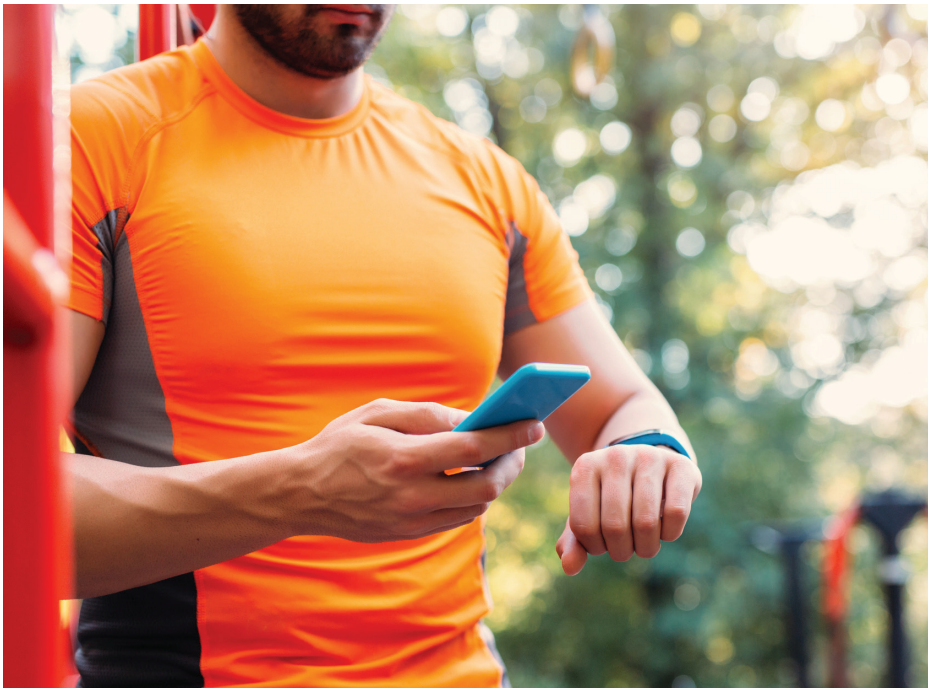
Harini Kolamunna, Jagmohan Chauhan and Yining Hu
University of New South Wales, Australia
Kanchana Thilakarathna *University of Sydney, Australia*
Diego Perino *Telefonica Research, Barcelona, Spain*
Dwight Makaroff *University of Saskatchewan, Canada*
Aruna Seneviratne *University of New South Wales, Australia*

Editor: Geoffrey Challen

ARE WEARABLES READY FOR SECURE AND DIRECT INTERNET COMMUNICATION?

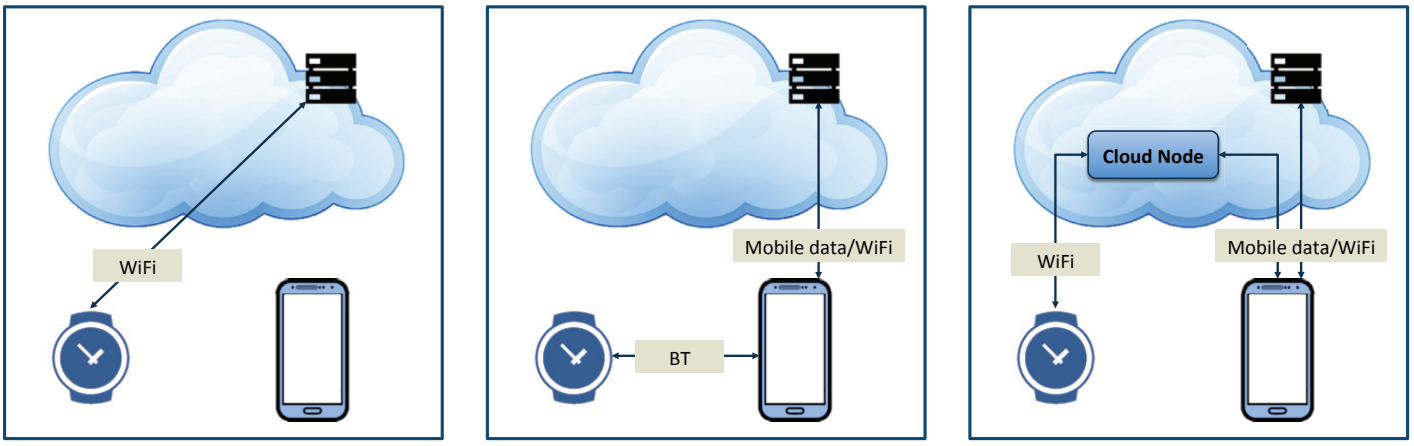
Recent advances in wearable technology tend towards standalone wearables. Most of today's wearable devices and applications still rely on a paired smartphone for secure Internet communication, even though many current generation wearables are equipped with Wi-Fi and 3G/4G network interfaces that provide direct Internet access. Yet it is not clear if such communication can be efficiently and securely supported through existing protocols. Our findings show that it is possible to use secure and efficient direct communication between wearables and the Internet.



