Understanding the Networking Performance of Wear OS

Xiao Zhu
University of Michigan
shawnzhu@umich.com

Yihua Ethan Guo
Uber Technologies, Inc.
yhguo@umich.edu

Ashkan Nikravesh
University of Michigan
ashnik@umich.edu

Feng Qian
University of Minnesota
fengqian@umn.edu

Z. Morley Mao
University of Michigan
zmao@umich.edu

ABSTRACT

Networking on wearable devices such as smartwatches is becoming increasingly important as fueled by new hardware, OS support, and applications. In this work, we conduct a first in-depth investigation of the networking performance of Wear OS, one of the most popular OSes for wearables. Through carefully designed controlled experiments conducted in a cross-device, cross-protocol, and cross-layer manner, we identify serious performance issues of Wear OS regarding key aspects that distinguish wearable networking from smartphone networking: Bluetooth (BT) performance, smartphone proxying, network interface selection, and BT-WiFi handover. We pinpoint their root causes and quantify their impacts on network performance and application QoE. We further propose practical suggestions to improve wearable networking performance.

CCS CONCEPTS

• **Networks → Network protocols**; **Network performance evaluation**; • **Human-centered computing → Ubiquitous and mobile devices**.

KEYWORDS

Wearable Networking; Bluetooth; WiFi; Proxy; Interface Selection; Vertical Handover

ACM Reference Format:
Xiao Zhu, Yihua Ethan Guo, Ashkan Nikravesh, Feng Qian, and Z. Morley Mao. 2019. Understanding the Networking Performance of Wear OS. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '19 Abstracts), June 24–28, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3309697.3331475

Smart wearable devices are becoming increasingly popular. Take smartwatches, arguably the most important type of smart wearables, as an example. According to a market research report published recently [4], the global market value of smartwatches was estimated to be $10.2 billion in 2017 and will experience an annual growth rate of 22.3% from 2018 to 2023.

In the literature, several efforts have been made towards understanding the OS execution performance [7], power management [8], graphics and display [10], storage [6], and user interface [11] of wearable OSes. In this work, we investigate an important yet underexplored component: *the wearable networking stack.* We conduct to our knowledge a first in-depth investigation of the networking performance of Wear OS, one of the most popular OSes for wearables. Wear OS is a version of Google's Android OS tailored to small-screen wearable devices. Used by a wide range of smartwatches and potentially other wearables, Wear OS is expected to account for 41.8% of the market share of smartwatch OSes in 2020 [3].

Wearable networking is *important.* Take smartwatches as an example. One may argue they only incur light traffic such as push notifications. This might be true for the *current* smartwatch ecosystem where traffic flows are largely small, short, and bursty [8]. However, we envision that future wearable apps will be more network-intensive by incurring much heavier network activities as fueled by new hardware, OS support, and applications. For example, recently debuted speaker/LTE-capable watches such as LG Watch Urbane 2nd Edition allow users to directly make hands-free VoIP calls; the latest Wear OS 2.x allows standalone apps on wearables; also, many emerging wearable applications incur heavy network traffic such as continuous computer vision on smart glasses [5], remote camera preview [2], real-time screen projection [1], and network-level collaboration between phone and watch [9].

Wearable networking is also *different* from smartphone networking that has been well studied in the past decade. First, wearables oftentimes do not directly access the Internet; instead, it uses its paired smartphone as a "gateway", which, if not carefully designed, may incur additional performance degradation. Such a gateway mode accounts for 84% of the daytime usage period as measured by a recent user study [8]. Second, the communication between a wearable and the phone is usually through Bluetooth (BT) or Bluetooth Low Energy (BLE), whose characteristics are vastly different from WiFi and cellular that dominate the smartphone interface usage; also the cross-layer interaction between BT and upper-layer protocols such as TCP remains underexplored. Last but not least, due to BT's short range, network handovers frequently occur on a wearable: when it moves away from the phone, the BT connectivity will be torn down and the wearable has to use standalone WiFi or LTE to communicate with the external world.

Understanding the networking performance of commercial wearables is challenging, as it involves multiple devices, networks, and protocols, which incur complex interactions. The proprietary nature of Wear OS makes it even harder to gain deep visibility into the

Table 1: Study summary: examined aspects, methodologies, key findings, root causes, and recommendations.

Aspect	Methodologies	Key findings	Root causes of inefficiency	Recommendations
BT radio resource management	Cross-device state machine inference.	Different state machine models on phone and wearable; BT download experiences frequent "blackout" periods.	Undesired BT sniff (sleep) mode is triggered during a continuous download.	BT state machine should be aware of bidirectional traffic and judiciously determine when to enter the sniff mode.
Proxying at paired smartphone	Breakdown analysis of end-to-end latency.	Phone-side bufferbloat inflates the end-to-end latency to tens of seconds; real-time apps' performance is severely affected.	Multiple buffers at different layers on the smartphone proxy, in particular the large TCP receive buffer, cause the high buffering delay.	The proxy should be aware of the path asymmetry between server-phone and phone-wearable paths.
Network interface selection	Quantify energy-performance tradeoffs using real wearable apps; develop a multipath framework for wearables.	Wear OS's default interface selection policy is often suboptimal; BT+WiFi multipath brings limited performance gain.	Static interface preference is not aware of apps' QoE requirements; BT and 2.4 GHz WiFi cause interference.	Need QoE-aware interface selection; need 5GHz WiFi support on wearables; need BT-aware multipath scheduler.
Network handover	Conduct an IRB-approved user study; examine each phase of a handover process.	Frequent BT-WiFi handovers in the wild; short BT range on commodity wearables; BT-WiFi handovers may take 60+ seconds.	Handovers are performed reactively; both OS and app contribute to high handover delay; lack protocol/OS support for handover.	Proactively predict a handover; need multipath support to facilitate seamless handovers.

wearable networking stack. Note that unlike Android for handheld devices, Wear OS is not open-source.

