DOI:10.1145/3471933

A look through the lens of neighbor discovery protocols reveals significant potential to improve electronic contact-tracing accuracy.

BY PHILIPP H. KINDT, TRINAD CHAKRABORTY,
AND SAMARJIT CHAKRABORTY

How Reliable Is Smartphone-Based Electronic Contact Tracing for COVID-19?

WITH GLOBAL SURGES of the novel coronavirus SARS-CoV-2 in 2020 and 2021, electronic contact tracing has been adopted in different countries, the goal being to identify the most relevant contacts with a reasonable reliability. Owing to the need to quickly reduce the number of transmissions, contact-tracing solutions built on smartphones were developed because they could be mass-deployed on short notice. Their major advantage was that the hardware was already deployed and only the software remained to be developed.

Studies have shown that smartphone-based solutions can indeed help to control the spread of infectious diseases.³⁰ However, as we discuss in this article, they cannot reliably record *all* contacts. Furthermore,

the classification of whether a contact is relevant is based on contact duration and proximity, which can only be determined with limited granularity using smartphones. On the other hand, dedicated contact-tracing devices were not available immediately because they required the development and production of custom hardware. Such custom-designed hardware in the form of wearable devices, along with custom-built communication protocols for contact tracing, can potentially overcome many limitations of smartphones. While such devices are being developed, they have not attracted sufficient attention of public health policymakers, and very little discussion about them has reached the public sphere. Moreover, many medical professionals and policymakers are not adequately aware of the limitations of smartphones or the advantages of dedicated wearable contact-tracing devices. The goal of this article is to highlight this gap, especially considering that contact tracing will be an important requirement in the future, even after we overcome this ongoing COVID-19 crisis.

When modeling the spread of infections in a pandemic, every infected person is considered to infect R others, where R is the effective reproduction rate. Recent studies—for example, Ferretti et al.¹¹—suggest that the value of R can be reduced using electronic con-

» key insights

- **Smartphones enable contact-tracing apps to be deployed quickly at the onset of a disease outbreak. However, while they support reasonable tracing, the fraction of all contacts they can reliably record is subject to limitations.**
- **Reliable contact tracing involves both contact duration and distance estimation. No matter how well a contact-tracing app is designed, neither can be estimated with a high granularity using smartphones.**
- **The reliability of electronic contact tracing could be improved with dedicated wearable devices that use techniques beyond Bluetooth, such as ultra-wideband and time-of-flight. Such devices could be used for a very long time without having to be recharged.**

IMAGE COMPOSITE BY ANDRIJ BORYS ASSOCIATES, USING PHOTO BY WILLIAM PERUGINI/SHUTTERSTOCK.COM

tact tracing. The extent of this reduction depends on two factors. The first, adoption rate, is the percentage of the population that uses a smartphone-based contact-tracing app. The second factor is determined by the probability with which a smartphone can reliably detect and classify a contact, which again depends on multiple factors discussed later in this article.

Much of the current discussion about smartphone-based contact tracing's success has focused on user adoption rates. However, most work on modeling the effect of electronic contact tracing has implicitly assumed that whenever a smartphone is used, it reliably registers all or at least a certain percentage of relevant contacts. The validity of this assumption has not been sufficiently studied and is the focus of this article.

Reliability of Contact Tracing

Multiple factors influence contact-tracing reliability. As mentioned, a sufficiently high percentage of the population must install an app. There is also a minimum duration for which two devices need to be within range to be able to detect each other. This article addresses that aspect in detail by quantifying such durations and by studying secondary aspects, such as corresponding battery runtimes. Additionally, when multiple smartphones are within reception range, their emitted signals interfere with each other, thereby potentially preventing their mutual detection. We also study the impact of collisions on the discovery procedure. Lastly, after two devices have discovered each other, they estimate their distance to evaluate the transmission risk. The accuracy of this estimation also contributes to reliability.

As already mentioned, electronic contact tracing will remain important beyond the coronavirus pandemic, and the notion of tracing reliability will depend on the use case. For example, at the onset of a pandemic, when very few people are infected, the goal is to reliably record *each* contact a person has. Here, the overall number of quarantined people will remain low, even when all contacts—including very brief ones—require a person be isolated. Similarly, other use cases—for instance,

A contact-tracing system should ideally be able to record every encounter between two subjects and then estimate the risk based on the recorded data.

in a hospital—require all contacts be reliably recorded.

On the other hand, in a situation with many infected people, contact-tracing systems need to distinguish between relevant and irrelevant contacts. Current-tracing apps do this based on contact duration and distance. Here, reliability is defined by the granularity at which the contact duration can be measured, as well as by the accuracy of distance estimation. As discussed in this article, smartphones cannot *reliably* record *all* contacts, and the accuracy of the contact classification procedure has certain limitations. There is also little understanding within the broader public health and information technology communities about the factors affecting the reliability of smartphone-based contact tracing.

Neighbor Discovery— The Contact-Tracing Basis

Because localization systems, such as global positioning systems (GPSs), are inaccurate inside buildings, devices within proximity are detected using short-range wireless signals. The mechanism that lies at the heart of a smartphone's ability to detect the existence of another smartphone in its vicinity is called neighbor discovery (ND). It is based on a phone emitting and scanning for Bluetooth signals; successful reception by another phone and vice-versa will lead to mutual discovery. Both the specific ND protocol and the way it is parameterized determine the discovery latency—that is, how quickly two smartphones can discover each other when they come in close contact. Energy consumption and battery runtime also play a role. Other properties, such as the reliability of operation when many different phones are within reception range, are also determined in this way. However, smartphones constrain the degrees of freedom when realizing a protocol for ND, such that these aspects cannot be optimized by an app.