Photo, istockphoto.com

Wearables such as smartwatches and smartglasses are becoming increasingly popular as they show promise in providing a number of attractive and useful services [1]. Wearables are equipped with sophisticated sensors and connectivity options that are capable of collecting personal data to support personalized services, such as fitness and health apps. More often than not, this information is transferred to cloud servers for processing, storage, and visualization. Furthermore, there are many popular apps that transfer data from cloud servers to wearables to provide real-time notifications. Due to the sensitivity of the transferred personal data, secure communication protocols must be used. Our study on most popular wearable apps shows that the majority of today's wearable apps rely on a paired smartphone via Bluetooth to provide secure Internet communication despite direct Internet connection capability. Therefore, this article attempts to answer the critical question – "Are wearables ready for secure and direct Internet communication?" – by means of real experiments with modern wearables.

In this paper, we first analyze the Internet communication methods, protocols, amount of data, and communication frequencies for the most popular smartwatch apps to profile current wearable



(a) Direct communication.

(b) Relaying through the paired smartphone via Bluetooth.

(c) Relaying through a cloud server (Wear Cloud) and the paired smartphone via Wi-Fi.

FIGURE 1. Methods of Internet communication for smartwatch apps.

app-generated traffic. The results show that smartwatches are not yet considered as standalone devices, for the following possible reasons: 1) enabling secure communication from the wearables is considered a significant burden due to the lower capabilities of wearables compared to a smartphone; and/or 2) relaying through a smartphone is considered more efficient (in terms of energy and time) than direct Internet communication. Therefore, we investigate the impact of these two factors on wearables in comparison to a smartphone through a series of experiments. The experimental results show that secure, standalone apps and platforms can be practically and efficiently realized on current wearables and, thus, we provide sufficient evidence to app developers to motivate them to create standalone apps and platforms in the foreseeable future.

INTERNET COMMUNICATION FROM SMARTWATCHES

Measurement methodology: We used an LG Urbane smartwatch running Android Wear and a Nexus 4 smartphone running Android 5.1.1. We considered the 10 most popular Android wear apps in Google Play.[1] Seven apps out of the 10 *(Google Fit, Google Translate, Reminders, Google Keep, Hangouts, Weather, Google)* have been already installed as default apps, and the next three apps *(To-Do List, Shazam, Runtastic)* were downloaded from Google Play. To gain insight on communications, we run *tcpdump* and *Bluetooth HCI Snoop log* on each device. We mimicked typical usage of these apps and captured incoming and outgoing data from both the smartphone and smartwatch for two scenarios; i) turning-off Bluetooth while turning-on Wi-Fi on the smartwatch, and ii) turning-on Bluetooth on both the smartwatch and smartphone (Wi-Fi of the wearable is turned off automatically). Smartphone Wi-Fi is turned on at all times. We investigated the apps' Internet communication methods and protocols in both of the scenarios. We also measured the amount of data and frequency of Internet communication for the most popular communication method.

