Exploiting Smartphone Peripherals for Precise Time Synchronization

Sandeep Singh Sandha*, Joseph Noor*, Fatima M. Anwar, Mani Srivastava
University of California, Los Angeles
{ssandha, jnoor, fatimanwar, mbs}@ucla.edu

***Abstract*—Achieving precise time synchronization across a collection of smartphones poses unique challenges due to their limited hardware support, exclusively wireless networking interface, and restricted timing stack control. Given the ubiquity and popularity of smartphones in modern distributed applications, clock discrepancies often lead to degraded application performance. In this paper, we present and evaluate alternative approaches to attain precise time synchronization by leveraging the various peripherals available on modern smartphone devices. Our evaluation across Android smartphones typically attains synchronization accuracy within (i) $200\mu s$ using audio, (ii) $3000\mu s$ using Bluetooth Low Energy, and (iii) $1000\mu s$ using Wi-Fi. Under certain conditions, we show that smartphones synchronized using one peripheral can accurately timestamp and generate synchronous events over other peripherals. The provided guide and accompanying open-source implementations offer developers a means to select the appropriate time synchronization technique when building distributed applications.**

***Index Terms*—Clock Synchronization, Mobile Devices, Distributed Applications, Peripherals, Timestamping Delays**

I. Introduction

Smartphones are by and large one of the most widely pervasive computing devices. Given their extraordinary availability and wide array of sensing modalities, modern smartphones serve as an integral component in many emerging distributed applications, including those spanning the domains of crowdsensing, smart homes, and mobile health [1]. A unique aspect of these domains is their transitioning towards the deriving of complex inferences by combining data across a set of devices.

A shared notion of time is an essential requirement for applications intending to fuse data and/or coordinate concerted actions across multiple devices [2]. Applications on smartphones including beamforming and localization require precise time synchronization on the order of microseconds to maintain correctness [3]. Popular apps such as AmpMe, which converts a collection of smartphones into a distributed music player, requires sync error of less than 10 milliseconds to prevent ear fatigue [4]. Other IoT systems where smartphones serve an integral role, including crowdsensing, robotics, and cross-modal deep learning at the edge, demand best-effort synchronization on the order of milliseconds [2], [5].

Achieving precise time synchronization across a suite of smartphones pose unique challenges. First, hardware support for protocols such as PTP [6] is unavailable. Second, these devices operate exclusively via wireless networking, which has been shown to exacerbate the variability of achievable accuracy [7]. Third, given the overall difficulty in achieving precise time, evaluating the accuracy of a particular synchronization mechanism is in-and-of-itself a troublesome task; that is, a ground truth baseline is impossible to attain. Finally, and most importantly, the smartphone platforms (e.g., Android, iOS) restrict timing stack control solely to the underlying operating system, with poor clock maintenance [7].

Given the rich suite of capabilities on smartphones, there is a myriad of alternative techniques that can be employed to achieve clock synchronization. For local area relative synchronization, Lazik et al. [3] introduced an audio-based clock synchronization method that relies on external beacons to achieve sub-millisecond precision. For wide-area synchronization, Yan et al. [8] leverage skin electric potentials measured from external wearables, achieving accuracy on the order of milliseconds. For general-purpose timing, wireless NTP client libraries are available that can typically provide accuracy in the order of tens of milliseconds [9].

Each of the available mechanisms for smartphone synchronization possesses unique tradeoffs that result in *one size does not fit all* policy. Selecting the appropriate technique then becomes a function of the requirements and characteristics of a particular application over distributed devices. Some synchronization requires external infrastructure or tightly integrated peripheral sensors [10]. Others are restrictive to local area networks [3]. In considering these tradeoffs, the questions we pose and strive to answer in this paper are: (1) *which peripherals can be used to synchronize distributed smartphones*? (2) *Which synchronization technique best suits a particular peripheral*? Also, (3) *how do different peripheral synchronizations compare with each other?*

To answer these questions, this work exploits various peripherals on smartphones to provide a comprehensive implementation and evaluation of precise clock synchronization solutions for modern distributed applications spanning smartphones. We begin by describing various smartphone peripherals over which clock synchronization can be achieved.

