A Study of Bluetooth Low Energy Performance for Human Proximity Detection in the Workplace

Alessandro Montanari, Sarfraz Nawaz, Cecilia Mascolo
Computer Laboratory, University of Cambridge
Cambridge, United Kingdom
Email: {name.surname}@cl.cam.ac.uk

Kerstin Sailer
Space Syntax Laboratory, University College London
London, United Kingdom
Email: k.sailer@ucl.ac.uk

***Abstract*—The ability to detect and distinguish interactions in the workplace can shed light over productivity, team work and on employees' use of space. Questionnaires and direct observations have often been used as mechanisms to identify office based interactions, however, these are either very time consuming, yield coarse grained information or do not scale to large numbers of people. Technology has been recently employed to cut costs and improve output, however precise interaction dynamics gathering often requires individuals to wear custom hardware.**

In this paper, we present an extensive evaluation of Bluetooth Low Energy (BLE) as a technology to monitor people proximity in the workplace. We examine the key parameters that affect the accuracy of the detected contacts and their impact on power consumption. We study how this system can be implemented on popular wearable devices (i.e., Android Wear and Tizen) and the resulting limitations. Through a real world deployment in a commercial organisation with 25 participants we evaluate the performances of a BLE-based proximity detection technique.

Our results show the suitability of BLE for workplace interaction detection and give guidance to vendors and Operating System (OS) developers on the impact of the restrictions regarding the use of BLE on commodity wearables.

I. Introduction

Interactions in the workforce play an important role in team performance and productivity [1], [2]. For example, informal inter-team interactions have been shown to be an important trait of successful teams [3]. Work interactions can also influence the design of physical spaces [4] or help in developing an understanding of disease spreading [5], [6]. Researchers have relied on surveys and observations for years to gather data about these phenomena. However, the cost of observations is high as they usually involve long hours of monitoring. They might also not scale to many participants and can therefore only be applied to phenomena restricted temporally. Surveys instead scale better but offer a much coarser grained view as people might forget to report [7].

Usually two contrasting needs are faced when trying to capture such interactions automatically: (1) the need to collect accurate and reliable data and (2) the need to have large deployments to get a clearer picture of human behaviour. Existing solutions usually tend to tackle more one problem or the other. Bluetooth-based systems for example can rely on widespread adoption but are usually power hungry and do not offer fine grained data [8], [9], [10], [6]. On the other hand, systems based on custom built devices can provide fine granularity but require dedicated hardware which hinder adoption [11], [12], [13]. The recent interest in wearable devices [14] has brought us to question if those devices are able to fulfill both needs. In particular we directed our attention towards Bluetooth Low Energy (BLE) which is included in all current wearables. We envision an interaction sensing system that can be easily installed on a wearable device like a smart watch thus extending its functionality to interaction sensing and offering widespread adoption. However, before this can become a reality there are fundamental questions which need answers. Namely: How accurate could BLE proximity detection be? What could be the expected lifetime of this system on an off-the-shelf device? How can it be employed for social interaction sensing and space occupancy monitoring?

In this work, we analyse the potential of BLE to monitor people proximity as first step towards a social interaction sensing system. The objective is to asses its capabilities, first by analysing its parameters and their impact on both accuracy and power consumption, and then, from a practical perspective, with a large user study in a real workplace.

Current hardware, available in modern wearables and smart watches, offers the key functionality for proximity detection: the ability to detect nearby devices and be detected by them by alternating between transmitting and scanning. While the manufactures have recently updated device firmwares and software stacks to support this kind of behaviour there are still several limitations that prevent an accurate study of all the key factors involved in proximity monitoring. In particular, mobile operating systems do not allow the application developer to freely control all the BLE parameters. Thus, we build and use a custom made wearable prototype in which we are in control of all parameters. This allows us to study in detail the interplay of all the BLE parameters and their impact on power consumption. Using our prototype we collected proximity traces in a commercial organisation with 25 participants to extend our analysis to a real world scenario and to not limit ourself only to lab experiments. We were then able, through data post-processing, to investigate the achievable performances if our system were to run on off-the-shelf wearable devices and understand their strengths and weaknesses. This led us to the important conclusion that large scale proximity studies are viable, even at the accuracy level required by domain scientists, with off-the-shelf devices.

To the best of our knowledge, this is the first study of BLE radios on a wearable platform for proximity monitoring which provides useful insights for social interaction sensing applications. This paper also offers guidance to OS developers and manufacturers on the impact of the limitations of their APIs and informs application developers on the flexibility of off-the-shelf wearables. Our contributions are:

- a detailed analysis of BLE parameters that play a central role in proximity detection;
- the first analysis of BLE capabilities and limitations on commercial wearable devices (Android Wear and Tizen);
- an extensive experimental validation with lab experiments and a longitudinal user study with 25 participants in an office environment. We confirm the BLE suitability for accurate proximity monitoring with 97% accuracy at 10 seconds granularity. Ground truth observation for around 19 hours was performed to support our evaluation.
- a discussion on the restrictions imposed by OS developers on the use of BLE for proximity detection.

II. Motivation and Related Work

Accurate human proximity detection has the potential to influence many different disciplines (e.g., social science, architecture and health). Proximity detection often depends on the phenomena under observation: in certain cases only short events with a very small distance between the participants are relevant (e.g., when certain virus spreading is considered), in others, only prolonged interactions that would give individuals a chance to have meaningful conversations are pertinent.

The specific scenario considered for this paper is the one that takes into consideration office based social interactions. In such setting, serendipitous interactions, where, say, a user glances from an office doorway, might be meaningful and indicative of productivity [3], [15]. This is, of course, in addition to prolonged and repeated interactions.

The detection of fairly long lived interaction (of the order of several tens of seconds) has been accomplished by technology reasonably successfully. Bluetooth Classic has often been at the basis of these platforms mainly due to its availability on consumer devices, which makes it extremely suitable to large deployments [8], [6], [16]. However, several works [9], [10] have tried to improve the temporal and spatial granularity of traces collected with Bluetooth Classic. In fact, Bluetooth's main drawbacks reside in the high power consumption and low granularity of the traces. Usually, it is sampled every few minutes [8], [16] to avoid draining the battery too quickly and the range of transmission is around 10m [6].

