Improving the Reliability of Bluetooth Low Energy Connections

Michael Spörk[†], Jiska Classen[§], Carlo Alberto Boano[†], Matthias Hollick[§], and Kay Römer[†]
[†]Institute of Technical Informatics, Graz University of Technology, Austria
[§]Secure Mobile Networking Lab, Darmstadt University of Technology, Germany
{michael.spoerk, cboano, roemer}@tugraz.at, {jclassen, mhollick}@seemoo.de

Abstract

To sustain a reliable data exchange, applications based on Bluetooth Low Energy (BLE) need to effectively blacklist channels and adapt the physical mode of an active connection at runtime. Although the BLE specification foresees the use of these two mechanisms, their implementation is left up to the radio vendors and has not been studied in detail yet.

This paper fills this gap: we first investigate experimentally how to assess the quality of a BLE connection at runtime using information gathered from the radio. We then show how this information can be used to promptly blacklist poor channels and select a physical mode that sustains a high link-layer reliability while minimizing power consumption. We implement both mechanisms on two popular platforms and show experimentally that they allow to significantly improve the reliability of BLE connections, with a reduction in packet loss by up to 22 % compared to existing solutions.

Categories and Subject Descriptors

B.8 [**Performance and Reliability**]

General Terms

Design, Measurement, Performance, Reliability.

Keywords

BLE, PHY Mode, Adaptive Frequency Hopping.

1 Introduction

BLE is a low-power wireless technology that is increasingly used to create pervasive Internet of Things (IoT) applications, *e.g.*, in the smart health [13], smart city [12], and smart grid [8] domains. Many of these applications are safety critical and impose strict requirements on communication performance, especially with respect to the *reliability* of the data exchange; that is, BLE systems are expected to sustain a minimal packet loss and to ensure short transmission delays.

To increase the reliability of the data exchange, one can make use of information available on the BLE host to adapt BLE's connection parameters at runtime [17, 32]. This helps developers to *cope with packet loss* at the link layer by minimizing its impact on transmission delays. However, it *does not allow to prevent packet loss* at the link layer, which is necessary to maximize the reliability of a BLE connection. The BLE specification [4] foresees two mechanisms to improve the link-layer reliability of BLE connections: adaptive frequency hopping with *channel blacklisting* and *physical (PHY) mode adaptation*. Channel blacklisting allows to exclude poor-performing channels from being used for data exchange. PHY mode adaptation allows to trade receiver sensitivity and error correction capabilities (improving communication range and robustness) for a higher data rate.

The problem. While the primitives for channel blacklisting and PHY mode adaptation are fully embedded in the BLE specification, how these mechanisms should actually be used to improve link-layer reliability is not defined and left to the radio vendors. This results in some BLE platforms not implementing blacklisting at all (*e.g.*, the Nordic Semiconductor nRF52), and other platforms employing blacklisting strategies that were shown to be ineffective in real-world settings (*e.g.*, the Raspberry Pi 3) [32]. Similarly, the lack of guidance on how to use the various PHY modes has triggered several studies investigating their performance [3, 9, 33], but no concrete solution employing them to improve link-layer reliability at runtime has been proposed yet. How to effectively blacklist channels and adapt the PHY mode to minimize link-layer packet loss remains an open question.

The challenges. Link-layer transmissions may fail due to several reasons, such as weak signal strength, multipath fading, or external radio interference [5, 36]. All these factors decrease link-layer reliability, which leads to higher power consumption and transmission delays. To avoid this, one needs to detect these factors and react accordingly, which requires insights about the quality of the used BLE channels.

How to measure link quality? Understanding the quality of the overall BLE connection and individual channels requires to investigate in depth what kind of information can be gathered by the BLE radio at runtime and how this information may be used to blacklist individual channels or adapt the connection's PHY mode.

How to effectively blacklist channels? Based on the information available in the BLE radio, we need to promptly

International Conference on Embedded Wireless Systems and Networks (EWSN) 2020
17–19 February, Lyon, France
ISBN: 978-0-9949886-4-5

detect and blacklist channels with poor quality, while leaving enough channels available for a reliable communication.

How to determine the most suitable PHY mode? Based on the information available in the BLE radio, we need to select the PHY mode that allows to sustain a high link-layer reliability while minimizing the power consumption.

How to design a general solution? In order to be usable by a large fraction of BLE platforms on the market, we need to design and implement effective blacklisting and PHY mode adaptation mechanisms such that both techniques can be used cooperatively on any standard-compliant BLE device that allows link-layer access.

Contributions. We tackle each of these challenges and ultimately improve the link-layer reliability of BLE connections by designing an effective channel blacklisting and PHY mode adaptation mechanism for standard-compliant devices.

To this end, we first perform an extensive experimental campaign to gain a detailed understanding of the information provided by different link quality metrics available in the BLE radio. Such an experimental study fills the gap of existing research and serves as a reference to guide researchers and practitioners working on BLE's link-layer.

Based on this experimental study, we design and evaluate different channel blacklisting mechanisms and observe that passively monitoring the Packet Delivery Ratio (PDR) of individual channels is the most effective way to promptly detect poor channels in order to blacklist them. Our proposed channel blacklisting mechanism is hardware-independent and can be used in any standard-compliant BLE device that allows access to the BLE link layer.

We also design a PHY mode adaptation mechanism that monitors recent Signal-to-Noise Ratio (SNR) measurements to dynamically change the used PHY mode when necessary. The proposed mechanism allows to sustain a specified minimum link-layer reliability while limiting power consumption; all of this using standardized BLE primitives, such that it can be used by any device supporting multiple PHYs.

We implement both mechanisms on two popular hardware platforms: the Nordic Semiconductor nRF52 and the Raspberry Pi 3 (Pi3). As we only make use of standardized BLE primitives, our approach can be easily ported to other standard-compliant BLE platforms. Finally, we experimentally show that our mechanisms on the nRF52 cooperatively improve link-layer reliability by up to 22 %. Using our improvements, the nRF52 sustains a link-layer reliability of over 99 % in all experiments. Our improvements on the Pi3[1] increase the link-layer reliability of BLE connections by up to 10 % without incurring additional power consumption.

After providing some background on connection-based BLE in Sect. 2, this paper makes the following contributions:

- We experimentally study the link quality metrics that can be gathered by the BLE radio at runtime and discuss the insights that each metric provides (Sect. 3).
- We design an effective channel blacklisting mechanism that uses recent PDR measurements to detect and blacklist BLE channels with poor link quality (Sect. 4).

[1]Our improvements on the Pi3 are openly available at `https://github.com/seemoo-lab/internalblue/tree/master/examples`

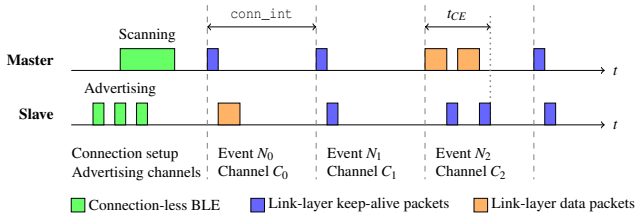


Figure 1. BLE connection between a master and a slave.

- We design an effective PHY mode adaptation mechanism using SNR readings that can sustain a given reliability while minimizing power consumption (Sect. 5).
- We implement the proposed mechanisms on two popular hardware platforms (Sect. 6) and evaluate their performance in different real-world scenarios (Sect. 7).

After summarizing related work in Sect. 8, we conclude this paper in Sect. 9 along with a discussion of future work.

2 Background on BLE Connections

BLE provides two communication modes: a *connectionless* and a *connection-based* mode. While the connectionless mode makes use of three advertisement channels to broadcast short data packets, the connection-based mode supports bidirectional data transfer. To enter the connection-based mode two devices use connection-less primitives to establish a BLE connection. In this connection, one device acts as master and the other as slave. Data exchange happens only during *connection events* (N_0 ... N_i), as shown in Fig. 1.

The *connection interval* (`conn_int`) specifies the time between the start of two consecutive connection events. During a connection event, master and slave exchange link-layer packets until both devices have no more data to send or the *maximum connection event length* (t_{CE}) is reached. These link-layer packets either carry application data (orange) or an empty payload to keep the connection alive (blue). Every connection event starts with a transmission by the master, to which the slave responds. If no data needs to be sent, only mandatory keep-alive packets are exchanged. The last link-layer packet in a connection event is always sent from slave to master, after which the connection event is closed and communication is resumed at the next connection event.

In the example shown in Fig. 1, the master starts connection event N_0 by sending a keep-alive packet to the slave, and the slave responds with a link-layer data packet carrying application data. In connection event N_1, master and slave have no data to send and therefore only exchange the mandatory keep-alive packets. During connection event N_2, the master transmits data to the slave. Because this data exceeds the maximum length of link-layer data packets, the master splits the data into two link-layer packets that are both acknowledged by the slave with a link-layer keep-alive packet.

AFH algorithm. At the beginning of every connection event, one out of 37 data channels is selected by BLE's adaptive frequency hopping (AFH) algorithm.[2] A new channel is chosen for every connection event and is used for all transmissions taking place during this event.

