Wearable Devices: Smartwatch, Fitness Tracker and Call Notifications Delay

Taniza Sultana
tsultana@uccs.edu
University of Colorado Colorado Springs
Colorado Springs, Colorado

ABSTRACT

Internet of Things (IoT) opened a new technological world and connected the digital world with the physical in our fingertips. IoT digital communication is making it possible to have smart wearable devices such as smartwatches and fitness trackers to communicate with their connected smartphones to provide real-time data and notification alerts even when the user is not at close proximity with their smartphone. Receiving notifications with wearable smart devices is a very popular feature and it is important that users receive notifications in real-time. In some cases, there is a significant delay between the smartphone generating notifications that are pushed the to smartwatches/fitness trackers receiving them. This paper focuses on call notification delays in smartwatches/fitness trackers and identifies the device(s) with delays in call notifications for further evaluation and analysis. During the research study, we manually tested 32 sets of smartphones and connected smartwatches/fitness trackers and collected the data for evaluation. Some devices display the delay notification behavior; most newer devices performed the push notification almost immediately after initiating calls. Based on the results we determined the notification push delay is more common in older versions of Android phones than the newer versions. The research also examines the separating factors in network architectures, system and software designs between the devices with the delay behavior in call notifications and the high performing devices. As a result, we are able to identify and present many common vectors that influences call "notifications push" which can be easily overlooked.

KEYWORDS

Wearable devices, smartwatch, smartphone notifications, delay tolerant network, push notifications

1 INTRODUCTION

Wearable smart technology is an astonishing creation of the Internet of things (IoT). The popularity of wearable deceives such as smartwatches, smart glasses, and fitness trackers has grown significantly over the past few years. It is estimated that the global smartwatch market will reach $32.9 billion by 2020 [2]. To up-keep with demand and popularity, more and more technological capability and functionality have been also introduced to these devices over the years. In our modern technological era, IoT and wearable devices connecting the physical world to our fingertips. Network connectivity is vital for these devices since they are considered as low resource device due to their physical and memory size [10]. Some wearable devices maintain their connectivity via Wi-fi or cellular signals while others use their Bluetooth connection capability by paring with a smartphone or other smart devices (i.e. tablet, iPad etc.).

One of the popular and most commonly used wearable devices is smartwatch and fitness tracker. Wearable smartwatches and fitness trackers are full of various applications such as activity trackers, heart rate monitoring, sleep tracker, GPS, and even many other health applications. One of their smart functionalities is their communication and notification capability such as receiving and displaying text and call notifications from connected smartphones on the smartwatch, also known as *"smartphone notifications"* or *"push notifications"* [5]. From the user's prospective, smartphone notifications (i.e. call notifications alert) make smartwatches and other fitness trackers with similar functionality unique. Notifications capability allows them to receive texts or incoming call alerts in their connected smartphone while they are not at close proximity with their smartphone, and within the maximum range of the connected smartwatch or fitness tracker. Given that these devices have their limitation with the Bluetooth connectivity range, physical proximity plays a role in whether it will receive smartphone notifications or not. But smartphone notifications should not be affected when both devices are within the range or at close proximity. However, the two initial observed case study suggested differently.

Observed Case Study 1. *A Fitbit Charge 3 (fitness tracker with smartphone notification capability) connected to an Android device (Samsung Galaxy S7) had several seconds delay in smartphone notifications at close proximity.*

Observed Case Study 2. *A Fitbit Alta HR (fitness tracker with smartphone notification capability) connected to an Android device (Samsung Galaxy S7) had several seconds delay in smartphone notifications at close proximity but less time delay comparing to Fitbit Charge 3.*

In both cases, connected smartphones were the same model and in the same cellular network, while two different models of fitness trackers were connected to these smartphones. In both cases, some levels of delays in call notifications were observed. We conducted a case study among 30 additional sets of Android phones and their connected smartwatches.

Delay in receiving notification may defeat the purpose of the notifications alert function in smartwatch and fitness tracker. Users prefer this function so that they can continue receiving messages and call alerts while not at close proximity with their smartphone. Call alerts are more critical compared to other messaging alerts especially if its a time sensitive call (i.e. an emergency call from either family member or work related requiring immediate attention). Receiving delayed call alerts while away from the smartphone will affect users how soon they can reach to their smartphone to answer a call and most cases might require them to call back instead

of answering promptly. Therefore, the current overall goal of this research study is to identify a device (smartwatch or fitness tracker) with the least amount of delay in call notifications.
Smartwatch sensor technology and notification capability makes it ideal for health monitoring and tracking [12]. Various health monitoring apps are available in the market for use and many more under are development. Since smartwatch technology is capable of linking to a smartphone or even can function as a standalone device, it can easily monitor the users health through its sensors and generate notifications alert based on its set variables [2]. In many cases, smartwatch technology such as HealthPatch MD, Xio XT can even generate alerts for healthcare providers. If there is a push notification delay in smartwatch's call notifications (alert) process, there is a tangible chance it will affect the medical alert notifications push as well. [7].