Smartphone Limitations

Hence, when a smartphone is used for contact tracing, a relevant question is: whenever two or more people come in contact, what is the probability that their respective smartphones can record that encounter? Will the

devices still be detected if the contact duration is very brief—for instance, a couple of seconds? Will smartphones be able to detect the type and proximity of the contacts? For example, were the two subjects within 5 m of each other or at less than 0.1 m? Answers to these questions in the context of smartphone-based contact tracing are unclear. Nevertheless, contact-tracing apps are now seen as the Holy Grail solution to this problem. However, what such an app might or might not be able to do will be fundamentally limited by the underlying hardware and protocol configurations of ND mechanisms supported by the smartphone.

Aside from the success probability of the ND procedure itself, the estimation of the distance between two subjects is susceptible to errors. This estimation is based on sensing the attenuation of the wireless signal, which in free space correlates to the distance between two devices. However, signals can be reflected in the environment or attenuated by the human body. Hence, contacts that are not relevant could be misclassified as such—resulting in false positives—while relevant contacts might not be registered. Too many false positives would lead to testing and/or isolating uninfected people, hence a sufficient safety margin must be built into the distance estimation procedure, which will lead to a higher rate of unregistered relevant contacts.

In this article, we attempt to address these questions. We systematically evaluate the suitability of ND configurations in commercially available smartphones for the purpose of electronic contact tracing. Our study exposes the fundamental limits that any smartphone will have, no matter which contact-tracing app is used. We finally argue that smartphones are not suitable for “gapless” contact tracing. While performance of the ND procedure on smartphones could improve if smartphone operating systems would grant unrestricted access to the phone’s Bluetooth capabilities, distance estimation will nevertheless remain unreliable. Driven by this insight that detection reliability is a critical variable, we discuss how the limitations of smartphones can be overcome using existing and future wearable solutions.

In summary, our main contributions are as follows: (a) We debunk the currently held notion that smartphones can reliably register every contact and that the only obstacle is their adoption. (b) To that end, we lay the foundations for quantifying the reliability and accuracy of contact tracing using currently available smartphones. (c) We briefly discuss existing and future wearables for contact tracing, which can resolve some smartphone limitations.

The remainder of the article offers a brief description of the theoretical foundations of energy and latency-optimal solutions for contact tracing, followed by an evaluation of the performance of contact tracing using currently available iOS and Android smartphones. Then, we discuss wearable devices for contact tracing.

Neighbor Discovery

In this section, we illustrate the design space of the ND procedure, which underlies electronic contact tracing. It involves multiple trade-offs—for example, latency versus energy consumption versus resilience in crowded situations. Any smartphone application for contact tracing will build on a restricted version of this procedure, which we study later, in the section “Discovery on Smartphones.”

Let us first consider two wireless devices that are unaware of their mutual presence but would like to discover each other as soon as they are in proximity. One of them acts as a sender and the other as a receiver. The sender continually broadcasts beacons, while the receiver continually listens to the channel for certain time intervals. All transmissions and receptions are scheduled following a certain, repetitive pattern. The receiver discovers the sender once a beacon is sent within a reception window of the receiver.

The main reason for both transmission and reception to be continual and not continuous is energy. To save energy, wireless radios go into a “sleep” or “power-down” mode when not sending or receiving. From the perspective of ND, the energy consumption of each device is determined by the fraction of time spent transmitting or receiving—that is, by the duty-cycle for transmission β and for reception γ . For a ND

protocol to be efficient, the goal is to identify a transmission-and-reception pattern that, for a given sum, $\beta + \gamma$, minimizes the time until a beacon is guaranteed to overlap with a reception window in the worst case. Several published papers—for example, Dutta and Culler⁹ or Kindt, et al.¹⁴—have been concerned with optimizing this trade-off between latency and energy. In the context of contact tracing, the ND procedure should guarantee discovery within a short period of time. In other words, it should be able to register contacts even when two people come in proximity for relatively short time intervals—for instance, when shopping at a grocery store or jogging in a park. Next, we discuss the latency within which a smartphone should guarantee the discovery of a remote device.

Required latency for contact tracing.

Attempts to establish the minimum contact duration required for a significant virus or infection transmission risk have not reached a consensus. For example, the Centers for Disease Control and Prevention (CDC) considers close contacts of a cumulative duration of at least 15 mins within 24 hrs as relevant for transmitting SARS-CoV-2. However, data from several studies shows transmission risks also exist for brief contacts. Results obtained using manual contact tracing⁸ indicate five out of 199 transmissions with a contact duration below 15 mins. Moreover, there are reports²⁰ claiming that a fleeting contact of only a few seconds led to an infection with the contagious Delta variant. Hence, there is no clear threshold below which a contact is not relevant for transmission.

Since transmission risk increases with the accumulated contact duration—that is, the sum of the durations of multiple brief contacts—a contact-tracing system should ideally be able to record every encounter between two subjects and then estimate the risk based on the recorded data. Otherwise, multiple brief contacts, which can accumulate to a significant duration, risk going unnoticed. When two subjects pass each other with zero distance at the closest point with, for example, a 4-km/h velocity and assuming a 2-m range for tracing relevant contacts, they would stay within transmission range for 1.8 s. Hence, as a rule of

thumb, contact-tracing systems need to guarantee worst-case latencies of around 2 s for reliably detecting all contacts. While larger latencies in the range of up to tens of seconds might still be sufficient in many practical scenarios, minute-range latencies are not sufficient for gapless tracing.

Latency/energy trade-off. However, being able to do so should not quickly drain the battery of the device. We have previously determined¹⁵ the lowest discovery latency L that any receiver with a duty-cycle of γ and any sender with a duty-cycle of β can achieve, as follows (ω is the transmission duration of a beacon):

$$L = \left\lceil \frac{1}{\gamma} \right\rceil \cdot \frac{\omega}{\beta} + \omega \quad (1)$$

For example, if we require that only contacts lasting for at least two seconds are relevant for disease transmission and need to be registered, then $L = 2$ s. The minimal data to be exchanged by two contact-tracing devices in range is a device identifier. To be able to provide at least one unique ID for each device requires four bytes or more. In addition, a preamble of at least one byte needs to be added, for technical reasons, on most radios. Hence, we assume that a five-byte packet needs to be transmitted for contact tracing on a Bluetooth Low Energy (BLE)-like physical layer, leading to a transmission duration of $\omega = 40$ μ s.