The research reported in this paper was sponsored in part by the National Science Foundation (NSF) under award # CNS-1329755, by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and by the Army Research Laboratory (ARL) under Cooperative Agreement W911NF-17-2-0196. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL, DARPA, NSF, SRC, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

*Both authors contributed equally to this research.

Provided implementations are based on either receiver-receiver [11] or sender-receiver [12] synchronization protocols. Due to fundamental hardware limitations of specific smartphone peripherals and their reliance on external infrastructure for synchronization, we focus on audio subsystem, Bluetooth Low Energy, and Wi-Fi in particular. Given the difficulty in evaluating time sync approaches on smartphones, we show how to probe the accuracy of a particular technique effectively.

We present the first work that provides a detailed comparison of cross-peripheral synchronization on a smartphone. Traditionally, reliance on specialized hardware for low-level timestamping and precise synchronization of peripheral clocks has restricted the use of one peripheral sync for the other. In this work, we argue that cross-peripheral sync is viable without low level timestamping or specialized hardware support by using the shared monotonic clock. Given symmetric stack delays of a particular peripheral across smartphones, we provide cross-peripheral sync that is good for timestamping and generating synchronous events across different peripherals.

Our contributions can be summarized as follows:

1) A comprehensive breakdown of the various available peripherals and techniques to achieve precise time synchronization across smartphones.
2) A variability and cross-peripheral evaluation methodology to probe the accuracy and applicability of each available sync solution.
3) An offering of open source library implementations for specific techniques to reduce the developer overhead [13].
4) A *cheat-sheet* guide serving to help steer developers into selecting the best-fit solution by detailing the different tradeoffs of each approach.

II. Related Work

Peripherals are key to synchronizing distributed devices with each other. For example, audio peripherals form synchronized acoustic sensing arrays for range finding and target localization [14]. The availability of Bluetooth Low Energy (BLE) in consumer electronics and its low power architecture has made it a viable choice for sensor fusion and time sync [15]. Packet-based synchronization protocols over IEEE 1588 [6], 802.11 [16], 802.14.5 [17] are used extensively for high precision depending upon the quality of transmission and reception timestamps. Note that most of these peripherals expose IO capabilities on embedded platforms for precise synchronization. Although smartphones possess a rich suite of peripherals, they do not provide IO capabilities that can be exploited for high quality timestamping and synchronization.

Synchronizing smartphone clocks in the context of localization has been discussed in recent literature. Lazik et al. localize a smartphone by synchronizing to a network of beacons producing ultrasonic chirps [3]. Beacons are also fed to a smartphone audio peripheral to synchronize a phone with a speaker [10]. The audio peripheral is widely used in smartphone-based localization, either with infrastructure beacons or speakers. In this work, however, we seek to go beyond acoustic capabilities and compare the synchronization capabilities of various peripherals on a smartphone.

There are many applications in distributed sensing that rely on time synchronized smartphones without any infrastructure support. The only available system clock – maintained and controlled by the operating system – is disciplined via NTP and NITZ protocols [1] [7]. Timestamping variability and poor disciplining mechanisms of system clock exacerbate timing precision in smartphones [7]. In contrast, this work provides various alternatives to smartphone synchronization with a comprehensive guideline to which synchronization technique best meets the needs of a particular distributed application.

III. Smartphone Time Synchronization

Given the diverse set of hardware attachments and sensing modalities on modern smartphones, a plethora of techniques can achieve clock synchronization. However, due to the differences in the underlying hardware characteristics (e.g. sampling rate, variance, latency), some approaches are more fundamentally limited in achievable performance.