[2] With the release of BLE 5, BLE devices can use one of two possible AFH algorithms. As we show in Sect. 7, the employed AFH algorithm does not have significant impact on the link-layer reliability of a BLE connection.

Data channel selection. All 37 data channels are located in the license-free 2.4 GHz Industrial, Scientific, and Medical (ISM) band, which makes them likely to experience interference by other radio technologies using the same frequencies, *e.g.*, Wi-Fi, classic Bluetooth, or IEEE 802.15.4. Such interference, as well as multipath fading, may cause packet loss that decreases the link-layer reliability of BLE.

To mitigate the effects of link-layer packet loss, BLE radios may *blacklist* channels with poor quality by updating the *channel map* (C_{map}) of the connection at runtime. A connection's C_{map} specifies which data channels may be selected by the AFH algorithm; a channel disabled in the C_{map} will not be used for communication until being *whitelisted* (re-enabled) again. The BLE specification defines standardized commands that a master can use to update the C_{map} of an active connection; a slave is not allowed to change the C_{map}. However, when and how a master updates the C_{map} is not defined by the BLE specification. This leaves it up to developers to implement an effective blacklisting strategy, often leading to ineffective solutions on existing systems [32].

Link-layer ACK and flow control. To provide a reliable data exchange, the BLE link layer automatically handles packet *acknowledgment* (ACK) and flow control. This is achieved using a 1-bit *Sequence Number (SN)* and a 1-bit *Next Expected Sequence Number (NESN)* in each link-layer header. A link-layer packet is only successfully acknowledged when a BLE radio receives a NESN that is not equal to the SN of the transmitted packet. The link-layer packet is automatically retransmitted until a valid ACK is received.

PHY modes. Devices supporting BLE version 5 and above are able to choose one out of four physical (PHY) modes: the 1M, the 2M, the Coded S2, and the Coded S8 PHY [4].

The *1M PHY*, where the M stands for Megasymbols/s (Msym/s), is the original mode of BLE and is the only available PHY on BLE devices with a version below 5. The 1M PHY uses a physical modulation of 1 Msym/s, no symbol coding, and no Forward Error Correction (FEC). Similarly, the *2M PHY* mode uses no symbol coding and no FEC; however, it uses a physical modulation of 2 Msym/s leading to twice the data throughput compared to the 1M PHY [33].

The *Coded S2* or the *Coded S8 PHY* modes use symbol coding and FEC to reconstruct flipped bits in received packets, leading to a more robust communication [33]. Like the 1M PHY, both Coded PHYs use a physical modulation of 1 Msym/s. The Coded S2 PHY uses a symbol coding of 2, resulting in a maximum physical data rate of 500 kb/s. Likewise, the Coded S8 PHY uses a symbol coding of 8, resulting in a maximum physical data rate of 125 kb/s.

3 Experimental Study of BLE Reliability

We investigate next how to estimate the link quality of BLE links based on information gathered by the link layer on standard BLE radios. We first list available link-layer metrics in Sect. 3.1 and show their behavior in Sect. 3.2. We further evaluate the impact of different PHY modes on BLE reliability in Sect. 3.3 and discuss our findings in Sect. 3.4.

3.1 BLE Link Quality Metrics

We estimate the link quality of BLE data channels on master devices, which has several benefits that originate in the BLE specification [4]. First, the BLE master is usually less energy constrained (it is constantly powered or frequently charged) and can afford to probe the RF environment (*e.g.*, to measure the noise floor). Second, the master receives feedback on link-layer transmission within the same connection event, while on the slave this feedback is delayed (see Sect. 2). Third, only the master is allowed to black- and whitelist data channels used by the BLE connection.

To be compliant with the BLE specification [4], we *passively* estimate the link quality based on metrics available in the BLE link layer. Using an active approach that exchanges link-layer probe packets as done in the context of IEEE 802.15.4 [14] is not suitable for our approach, because sending additional probe packets is not compliant with the BLE specification. Furthermore, probe packets would introduce additional radio time and increase power consumption.

We focus on link-layer metrics that are hardware-agnostic and available on standard BLE radios allowing access to the link layer. We hence consider the following metrics:

Noise floor. We measure the noise floor of an individual data channel by probing the channel when the BLE radio does not exchange packets. This provides us with information on any nearby technology using the 2.4 GHz ISM band.

Signal-to-Noise Ratio (SNR). We calculate the SNR of every successfully received link-layer packet, by measuring a packet's Received Signal Strength Indicator (RSSI) and subtracting the current noise floor on the used data channel.

Packet Delivery Ratio (PDR). To measure the PDR, we adapt the work in [11] (based on IEEE 802.15.4) to be used in BLE connections. Because we are unable to arbitrarily probe individual channels, we use existing link-layer header fields to calculate the PDR of link-layer exchanges. The PDR measures the round-trip reliability of individual link-layer transmissions issued by a master and is computed as:

$$PDR = \frac{\#ACK(S \to M)}{\#TX(M \to S)}, \tag{1}$$

where $\#TX(M \to S)$ is the number of issued link-layer transmissions from master to slave and $\#ACK(S \to M)$ is the number of received valid link-layer acknowledgments. A link-layer acknowledgment from the slave is considered valid if it carries a valid CRC checksum and an updated NESN. As a result, every *individual* link-layer transmission can be either successful ($PDR = 100\%$) or unsuccessful ($PDR = 0\%$).

3.2 Measuring BLE Link Quality

We study the metrics from Sect. 3.1 in multiple scenarios to gain an understanding about the insights they provide.

Experimental setup. To have full control over the RF environment, we perform all experiments in a wireless testbed located in a laboratory. During all our tests, the lab is vacant and all Wi-Fi access points in proximity are deactivated, to limit the impact of Wi-Fi activity on our results.

We use two *Nordic Semiconductor nRF52840 DK (nRF52)* [24] exchanging data over a BLE connection: one device acts as master and the other one as slave. While the master uses its on-board PCB antenna for communication, we mechanically remove the PCB antenna of the slave to eliminate undesirable overlapping antenna effects and connect a programmable attenuator [21] as well as a 2.4 GHz

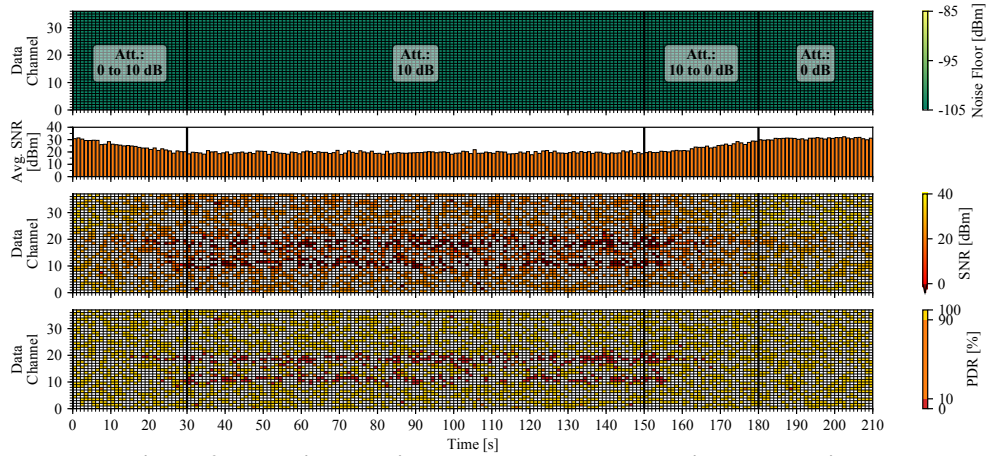


Figure 2. BLE link quality under gradually changing attenuation.

antenna (with a gain of 3 dB) to the external antenna connector of the nRF52. This setup allows us to manually reduce the power of the signal that is sent and received by the slave.

For this experimental campaign, master and slave have free line of sight and a distance of approx. 10 m. The master initiates a BLE connection with a connection interval of 50 ms and a slave latency of 0 and subscribes to a BLE GATT attribute on the slave. The slave issues a GATT indication on this attribute (with a link-layer length of 27 B) every 500 ms.

Both master and slave run the Zephyr OS [35], which comes with a standard-compliant BLE stack supporting BLE version 5. We adapt the link-layer implementation of the master to log the PDR and SNR of every link-layer transmission. The master also measures and logs the noise floor of each data channel after every connection event.

For these experiments, the connection uses the Channel Selection Algorithm (CSA) #2 of BLE 5 and the 1M PHY mode. The master does not use blacklisting, which is the default behavior of an nRF52 device running the Zephyr OS.

Methodology. We measure the behavior of the link quality metrics under changing antenna attenuation (Sect. 3.2.1), Bluetooth (Sect. 3.2.2), and Wi-Fi interference (Sect. 3.2.3).