To realize $L = 2$ s with a beacon length of $\omega = 40$ μ s, both devices need to be active for $\beta = \gamma \approx 0.45\%$ of their time.

Which transmission and reception patterns can be used to achieve this performance? The only known patterns that achieve optimal discovery latencies are based on periodic intervals.¹⁴ Here, one device periodically broadcasts beacons with a period T_a , whereas the other device switches on its receiver for a window of d_s time units after every T_s time units. This scheme—with some minor modifications that we describe below—is used by the BLE protocol implemented inside smartphones.

Recall that for one sender and one receiver, the optimal discovery latency for a given energy budget is known (see Equation 1), and this latency is guaranteed in 100% of all discovery attempts. But the moment both devices act as both senders and receivers, the

probability of discovery within the same time interval L now drops. Why is that?

First, when each device both receives and transmits, it is unavoidable that the scheduled points in time of a reception window and a beacon transmission of the same device overlap in time (read Kindt and Chakraborty¹⁵ for details). Since a device cannot receive and transmit at the same time, some transmission or reception windows need to be re-scheduled, which leads to a certain fraction of failed discoveries. Nevertheless, existing protocols can achieve latencies close to L from Equation 1 in more than 99.9% of all attempts for a pair of devices discovering each other using practical duty cycles. However, when multiple devices are in range and transmit beacons, some transmissions might overlap in time and hence fail to be received.

ND in crowded scenarios. There are multiple situations that are potentially relevant for virus transmission in which a larger number of people are in proximity. For example, consider a crowded public bus or ski gondola. As a worst-case scenario, the maximum density of crowds without squashing and tilting the human body has been estimated to be six persons per square meter.²² If we assume that the radio only detects devices within a 2-m range, the worst-case number of people/phones in a collision domain is 75. For a given number of devices, the probability of collisions is determined by the transmission duration of each beacon, ω , and the rate of beacon transmissions of each device, which results in a certain duty cycle for transmission, β . With $\omega = 40$ μ s, when using known approaches for realizing $L = 2$ s—such as Kindt et al.¹⁴—about 50% of all discovery attempts will fail due to colliding beacons.¹⁵ This implies that a significant number of contacts will not be registered. However, if some increase in the worst-case latency L is tolerated, the following two techniques, which are used in smartphones, can be exploited to make ND protocols more robust against collisions.

Reducing the channel utilization. The probability of collisions is given by the fraction of time each device uses the channel—that is, β . If fewer beacons are sent, the collision probability decreases. As a drawback, since

beacons are then sent less frequently, the worst-case latency L will increase, or the duty cycle for reception, γ , needs to be increased to compensate for the reduced β (see Equation 1). Reducing β and, in turn, increasing γ to reach the same L will, however, increase the overall energy consumption.

For example, when choosing a reduced channel utilization of $\beta' = 1/4 \cdot \beta$ and an increased duty cycle for reception of $\gamma' = 4 \cdot \gamma$, the collision probability for the protocol described above—when simultaneously discovering 75 devices—can be reduced from 50% to about 15% without increasing L . For a radio that consumes the same current during transmission as during reception, the lowest worst-case latency for a given energy budget, $\eta = \beta + \gamma$, is achieved for $\beta = \gamma = 1/2(\beta + \gamma)$.¹⁵ Reducing the channel utilization to β' while increasing γ to γ' will result in a larger sum, $\eta' = \beta' + \gamma' = 17/8 \cdot \eta$, and therefore greater energy consumption, while leading to the same L . As we will show, smartphones use a very low channel utilization, $\beta \ll (\beta + \gamma)/2$, thereby achieving low collision rates at the expense of energy consumption.

Decorrelation. A technique used by the BLE protocol is decorrelation, which reduces the chance of multiple, subsequent colliding beacons. Instead of sending beacons with periodic intervals, two consecutive beacons are separated from each other by a fixed amount of time plus a certain random component. Nevertheless, to guarantee the same worst-case latency L , the reception duty-cycle, γ , needs to be increased. Consider, for example, a configuration where $T_a = d_s - \omega$, which has been shown to achieve the smallest possible worst-case latency.¹⁵ Here, a beacon will coincide with every reception window, since the distance between two consecutive beacons does not exceed the reception window length. Now, if T_a is also increased by a random component, $\rho \in [0, b]$, then d_s must also be extended by b time units to ensure that a beacon will still fall within every scan window. Otherwise, another beacon must overlap with a later scan window for realizing discovery, thereby increasing L . However, if one pair of beacons from two different devices collides, a later pair of beacons only

collides with a probability below unity. Now, if multiple beacons coincide with one or more scan windows of a remote device, there is a greater chance that one of them will not collide, and hence, the probability of a successful discovery increases.

In summary, there are multiple degrees of freedom for optimizing the ND procedure. Identifying the optimal trade-off between discovery latency, energy consumption, and success probability is crucial for efficient and reliable contact tracing. Unfortunately, on a smartphone, these degrees of freedom are not exposed to a contact-tracing app, and several protocol parameters are already predetermined. In light of this, the next section examines smartphone performance.

Discovery on Smartphones

On a smartphone, the degrees of freedom are restricted both by the BLE protocol, which is used to realize ND, as well as the Android or iOS operating system. We first describe these restrictions and then study the performance achieved under them. Our goals are to 1) assess tracing performance on existing smartphones, and 2) identify the most suitable parameterizations.