A. Time Synchronization Approaches

There are two general approaches to synchronizing a collection of devices, commonly referred to as receiver-to-receiver (R2R) and sender-to-receiver (S2R). R2R relies on a common event observed by all devices, thus providing a unified reference point for each device to independently establish and maintain its own clock relative to the reference signal [11]. In recent literature, RF [11], acoustic [14], and power line signals [18] provide reference signals for R2R. S2R relies on a server-client model, with a designated device serving as a reference clock by which other devices attempt to relatively synchronize their clocks [12] [9]. Both R2R and S2R techniques have their pros and cons; while R2R eliminates sender side non-determinism, S2R compensates for propagation delays in the network.

Across distributed smartphone devices, R2R and S2R synchronization can be achieved through a variety of alternative peripherals. An R2R broadcast reference signal may be either opportunistically observed or intentionally generated, and can potentially span any of the available sensing modalities available (e.g. audio, ambient light, proximity, IMU, camera, bluetooth, network). Similarly, S2R synchronization may occur over any of the available wireless networking mediums (e.g. Wi-Fi, Bluetooth, cellular). We select and implement a subset of possible sync techniques across different peripherals; open-source implementations are available at [13]. Our goal is to provide a set of plausible, real-world solutions that can be reasonably integrated without requiring a fundamental reworking of the overall system.

1) Receiver-to-Receiver: Many of the sensing modalities on smartphones that can be leveraged for R2R synchronization possess fundamental hardware, infrastructure, or compute limitations restricting its practicality in deployment. The IMU,

[1]To the best of our knowledge, GPS is not used to adjust the Android system clock.

ambient light, and proximity sensors operate at frequencies of 100Hz or less, resulting in a theoretical clock sync limitation of 10ms or more [19]. The camera can offer a slightly higher sampling rate (up to 240Hz), but requires significant compute overhead to inference and extract reference signals, making it generally impractical. Furthermore, generating a reference signal in many of these domains require external infrastructure.

We select the audio subsystem for our R2R implementation due to its high sampling rate. A notable benefit of the audio subsystem is its support for system timestamping, which results in comparatively less jitter in timestamp delays. As such, audio presents the best potential for achieving good performance in precise clock synchronization. Finally, generating an audio broadcast signal is relatively straightforward for smartphones due to the presence of speakers.

To achieve audio clock synchronization, all smartphones listen for a broadcasted audio event, which may be opportunistically observed or intentionally generated. Each smartphone independently timestamps the event upon observation using the reported system timestamping. One smartphone is designated as the reference clock and shares its observed audio event timestamp with all other smartphones. Each device then computes its relative clock offset with respect to the reference device. Previous work by Lazik et al. [10] noted high audio sampling variability in Android, which potentially leads to degraded clock sync performance. In our provided implementation, we account for this variability to enable precise audio clock sync across Android smartphones.

2) Sender-to-Receiver: S2R synchronization can be accomplished through various peripherals, including cellular, wifi, and bluetooth. Cellular in particular often incurs large latencies and asymmetry; as such we collapse its usage into a Wi-Fi based implementation. In contrast, bluetooth offers a close proximity carrier signal with efficient energy consumption, making it a valuable alternative to standard networking. Given that many embedded sensing platforms support Bluetooth Low Energy (BLE), oftentimes without Wi-Fi (e.g. wearables), a bluetooth sync solution improves compatibility with a wider application domain. GPS is left out due to poor location and time accuracy in an indoor setting. Devices periodically ping a reference device, either an external server or one of the smartphones, and compute a relative clock offset. Application-level timestamping is used due to the lack of system timestamping of bluetooth or Wi-Fi events in the smartphone.

In summary, some peripherals serve as sub optimal mediums for precise clock synchronization. Therefore, we select and implement three synchronization techniques across the audio, bluetooth, and Wi-Fi peripherals.