2 BACKGROUND WORK

The smartphone push notification process starts with the device trying to gain user's attention through their paired wearable device and its preferred settings (i.e. vibrations, sounds or visual) [11]. With a real-time push notifications the smartphone system continuously monitors streams to alert the users. Real-time push notifications usually alerts the users immediately after a notification is generated in their mobile device [8].

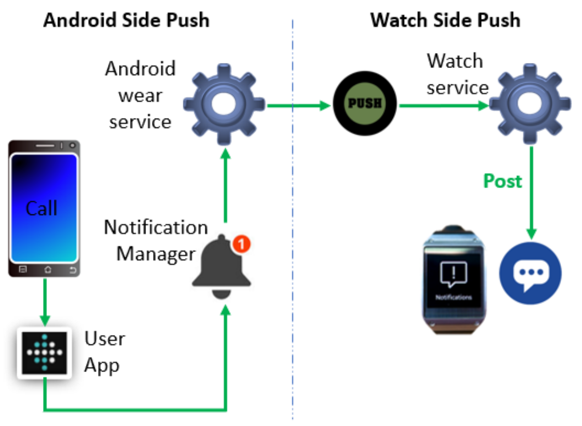

Figure 1: One-way communication between smartphone and smartwatch (adapted from [5])

But for their connected smartwatches/fitness trackers, the push notifications are usually not performed at same speed even if it follows same concepts. In these cases, push notification may take several seconds before waking up the connected smartwatches and fitness trackers to perform the push notification. And this is mostly due to its embedded delay-tolerant network system.
Liu et al. [5] analyzes and explains communications between the phone and paired watch in detail. As shown in figure 1 with green arrows, when a notification is available in the phone, the apps in the Android side (i.e. Fitbit app) wake up notification manager and post a notification, then the Android wear service will push the notification to watch-side service. Once the smartwatch service receives notification watch-side app will post the notification. This process is more common to one way communication from smartphones to smartwatches.

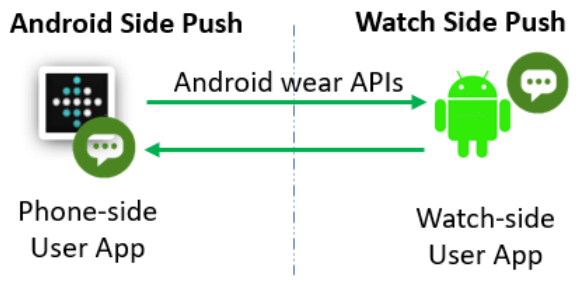

Figure 2: Two-way communication between smartphone and smartwatch (adapted from [5])

To perform two-way communication, the smartwatch is usually linked with a watch phone app and they communicate back and forth by "exchanging arbitrary data". As shown in figure 2, upon receiving notification in watch-side Android wear service, it pushes the notification to watch-side service, which wakes up the watch. Watch-side app either post a notification or take other actions based on the watch service and app settings. A similar process takes place when the watch-side user app takes action and communicates back to Android wear service. In that case, the watch-side app service will determine the push response and any return actions back to phone app service [5].
Most smartphones are designed to push notifications as soon available to minimize latency. However, Liu et al. [6] identifies that when a notification is available, phone-side user apps push to Android wear service policy to determine the notifications push. Many Android apps push notifications to smartwatch only when other phone-sides apps are not running. When other apps are running in the phone, the service policy assumes that user interacting with their device and able to see the notifications, therefore, no push is necessary. This service policy is not entirely ideal and can cause unnecessary latency in the push notifications process or even skipped notifications because some cases a user may leave an app running in their phone but not at close proximity to see an incoming call or message.

3 RELATED WORK

There is not a lot of research available specific to smartwatch's and fitness tracker's call notifications delay. However, there are some study discusses the overall push notification process and the delay-tolerant network architectures for smartphones and wearable mobile devices. Surveying both push notifications process and delay tolerant network will contribute to defining the overall process starting from smartphone pushing notifications to smartwatch/fitness tracker and receiving notifications in these wearable devices.