The noise floor shown in Figs. 2 to 4 is the maximum noise floor recorded on each channel within every second. Likewise, the SNR plots in Figs. 2 to 4 show the average SNR within a second on every channel. When the master does not receive a link-layer ACK from the slave (and hence no RSSI value), the SNR of this unsuccessful packet exchange is discarded and these exchanges are marked in brown. In addition to the SNR per channel, we calculate the average SNR (*Avg. SNR*) across all used BLE data channels.

The PDR of individual link-layer transmissions is calculated as explained in Sect. 3.1. Figs. 2 to 4 show the average PDR within a second per channel. We classify each channels into *good*, *intermediate*, and *poor* based on its PDR, adapting the approach proposed by Srinivasan et al. [34]. Channels with a PDR $< 10\%$ are classified as poor; channels with a PDR between 10% and 90% are classified as intermediate; channels sustaining a PDR $\geq 90\%$ are classified as good.

Please note that, in Figs. 2 to 4, when a data channel is not used within a second, its PDR and SNR are marked in white.

3.2.1 *Changing Antenna Attenuation*

First, we investigate how the link quality of the BLE data channels is affected when the wireless signal is attenuated, *e.g.*, due to an increasing communication distance or obstacles blocking the line of sight. Using the programmable attenuator, we change the slave's antenna attenuation over time, mimicking a slave that moves away from the master.

Fig. 2 shows the measured noise floor, the average SNR of all used data channels, the SNR per data channel, and the PDR per data channel under changing attenuation over time. The effective attenuation of the BLE master starts at 0 dB (3 dB from the antenna gain minus a programmed attenuation of 3 dB) and, from time 0 to 30 s, the effective attenuation increases linearly to 10 dB. The attenuation stays at 10 dB for 120 s, before gradually going back to an effective attenuation of 0 dB, where it stays until the end of the experiment. We can see that the Avg. SNR reflects this change in attenuation and that the PDR of some data channels decreases significantly when a high attenuation is used.

The data in Fig. 2 shows that the noise floor measurements are not able to detect any of these link quality problems. As expected, the Avg. SNR captures the change in signal strength and the SNR measurement per channel detects when the master is not able to successfully receive a link-layer packet. SNR measurements, however, do not detect when a valid link-layer packet was received, but its CRC or NESN indicate a bad packet transmission. Only the PDR measurements are successfully able to capture all link-layer packet loss caused by a high antenna attenuation.

3.2.2 *Classic Bluetooth Interference*

Next, we measure BLE's link quality under external interference caused by co-located classic Bluetooth devices.

Like BLE, classic Bluetooth uses the 2.4 GHz ISM band and employs frequency hopping. However, classic Bluetooth uses a different modulation scheme with 1 MHz wide channels and is optimized for data throughput. In our experiments, we keep the effective attenuation at 0 dB and use two pairs of Pi3 sending Bluetooth RFCOMM packets of 1000 B length every 11.034 ms. This results in two classic Bluetooth connections, each exchanging packets at 725 kbit/s.

Fig. 3 shows the various link quality metrics in the presence of Bluetooth interference, generated for roughly 120 s,

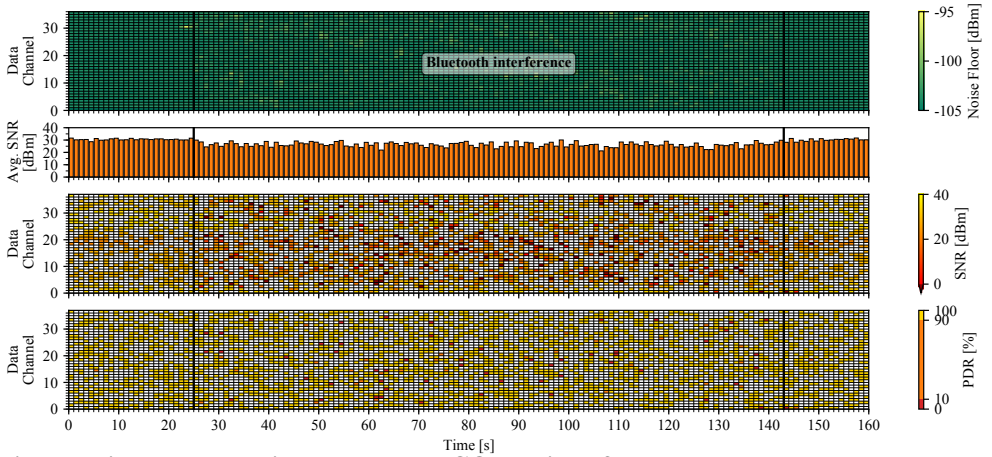


Figure 3. BLE link quality under classic Bluetooth RFCOMM interference on two co-located Bluetooth connections.

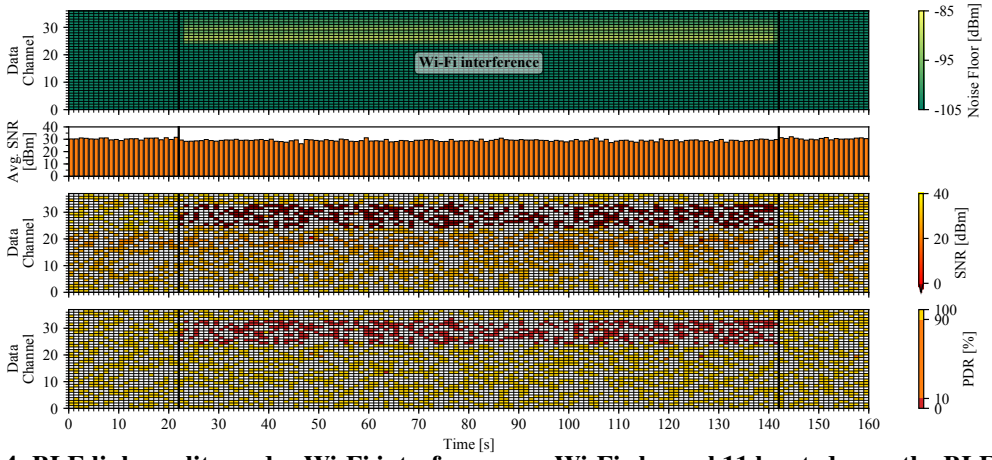


Figure 4. BLE link quality under Wi-Fi interference on Wi-Fi channel 11 located near the BLE slave.

starting 30 s from the beginning of the experiment. The data shows that classic Bluetooth interference decreases the reliability of the BLE connection on all data channels.

In this scenario, the noise floor measurements are barely able to detect the co-located radio communication. The Avg. SNR and the SNR per channel are able to detect the Bluetooth interference, but SNR readings may actually underestimate the link quality in this case, as several SNR values are very low ($SNR < 5$ dBm), even though a successful packet exchange is possible. Similar to the previous scenario, only the PDR is able to accurately capture link-layer packet loss caused by co-located Bluetooth RFCOMM interference.

3.2.3 Wi-Fi Interference

We measure next the link quality of the data channels in the presence of radio interference generated by co-located Wi-Fi devices. To generate the Wi-Fi interference, we use a Pi3 in our testbed, located near the BLE slave and run JamLab-NG [28] to generate Wi-Fi packets on Wi-Fi channel 11 with a length of 1500 B every 10 ms and a transmission power of 30 mW. In this experiment, we keep the effective antenna attenuation constant at 0 dB.

Fig. 4 shows the link quality metrics in the presence of Wi-Fi interference, generated for roughly 120 s, starting 30 s from the beginning of the experiment: overall, the Wi-Fi interference significantly decreases the link-layer reliability.

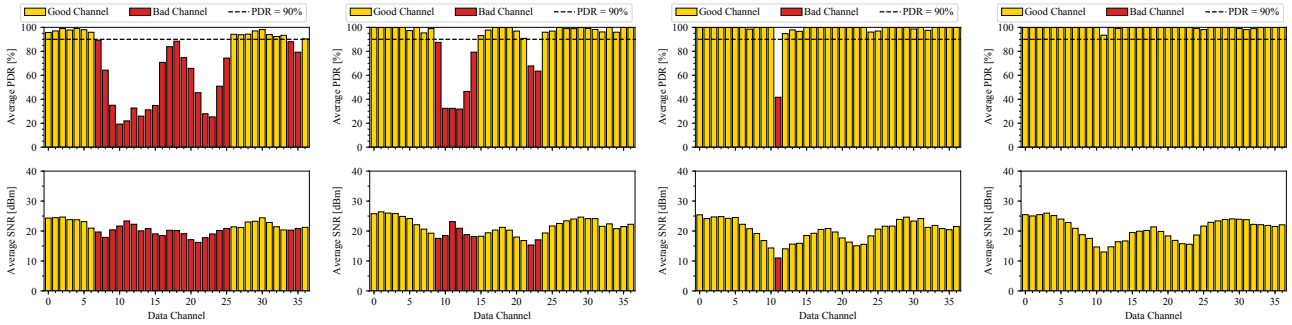
In this scenario, the noise floor measurements detect the Wi-Fi interference. Similar to Sect. 3.2.1, the SNR measurements detect when the master is not able to successfully receive link-layer packets, but do not detect failures due to invalid CRC or missing NESN updates. PDR measurements accurately detect link-layer errors due to Wi-Fi interference.