Restrictions on Smartphones

Bluetooth low energy. In BLE, so-called advertising events, within which beacons are sent, are scheduled with a period of T_a .⁶ Reception windows of length d_s are repeated within a period of T_s . Here, T_a is composed of a static part, $T_{a,0}$, plus a random delay of $\rho \in [0, 10 \text{ ms}]$. Furthermore, each advertising event consists of a sequence of three beacons. Each beacon is sent on a dedicated wireless channel; the three channels are the same for all devices and advertising events. The receiver toggles between the three channels after each instance of T_s . The values of $T_{a,0}$, T_s , and d_s can be freely chosen within a large, quasi-continuous range. The three-channel procedure increases the duty-cycle for transmission β by a factor of three (since three beacons are sent every T_a time units). This increased duty cycle leads to essentially the same discovery latency as the original one for typical parameterizations used on smartphones. In BLE, the minimal beacon

Identifying the optimal trade-off between discovery latency, energy consumption, and success probability is crucial for efficient and reliable contact tracing.

length is 16 bytes because of multiple overheads. This further increases β by a factor of around three compared to the protocol we described in the previous section.

In summary, compared to an energy-optimal protocol, BLE introduces an overhead of a factor of approximately six to the active time spent for transmission. An additional overhead to γ is caused by the random delay, ρ , since d_s needs to be increased to guarantee the same worst-case latency. On the other hand, some of these overheads improve the reliability of the discovery procedure. In particular, a cyclic redundancy check (CRC) allows beacon collision detection, and the random delay decorrelates the collision probabilities of multiple, subsequent beacons from each other. Due to fixed energy overheads that do not depend on the number of transmitted bytes, the overhead is lower in terms of energy than in terms of transmission/reception time.

Android and iOS. While BLE allows essentially any configuration for the tuple $(T_{a,0}, T_s, d_s)$, Android constrains the design space to three different settings that determine (T_s, d_s) and another three that determine $T_{a,0}$. There is also a dedicated parameterization for the contact-tracing service offered by Google and Apple, called the Google/Apple Exposure Notification Service (GAEN).⁷ Furthermore, Android provides a batch mode, where multiple discovered devices are reported jointly with a certain delay. This mode supports three additional configurations of (T_s, d_s) , but these are not considered in this article due to lack of available documentation.

On a smartphone, fixed values for these parameters are not always feasible, because the hardware might be forced to vary them during runtime. First, the Bluetooth system-on-a-chip (SoC) might need to carry out advertising and scanning in parallel to other tasks, such as streaming audio to a wireless headphone. As a result, the points in time when both tasks need to be served might overlap. Since the device can carry out only one task at a time, some scheduling is needed to resolve this conflict, which might require adapting the values of T_a , T_s , and d_s online. In other words, the effective values

of T_a , T_s , and d_s that are used could be different from the ones that were chosen by a contact-tracing application. In addition, many smartphones, such as the Samsung Galaxy S1, share the radio and/or the antenna to support IEEE 802.11 (Wi-Fi) and Bluetooth communication.²⁵ However, it is not possible to, for example, transmit a Wi-Fi frame and a BLE beacon at the same time. Indeed, SoCs that combine BLE and Wi-Fi use arbitration mechanisms for this purpose.³ To the best of our knowledge, the exact methods for resolving such conflicts depend on the SoC and potentially vary across smartphone models. In general, there are two possibilities for resolving them. First, the parameters $T_{a,0}$, T_s , and d_s could be chosen, such that no advertising packet or scan window overlaps with any other task. Second, if this is not possible, an advertising packet could be skipped, sent earlier/later, or the parameters $T_{a,0}$, T_s , and d_s might be altered repeatedly on a short-term basis.

The values of T_a , T_s , and d_s used in Android smartphones are not specified in the official documentation—possibly due to a potential need for the short-term changes previously discussed. However, Android is open source software, and information that is not provided in its specification documents can be looked up directly from its source code. Android source code contains different values of $T_{a,0}$, T_s , and d_s , which are summarized in Table 1. An application can select a pair of values for T_s and d_s by choosing one of the SCAN_MODE settings, whereas the value of $T_{a,0}$ can be selected by using one of the ADVERTISE_MODE settings.

Since iOS is closed source software, we cannot obtain its parameter values from the source code. However, Apple recommends certain values of $T_{a,0}$ for gadgets communicating with iOS devices,¹ which are given in Table 2. We assume that iOS devices themselves will also use them. No data is available on T_s and d_s .

Especially on iOS devices, restrictions on using BLE in the background prevent applications from using the original BLE API with the parameter values described above. For example, an early version of the U.K.'s NHS corona app was reported to work reliably only when the screen is unlocked and

Smartphone-based distance estimation cannot always distinguish between relevant and irrelevant contacts.

the app is visible.²⁸ To overcome this, most recent apps use GAEN. Here, the operating system carries out the actual contact tracing, while reporting digests to the tracing apps. GAEN uses its own set of parameter values. The advertising interval is specified to sit between 200 and 270 ms and the scan interval to be at most 5 mins, whereas no information on the scan window is given.⁷ Hence, the exact values that are used are not fully specified. Further, directly measuring these values is complicated by the Google/Apple policy granting GAEN access only to applications that are government authorized. However, experimental studies on GAEN have nevertheless been carried out.^{17,18}

We have estimated the parameter values given in Table 1 from publicly available logs^a of received packets. Here, we have assumed that multiple reported receptions in close temporal proximity belong to the same scan window, allowing us to infer T_a , T_s , and d_s based on 119,550 reported receptions. We have excluded scan windows with few (less than five) receptions. For each pair of handsets contained in the dataset (except the Huawei P Smart, which showed inconsistent results), we have computed the median values of T_a , T_s , and d_s . The values in Table 1 are the mean values of those medians. Parameter values can vary significantly over time. It is worth mentioning that when some other app running on the phone activates BLE scanning in a certain mode, GAEN will reuse the selected parameter values and deviate from the values in Table 1.

Performance evaluation. In this section, we evaluate how BLE ND performs on smartphones, assuming 1) a packet length of 16 bytes and 2) the values from Table 1 for Android and the values of $T_{a,0}$ from Table 2 for iOS.

Discovery latency. We now study the discovery latency for two devices. Most of the $T_{a,0}$ values for Android and iOS fulfill $T_a < d_s$, which is illustrated in Figure 1. Once two devices are within their reception range, the first advertising beacon of one device falls within an arbitrary instance of a scan interval. Since $T_a < d_s$, the latency measured from the first beacon to the successfully received one is limited by roughly

a <https://github.com/sftcd/gaen-pairwise-1m>.