3.1 Push Notification

Push notification is a very important application embedded in smartwatches/fitness trackers which allows paired smartphones to communicate with its connected wearable device and send notifications.

In a publication Liu et al. [5] identifies that 200+ apps utilizes push notifications and exhibits a *"bursty arrival patterns"* and they are mostly dominated by instant messaging and emails notifications. They identify that 84% of the time the smartwatches were paired with their connected smartphones while the network traffic flows were short, small, and slow. During that time Bluetooth traffic was noted to be 91% of the overall traffic. In their publication they also characterizes the push notification process in detail.
In Android devices, Google pushes various notifications on regular basis such as traffic updates, weather alerts, news feeds etc, while other apps such as instant messaging, text, call, social media are also performing pushes. In their study, 230 apps were observed to create pushes while 190 of them didn't have watch-side version of the user apps. Despite all of that, they identify *"the median inter-arrival time of notifications across all users is only 49 seconds"* indicating a short-term expected push notifications [5].

3.2 Delay Tolerant Network

Energy savings and increased battery life certainly plays a big role in development of new capability and functionality in connected wearable devices. For the same reason, most notifications pushes are delay-tolerant in wearable devices [5][12]. Delay-tolerant addresses *"the lack of continuous network connectivity"*[3]. Benhamida et al.[3] conducts a survey to look into the use of Delay Tolerant Network (DTN) in *"IoT applications to overcome connectivity problems"* with the purpose to identify *"most recent solution that enables delay tolerant IoT"*. In the survey, they identify two main characteristics for DTN that effect IoT scalability are the "delivery latency" and "network coverage". Delivery latency measures the duration between a message being generated and its delivery, it also captures efficiency of the routine path [3]. In another publication Li et al. discuss that DTN performance relays on two key factors that are the routine and forwarding algorithms and their design compatibility with mobility patterns [4]. The Mobile Delay Tolerant Network (MDTN) are used to establish communication when there is a lack of infrastructure. To improve the data accessibility and efficiency, the users tend to utilize cooperative caching schemes, however, in the MDTN contact pattern remains a challenge [9].

3.3 Bluetooth Low Energy

To save energy and battery power, many IoT and wearable devices rely on *"low-power wireless communication"* such as Bluetooth Low Energy (BLE). Almost all smartwatches use BLE communication which makes them *"detrimental to the growth and usefulness of this class of devices"* [13]. BLE communication links one wearable device to only one smartphone which results in a smartwatch to lose its ability to receive notifications while the linked smartphone is either turned off or outside of its connectivity range [13].

3.4 Bluetooth Range

Wearable connected devices mostly communicate via Bluetooth, BLE or built-in Wi-fi [5][6]. Some cases the Bluetooth range (BT-range) are intentionally kept short to save energy, as a results many device frequently handover between Bluetooth and Wi-fi repetitively. With Android devices, the handover can take long time to finish, causing a delay in their push notifications [6]. When they are both connected to Wi-fi, they usually communicate through their designated servers (i.e. Google's servers).

4 METHODOLOGY

Connected mobile device's (such as smartwatches) embedded system testing is a very complex process. Many factors need checks and balances that can compromise the test integrity if proper methods and consistency with the testing are not taken into consideration. Since most smartwatches and fitness trackers are connected to their smartphones via Bluetooth, it will be very important factor in this study to ensure the smartphone and Bluetooth settings are the same for all testing cases.
There are many tools and platforms available for smartwatch/fitness tracker system testing. They are mostly designed to test sensor accuracy, connectivity, battery life, and even how fast devices can receive data from an connected smartphone. Finding an appropriate tool that can support testing how quick a device receives call notifications still remains as a challenge. The appropriate tool must support the test case of sending all push notification types from the smartphone to its paired smartwatch/fitness tracker and the push should be visible on the smartwatch/fitness tracker to record any time delay. Without customization it will be difficult to find a testing tool that can serve this research purpose. Developing a tool, instead of using an existing tool, creates a challenge due to time constraints.

4.1 Research Approach and Matrix

Many tools are available for smartphone testing, most of them are not designed for the smartwatch and unable to collect data specific to smartwatch call notifications. It would be ideal to build our own tool for testing, but time constraints played a constrictive role. Therefore, an alternative method was followed to conduct a comparison study by manually testing 30 sets of devices (various smartwatches and connected Android phones). All sets of devices were studied in this research in order for us to examine and identify the device with high-performing call notifications push and devices with a comparably higher delay in the notifications. We also look into their embedded communication systems, user settings, and network architecture to decide the separating factors. These comparative studies and separating factors assisted in determining the following research questions:

- Is there a correlation between smartwatch/fitness tracker call notification delay and the physical proximity between the smartwatch/fitness track and connected smartphone?
- Does the delay vary between the types of smartwatch/fitness tracker and the types of connected smartphone?
- Does the delay vary between smartphone's Wi-fi connection and phone carrier service?
- Is there a significant difference in hardware design, software, network architecture between devices with the least-delay and devices with the most-delay in call notifications?