B. Time Synchronization Comparison

When two smartphones are synchronized using a particular peripheral, the clock difference fails to capture the true offset, as it is affected by the peripheral timestamping delays. Fig. 1 illustrates the timestamp delays in audio (t_{audio}), Wi-Fi (t_{wifi}) and BLE (t_{ble}) peripherals in Android to capture the occurrence of an event observed by the hardware. Audio, Wi-Fi and BLE peripherals have access to timestamping capabilities at different stack layers. Kernel level timestamping is available for audio, whereas for Wi-Fi and BLE peripherals timestamping is only available in the application layer. If smartphone A and B are synchronized using a particular peripheral p, the computed $offset$ between two clocks in timestamping p's events is, $offset_p = offset_{true} + (t_p^A - t_p^B)$, where $p \in \{audio,\ ble,\ wifi, ...\}$, $offset_{true}$ is the difference between smartphones monotonic clocks in the absence of the timestamp delays, and t_p^A and t_p^B correspond to the timestamp delays for a peripheral p on smartphone A and B, respectively.

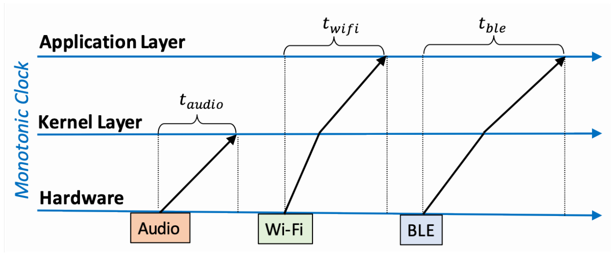


Fig. 1: Timestamping events for audio, Wi-Fi and BLE peripherals in Android. Timestamp delays are not drawn to scale.

After synchronizing smartphones clocks using one peripheral, the same peripheral can be used to listen and generate synchronous events. More precisely, if A and B are synchronized using audio peripheral, the expected error in capturing synchronous audio events on both smartphones is the variability of the term $(t_{audio}^A - t_{audio}^B)$ i.e, the sync error corresponds to timestamp delay jitter of the phones audio pipeline after accounting for clock drift. For minimum variability, it is preferred that timestamp delays are minimized and taken as close to the hardware as possible. Similarly, the accuracy of the Wi-Fi peripheral to observe and generate synchronous network events is the variability of the term $(t_{wifi}^A - t_{wifi}^B)$. The same applies to the BLE peripheral.

Under certain conditions, one synchronized peripheral can be used to timestamp and generate distributed synchronous events across other peripherals. For example, if smartphone A and B are synchronized using audio peripheral, this synchronization can be used to generate synchronous network events on the Wi-Fi peripheral. This is possible only if timestamp delays for a particular clock are symmetric across different devices for the same peripherals. We capture this condition in the following equations:

$$offset_{p1} = offset_{true} + (t_{p1}^A - t_{p1}^B) \quad (1)$$

$$offset_{p2} = offset_{true} + (t_{p2}^A - t_{p2}^B) \quad (2)$$

Subtracting (1) and (2),

$$offset_{p1} - offset_{p2} = (t_{p1}^A - t_{p1}^B) - (t_{p2}^A - t_{p2}^B) \quad (3)$$

In these equations, note that both peripherals $p1$ and $p2$ timestamp events relative to the same monotonic clock as shown in Fig. 1. If $p1$ is audio and $p2$ is Wi-Fi, this error comparison captures the difference between audio and Wi-Fi offset calculations, which directly corresponds to the mismatch in timestamp delays across these peripherals. According to (3), if the timestamping delays of the audio peripheral in devices A and B are the same, and the timestamping delays of the Wi-Fi peripheral in A and B are the same, then their difference in clock offsets are the same. This implies that synchronization across one peripheral can be used for other peripheral events if there is no mismatch in their timestamping delays. In summary, the above equations capture the heterogeneity in timestamp delays for different peripherals stacks. An important observation is that timestamping delays are symmetric when the devices possess the same hardware, kernel, and application load.

IV. Evaluation

Given a particular sync approach and implementation, a challenge arises in how to determine its accuracy. As a ground truth baseline is impossible to attain on smartphones, alternative techniques must be used to approximate overall accuracy and precision of a given time synchronization solution.

