Received October 19, 2021, accepted November 27, 2021, date of publication January 6, 2022, date of current version January 27, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3140803

Identifying the BLE Advertising Channel for Reliable Distance Estimation on Smartphones

CHRISTIAN GENTNER[1], DANIEL GÜNTHER[2], AND PHILIPP H. KINDT[3]
[1]Institute of Communications and Navigation, German Aerospace Center (DLR), 82234 Wessling, Germany
[2]Technical University of Munich (TUM), 80333 München, Germany
[3]Faculty of Computer Science, Chemnitz University of Technology, 09111 Chemnitz, Germany

Corresponding author: Christian Gentner (christian.gentner@dlr.de)

Christian Gentner, Daniel Günther, and Philipp H. Kindt contributed equally to this work.

ABSTRACT Estimating the distance between two smartphones plays an important role in a host of applications. For this purpose, smartphones emit and scan for Bluetooth Low Energy (BLE) signals. When a device is detected, the distance is estimated by evaluating the received strength of these signals. The main insight that is exploited for distance estimation is that the attenuation of a signal increases with the distance along which it has traveled. However, besides distance, there are multiple additional factors that impact the attenuation and hence disturb the distance estimation procedure. Among them, frequency-selective hardware and signal propagation belong to the most significant ones. For example, a BLE device transmits packets on three different frequencies (channels), while the transmit power and the receiver sensitivity depend on the frequency. As a result, the received signal strength varies for each channel, even when the distance remains constant. However, the information on which wireless channel a packet has been received is not made available to a smartphone. Hence, this error cannot be compensated, e.g. by calibration. In this paper, we for the first time provide a solution to detect the wireless channel on which a packet has been received by a smartphone application. We experimentally evaluate our proposed technique on multiple different smartphone models. Our results help to make distance estimation on smartphones more robust and accurate.

INDEX TERMS Bluetooth, bluetooth low energy, BLE, channel detection, contact tracing, neighbor discovery, distance estimation, smartphone, android.

I. INTRODUCTION

Estimating the distance between two wireless devices has been studied actively by the community, and techniques such as ultra-wideband (UWB), WiFi fine time measurements (FTM) and time-of-flight (TOF) have increased the estimation accuracy considerably throughout the recent years [1]–[5]. However, in many scenarios, at least one of these two devices is a smartphone, on which such approaches are mostly unavailable. Nevertheless, distance estimation using smartphones plays a key role in a host of applications. For example, location-based services, where e.g. a message is pushed on each smartphone in proximity, are growing in importance. Here, the main goal is estimating the distance between a BLE device and a smartphone as accurately as possible. Similarly, accurate distance estimation between a phone and a gadget is essential when using a phone to search for lost items, e.g. by using a keyfob that is applying the BLE find me profile [6]. Another important application that relies on distance estimation are smartphone-based contact tracing apps against the novel coronavirus [7]. Here, every device continually transmits BLE packets and listens for incoming transmissions. As soon as a packet from another device is received, the distance between both of them is estimated [8]. Similarly, in indoor localization applications, multiple BLE location beacons are installed in a building, and smartphones attempt to determine their positions by estimating the distance from each of them, see e.g. [5]. Distance estimation using smartphones has been studied thoroughly throughout the last years, e.g. in [5], [9]–[12]. Currently, due to the high relevance for contact tracing, analyzing and increasing the estimation accuracy is receiving considerable attention by the scientific community [13], [14].

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

A. DISTANCE ESTIMATION

As already mentioned, there are established standards for distance estimation, e.g. UWB combined with TOF, WiFi-FTM, or TOF using BLE. As we describe in Section III, these techniques are not widely available on smartphones. Hence, estimating the distance between a pair of phones relies on sensing the attenuation using the following procedure.

One device transmits a packet with a certain transmit power P_t. This value is piggy-backed onto the packet. The wireless signal then undergoes a certain *path loss*, which depends on the distance along which the signal travels. The opposite device will receive the packet with a certain power P_r. In free space, a wireless signal traveling along a distance d between a sender and a receiver will be received with a power of

$$P_r = P_t \cdot G_t G_r \cdot \left(\frac{\lambda}{4\pi d} \right)^2 , \tag{1}$$

see [15]. In Equation (1), $\lambda = c_0/f$ is the wavelength of the signal, where c_0 is the speed of light and f the frequency. G_t and G_r are the gains of transmitter and receiver, respectively. Since P_r and P_t are known by the receiver, the distance d can be estimated by solving Equation (1) by d.

It is obvious that this principle only works for values of P_r above a certain threshold, below which a receiver cannot successfully decode the incoming packet. Hence, the maximum range within which distance estimation is possible is limited by this threshold, G_t, G_r, the transmit power P_t of the emitted signals and the wireless propagation environment.

While Equation (1) holds true for ideal devices in free space, in practical setups, distance estimation is aggravated by multiple effects, of which the following ones are most important.

- The antennas of both devices are usually directional and hence, the orientation between both devices impacts P_t and P_r. This implies that the estimated distance would change if at least one of both devices was rotated by a certain angle, even when the positions of both devices remained the same.
- Human tissue attenuates the signal by a considerably higher degree than free space. For example, the attenuation between the chest and the back of the human body has been reported as 19.2 dB [16]. As a result, the estimated distance is strongly disturbed when human tissue is within the direct signal path, and hence also by how two human bodies are oriented relatively to each other, and where the phones are worn on the body.
- Multipath propagation, e.g. caused by reflections on surfaces, can lead to interference between the signal traveled along the line-of-sight and reflected ones.
- Multiple transmission properties depend on the frequency on which a packet is transmitted. We describe these properties in the next section.

This paper is on mitigating the errors induced by frequency-dependent effects. Existing approaches for distance estimation on smartphones are unaware of the frequency on which a packet has been received. As we will show in Section II, when accounting for this, the accuracy of such distance estimation procedures can be improved significantly.

On smartphones, the received power of a BLE signal is available in the form of a received signal strength indicator (RSSI), which is provided by the Bluetooth radio. It is an integer value representing the received power in dBm. After converting it into a linear unit (i.e., mW), it can be inserted into Equation (1) for estimating the distance. We next describe how this procedure is impacted by frequency-dependent distortions.

B. CHANNEL-DEPENDENT RSSI

In the BLE protocol, which is used for distance estimation on smartphones, advertising packets are sent on the 3 different channels 37, 38 and 39, which are spread over almost the entire frequency band used. They use center frequencies of 2.402 GHz, 2.426 GHz and 2.480 GHz [17]. Because of the following effects, the RSSI depends on the channel on which the packet is received.

1) Almost every device has frequency-dependent values of G_r and G_t. In other words, a packet sent on a certain channel (and hence frequency) will have a larger power than when being sent on a different channel, and the receiver will similarly sense different RSSI values for the same actual signal power on different channels.
2) The *path loss* of a signal, i.e., its attenuation over a certain distance, depends on the channel/frequency on which the signal is transmitted, see Equation (1).
3) Packets sent on different channels propagate differently in the environment. BLE signals are reflected, scattered and diffracted by objects in the surrounding. Hence, the signal reaching the receiving antenna consists of multiple replicas of the transmitted signal, which are called *multipath components*. These replicas interfere with those transmitted along the direct path, i.e., the line-of-sight. When interfering constructively, the RSSI increases, whereas it is reduced in the case of destructive interference. Whether constructive or destructive interference occurs depends on the distance between the sender and receiver, and the frequency of the signal. Therefore, the estimated distance might be drastically changed due to the received signal being distorted by multipath propagation.

For achieving the maximum possible estimation accuracy, these effects have to be mitigated when estimating the distance. If the channel on which a packet has been received was known, frequency-specific values of G_t and G_r could be used instead of relying on a mean value comprising all possible frequencies. This would cancel the error induced by frequency-dependent values of G_t and G_r. Also the error induced by the frequency-dependent propagation, as given by Equation (1), could be cancelled by inserting the wavelength λ that corresponds to the reception channel. Though errors induced by multipath propagation cannot be fully compensated (since multipath propagation is also present when only one carrier frequency is used), the information on which channel a packet has been received can nevertheless be exploited to reduce this error. In particular, multipath components cause fading. Hence, packets received on the different channels usually differ in their RSSI values. Therefore, when packets received on different channels are available, the distance estimation error can be reduced by averaging their RSSI values before carrying out the estimation, or by applying more elaborate methods, as proposed e.g. in [18].

As we will show in Section II, utilizing the information on the reception channel can indeed significantly increase the estimation accuracy. This conclusion has also been drawn from prior studies, e.g. [19].