Related work suggests that besides from push notifications and DTN, two other vectors can influence the call notifications, BLE and connected device's Bluetooth range (figure 3). During the case study, these embedded systems and processes were evaluated and analyzed to identify the separating factors for smartwatches. The overall possibility of exploring a large number of smartwatches and their call notifications data could provide a ground for future research.

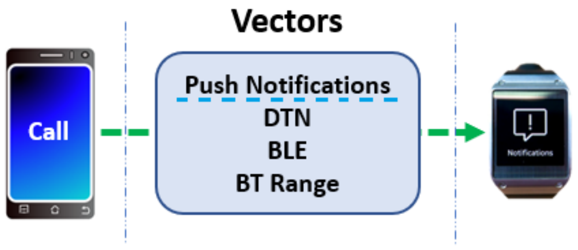

Figure 3: Push Notification Vectors

4.2 Research Study Design

Manual testing of the devices requires significant attention to produce valid data and to maintain data integrity. We took 30 sets of Android smartphones and their connected smartwatches for testing. The two sets of devices that were used in the initial observation were kept as the control since they both were same model devices under the same carrier network. It was easier to compare their push behavior and separating factors by testing the low performing set in various conditions to see what improves the performance. Both control sets were tested in a home Wi-fi network and alternate testing site's Wi-fi network to determine if there was any significant difference in data based on the network connection. There was no significant difference noted during the controlled testing of these two sets of devices in both locations. Therefore, it was determined testing in both locations will be considered valid. During the manual testing phase, calls were made to these phones to record call notification times in seconds to their connected smartwatch. We recorded the time between the moment a call displayed on the phone to the moment call notification displayed on the smartwatch. Each set was tested multiple times at four different physical proximity (0 feet, 5 feet, 10 feet, and 20 feet) from their connected device (most of the smartwatch/fitness tracker's BT-range are at 20-30feet). In addition, during the study, each phone's available memory and service carrier's information were also recorded. All test were conducted in a controlled environment with the same applications setting and four set conditions.
All collected data were analyzed to record call notifications push time. Depending on smartphone types and their settings, on average it takes a phone 8-10 seconds before it pushes a call to voice-mail. Therefore, We determined if the notification push takes longer than 10 seconds, we will consider notification push as “delay". We created a simple mathematical equation for our statistical analysis where variables:

Call notification $Delay = Dt$
$totaltime = t$
Notification time in $Phone = nP$
Notification time in $Watch = nW$
then $t = nW - nP$
if $t > 10$ (value in seconds)
then $t = dt$

The following combinations of Android smartphones and smartwatches were part of this manual testing and the case study. All sets of devices were tested under each case study defined below.

Galaxy S6 - Fitbit Alta	Galaxy S9 - Fitbit Versa
Galaxy S6 - Fitbit Charge 2	Galaxy S9 - Fitbit Char 3
Galaxy S6 - Fitbit Charge 3	Galaxy S9+ - Galaxy Watch
Galaxy S6 - Gear S3	Galaxy S9+ - Fitbit Versa
Galaxy S6 - Galaxy Watch	Galaxy S9+ - Fitbit Char 3
Galaxy S7 - Fitbit Charge 3	Galaxy S9e - Fitbit Versa
Galaxy S7 - Gear S3	Galaxy S9e - Fitbit Char 3
Galaxy S7 - Galaxy Watch	Galaxy S10 - Iconic
Galaxy S8 - Fitbit Versa	Galaxy S10 - Fitbit Versa
Galaxy S8+ - Gear Fit2 Pro	Galaxy S10+ - Fitbit Versa
Galaxy Note8 - Fitbit Char 3	Galaxy S10e - Versa Lite

Table 1: Smartphone - Smartwatch Combination List

4.2.1 *Case Study 1.*

In our first case study, we tested all phones while they were connected to the same Wi-fi. We also ensured their watch-side applications notification function in the phone were set to “call notifications" only. The collected data were set as the baseline for each set of devices.

4.2.2 *Case Study 2.*

In our second case study, we ensured all phones were connected to the same Wi-fi while the watch-side applications notification function in the phone set to “call notifications" and “text notifications". We collected the data and tried to see if smartwatches receiving multiple types of notifications has any effects comparing to the baseline data from the first case study.