Communication modes of Android wearables

Current smartwatch apps leverage three modes of Internet communications supported by Android; *Method-1:* Direct Communication; *Method-2:* Relaying through the paired smartphone via Bluetooth; *Method-3:* Relaying through a cloud server (Wear Cloud) and the paired smartphone via Wi-Fi (see Figure 1). In *Method-1*, the smartwatch directly communicates with the targeted servers via WiFi or cellular data. In *Method-2*, the smartwatch communicates with the smartphone counterpart of the same app via Bluetooth and relies on the smartphone app for any external communication. *Method-3* is used as long as both devices have Internet connectivity. Instead of communicating directly with the end servers, the smartwatch is connected to a trusted cloud server, Wear Cloud, provided by Google [2]. Wear Cloud relays all requests from wearables to the respective smartphone counterpart apps, and then the smartphone communicates with the targeted server and relays data back to the wearable through Wear Cloud.

We observed that the default communication mode of choice for all analyzed apps is *Method-2*, i.e. smartphone relaying, irrespective of the availability of a direct Internet connectivity. When Bluetooth of the wearable is turned-off, only 2 apps *(Google Fit, Google Translate)* were capable of communicating directly with the servers using *Method-1*. The remaining 8 apps attempted to leverage *Method-3*, which is subjected to the Internet connectivity of the paired smartphone. When the paired smartphone is not connected to Internet, these eight apps do not support functionalities that require Internet connectivity. This is a significant drawback of the current wearable eco-system, which essentially constrains the potential of wearable devices. In the case of Android,

[1] https://play.google.com/store/apps/category/ANDROID_WEAR?hl=en

TABLE 1. Data sizes, data transfer direction and communication frequencies in Internet communication

App Name	Size of the transferred application data from and to servers	Direction of data transfer	Communication frequency
Runtastic	1 KB	Upload	Once per every 90 seconds
Google Fit	1 KB	Upload	Once per every 10 minutes
Google	10 KB	Download	As requested
Shazam	16 KB	Upload/Download	
To-Do List	20 KB	Upload	
Google Keep	26 KB	Upload	
Google Translate	44 KB	Upload/Download	
Hangouts	50 KB	Upload/Download	
Weather	50 KB	Download	
Reminders	112KB	Upload/Download	

the developer documentation encourages developers to use *Wearable Data Layer APIs*, but these APIs do not support direct communication, and this hiding the true communication mode from the developer. If a developer wants an app to communicate with the Internet directly, the developer should use basic APIs.

In the relayed communication (*Method-2* and *Method-3*) the counterpart app for the smartphone behaves in different ways for different apps. The counterpart app either displays statistics, does some processing to the data before transferring, or just relays data to/from Internet servers. As examples, *Google Fit* and *Runtastic* in both devices measure, display and exchange data before Internet communication. Also, *Google Translate*, *Reminders*, and *Google Keep* process data received from the smartwatch to perform voice recognition. Moreover, *Reminders*, *Google Keep*, *Hangouts*, *To-Do List*, and *Shazam* store the statistics and display the data received before sending back to the smartwatch. On the other hand, *Google* and *Weather* apps are using the smartphone just as a relay device.

Communication protocols

We also investigated the utilization of communication protocols in wearable-Internet communication. Several protocols are available for the transfer of web objects. HTTP is the standard original Web Traffic protocol. It does not provide end-to-end security. Security is provided by the HTTPS protocol [3]. Recently Google has released versions of the QUIC (Quick UDP Internet Connections) protocol, which focuses on low latency for connections and transport data, and was designed to provide TLS/SSL equivalent security [4]. Although the smartphone's connections with the Internet use HTTPS, HTTP2 and QUIC protocols, all the direct connections from the smartwatch only use HTTPS.