C. UNAVAILABILITY OF CHANNEL INFORMATION

Existing approaches for distance estimation on smartphones do not exploit any information on the channel on which a packet has been received. This is because the BLE radio does not relay the information on which channel a packet was received to the smartphone's operating system. Indeed, the BLE *host control interface*, which is used for data exchange between radio and smartphone, specifies that incoming advertising packets are reported to the smartphone without containing any channel information [17]. As a result, the smartphone is not aware of the channel a packet has been received on. We in the following briefly discuss multiple potential workarounds, and why they do not help.

At a first glance, simply averaging over a large number of received packets could potentially cancel frequency-dependent errors. However, computing the average RSSI over multiple packets can only reduce frequency-dependent errors notably, if their reception channels are known. Otherwise, it cannot be guaranteed that the same number of packets for each channel are forming the average, which will bias the result. Furthermore, as we describe in Section IV, subsequently received packets typically all belong to the same channel, and packets belonging to another channel only become available after several seconds. Hence, for obtaining at least a similar number of packets for each channel with a high probability, capturing needs to continue over large amounts of time. Within such a time-frame, the distance between two mobile devices will already have changed in most cases. Hence, this does not cancel frequency-dependent errors in practical scenarios.

Another potential workaround could be piggy-backing the transmission channel on the payload of each packet. However, this requires low-level control over the transmitting device and thus, most off-the-shelf devices and BLE stacks cannot be used. In particular, estimating the distance between a pair of smartphones is infeasible, because a smartphone is unaware of the channel on which a particular packet is transmitted, and therefore cannot piggy-back this information.

We conclude that though the information on the reception channel is very valuable for distance estimation, it is unavailable on smartphones. This paper addresses this issue by proposing a technique to retrieve the channel information on a smartphone, which is described next.

D. PROPOSED SOLUTION

In this paper, we, for the first time, propose a technique to detect on which channel an advertising packet was received on an Android smartphone. We thereby exploit undocumented behavior of many wireless System-on-a-Chips (SoCs) found in recent smartphones. In particular, after scanning for incoming packets has been activated by an application (app), we observe that on most smartphones we have tested, scanning starts on channel 37. We can exploit this behavior for obtaining the first channel on which the smartphone scans. Furthermore, the time between two consecutive reception windows is large and deterministic. Hence, we can then classify the channel of later received packets based on associating their reception time with the time windows during which the receiver listens for incoming packets. For example, if we predict all points in time at which the receiver scans on channel 37, we know that all packets received at these points in time have also been transmitted on channel 37.

E. CONTRIBUTIONS

Compared to existing works, we make the following contributions in this paper:

1) We show how exploiting the information on which channel a BLE advertising packet has been received leads to more accurate distance estimations and quantify the achieved accuracy improvements using real-world data.
2) We, for the first time, propose a technique to identify the channel on which a BLE advertising packet has been received on a smartphone.
3) We evaluate the detection probability of our proposed method in detail. Our results suggest that the channel of reception can be detected reliably in 100 % of all attempts.
4) We test our proposed methodology on different smartphone models from different manufactures and show that it is compatible with the vast majority of the phones we have tested.

F. PAPER ORGANIZATION

The rest of this paper is organized as follows. In the next section, we show how the accuracy of distance estimation can be improved by utilizing the information on which channel a packet has been received. We then provide an overview on related work in Section III. In Section IV, we describe how the procedure used for distance estimation in BLE, called *advertising and scanning*, works. Next, in Section V, we describe our proposed technique for channel detection. We experimentally evaluate this technique in Section VI and conclude our findings in Section VII.

II. HOW KNOWING THE RECEPTION CHANNEL INCREASES THE ACCURACY

In the previous section, we have claimed that the accuracy of distance estimation can be improved when the channel on which a packet has been received is known. In this section, we justify this claim using a practical experiment. In particular, we measured the RSSI along with the exact

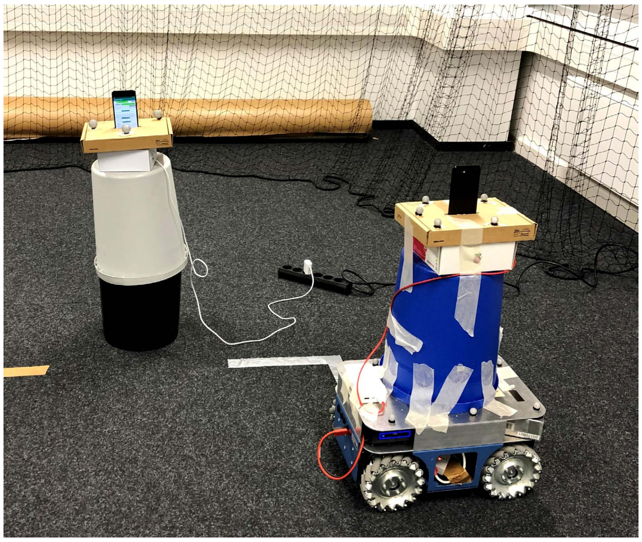


FIGURE 1. Experimental setup: a *Google Pixel 2* smartphone was used as a stationary transmitter; a *Google Pixel 3* was used as a receiver and mounted on a mobile robot. The *Google Pixel 2* was kept on a static position, while the *Google Pixel 3* moved along a large number of different positions.

distance between two smartphones for a large number of different positions in an indoor environment. We then partitioned the resulting data into two sets. One of them was used to calibrate a function that maps RSSI values to distances, i.e., for generating a distance model. The remaining set was used to verify the accuracy of estimating the distance using this function. We have carried out this evaluation procedure with and without utilizing the information on the reception channel, and confront the results in this section. We thereby show that the availability of the channel information leads to a more accurate distance estimation. We next describe our experimental setup.

A. EXPERIMENTAL SETUP

Figure 1 depicts the setup we have used for recording RSSI and distance data. We used a *Google Pixel 2* smartphone as a BLE transmitter, which remained at a fixed position. The BLE radio of the smartphone was configured using the ADVERTISE_MODE_LOW_LATENCY mode (see Section IV for details), which implies that advertising packets were sent with a period of $T_a = 100\,\text{ms}$. We used the highest supported transmit power to cover the maximum range. A *Google Pixel 3* smartphone was used as a receiver, which was mounted on a mobile robot. Both smartphones were located in an indoor environment, their height above the ground was 67 cm.

The robot that carried the receiving smartphone was moved to 340 different positions. At each such position, multiple hundred RSSI values were sampled, while the robot did not move during each measurement. Hence, each sequence of measured values corresponds to a unique distance between the sender and receiver, as well as to a unique orientation of the receiver antenna.

In addition to RSSI data, we also recorded accurate position information. This was done using a camera-based setup, consisting of 16 infrared cameras and infrared strobes. This setup, based on a system from *Vicon*,[1] can locate objects in an area of approximately $10\,\text{m} \times 4\,\text{m}$ with a distance error of less than 1 cm. The receiving and transmitting smartphones were tagged with a reflector for being recognized by the cameras. The resulting data was used as follows.

B. DATA PROCESSING

1) OVERVIEW

The data obtained from this experiment was then used to evaluate the difference in the distance estimation accuracy with and without the availability of the reception channel. To this end, we first partitioned the data consisting of distance and RSSI values into the following two sets.

- The data belonging to 140 different, randomly chosen positions were assigned to the set S_{model}. It was used to generate a model that assigns a distance estimation to every possible RSSI value.
- The data belonging to the remaining 200 positions, denoted by S_{eval}, was used for evaluating the accuracy of this model.

We next describe how the model was generated, and then give more details on evaluating this data with and without channel information.

2) DISTANCE MODEL

Equation (1) characterizes the path loss of a wireless signal traveling through free space as being inversely proportional to the squared distance between the transmitter and the receiver, i.e., d^2. Practical scenarios always comprise the presence of at least the earth surface, and the signal propagation therefore deviates from free space. To model such a propagation, we use the following generalization of Equation (1).

$$P_r = P_t \cdot G_t G_r K \cdot \frac{1}{d^\gamma} \tag{2}$$

Here, γ is the *path loss exponent*. It is used to account for different environments. K is a constant fitting value. For the case of free space, it is $\gamma = 2$ and $K = (\lambda/4\pi)^2$, and Equation (2) becomes equal to Equation (1). Furthermore, the received and transmitted power P_t and P_r are usually expressed in dBm. The relation between a power $\hat{P}$ in dBm and P in Watt is given by $\hat{P} = 10 \cdot \log_{10} \frac{P}{1\,\text{mW}}$. We can easily solve Equation (2) by d and insert $P_t = 1\,\text{mW} \cdot 10^{\hat{P}_t \cdot \frac{1}{10}}$ and $P_r = 1\,\text{mW} \cdot 10^{\hat{P}_r \cdot \frac{1}{10}}$. This leads to the following equation, which is usually referred to as the log-distance path loss model, see e.g. [20], [21].

$$\begin{aligned} \log_{10}\left(d(\hat{P}_r)\right) &= \frac{1}{10 \cdot \gamma}\hat{P}_t - \frac{1}{10 \cdot \gamma}\hat{P}_r + \frac{1}{\gamma}\log_{10}(G_t G_r K) \\ &= a + b \cdot \hat{P}_r \end{aligned} \tag{3}$$

[1]https://www.vicon.com

Here, $\hat{P}_r$ is identical to the RSSI reported by the smartphone. The parameters a and b account for the values of G_t, G_r, γ, and P_t. Instead of obtaining them by computation, it is more practical to determine them through regression.