4.2.3 *Case Study 3.*

In the third case study, all phones were disconnected from Wi-fi and used their own phone carrier service while the watch-side applications notification function in the phone set to “call notifications" (similar setting to baseline study). This helped us identify if there were any notifications time increases or decreases comparing to the baseline data collected during case study 1; and also to identify if the notifications time varies by different mobile carriers.

4.2.4 *Case Study 4.*

In the last case study, we removed all smartwatches from their connected phone first then reconnected back as adding a new device and sync them back to their app. The phones were connected to the same Wi-fi while the watch-side applications notification function in the phone set to “call notifications" only. This study was simply conducted to see if removing a connected device and re-sync them back makes any difference since many cases various updates in devices and their applications can have effects or unknown interference if they are not all updated properly.

5 DISCUSSION/RESULTS

When an Android phone receives call, text, email or any other types of messages it pushes the notification to Android wear application service to determine “to push" or “not to push" a notification to a synced wearable device. Notifications from all applications in smartphone go through this process. Once the application service determined to push notification, it sent a notification to the synced smartwatch via Bluetooth connectivity [6]. Even though the process is designed to display notifications almost immediately upon receiving in the phone, that might not be always the case. During our case study, we closely observed additional 30 sets of different Android phones and their connected wearable smartwatches in three different settings (all testing was conducted in same location and under same Wi-fi network connection). In our fourth case study, we removed synced smartwatches from their phone then connected them back as they were a new device. When a connected device is sync with a smartphone, it automatically looks for various application updates. This observation simply performed to eliminate if users missed any applications update that might decrease their smartphone's performance and communication network. Each case studies discussed in details below.

5.1 Case Study 1

In our first case study, we observed push notifications for all sets of devices (including two controlled sets) in the various distance while were connected to Wi-fi network. We also set all device notification setting to "call notification" only to control the notification volume from other applications such as text, social media, emails, etc.

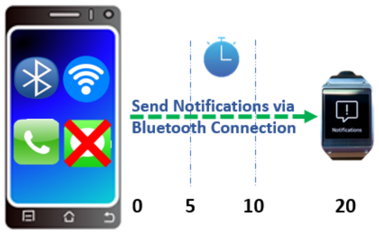

Figure 4: Case Study 1 Setup and Observation

The goal of this study was to set a baseline with minimum notification setup and to monitor if physical proximity between devices plays any role in their call notification push. There was no significant difference noted between t or Dt at their different physical proximity. However, only a few devices were observed to dropped push notification completely at a certain distance. Which we evaluated and analyzed later for their architecture, design and push notification vectors to include the BT-range.

5.2 Case Study 2

We set-up all sets of devices similar to our first case study except this time we turned on notifications for other apps such as text notification as well.

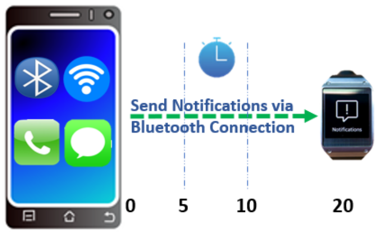

Figure 5: Case Study 2 Setup and Observation

The goal of this study was to observe all sets of devices to see if they have any changes in call notification behavior if notification types for push are increased. During the study, none of the devices had any significant changes in their call notifications push comparing to the data from the previous case study. There was also no difference noted between physical proximity of the devices and in their push notification time other than a few devices already identity during the first case study.

5.3 Case Study 3

In this case study, we set up all devices to their carrier network service instead of Wi-fi connection. Our goal was to monitor if delay behavior varies when Android phones are not connected to Wi-fi.

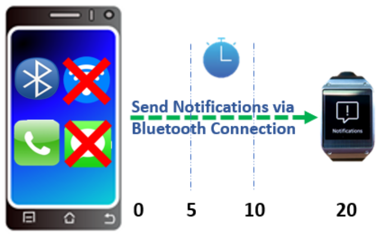

Figure 6: Case Study 3 Setup and Observation

During the observation, seven sets of devices displayed delay behavior and some of them even dropped the notification push when they were set to their phone carrier service. When the devices were connected back again to Wi-fi, their t value decreased immediately and no notification push was dropped. Seven sets of devices are not a large number, but this case study indicates that there might be a correlation between push notification delay and the carrier service in the wild. A large scale further study and analysis will be needed for future work to determine how and what levels of effect carrier services might have in the push notification process.