3.3 Using Different PHY Modes

We investigate how the used PHY mode of the BLE connection affects reliability. In contrast to existing work [33], we evaluate how individual data channels behave under various link conditions when using the different PHY modes.

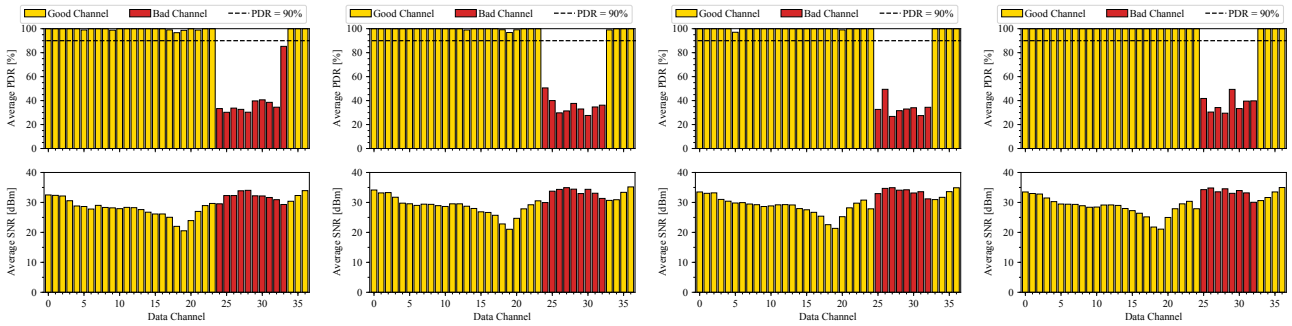
Antenna attenuation. We repeat the experiment from Sect. 3.2.1 with all four PHY modes of BLE 5. Fig. 5 shows the average PDR and average SNR per channel for the different PHYs measured while applying an effective attenuation of 10 dB. Data channels with an average PDR < 90 % are classified as *bad* (marked in red), while channels with an average PDR ≥ 90 % are classified as *good* (marked in yellow).

In this experimental scenario, the used PHY has a significant effect on the overall reliability of the BLE connection and the number of data channels classified as good. When using the fastest 2M PHY, 22 of the available data channels are bad, compared to the 8 bad channels when using the 1M PHY. Due to their use of FEC and symbol coding, the Coded



(a) 2M PHY Mode. (b) 1M PHY Mode. (c) Coded PHY (S2) Mode. (d) Coded PHY (S8) Mode.

Figure 5. Average PDR and SNR of all data channels using different PHYs and an antenna attenuation of 10 dB. *We see that the used PHY mode of the BLE connection significantly affects the overall link-layer reliability in this scenario.*



(a) 2M PHY Mode. (b) 1M PHY Mode. (c) Coded PHY (S2) Mode. (d) Coded PHY (S8) Mode.

Figure 6. Average PDR and SNR of all data channels using different PHYs under Wi-Fi interference on channel 11. *We see that the used PHY mode of the BLE connection does not significantly affect the overall link-layer reliability in this scenario.*

S2 and S8 PHYs are able to sustain a high PDR even for channels that have a low average SNR. Therefore, switching to a more robust PHY mode when experiencing a low signal strength significantly increases link-layer reliability.

Radio interference. Next, we repeat the experiment from Sect. 3.2.3 with all PHY modes of BLE 5. Fig. 6 shows the average PDR and average SNR per channel measured when Wi-Fi interference was generated near the BLE slave.

Unlike the previous experiment, the used PHY mode does not significantly improve the reliability of the BLE connection and the number of good data channels under external interference. All four PHYs experience similar link quality problems on the data channels affected by the co-located Wi-Fi interference on Wi-Fi channel 11. The 2M PHY provides an overall PDR of 83 %, while the Coded S8 PHY leads to an overall PDR of 86 %. Switching the PHY from the 2M PHY to the most robust Coded S8 PHY would only increase the PDR of the BLE connection by 3 %. If we would blacklist all poor channels in this scenario, instead, the BLE connection would have an overall PDR $> 99\,\%$ for all PHYs. Note that, in contrast to the data shown in Fig. 5, link-layer packet loss caused by radio interference does not cause the average SNR on the affected data channels to drop.

3.4 Lessons Learned

Based on our results, we draw the following conclusions.

Noise floor. Noise floor measurements of data channels successfully detect link-layer loss caused by external radio interference (Sect. 3.2.2 and 3.2.3). However, the noise floor fails to capture link-layer loss caused by a weak signal strength, *e.g.*, due to a large communication distance (Sect. 3.2.1).

SNR per data channel. SNR measurements detect link-layer loss on individual channels, when link-layer packets are missing and therefore no SNR can be calculated. SNR readings, however, miss link problems indicated by an invalid CRC or missing update of NESN. Furthermore, classifying channels based on recent SNR readings may result in an inaccurate classification, as some successful link-layer exchanges may have an SNR below 5 dBm that would indicate a problem based on the measurements shown in Fig. 5.

Average SNR. The Avg. SNR across all used channels accurately captures the signal strength of the BLE connection. The Avg. SNR can be used to detect when a BLE connection is experiencing link-layer errors due to weak signal, *e.g.*, caused by a long communication distance. Furthermore, the Avg. SNR is not significantly affected by external radio interference, as shown in Fig. 3 and Fig. 4. Therefore, the SNR can be used to detect when the connection's signal strength is the problem for link-layer loss, as we show in Sect. 5.

Packet Delivery Ratio. As expected, PDR readings accurately detect any link-layer packet loss across all of our experiments. This makes PDR the most suitable metrics to detect and blacklist poor data channels, as we show next.

4 BLE Channel Blacklisting

Using our findings from Sect. 3, we design an effective channel blacklisting mechanism that *passively* monitors the individual data channels and classifies them into *good* and *bad*. Good channels experience zero or few link-layer er-

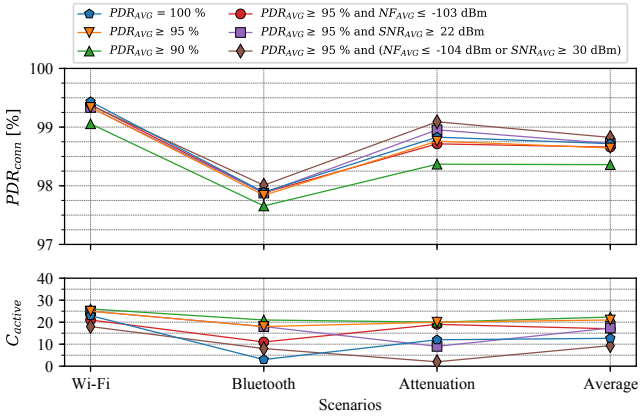


Figure 7. PDR of the BLE connection (PDR_{conn}) and number of active channels (C_{active}) at the end of each experiment for different filtering mechanisms. *Overall, the $PDR_{AVG} \geq 95\%$ approach provides the best trade-off between reliability (PDR_{conn}) and data channel usage (C_{active}).*

rors, while bad channels have an insufficient link quality and should not be used for communication. Similar to the work in [11], our goal is not to exactly estimate the link quality of a channel, but to detect when a channel is experiencing problems (*e.g.*, due to radio interference or weak signal strength) in order to blacklist it. To this end, we study the best way to detect bad channels at runtime (Sect. 4.1), when and how to blacklist channels (Sect. 4.2), as well as when to whitelist the channels of an active BLE connection (Sect. 4.3).

4.1 Detecting Bad Channels at Runtime

To find the most suitable way to detect bad data channels, we use the experimental traces from Sect. 3.2 and investigate the performance of different channel classification approaches. For every scenario, we simulate the behavior of one channel classification approach and calculate the overall PDR of the BLE connection (PDR_{conn}) and the number of active channels (C_{active}) at the end of each experiment.

Therefore, we step through each experimental trace and increase $\#TX(M \rightarrow S)$ by one for every issued link-layer packet. If the transmission was successful ($PDR = 100\%$), we also increase $\#ACK(S \rightarrow M)$ by one. With $\#TX(M \rightarrow S)$ and $\#ACK(S \rightarrow M)$, we calculate PDR_{conn} using Eq. 1. While stepping through an individual trace, we simulate the behavior of different channel classification approaches. Whenever the used classification approach detects a bad data channel, we mark it as blacklisted and do not count any subsequent link-layer exchanges on this channel. This simulates the actual channel blacklisting behavior that we implement on standard BLE devices, which is described in Sect. 6.

Fig. 7 shows PDR_{conn} and C_{active} of the six investigated channel classification approaches in three different experimental scenarios. All investigated approaches calculate the moving average (PDR_{AVG}) of recent PDR values using a window length W_{PDR}, as only the PDR detects all link-layer failures independent of their cause (see Sect. 3).