To this end, we have computed the values of a and b that minimize the mean squared error between Equation (3) and the ground-truth distance when considering all data contained in S_{model}. The details of this regression are described in the Appendix.

With known values for a and b, Equation (3) translates any RSSI value into a distance estimate. However, evaluating it can be done in two different ways, based on whether the channel on which a packet has been received is known or not.

In particular, when it is known on which channel a packet has received, we can generate 3 different models. For the packets received on channel 37, we obtain a tuple of fitting parameters (a_{37}, b_{37}). Inserting them into Equation (3) leads to the distance model $d_{37}(\hat{P}_r)$. Similarly, we obtain separate tuples (a_{38}, b_{38}) and (a_{39}, b_{39}) for packets on channels 38 and 39, leading to the distance models $d_{38}(\hat{P}_r)$ and $d_{39}(\hat{P}_r)$. Additionally, if the channel of reception is unknown, every single RSSI measurement contained in S_{model}, irrespective of its channel, is used to compute a tuple (a_{agg}, b_{agg}). The resulting parameter values (a, b), in dependence of the channel on which the considered packets were received, are given in Table 1.

TABLE 1. Model parameters for the considered scenario.

	a	b
$d_{agg}(\hat{P}_r)$	−2.3356	−0.0378
$d_{37}(\hat{P}_r)$	−1.9855	−0.0323
$d_{38}(\hat{P}_r)$	−1.9392	−0.0321
$d_{39}(\hat{P}_r)$	−2.5024	−0.0395

In Figure 2, every circle represents the mean RSSI value for all packets received on channel 37 that belong to the same distance. We have considered all measurements in S_{model} and S_{eval}. Similarly, every diamond represents channel 38 and every square channel 39. Every "x" represents the mean RSSI for all measurements belonging to the same distance irrespective of the channel. Furthermore, the distance models $d_{37}(\hat{P}_r)$, $d_{38}(\hat{P}_r)$, $d_{39}(\hat{P}_r)$, $d_{agg}(\hat{P}_r)$, which have been computed based on S_{model} as described above, are depicted. As can be seen, these models differ significantly from each other, which further underpins the hypothesis that exploiting the channel information can increase the accuracy of the distance estimation procedure. We next evaluate the accuracy achieved when either $d_{agg}(\hat{P}_r)$, or the triple $(d_{37}(\hat{P}_r), d_{38}(\hat{P}_r), d_{39}(\hat{P}_r))$ is utilized for distance estimation.

3) COMPARISON OF ACCURACY

To quantify the accuracy of both possibilities, we have computed the error E between the estimated distance $d(\hat{P}_r)$ and the actual distance $\hat{d}$ for every RSSI/distance pair contained in S_{eval} as follows.

$$E = \| d(\hat{P}_r) - \hat{d} \| \tag{4}$$

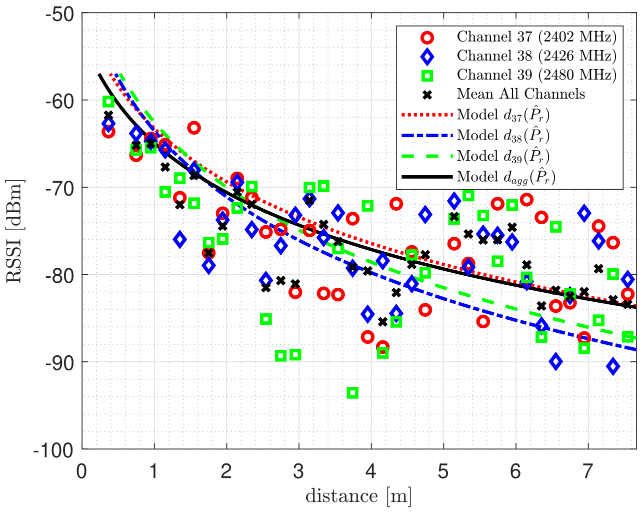


FIGURE 2. Measured mean RSSI values and corresponding distance models $d_{37}(\hat{P}_r)$, $d_{38}(\hat{P}_r)$, $d_{39}(\hat{P}_r)$, $d_{agg}(\hat{P}_r)$.

E_{agg} accounts for the error when the reception channel is unknown. It has been obtained by estimating the distance of each sample in S_{eval} using $d_{agg}(\hat{P}_r)$. E_{ch} represents the estimation error when the reception channel is known. It has been obtained by evaluating the distance of each measurement in S_{eval} with the model for the corresponding channel, i.e., either $d_{37}(\hat{P}_r)$, $d_{38}(\hat{P}_r)$ or $d_{39}(\hat{P}_r)$.

The results of this are shown in Figure 3. As can be seen, the CDF for E_{ch} approaches unity for lower errors than E_{agg}. The maximum error of the channel aware distance estimation (i.e., the maximum value of E_{ch}) is 17.8 m. The corresponding root mean square error (RMSE) is 2 m. For the distance estimation using d_{agg}, we obtain a maximum error of 21.9 m and a RMSE of 2.5 m. These results suggest that channel-aware distance estimation is significantly more accurate than estimations with an unknown reception channel.

III. RELATED WORK

In this Section, we briefly summarize work related to distance estimation using smartphones.

A. DISTANCE ESTIMATION IN STANDARD PROTOCOLS

Multiple wireless standard protocols, such as IEEE 802.11 (WiFi) or BLE, contain dedicated distance estimation procedures. In the following, we briefly describe why they do not solve the problem of estimating the distance between two smartphones. The recent WiFi IEEE 802.11-2016 standard specifies a signal propagation time-based protocol for ranging, called FTM. It is more commonly known as the WiFi Round Trip Time (RTT) protocol. However, only very few smartphones already support it as of today [22], and it remains unclear whether it will become widely supported in the future. Even if all manufacturers decided to add support for WiFi RTT in the future, it would take many years before the probability of a random pair of phones meeting each

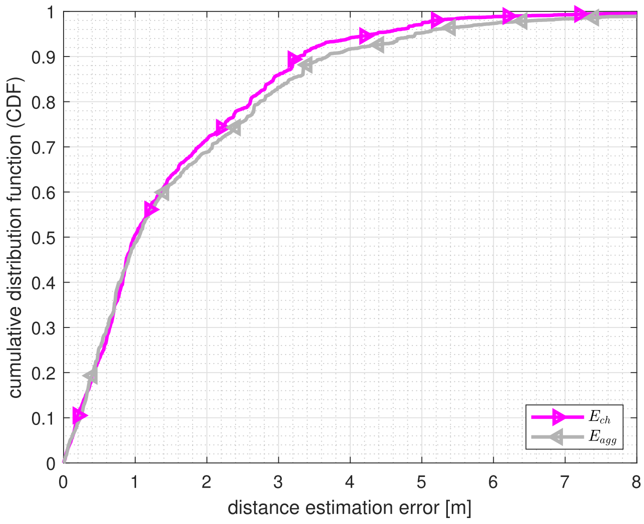


FIGURE 3. CDFs of distance estimation errors. E_{agg} corresponds to the distance estimation error with unknown channel, whereas E_{ch} correspond to the errors when the channel is known.

other are both RTT-enabled becomes reasonably high. Similarly, though TOF-based distance estimation has recently been introduced into the latest Bluetooth standard, to the best of our knowledge, this isn't supported by any of the available phones, yet. Even when future smartphones might add support for it, recent studies [19] indicate that it is not advantageous when the distance needs to be calculated based on only a few received packets. Hence, particularly when one of the two devices is potentially in motion and thus only a few packets can be received before the distance changes, the currently used RSSI-based procedure will remain important. Similar to this, distance estimation techniques known from other domains, such as TOF and UWB, are not available on smartphones. Therefore, other solutions have been developed for them, which are described next.