Other technologies have also been proposed, for instance, RFID [17], Zigbee radio [18], infrared sensors [12] and hybrid approaches with radio and ultrasound sensors [13]. These devices offer better performance (e.g. temporal granularity from 20s to 2s with still reasonable battery consumption) but are not suitable for wide scale and long term adoption because they rely on dedicated hardware which needs to be deployed just for the purpose of the study. In fact, problems have been reported with the usability of these devices [7].

Few works have used specifically BLE to collect data about human behaviour. Townsend et al. tested 4 different smartphones to asses if they could detect each other using BLE [19]. Boonstra et al. deployed an Android and iOS app to 14 participants for a period of one working week to collect data about social contacts [20]. However, the authors offered a limited evaluation of their system by using only two meetings during the study period to validate their methodology and they did not collect participants' locations, which is a valuable piece of information when studying social dynamics. Other works instead have used simple wearable BLE tags, capable of transmitting only, to study mobility patterns of large gatherings [21], [22]. Radhakrishnan et al. have recently analysed BLE characteristics on Android mobile phones for indoor localisation [23]. They implemented a BLE-like duty cycling on top of the Android BLE stack which already performs duty cycling in accordance to the BLE specification. Here we offer instead an analysis of the low level BLE parameters (as defined in the standard) to understand if BLE can be employed to collect fine grained and accurate encounter traces. We then analyse if the approach could be adopted on commercial wearable devices to free proximity-based systems from the need for custom devices, which is usually one of the limiting factors of long term studies. Additionally, the availability of our prototype allowed us to test the impact of these parameters on a large scale deployment.

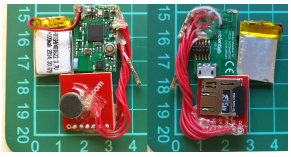
III. Proximity Sensing with BLE

We now discuss the different BLE parameters and we present a detailed analysis of their impact on proximity detection accuracy and energy consumption using our prototype.

A. BLE Modes of Operation

BLE provides two modes of communication: *connection based* and *broadcast based* [24]. The first one requires two devices to establish a connection before exchanging data. This is not suitable for proximity sensing as it can introduce delays and is also restricted to only two devices.

The broadcast mode instead allows a *Broadcaster* to send data to several *Observers* simultaneously without establishing a connection. The Broadcaster periodically sends data (maximum of two 31-byte packets) on three predefined BLE advertisement channels (37, 38 and 39). To adjust the transmission frequency, the BLE specification defines a parameter called Advertising Interval. It is the time between the start of two consecutive advertisements. On the other hand, the Observer listens on an advertising channel for the duration of the Scan Window at every Scan Interval. At each Scan Window, it listens on a different advertisement channel, until all three are used and then repeats. When the current scan channel is aligned with the current advertising channel of another device, the Observer receives the advertisement packet from the Broadcaster and thus detects its presence.



(a) Electronics front and back view. Scale in centimeters.

(b) Prototype enclosed in 3D printed box attached to wristband.

Fig. 1: Prototype wearable platform.

The key for proximity detection lies in the fact that each device should alternate between the two roles: when a device is in Broadcaster role, it transmits an advertisement that can be detected by other devices and when it is in Observer role, it can detect other devices by listening for advertisements.

Although BLE is supported by most wearables, including smartwatches, OSs running on these devices prevent complete access to all BLE parameters. In order to analyse the effect of BLE parameters on sensing accuracy, we developed a prototype that allows us to freely control every parameter.

B. Wearable Platform Prototype

Our prototype is based on the Nordic's nRF51822 BLE SoC that includes a 32bit ARM-M0 CPU and a 2.4GHz radio transceiver. We use a developer board from Mbienlab Inc. that contains the main SoC along with the associated circuitry, a Freescale MMA8452Q 3-Axis Accelerometer, an RGB LED, a push-button switch and a vibrator motor. The entire prototype is powered by a 100mAh 3.7V lithium battery that can be recharged through a micro USB interface. We attach an SD card to log nearby BLE devices. For each device we log the MAC address, the Received Signal Strength (RSS) and the channel on which the packet has been received (37, 38, 39). Figure 1a shows the current prototype. We designed a 3D printed box (3x4x1.5cm) to contain the device and we used velcro straps to wear the device on the wrist (Figure 1b).

We use the S110 SoftDevice BLE stack by Nordic [25] for the Broadcaster role and to run the Observer role concurrently we use an open source library [26]. This library uses the Concurrent Multi-protocol Timeslot API to give access to the radio resource concurrently with the SoftDevice.

C. BLE Parameters

The parameters that characterise a BLE-based proximity sensing system are: *Advertising Time* (time to send a packet on three channels), *Advertising Interval* (time between each packet), *Scan Interval* (time between scans), *Scan Window* (duration of each scan) and *Transmission Power* (transmit power for each packet).

Advertising Time, Advertising Interval, Scan Interval and Scan Window affect how quickly a specific device can detect other devices in the vicinity and is detected by them. Intuitively, it is necessary to advertise more frequently than scanning in order to ensure that at least one advertisement will be captured during a scan and the Scan Window should be long enough to capture at least one advertisement on one channel. These parameters are inter-dependent and dictate the

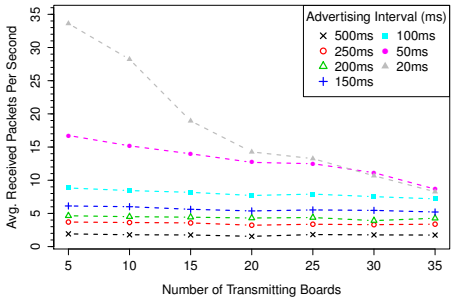


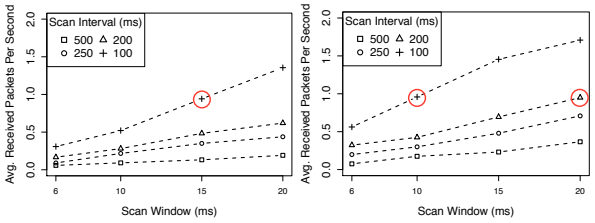
Fig. 2: Average number of received packets changing the Advertising Interval and the number of transmitting devices.

actual packet reception rate achieved by a device. It is not possible to achieve a higher rate and thus higher temporal granularity by simply advertising more frequently because it is also necessary to scan frequently and for longer periods. The Transmission Power is the only parameter available to control the maximum distance at which a contact can be detected: it allows to change the range at which other devices can still correctly receive a packet.