At the data link layer, the smartwatch's connections with the smartphone are established via Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) in all the apps, referred to as Bluetooth Classic (BT-classic). Surprisingly, no apps utilize Bluetooth Low Energy (BLE) for such communication, although it is intuitive to use a protocol that consumes low energy for wearables [5]. We will investigate the consequences of using BLE and BT-Classic in the remainder of this article.

For a deeper understanding of current wearable traffic, we then investigated the amount of data and frequency of communication from the smartwatch apps, while the app is running in the foreground. Table 1 shows the total number of bytes transferred from and to each app, and the frequency of the communication. We selected *Method-2* as it is common for all 10 apps and measured the total amount of application layer data transmitted between the server and smartphone. The two fitness-tracking applications (*Google Fit* and *Runtastic*) transfer data in the range of 1 KB. However, these connections are periodic (i.e., *Google Fit* transfer in every 10 minutes, *Runtastic* transfer in every 90 seconds) and eventually transfer a larger amount of data. All the other apps communicate on demand and transfer more than 10 KB of data for each connection. *Reminders* app transfers the highest amount of data, which is more than 100 KB.

Not all of the wearable apps are designed to collect data and upload to the Internet. There are apps that are purely designed to download data from the Internet, such as *Weather* and *Google*. This downloaded data is not required to be synced with the smartphone. Also, *Google Translate*, *Reminders*, *Google Keep*, *Google Fit* and *Hangouts* apps perform both uploads and downloads in certain cases. However, these apps are linked with the user's Google account; therefore, if the smartphone is connected to the Internet, the data can be retrieved by synchronizing with the Google account. Similarly, each user has their own account for *To-Do List*, *Shazam* and *Runtastic* apps that allows data to be synchronized if the user logs into their account from the smartphone. Computation for other apps, such as *Google Translate*, *Reminders*, and *Google Keep* can also be moved to the cloud and bypass the smartphone. Thus, all the analyzed apps can take advantage from direct and secure Internet connectivity rather than always relaying through the smartphone.

However, leveraging the smartphone for Internet communication is the chosen method for most current apps, despite not being a functional requirement. The observed traffic profiles of wearable apps in terms of transmitted amount of data, frequency of communication and connection duration make a strong case for utilizing direct, secure connectivity. Thus, we experimentally evaluate the practical feasibility of utilizing direct and secure Internet communication on wearables in the remainder of this article.

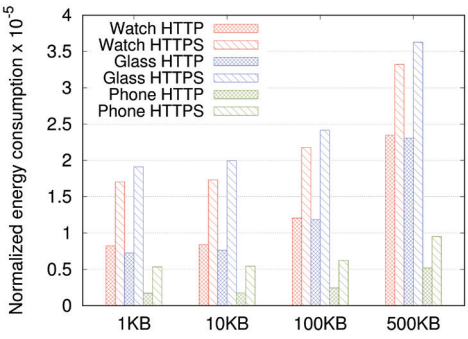
SECURITY-PERFORMANCE TRADE-OFF OF DIRECT COMMUNICATION

In this section, we analyze the cost of enabling secure Internet direct communication with a smartwatch and a smartglass as they represent two most advanced wearable categories of devices. Specifically, we measure energy, time and downloaded data overhead associated with HTTPS in a controlled experimental environment and compare it with HTTP and a smartphone as baselines.

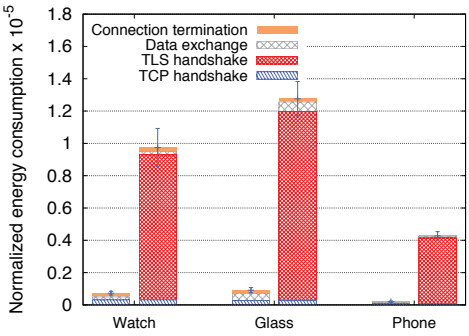
Measurement methodology: We added a Google Glass Explorer edition running Android XE 18.11 to the previous hardware/software configurations. We developed an Android app for all three platforms that simply fetches objects from a given address using either HTTP or HTTPS (*Method-1*). We downloaded objects of multiple sizes (1 KB, 10 KB, 100 KB and 500 KB) from an Apache web server connected to the local network with end-to-end security (HTTPS) and without security (HTTP). To measure the transfer times and downloaded bytes, we run a *tcpdump* on each device. Energy consumption is measured with a Monsoon power monitor[2] directly connected to each device via USB. Due to the significant difference in battery capacity among the compared devices, the absolute energy consumption is normalized by the battery capacity.