B. DISTANCE ESTIMATION USING SMARTPHONES

Estimating the distance between a pair of smartphones has been studied using different techniques, such as correlating the measurements of the ambient magnetic field by different smartphones [23], acoustic ranging [24], or the RSSI of the Bluetooth or IEEE 802.11 (WiFi) radio. Among these approaches, RSSI-based methods have turned out to be the most practical ones. Since BLE is only available in relatively recent smartphones, early studies have focused on distance estimation using WiFi or the "legacy" Bluetooth, called *Bluetooth BR/EDR*. For example, *Comm2Sense* [25] configures smartphones as mobile WiFi-hotspots to estimate the proximity of two devices based on the WiFi RSSI. Since setting up mobile WiFi-hotspots on smartphones is inconvenient for the user and the energy consumption of WiFi drains the battery quickly, a large number of other approaches, e.g. [26], [27], are built on the RSSI of Bluetooth BR/EDR. Whereas the RSSI in Bluetooth BR/EDR is measured relatively to a "golden receive power range" [17], the BLE protocol specifies that the RSSI is an absolute received power. In addition, BLE is designed for scanning for incoming packets continuously in the background. As a result, more recent approaches on distance estimation build upon the BLE protocol. For example, [9] uses 3 different approximation models, which are selected based on a coarse classification of the estimated distance. A large number of approaches, e.g. [28]–[30], have studied the analysis of the sensed RSSI data. Furthermore, the work in [14], [31] has experimentally evaluated the accuracy of distance estimation using smartphones in the context of contact tracing. However, the accuracy of the RSSI data itself has always been considered as immutable. Hence, to the best of our knowledge, detecting the channel on which a packet was received for improving the accuracy of distance estimation has not been considered previously to this work.

C. LOCALIZATION USING FINGERPRINTING

Instead of directly estimating the distance between two devices, fingerprinting approaches [11], [32], [33] can recognize different previously recorded device positions. In particular, a device can identify its position, if the RSSI matches a corresponding, previously recorded *fingerprint* being unique to this position. Such approaches typically require *multiple* transmitting devices at static positions. Different works on fingerprinting, e.g. [5], [11], [34], have also studied the frequency dependent RSSI and conclude that the information on which channel a packet has been received improves the localization accuracy. In particular, [5] also shows that distance estimation becomes more accurate when the reception channel is known. While fingerprinting approaches are state-of-the art in scenarios with static anchor devices that provide the necessary signals to be fingerprinted, they are not feasible in mobile networks, in which most devices do not remain at static positions. Hence, they cannot be used for estimating the distance between two smartphones.

IV. ADVERTISING AND SCANNING IN BLE

Distance estimation using smartphones builds upon the following procedure for detecting devices in its surrounding. It is provided by the BLE protocol [17] and referred to as *advertising and scanning*. Every device periodically schedules an *advertising event* once per T_a time-units. T_a is called the *advertising interval* and is composed of a static part $T_{a,0}$ plus a random delay $\rho \in [0, 10\,\text{ms}]$. In each such event, three packets in a row are sent. The first of them is sent on channel 37 (which corresponds to a center frequency of 2.402 GHz), the second one on channel 38 (2.426 GHz) and the third one on channel 39 (2.480 GHz) [17]. This is depicted in Figure 4. Here, every arrow stands for an advertising packet. Three such packets in a row on different channels form an advertising event. The first event falls with a random offset of Φ into an instance of the sequence of scan windows, in which we define the end of the scan window on channel 37 as the origin.

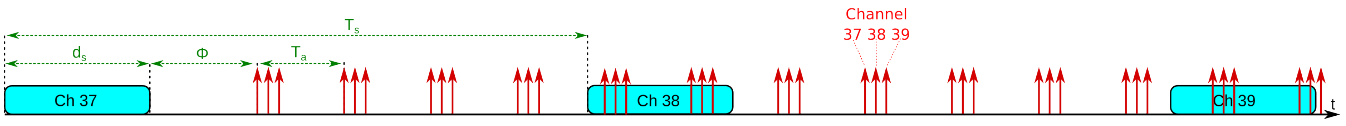


FIGURE 4. Advertising and scanning in BLE. Arrows depict advertising packets, which are grouped in advertising events of 3 consecutive packets on 3 different channels. The rectangles depict scan windows.

For being able to receive incoming packets, every device also listens to the channel by using so-called *scan windows*. Every scan window has a duration of d_s time-units, and there is one such window every *scan interval* T_s. After every instance of the scan window, the channel for the succeeding window is toggled between channel 37, 38 and 39 in a round-robin fashion. This is also depicted in Figure 4.

A device can detect the presence of another device, once a packet from the remote device coincides with one of its scan windows [35]. Most values for (T_a, T_s, d_s) supported by Android fulfill $T_a < d_s$ (cf. Figure 4) and hence, the reception of at least one packet is guaranteed in each scan window.

The Android operating system does not allow an app to select these parameter values directly. Instead, an app can chose between three different settings that determine T_a and three different ones that determine T_s and d_s. These settings are listed in Table 2.

TABLE 2. BLE parameterizations in Android.

Android Setting	$\mathbf{T_a}$ [s]	$\mathbf{T_s}$ [s]	$\mathbf{d_s}$ [s]
SCAN_MODE_LOW_POWER	-	5.120	0.512
SCAN_MODE_BALANCED	-	4.096	1.024
SCAN_MODE_LOW_LATENCY	-	4.096	4.096
ADVERTISE_MODE_LOW_POWER	1.000	-	-
ADVERTISE_MODE_BALANCED	0.250	-	-
ADVERTISE_MODE_LOW_LATENCY	0.100	-	-

It needs to be mentioned that there is no transparent mapping between these settings and the corresponding values. First, the values that correspond to a certain setting (e.g. SCAN_MODE_BALANCED) are not officially specified by Google. We have therefore obtained them from the source-code of the latest version of Android.[2] Second, the values that are actually used could differ from those given in Table 2 due to scheduling conflicts. In particular, the radio might maintain other Bluetooth connections, and the points in time at which other packets are exchanged might overlap with those needed for advertising and scanning. In addition, the Bluetooth radio is also used for WiFi on many devices, which could lead to additional scheduling conflicts. However, we found in our experiments that the values from Table 2 are actually used during normal operation, i.e., when no scheduling conflicts are present.

For the sake of completeness, we here also mention three other configuration options. First, the SCAN_MODE_OPPORTUNISTIC setting can be used by an app to obtain scan results when scanning has been triggered by a different app, without triggering the scanning itself. Second, a different set of values for T_s and d_s is available when using *batch scanning*, where multiple received packets are reported to the app jointly after some time instead of immediately after discovery. However, we could not find any documentation of this feature and hence did not study it in detail in this paper. Third, Google and Apple have drafted an *Exposure Notification* service [36] for contact tracing. Here, an advertising interval of 200 ms to 270 ms is specified, while no values for T_s and d_s are given. However, only approved tracing apps can make use of this interface. None of these additional options will supersede the need for channel detection, as we propose in this paper.

[2]In older Android versions, the parameter values of the scan modes are different.

V. CHANNEL DETECTION

In this section, we describe how the channel on which an incoming packet is received can be detected on a smartphone. As already mentioned, the BLE radio does not relay this information to the smartphone's operating system, since the Bluetooth standard does not specify an interface for this.

According to the BLE specification [17], the channel on which the radio scans is toggled after *every* scan window. Thereby, the same order of channels 37, 38, 39, 37, …is always pursued. Though the BLE specification does not specify on which channel the radio has to scan first after its activation, we could observe on different smartphone models (see Section VI and Table 4) that when scanning is activated, the device will always begin with channel 37. In other words, after every reset of the BLE radio, also the channel to be scanned on first will be reset to channel 37.

Let the point in time at which scanning was activated be given by t. Then, incoming packets will only be received on channel $c \in \{37, 38, 39\}$, if their reception time falls within a time-interval $I_c(k) = [t_{l,c}(k), t_{r,c}(k)]$, $k = 1, 2, 3, 4, \ldots$, with

$$\begin{aligned} t_{l,c}(k) &= t + 3 \cdot (k-1) \cdot T_s + (c-37) \cdot T_s \\ t_{r,c}(k) &= t + 3 \cdot (k-1) \cdot T_s + (c-37) \cdot T_s + d_s \,. \end{aligned} \quad (5)$$

Equation (5) directly follows from Figure 4. Therefore, we can detect the channel on which a packet was received by classifying the time of each packet reception into I_{37}, I_{38} or I_{39}.