The approach $PDR_{AVG} = 100\%$ aggressively blacklists individual data channels on their first link-layer packet loss. The $PDR_{AVG} \geq 95\%$ approach uses a $W_{PDR} = 20$ and blacklists a channel when its PDR_{AVG} drops below 95 %. Similarly, $PDR_{AVG} \geq 90\%$ uses a $W_{PDR} = 10$ and a threshold of 90 %. The chosen W_{PDR} for every approach is the minimum window that still allows to measure the PDR_{AVG} in the necessary resolution. To measure the PDR of a channel with a resolution of 5%, which is necessary for our $PDR_{AVG} \geq 95\%$ approach, we need at least a $W_{PDR} = 20$.

Out of these classification approaches, $PDR_{AVG} \geq 95\%$ is able to sustain an average $PDR_{conn} = 98.6\,\%$ and a minimum $PDR_{conn} = 97.8\,\%$ in all three scenarios. Although the $PDR_{AVG} = 100\%$ provides a slightly higher average $PDR_{conn} = 98.7\,\%$, its aggressive behavior leads to a significantly lower C_{active}, which may result in a terminated BLE connection when sudden radio interference appears.

In addition to the first three approaches using only PDR measurements, we investigate if additional information about the average noise floor (NF_{AVG}) or SNR (SNR_{AVG}) improves channel blacklisting. We use the $PDR_{AVG} \geq 95\%$ approach and combine it with NF_{AVG} and SNR_{AVG}, leading to three additional classification approaches; all of them use a $W_{PDR} = 20$ for filtering PDR_{AVG}, NF_{AVG}, and SNR_{AVG}. The individual configurations for these approaches were chosen by running every scenario with every threshold combination and selecting the best combination for the average case.

Overall, the $PDR_{AVG} \geq 95\%$ approach is the most suitable channel classification approach for blacklisting. With its ability to sustain a PDR_{AVG} above 97.8 % while providing an average C_{active} of 21 channels, $PDR_{AVG} \geq 95\%$ provides the most suitable trade-off between link-layer reliability and number of active channels. Based on our experiments in Sect. 3, we see that sudden Wi-Fi interference may affect up to 10 subsequent BLE data channels. Sustaining a $C_{active} > 10$, as the $PDR_{AVG} \geq 95\%$ is able to do, mitigates BLE connection loss due to sudden co-located Wi-Fi traffic.

4.2 Blacklisting BLE Data Channels

Next, we discuss the necessary steps between detecting a bad channel and excluding it from further communication.

Issuing a channel map update. As detailed in Sect. 2, only the BLE master can update the used channel map (C_{map}) of a BLE connection by sending an `LL_CHANNEL_MAP_IND` request to the slave. This request carries a 37-bit data C_{map} that indicates if a channel is used in the connection or not. If the corresponding bit of a data channel is set, the channel is actively used for communication, otherwise it is not and is hence *blacklisted*. A blacklisted channel stays inactive until another `LL_CHANNEL_MAP_IND` request *whitelists* (re-enables) the channel. A slave receiving this request cannot negotiate and needs to adhere to the received information.

In our approach, we update C_{map} as soon as we detect that a data channel is bad. If the master has recently issued a C_{map} update that has not yet been acknowledged by the slave, the master waits for this ACK and then immediately issues a new C_{map} update. This approach may create a separate `LL_CHANNEL_MAP_IND` request for every blacklisted data channel, resulting in additional radio time. However, sending multiple `LL_CHANNEL_MAP_IND` requests does not significantly increase the power consumption, as shown in Sect. 7.

Mandatory update delay. According to the BLE specification [4], a new C_{map} only takes effect after a mandatory

delay of at least 6 connection events. This means when a BLE master detects a bad channel and therefore issues an LL_CHANNEL_MAP_IND request at connection event N, the new C_{map} is used starting from connection event $N+6$. Hence, a channel already marked as blacklisted may be used in another connection event, before being actually disabled. To maintain interoperability with standard-compliant BLE devices, however, we adhere to this mandatory delay.

Clearing information about blacklisted channels. As soon as a channel is blacklisted, we are not able to estimate its link quality using our approach. Therefore, any link-layer information of blacklisted channels needs to be cleared, as it may have expired. Only when a data channel is whitelisted, we collect fresh information to accurately estimate its quality.

4.3 Whitelisting BLE Data Channels

As mentioned above, when a channel is blacklisted all of its link quality information may be outdated. Hence, we cannot observe when a blacklisted channel turns good in order to whitelist it. One possible solution for this is to exchange additional probes on blacklisted channels to measure their PDR. This, however, is not compliant to the BLE specification and would introduce an unnecessary overhead. Another approach is to whitelist data channels on a time basis, *i.e.*, re-enable blacklisted channels several seconds after being blacklisted. Finding a suitable timeout, however, largely depends on the actual cause of current link-layer packet loss.

In this work, we trigger a whitelisting whenever the number of active channels in the current data channel map drops below the minimum number of data channels (C_{min}).[3] In particular, we re-enable all 37 data channels and probe them using regular connection events to measure their PDR_{AVG}: this allows us to get the most accurate link quality estimation of the usable channels. Using this approach, whenever the master blacklists a bad channel, it first checks the number of used data channels: if the number of used channels drops below C_{min}, the master performs the following procedure.

Updating the connection interval. During whitelisting, we re-enable data channels with unknown link quality, which can cause link-layer errors leading to high latencies [32]. To ensure a reliable data exchange during whitelisting, we adapt the used BLE connection interval to a faster setting before re-enabling any data channel. During whilelisting, we temporarily overprovision (O_{WL}) the BLE connection by:

$$O_{WL} = \lceil C_{max}/C_{min} \rceil, \tag{2}$$

where C_{min} is the number of channels used before initiating a whitelisting and C_{max} is the number of channels that are probed during whitelisting. Since we enable all 37 data channels during whitelisting, $C_{max} = 37$ for our approach.

The faster connection interval ($conn\_int_{WL}$) temporarily used during whitelisting is calculated as:

$$conn\_int_{WL} = conn\_int / O_{WL}, \tag{3}$$

where $conn\_int$ is the connection interval used before whitelisting and O_{WL} is the necessary overprovisioning calculated by Eq. 2. For example, if we use a $C_{min} = 10$ and a $C_{max} = 37$, we need to change the connection interval to be $O_{WL} = 4$ times faster to mitigate the effects of potential link-layer errors on BLE transmission delays during whitelisting.

Similar to the channel map update, updating the connection interval requires a delay of at least 6 connection events between issuing and using the new connection interval [4].

Probing data channels. After the temporary $conn\_int_{WL}$ has been set, we issue a data channel map update re-enabling all 37 BLE data channels to probe their link quality. During this probing phase, we use ordinary BLE connection events to measure the PDR_{AVG} and link quality of the individual data channels, as discussed in Sect. 4.1. In this phase, however, we temporarily disable channel blacklisting to get the most accurate estimation of each channel's link quality.

This probing phase lasts for t_{probe}, in which we probe every data channel $S_{channel}$ times. t_{probe} is calculated as:

$$t_{probe} = S_{channel} \cdot C_{max} \cdot conn\_int_{WL}, \tag{4}$$

where $S_{channel}$ is the number of samples per channel, C_{max} is the number of BLE data channels, and $conn\_int_{WL}$ is the connection interval used during probing, calculated with Eq. 3.

After the probing phase has ended, we blacklist any poor channels and revert back to the original $conn\_int$, resuming communication with the original BLE connection parameters and a new channel map with only reliable data channels.

5 BLE PHY Mode Adaptation

We design a PHY mode adaptation mechanism allowing BLE devices to sustain a specified link-layer reliability while limiting unnecessary power consumption. Specifically, the proposed adaptation mechanism *passively* monitors recent SNR measurements to detect when a change is necessary.

As discussed in Sect. 3, the average SNR over used data channels accurately captures the BLE signal strength and is not significantly affected by external radio interference. As a result, our proposed PHY mode adaptation mechanism is independent from the blacklisting mechanism presented in Sect. 4, making both mechanisms easily portable to other hardware platforms. In case a device does not support different BLE PHY modes, such as the Raspberry Pi 3, we can use our channel blacklisting mechanism to improve reliability. On devices supporting different PHY modes, both mechanisms work together in parallel to improve reliability while minimizing power consumption, as we show in Sect. 7.

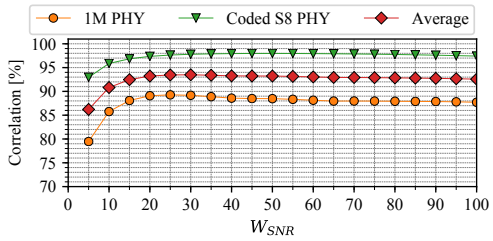
As the BLE specification allows slaves to change the used PHY mode, any BLE device may make use of our proposed PHY mode adaptation to increase link-layer reliability.

Next, we discuss how to filter recent SNR readings (Sect. 5.1), choose a suitable PHY mode (Sect. 5.2), and adapt the used PHY mode of a BLE connection (Sect. 5.3).

