Step-by-step Detection of Personally Collocated Mobile Devices

Animesh Srivastava
Duke University
animeshs@cs.duke.edu

Jeremy Gummeson
HP Labs Palo Alto
jeremy.gummeson@hp.com

Mary Baker
HP Labs Palo Alto
mary.baker@hp.com

Kyu-Han Kim
HP Labs Palo Alto
kyu-han.kim@hp.com

ABSTRACT

Many people now carry multiple mobile devices on a daily basis. Wearables, smartphones, tablets, and laptops all have their different advantages, but collectively they can increase a user's device management burden. Management problems include leaving a device behind accidentally, receiving notifications on the wrong device, and failing to secure all of the devices as needed. Reducing this burden requires detecting which of a user's devices are "personally collocated" – those devices he currently wears, carries, or has under his immediate physical control. We present a lightweight method to detect personal collocation by comparing accelerometer-based footstep signatures across the devices over time. Through several experiments, we demonstrate that the technique is lower latency and lower power than state-of-the-art RSSI-based collocation techniques. We describe other advantages and limitations of our method and also provide several examples of higher-layer applications and services that can make use of personal collocation information.

1. INTRODUCTION

Many current smartwatches alert their owner if he leaves his phone behind. When this popular feature works well, it helps users avoid leaving phones unattended or losing them altogether. These alerts could usefully apply to other devices as well, including tablets, phablets, and laptops. We call the set of devices currently worn, carried, or physically controlled by a user his *personally collocated* devices, and since each of these form factors has its own set of advantages, many people now use more than one of them on a daily basis [4]. Carrying more devices increases the management burden on the user, making it easier for him to lose them or leave them unattended, but a useful alert could inform the user of any change in his set of personally collocated devices.

Unfortunately, the predominant method for providing this alert is often not fast enough, and people might not receive the alert until they have strayed far away from their neglected devices. The problem with the current method is that it employs the received signal strength (RSSI) of Bluetooth Low Energy (BLE) beacons and assumes that an RSSI value below a certain threshold indicates separation of the devices. RSSI values vary significantly depending on the device placement around the body, so it is difficult to pick a meaningful threshold.

We illustrate this problem in Figure 1 where we plot BLE beacon RSSI values sent by a BLE system on chip (SoC) worn on the wrist and measured by a phone placed in several configurations around a user (phone in a pocket, on the table by the user, or held to his ear; wrist wearable in his lap or on the table). We see from these box plots, each of which represents one minute's worth of data, that the configuration greatly affects received signal strength, despite the devices remaining close. For comparison, we also plot the median signal strength of BLE beacon packets received by the mobile phone in a user's pocket when the user increases his distance from the beaconing BLE device. Using distance values from this plot and the RSSI values of the configurations, we would incorrectly assign separation distances of anything between 2 and 45 meters to the various configurations.

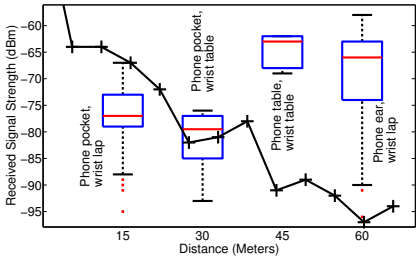

Figure 1: Signal strength of BLE beacons from a wrist wearable measured on a mobile phone. We also overlay a plot of RSSI versus distance. The X axis (distance in meters) applies only to this RSSI plot.

In this paper, we describe a new lower-latency method for detecting the personal collocation of mobile devices by comparing footstep patterns detected on each device. Our method is suitable for any collection of personal devices, although we focus here on phones and smartwatches. Nearly all current types of mobile devices include a 3-axis accelerom-

HOTMOBILE '15 Santa Fe, New Mexico USA

eter useful for recognizing various human activities. Accelerometers are now sufficiently low power to run continuously with minimal energy overhead [6]. We also see a trend toward using special-purpose processing cores to minimize the cost of analyzing motion data [1]. Human mobility is a natural fit to achieve detection of device separation, and its latency scales according to the speed of the given user.