In the following analysis we excluded Advertising Time and Transmission Power because they do not have a significant impact on the accuracy of the system and on the energy consumption. In particular, these parameters cannot be optimised for power consumption because they are often determined by other factors: the amount of data that needs to be transmitted for the Advertising Time (e.g. it may include diagnostic and identification information) and the target range for the Transmission Power (i.e. maximum distance between two individuals that one wants to consider in proximity). We now systematically inspect the remaining parameters.

Advertising Interval: it controls how frequently advertisements are transmitted and thus it affects how quickly a device can be detected by other nearby devices. Assuming that an Observer device is scanning continuously, the time between the reception of two advertisements should, on average, be equal to the Advertising Interval under ideal conditions. However, packet loss due to collisions and environmental factors can affect how frequently advertisement packets are received. We, therefore, devise an experiment to understand the effect of Advertising Interval and the number of transmitting devices on the number of received packets. We configure one device to scan continuously and every 5 minutes we add 5 devices transmitting with a fixed interval, up to a total of 35 devices. The experiment was repeated for 7 different intervals.

Figure 2 shows that the number of transmitting devices affects the number of received packets, especially at high rates (advertising interval of 20ms and 50ms). At these rates, every time another set of devices were added, the average number of received packets dropped resulting in around 10 packets per second when 35 devices were transmitting at the same time. On the other hand, at lower rates (10Hz and downwards) the number of received packets remains constant even when the number of devices increases. We, therefore, choose 100ms as the lower bound advertising interval. This experiment shows



(a) Adv. Interval 150ms. (b) Adv. Interval 100ms.

Fig. 3: Average number of received packets changing the Advertising Interval, the Scan Interval and the Scan Window.

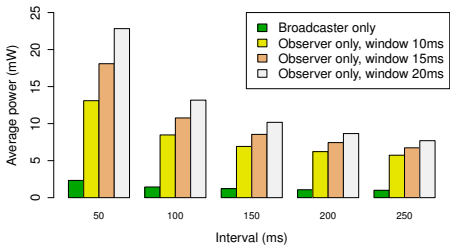


Fig. 4: Average power consumption in isolation by Broadcaster and Observer roles for different combinations of Advertising and Scan Interval (x-axis) and Scan Window (last three colours).

that small advertising intervals do not necessarily lead to high reception rates (considering constant the scan rate and window) especially with high density of devices. Moreover, it can be detrimental for the battery lifetime as packets lost due to collisions represent wasted energy.

Scan Interval and Window: in the previous paragraph, the Observer was scanning continuously, but as we will show in Section III-D, continuous scanning has a significant impact on battery life. Therefore it is necessary to duty cycle the scan operation using the Scan Interval and Window parameters. To study the effect of these parameters on the receive rate we run several experiments where one device transmits at one of the Advertising Intervals tested in the previous paragraph and a second device performs scans with a particular Scan Interval and Window. For each Advertising Interval, we use the values in these sets, {100ms, 200ms, 250ms and 500ms}, {6ms, 10ms, 15ms, and 20ms}, respectively for Scan Interval and Scan Window, combining them in each possible way.

Figure 3 shows the results of these experiments. It demonstrates: (1) the interplay between Broadcaster and Observer parameters, and (2) how Scan Interval and Window can be combined to obtain specific receive rates. These results show that it is not possible to consider the Broadcaster and Observer roles in isolation when designing a proximity-based system. For example, an average receive rate of 1 packet per second can be achieved with three different combinations of the three parameters (red circles in Figure 3). We will show how this can be used to optimise the power consumption of the system in Section III-D. Due to space constraints we do not report results for the other Advertising Intervals analysed in Figure 2, however, they follow the same trend.

D. Parameters' Impact on Power Consumption

We have seen that a specific receive rate can be achieved with different combinations of the parameters. Thus, it is important to consider the effect of each parameter on power consumption to select the combination that provides the desired receive rate and the least power consumption.

To study the impact on power consumption we configured our device in Broadcaster only mode first and then in Observer only mode, with different combinations of the three parameters and we measured the average power.

Figure 4 shows the results of these experiments. Even at the same rates, the Observer role has a greater impact on power consumption as compared to the Broadcaster role. Therefore, to achieve a certain desired receive rate, it is better to scan with a low frequency and for short periods and transmit more frequently in order to have a lower impact on the power consumption. However, as explained in the previous section, high transmission rates can lead to collisions if the density of devices is high: this must be kept in consideration when designing a proximity-based system for crowded environments.

As described above, Advertising Time and Transmission Power cannot be optimised for power consumption and their impact on battery lifetime is minimal. In fact, advertising 3 or 31 bytes every 100ms increases the power consumption of 0.18mW. Instead, advertising at the highest (4dBm) and at the lowest (-20dBm) power level increases the consumption of about 0.37mW (14 bytes every 100ms). This shows that the impact of these parameters on battery lifetime is much less significant than the other parameters.

IV. Proximity Sensing on Commercial Devices

After the study of BLE parameters using our prototype, we now analyse to what extent the same parameters can be exploited on commercial wearables. These devices are equipped with a BLE chip used for communication with the user's phone and being always co-located with (worn by) the user they offer a great advantage for proximity sensing. The platforms we used for our analysis are Android Wear 5.0 [27] and Tizen Wearable 2.3.1 [28]. While it was already possible to implement the Observer role, the Broadcaster role has been enabled in recent releases (e.g., September 2015 for Tizen).

The actual devices we used for our experiments are a Samsung Gear S2 [29] for Tizen and a Samsung Gear Live [30] for Android Wear. We developed an application for each device that allowed us to change the parameters and start/stop the advertising and scan operations. Both devices are able to transmit and scan in background and when they are connected to a phone, so they can still receive notifications from the paired phone as during normal operation. In all the experiments, both watches were connected to an Android phone.

Similarly to what we did with our custom device in Section III-C we now systematically inspect the parameters individually to understand capabilities and limitations of BLE on

commercial devices. For completeness, we analysed the Advertising Time and Transmission Power parameters because, even if they cannot be optimised for power consumption, their control is important (e.g, diagnostic and transmission range).

Advertising Time: The only way to control this parameter is to vary the number of bytes included in the advertising packets. Both operating systems expose APIs to configure the content of the BLE packet (i.e. device name, service and manufacturer data, etc.). The only difference is that Tizen offers the possibility to set the appearance of the device and solicitation UUIDs while Android Wear does not.