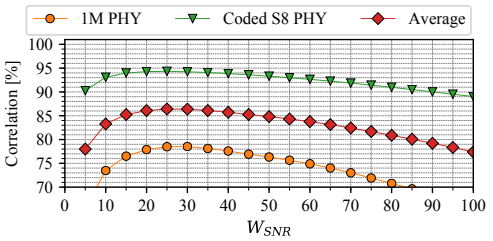
5.1 Filtering SNR Readings

We use a moving average filter on recent SNR values on all used data channels to predict the SNR of future link-layer exchanges. To find a suitable window length W_{SNR} for our SNR filter, we reuse the experimental setup employed to derive Fig. 2 for different antenna attenuation settings and PHY modes. We step through these traces and investigate which W_{SNR} leads to the best predictions of the average SNR of subsequent link-layer exchanges. For this evaluation, we choose a prediction window of 100 future link-layer exchanges.

[3] Since BLE 5, a slave can change the minimum number of data channels by sending an LL_MIN_USED_CHANNELS_IND request to the master. Older BLE devices do not have this possibility and only mandate that $C_{min} \geq 2$.

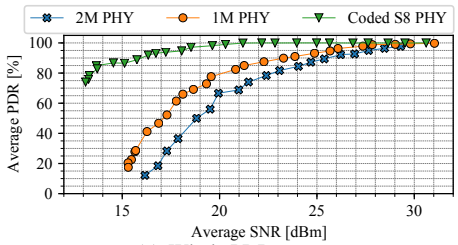


(a) Attenuation = 10 dB.

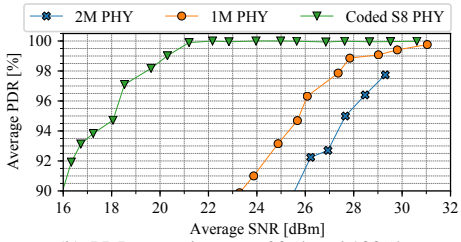


(b) Attenuation = 20 dB.

Figure 8. Correlation of the predicted and actual future SNR for different observation window lengths (W_{SNR}). *The data suggest that using a W_{SNR} of 25 provides the best prediction of future SNR values across both scenarios.*



(a) Whole PDR range.



(b) PDR range between 90 % and 100 %.

Figure 9. Relationship between average PDR and the average SNR of a BLE connection for different PHY modes.

Fig. 8 shows the correlation between predicted average SNR, filtered with different W_{SNR}, and the actual future average SNR. The data shows that using a $W_{SNR} = 25$ to filter recent link-layer exchanges provides the most accurate estimation of future SNR across the investigated traces.

5.2 Choosing a Suitable PHY Mode

Using the filtered SNR readings, a device can choose the most suitable PHY mode that sustains a specified minimum link-layer reliability (PDR_{min}) while limiting unnecessary power consumption. To find the relationship between average SNR and PDR, we re-use the data captured in Sect. 5.1 for different attenuation settings. We process each trace and calculate the average PDR and the average SNR during the time of constant attenuation (time 30 to 150 s in Fig. 2).

Fig. 9 shows the relationship between average SNR and PDR for three different PHY modes. We only investigate the Coded S8 PHY because the symbol coding used by the Coded PHY (either S2 or S8) is decided by every individual BLE device and cannot be negotiated at runtime. Therefore, we fix symbol coding for the Coded PHY mode to S8 on our devices, as this provides the highest reliability (see Fig. 5).

As expected, the Coded S8 PHY mode sustains the highest average PDR for a given SNR. For example, while the Coded S8 PHY provides an average PDR above 98.5 % for an SNR of 20 dBm, 1M and the 2M PHY only sustain an average PDR of approx. 82 % and 66 %, respectively. This increased reliability, however, comes with the cost of additional power consumption caused by the longer radio times of the Coded S8 PHY. Using the setup from Sect. 3.2, a slave using the Coded S8 PHY has an average power consumption of 581.79 μA, while the same slave using the 1M or the 2M PHY consumes 407.93 μA and 397.07 μA, respectively.

Our measurements show that a slave using the 1M PHY consumes only slightly more power (approx. +2.73 %) compared to using the 2M PHY, but the 1M PHY provides a significantly higher link-layer reliability across all our tests. We therefore argue that using the 2M PHY does not pay off in application scenarios where devices need to sustain a given PDR_{min} on a constrained energy budget and data throughput is not an issue. In such applications, one should make use of the data shown in Fig. 9 to decide between the Coded S8 and the 1M PHY mode based on the average experienced SNR.

5.3 Adapting the Used PHY Mode

Using a specified PDR_{min}, our PHY mode adaptation selects a SNR threshold (SNR_{PHY}) based on the data in Fig. 9. When the average SNR is $\geq SNR_{PHY}$, our mechanism uses the 1M PHY to conserve energy. When the average SNR is below SNR_{PHY}, our mechanism chooses the Coded S8 PHY to sustain PDR_{min}. If a device operates at approximately SNR_{PHY}, it may continuously switch between the two PHY modes, which may lead to additional power consumption or a PDR below PDR_{min}. To mitigate such behavior, we switch to the Coded S8 PHY when the average SNR drops below SNR_{PHY}, but only switch to the 1M PHY, when the average SNR $\geq SNR_{PHY} + SNR_{offset}$. To find a suitable SNR_{offset} for our PHY adaptation mechanism, we use the traces collected in Sect. 5.2 and investigate the number of PHY mode adaptations (N_{adapt}) for different SNR_{offset} and attenuations when using the filtering mechanism from Sect. 5.1. Fig. 10 shows that even a SNR_{offset} = 1 dBm mitigates unnecessary PHY mode adaptations across all our tests.

To adapt the used PHY mode, we use the standardized *PHY update procedure* defined by the BLE specification [4]. Similar to the channel map update, adapting the PHY mode requires a mandatory delay of 6 connection events.

6 Implementation

To show the portability of our mechanism, we implement our approaches on two popular platforms: the Nordic Semiconductor nRF52 (Sect. 6.1) and Raspberry Pi 3 (Sect. 6.2).

6.1 Nordic Semiconductor nRF52

We run the Zephyr OS [35] on the `nrf52840` chip, which embeds an ARM Cortex-M4F processor providing 1024 kB

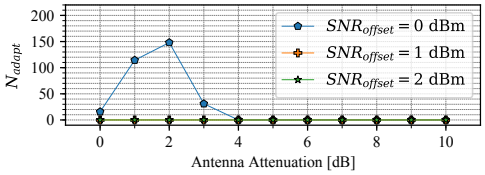


Figure 10. Number of PHY mode adaptations (N_{adapt}) for different thresholds and antenna attenuation values. *Even an $SNR_{offset} = 1\,dBm$ eliminates unnecessary PHY mode adaptations (N_{adapt}) on stable links.*

of flash and 256 kB of memory, as well as a standard-compliant BLE 5 radio supporting all four PHY modes. Although we use the `nrf52840` chip in our experiments, our code runs on all chips that are part of the nRF52 series.

Zephyr already provides a fully standard-compliant BLE communication stack that allows link-layer access. We modify the existing BLE stack by extending the `lll_conn_isr_rx` function of the link-layer implementation. This function is called after every link-layer packet reception and provides information about the packet's data channel, PDR, and RSSI. After every successfully received packet, we probe the noise floor of the used channel to calculate the packet's SNR.

Our channel blacklisting mechanism is notified about the used channel and PDR of every link-layer packet exchange by a callback in `lll_conn_isr_rx`. Using this information, our blacklisting mechanism detects bad channels and blacklists them, using the $PDR_{AVG} \geq 95\%$ channel classification, a $C_{min} = 10$, and a $S_{channel} = 10$, as described in Sect. 4.

Similarly, our PHY mode adaptation mechanism receives a new SNR readings after every successful packet reception via a callback in `lll_conn_isr_rx`. Using recent SNR values, our PHY mode adaptation chooses the best PHY that sustains a $PDR_{min} > 99\%$ while minimizing power consumption, following the approach discussed in Sect. 5.

6.2 Raspberry Pi 3

The Pi3 uses Broadcom's `BCM43430A1` chip for BLE communication [26]. This proprietary radio chip is closed source and autonomously handles all BLE's link-layer functionality, as well as classic Bluetooth and Wi-Fi communication. With *InternalBlue* [19], the firmware of most Cypress and Broadcom chips, including the `BCM43430A1`, can be analyzed and even patched with custom Assembly code.

To use our channel blacklisting mechanism on the Broadcom BLE radio, we use InternalBlue to analyze the handling of BLE link-layer packets in this radio chip. We detect that the function `_connTaskRxDone` is called upon reception of any BLE link-layer packet. In this function, we can retrieve the used data channel and PDR of the most recent link-layer exchange, which we use for blacklisting. We extend the `_connTaskRxDone` function by patching the radio's firmware to send a custom Host Controller Interface (HCI) event, containing the most recent data channel and PDR, over the standardized HCI to the BLE host after every link-layer transmission. We parse these HCI packets in InternalBlue and perform channel blacklisting as described in Sect. 4. Whenever we need to update the data channel map of the connection, we issue a standardized `Host_Channel_Classification` command via the HCI to the BLE radio. Using the standardized HCI, our approach is still fully compliant to the BLE specification.