Table 1. Parameter settings supported by Android and GAEN in ms.

Mode	$T_{a,0}$	d_s	T_s
ADVERTISE_MODE_LOW_LATENCY	100	—	—
ADVERTISE_MODE_BALANCED	250	—	—
ADVERTISE_MODE_LOW_POWER	1000	—	—
SCAN_MODE_LOW_POWER	—	512	5120
SCAN_MODE_BALANCED	—	1024	4096
SCAN_MODE_LOW_LATENCY	—	4096	4096
GAEN (mean)	279	3587	259694

Table 2. Advertising intervals $T_{a,0}$ supported by iOS.

$T_{a,0}$		
152.5 ms	211.25 ms	318.75 ms
417.5 ms	546.25 ms	760 ms
852.5 ms	1022.5 ms	1285.0 ms

$T_s - d_s$ time units. Because both devices might have been brought into range by up to $T_{a,0} + 10$ ms before the first beacon was sent, the worst-case latency is bounded by approximately $L = T_s - d_s + T_{a,0} + 10$ ms (see Figure 1). As an example, for the SCAN_MODE_LOW_POWER setting and a value of $T_{a,0} = 100$ ms, the worst-case latency is approximately 4.718 s.

On the other hand, $T_a > d_s$ for some other settings, such as ADVERTISE_MODE_LOW_POWER in combination with SCAN_MODE_LOW_POWER. Here, the first scan window might be missed in some cases, and only a later scan window can lead to a successful discovery. Figure 2a depicts the simu-

lated discovery latencies measured from the time the first beacon is sent after two devices come within range. This is with the SCAN_MODE_LOW_POWER setting. Similarly, Figure 2b depicts the discovery latencies for the SCAN_MODE_BALANCED setting. For every value of $T_{a,0}$ supported by BLE in the depicted range, we have carried out 1,000 simulations. For each of them, we have computed the maximum and mean latencies. The solid, darker-red, vertical lines correspond to the values of $T_{a,0}$ supported by the native BLE interface on Android, and the dashed lines in light red are those supported by iOS.

Let us first consider the SCAN_

Figure 1. Sequence of beacons (red arrows) with $T_a \leq d_s$ falling into an instance of T_s . **Figure 2. Simulation of maximum and mean discovery latencies in Android for different values of $T_{a,0}$. The vertical lines depict values of $T_{a,0}$ used by Android (solid lines) and iOS (dashed lines).**

MODE_LOW_POWER setting. As can be seen, for values with $T_a < d_s$, the maximum latency is approximately 5 secs. However, for $T_a < d_s$, some parameterizations lead to high maximum and mean latencies. For example, for $T_{a,0} = 1,022.5$ ms, which is supported by Android, the maximum resulting latency is 172.5 seconds. For the purposes of gapless contact tracing, such a latency could be unacceptable, since a close contact of less than three minutes could already be highly relevant for a virus transmission. Recall from our previous discussion that because of issues such as resource sharing, smartphones might also deviate from the given parameter values, and there are no guarantees that these maximum latencies are never exceeded by any smartphone model. For the SCAN_MODE_BALANCED setting, the overall situation looks similar (see Figure 2b), but a larger fraction of the values of $T_{a,0}$ leads to latencies below 5 secs.

Different smartphone manufacturers might have altered these values in their adopted versions of Android or might do this in the future. Therefore, compatibility among all different smartphone models—for the purpose of estimating a worst-case dis-

covery latency—is also not guaranteed. Given all of this, we must consider a smartphone as a device that maintains a certain maximum discovery latency in most cases, while no absolute guarantees can be given for the worst case.

For GAEN, the worst-case latency lies around 5 mins. This is because it schedules one scan window that is significantly larger than the distance between two consecutive advertising packets every five minutes at most in the worst case. We found that T_a , T_s , and d_s vary significantly over time, and we also noted instances in the dataset where $T_a > d_s$. Hence, a latency of more than 5 mins might also be possible in rare cases. However, we could not verify from the dataset whether two consecutive packets were indeed scheduled such that their time difference exceeds the shortest measured instance of d_s . Other effects might also be possible and lead to failed receptions, such as packets colliding with those of other devices or devices being intermittently out of range. The 5-min latency bound also limits the granularity of the contact duration estimation to up to 5 mins, which could lead to missing multiple brief contacts.

Table 3. Reduction of the battery runtime by continuous contact tracing using different Android scan modes.

Scan Mode	nRF52832	BLE112
LOW_POWER	0.52–0.57%	2.13–2.35%
BALANCED	1.30–1.35%	5.30–5.52%
LOW_LATENCY	5.20–5.25%	21.14–21.36%
GAEN	0.09%	0.38%

Energy consumption. We now study how the settings from Table 1 impact the energy consumption of Android phones. We have computed the mean current draw I_{BLE} of a BLE radio that scans and advertises according to different settings. The energy needed to wake up the phone's CPU and gather data from the radio is not included.

Different smartphones use different radios with different firmware and, accordingly, they consume different amounts of energy. We computed the energy consumption for two SoCs with published energy data, but we could not find any smartphone using these chips. Smartphones typically use dual-mode chips that support WiFi and Bluetooth, for which detailed energy information is not widely available. Therefore, we studied two different, well-chosen Bluetooth radios, and assumed that the energy consumption of the SoCs used on smartphones lies in between.

First, we considered a Bluegiga BLE112 module, which was among the first BLE radios available. We also considered the more recent Nordic nRF52832 SoC. Estimation for the BLE112 device was carried out using the energy model proposed in Kindt et al.¹⁶ For the nRF52832 SoC, we used the energy model provided by the device manufacturer,²³ which we combined with values from the device's data-sheet.²¹ We have assumed a 3-V supply voltage of the radio.

