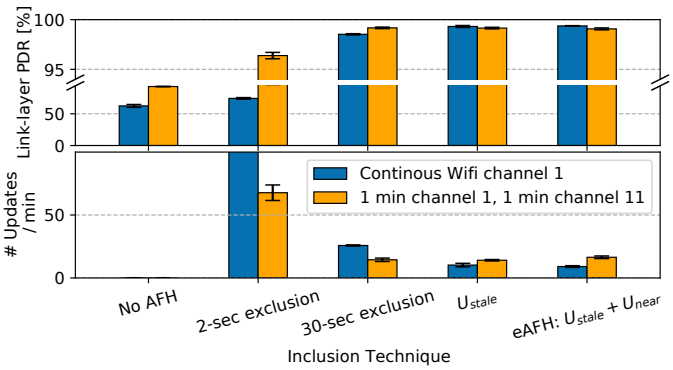

Fig. 5. Channel Inclusion. Static timeouts create a large communication overhead, while eAFH's dynamic exploration provides high reliability and low overhead.

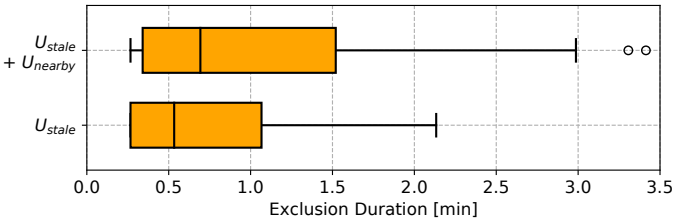

Fig. 6. Improving Uncertainty using channel loss correlation. Detecting losses on nearby channels with U_{near} leads to longer exclusion of channel interfered by WiFi signals.

U_{stale} (see Eq. 1), acting as an improved exponential backoff, with default inclusion after 2 seconds; 4. eAFH using both U_{stale} and U_{near} (see §III-B), thus taking advantage of the performance of nearby channels. We evaluate these approaches in two scenarios: a. a static setup with continuous jamming, where frequent exploration is detrimental, and b. two 1-min jamming on two different WiFi channels representing dynamic scenarios featuring mobility, where exploring is beneficial. We run 5$times$ 5-min experiments for each approach and scenario.

Results. Fig. 5 depicts the link-layer PDR and the number of update procedures exchanged between the central and the peripheral, i.e., the control overhead. Against continuous jamming, the baseline only achieves 62% link-layer PDR. By excluding channels and re-including them every 2 seconds, reliability increases to 74%. However, the static re-inclusion timeout induces 257 channel map updates per minute, this represents an 8.6% communication overhead when communicating every 20 ms. A longer static timeout improves reliability and reduces overhead, but continues to sample interfered channels over time: the 30-sec timeout achieves 98.5% PDR and reduces its overhead to 0.86% (25 updates per minute). By using a PDR-driven exponential backoff (U_{stale}), eAFH requires fewer and fewer updates as time passes. The U_{stale}-only approach achieves 99.3% and requires 10.2 updates per minute (0.34% overhead) for a 5 min experiment. For longer experiments, U_{stale}'s overhead would slowly decrease. By using both U_{stale} and U_{near}, eAFH further achieves 99.4% with 9.1 updates per minute (0.30% overhead).

In the presence of two short 1-min WiFi bursts, the baseline achieves 93% PDR. The 2-sec timeout increases reliability to 96.4% and requires 68 updates per minute (2.3% overhead). Most updates happen during the jamming phases and stop once the spectrum is left undisturbed. The 30-sec timeout achieves a PDR of 99.2% with 14 updates per minute (0.5% overhead); it only re-includes an interfered channel twice before the interference stops, and requires at most 30 seconds before all free channels are re-included. In contrast, eAFH starts with very short exclusion periods to ensure fast adaptivity to short disturbances but increases its timeout as the losses continue. Although its initial exclusion period is 15$\times$ shorter, U_{stale} obtains similar performance than the 30-sec static timeout: 99.2% PDR, with 14 updates per minute (0.5% overhead). eAFH achieves similar performance as U_{stale} as the knowledge of nearby channels exclusion is particularly important against long-term and wide-band interference.