While we can detect many interesting movement features from full-bandwidth accelerometer data, we only need to detect walking patterns. Since normal human movement patterns include regular footstep intervals when moving from one location to another, detecting and comparing signatures based on the timestamps of these footstep events suffices to detect personal device collocation in most cases. Since human footsteps occur at relatively low rates, processing requirements are low and can execute periodically, at a rate proportional to human mobility rates.

To avoid transmitting bulky raw accelerometer data, our mobile devices run a lightweight step detection algorithm locally and transmit the timing of step events in a single BLE packet. Since connection maintenance requires exchanging periodic radio packets anyway, exchanging step signatures consumes negligible additional energy overhead.

After comparing footstep data on a chosen device (in this case, the user's mobile phone), our technique informs personally collocated mobile devices of their status. With this method, low-latency detection implies that the devices are not yet very far apart, which is an advantage for the applications we describe in the next section. Note that by itself, our method is not foolproof: it is vulnerable to spoofing, as described later in this paper. Instead, it provides an early warning of device separation suitable for many applications. We also find that combining our technique with the current RSSI-based technique strengthens the behavior of both.

2. APPLICATIONS

What are some potential applications for our collocation technique? We highlight a few below, all of which assume that a wearable device, such as a smartwatch, remains on its owner. Devices closely collocated with the smartwatch are therefore personally collocated.

Forgot my device: The more mobile devices we carry with us, the easier it is to lose one of them. We can use a wearable such as a smartwatch to alert us to separation from other devices. Low-latency detection can mean the difference between reclaiming the phone you left a couple of seats behind you, or watching the bus drive away it. Low-latency detection also gives bad guys less time to steal or otherwise compromise the forgotten device.

Automated message responses: From the point of view of a user's phone, the sooner it knows it is no longer with its owner, the sooner it knows the owner cannot respond to calls or messages. The phone can then trigger automated responses to senders or redirect their communications toward another device known to be near the owner.

Early screen lock: Managing the security of our devices is more burdensome as the number of devices increases. Mobile phones, for example, typically use a screenlock to protect the contents of the phone after an idle period. A short idle period reduces the chance of compromise when a user leaves his phone unattended, but it can also frustrate the user who must more often unlock his phone. Our technique can lock the phone quickly if the user is not with it, which allows for longer timeout periods while the device is still personally collocated.

3. OUR APPROACH

In this section, we explain our accelerometer-based personal collocation system. First we show how to establish and revoke a personal collocation relationship between a pair of devices based on their mobility. Next, we provide an overview of the step detection algorithm we use to identify this mobility. Finally, we describe the algorithm that compares device mobility to determine whether devices are personally collocated.

3.1 Sensing Collocation from Mobility

Personal collocation and mobility are related, but not synonymous. Two devices could be moving, but not with the same user. They could also be stationary, but physically proximate to the same user. We ask the question: *How can we use human footstep patterns detected on different devices to determine which subset of devices are currently with a given user?*

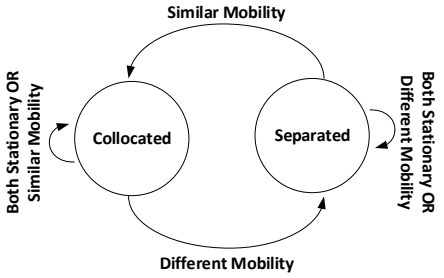

Figure 2: We model mobility-based personal collocation as transitions between two logical states.

In Figure 2, we model collocation as a state machine and show how the presence or absence of human mobility maps the devices to a *collocated* or a *separated* state. Initially, we have no information to determine whether two devices are together or not, and we assume they are *separated*. Devices remain in the *separated* state until they both move with their user, experiencing similar human mobility. After some amount of similar mobility, we consider the devices *collocated*. If both collocated devices keep moving or stop moving together, they remain *collocated*. If one device experiences mobility and the other does not, they transition back to the *separated* state. Next we describe how we detect human mobility.

3.2 Detecting Human Mobility

To detect whether or not two devices are with the same user, we use a human-centric mobility detection mechanism – sensing footsteps. Mobile devices that include accelerometers that are rigidly attached to the body or loosely held in the user's hand or pocket each observe the same footstep events. To disambiguate footsteps from other movement features, we modify a well-established technique [8] to suit low-power wearable devices, as described next.