Advertising Interval: This parameter can be adjusted on both platforms in similar ways. Three values are allowed: (1) *Low Latency*, (2) *Balanced* and (3) *Low Power* (called *Low Energy* in Tizen). However, the resulting behaviour is different for the two operating systems. In Tizen the three values correspond to an Advertising Interval respectively of 150ms, 500ms and 1s. By contrast, the Android Wear watch, regardless of the value set, starts advertising with a 30ms interval for about 150/180 seconds and then switches to an interval of 1280ms. This shows that for the Android Wear platform the Advertising Interval in practice cannot be controlled and the only usable value is 1280ms.

Scan Interval and Window: For what concerns the Observer role, the Tizen OS does not allow to set any parameter, it only allows the developer to start and stop the scan operation. Android Wear on the other hand does not permit to configure the Scan Interval and Window individually but it allows to choose among three global values for the scan operation: (1) *Low Latency*, (2) *Balanced* and (3) *Low Power*. To test the actual achievable receive rate with these three values we configured one of our custom devices in Broadcaster only mode with an Advertising Interval of 100ms and we let the Android Wear watch scan. If the scan is configured in *Low Latency* mode the watch scans continuously, achieving in this case an average receive rate of around 9Hz. In *Balanced* mode instead the average receive rate is halved (around 4Hz) and in *Low Power* is one-tenth (around 1Hz).

Analysing the data collected when in Balanced mode we notice that the watch performs a scan around 12 times in a minute. Therefore we assume the Scan Interval is roughly 5s. We also notice that the Scan Window is around 2s because we observe received packets only during this period. In Low Power mode we found a similar pattern (around 12 scans in a minute so Scan Interval of about 5s) but with a smaller Scan Window of 500ms. In Low Latency instead the packets are uniformly distributed across the scan period.

As mentioned earlier, in Tizen it is not possible to select any setting for the scan operation. Performing the same experiment we did with the Android Watch (with a device transmitting at 10Hz) we discovered that for the Tizen watch the average receive rate is around 1Hz and the Scan Interval and Window are equal to the ones adopted by Android Wear in Low Power mode (scan for 500ms every 5 seconds).

Transmission Power: Tizen does not provide any API to control the transmit power therefore it is not possible to control the transmission range. The transmit power level included in the advertisement packets is 12dBm and the average RSS at 1 meter is around -78dBm.

By contrast, Android Wear offers an API that permits to choose between four different values: *High*, *Medium*, *Low* and *Ultra Low*. However, regardless of the value set, the watch we tested uses the same power level and includes the value -21dBm in the advertisement packets. This is also confirmed by the fact that even setting a different value, there is no substantial difference in the RSS we measured at 1 meter and its average is always around -66dBm.

To summarize, we have observed that both systems do not give complete freedom on the setting of the parameters, they rather allow to choose between predefined values. Android Wear offers APIs to control all the BLE parameters but only two of them work on the watch we tested (Advertising Time and Scan related parameters). On the other hand, Tizen offers APIs only to modify Advertising Time and Advertising Interval. Table I summarises the parameters that can be controlled on the two platforms.

TABLE I: Summary of control possibilities on Android Wear and Tizen. The asterisk character ('*') indicates that the APIs offer the possibility to set different values but they have no effect on the watch we tested.

Parameter	Gear Live (Android Wear)	Gear S2 (Tizen)
Advertising Time	Yes	Yes
Advertising Interval	No*	Yes
Scan Interval and Window	Yes	No
Transmission Power	No*	No

A. Power Consumption

In this section we analyse the impact of the different adjustable parameters on the watches' power consumption. All the measurements have been taken with the watch connected to an Android phone and with the screen off.

1) Android Wear: The Gear Live has a 3.7V, 300mAh battery and the power consumption when idle with the screen off is 10.29 mW. The only parameters that can be controlled are the Advertising Time and the combination of Scan Interval and Window.

As expected, the Advertising Time has a limited impact on power consumption. For example, the power difference when transmitting 6 or 31 bytes is around 0.19mW which gives a difference in lifetime of only 1.6 hours for a 300mAh battery.

Regarding the Scan Interval and Window, we tested the three possible global values, *Low Latency*, *Balanced* and *Low Power*. We observe a more substantial effect on power consumption. Table II shows that when the *Low Latency* mode is selected, which corresponds to the watch scanning continuously, the power consumption is very high and in this case the expected battery life would be only around 5 hours. This would make impossible the deployment of a proximity-based application in a workplace environment because it could not cover the standard 8 hours of work. Similarly, the *Balanced* mode would result in an expected battery lifetime of slightly more than 8 hours. The only mode that would enable this kind

TABLE II: Power consumption of Scan and Advertising modes for the two smartwatches. The packet used for the Tizen experiments is 14 bytes long.

Mode	Avgerage Power Android Wear (mW)	Avgerage Power Tizen (mW)
Scan Low Latency	227.25	-
Scan Balanced	124.32	-
Scan Low Power	82.18	-
Advertising Low Latency	-	8.21
Advertising Balanced	-	6.61
Advertising Low Power	-	6.19

of deployment is the *Low Power*. In this mode, in fact the watch has an estimated battery life of more than 13 hours but during a typical working day the proximity detection system would remain active for 8 hours only. This means that in the remaining part of the day the power consumption will be lower because the watch is not scanning and advertising periodically and this should guarantee enough energy for normal usage.

2) Tizen Wearable: The Samsung Gear S2 (3.8V, 250mAh battery) allows to control Advertising Time and Interval but not the scan parameters and the Transmit Power. This watch consumes 5.81mW when idle and with the screen turned off.

For the Advertising Time, we observe in this case a greater impact on power consumption. Indeed advertising 31 bytes (in Low Latency mode) will deteriorate the battery life of around 23 hours when compared to advertising just 6 bytes.

As opposed to Android Wear, Tizen OS permits the developer to select one of three different Advertising Intervals. In this case, as it is possible to see in Table II, the average power consumed, even at a relatively high transmission rate (i.e. *Low Latency*), is limited and it is considerably lower than during the scan operation, which for this watch is 55.56mW.

V. Workplace Deployment

We now evaluate the overall performance for proximity sensing with a deployment in a workplace environment.