First and foremost, a sync solution performed via a particular peripheral is best suited to measure events that occur across that same peripheral. For example, audio subsystem sync is able to most precisely record audio events. Previous work by [3] introduced a methodology of observing variability in sync offsets (Equation 1) after repeated audio sync attempts as a means of quantifying the accuracy of the audio sync solution. More generally, a broadcasted reference event (e.g. audio chirp) can be repeatedly observed and timestamped across multiple smartphones to evaluate sync precision. Hence, variability evaluations capture sync precision for the same peripheral.

An alternative evaluation methodology is to compare two sync solutions that are performed in parallel. This cross-peripheral comparison better quantifies the ability of a particular sync technique using one peripheral to accurately capture events over other peripherals. For example, one might evaluate the accuracy of a Wi-Fi sync implementation with respect to an audio sync implementation. The observed results are fundamentally limited by the combined precision of the two sync solutions. For these results to be meaningful, the baseline must be more accurate than the evaluated approach. As such, this methodology offers an upper bound on precision, indicating how well a particular sync solution maps to other forms of events observable by the smartphone device.

A. Experimental Setup

Two Pixel 3 and two Nexus 5X smartphones comprised the set of devices used for the evaluation. One Pixel 3 was designated as the reference device onto which other smartphones attempt to relatively synchronize their clock. Note that all peripherals are synchronizing the monotonic clock maintained by the operating system. We disable other system level sync attempts to this monotonic clock. The experimental setup consisted of repeated sync attempts for each of the three peripherals. After every successful sync, the computed relative

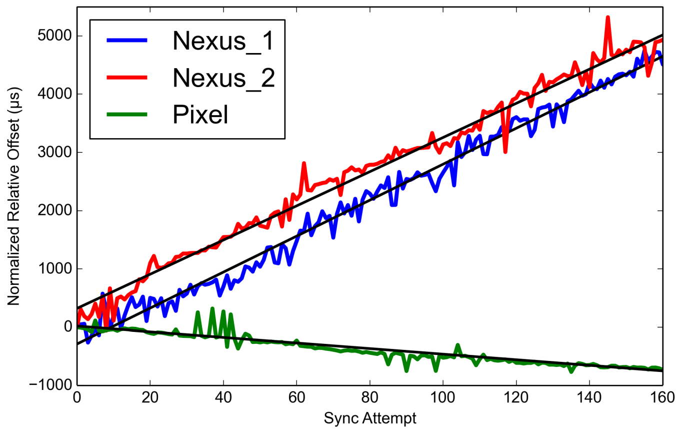


Fig. 2: Relative clock offset for audio-based sync over time, with respect to a fourth phone (Pixel) serving as a reference clock. Results are normalized to the initial computed offset for each device. Due to clock drift, offsets change as a function of the relative drift between the synchronizing device and the reference device. A regression line for each device indicates the overall relative drift trend.

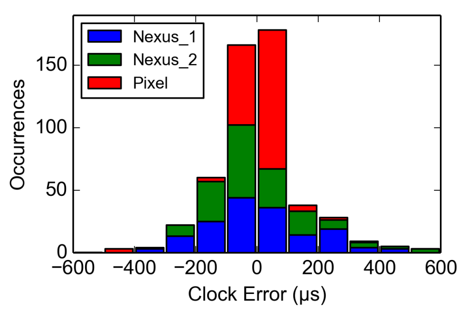
clock offset is recorded. For Wi-Fi and BLE sync, our S2R implementations are based on NTP. Evaluations based on NTP over Wi-Fi best captures performance both at local and global scale for smartphones. Other S2R protocols such as PTP [6] and TPSN [12] require specialized hardware support and as such are not optimized for smartphone sync.