We looked at I_{BLE} as it relates to the smartphone's mean current draw. For this purpose, we assumed that the smartphone is equipped with a 3,000-

mAh battery. We further assumed that this capacity is drained within 24 hrs when contact tracing is not carried out, which leads to an average current consumption of $I_p = 3,000 \text{ mAh}/24 \text{ h}$ of the smartphone. We can now form the quotient I_{BLE}/I_p , which gives us the percentage of time by which continuous contact tracing will drain the battery earlier. Table 3 shows the results of this computation. The range of values shown for each scan setting comprises all different advertising intervals $T_{a,0}$ supported by Android, iOS, and GAEN. For the BLE112 radio, the SCAN_MODE_LOW_LATENCY setting reduces battery lifetime by about 20%, which will be noticeable in practice.

The SCAN_MODE_BALANCED setting reduces battery lifetime by about 5% and SCAN_MODE_LOW_POWER by about 3%. GAEN drains the battery by less than 0.4% earlier, which is not noticeable to the user. For the nRF51822 radio, energy consumption is approximately 75% lower in all modes of operation.

It should be mentioned that compared to a ND solution with optimal energy efficiency (see Kindt and Chakraborty¹⁵), the parameterizations supported by Android, iOS, and GAEN require significantly more energy to provide the same worst-case latency. However, the overall energy demand of the Bluetooth radio is small compared to the capacity of batteries in currently available smartphones.

Collision behavior. If two or more devices transmit a beacon at the same time, the beacons will overlap and collide. With high probability, the beacons will not be successfully received, even when they coincide with a scan window. This might increase discovery latency or even prevent discovery in some cases. The choice of $T_{a,0}$ impacts the collision rate, since it affects the fraction of time β during which the channel is busy. To quantify this, we have simulated 10,000 discovery procedures for each parameterization. To cover the absolute worst case, we assumed that up to 100 devices can interfere with each other. We considered a discovery procedure successful if at least one non-colliding beacon coincided with a scan window within 10 secs. This simulation can be regarded as a worst-case estimation. Specifi-

Figure 3. Probability of discovery in a crowd of 100 subjects for SCAN_MODE_LOW_POWER.

cally, our simulation considers every pair of simultaneously sent packets by different devices as a collision that leads to a loss of both packets. In practical setups, one of these two packets might still be successfully received—for instance, because the signal of the interfering packet is lower-power than the successful one. The purpose of this study is, therefore, to show that the resulting performance is sufficient for practical needs, even under pessimistic assumptions.

For the `SCAN_MODE_LOW_LATENCY` and `SCAN_MODE_BALANCED` settings, assuming every smartphone uses the same parameter values, the success probability is 100% for all values of $T_{a,0}$ supported by Android and iOS, barring one exception for `SCAN_MODE_BALANCED` with $T_{a,0} = 1.285$ secs, where the success probability is reduced to 96.4%. Figure 3 depicts the probabilities for the `SCAN_MODE_LOW_POWER` setting. As can be seen, the success probabilities vary between 99.4% and 39.1%. For values of $T_{a,0} > 512$ ms, the low success probability is caused both by colliding beacons and by the theoretical worst-case latency—that is, the latency when no collision occurs—exceeding 10 secs. For GAEN, when assuming the interval lengths in Table 1, the probability that at least one beacon is received in every scan window is also essentially 100%. The reasons for the high success probability for most parameterizations are a) the relatively low channel utilization, $\beta = \omega / T_a$, of each radio, and b) the large scan window d_s in combination with the random delay. Since multiple beacons are sent within every scan window, the probability that one of them is received without colliding with a beacon from a different device is high. Since the channel is only moderately used in GAEN and the other available settings, ubiquitous tracing devices are not expected to disturb any other BLE applications, either on the same device or on devices in the surrounding environment.

Other limitations. Besides limitations concerning discovery latency, additional limitations reduce the performance and reliability of contact tracing on smartphones.

1) For estimating the proximity between two subjects, every received packet comes with a certain *received signal strength indicator* (RSSI), which

A dedicated wearable device could be designed to optimize every aspect of the tracing procedure and potentially provide a significantly higher tracing reliability.

represents the path loss between a sender and the receiver. Proximity estimation assumes that every RSSI corresponds to a unique distance. However, the RSSI is also influenced by multiple other factors, such as the smartphone model, orientation of the sending and receiving antennas, the environment (for example, metal parts in vicinity), and the wireless channel used. While some of these uncertainties can be mitigated through various techniques, the following problem remains unresolved. Whenever the wireless signal must pass through the human body, the attenuation is much larger than in free space. For example, Alomainy et al.² reports an attenuation of up to 19.2 dB between the chest and the back of a human body. How does this impact distance estimation in contact tracing?

We placed two identical smartphones on a table at a distance of around 10 centimeters. When running the ITO demonstration app for contact tracing,¹³ our experiments showed that the estimated distance was around 50 centimeters when both smartphones were within line of sight. But when a human arm was placed between them, the estimated distance increased to almost 5 meters. Hence, in some situations, a smartphone would classify a contact as being “far away” though it was nearby. For example, consider two people walking side by side, holding hands, with their smartphones in opposite pockets. Though this would represent a relevant contact, the obstructed line of sight would, with a high probability, lead to a larger distance estimation and a misclassification of the contact. Indeed, experimental studies using GAEN suggest that distance estimation can be prone to errors in certain situations. For example, Leith and Farrell¹⁷ report that based on data obtained in an experiment involving multiple smartphone users on a light-rail tram, the German tracing app would trigger no exposure notification at all, while the Italian and Swiss apps would lead to 50% true and 50% false positives. These results suggest that when many metal parts are in the surrounding environment, distance estimation using smartphones can become unreliable.

2) For tracing all relevant contacts using a wireless solution, it is impor-

tant that as many people as possible participate. However, contact tracing on smartphones requires an installed and activated application as well as the appropriate permissions. Hence, a certain part of the population may be physically or mentally unable to handle a smartphone, such as a small child or an elderly person.