A. Experimental Method and Testbed

A proximity sensing system is characterised by many parameters that affect the performances. As shown in the previous sections, often these parameters are inter-dependent. Therefore, to evaluate different parameters combinations in a real environment multiple deployments would be necessary.

Our approach instead was to deploy our prototype, giving us greater flexibility, and then test different combinations of the parameters by post-processing the collected data. In particular, we are interested in knowing how a proximity detection system would work on wearable off-the-shelf devices.

Our testbed consists of an architecture company (Spacelab Ltd.) which employs more than 35 people. The building includes two floors with a staircase opening in the middle. The employees do not have assigned desks and the working style is very dynamic. We recruited 25 participants for a period of four weeks. Before beginning with the deployment our work has been approved by the University of Cambridge ethics committee[1]. All the participants accepted to take part in the study according to their will after being informed about the purposes of the study. All collected data is anonymous and has no reference to the single participants.

Wearable Devices: Each participant has been asked to wear, on the wrist and only when inside the office, our wearable prototype. We provided a charging station where all devices were recharged during the night and where some spare devices were stored as replacement in case of failures. Every night an Android Phone collected all the data from the wearables and uploaded it to our servers.

The devices were programmed with an Advertising Interval and Scan Interval of 100ms and a Scan Window of 20ms. The transmit power has been set at -8dBm. This configuration allowed us to achieve an average receive rate of 2.15Hz and a range of around 6 meters. This configuration was selected because it represented the best compromise between battery lifetime (around 20 hours to cover a working day) and granularity of the collected data.

Static BLE beacons: Seventeen BLE static beacons were deployed in the building with the purpose of giving coarse grained (at the desk level) location information about the participants. One beacon was placed on each desk or, if the desk was too big, two beacons were used. The beacons were configured to a beacon rate of 5Hz and -12dBm transmit power. We highlight that the static beacons have been used in this work for evaluation purposes, however our wearable prototype could be used for proximity detection even when those beacons are not available (e.g. outdoor).

Ground Truth: In order to collect ground truth data, one researcher performed observations for three days during the study. During each observation, the researcher followed a person and annotated all the social interactions the person had. Since we are interested in detecting fine grained proximity between people, the researcher recorded only those interactions that happened in close proximity, i.e. up to a distance of 3 meters between people, and involved actual communication for few seconds or more. For each interaction event the researcher recorded the start time, the end time, the location inside the office and the unique ID of the people involved. In total we observed 18 different participants who have been chosen in order to represent the teams in the company. This resulted in 19 hours of observations during which we captured 401 interactions. On average an interaction is 1 minute and 13 seconds long and 70% of the interactions are shorter than 1 minute while only 5% are longer than 5 minutes. The largest interaction captured involved a group of 5 people.

B. Proximity Detection Technique

For proximity detection we are interested in distinguishing when two or more people are in close proximity to each other. This means that we are not interested in measuring the actual distance between them but only if they are close to each other as during a normal conversation.

To achieve the goal, we adopt a supervised machine learning approach where we feed a binary classifier with a set of examples labelled as “proximity” or “non-proximity”. To label the

[1]Our agreement with Spacelab does not include the publication of the collected dataset.

positive examples ("proximity" label) we use the time intervals when interactions have been reported during the ground truth observations. We recall that the researcher was instructed to record only interactions that involved close proximity between the participants, assuming that social interaction events are examples of close proximity. Instead, the negative examples ("non-proximity" label) have been labelled using the static beacons. From the logged data we compute the beacon with the strongest signal strength (i.e. the closest one) at each point in time and we co-locate the participant with that beacon. For each pair of participants it will happen that for some time periods they will be co-located with the same beacon (e.g. when they are sitting at the same desk), and for other periods they will be co-located with different beacons (e.g. when they are at different locations in the building). We select those periods where the two participants are at different locations and we use them as "non-proximity" examples.

For each pair of individuals that have been observed we extract from their devices the stream of data relative to the other device. We then merge the two streams into one in order to have more data for the classification, taking advantage of the fact that the data collection is symmetric for both devices. This stream of data is then split into non-overlapping windows of different sizes (1, 5 and 10 seconds). For each window we compute the following features: median RSSI, min RSSI and max RSSI which, after several tests, are resulted to be the ones that perform better. When two people are very far from each other (e.g. in different floors of the building) the two devices will not receive any packet and this will result in missing values in the dataset. For us those missing values are meaningful because they indicate that the two devices are not in range at all and we do not want a machine learning algorithm to ignore them. Therefore we replace them with the value -110 which represents a very low RSS and it is below the minimum detectable power by our device (-105dBm).

The resulting dataset presents a significant class imbalance because for each pair of people we label the positive examples from the interaction events, which represents a limited period of the day (e.g. 30 minutes), but we derive the negative examples from the moments when they are at different locations in the building and these could cover several hours of the same day. Thus we over-sample the minority class generating synthetic examples using the SMOTE technique [31] in order to balance the two classes.

For the classification we adopted Decision Trees (C4.5) and stratified 10-fold cross-validation. We tested other algorithms but the results were only slightly different therefore we present here only the results for the Decision Trees. The algorithms have been taken from Weka version 3.7.13 [32].

C. Results

Before analysing the classification results we report the metadata about the study. Given that each device records the data on a new file every day, we use the number of correctly recorded files as a measure of robustness of our prototype. In total we expected to collect 500 files (20days * 25devices = 500files) but we actually collected 446 files (10.8% of failures). These failures are due to different causes: device failure, device out of battery, device forgotten at home or lost (2 devices were lost due to problems with the plastic box). For 30% of the days (150) the devices worked properly but they were not in use, they were charging at the charging station. This could be due to the fact that the working style is very dynamic and people do not have a fixed schedule but are often outside to visit construction sites.

TABLE III: Parameters configurations and expected battery life for the two considered wearable platforms and for our custom device when the Broadcaster and Observer role are enabled at the same time.

Configuration Name	Advertising Interval (ms)	Scan Interval (ms)	Scan Window (ms)	Average Receive Rate (Hz)	Expected Battery Life (Hours)
Custom Device	100	100	20	2.15	19.33
Android Wear Low Power	1280	5000	500	0.08	13.74
Tizen Low Latency	150	5000	500	0.62	14.95
Tizen Balanced	500	5000	500	0.19	16.36
Tizen Low Power	1000	5000	500	0.1	17.02

TABLE IV: True positive rate (TP), false positive rate (FP) and area under ROC curve (AUC) for different windows when the raw data is down-sampled uniformly. The configuration names refer to Table III. The data from the custom device has not been post-processed.