Acceleration magnitude: Depending on the orientation of a device, one of the three axes of the accelerometer registers higher variation in acceleration due to the footstep. To avoid the computation overhead of tracking device orientation or handling data from each axis separately, our version

of the algorithm simply computes the magnitude of the accelerometer axis samples:

$accelmag = \sqrt{accel_x^2 + accel_y^2 + accel_z^2}$

Noise smoothing and activity filtering: We then filter out spurious accelerometer noise by using a moving windowed average computed across accelerometer magnitude samples. The size of the window over which we smooth the accelerometer samples depends upon the frequency of input samples. In our case we found that for an input size of 50 samples per second, a window size of four works well:

$magnitude_t = (accelmag_t + accelmag_{t-1} + accelmag_{t-2} + accelmag_{t-3})/4$

After smoothing the data, our algorithm estimates the variability of samples by measuring the difference between maximum and minimum accelerometer magnitude. We find that a magnitude greater than 0.4 g (refer to section 5) detects footsteps across various human activities, and filters out lesser motion due to activities such as typing.

Dynamic threshold step detection: Next, we compute a dynamic threshold to detect the presence of a step. We estimate this quantity, updated every second, as:

$dyn = \frac{(max\_magnitude + min\_magnitude)}{2}$

We detect a step if the accelerometer magnitude shows negative slope crossing this dynamic threshold.

Frequency filter: We use one further filter to shape the data. Large but rapid vibrations recorded by the accelerometer unrelated to walking or running may count as steps. We discard these spurious movements by constraining the frequency of step counts. We assume that humans typically can run as fast as five steps per second but will walk no slower than one step per second. Therefore the time gap between two valid steps should be in the range [0.2 sec, 1 sec]. Next, we discuss how we infer mobility pattern similarity observed by multiple devices.

3.3 Comparing Human Mobility Patterns

After detecting footsteps on each device during a given one second interval, we need a robust mechanism to decide whether both of the devices are in motion, both of the devices are stationary, or only one of them is in motion. Since the detection of individual footsteps on a particular device is not completely reliable, we need a method that is robust to variations in the timing of individual steps. In particular, it can be challenging to match step data for devices at different locations on the body (i.e. phone in the pocket, smart watch on the wrist). The intuition behind our technique is to filter out short-lived variations in step counts by looking for sustained periods of motion on the devices. Later in the evaluation section we show how much filtering we require for practical application scenarios.

We implement our collocation algorithm as a finite state machine that is somewhat more complex than previously illustrated, because the state transitions between *Separated* and *Collocated* require some counting. Both devices start in the state *Separated*, since we have not yet collected any motion data. During a one second interval, if both devices detect one or more footsteps, we transition to state *Maybe Collocated*. The devices will remain in this state and increment a counter, provided that both devices continue to see at least one footstep every interval. If this condition is violated, we reset the counter and we transition back to state *Separated*. When the count value reaches at least threshold value x, we transition to state *Collocated*. The two devices remain in state *Collocated* until one of the two devices detects footsteps while the other does not. If that happens, we transition to the state *Maybe Separated*. For each interval that we see activity on one of the devices and not the other, we increment a counter. We transition back to state *Separated* when the counter reaches threshold value y. We refer to the number of consecutive intervals that contain footsteps as the *consecutive interval threshold*.

4. IMPLEMENTATION

To evaluate step-based personal collocation, we use the combination of a trace-driven simulator and a hardware implementation. The trace-driven simulations consist of an offline, simulated version of our algorithm that uses acceleration data collected from a commodity smartwatch and smartphone to determine their collocation state. The advantage of this approach is that it provides access to full-bandwidth accelerometer data and allows us to tune various system parameters to understand their impact on performance. Our hardware system implementation determines collocation state in real-time and uses custom hardware – this platform enables us to understand the power profile of our algorithm, but it lacks the full bandwidth data required for tuning.

Both implementations have three main components: a *step-detection module* running on both the smartphone and wearable device, a *step-data handler* to relay the steps detected on the wearable device to the smartphone, and a *collocation-detector module* running on the smartphone.