5.4 Case Study 4

In this case study, we observed all 30 sets of devices for any unnoticed or hidden update and compared the data to see if a low performing device's push notifications can be improved by re-sync and updating applications. During the study, three sets of devices performed slightly ($t < 10$ but $t >= 5$) lower compared to other devices. Out of these three sets of devices, two devices (Samsung Galaxy S8) had a pending application updates. Once the smartwatch was re-synced back, the application update request was generated.

Upon completing the update test was performed and a slight performance increase was noted immediately. Even this application was not directly linked to the Android ware application, by updating the app improved push notification performance. Other set of devices didn't generate any update notification when re-sync smartwatch, however, both sets performance were also improved compared to their data prior to the device removal and re-synced.

6 EVALUATION AND ANALYSIS

The research study was completed in a very short period and by manually testing devices. It will be ideal in the future to conduct this testing with a customized tool and in a large platform of devices to confirm all our findings. Based on our four case study results, we analyzed all low performing devices and evaluated all factors with the tendency to contribute to the push notification "delay".

6.1 Device Comparison

The two sets of controlled devices from our initial observations were compared side by side to determine all separating factors between them. In our initial observation, the first controlled device (C1) performed much more poorly compared to the second controlled device (C2). We evaluated both sets of devices and swapped their connected smartwatches to see if their performance changed based on their connected device. There were no notable changes in the performance during the swap. Aside from their design space and network, we also evaluated each of the devices settings, available memory space, and application updates (shown in Table 2 below).

Evaluation Benchmark	Samsung Galaxy S7 (C1)	Samsung Galaxy S7 (C2)
Wi-fi Network	✓Same	✓Same
Bluetooth Settings	✓Same	✓Same
Carrier Service	Sprint	Sprint
Test Location	✓Same	✓Same
Watch App Settings	✓Same	✓Same
Device Memory	32GB	32GB
Stored Data Volume	31.2GB	30GB
Free Memory	856MB	2GB
Android Service Setting	Do Not Disturb	None
Application Updates	None	None

Table 2: Controlled Devices Evaluation & Comparison

Based on the analysis, both Android phones were exactly the same and in the same network connection. However, two factors among them stand out were their available memory space and Android service settings. The low performing device (C1) had much less memory space available comparing to the higher performing device (C2). Device C1 also was set to "do not disturb" mode. To confirm these identified factors might be the contributing factors in the push notifications delay, we first removed the "do not disturb" settings to allow all calls to meet Android wear service, then we cleared memory space to increase available memory space for the device (C1). Once both changes were made to the device (C1), we perform the test again under all four defined case study and collected data. Our new data indicated a performance improved of 75% in the device (C1). We evaluated other low performing devices for similarity and identified another set which also had very low free memory space (200MB). As far the "do not disturb" service setting goes, it only prevents Android wear service from push determination. In which case, the push will never perform.

6.2 Data Analysis

The manual testing data for all devices were recorded in an excel spreadsheet for review and analysis. All data collected from our case studies were analyzed thoroughly then injected into a scatter chart (as shown in Figure 7) for further evaluation. In the chart, each case study data displayed in their individual case study blocks, where y values are the push response time in seconds and all four testing distance are color-coded in four categories. A red line place at y value = 10 mark for "Delay". All data above the line are considered clear "delay" and data below the line considered no notable delay. Data with y value $t > 5$ and $t < 10$ are considered to be displaying comparably delayed behavior but not a true push notification delay based on our analysis where y value must be greater than 10 seconds. Our low performing device from the initial observed case study 1 (C1) data recorded in black dots and connected lines. The red dot indicates the device was unable to perform a notification push.