Window Size (s)	Custom Device		Tizen Low Latency		Tizen Balanced		Tizen Low Power		Android Wear Low Power	
	TP/FP	AUC	TP/FP	AUC	TP/FP	AUC	TP/FP	AUC	TP/FP	AUC
1	0.79 0.21	0.85	0.68 0.32	0.71	0.59 0.41	0.60	0.55 0.45	0.55	0.54 0.46	0.54
5	0.94 0.06	0.97	0.90 0.10	0.94	0.81 0.19	0.85	0.73 0.27	0.77	0.71 0.29	0.73
10	0.97 0.03	0.98	0.96 0.04	0.98	0.92 0.08	0.95	0.84 0.16	0.90	0.81 0.19	0.86

We now present the classification results we are able to achieve using the raw data collected with our device and we show the results that would have been achieved by the two off the shelf wearable platforms. We do this through down-sampling of the raw data to match the wearable devices' rates.

Table III summarises the different configurations identified for the two platforms and the configuration we used on our device. It also reports the expected battery life achievable by each device when the Broadcaster and Observer role are enabled simultaneously. We decided not to include the configurations *Android Wear Low Latency* and *Android Wear Balanced* because, as observed in Section IV-A1, they present an excessive power consumption for our target environment.

1) Custom Device: With our prototype we are able to achieve an accuracy of 97% with a resolution of 10 seconds (Table IV). At this resolution we detected 21284 proximity contacts over the entire duration of the study. To make sure our devices are able to capture social dynamics accurately, we compare the devices' data with participants observation data and with other datasets publicly available collected with

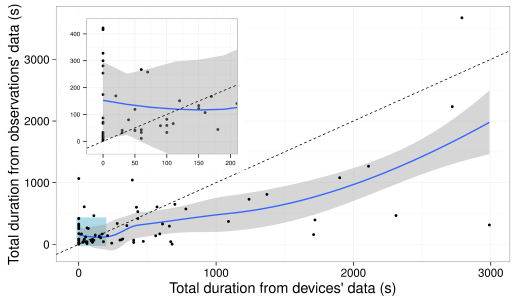


Fig. 5: Relationship between total contacts duration recorded by the observations and by our devices for each pair of participants observed. The black dashed line represents the equation y = x. The blue solid line represents the LOESS polynomial fitting with 95% confidence intervals. The inset plot is a magnification of the area highlighted by the light-blue semi-transparent square.

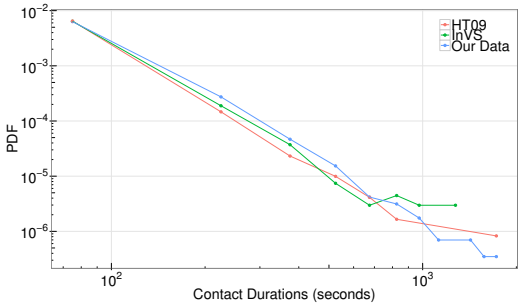


Fig. 6: Probability Distribution Function of contact durations from our study and from two other studies that employed RFID tags: a conference (HT09) [37] and a workplace (InVS) [33].

a similar radio technology (i.e., RFID). This is particularly important given that we employed SMOTE to generate synthetic examples to balance the two classes (Section V-B).

Looking at the data, we observe a very broad distribution: most of the contacts are brief but also few long-lasting contacts have been recorded. Similar results have been reported in different contexts such as workplaces [33], conferences [34], [17], [35] and high schools [36]. For each observation period we computed the total duration of contacts between each pair of participants that have been observed. Figure 5 reports on the x-axis the total contact duration as recorded by our devices, while on the y-axis as recorded during the observations (each point represents a pair of participants). In total during the observations we recorded contacts among 84 distinct pairs of people and, due to failures, for 25 of these the aggregated contact duration recorded by our devices was zero. Given that the two data sources capture different aspects of interactions (i.e., conversations vs proximity) and have different types of biases it is impossible for the points to lie exactly on the ideal line (black dashed line). However, we observe that most of the points are below the ideal line as expected by a system that collects proximity traces. In fact, our device captures longer durations than the actual communication because it considers as contact a situation when people are close to each other but do not interact verbally. Moreover, we observe that the co-presence can be used as proxy for communication as shown by the trend highlighted by the blue line. This is also shown by an highly significant positive correlation (R^2 = 0.67, p-value < 0.01 and R^2 = 0.54, p-value < 0.01 when computed on the logarithmic contact durations).

Additionally, we used open datasets available online from previous works and we plot (Figure 6) the Probability Distribution Function of two datasets (HT09 and InVS) in comparison with our dataset (Our Data). The two datasets have been collected with the Sociopatterns tags [17] and they are discussed by Isella et al. (HT09, conference) [37] and by Gnois et al. (InVS, workspace) [33]. As it is possible to see from Figure 6, the data collected during our study has a very similar distribution to the data collected in other settings.

These results firstly validate the data we collected with our devices by showing the similarity with the observations' data and the fact that the patterns we observe are expected for data collected with a proximity sensing system; as already reported in previous works that adopted a similar technology. Secondly, they show that even if our devices do not capture the specific communication events, the proximity information they provide represents a good approximation for them.

2) Commercial Devices: We then post-process the data collected with our custom device in a simple uniform way by removing data points uniformly to match the watches' data rates. Table IV reports the classification accuracy. Firstly, we notice that increasing the window size the accuracy increases. This is because the RSS data has high-frequency noise which is increasingly attenuated by computing the features over a larger number of data points. However, this impacts the granularity of the detected proximity events. For example, using a 10-second window it is impossible to say if the proximity event was 5 or 8 seconds long. The second important aspect to observe is that using the configuration *Tizen Low Latency* it is possible to achieve a high detection accuracy (similar to the one we obtained with our custom device) especially for large window sizes (from 10s onwards).

For what concerns Android Wear instead, even if it offers the same Tizen's duty cycle for the scan operation, its large Advertising Interval does not allow to obtain a receive rate that is high enough. This results in the need to use larger window sizes to improve the accuracy.