B. Variability Evaluation

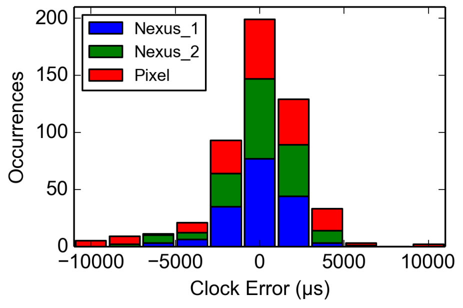
For each of the synchronization techniques introduced, we begin with a reflective evaluation; that is, the precision of each of audio, BLE, and Wi-Fi sync in capturing audio, bluetooth, and wifi events, respectively. These results indicate the ability of a particular sync solution when solely capturing events generated by the same peripheral used to perform synchronization.

1) Drift Correction: Fig. 2 indicates the normalized relative offset for three of the smartphones over the course of an hour, specifically for audio sync. The fourth phone, serving as the reference, is excluded as it would by definition maintain an offset of exactly zero. Due to the differences in clock drift between each smartphone and the reference, each smartphone experienced a different rate of overall change in the total clock offset. A least-squares regression line is plotted for each phone, with the slope indicating the rate of relative drift between each smartphone clock with respect to the reference. The error of each sync attempt can then be estimated as the difference between recorded offsets and this regression line.

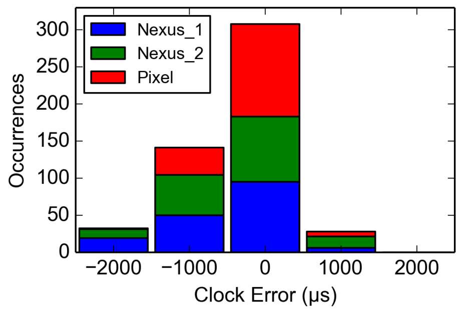
2) Results: Fig. 3 presents three histograms detailing the estimated sync error for the audio (3a), bluetooth (3b), and Wi-Fi (3c) implementations in capturing their own respective peripherals. Audio achieves the best accuracy, with 86% of sync attempts falling within $200\mu s$ of the estimated true offset, and a total spread of just over one millisecond. Bluetooth



(a) Audio sync variability



(b) BLE sync variability



(c) Wi-Fi sync variability

Fig. 3: Sync offset variability with respect to the fourth (Pixel) reference device for (a) audio, (b) BLE, and (c) Wi-Fi implementations. 86% of audio sync attempts fall within $\pm 200\mu s$. 85% of BLE sync attempts fall within $\pm 3000\mu s$. 95% of Wi-Fi sync attempts fall within $\pm 1000\mu s$.

TABLE I: Clock synchronization accuracy across peripherals for three smartphones with respect to a reference Pixel phone. Results are 95% confidence intervals in milliseconds (ms).

	Audio vs Wi-Fi	Audio vs BLE	Wi-Fi vs BLE
Pixel 3	1.51 ± 1.50 ms	2.20 ± 5.06 ms	0.51 ± 5.06 ms
Nexus $5X_1$	13.12 ± 1.50 ms	13.85 ± 4.00 ms	2.15 ± 5.46 ms
Nexus $5X_2$	12.39 ± 1.50 ms	12.93 ± 3.74 ms	-0.86 ± 4.5 ms

sync had an order of magnitude more variability, with 85% of sync attempts falling within $3000\mu s$, and a spread of approximately $20ms$. Wi-Fi sync variability fell in between the two, with 95% of sync attempts falling within $1000\mu s$ and a spread of $3ms$. Despite the fact that wireless NTP has been previously shown to present errors on the order of tens of milliseconds [7], our controlled experiment with a collection of local smartphones synchronizing with the same server (17.253.26.253) over campus Wi-Fi was able to achieve a precision of only three milliseconds. Wi-Fi NTP round trip time was commonly observed between 4 to 8ms, whereas BLE NTP round trip time was typically between 35 and 50ms.