Figure 2(a) depicts the normalized energy consumption for downloading web objects. Energy consumption increases with object size in all cases due to the longer transfer duration, but the overhead of HTTPS is more apparent for smaller object sizes. Normalized energy consumption for both wearables is significantly greater than for the smartphone. In terms of downloaded data, there is an increment of 5421 bytes during the TLS handshake phase, in addition to the ~0.2% increment in payload due to encryption. These observations are common for all the devices as long as the same TLS parameters are used. However, the ratio between HTTPS/HTTP for wearables is comparable with the smartphone. A comprehensive analysis of the impact of secure communication on wearables is presented in our earlier publication [6].

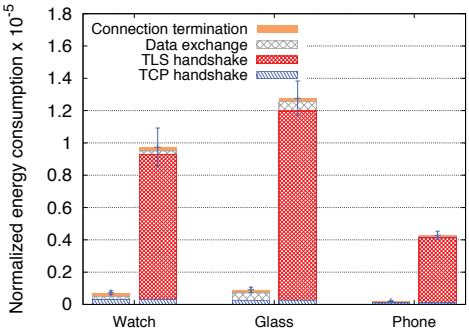
[2] https://www.msoon.com/LabEquipment/PowerMonitor



(a) Normalized energy consumption for different file sizes.



(b) Energy consumed: 1 KB download.



(c) Data transfer time: 1 KB download.

FIGURE 2. Cost of handling security.

As the sizes of the majority of the objects transferred between wearables and the Internet are at least a few KBs as shown earlier, we further quantify the energy consumption for each phase of the communication for 1KB object size. Figure 2(b) and 2(c) illustrate that the majority of both energy and time is consumed during the TLS handshake. Due to the key generation process, the time consumption and energy footprint of the TLS handshake is significant in comparison to the amount of data generated in all three devices. Moreover, the processing overhead of encrypting data increases the data transfer time and hence the energy consumption. Although secure communication is more energy expensive, the next section will show that it is still more energy efficient than leveraging the smartphone as a relay.

Further, we investigate the benefits of using the modern HTTP extensions, like HTTP2, which enables long-lived connections and amortize the cost of re-establishing the TLS connections, as the TLS handshake overhead is more significant for smaller data transfers. However, the long-lived connections consume additional energy during the idle connection time because of the elevated power state of devices. Figure 3 illustrates the analyses of energy saving offered by keeping the connection alive for different traffic patterns and device types, which is based on measured energy consumption of different phases of connection and periodic 1 KB downloads. Long-lived connections are beneficial only when idle energy consumption is relatively lower and the time interval between two transfers are smaller (color shades of the figure show the percentage of energy saving). For a typical Wi-Fi idle power consumption of 200mW, long-lived connections are beneficial only if the time interval is less than 500 ms for a 1 KB data transfer. The time interval and idle energy values that are above the 0% curve will not benefit from long-lived connections. Therefore, a typical app will not benefit from keeping a connection alive as per the observed traffic profiles in the previous section.

COST OF DIRECT VS. RELAYED COMMUNICATION

In this section, we evaluate the relative time and energy efficiency of leveraging direct Internet communication. Relayed communication can be established via BT (i.e. BT-Classic, BLE) or Wi-Fi as shown in Figure 1. Wi-Fi relaying, as in *Method-3*, incurs significant extra delays and energy consumption compared to the other methods as it requires two communications with the Internet (i.e., smartwatch to smartphone via Wear Cloud and Smartphone third party server). Therefore, Wi-Fi relaying is not considered for this analysis.