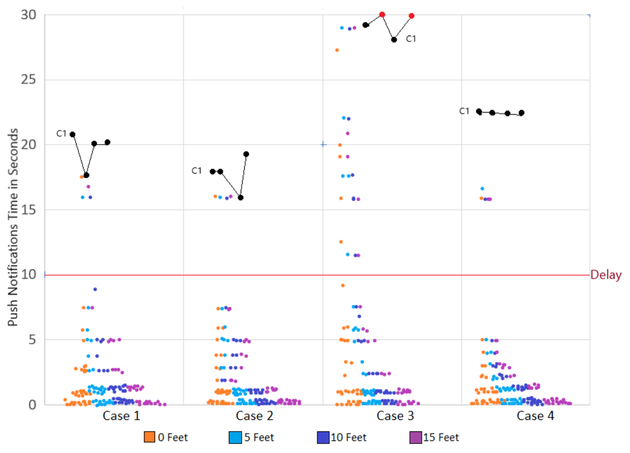

Figure 7: Case Study Observation Data

The data from the case study 1 (only call notification on) and case study 2 (both call and text notifications on) were compared to identify any notable differences. Which will allow us to determine if reducing push service volume from multiple applications can improve Android wear service push notifications performance. There were no differences noted in overall notifications push time if there were one application or multiple applications were selected for Android wear service to perform push. Android wear service processes its call notification determination as soon an alert is available on phone-side apps and immediately moves it to push determination then waits for the next available alert without causing any delay. We analyzed the case study 3 to identify if placing the device under their carrier service has any effects on push notifications. We observed performance dropped (the notification push time increased) in eight sets of devices compared to their data collected during case

study 1 and case study 2. Those sets of devices performance were back to their baseline when tested again under their Wi-fi connection. These data indicated that push notification can be effected in the wild when phones are performing under their carrier service network and low network connectivity can delay the overall push notifications.
Many cases smartphone users may pause an update multiple times for later and forget to return to it. Our fourth case study was designed to capture any unnoticed or hidden applications updates in the phones. We identified two devices with pending application (image gallery and Venom) updates. Once the applications were updated on both devices and tested again, we noted that both device's performance improved immediately.
In our last step of data analysis, we separated all data for the devices that performed out of norms due to low memory space, applications update, etc. Then we calculated the average push time in each case by phone types (as shown in table 3).

Date	Case 1	Case 2	Case 3	Case 4
Galaxy S6	6.2	6.05	15	5.5
Galaxy S7	2.56	3	8.69	2.19
Galaxy S8	2.25	3	2	2.25
Galaxy S9	1	0	1	0
Galaxy S10	0	0	0	0

Table 3: Average Push Notification Time by Phone

The goal of this step was to produce a top-level view of the data for all tested smartphones. Based on the analysis we determined to push delay almost non-existence on the newer version of Android phones. The older model devices have very slight delay behavior in their push notifications, however, the $dt < 10$ therefore, we considered the behavior as slower performance rather than an actual push notifications delay. We rejected the Samsung Galaxy S6 dt value ($dt > 10$) listed under Case study 3 since we previously determined some device has the potential to display delay behavior under their carrier network service.

6.3 Research Questions Review

Based on the evaluation and the collected data analysis, we were able to make determinations on the research questions we listed earlier in the research approach section.

There is no significant correlation between smartwatch call notification push and the physical proximity between the smartwatch and connected smartphone.

Base on all four case studies, there was no difference noted in the call notification push at various proximity between smartphones and smartwatches. Therefore, we determined that physical proximity does not play any role in call notification push unless the devices are outside of their Bluetooth range. In that case, the connected smartwatch will not receive any notification at all and rather dropped the push notification if outside of a certain BT-range.

Call notification push doesn't vary between smartwatch types, but it may vary between smartphone types.

In our study, some devices displayed delay behavior in notification push. For those sets of devices, we looked at both the phone and smartwatch separately to determine if the delay notification behavior is in the phone or in the smartwatch. Those specific smartwatches didn't display the same behavior when they were connected with another model smartphone. The delay behavior in call notification push was mostly observed in older models (i.e Samsung Galaxy S6 and Samsung Galaxy S7).

There is a slight difference in call notifications push between Wi-fi connection versus phone carrier service.

Even in most sets of devices there was no significantly notable difference in call notification push between testing under Wi-fi connection and testing under phone carrier service; a few (24% devices within two carrier service) devices displayed delay behaviors in their call notification push comparing to Wi-fi connection when tested under their carrier service network.

There is no significant difference in hardware design, software and architecture, between low performing and high performing devices other than their memory space, physical looks, and processors.

Even there is no significant difference in low performing and high performing device's architecture and software the memory space definitely played a role in how fast the push notifications were processed. Base on the data collected, all newer smartphones have grater memory available comparing to the older version of the phone. Push delay is almost non-existent in Samsung Galaxy 9 and 10 models. These models also have installed memory space between 64MB to 128MB and expanded memory up-to 512MB.

6.4 Push Notifications Vectors

An Android smartwatch mostly maintains four states that are intended to save battery power: 1) watch is fully awake, 2) watch is dozing, 3) watch is sleeping, and 4) charging. Android smartwatch's wake-up period usually last only 2% of its overall uses period. Beside from "flick and look" one main and most common triggering factor is push notification [5]. The purpose of push notification is to wake up the watch when there is a notification available on the phone side and post the notification in the watch, however, vectors such as DTN, BLE and BT-range sometimes plays a role in the Android wear service, and how quick it can respond "to push" or "not to push".