4.1 Trace-Driven Simulations

To understand and evaluate the efficacy of our system, we need to understand the impact of various parameters on system performance. We created a trace-driven simulation framework that allows us easily to modify system parameters and compare performance. We use the MotoACTV (Android 2.1) smartwatch (`https://motoactv.com/home/page/features.html`) and a Samsung Galaxy S3 (Android 4.3) smartphone to collect time synchronized 3-axis acceleration values.

We developed an app for these devices that is configured to label a set of tasks such as sitting, walking and running. We ask users to perform a sequence of predetermined tasks and press a button when transitioning between these tasks. We use the system timestamps of these button presses as ground truth for activity timing in the evaluation. The app continuously records accelerometer values obtained from the device, as well as the signal strength of Bluetooth devices encountered during the duration of the experiment. To ensure the data from both devices is properly aligned, we synchronize them to the same Network Time Protocol (NTP) server just prior to data collection. After data collection, a set of MATLAB scripts parse through the accelerometer data and determine collocation state across the entire length of the trace. We compare the times of collocation state transitions to the labeled ground truth to determine the overall latency of collocation detection.

4.2 Hardware and Software Implementation

Wrist Wearable Device To understand the power profile of a state-of-the-art wearable device we use a development board from Nordic Semiconductor based on the 51822,

augmented with a 3-axis accelerometer (ADXL362) for step detection. The accelerometer continuously samples acceleration values at 50 hz and stores the data in a FIFO buffer. Once every second, the Nordic SoC reads the 50 accumulated samples and passes the data to the step-data handler service (described below) for further encoding and transmission to the smartphone over a BLE channel.

We designed the step-data handler to minimize the number of BLE packet transmissions required to relay the complete step-data from the wearable device to the smartphone. Upon receiving the step data from the step detection module, the step-data handler extracts the following information: number of steps detected N_{step}, the timestamp of the first step detected T_{first_step}, and the indices out of 50 accelerometer samples at which subsequent steps occur t_i. The BLE data packet therefore has the following structure: $<N_{step}, T_{first\_step}, t_1, t_2, ..., t_{N_{step}-1}>$. Since the device samples the accelerometer values at a uniform rate f_{sample} ($\frac{1}{50}$ in our case), we can compute t_i as:

$t_i = T_{first\_step} + i * f_{sample}$

N_{step} and t_i are one byte long and T_{first_step} is four bytes long. The maximum amount of data a BLE packet can hold is 27 bytes. Therefore, one BLE packet is enough to relay data for 23 steps which is more than sufficient for our purposes. The BLE packet is then immediately queued for transmission. In our solution, the wearable and phone form a connection to exchange step and collocation state packets; while BLE supports a range of different interval settings for maintaining a connection, we choose an interval identical to the data generation rate (once per second) to minimize connection overheads.

Smartphone: We use a Samsuing Galaxy S3 as the smartphone device in our hardware implementation. The step-detection module on the smartphone samples accelerometer magnitudes at 50 hz and runs the same step detection algorithm as the smartwatch. Instead of trying to align the timestamps of individual footsteps, the collocation-detector module partitions the step data into multiple intervals (bin) of identical width for both the wearable and the phone and focuses on comparing properties of the bins such as total number of steps and amount of time between the steps. The collocation detector uses the state transition rules previously described to determine the collocation of the devices.

5. EVALUATION

In this section we evaluate our system using the implementations described in Section 4. We compare the performance of our system against RSSI-based collocation techniques and show that we provide performance improvements while operating within a minimal power budget. In each experiment, we evaluate our system by analyzing the accelerometer traces collected from a watch and smartphone for various daily activities. We categorize these activities as:

Steps detected on one device: *Drinking*, *Typing* and *Gesturing* represent scenarios where the watch accelerometer detects motion from the user's hands while the smartphone registers no activity because it is in the user's pants pocket or next to them on a table.

Steps detected on both the devices: *Walking*, *Running*, and *Stair-climbing* represent activities where both the devices register vibrations due to footsteps.