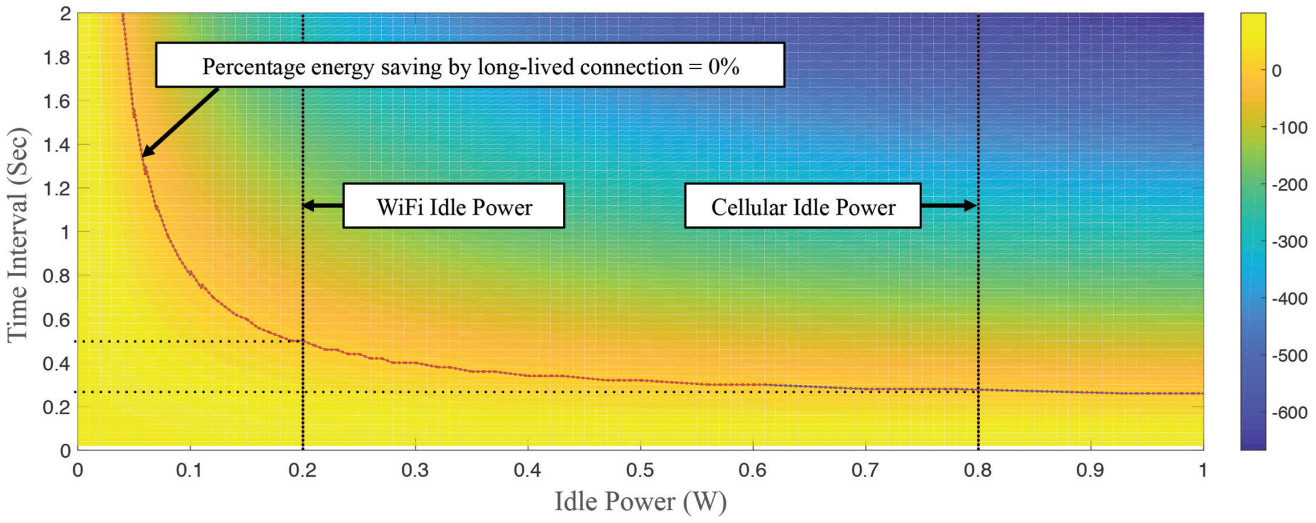
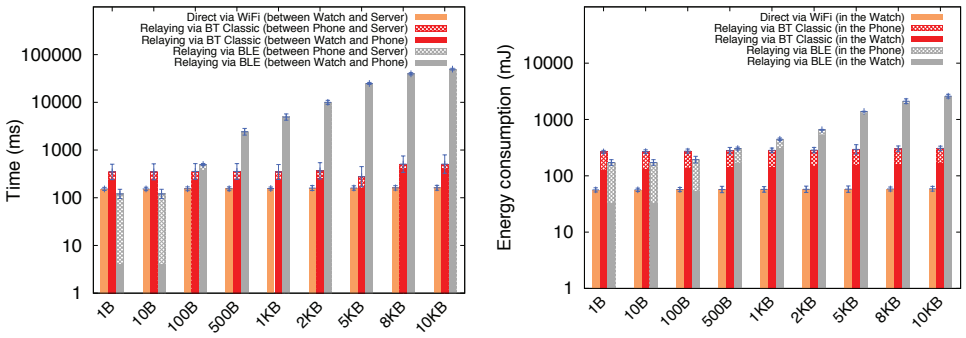


FIGURE 3. Percentage energy saving by long-lived connections.



(a) Time consumption of relayed and direct communication.

(b) Energy consumption of relayed and direct communication.

FIGURE 4. Cost of direct vs. relayed communication.