The Broadcom radio already implements basic BLE channel blacklisting and radio co-existence mechanism that run autonomously in the background. Our channel blacklisting extends the channel blacklisting on the Pi3, but does not disable these existing link-quality improvements.

As the `BCM43430A1` radio only supports the 1M PHY, we do not implement PHY mode adaptation on the Pi3.

7 Evaluation

In this section, we experimentally study the performance of the proposed blacklisting mechanism alone (Sect. 7.1) and in parallel to the PHY mode adaptation scheme (Sect. 7.2).

Experimental setup. For this evaluation, we use an experimental setting similar to the one described in Sect. 3. To evaluate the power consumption of the BLE slave, we measure the slave's average current draw (I_{Slave}) using D-Cube [27]. We focus on the consumption of the slave, as the latter usually operates on a tight energy budget (see Sect. 2).

To measure the overall link-layer reliability (*PDR*), we parse the link-layer logs of the used master devices and calculate the PDR across the whole BLE connection.

Experimental scenarios. Following the methodology used in Sect. 3.2, our experiments start with an effective antenna attenuation of 0 dB and without any external radio interference. After the BLE connection is established, we wait for 60 s before either changing the attenuation or introducing radio interference. We gradually vary the effective attenuation of the slave antenna over 30 s from 0 dB to either 10 dB or 20 dB. We keep the attenuation at this setting for 600 s before gradually reverting back to 0 dB over 30 s. For scenarios investigating the reliability under interference, we start interference for a duration of 600 s, as described in Sect. 3.2.

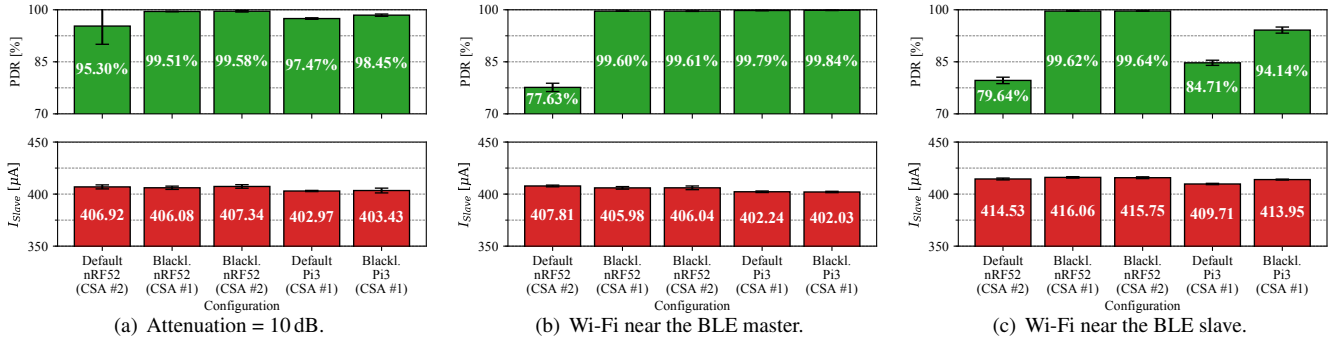
7.1 Evaluating BLE Channel Blacklisting

We evaluate the performance of our proposed channel blacklisting mechanism by measuring *PDR* and I_{Slave} in three different experimental scenarios and on two different hardware platforms, as described in Sect. 6. During this experiments, we disable our PHY mode adaptation mechanism.

Fig. 11 shows the average *PDR* and I_{Slave} for five different blacklisting mechanisms in three scenarios, where every experiment was repeated 5 times. Two bars show the default behavior of the nRF52 (*Default nRF52*) and the Raspberry Pi 3 (*Default Pi3*). The two bars named *Blackl. nRF52* show the performance of our channel blacklisting mechanism implemented on the nRF52 master for Channel Selection Algorithm (CSA) #1 and CSA #2 . The bar named *Blackl. Pi3* shows the Pi3 using our blacklisting mechanism.

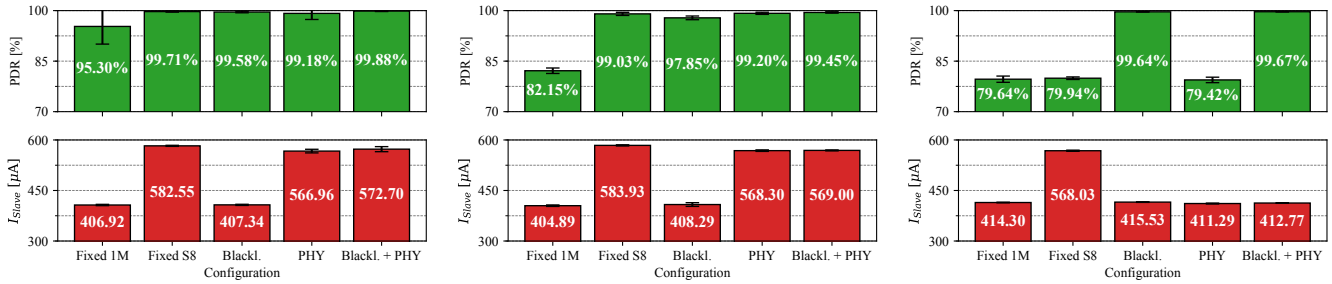
Overall, we see that our proposed channel blacklisting mechanism significantly increases the reliability (*PDR*) without introducing additional power consumption (I_{Slave}). Compared to the default behavior of the nRF52 and Pi3, our mechanism improves the *PDR* by up to +22 % and +10 %, respectively. The used CSA does not significantly impact the link-layer reliability, as shown by the similar performance of *Blackl. nRF52(CSA #1)* and *Blackl. nRF52(CSA #1)*.

In case of an antenna attenuation of 10 dB (Fig. 11(a)), our mechanism on the nRF52 sustains a PDR over 99 %.



(a) Attenuation = 10 dB. (b) Wi-Fi near the BLE master. (c) Wi-Fi near the BLE slave.

Figure 11. Link-layer reliability (*PDR*) and the slave's average current consumption (I_{Slave}) of different blacklisting mechanisms in three scenarios. *The connection was either using Channel Selection Algorithm #1 (CSA #1) or CSA #2.*



(a) Attenuation = 10 dB. (b) Attenuation = 20 dB. (c) Wi-Fi near the BLE slave.

Figure 12. Link-layer reliability (*PDR*) and the slave's average current consumption (I_{Slave}) for five different configurations. *Running both mechanisms in parallel (Blackl. + PHY) provides a PDR > 99% while minimizing power consumption.*

On the Pi3, our mechanism improves the PDR by over 1 %, reaching an overall PDR of 98.45 %. The reason for the slightly lower PDR on the Pi3 compared to the nRF52 is the proprietary radio coexistence mechanism constantly running on the Pi3, which autonomously re-enables data channels.

Under Wi-Fi interference near the BLE master (Fig. 11(b)), the default behavior of the Pi3 sustains a *PDR* above 99 %. This matches findings by Spörk et al. [32] showing that the Pi3 likely uses noise floor measurements to proactively blacklist interfered channels. Nevertheless, our passive blacklisting mechanism sustains a comparable *PDR* in this setting. Under Wi-Fi interference near the BLE slave (Fig.11(c)), the default blacklisting mechanism of the Pi3 does not detect link-layer errors and is only able to sustain a PDR of 84.71 %. Our proposed blacklisting mechanism, instead, increases the *PDR* on the Pi3 by almost 10 %. Overall, in all three experimental scenarios, our blacklisting mechanism on the nRF52 is able to sustain a PDR above 99 %.

7.2 Evaluating PHY Mode Adaptation

To evaluate the performance of our proposed PHY mode adaptation mechanism on the nRF52, we re-run the experiments from Sect. 7.1 with five different configurations: no blacklisting and a fixed 1M PHY mode (*Fixed 1M*); no blacklisting and a fixed Coded S8 PHY mode (*Fixed S8*); and our blacklisting mechanism and a fixed 1M PHY mode (*Blackl.*); no blacklisting and only our PHY mode adaptation (*PHY*); running both proposed mechanisms in parallel (*Blackl. + PHY*). We configure the PHY mode adaptation to sustain a minimum PDR of 99 %, as described in Sect. 6.

Adapting the PHY mode improves the PDR when the signal strength drops (shown in Figs. 12(a) and 12(b)). Changing to a more reliable PHY mode, however, introduces an additional current consumption (I_{Slave}) of approx. +38.9 % on the BLE slave. We can also see in Fig. 12(c) that our PHY mode adaptation does not adapt the PHY mode when co-located Wi-Fi interference is introducing link-layer errors. This shows that our blacklisting mechanism and PHY mode adaptation mechanisms work in parallel and do not conflict.

Our PHY adaptation mechanism alone is able to sustain a PDR of 99% while minimizing power consumption when possible. This is shown by the similar PDR and I_{Slave} sustained by our PHY mode adaptation compared to the use of a fixed Coded S8 PHY (Fixed S8).