Dataset	mean(g)	std (g)	min (g)	max(g)
Drinking	0.94	0.08	0.75	1.70
Typing	0.93	0.04	0.53	1.21
Gesturing	0.97	0.14	0.44	1.86
Walking	1.08	0.19	0.69	1.83
Running	1.91	1.17	0.39	6.56
Stair-climbing	1.06	0.21	0.57	1.79

Table 1: Summary of wrist acceleration datasets used for evaluating the stability of collocation detection.

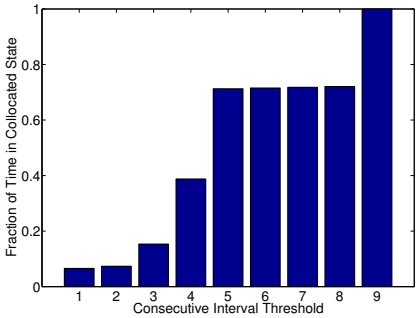

Figure 3: The fraction of time spent in the collocation state while a user gestures with his hands for 2 minutes with a device on his wrist and the phone in his pants pocket.

5.1 Stability of Collocation State

One measure of the performance of our system is its stability in the *collocated* state under adverse conditions. For example, a user with both a smartwatch and phone walks into her office, sits down in front of her computer, puts her phone on the desk and starts working. In this scenario, both her devices will observe footsteps until she puts the phone on the desk. While the phone is on the desk, the watch may still detect arm motion, but both devices should remain in a collocated state. Therefore, our algorithm should reject false-positive steps from arm motions to avoid falsely reporting the devices have separated.

To tune the parameters of our collocation technique for stability, we look at the motion properties of the benchmark datasets shown in Table 1. We collect five example traces from the wrist of a single user, where each trace contains two minutes of activity data. We first observe that there is more variability in acceleration magnitude when the user is moving rather than when stationary. In the walking, running, and stair-climbing benchmarks, we find that the devices continue to stay in *collocated* state, provided that we require the difference between the maximum and minimum acceleration values be more than 0.4g during each one-second interval in which we detect steps. For the stationary benchmarks, we pre-initialize both the devices to the *collocated* state. During these experiments, we leave the phone on the table while the user performs the activities, meaning that our step detection algorithm could falsely determine the user has walked away from the phone. For typing and drinking we find no cases where the collocation state falsely transitions to *separated*. However, we had more trouble in case of *gesturing*, since this activity involves large and rapid motion of the hands.

To reject these motions, one technique is to increase the acceleration magnitude threshold required during each interval– we find that by increasing this value to 0.5 g, we can eliminate most occurrences of the false positives. However, this

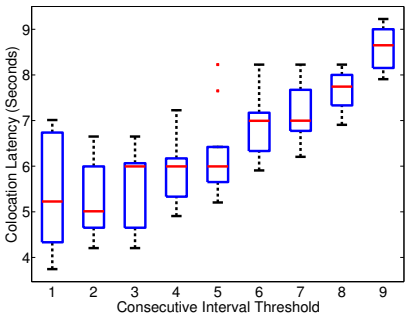

Figure 5: Latency of establishing collocation.

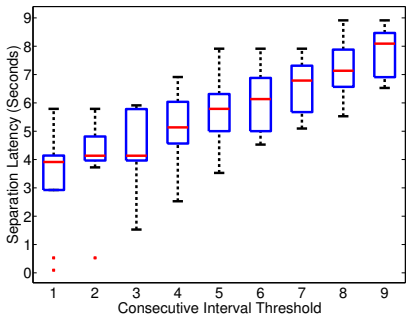

Figure 6: Latency of establishing separation.

threshold setting results in the step detection algorithm missing steps during walking activites. Instead, we filter these transitions by increasing the consecutive interval threshold described in section 3.3 and show the resulting impact in Figure 3. For evaluation, we compute a mean across all the gestre traces of the fraction of time the devices spend in the collocated state – ideally this should be one. Since gesturing involves significant vertical motions of the wrist, we find that we need to increase the interval threshold to nine to filter these bursts of motion and remain in the *collocated state*. The drawback of using such an agressive threshold is an increase in the latency of the detection of true separation events, as described below.

The stability of the *separation* state is also important to the user experience, but it is unlikely that two devices will accidentally show similar enough motion to become collocated, unless the motion is injected by a malicious user (see Section 7).