The values for d_s and T_s can be obtained from Table 2. However, recall that especially in case of scheduling conflicts, the phone might potentially deviate from this periodic scheme. Though we could observe that the values from Table 2 appear to be used in *most* cases (i.e., when no scheduling conflicts are present), there might potentially be (slight) changes of these parameter values, or even dropped scan windows or packets. In our experiments, the scan windows always occurred at the expected points in time given by Equation (5), even when WiFi was activated. However, some of the transmitted packets were not received, indicating that the scan windows were interrupted on a short-term basis for carrying out WiFi communication. This does not negatively impact channel detection, since the classification of reception times according to Equation (5) remains unaffected. The same holds true when packet transmissions are omitted or their transmission times change due to scheduling conflicts.

Because the clocks of the smartphone and the Bluetooth radio are not synchronized, they could drift against each other. This might disturb the channel detection based on Equation (5), since the classification is carried out within an app that relies on the clock of the smartphone, whereas the scan windows are scheduled using the clock of the radio. To mitigate the effects of this, we slightly modify the interval borders of $I_c(k)$ from Equation (5) to $\hat{I}_c(k) = [\hat{t}_{l,c}(k), \hat{t}_{r,c}(k)]$, $k = 1, 2, 3, 4, \ldots$, with

$$\hat{t}_{l,c}(k) = t + 3 \cdot (k-1) \cdot T_s + (c-37) \cdot T_s + t_g/2$$
$$\hat{t}_{r,c}(k) = t + 3 \cdot (k-1) \cdot T_s + (c-36) \cdot T_s - t_g/2 \,. \quad (6)$$

Hence, for realizing a more robust detection, we here classify each received packet by into which instance of T_s it falls, even when being received outside of the (estimated) scan window. Further, t_g is a guard time to compensate for the clock drift, for which we propose a concrete value in Section VI. In addition, for limiting this drift, we propose to regularly re-start the scanning procedure after a certain period of time. We evaluate after which amount of time such a re-start should occur in Section VI.

Algorithm 1 shows the pseudo-code of our proposed algorithm for detecting the channel on which a packet was received. In order to limit the power consumption of the smartphone, the algorithm starts the BLE scanner using the SCAN_MODE_LOW_POWER setting (c.f. Line 4 in Algorithm 1). As soon as BLE signals are detected, the main part of the algorithm is executed and the BLE scanner is re-started using the SCAN_MODE_LOW_LATENCY setting (c.f. Line 9-12 in Algorithm 1). We remember the timestamp t when scanning was re-started. Note that this timestamp does not perfectly coincide with the actual re-start of the BLE scanning procedure, since there might be delays and/or jitter. The effect of such misalignments are mitigated through the guard time t_g in Equation (6). All subsequent packets are handled in the inner while loop starting in Line 13 of Algorithm 1. For each such packet, the reception time is computed and t (i.e., the time when scanning was started) is subtracted. The resulting time difference is used by the *ClassifyChannel()*-function, which infers the channel of reception by evaluating Equation (6). After *Max-Scan-Time* (cf. Line 16 in Algorithm 1) has passed, the scanning procedure is re-started to limit the clock drift between smartphone and radio, as already explained. The guard time t_g compensates for any clock drift before this re-start.

In the next section, we evaluate the accuracy of channel detection using this algorithm on different smartphone models and for different re-start intervals of the BLE scanning procedure.

Algorithm 1: Android BLE Channel-Detector

```
1  doInit = true;
2  while 1 do
3      if doInit then
4          Scan-Mode = SCAN_MODE_LOW_POWER;
5          ReStartScan();
6          doInit = false;
7          Channel-Detection = false;
8      if BLE signals detected then
9          Channel-Detection = true;
10         Scan-Mode =
             SCAN_MODE_LOW_LATENCY;
11         ReStartScan();
12         t = GetSystemTime();         // see Eq. (6)
13     while Channel-Detection do
14         if BLE signal received then
               /* classify BLE signal into
                  Î37, Î38 or Î39 using Eq. (6)
                  */
15             ClassifyChannel(t, GetSystemTime());
16         if GetSystemTime() - t > Max-Scan-Time then
17             ReStartScan();
18             t = GetSystemTime() ;   // see Eq. (6)
19         if No signals detected then
20             doInit = true;
21             break;
```

VI. EVALUATION

In this section, we evaluate the proper functioning of our proposed technique for channel identification and the accuracy of Algorithm 1. Towards this, we first describe our experimental setup and then give experimental data.

A. EXPERIMENTAL SETUP

In order to evaluate our approach and Algorithm 1 on different smartphone models, we have set up the following test environment. Four Raspberry Pis,[3] denoted as R(i), $i = 1 \ldots 4$ continuously transmitted BLE advertising packets. Each Raspberry Pi was configured to transmit on a different set of channels, viz., R(1) on channel 37, R(2) on channel 38, R(3) on channel 39 and R(4) on all three channels,

[3]In our experiments with Samsung Galaxy S5 and Xiaomi MI-A2 smartphones, these Raspberry Pis were replaced by Bluegiga BLE112 radios, which carried out the same tasks as the Raspberry Pis.

as also shown in Table 3. The advertising interval was set to 100 ms, which is way below d_s (cf. Table 2). Hence, multiple advertising packets will fall into every scan window. Note that $T_a = 100$ ms is used by the ADVERTISE_MODE_LOW_LATENCY setting on Android smartphones. Therefore, though having been obtained using Raspberry Pis as senders, our results remain valid when packets are sent using smartphones. It is worth mentioning that we have tested our proposed methodology also with all other advertising intervals supported by Android (viz., 100 ms and 1000 ms), and found it to be working successfully irrespective of the value of T_a used.

In BLE, the services a device offers are advertised with the payload of its packets. Such services are identified by unique identifiers, so-called *Universally Unique Identifiers (UUIDs)*. We have assigned a different UUID to the packets of each Raspberry Pi. Note that this is only possible when a Raspberry Pi is used, which provides direct access to the BLE API. On a smartphone, it is not possible to emit packets only on a certain, adjustable set of channels. Hence, this technique cannot be used to detect the advertising channel when two smartphones estimate their distance. We use it as a ground-truth for our evaluation instead.

TABLE 3. Experimental setup with 4 Raspberry Pis advertising on different channels.

Raspberry Pi	Channel	Center Frequency in [GHz]
R(1)	37	2.402
R(2)	38	2.426
R(3)	39	2.480
R(4)	37,38,39	2.402, 2.426, 2.480

B. RESULTS

1) BEHAVIOR OF THE BLE RADIO

Figure 5a shows the reception times of all packets in an experiment using the SCAN_MODE_LOW_LATENCY setting. Figure 5b depicts the results from an experiment in which the SCAN_MODE_BALANCED setting was used. Both experiments were carried out using a *Google Pixel 3* smartphone. The individual measurements lie in such a close proximity that they appear as lines rather than as individual points, since the time between two adjacent packets is short. We have sorted the packets by their UUIDs, which identify their channel. The different background colors in the figure depict the estimated instances of the scan interval and their channel, as given by Equation (6). Assuming that scanning started on channel 37, red indicates that in this interval, the scan window was listening on channel 37. Similarly, green corresponds to channel 38 and blue to channel 39. Note that the colors of the depicted received packets do not follow this scheme, since they would be indistinguishable if they had the same color as the background.

In both experiments, scanning was activated at time $t = 0$ s. In the SCAN_MODE_LOW_LATENCY setting, as can be seen in Figure 5a, the received packets follow the pattern predicted by Equation (6), which is exploited by Algorithm 1. In particular, starting from time $t = 0$ s, packets are received on channel 37 for T_s time units. Next, a sequence of packets is received on channel 38, then channels 39, 37, 38, 39, ...

In contrast, when using the SCAN_MODE_BALANCED setting, we found that the first "regular" scan window begins after a certain offset t_o from the starting time, which is indicated by the white background color in Figure 5b. Before t_o has passed, the smartphone scans using an unpredictable pattern. In the situation exemplified in Figure 5b, the device first scans on channel 37 and then on channel 38, each for an amount of time that exceeds d_s. After t_o has passed, the device re-sets to channel 37 and the scheme as given by Equation (5) begins. Hence, if the offset t_o would be known, the channel could be classified correctly according to Equation (6). However, our measurements with different smartphone models showed that this offset t_o varies every time the scanning procedure is started, and in addition depends on the smartphone model. This justifies that our algorithm is built upon the SCAN_MODE_LOW_LATENCY setting for detecting the channel of an incoming BLE signal, despite this setting being the most power-hungry one on Android devices. Since we only switch to this mode for short amounts of time after an initial reception (i.e., until the channel has been identified and a sufficient number of packets for computing the distance have been received), the energy overhead of this will be acceptable. Recent results [8] show that - depending on the smartphone model - the battery of the smartphone is drained by between 5 % and 20 % earlier compared to when Bluetooth is switched off, if the SCAN_MODE_LOW_LATENCY setting is used during all times. Since we only use this mode for small fractions of the time, the reduction of the battery runtime will be way below this.