Effect of U_{near}. The inclusion evaluation shows that the exponential backoff U_{stale} accounts for an important part in the performance of eAFH. We now investigate the exact effect of including U_{near}. by design, U_{near} should cause longer exclusion periods before re-exploring a channel if nearby channels are also affected by interference. We run eAFH with and without U_{near}, against continuous jamming on WiFi channel 1, for 5 minutes, and repeat the experiment 10 times. Fig 6 depicts the distribution of how long channels interfered by WiFi are excluded. When eAFH uses both U_{stale} and U_{near}, channels are excluded 37% longer than when eAFH uses U_{stale} as its sole driver for exploration. In contrast, channels that are excluded but did not suffer WiFi interference, i.e., excluded channels that do not overlap with WiFi channel 1, were not excluded longer using U_{near} (no difference at all, not shown here).

Main findings. These results show that, although theoretically at disadvantage against short bursts, eAFH outperforms static timeouts proposed by Mast et al. [12] in both static and dynamic scenarios; its exploration mechanism fine-tunes the exclusion timeout to reduce the control overhead and achieves the best reliability against all baselines. Further, by using losses correlated across nearby channels, eAFH avoids exploring channels that are affected by wide-band interference such as WiFi.

D. State-of-the-art Comparison

Scenario. We now compare eAFH against the No AFH baseline (channel hopping without exclusion), and PDR-Exclusion, the PDR-driven channel exclusion proposed by Spörk et al. [8]. We investigate four scenarios: 1. static setup with constant WiFi jamming (channel 1); 2. slow dynamics: 30 seconds jamming on a random WiFi channel every minute, emulates a user walking along a non-busy shopping mall (see §IV-A); 3. fast dynamics: constant WiFi jamming randomly hopping channels every 30 seconds, emulates a busy mall with mobility; 4. against real-life disturbances in a residential building (many WiFi access-points, Bluetooth, and BLE traffic). In the residential building scenario, the two BLE

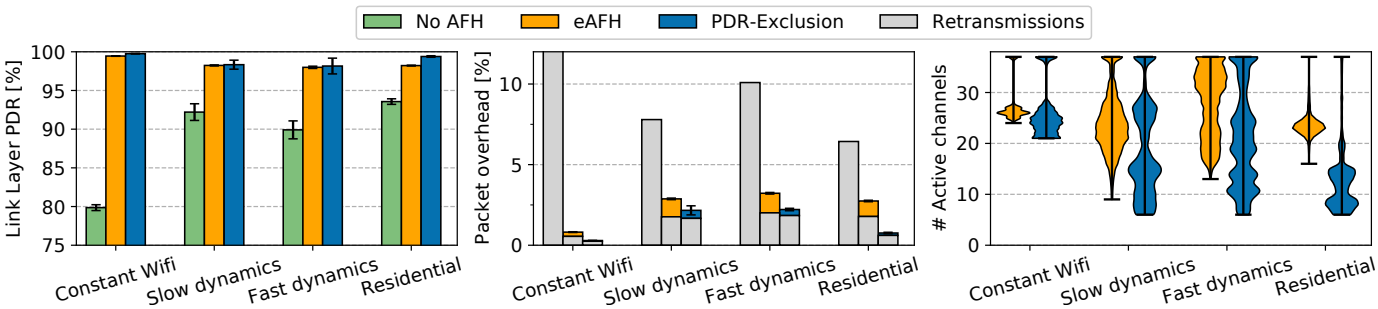

Fig. 7. AFH techniques in different environments. Both eAFH and PDR-Exclusion greatly improve the link-layer PDR in all environments. Exploration in eAFH induces a small control overhead but is able to maintain a higher channel diversity as a result. The AFH overhead is well below the number of retransmissions required by No AFH, meaning eAFH saves energy compared to No AFH. PDR-Exclusion often reaches the limit C_{min} of active channels and must reset as a result, while eAFH always provides more active channels than the limit.