Measurement methodology: We only use the LG Urbane watch running Android Wear that runs *tcpdump* and *Bluetooth HCI Snoop* in the device. Energy consumption is measured with a Monsoon power monitor directly connected via USB. In direct communication, the smartwatch app sends HTTPS requests directly to a local server via Wi-Fi and downloads the requested object. In relayed communication, the smartwatch app sends download requests to the companion app on the smartphone via Bluetooth and then the smartphone sends HTTPS requests to the server, downloads the requested object and instantly uploads to the smartwatch. Although BLE is not used in any of the apps considered in the previous experiments, it is a natural candidate for relayed communication for *Method-2*. Therefore, we conducted the above measurements using both BLE and BT-Classic. We chose HTTPS in direct connections, as it is the only protocol used in direct Internet communications from wearables. In Android BLE development, we used *BluetoothGatt* and *BluetoothGattCallback* classes in order to connect to a BLE-enabled device. In direct communication, we measured the energy consumption in the smartwatch and time taken from smartwatch to the third-party server. In relayed communication, we measured energy consumption in both smartwatch and smartphone, and time usage from smartwatch to smartphone and from smartphone to the third-party server.

Figure 4(a) shows the time taken for all the three methods (note the log-y axis). The total time for the smartwatch to download the object by relaying via BLE is less for smaller file sizes (1 byte to 10 bytes) as the smartphone downloads the objects faster than the smartwatch and BLE also assures the least connection establishment time compared to BT-Classic [7]. However, since BLE allows for short bursts (20 bytes) and is not designed for continuous connections, larger objects are transmitted in smaller chunks [5]. Therefore, completion time increases drastically with object size for BLE due to the lower effective data rate. Direct Wi-Fi communication took the least time (including Wi-Fi association time) for all object sizes larger than 100 bytes.

Relaying via BLE is the most energy-efficient method in the smartwatch, if the object size is smaller than 100 bytes as shown in Figure 4(b). In contrast, energy consumption for direct communication is larger for smaller object sizes in the smartwatch, due to the costly TLS handshake phase in HTTPS. However, when the file size increases, BLE becomes the worst mode among the three methods, because the lower data rate in BLE has a greater effect than connection establishment

cost in BT-Classic and TLS handshake phase in HTTPS. Therefore, direct Internet communication via Wi-Fi is the best method in terms of energy consumption in the smartwatch for larger file sizes. However, the cumulative energy consumption in the smartwatch and the smartphone is larger in BLE relaying than in direct communication for all the sizes.

All the considered applications in the previous section require 1 KB or larger files transmission. In this case, Wi-Fi provides the minimal power/throughput ratio, and is thus more power efficient than BT-Classic and BLE relaying. We have shown that the direct Internet communication is much more efficient than any other relaying method for object sizes currently used in wearable apps.

CONCLUDING REMARKS

Secure, direct Internet Wi-Fi connectivity on wearables is practically feasible. In fact, the direct Internet communication reduces energy consumption and improves performance compared to the current norm of Bluetooth relaying. Therefore, we advocate that wearable app/system developers should leverage these advanced capabilities of wearables to drive a paradigm shift toward standalone wearable devices. In particular, we provide the following recommendations:

- Relayed communication via smartphone is not imperative for functionality of many apps and, thus, wearable apps can directly communicate with the Internet.
- The relative cost of enabling secure direct communication on wearables is comparable to smartphones, despite the resource constraints on wearable devices. Thus, standalone secure wearable applications can be practically realized utilizing existing secure protocols such as HTTPS.
- Although TLS overhead is significant for smaller data transfers, keeping the connection live is only beneficial if the interval in between two transfers is less than 500 ms for modern wearables with Wi-Fi connectivity, which is almost a continuous stream of data.
- Bluetooth Low Energy (BLE) is not energy- and time-efficient for transferring 1 KB or larger files. As typical wearable data communications involve 1 KB or larger files, we do not recommend BLE be used for relayed Internet communication from wearables.
- Most current wearable apps leverage the smartphone for Internet connectivity, regardless of device capability. However, Bluetooth relaying increases both the energy consumption and data transfer time. Therefore, future wearables apps can be developed to leverage direct Wi-Fi connectivity whenever Wi-Fi connectivity is available. To realize this, we recommend that app developers utilize basic networking APIs rather than Wearable APIs (e.g., *HttpsURLConnection* class rather than Wearable *Data Layer APIs*) for Android Wear. ■