Now we consider a different way to post-process the data which is more similar to how the watches perform the scan operation. Watches in fact scan every 5 seconds and just for 500ms, therefore in this case we emulate this behaviour by keeping the first 500ms of data every 5 seconds and then making sure that the average receive rate matches the one achievable with each configuration. In this case we could not consider the configuration *Tizen Low Latency*. When choosing the deployment parameters for our device we had to find the best compromise between device's lifetime and data collection rate in order to ensure a realistic scenario were people would wear the device for at least 8 hours a day. This resulted in a rate

TABLE V: True positive rate (TP), false positive rate (FP) and area under ROC curve (AUC) for different window sizes when the raw data is post-processed to emulate the watches's scan behaviour (scan for 500ms every 5 seconds). The configuration names refer to Table III. The data from the custom device has not been post-processed.

Window Size (s)	Custom Device		Tizen Balanced		Tizen Low Power		Android Wear Low Power	
	TP/FP	AUC	TP/FP	AUC	TP/FP	AUC	TP/FP	AUC
1	0.79 0.21	0.85	0.55 0.44	0.57	0.53 0.47	0.53	0.53 0.47	0.53
5	0.94 0.06	0.97	0.79 0.21	0.83	0.66 0.33	0.68	0.66 0.33	0.68
10	0.97 0.03	0.98	0.90 0.10	0.93	0.77 0.23	0.81	0.77 0.23	0.81

(2.15Hz) that prevented us the ability to post-process the data to match the *Tizen Low Latency* configuration. Tizen version 2.3.1 was released after we did the deployment, therefore we could not predict this possibility.

From Table V it is possible to notice that, as the window size increases the accuracy increases and gets closer to the one we achieved with our custom device. This is because with larger windows the fact that the watch scans with a low duty cycle is mitigated. Indeed, with a larger window, data from different scans is considered and this increases the accuracy but reduces also the granularity. Instead, with shorter windows the accuracy declines drastically because there will be more windows with no data at all (between two consecutive scans for example) which will be misclassified.

VI. Discussion

The study has confirmed that BLE is an appropriate technology for the automatic detection of individuals proximity in the workplace. Our device is able to reach a considerable accuracy (97%) with a relatively small time window of 10 seconds. This represents an improvement when compared to the 20 seconds granularity of dedicated RFID devices [17] or to the few minutes of Bluetooth Classic [8], [16].

Our work also explored how current commercial wrist-worn devices would perform in such deployments. We highlight that the concurrent use of Broadcaster and Observer roles on these devices does not affect their usability, except for an increased battery consumption. In general, the OS vendors tend to be conservative in terms of energy consumption in order to provide a satisfiable experience to the end-users. For this reason they limit the configurable options to the ones that impact the least the battery consumption. This is the case for Tizen OS which permits to set different Advertising Intervals but offer no options for the scan operation. Android Wear instead allows to scan quite aggressively in *Low Latency* and *Balanced* modes. However, we showed in Section IV-A that these settings result in an excessive power consumption which would make an office deployment unfeasible.

The Samsung Gear S2 could well support proximity-based applications because it allows to detect proximity with an accuracy of around 90% with a 5-second window (when the data is down-sampled uniformly). By contrast, Android Wear would require an increased transmit rate or a bigger scan window. In fact, the 1280ms Advertising Interval is too big to achieve a useful receive rate. Our results show that with an Advertising Interval of around 100/200ms also an Android Wear watch would be able to capture short-lived proximity events as Tizen. However, at the moment Android Wear suffers an high power consumption which should also be addressed in order to make longitudinal studies with this platform feasible.

Comparing Table IV and V it is noticeable that even using the relatively low receive rates achievable by the watches it is possible to slightly improve the granularity and accuracy by having more uniform data. In fact, when the raw data has been post-processed to match the watches' rates down-sampling it uniformly (Table IV), the accuracy is higher even for smaller windows. This suggests that OS vendors could improve the proximity detection on wearables (although only of a few percentage points) by allowing their devices to scan on a more regular basis. At the moment in fact, the few seconds of gap with no data, due to the scan being performed every 5 seconds, is detrimental for detection accuracy. The OS should allow to scan more frequently but for less time in order to obtain more uniform data while keeping the current rate and a similar power consumption. We also note that giving more control to the developer on the Advertising Interval setting would not be counterproductive in terms of end-user experience. In fact, this parameter is the one that affects the least the power consumption of the wearable device.

The Transmission Power instead is a parameter that is non-adjustable on both tested watches (at least Android Wear offers the API so we can speculate that it may be enabled in future releases). This could be a limiting factor for proximity-based applications. A power that is too high would require filtering on the RSS values to remove data that correspond to devices that are too far away or could create collision problems in crowded environments, while a power that is too low could result in missing contacts. Again, since this parameter does not affect the battery life dramatically the possibility to adjust it would not worsen the user experience.

VII. Conclusion

This work highlights the feasibility of workplace interaction studies using commercial BLE wrist-worn devices: it has explored the parameter space through a prototype platform on which BLE could be used without constraints. This allowed us to infer the proximity detection power of COTS devices and a discussion of the limitations. We hope this study can offer guidance to developers and hardware producers regarding their APIs and specifications. In the future, we plan to integrate our findings into a framework which offers prompt user feedback.

Acknowledgments

Alessandro Montanari acknowledges the support of the EPSRC and Qualcomm Inc. through a Research Studentship in Computing. The authors thank Spacelab employees for their support and participation in the user study and Efstathia Kostopoulou for the participant observations.

REFERENCES

[1] D. Krackhardt and J. Hanson, "Informal networks: The company behind the chart," *Harvard Business Review*, vol. 71, no. 4, pp. 104–111, jul/aug 1993.

[2] P. Jeffrey and A. McGrath, "Sharing serendipity in the workplace," in *Proceedings of the Third International Conference on Collaborative Virtual Environments*, ser. CVE '00. New York, NY, USA: ACM, 2000, pp. 173–179. [Online]. Available: http://doi.acm.org/10.1145/351006.351037

[3] A. Pentland, "The new science of building great teams," *Harvard Business Review*, vol. 90, no. 4, pp. 60–69, 2012.

[4] C. Brown, C. Efstratiou, I. Leontiadis, D. Quercia, C. Mascolo, J. Scott, and P. Key, "The architecture of innovation: Tracking face-to-face interactions with ubicomp technologies," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 811–822. [Online]. Available: http://doi.acm.org/10.1145/2632048.2632056

[5] E. Yoneki, "Fluphone study: Virtual disease spread using haggle," in *Proceedings of the 6th ACM workshop on Challenged networks*. ACM, 2011, pp. 65–66.