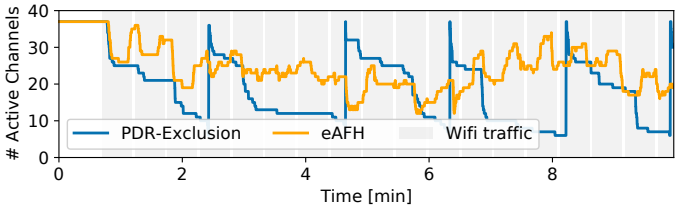

Fig. 8. Representative run of eAFH and PDR-Exclusion against WiFi interference hopping between channels. PDR-Exclusion continuously excludes channels until the minimum is reached, and resets its channel map.

devices are standing 1.5 meters apart, while they stand in close proximity in all former scenarios (roughly 15 cm). The slow and fast dynamics represent a user walking in a shopping mall and encountering new access points periodically. In the slow dynamics, the entire spectrum is undisturbed for 30 seconds before the next WiFi traffic, the fast dynamics scenario does not feature free episodes (cf. grayed-out blocks in Fig. 8).

Results. Fig. 7 presents the link-layer PDR, communication overhead, and the distribution of the number of active channel maps of all approaches. In all scenarios, eAFH and PDR-Exclusion greatly improve reliability compared to the No AFH baseline that does not exclude channels. Against constant WiFi traffic, the baseline achieves 79.9% link-layer PDR, i.e., 20% of all packets need to be retransmitted at least once due to losses, while eAFH and PDR-Exclusion achieve 99.5% and 99.8%, respectively.

In dynamic environments, the No AFH baseline achieves 92.2% and 89.9% reliability against slow and fast dynamics, respectively. Both eAFH and PDR-Exclusion provides improved performance, 98.2% and 98% for eAFH, 98.3% and 98.2% for PDR-Exclusion. The exploration mechanism of eAFH has a slightly larger communication overhead; eAFH induces a 1.1% communication overhead to explore, while PDR-Exclusion creates a 0.5% overhead by only excluding channels until triggering a full C_{map} reset. In the slow dynamics scenario, the baseline requires an 8% communication overhead to retransmit failed packets, eAFH requires 2.9% overhead: 1.8% for retransmissions, and 1.1% for exploration. eAFH is thus more energy-efficient than the baseline. PDR-Exclusion requires a 2.2% total overhead in the slow dynamics scenario, most of it coming from retransmissions.

Furthermore, eAFH always improves channel diversity compared to PDR-Exclusion: it continuously excludes frequencies until the bare minimum is left ($C_{min} = 6$ here) and simply resets the channel map, as depicted in Fig. 8. This causes a temporary higher loss rate until interfered channels are again excluded. If the connection interval is low, this might also induce a higher latency at the application level. eAFH provides a median number of 23 active channels against slow dynamics, and 29 channels against fast dynamics, while PDR-Exclusion provides a median number of active channels of 16 against slow dynamics, and 19 against fast dynamics. More importantly, eAFH always provides more than the C_{min} active channels required, and never reaches the C_{min} limit. In contrast, PDR-Exclusion reaches C_{min} on average 3.2 times, and thus resets 3.2 times in 10 minutes, under fast dynamics. Under slow dynamics, PDR-Exclusion resets 2.6 times. In a residential setting with devices set 1.5 meters apart, eAFH achieves 98.2% reliability with 24 active channels, while PDR-Exclusion achieves 99.2% by deactivating almost all channels.

Main findings. eAFH achieves 98-99.5% link-layer PDR against static and dynamic interference. The 1.1% communication overhead due to exploration remains smaller than the 6-20% overhead due to retransmissions in the No AFH case, i.e., eAFH saves energy. Finally, with its exploration, eAFH provides 40% more active channels than PDR-Exclusion in dynamic settings: this is primordial since channel hopping is the cornerstone of BLE communication. By reactivating all channels at once, PDR-Exclusion temporarily causes high losses and delays until it converges to a stable channel map. In contrast, by relying on constant and informed exploration, eAFH avoids re-including channels that are known to suffer from interference while maintaining a higher channel diversity.