Harini Kolamunna is a PhD candidate at the School of Electrical Engineering and Telecommunications, UNSW Australia, and attached to Data61-CSIRO. She received a bachelor's degree in Electrical & Electronics Engineering with a First Class Honours from University of Peradeniya, and worked as a research assistant at National University of Singapore (2013-14). Her current research interests include wearable/IoT technologies, networking and communications.

Jagmohan Chauhan is a PhD candidate in Electrical Engineering and Telecommunications at UNSW, Australia. Before joining UNSW, he received a MS from University of Saskatchewan in 2013. His research interests include mobile systems, security and deep learning.

Yining Hu is a PhD candidate at UNSW, Australia, affiliated to Data61-CSIRO. She received a bachelor's degree in Engineering (Electrical) with First Class Honours from the University of Sydney, Australia, and Harbin Institute of Technology, China in 2016 (joint program). Her current research interests include data integrity preservation in wearable systems and blockchains.

Kanchana Thilakarathna is a Lecturer in Distributed Computing at the School of Information Technologies, The University of Sydney. Previously, he was a Research Scientist at Networks Group, Cyber-Physical Systems Research Program at Data61-CSIRO. He received his PhD in Electrical Engineering and Telecommunications from UNSW Australia. His research interests include developing technologies for resource allocation, secure communication and authentication, and privacy-preserving data sharing in wearable/mobile/IoT networks.

Diego Perino is a researcher at Telefonica Research in Barcelona, Spain. He received a PhD in Computer Science from the Paris Diderot-Paris 7 University, and MS from Politecnico di Torino and Eurecom institute. His research focuses on design and performance evaluation of networking protocols and systems. He has published several papers at international conferences and in journals, and also filed several patents.

Dwight Makaroff is a professor in the Computer Science Department at the University of Saskatchewan, Canada. He obtained his PhD at the University of British Columbia in 1998 in Multimedia Systems. His current research interests are distributed systems performance, including network protocols, sensor networks, big data architecture frameworks, wearable systems, and networked games.

Aruna Seneviratne is currently the research director of Cyber-Physical Systems Research Program in Data61-CSIRO and a professor at UNSW, Australia. He was the foundation professor of Telecommunications at UNSW, where he holds the Mahanakorn Chair of Telecommunications. He received his PhD in electrical engineering from the University of Bath, UK. His research interests are in mobile technologies, networking and communications, and computer system security.

REFERENCES

[1] T. Danova. "The wearables report: Growth trends, consumer attitudes, and why smartwatches will dominate," http://www.businessinsider.com.au/the-wearable-computing-market-report-2014-10 , October 2014.
[2] Android.com, "Sending and syncing data", https://developer.android.com/training/wearables/data-layer/index.html.
[3] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K.Papagiannaki, and P.Steenkiste, "The cost of the "S" in HTTPS", CoNEXT '14, Dec. 2014, pp. 133-140.
[4] J. Rosskind, "Multiplexed transport over UDP," Google White Paper, 2013.
[5] Bluetooth SIG, "Bluetooth core specification," https://www.bluetooth.com/specifications/bluetooth-core-specification.
[6] H. Kolamunna, J. Chauhan, Y. Hu, K. Thilakarathna, D. Perino, D. Makaroff and A. Seneviratne, "Are wearable devices ready for HTTPS? Measuring the cost of secure communication protocols on wearable devices," https://arxiv.org/abs/1608.04180, December 2016.
[7] Link Labs, "Bluetooth vs. bluetooth low energy: What's the difference?" https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy, November 2015.