It is worth mentioning that though it needs almost 10 s until a packet has been received on all 3 channels at least once, distance estimation can be done earlier, since not all channels need to be exploited. The improvement in accuracy stems from the information on which channel a packet has been received.

2) CLASSIFICATION ACCURACY

While Figure 5 shows that the behavior of the BLE radio on multiple smartphone models appears to be suitable for channel detection, in which fraction of all attempts can the channel be classified correctly using Algorithm 1? In order to evaluate the success rate of the channel detection algorithm, which we call the *detection accuracy*, Figure 6 shows the fraction of packets for which the channel was detected correctly as a function of the time since the last re-start of the scanning procedure for the *Google Pixel 2* and *Google Pixel 3* smartphones. Both smartphones continuously recorded the received BLE packets of the Raspberry Pis for 24 h. Scanning was re-started every 30 min. This re-starting was necessary because the Android operating system automatically switches from the SCAN_MODE_LOW_LATENCY to the SCAN_MODE_OPPORTUNISTIC setting after

TABLE 4. Tested Android devices.

Device Name	Model	Compatible	Classification Accuracy	Android Version	Comments
Google Pixel 2	Pixel 2	✓	100 %	10	-
Google Pixel 3	Pixel 3	✓	100 %	10	-
Google Pixel 4a (5G)	Pixel 4a (5G)	✓	100 %	11	-
Google Pixel 5	Pixel 5	✓	100 %	11	-
OnePlus 5	ONEPLUS A5000	✓	100 %	9	-
Samsung Galaxy S5	SM-G900F	✓	99.5%	8.1.0	Older API: $T_s = 5$ s, Lineage OS.
Xiaomi MI-A2	M1804D2SG	✓	100 %	9	-
Xiaomi Mi 9T Pro	M1903F11G	✓	100 %	10	-
Huawei P20 lite	ANE-LX1	✗	-	9	Different scan intervals, no channel reset
Samsung Galaxy M20	SM-M205FN	✗	-	10	Different scan pattern, no channel reset
iPhone 6s	MN1E2LL/A	✗	-	13.5.1 (iOS)	No channel reset

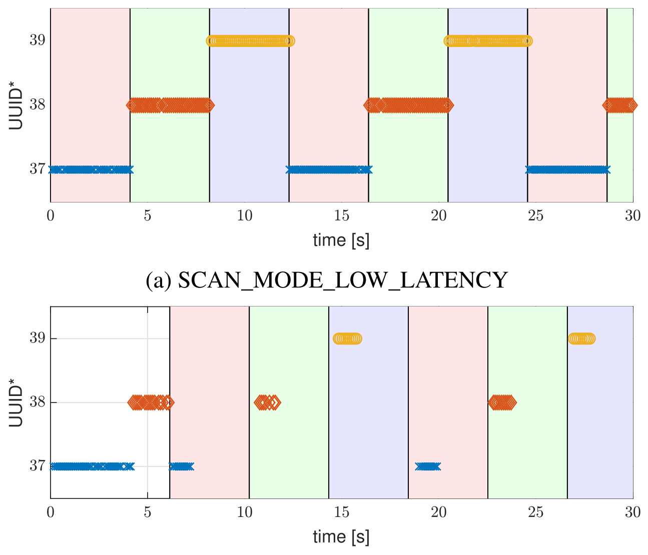


FIGURE 5. Sequence of received packets classified by their channel of reception on the *Google Pixel 3* smartphone. The different background colors indicate the estimated channel based on Equation (6). Four Raspberry Pis transmitted BLE advertisement packets on different frequencies/channels, see Table 3. Unique UUIDs allowed us to detect on which channel a packet was received.

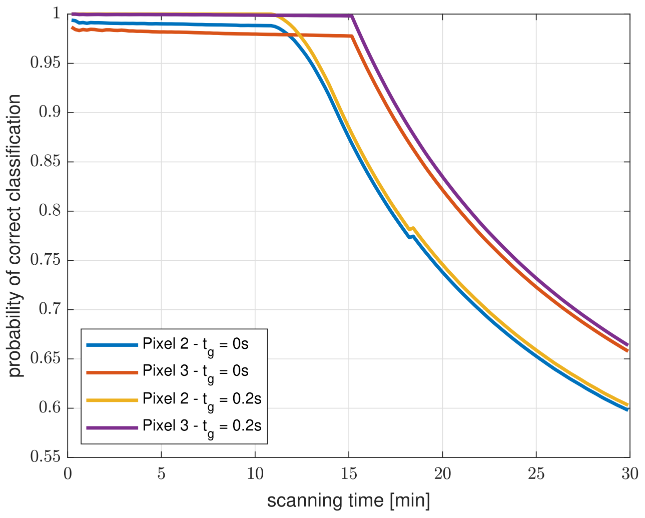


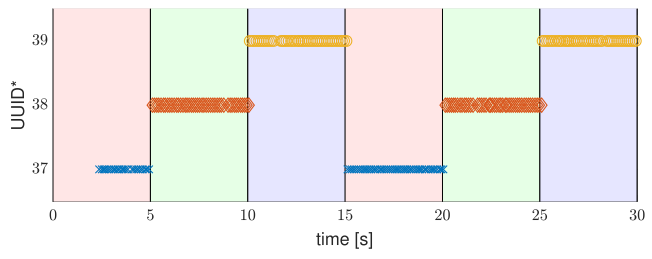
FIGURE 6. Probability of classifying the channel correctly (detection accuracy) as a function of the scanning time without re-starting for the Pixel 2 and Pixel 3 smartphones. Two guard times $t_g = 0$ s and 0.2 s were applied. The BLE scanner was re-started at $t = 0$ s.

30 min of continuous scanning. In the SCAN_MODE_OPPORTUNISTIC setting, the device only schedules scan windows when a different app explicitly triggers the scanning (i.e., by using a different mode than SCAN_MODE_OPPORTUNISTIC), which would have interrupted our measurements.

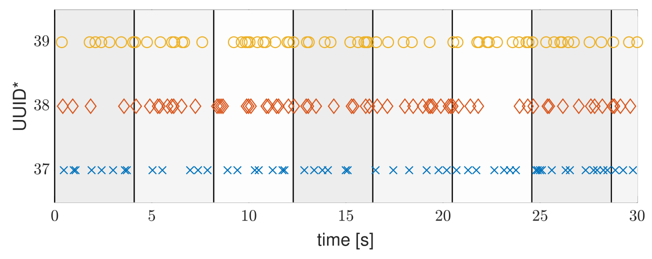
We used a guard time of $t_g = 0$ s and $t_g = 0.2$ s, respectively. As can be seen in Figure 6, by using a guard time of $t_g = 0.2$ s, we obtain a detection accuracy of 100 % for more than 10 min of contiguous scanning without re-starting for the Pixel 3 smartphone, and 15 min for the Pixel 2 smartphone. For a guard time of $t_g = 0$ s, the initial detection accuracy is slightly reduced to around 97 %. When scanning is carried out for more than 10 min without reset, the detection accuracy gradually becomes smaller for both values of t_g. This is caused by clock drift between the smartphone and radio, as already described. We therefore propose to set t_g to 0.2 s and *Max-Scan-Time* in Algorithm 1 to 10 min. In general, for a larger value of t_g, Algorithm 1 can be run for a longer time without the need of a reset of the scanning procedure. On the other hand, an increasing number of packets need to be discarded, if t_g is increased.

3) DIFFERENT SMARTPHONE MODELS

We have shown that our proposed algorithm works in principle, but will it work for *all* smartphone models from different manufacturers? Since the observation of smartphones always starting on channel 37 after a reset is an unspecified behavior, this requires further investigation. For answering this question, we have tested different smartphone models for their compatibility with our proposed methodology. In particular, we have tested whether they always start scanning on channel 37 and then switch to the next channel after every instance of T_s. We also evaluated the detection accuracy for each of them. Table 4 summarizes the results of this experiment. Out of 11 smartphones from different manufacturers we have tested, Algorithm 1 is compatible with 8. Table 4 shows the classification accuracy for a recording time of 7 min and a guard time of $t_g = 0.2$ s, similarly to Figure 6. Here,



(a) Samsung Galaxy S5



(b) Huawei P20 Lite

FIGURE 7. Sequences of received packets classified by their channel of reception over time in the SCAN_MODE_LOW_LATENCY setting for the Samsung Galaxy S5 and the Huawei P20 Lite smartphones. The different background colors in the upper part of the figure indicate the estimated channel of the scan intervals given by Equation (6).

we re-started the scanning procedure after every minute, since we feel that a 1 min time-frame is practical for estimating the distance between a pair of phones. We obtained a classification accuracy of 100% for 7 of the 8 compatible devices, while the accuracy for the remaining smartphone (viz., the Galaxy S5) is only reduced by 0.5 %. These results can be further improved when tweaking the parameters t_g and the interval after which the scanning is re-started individually for each smartphone model. For most smartphones, a classification accuracy of 100 % can be maintained for a longer duration than 1 minute (cf. Figure 6).