V. Related Work

Adapting BLE. Spörk et al. evaluate the selection of metrics for channel exclusion in BLE and demonstrate that the channel PDR is the best performance estimator as the noise-floor and SNR measurements fail to encompass all

possible communication failures [8]. The authors present a channel exclusion mechanism building on the PDR, but only re-include channels once too many channels are excluded. In contrast, this paper focuses on the problem of channel inclusion and introduces informed exploration as re-inclusion heuristic. Mast et al. propose a simple channel re-inclusion following static timeouts, and evaluate some form of exponential backoff [12]. In contrast, eAFH is a full-fledged system: it combines PDR, staleness, and nearby channels' statistics to drive informed exploration, eAFH is shown to improve performance against static timeouts and is open-source.

Other works propose to modify the Channel Selection Algorithm (CSA) used by BLE to better avoid interfered channels. For example, Pang et al. introduce the Interference Awareness Scheme (IAS) [13]: a channel is excluded if no packets could be received during a connection event, or if too many packets are lost during an event. The authors also propose a new CSA: the channels are weighted by the probability that interference affects the channel, and the new CSA skips channels that are likely to be interfered. Cheikh et al. introduce SAFH and also rely on weights to improve channel selection [14]: channels with low frame error rates have a lower probability to be used next or are excluded altogether, while good channels are more likely to be used. eAFH does not rely on a modified CSA but rather on standard-compliant procedures. As such, eAFH works with all commercial peripherals, while both IAS and SAFH require modified BLE peripherals to operate.

Park et al. improve energy consumption and QoS in BLE with AdaptaBLE [15], by controlling the transmission power, BLE physical mode, and connection, but without managing the channels. AdaptaBLE and eAFH are two orthogonal approaches and can be used together to further improve Quality of Service and energy.

Other technologies. Frequency hopping is an effective measure to mitigate interference and is used in many technologies, e.g., 802.15.4e-TSCH [16], [17], Bluetooth Classic [18], [19] and 5G [20]. A-TSCH excludes channels via RSSI measurements [16]; Elsts et al. combine RSSI measurements and Packet Reception Rate (PRR) to detect channels with low performance [17]; while LABeL uses an exponential weighted moving average of the PDR [21]. Machine learning techniques have also been explored: MABO-TSCH uses Multi-Armed Bandits to estimate link quality [22], while Farahmand and Nabi use self-supervised deep learning to model future link quality and drive blacklisting [23]. In Bluetooth Classic, Lee et al. propose to use adjacent channels' PER [18], while Rhode-Schwarz uses packet losses [19]. However, as for BLE-oriented works, they lack efficient channel re-inclusion mechanisms.

VI. Conclusion

By constantly hopping between frequencies with its Adaptive Frequency Hopping (AFH) mechanism, Bluetooth Low Energy mitigates the effects of interference and avoids consecutive losses. Further, BLE allows the exclusion of channels to avoid hopping back under the same disturbances. However, the problem of channel re-inclusion is crucial in dynamic environments where interference sources come-and-go but has received little attention from the literature. We present *eAFH*, a channel management mechanism with a special focus on efficient channel re-inclusion. eAFH introduces the notion of *informed exploration* as a key building block to efficiently re-include channels from past measurements. As a result, eAFH adapts to dynamic scenarios where interference sources vary over time. In eAFH, only the central needs to implement the exploration mechanism and thus works with all commercial peripherals available today. We experimentally show that eAFH achieves 98-99.5% link-layer reliability in the presence of dynamic WiFi interference, improves channel diversity by 40% compared to state-of-the-art approaches, while only inducing a 1% control overhead.

References

[1] Bluetooth SIG, "Bluetooth 2021 Market Update," 2021.