C. *Cross-Peripheral Evaluation*

A variability evaluation characterizes the precision of a particular sync implementation in capturing events generated across its own peripheral. However, in practical deployment synchronized clocks are often used to timestamp events across a wide suite of peripherals. Given the diverse set of capabilities on smartphones (e.g. ambient light, audio, bluetooth, camera, IMU, proximity, network, proximity), exclusively reporting a sync method's precision with respect to its own peripheral is insufficient. A complete evaluation captures the differences across peripherals; this can be accomplished by comparing two parallel synchronization methods using alternate peripherals. Table I presents the results from comparing each of the three clock sync techniques with respect to the other two synchronized clocks using Equation 3. One of the Pixel phones was reserved as the reference device to which other phones are attempting to synchronize. Each sync technique was performed and compared at least 100 times; 95% confidence intervals are reported. Confidence interval sizes are a function of the combined variability of the two peripherals; as such, the comparisons using BLE present the highest error bounds. One interesting observation is the significant bias induced when comparing clock sync solutions between varying hardware. While the Audio-WiFi and Audio-BLE overall comparison are near zero when comparing two Pixel devices, a bias of ˜13ms are induced when comparing Nexus to Pixel devices. This is a direct result of the varying audio, Wi-Fi, and BLE stack latencies between Pixel and Nexus smartphones. These results indicate that an audio sync solution in particular fails to extrapolate to other peripherals; in contrast, WiFi-BLE comparisons maintain reasonably comparable results independent of the hardware devices evaluated. On the other hand, when comparing peripherals across similar hardware, audio sync can be used to synchronously generate Wi-Fi and BLE events; this is shown by the minimal differences in stack latencies across both Pixel devices.

V. Discussion

The choice in selecting a sync solution for a smartphone incurs varying tradeoffs. No technique is objectively optimal; the appropriate selection is dependent on a number of application factors, most notably the range of participating devices and the active peripherals required.

A. *Tradeoffs*

Bluetooth presents the most straightforward tradeoff; it has high variability and limited range, and as such should be reserved for usage where the gain in energy efficiency is worth the sacrifice in precision. Averaging offsets from repeated bluetooth sync attempts can help alleviate the impact of this variability. The exception to this tradeoff is when interfacing with devices that only support bluetooth; when capturing bluetooth events, bluetooth sync will most accurately account for differing BLE stack latencies across varying hardware.

The audio subsystem tradeoffs are more complex. While system timestamping provides audio sync with the lowest variability, processing broadcasted audio events demand low relative ambient noise in the generated audio frequency band (i.e.

sufficient signal-to-noise ratio). This restricts the environments where audio sync can be deployed, and places a dynamic range limitation. Finally, despite the fact that audio has the lowest variability, Table I indicates that clocks synchronized with audio events are limited in their accuracy of cross-peripheral timestamping; this is likely due to notable differences in audio subsystem latencies across varying hardware platforms.

Wi-Fi synchronization is the most general-purpose and flexible solution. While other techniques rely on local-area relative synchronization, NTP over Wi-Fi can support wide-area deployments. However, as network characteristics differ, so does the achievable clock sync precision. Nevertheless, NTP over a local network can achieve precision on the order of a few milliseconds with minimal effort and relatively low-overhead (i.e. does not require microphone, speaker, or bluetooth to be active).

B. Recommended Sync Solution

Given the order of magnitude reduction in precision when comparing same peripheral variability to cross-peripheral clock comparisons, an overwhelmingly clear message arises: in order to precisely timestamp events arriving across a particular peripheral, that same peripheral should be used to perform clock synchronization. For applications combining information across multiple peripherals and multiple smartphones, one phone should be selected as a reference device onto which all other smartphones synchronize each peripheral independently. An exception to this rule is when peripherals incurs similar latencies across different devices; in this case, one peripheral synchronization can be used to synchronously sense and/or generate events across the other peripheral.

VI. Conclusion

Achieving precise time synchronization across smartphones is a challenge that is further exacerbated by the diverse suite of peripherals available on these platforms. We present an analysis of various alternate mechanisms to achieve clock synchronization on smartphones, including both receiver-to-receiver and sender-to-receiver techniques. Open-source implementations are provided for particular peripherals (audio, bluetooth, and network) that can achieve reasonable performance while maintaining compatibility across a wide array of distributed applications spanning smartphones.