5.2 Latency of Collocation Detection

Another measure of our system is how long it takes to detect transitions between personal collocation of devices and their separation. The latency of detecting these transitions is essential to the user experience for applications. To explore the latency tradeoffs, we collect data similar to that shown in Figure 4. A user wearing a smartwatch and carrying a phone in his hand walks towards a desk, sits down, leaves the phone on the desk, walks away, walks back towards the desk and then walks away with the phone and the wearable. We collected 10 instances of a user performing this series of motions. In 100% of these examples, we correctly identify the states of *separation* and *collocation* as compared to user-generated ground truth, but the latency of detection varies. Figures 5 and 6 show the impact of increasing the interval threshold on collocation and separation detection. A small interval threshold results in fast detection of separation and collocation but will have the aforementioned stability issues when spurious wrist motions are present. A larger interval threshold provides stability in the presence of noisy wrist data at the cost of higher detection latency. Nonetheless, even the higher-latency results compare favorably to using RSSI to detect collocation, as described next.

5.3 Comparison to RSSI based techniques

We compare the performance of step-based personal collocation to one that uses received signal strength to determine that two devices are separated. On his wrist, a user wears both a Nordic evaluation board beaconing at +4dBm every 25 ms, and a MotoACTV logging acceleration. Since the

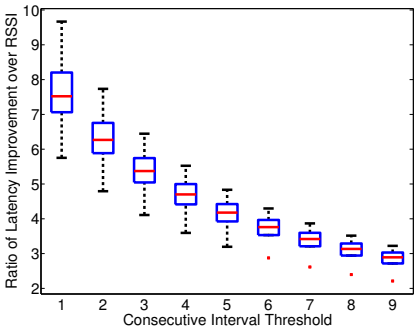

Figure 7: The latency of device separation is much lower using our step-based technique, rather than an RSSI-based technique.

RSSI values contain a significant amount of noise, we use a 30-sample windowed average to smooth the data. The phone logs its own acceleration as well as the signal strength of the beacons transmitted by the Nordic SoC. The user leaves the phone on the table and walks away. The RSSI-based technique assumes that the two devices become separated at the distance when RSSI drops below -85 dBm – we chose this threshold based on the data presented in Figure 1. We run the step-based collocation algorithm with a varying window threshold parameter and compute the latency in seconds between when the user starts walking and when the two devices are considered separated. In Figure 7 we show the improvement of step-based collocation as compared to the RSSI-only technique. With a window threshold parameter of 1, step-based personal collocation achieves ∼7.5x lower latency than the RSSI-based technique. In a noisier scenario, such as when the user is gesturing, we need a window threshold of 9 as shown in Figure 3. Even with this aggressive setting, step-based collocation still performs ∼ 3x better.

5.4 Energy Efficiency

To evaluate the energy efficiency of our solution, we measure the energy consumed by the smartwatch hardware implementation described in Section 4.2 and summarize the average power overhead imposed by our approach. When considering the CPU, accelerometer, and radio components of a wearable device, the combined costs result in only 21.4 μW of average power. To put this in perspective, our service consumes 1.94 Joules of energy per day out of the typical ∼4428 Joules available on a modern wrist wearable (Samsung Gear Live used as reference platform).

The power consumption of the step detection algorithm

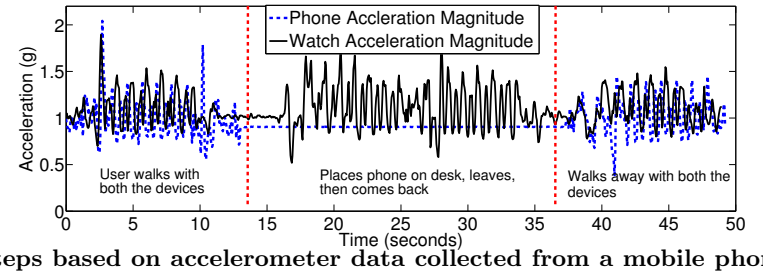

Figure 4: We detect steps based on accelerometer data collected from a mobile phone held in the hand and a smartwatch. This example shows transitions between collocated and separated states.

running on the smartphone is also important. We leave a detailed power consumption analysis to future work. However, we note that modern smartphones (i.e. Motorola Moto X, LG Nexus 5, Apple iPhone 5S) have dedicated coprocessors for collecting step data. Pedometer APIs in Android 4.4 enable apps to get a step count several times a second while leaving the CPU in a sleep state, which indicates our approach should have negligible impact on battery life.