Below, we discuss multiple smartphone models that exhibited an abnormal behavior in detail.

a: SAMSUNG GALAXY S5

The Samsung Galaxy S5 we have tested used an older version of Android, in which the scan intervals that were actually used differ from the ones in the most recent Android version. Nevertheless, our proposed algorithm works successfully when adjusting T_s. The results of this experiment are shown in Figure 7a. Similarly to Figure 5, the received packets are classified by their channel of reception in Figure 7a. In our experiment, the SCAN_MODE_LOW_LATENCY setting was used. As can be seen, when adjusting the scan interval, the channel can be tracked reliably over time.

b: HUAWEI P20 LITE

The Huawei P20 lite and the Samsung Galaxy M20 smartphones were the only two Android smartphones we have tested, on which our proposed methodology did not work. In our experiments, we found that the Huawei P20 Lite smartphone used a scan window between 100 ms and 200 ms. Figure 7b depicts the results of this experiment with a Huawei P20 smartphone, in which the SCAN_MODE_LOW_LATENCY setting was used. As can be seen, the channel is switched much more frequently than expected. However, we could not identify a predictable toggling behaviour of the Huawei P20 smartphone in our experiments. Hence, our algorithm was not able to classify the BLE signals into $\hat{I}_{37}$, $\hat{I}_{38}$ or $\hat{I}_{39}$ using Equation (6).

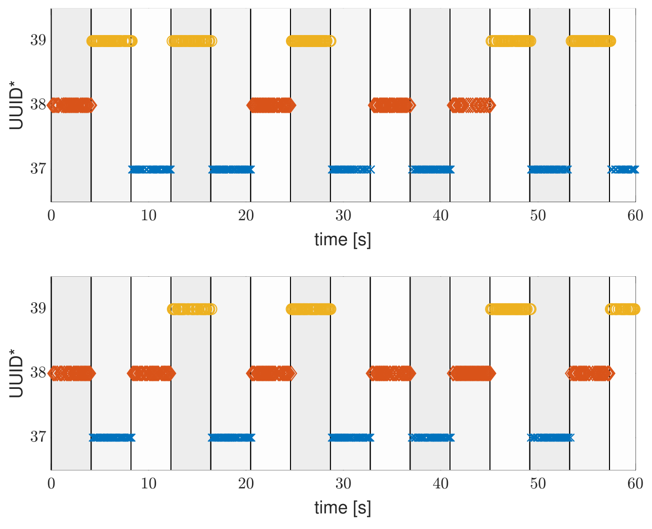


FIGURE 8. Two different sequences of received packets classified by their reception channel over time for the Samsung Galaxy M20 smartphone. Here, the SCAN_MODE_LOW_LATENCY setting was used.

c: SAMSUNG GALAXY M20

Figure 8 depicts the received packets classified by their reception channels in two different experiments using the Samsung Galaxy M20 smartphone. The Samsung Galaxy M20 smartphone also uses scan windows of length $d_s = 4.096$ s in the SCAN_MODE_LOW_LATENCY setting. However, as can be seen, the channels are toggled in a different order than specified by the BLE specification. For example, in the upper part of the figure, the device started scanning on channel 38, then switched to channel 39, then to channel 37 and then to 39 again. The lower part of the figure depicts another such abnormal scanning sequence. In addition to this abnormality, the starting channel is not reset when the scanning procedure is re-started.

d: APPLE IPHONE 6S

We also wrote a dedicated Apple iOS app for testing our approach on such devices. Using this app, we could reveal the following behavior on an iPhone 6s. After resetting the scanning procedure, the device always continues to scan on the channel on which it had scanned at the time it was stopped. Clearly, this prevents using our proposed methodology. In addition, the scan windows were scheduled in an unpredictable manner in our experiments. In particular, the scan interval appeared to increase over time. Furthermore, unlike on Android, iOS does not allow an app to choose among different settings that determine T_s and d_s. Finally, transmitting in the background, i.e., when the screen is locked, is not possible on iOS.

VII. CONCLUDING REMARKS

As a part of the Bluetooth Low Energy (BLE) standard, the host control interface (HCI) specifies how a Bluetooth radio on a smartphone communicates with the phone. As a result of this specification, the information on which wireless channel a packet has been received is not made available to a smartphone. Fortunately, as we have shown in this paper, the behavior of BLE radios used on many recent smartphone models allows for reliably detecting the channel indirectly, since the radio always starts scanning on channel 37 after a re-start. Hence, we in this paper, for the first time, proposed a solution to reliably detect the wireless channel on which a packet has been received on a smartphone. We also showed that the channel on which the smartphone scans can be tracked over time by an Android application after re-starting the BLE scanning procedure. The proposed technique was experimentally evaluated on multiple different smartphone models. In particular, we showed that a probability of detecting the correct channel of 100 % can be obtained for the *Google Pixel 2* and *Google Pixel 3* smartphones, and similar probabilities for other smartphone models.

The information on which channel a packet has been received is highly relevant for distance estimation. In particular, the error induced by the frequency-dependent gains G_r and G_t, as well as the frequency dependent signal propagation in free space, can be cancelled easily. Also the issue of multi-path propagation, which can occur especially in indoor environments, can be mitigated when the channel on which a packet was received is known. Our proposed approach therefore helps towards reliably estimating the distance between two phones.

APPENDIX

Similar to [5], the parameters a and b of Equation (3) can be estimated as described in the following. We obtain the sum of the model error squares from Equation (3) as

$$E(a,b) = \sum_{j=1}^{N_M} \left(\log_{10}\left(d_j(\hat{P}_{r,j})\right) - \log_{10}\left(\hat{d}_j\right)\right)^2 \qquad (7)$$

In the equation above, N_M is the number of calibration measurements, $\hat{d}_j$ is the actual distance, $d_j(\hat{P}_{r,j})$ is the estimated distance and $\hat{P}_{r,j}$ is the measured received signal strength indicator (RSSI) value of the jth calibration point. Equivalently to [5], from the criterion of least squares estimation, $E(a,b)$ has to be minimized, hence,

$$\frac{\partial E(a,b)}{\partial a} = 2\sum_{j=1}^{N_M}\left(a + b\hat{P}_{r,j} - \log_{10}\left(\hat{d}_j\right)\right) = 0 \qquad (8)$$

and

$$\frac{\partial E(a,b)}{\partial b} = 2\sum_{j=1}^{N_M}\left(a + b\hat{P}_{r,j} - \log_{10}\left(\hat{d}_j\right)\right)\hat{P}_{r,j} = 0 \qquad (9)$$

From Equation (8) and Equation (9) we obtain

$$aN_M + b\sum_{j=1}^{N_M}\hat{P}_{r,j} = \sum_{j=1}^{N_M}\log_{10}\left(\hat{d}_j\right) \qquad (10)$$

$$a\sum_{j=1}^{N_M}\hat{P}_{r,j} + b\sum_{j=1}^{N_M}\hat{P}^2_{r,j} = \sum_{j=1}^{N_M}\log_{10}\left(\hat{d}_j\right)\hat{P}_{r,j}, \qquad (11)$$

which can be written in matrix representation as

$$\mathbf{M}\cdot\begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{x} \qquad (12)$$

where

$$\mathbf{M} = \begin{bmatrix} N_M & \sum_{j=1}^{N_M}\hat{P}_{r,j} \\ \sum_{j=1}^{N_M}\hat{P}_{r,j} & \sum_{j=1}^{N_M}\hat{P}^2_{r,j} \end{bmatrix}, \qquad (13)$$

$$\mathbf{x} = \begin{bmatrix} \sum_{j=1}^{N_M}\log_{10}\left(\hat{d}_j\right), & \sum_{j=1}^{N_M}\log_{10}\left(\hat{d}_j\right)\hat{P}_{r,j} \end{bmatrix}^T. \qquad (14)$$

Finally, we can calculate a and b by

$$\begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{M}^{-1}\cdot\mathbf{x}. \qquad (15)$$

ACKNOWLEDGMENT

The authors would like to thank Prof. Dr. Christoph Günther, Dr. Armin Dammann, and Prof. Dr. Samarjit Chakraborty for many interesting discussions on this paper.