[2] T. Suzuki, H. Tanaka, S. Minami *et al.*, "Wearable wireless vital monitoring technology for smart health care," in *ISMICT*, 2013.

[3] M. Collotta and G. Pau, "A novel energy management approach for smart homes using bluetooth low energy," *IEEE J. on Selected Areas in Communications*, vol. 33, no. 12, pp. 2988–2996, 2015.

[4] P. Spachos and K. Plataniotis, "BLE beacons in the smart city: Applications, challenges, and research opportunities," *IEEE IoT Mag.*, 2020.

[5] Bluetooth SIG, "Bluetooth Core Specification v5.3," 2021.

[6] "Silicon Labs BLE stack documentation v3.2," https://docs.silabs.com/bluetooth/latest/, accessed: 2021-07.

[7] M. Spörk, C. A. Boano, and K. Römer, "Improving the timeliness of bluetooth low energy in dynamic RF environments," *ACM Trans. Internet Things*, vol. 1, no. 2, Apr. 2020.

[8] M. Spörk, J. Classen, C. A. Boano *et al.*, "Improving the reliability of bluetooth low energy connections," in *EWSN*, 2020.

[9] M. Omar Al Kalaa, W. Balid, N. Bitar *et al.*, "Evaluating bluetooth low energy in realistic wireless environments," in *IEEE WCNC*, 2016.

[10] R. Natarajan, P. Zand, and M. Nabi, "Analysis of coexistence between IEEE 802.15.4, BLE and IEEE 802.11 in the 2.4 GHz ISM band," in *IEEE IECON*, 2016, pp. 6025–6032.

[11] Zephyr Project, "Zephyr Real-Time Operating System," 2016.

[12] J. Mast, T. Hänel, and N. Aschenbruck, "Enhancing adaptive frequency hopping for bluetooth low energy," in *IEEE LCN*, 2021, pp. 447–454.

[13] B. Pang, K. T'Jonck, T. Claeys *et al.*, "Bluetooth low energy interference awareness scheme and improved channel selection algorithm for connection robustness," *Sensors*, vol. 21, no. 7, 2021.

[14] S. Ben Cheikh, T. Esemann, and H. Hellbrück, "SAFH - smooth adaptive frequency hopping," in *Intl. Workshop on Cross Layer Design*, 2011.

[15] E. Park, M.-S. Lee, H.-S. Kim *et al.*, "AdaptaBLE: Adaptive control of data rate, transmission power, and connection interval in bluetooth low energy," *Computer Networks*, vol. 181, 2020.

[16] P. Du and G. Roussos, "Adaptive time slotted channel hopping for wireless sensor networks," in *CEEC*, 2012, pp. 29–34.

[17] A. Elsts, X. Fafoutis, R. Piechocki *et al.*, "Adaptive channel selection in ieee 802.15.4 tsch networks," in *GIoTS*, 2017, pp. 1–6.

[18] S.-H. Lee and Y.-H. Lee, "Adaptive frequency hopping for bluetooth robust to wlan interference," *IEEE Communications Letters*, 2009.

[19] "Bluetooth adaptive frequency hopping on a R&S CMW," http://www.rohde-schwarz.com/appnote/1C108, accessed: 2021-07.

[20] A. Li, G. Han, J. J. P. C. Rodrigues, and S. Chan, "Channel hopping protocols for dynamic spectrum management in 5g technology," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 102–109, 2017.

[21] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios *et al.*, "Label: Link-based adaptive blacklisting technique for 6tisch wireless industrial networks," in *ACM MSWiM*. ACM, 2017, p. 25–33.

[22] P. H. Gomes, T. Watteyne, and B. Krishnamachari, "MABO-TSCH: Multihop and blacklist-based optimized time synchronized channel hopping," *Trans. on Emerging Telecommunications Technologies*, 2018.

[23] M. Farahmand and M. Nabi, "Channel quality prediction for TSCH blacklisting in highly dynamic networks: A self-supervised deep learning approach," *IEEE Sensors J.*, 2021.