Evaluating the precision of a synchronization technique is in-and-of-itself a challenging problem, particularly for smartphones devices. We discuss two complementary evaluation methodologies: variability analysis of the same peripheral to determine the ability of a sync technique to accurately timestamp with respect to its own modality, and a cross-peripheral analysis to evaluate the utility of a particular sync solution in timestamping other device peripherals.

In future work, we aim to quantify the varying delays across the different peripheral stacks, so that a device may be precisely synchronized across all sensors. This will enable multiple devices and multiple peripherals to accurately synchronize, thus providing a true shared notion of time across varying hardware platforms, vendors, smartphones, and other wireless and embedded devices.

References

[1] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 402–427, 2013.

[2] S. D'souza and R. R. Rajkumar, "Time-based coordination in geo-distributed cyber-physical systems," in *9th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 17)*, 2017.

[3] P. Lazik, N. Rajagopal, B. Sinopoli, and A. Rowe, "Ultrasonic time synchronization and ranging on smartphones," in *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2015, pp. 108–118.

[4] J. Khari, "Ampme plans to kill bluetooth speakers by syncing music between smartphones," https://venturebeat.com/2018/07/03/ampme-plans-to-kill-bluetooth-speakers-by-syncing-music-between-smartphones/, 2018, accessed: 2019-06-28.

[5] T. Xing, S. S. Sandha, B. Balaji, S. Chakraborty, and M. Srivastava, "Enabling edge devices that learn from each other: Cross modal training for activity recognition," in *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*. ACM, 2018, pp. 37–42.

[6] J. Eidson and K. Lee, "Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems," in *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*. Ieee, 2002, pp. 98–105.

[7] S. K. Mani, R. Durairajan, P. Barford, and J. Sommers, "Mntp: Enhancing time synchronization for mobile devices," in *Proceedings of the 2016 Internet Measurement Conference*. ACM, 2016, pp. 335–348.

[8] Z. Yan, Y. Li, R. Tan, and J. Huang, "Application-layer clock synchronization for wearables using skin electric potentials induced by powerline radiation," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 2017, p. 10.

[9] D. Mills, "Executive summary: Computer network time synchronization," https://www.eecis.udel.edu/ mills/exec.html, 2012, accessed: 2019-03-26.

[10] P. Lazik, N. Rajagopal, O. Shih, B. Sinopoli, and A. Rowe, "Alps: A bluetooth and ultrasound platform for mapping and localization," in *Proceedings of the 13th ACM conference on embedded networked sensor systems*. ACM, 2015, pp. 73–84.

[11] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.

[12] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 138–149.

[13] S. S. Sandha, "Github: Time sync across smartphones," https://github.com/nesl/Time-Sync-Across-Smartphones, 2019, accessed: 2019-05-05.

[14] L. Girod and D. Estrin, "Robust range estimation using acoustic and multimodal sensing," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 3. IEEE, 2001, pp. 1312–1320.

[15] S. Sridhar, P. Misra, G. S. Gill, and J. Warrior, "Cheepsync: a time synchronization service for resource constrained bluetooth le advertisers," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 136–143, 2016.

[16] A. Mahmood, G. Gaderer, H. Trsek, S. Schwalowsky, and N. Kerö, "Towards high accuracy in ieee 802.11 based clock synchronization using ptp," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2011, pp. 13–18.

[17] F. M. Anwar and M. B. Srivastava, "Precision time protocol over lr-wpan and 6lowpan," in *2017 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, 2017, pp. 1–6.

[18] A. Rowe, V. Gupta, and R. R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from ac power lines," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2009, pp. 211–224.

[19] Google, "Android sensors," https://source.android.com/devices/sensors /index.html, 2019, accessed: 2019-03-26.