Overall, we see that it is best to use both of our improvements in parallel, as they do not conflict and as they can sustain, together, a PDR $\geq$ 99% across all of our experiments.

8 Related Work

BLE reliability. Several works focus on measuring [6, 20, 30, 37] or improving [7, 23] the coexistence of BLE with other radio technologies in the 2.4 GHz ISM band, but none of them detects and blacklists poor channels at runtime. Other studies have investigated the AFH algorithms of BLE using mathematical models and simulations, but did not make use of channel blacklisting [1, 2] or employed hardware-specific features to detect and blacklist poor channels [31]. A few works use standard information available in the BLE host to estimate link quality [17] and improve timeliness by adapting connection parameters [32]. However, these approaches only *counteract* the effects of link-layer loss and are not able to improve reliability. In this work, we use link-layer metrics available in standard BLE radios to reduce packet loss at the link layer and increase reliability.

BLE 5 PHY modes. A few works study the impact of different PHY modes in connection-less [3, 9, 25] and connection-based [33] BLE systems. While the focus of these works is on comparing the achievable performance only, our current work investigates how to dynamically adapt the PHY mode used by BLE connections at runtime to sustain a high reliability while minimizing the power consumption.

Other low-power radio technologies. Besides works simulating Classic Bluetooth's frequency hopping scheme [22], most of the research on channel diversity in low-power radios has focused on IEEE 802.15.4. This body of literature, however, does not use channel blacklisting [11, 15], statically selects a channel map [36], or periodically probes the quality of channels [10, 16]. Other studies change the data channel used for infrequent transmissions (1 packet/5 minutes) when the channel quality drops over long periods [29] or use machine learning to predict the link quality of IEEE 802.15.4 channels [18]. Unlike these works, we use link-layer information available in standard BLE radios to promptly blacklist bad channels and adapt the PHY mode.

9 Conclusion and Future Work

The BLE specification foresees mechanisms to improve link-layer reliability, but does not provide guidelines on how to efficiently design and implement them. As a result, BLE devices may use proprietary solutions that provide poor link-layer reliability in real-world settings. We propose two standard-compliant mechanisms that use existing BLE primitives to increase link-layer reliability while minimizing power consumption, significantly outperforming the default solutions of popular BLE platforms. Our next steps include (i) porting the proposed improvements to other BLE platforms, such as Samsung Galaxy S10 and iPhone 11 smartphones, as well as (ii) the dynamic adaptation of BLE's transmission power to further minimize power consumption.

Acknowledgments

This work has been performed within the LEAD project "Dependable Internet of Things in Adverse Environments" funded by Graz University of Technology and in the context of the LOEWE centre emergenCITY. This work has been funded by the DFG as part of SFB 1053 MAKI, and the BMBF and the State of Hesse within ATHENE. This work was also partially funded by DFG within cfaed and the SCOTT project. SCOTT (`http://www.scott-project.eu`) has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This joint undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway. SCOTT is also funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between May 2017 and April 2020. More info at `https://iktderzukunft.at/en`.

10 References

[1] M. Al Kalaa et al. Selection probability of data channels in Bluetooth Low Energy. In *Proc. of the* 11^{th} *IWCMC Conf.* IEEE, 2015.

[2] M. O. Al Kalaa et al. Evaluating bluetooth low energy in realistic wireless environments. In *Proc. of the* 12^{th} *IWCMC Conf.* IEEE, 2016.

[3] B. Al Nahas et al. Concurrent Transmissions for Multi-Hop Bluetooth 5. In *Proc. of the* 16^{th} *EWSN Conf.*, 2019.

[4] Bluetooth SIG. Bluetooth Core Specification v5.0, 2018.

[5] C. A. Boano and K. Römer. External radio interference. In *Radio Link Quality Estimation in Low-Power Wireless Networks*. 2013.

[6] W. Bronzi et al. Bluetooth low energy performance and robustness analysis for inter-vehicular communications. *Ad Hoc Networks*, 2016.

[7] O. Carhacioglu et al. Time-domain cooperative coexistence of BLE and IEEE 802.15.4 networks. In *Proc. of the* 28^{th} *PIMRC Symp.*, 2017.

[8] M. Collotta et al. A Solution Based on Bluetooth Low Energy for Smart Home Energy Management. *Energies*, 8, 2015.

[9] M. Collotta et al. Bluetooth 5: A concrete step forward toward the IoT. *IEEE Communications Magazine*, (7), 2018.

[10] P. Du et al. Adaptive time slotted channel hopping for wireless sensor networks. In *Proc. of the* 4^{th} *CEEC Conf.*, 2012.

[11] R. Hermeto et al. Passive Link Quality Estimation for Accurate and Stable Parent Selection in Dense 6TiSCH Networks. In *Proc. of the* 15^{th} *EWSN Conf.*, 2018.

[12] B. Islam et al. Rethinking Ranging of Unmodified BLE Peripherals in Smart City Infrastructure. In *Proc. of the* 9^{th} *MMSys Conf.*, 2018.

[13] H. Karvonen et al. Interference of Wireless Technologies on BLE Based WBANs in Hospitals. In *Proc. of the* 28^{th} *PIMRC Symp.*, 2017.

[14] J. Ko and M. Chang. Momoro: Providing mobility support for low-power wireless applications. *IEEE Systems Journal*, 2014.

[15] V. Kotsiou et al. Is local blacklisting relevant in slow channel hopping low-power wireless networks? In *Proc. of the ICC Conf.* IEEE, 2017.

[16] V. Kotsiou et al. LABeL: Link-based adaptive blacklisting technique for 6TiSCH wireless industrial networks. In *Proc. of MSWiM*, 2017.

[17] T. Lee, M.-S. Lee, H.-S.Kim, and S. S. Bahk. A synergistic architecture for RPL over BLE. In *Proc. of the* 13^{th} *SECON Conf.*, 2016.

[18] T. Liu and A. E. Cerpa. Foresee (4C): Wireless link prediction using link features. In *Proc. of the* 10^{th} *IPSN Conf.*, 2011.

[19] D. Mantz et al. InternalBlue - Bluetooth Binary Patching and Experimentation Framework. In *Proc. of the* 17^{th} *MobiSys Conf.*, 2019.

[20] A. Marinčić et al. Interoperability of iot wireless technologies in ambient assisted living environments. In *Proc. of the WTS Symp.*, 2016.

[21] Mini-Circuits. RCDAT-8000-90 - Programmable Attenuator, 2017.

[22] K. Morsi et al. Performance estimation and evaluation of Bluetooth frequency hopping selection kernel. In *Proc. of the JCPC Conf.*, 2009.

[23] R. Natarajan et al. Analysis of Coexistence between IEEE 802.15. 4, BLE and IEEE 802.11 in the 2.4 GHz ISM Band. In *Proc. of the* 42^{th} *IECON Conf.* IEEE, 2016.

[24] Nordic Semiconductors. nRF52840 Specifications, 2018.

[25] G. Pau et al. Bluetooth 5 Energy Management through a Fuzzy-PSO Solution for Mobile Devices of Internet of Things. *Energies*, 2017.

[26] Raspberry Pi Foundation. Raspberry Pi 3 Model B, 2018.

[27] M. Schuß et al. Moving Beyond Competitions: Extending D-Cube to Seamlessly Benchmark Low-Power Wireless Systems. In *Proc. of the* 1^{st} *CPSBench Worksh.*, 2018.

[28] M. Schuß et al. JamLab-NG: Benchmarking Low-Power Wireless Protocols under Controllable and Repeatable Wi-Fi Interference. In *Proc. of the* 16^{th} *EWSN Conf.*, 2019.

[29] M. Sha et al. ARCH: Practical channel hopping for reliable home-area sensor networks. In *Proc. of the* 17^{th} *RTAS Symp.*, 2011.

[30] S. Silva et al. Coexistence and interference tests on a Bluetooth Low Energy front-end. In *Proc. of the SAI Conf.*, 2014.

[31] M. Spörk et al. BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices. In *Proc. of the* 15^{th} *ACM SenSys Conf.*, 2017.

[32] M. Spörk et al. Improving the Timeliness of Bluetooth Low Energy in Noisy RF Environments. In *Proc. of the* 16^{th} *EWSN Conf.*, 2019.

[33] M. Spörk et al. Performance and Trade-offs of the new PHY modes of BLE 5. In *Proc. of the 2019 PERSIST-IoT Workshop.* ACM, 2019.

[34] K. Srinivasan et al. An empirical study of low-power wireless. *ACM Transactions on Sensor Networks*, 6(2), 2010.

[35] The Zephyr Project. Zephyr OS: An RTOS for IoT, 2018.

[36] T. Watteyne et al. Reliability through frequency diversity: why channel hopping makes sense. In *Proc. of the* 6^{th} *PE-WASUN Symp.*, 2009.

[37] T. Winkel. Robustness of Bluetooth Low Energy in In-Vehicle Networks-An Experimental Study. Master's thesis, 2016.