[6] E. Yoneki and J. Crowcroft, "Epimap: Towards quantifying contact networks for understanding epidemiology in developing countries," *Ad Hoc Networks*, vol. 13, pp. 83–93, 2014.

[7] K. Sailer, R. Pachilova, and C. Brown, "Human versus machine-testing validity and insights of manual and automated data gathering methods in complex buildings," in *Proceedings of the 9th International Space Syntax Symposium*. Sejong University Press, 2013.

[8] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland, "Social fmri: Investigating and shaping social mechanisms in the real world," *Pervasive and Mobile Computing*, vol. 7, no. 6, pp. 643–659, 2011.

[9] J. M. Cabero, V. Molina, I. Urteaga, F. Liberal, and J. L. Martín, "Acquisition of human traces with Bluetooth technology: Challenges and proposals," *Ad Hoc Networks*, vol. 12, 2014.

[10] S. Liu, Y. Jiang, and A. Striegel, "Face-to-Face Proximity Estimation Using Bluetooth On Smartphones," *IEEE Transactions on Mobile Computing*, vol. 13, no. 4, 2014.

[11] C. Brown, C. Efstratiou, I. Leontiadis, D. Quercia, and C. Mascolo, "Tracking serendipitous interactions: How individual cultures shape the office," in *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW '14, New York, NY, USA, 2014, pp. 1072–1081. [Online]. Available: http://doi.acm.org/10.1145/2531602.2531641

[12] T. Choudhury and A. Pentland, "The sociometer: A wearable device for understanding human networks," in *Proceedings of the CSCW'02 Workshop: Ad hoc Communications and Collaboration in Ubiquitous Computing Environments*, 2002.

[13] W. Huang, Y.-S. Kuo, P. Pannuto, and P. Dutta, "Opo: a wearable sensor for capturing high-fidelity face-to-face interactions," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 2014.

[14] R. Rawassizadeh, B. A. Price, and M. Petre, "Wearables: Has the age of smartwatches finally arrived?" *Commun. ACM*, vol. 58, no. 1, pp. 45–47, Dec. 2014. [Online]. Available: http://doi.acm.org/10.1145/2629633

[15] R. S. Burt, "Structural holes and good ideas," *American journal of sociology*, vol. 110, no. 2, pp. 349–399, 2004.

[16] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 244–251. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080142

[17] C. Cattuto, W. Van den Broeck, A. Barrat, V. Colizza, J.-F. Pinton, and A. Vespignani, "Dynamics of person-to-person interactions from distributed rfid sensor networks," *PLoS ONE*, vol. 5, no. 7, p. e11596, 07 2010. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0011596

[18] C. Martella, A. Miraglia, M. Cattani, and M. van Steen, "Leveraging proximity sensing to mine the behavior of museum visitors," in *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2016, pp. 1–9.

[19] S. Townsend, M. E. Larsen, T. W. Boonstra, and H. Christensen, "Using bluetooth low energy in smartphones to map social networks," *arXiv preprint arXiv:1508.03938*, 2015.

[20] T. W. Boonstra, M. E. Larsen, and H. Christensen, "Mapping dynamic social networks in real life using participants' own smartphones," *Heliyon*, vol. 1, no. 3, p. e00037, 2015.

[21] S. Jamil, A. Basalamah, and A. Lbath, "Crowdsensing traces using bluetooth low energy (ble) proximity tags," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. ACM, 2014, pp. 71–74.

[22] S. Jamil, A. Basalamah, A. Lbath, and M. Youssef, "Hybrid participatory sensing for analyzing group dynamics in the largest annual religious gathering," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 547–558.

[23] M. Radhakrishnan, A. Misra, R. K. Balan, and Y. Lee, "Smartphones and ble services: Empirical insights," in *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*. IEEE, 2015, pp. 226–234.

[24] S. Bluetooth, "Bluetooth core specification version 4.0," *Specification of the Bluetooth System*, 2010.

[25] N. Semiconductor, "Softdevice s110 specification v2.0," 2014, last Accessed 28/11/2015. [Online]. Available: https://www.nordicsemi.com/eng/Products/S110-SoftDevice-v7.0

[26] Nordic Semiconductor, "Concurrent MultiProtocol Library," last Accessed 28/11/2015. [Online]. Available: http://nordicsemiconductor.github.io/nRF51-multi-role-conn-observer-advertiser

[27] Google, "Android v5.0 BLE," last Accessed 28/11/2015. [Online]. Available: https://developer.android.com/about/versions/android-5.0.html#BluetoothBroadcasting

[28] Tizen Project, Linux Foundation, "Tizen 2.3.1 Release Notes," https://developer.tizen.org/development/tools/download/release-notes/2.3.1-sep-3-2015, last Accessed 28/11/2015.

[29] Samsung, "Gear S2," last Accessed 09/12/2015. [Online]. Available: http://www.samsung.com/global/galaxy/gear-s2/

[30] Samsung, "Gear Live," last Accessed 09/12/2015. [Online]. Available: http://www.samsung.com/global/microsite/gear/gearlive_design.html

[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, pp. 321–357, 2002.

[32] Machine Learning Group at the University of Waikato, "Weka 3: Data Mining Software in Java," http://www.cs.waikato.ac.nz/ml/weka/, 1997, last Accessed 09/12/2015.

[33] M. Génois, C. L. Vestergaard, J. Fournet, A. Panisson, I. Bonmarin, and A. Barrat, "Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers," *Network Science*, vol. 3, no. 03, pp. 326–347, 2015.

[34] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM, 2005, pp. 244–251.

[35] A. Barrat, C. Cattuto, V. Colizza, J.-F. Pinton, W. V. d. Broeck, and A. Vespignani, "High resolution dynamical mapping of social interactions with active rfid," *arXiv preprint arXiv:0811.4170*, 2008.

[36] R. Mastrandrea, J. Fournet, and A. Barrat, "Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys," *PloS one*, vol. 10, no. 9, p. e0136497, 2015.

[37] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck, "What's in a crowd? analysis of face-to-face behavioral networks," *Journal of theoretical biology*, vol. 271, no. 1, pp. 166–180, 2011.