REFERENCES

[1] G. Shi and Y. Ming, "Survey of indoor positioning systems based on ultra-wideband (UWB) technology," in *Wireless Communications, Networking and Applications*. Springer, 2016, pp. 1269–1278, doi: 10.1007/978-81-322-2580-5_115.

[2] C. Gentner and M. Ulmschneider, "Simultaneous localization and mapping for pedestrians using low-cost ultra-wideband system and gyroscope," in *Proc. Int. Conf. Indoor Positioning Indoor Navigation (IPIN)*, Sapporo, Japan, Sep. 2017, pp. 1–8.

[3] C. Gentner, M. Ulmschneider, I. Kuehner, and A. Dammann, "WiFi-RTT indoor positioning," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Apr. 2020, pp. 1029–1035.

[4] D. Giovanelli and E. Farella, "RSSI or time-of-flight for Bluetooth low energy based localization? An experimental evaluation," in *Proc. 11th IFIP Wireless Mobile Netw. Conf. (WMNC)*, Sep. 2018, pp. 1–8.

[5] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy, "Smartphone-based indoor localization with Bluetooth low energy beacons," *Sensors*, vol. 16, no. 5, p. 596, Apr. 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/5/596

[6] Bluetooth SIG. (Jun. 2011). *Bluetooth Profile Specification—Find Me Profile*. [Online]. Available: https://bluetooth.org

[7] C. Günther, M. Günther, and D. Günther, "Tracing contacts to control the COVID-19 pandemic," 2020, *arXiv:2004.00517*.

[8] P. H. Kindt, T. Chakraborty, and S. Chakraborty, "How reliable is smartphone-based electronic contact tracing for COVID-19?" *Commun. ACM*, vol. 65, no. 1, pp. 56–67, Jan. 2022.

[9] T. I. Chowdhury, M. M. Rahman, S.-A. Parvez, A. K. M. M. Alam, A. Basher, A. Alam, and S. Rizwan, "A multi-step approach for RSSi-based distance estimation using smartphones," in *Proc. Int. Conf. Netw. Syst. Secur. (NSysS)*, Jan. 2015, pp. 1–5.

[10] S. Liu and A. Striegel, "Accurate extraction of face-to-face proximity using smartphones and bluetooth," in *Proc. 20th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2011, pp. 1–5.

[11] J. Powar, C. Gao, and R. Harle, "Assessing the impact of multi-channel BLE beacons on fingerprint-based positioning," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Sep. 2017, pp. 1–8.

[12] B. Huang, J. Liu, W. Sun, and F. Yang, "A robust indoor positioning method based on Bluetooth low energy with separate channel information," *Sensors*, vol. 19, no. 16, p. 3487, Aug. 2019.

[13] F. Sattler, J. Ma, P. Wagner, D. Neumann, M. Wenzel, R. Schäfer, W. Samek, K.-R. Müller, and T. Wiegand, "Risk estimation of SARS-CoV-2 transmission from Bluetooth low energy measurements," 2020, *arXiv:2004.11841*.

[14] D. J. Leith and S. Farrell, "Coronavirus contact tracing: Evaluating the potential of using Bluetooth received signal strength for proximity detection," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 4, pp. 66–74, 2020.

[15] H. T. Friis, "A note on a simple transmission formula," *Proc. IRE*, vol. 34, no. 5, pp. 254–256, May 1946.

[16] A. Alomainy, Y. Hao, A. Owadally, C. G. Parini, Y. Nechayev, C. C. Constantinou, and P. S. Hall, "Statistical analysis and performance evaluation for on-body radio propagation with microstrip patch antennas," *IEEE Trans. Antennas Propag.*, vol. 55, no. 1, pp. 245–248, Jan. 2007.

[17] Bluetooth SIG. (Dec. 2019). *Specification of the Bluetooth System 5.2*. [Online]. Available: https://bluetooth.org

[18] C. Gentner, T. Jost, W. Wang, S. Zhang, A. Dammann, and U.-C. Fiebig, "Multipath assisted positioning with simultaneous localization and mapping," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6104–6117, Sep. 2016.

[19] D. Giovanelli and E. Farella, "RSSI or time-of-flight for Bluetooth low energy based localization? An experimental evaluation," in *Proc. 11th IFIP Wireless Mobile Netw. Conf. (WMNC)*, Sep. 2018, pp. 1–8.

[20] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[21] M. Hatay, "Empirical formula for propagation loss in land mobile radio services," *IEEE Trans. Veh. Technol.*, vol. VT-29, no. 3, pp. 317–325, Aug. 1980.

[22] Google. (2021). *Wi-Fi Location: Ranging With RTT*. [Online]. Available: https://developer.android.com/guide/topics/connectivity/wifi-rtt#supported-phones

[23] S. Jeong, S. Kuk, and H. Kim, "A smartphone magnetometer-based diagnostic test for automatic contact tracing in infectious disease epidemics," *IEEE Access*, vol. 7, pp. 20734–20747, 2019.

[24] B. Thiel, K. Kloch, and P. Lukowicz, "Sound-based proximity detection with mobile phones," in *Proc. 3rd Int. Workshop Sens. Appl. Mobile Phones (PhoneSense)*, 2012, pp. 1–4.

[25] I. Carreras, A. Matic, P. Saar, and V. Osmani, "Comm2Sense: Detecting proximity through smartphones," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom)*, Mar. 2012, pp. 253–258.

[26] S. Liu, Y. Jiang, and A. Striegel, "Face-to-face proximity estimation using Bluetooth on smartphones," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 811–823, Apr. 2014.

[27] N. Palaghias, S. A. Hoseinitabatabaei, M. Nati, A. Gluhak, and K. Moessner, "Accurate detection of real-world social interactions with smartphones," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 579–585.

[28] D. Kim, S. Kim, D. Choi, and S.-H. Jin, "Accurate indoor proximity zone detection based on time window and frequency with Bluetooth low energy," in *Proc. Int. Conf. Mobile Syst. Pervasive Comput. (MobiSPC)*, 2015, pp. 88–95.

[29] P. C. Ng, J. She, and R. Ran, "A compressive sensing approach to detect the proximity between smartphones and BLE beacons," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7162–7174, Aug. 2019.

[30] I. Bisio, A. Sciarrone, and S. Zappatore, "A new asset tracking architecture integrating RFID, Bluetooth low energy tags and ad hoc smartphone applications," *Pervasive Mobile Comput.*, vol. 31, pp. 79–93, Sep. 2016.

[31] D. Leith and S. Farrell, "Measurement-based evaluation of Google/Apple exposure notification API for proximity detection in a commuter bus," *CoRR*, vol. abs/2006.08543, pp. 1–8, Jun. 2020.

[32] K. Kaemarungsi and P. Krishnamurthy, "Modeling of indoor positioning systems based on location fingerprinting," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, vol. 2, Mar. 2004, pp. 1012–1022.

[33] R. Faragher and R. Harle, "Location fingerprinting with Bluetooth low energy beacons," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 11, pp. 2418–2428, Nov. 2015.

[34] G. De Blasio, A. Quesada-Arencibia, C. R. García, J. C. Rodríguez-Rodríguez, and R. Moreno-Díaz, "A protocol-channel-based indoor positioning performance study for Bluetooth low energy," *IEEE Access*, vol. 6, pp. 33440–33450, 2018.

[35] P. H. Kindt, M. Saur, M. Balszun, and S. Chakraborty, "Neighbor discovery latency in BLE-like protocols," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 617–631, Mar. 2018.

[36] Apple and Google. (Apr. 2020). *Exposure Notification Bluetooth Specification*. [Online]. Available: https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf

CHRISTIAN GENTNER received the M.Sc. and Dr.-Ing. (Ph.D.) degrees in electrical engineering from the University of Ulm, in 2009 and 2018, respectively. During his B.A. study, he received practical experience at Rohde & Schwarz, Munich. Since 2009, he has been working at the Institute of Communications and Navigation, German Aerospace Center (DLR). His current research interests include multipath assisted positioning and indoor positioning.

DANIEL GÜNTHER is currently pursuing the bachelor's degree in computer science with the Technical University of Munich (TUM). His research interest includes broad spectrum of information technologies.

PHILIPP H. KINDT received the Ph.D. degree in electrical engineering from the Technical University of Munich (TUM), in 2019. He is currently an Assistant Professor ("Juniorprofessor") of pervasive computing systems at the Chemnitz University of Technology, Germany. His research interests include wireless communications, mobile computing, and the IoT.