To address these challenges, we first build a wearable networking testbed consisting of commodity Wear OS based smartwatches, off-the-shelf smartphones, commercial wearable apps, as well as a series of tools we developed for instrumenting the system and collecting various types of data. We then leverage the testbed to conduct controlled experiments in a cross-device, cross-protocol, and cross-layer manner. Through judiciously designed experiments, we demystify the Wear OS networking stack and quantify how it affects the wearable networking performance. Our key findings consist of *several serious performance issues* regarding *all* three aforementioned aspects that distinguish wearable networking from smartphone networking.

• We perform a comprehensive analysis of the BT radio state machine on both the wearable and its paired phone. We find tricky yet critical differences of the state machine behaviors between the two sides. They lead to our key discovery that due to the wearable's unique BT radio management policy and its interplay with its counterpart on the phone, a download session on the wearable frequently (*e.g.,* every few seconds) experiences "blackout" periods lasting for about 1 second.

• When acting as a gateway proxy for a wearable, the phone dramatically inflates the end-to-end (server to wearable) latency to 30+ seconds due to its incurred "bufferbloat". We then break down the end-to-end latency into various components, and identify the root cause to be the phone-side TCP receive buffer, whose configuration does not take into account the path asymmetry between the wearable-phone path and the phone-server path.

• Wearables are equipped with multiple network interfaces such as BT and WiFi. When multiple networks are available, the Wear OS's default interface selection policy strictly prefers one interface (*e.g.,* BT) over others (*e.g.,* WiFi). However, we find that such a strategy oftentimes leads to suboptimal tradeoffs between performance and energy consumption. In addition, we explore the feasibility of performing multipath transport on wearables, and identify potential obstacles such as the interference between BT and 2.4GHz WiFi.

• BT's short communication range makes handovers occur frequently on wearables. Due to insufficient protocol support and poor cross-layer coordination, a BT-WiFi handover may last for more than 60 seconds, leading to significant disruption of the wearable application performance. By looking into each phase of a handover, we find that both the OS and user application are responsible for such unacceptably long handover delays.

The above performance inefficiencies are caused by the poorly designed networking stack of Wear OS. Our identified issues appear on all 8 wearables of heterogeneous vendors and Wear OS versions as well as a variety of paired phones as tested by us using synthetic and real apps. To mitigate the identified performance impairment, we design, implement, and evaluate several readily deployable mitigation solutions including the following. (1) We develop a lightweight module that completely eliminates the undesired behavior of the wearable's BT radio state machine, (2) We develop a simple yet effective flow control scheme that mitigates the phone-side bufferbloat problem, (3) We design and implement to our knowledge a first multipath transport framework for wearable devices that enables adaptive interface selection, multi-network bandwidth aggregation, and smooth handovers between IP and non-IP networks.

Table 1 is a summary of our empirical study. Our paper with all technical details can be found at [12].

REFERENCES

[1] 2017. Cicret Bracelet. https://cicret.com/wordpress/.

[2] 2017. ZenWatch Remote Camera. https://play.google.com/store/apps/details?id=com.asus.rcamera2.

[3] 2018. Market share of smart wristwear shipments worldwide by operating system from 2015 to 2020. https://www.statista.com/statistics/466563/share-of-smart-wristwear-shipments-by-operating-system-worldwide/.

[4] 2018. Smartwatch Market Size, Share, Growth, Industry Report, 2018–2023. https://www.psmarketresearch.com/market-analysis/smartwatch-market.

[5] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2014. Towards wearable cognitive assistance. In *MobiSys.* ACM.

[6] Jian Huang, Anirudh Badam, Ranveer Chandra, and Edmund B. Nightingale. 2015. WearDrive: Fast and Energy-Efficient Storage for Wearables. In *USENIX ATC.*

[7] Renju Liu and Felix Xiaozhu Lin. 2016. Understanding the Characteristics of Android Wear OS. In *MobiSys.* ACM.

[8] Xing Liu, Tianyu Chen, Feng Qian, Zhixiu Guo, Felix Xiaozhu Lin, Xiaofeng Wang, and Kai . Chen. 2017. Characterizing Smartwatch Usage in the Wild. In *MobiSys.* ACM.

[9] Xing Liu, Yunsheng Yao, and Feng Qian. 2017. Rethink Phone-Wearable Collaboration From the Networking Perspective. In *ACM WearSys.*

[10] Hongyu Miao and Felix Xiaozhu Lin. 2016. Tell Your Graphics Stack That the Display Is Circular. In *HotMobile.*

[11] Jian Xu, Qingqing Cao, Aditya Prakash, Aruna Balasubramanian, and Donald E. Porter. 2017. UIWear: Easily Adapting User Interfaces for Wearable Devices. In *ACM MobiCom.*

[12] Xiao Zhu, Yihua Ethan Guo, Ashkan Nikravesh, Feng Qian, and Z Morley Mao. 2019. Understanding the Networking Performance of Wear OS. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 1 (2019), 3.