Summary. Today, contact-tracing apps rely heavily on GAEN and can reliably detect only those contacts whose durations are at least $\approx$ five minutes long. Brief contacts, such as a short trip on a crowded subway, might remain unnoticed despite a potential transmission risk. Energy consumption is only of concern in the SCAN_MODE_LOW_LATENCY setting.

Overall, our performance evaluation shows that the SCAN_MODE_BALANCED setting guarantees low latencies of around 5 secs if advertising intervals below 1 sec are used. At the same time, all such configurations would provide a success probability of essentially 100% when multiple phones are within range. Also, the energy consumption would remain at a reasonable level. Therefore, contact-tracing applications on smartphones would benefit from operating systems providing access to the SCAN_MODE_BALANCED setting and at least one of the supported advertising intervals below 1 sec. Providing an optimized and dedicated parameterization for contact tracing would further decrease discovery latencies. For example, a shorter scan window could be more frequently scheduled, while mostly preserving energy. This could lead to worst-case latencies below 5 secs if the advertising interval was shortened accordingly. Further, the accuracy of distance estimation could potentially be improved by gathering a larger number of RSSI samples. This could be achieved by allowing a tracing app to temporarily listen continuously once an initial contact has been established.

Both theoretical considerations and practical experiments suggest that at least in certain situations—for example, on a tram or a bus—smartphone-based distance estimation cannot always distinguish between relevant and irrelevant contacts. Overall, smartphone-based contact tracing is a best-effort mechanism without any guarantees.

A dedicated body-worn device might not only lead to more reliable tracing, but could also resolve many privacy issues.

However, a dedicated tracing device could mitigate these limitations.

Contact Tracing Using Wearables

Wearable devices do not need to follow the constraints imposed on the ND procedure by smartphones or the Bluetooth standard. Therefore, they can leverage the entire design space of ND protocols and can easily achieve worst-case latencies of around 2 secs. Moreover, the distance estimation procedure can be facilitated, since wearables are not constrained to Bluetooth. Specifically, Ultra-Wideband (UWB) combined with Time-of-Flight can be used for distance estimation. Such methods also achieved reasonable errors when the human body attenuates the signal in ranging experiments.²⁶ Recent studies indicate that they can also improve the accuracy in contact-tracing scenarios.¹⁰ Furthermore, frequency bands other than Bluetooth's 2.4-GHz range can be used, mitigating susceptibility to in-body attenuation and multipath propagation. Additionally, wearables are active whenever they are worn, so they do not require handling or configuration. Thus, faulty handling or incorrect settings are ruled out.

Wearable applications also consume less power than smartphones, resulting in a drastic improvement in battery life. For example, consider a low-cost nRF52832 radio. We can estimate its energy consumption based on the data provided from the device manufacturer²³ and some additional overheads. We assume a 380-mAh battery, which is often found in smartwatches. To guarantee a discovery latency of 2 secs, using the parameterizations from Kindt et al.,¹⁴ a battery runtime of 235 days can be achieved, which could be further extended by switching the device off automatically when not being worn. Hence, dedicated wearable devices can use the entire design space to optimize every aspect of the contact-tracing procedure and potentially provide a higher tracing reliability.

Further, privacy concerns often counteract the deployment of tracing apps. Wearable options may prove more desirable in this regard because they are dedicated low-content devices that do not store unnecessary private data, such as pictures or personal messages. Wearables can be designed to not require In-

ternet access. However, users might also mistrust a dedicated body-worn device because it might also potentially collect data not needed for contact tracing.

While smartphones were the first platforms that became available after the onset of the COVID-19 pandemic, different wearable solutions have emerged in the meantime. For example, Singapore has released wearable tokens to complement tracing apps.⁵ These devices are Bluetooth-based and are compatible with smartphone apps—that is, apps and tokens can wirelessly detect each other. Their main goal was to provide a solution to users who do not own a mobile phone or who have concerns when using it for contact tracing. This “TraceTogether” token has a reported battery life of four to six months.²⁷ Similar wearables have been studied in academia²⁴ and industry. Some of them²⁹ provide alerts such as vibrating if two subjects come within a specified unsafe proximity, thereby helping to maintain social distancing. While most existing wearables are Bluetooth-based to remain compatible with tracing apps, they typically suffer from similar shortcomings as smartphones—for example, limited accuracy of distance estimation. Recently, wearables that use UWB for distance estimation have also become available, thereby mitigating these limitations. For example, one solution performs device-to-device ranging with a claimed precision of up to 0.1 m.⁴ However, its battery needs to be recharged after 24 hours on an average. Another UWB-based wearable claims a battery life between two to five days.¹⁹

In the future, though, devices must offer a high tracing reliability and a long battery life. To this end, hybrid solutions that use separate radios for distance estimation and for neighbor discovery—for instance, a Bluetooth and an UWB radio¹²—are being studied. Such scenarios seem capable of providing battery lifetimes of months, while still benefiting from UWB’s tracing reliability. Finally, as more contagious variants of SARS-CoV-2 have emerged, airborne infections have become a concern. Here, proximity-based contact tracing becomes less effective, since the proximity between two subjects is no longer the only determining factor. Future wearables, however, can possibly help detect such critical situations. For example, when multiple subjects are in a room

with limited ventilation, contacts need to be recorded even when the proximity is considerably larger than what would be relevant in an outdoor scenario. How such situations could be detected will become relevant for future research, and some infrastructure support in addition to the wearables would be necessary. Similarly, future wearables should also detect and distinguish events that increase transmission risk, such as two subjects touching each other.