6. RELATED WORK

Systems such as the Sound of Silence [7] use similarity of audio fingerprints to detect device collocation. However, these systems detect conversational-level collocation, which can range further than personal collocation.

Other work has also used motion detection to determine personal collocation of devices. One technique compares full-bandwidth accelerometer data from a set of medical sensors to determine whether they are on the same person [3]. This is more communication intense technique than our step-based approach, and the authors use it only to establish a secure pairing for the devices' Bluetooth radios. Another technique uses the Fast Fourier Transform of accelerometer data collected from each device and a coherence matrix to determine whether a pair of devices are together [5]. This work distinguishes whether devices are on the same person while the person is moving, but it does not handle corner cases when there may be no movement at all.

Another way to determine that devices are on the same person is capacitive communication through the user's skin [2]. While this is a robust method of detecting device collocation, it is limited to devices that have direct contact with the users skin. For example, a medical sensor could communicate with a wrist wearable device. However, this technique is harder to extend to mobile phones or tablets which are often in the user's pocket or bag.

7. LIMITATIONS AND FUTURE WORK

While our step-based method for detecting personal collocation has several advantages over existing RSSI-based methods, it also has limitations: the most important being that it is vulnerable to spoofing. Our method is intended to assist users with device management, but it does not protect against adversaries trying to fool the system. For example, if an adversary steals a user's device and walks alongside him, our collocation detector will say the two devices are collocated even though they are on different people.

Part of our future work is to combine our technique with RSSI-based methods. A decrease in observed signal strength will help alert us to separation of devices in the case of the adversary mention above, but the combination of techniques also addresses their different drawbacks. The problem for RSSI-based systems is that they have no ground truth reference for when devices are collocated. Using our approach, we can establish a baseline RSSI value when both devices are stationary but collocated. When one of the devices detects footsteps, we can then look at the trend of RSSI values. If the person is sitting still and these are false positive step events, we should observe no prolonged downward trend in RSSI readings and can retain the collocation relationship.

8. CONCLUSION

We describe a new, lightweight approach toward detecting personal collcation of devices using footstep signatures. Since the technique uses motion data directly to approximate distance between devices, it can detect collocation at lower latency than radio-based approaches that use time-varying, noisy RSSI data over distance. By leveraging low-power accelerometer hardware, our approach imposes little energy overhead beyond that required by BLE connections and pedometer-based functionality.

9. REFERENCES

[1] The iphone 5s, the moto x, and the rise of the co-processor. `http://arstechnica.com/gadgets/2013/09/the-iphone-5s-the-moto-x-and-the-rise-of-the-co-processor`.

[2] J. Bae, H. Cho, K. Song, H. Lee, and H.-J. Yoo. The signal transmission mechanism on the surface of human body for body channel communication. *Microwave Theory and Techniques, IEEE Transactions on*, 60(3):582–593, 2012.

[3] C. T. Cornelius and D. F. Kotz. Recognizing whether sensors are on the same body. *Pervasive and Mobile Computing*, 8(6):822–836, 2012.

[4] C. V. N. Index. Global mobile data traffic forecast update, 2011–2016, cisco systems. *Inc., Feb*, 2014.

[5] J. Lester, B. Hannaford, and G. Borriello. "are you with me?"–using accelerometers to determine if two devices are carried by the same person. In *Pervasive computing*, pages 33–50. Springer, 2004.

[6] B. Priyantha, D. Lymberopoulos, and J. Liu. Enabling energy efficient continuous sensing on mobile phones with littlerock. In *Proceedings of IPSN*, pages 420–421. ACM, 2010.

[7] W.-T. Tan, M. Baker, B. Lee, and R. Samadani. The sound of silence. In *Proceedings of SenSys*, page 19. ACM, 2013.

[8] N. Zhao. Full-featured pedometer design realized with 3-axis digital accelerometer. *Analog Dialogue*, 44(06), 2010.