Conclusion

We have studied the technical feasibility of using smartphones for contact tracing. Even though their radios support the essential features necessary for contact tracing, some contacts potentially cannot be detected due to the high worst-case latency, which is at around 5 mins for tracing solutions using GAEN. While this could be mitigated if tracing apps could make full use of the existing BLE APIs, contact classification using distance estimation cannot be achieved with sufficient accuracy in all situations. A dedicated wearable that does not rely on Bluetooth could overcome this limitation and increase the effectiveness of electronic contact tracing. **C**

References

- Accessory design guidelines for Apple devices. Apple, Inc. (2019), <https://developer.apple.com/accessories/accessory-design-guidelines.pdf>.
- Alomainy, A., Hao, Y., Owadally, A., Parini, C.G., Neehayev, Y., Constantinou, C.C., and Hall, P.S. Statistical analysis and performance evaluation for on-body radio propagation with microstrip patch antennas. *IEEE Transactions on Antennas and Propagation* 55, 1 (2007), 245–248.
- AN1128: Bluetooth coexistence with Wi-Fi. Silicon Laboratories Inc. (2020), <https://www.silabs.com/documents/public/application-notes/an1128-bluetooth-coexistence-with-wifi.pdf>.
- Anti-COVID-19 social distancing and contact tracing. Tsingol Technology Co. (2021), <https://www.social-distancing-contact-tracing.com/?matchtype=e>.
- Asher, S. TraceTogether: Singapore turns to wearable contact-tracing COVID tech. *BBC News* (July 5, 2020), <https://www.bbc.com/news/technology-53146360>.
- Bluetooth core specification 5.2. Bluetooth SIG (2019), <https://www.bluetooth.com/specifications/specs/core-specification-5-2/>.
- Contact tracing Bluetooth specification. Google (2020), https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf.
- Doung-ngern, P., Saphanchaimat, R., Panjangampattana, A., Janekrongtham, C., Ruampoom, D., Daochaeng, N., and Eungkanit, N., et al. Case-control study of use of personal protective measures and risk for SARS-CoV-2 infection, Thailand. *Emerging Infectious Diseases* 26, 11 (2020), 2607.
- Dutta, P. and Culler, D. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *ACM Conference on Embedded Network Sensor Systems* (2008), 71–84.
- Etzlinger, B., Nußbaumüller, B., Peterseil, P., and Hummel, K.A. Distance estimation for BLE-based contact tracing—A measurement study (2021), arXiv.org 2101.09075.
- Ferretti, L., Wymant, C., Kendall, M., Zhao, L., Nurtay, A., Abeler-Dörner, L., Parker, M., Bonsall, D., and Fraser, C. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science* (2020).
- Istomin, T., Leoni, E., Molteni, D., Murphy, A.L., Picco, G.P., and Griva, M. Janus: Efficient and accurate dual-radio social contact detection. (2021).
- ITO demonstration app. ITO Consortium (2020), <https://www.ito-app.org>.
- Kindt, P.H., Narayanaswamy, S., Saur, M., and Chakraborty, S. Optimizing BLE-like neighbor discovery. In *IEEE Transactions on Mobile Computing* (2020).
- Kindt, P.H. and Chakraborty, S. On optimal neighbor discovery. In *ACM Special Interest Group on Data Communication* (2019).
- Kindt, P.H., Yunge, D., Diemer, R., and Chakraborty, S. Energy modeling for the Bluetooth Low Energy protocol. In *ACM Trans. Embed. Comput. Syst. (TECS)* 19, 2 (March 2020).
- Leith, D.J. and Farrell, S. Measurement-based evaluation of Google/Apple Exposure Notification API for proximity detection in a light-rail tram. *PLoS One* 15, 9 (2020).
- Leith, D.J. and Farrell, S. GAEN due diligence: Verifying the Google/Apple COVID Exposure Notification API. *Network and Distributed System Security Symposium* (2021).
- LoposAlert. Lopos (2021), <https://www.lopos.be/loposalertwearable>.
- Lu, D. COVID Delta variant is “in the air you breathe”: What you need to know about Sydney outbreak strain. *The Guardian* (2021), <https://bit.ly/3xoaFZ0>.
- nRF52832 product specification v1.1. Nordic Semiconductor (2016), https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.1.pdf.
- Oberhagemann, D. Static and dynamic crowd densities at major public events. *Vereinigung zur Förderung des Deutschen Brandschutzes e. V. (vfdB), German Fire Protection Association, Technical-scientific advisory board (TWB) 13*, (2012).
- Online Power Profiler. Nordic Semiconductor (2020), <https://devzone.nordicsemi.com/nordic/power>.
- Open Badges Project. MIT Media Lab (2020), <https://www.media.mit.edu/projects/open-badges>.
- Samsung Galaxy S10+ Teardown. TechInsights, Inc. (2020), <https://www.techinsights.com/blog/samsung-galaxy-s10-teardown>.
- Tian, Q., Kevin, I., Wang, K., and Salcic, Z. Human body shadowing effect on UWB-based ranging system for pedestrian tracking. In *IEEE Transactions on Instrumentation and Measurement* 68, 10 (2018), 4028–4037.
- TraceTogether Token. Government of Singapore (2020), <https://www.tracetogether.gov.sg/common/token/index.html>.
- Vincent, J. Without Apple and Google, the UK’s contact-tracing app is in trouble. *The Verge* (2020), <http://www.theverge.com/2020/5/5/21248288>.
- Workplace safety with wearables. Estimote, Inc. (2021), <https://estimote.com/wearable>.
- Wymant, C., Ferretti, L., Tsallis, D., Charalambides, M., Abeler-Dörner, L., Bonsall, D., Hinch, R., Kendall, M., Milsom, L., Ayres, M., et al. The epidemiological impact of the NHS COVID-19 App. *Nature* (2021), 1–8.

Philipp H. Kindt (philipp.kindt@informatik.tuchemnitz.de) is an assistant professor at Chemnitz University of Technology, Chemnitz, Germany.

Trinad Chakraborty is a professor at Justus-Liebig University Giessen and the German Center of Infection Research, Germany.

Samarjit Chakraborty is a William R. Kenan, Jr. Distinguished Professor at the University of North Carolina at Chapel Hill, NC, USA.

This work is licensed under a Creative Commons Attribution-NoDeriv International 4.0 License. <https://creativecommons.org/licenses/by-nd/4.0/>

Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/smartphone-based-contact-tracing>