6.4.1 Delay Tolerant Network. The routing protocols in DTN tend to adapt themselves even when they are not continuously connected to their paired device. Especially when in an environment where they are in the wild (i.e. not connected to Wi-fi), they propagate "multiple copies of data packets to increase the probability" of the delivery [1]. Our study identified DTN does not affect the push notifications when they are in a Wi-fi network connection. However, in the wild (carrier service) DTN processes data in a multiple packets delivery methods and blindly forward copies of packets to any available nodes without a selection criteria[1]. Packets for Android wear service may not deliver directly and rather push towards whichever service it comes in contact with first. As a result, some packets may not reach the Android wear service immediately for push determination (to push or not to push) and ultimately can cause a delay in push notifications. In our study, eight devices displayed delayed behavior in their push notifications when they were removed from the Wi-fi connection and tested under their carrier service.

6.4.2 Bluetooth Low Energy. BLE is designed to perform short-range communications. BLE is also a low-powered controller to carry out their single-hop solution protocols. Most IoT devices are dual-mode and capable of implementing regular Bluetooth and BLE protocol stacks. When devices do not meet BLE range criteria or BLE protocol fails, it automatically can hop to regular Bluetooth protocol. Therefore, BLE wouldn't affect Android wear service push notification process and rather push it over to regular Bluetooth in the case of service failure.

6.4.3 Bluetooth Range. Smartwatches mostly communicate via their Bluetooth connectivity with their synced smartphone. However, in most smartwatches or other wearable devices, the Bluetooth range intentionally kept short to save energy. During our research study, we observed a few devices failed to push at a certain range. Even there was no correlation between the physical proximity of the connected devices and their push notification, the BT-range did play a role in when to push or not to push. As an example, when we tested Samsung Gear S3, no push delay or drop noted until at 15 feet distance. The Gear S3 had the BT-range of up-to 75 feet, therefore, notification push should be performed up to 75 feet as well. But the device was only capable of receiving push notifications at 15 feet distance and completely dropped at 20 feet distance. Though some devices may have longer BT-range, a push may not be performed at the same listed distance and rather performed to a much shorter distance due to their BLE communication.

6.5 Other Identified Vectors

During our study, we observed a few other vectors which can affect the push notifications in an Android phone. These vectors can vary by phones, watches, and their settings, etc. however, we were able to filter down a few common vectors with higher probability to disrupt the push notifications.

6.5.1 Stored Data Volume vs. Free Memory. Aside from the models of smartphone, another factor might influence the push notification is how much data such as images, documents files are stored in a device. Based on the behavior from the two sets (Samsung Galaxy S7) of controlled devices (which were the same model and in the same network) and analysis of their settings, available free memory and stored data volume, the main separating factor between them was the amount of stored data and available memory space.

6.5.2 Older Version of Phone. In our study, even though all older phone didn't display significant delay behavior ($t > 10$), their performance was somewhat slower (shown in Table 2) than newer devices. While newer devices push notifications performance almost at immediate, older devices performance varied between 5-10 seconds. Also in newer devices, the data storage capability is way larger than older devices. Which makes older devices less ideal for faster push notifications. For same reason, all medical alert wearable device, older models of Android phone should be avoided for patient monitoring.

6.5.3 Do Not Disturb Setting. The "do not disturb" setting in the phone provides with great benefits to the users where it can be set manually for a specific time or automatic start and ending time for every day. Even when do not disturb is set it will allow bypass calls or messages to the phone if the sender's contact saved to users' favorite. However, when the Android wear service look for when to push, if the phone is set to "do not disturb" setting, it will not push any notifications to its connected smartwatch regardless of favorite contact selection.

6.5.4 Phone Carrier Service. It was expected during our case study that the probability of the phone carrier service having an effect on push notification are very low since the push notifications communicate via Bluetooth connectivity once the Android service determines to push. Collected data suggested differently. Several sets of devices displayed delayed behavior when they were tested under case 3 environment.

6.5.5 Applications Updates. Device performance can be interrupted by lack of application updates. Even some application (i.e. image gallery) may not be directly related to call or text but still can influence the overall performance of the device and delay the Android wear service decision making process. As a result, it can delay the push notifications.

7 CONCLUSION

As the wearable devices communication capability and popularity continue to rise, many developments of these devices are now focusing on the medical industry to assist with varies health monitoring and related alerts. Some of these alerts could be time sensitive and require immediate attention. If there is a notification push delay in smartphone call notifications process, then the same delaying vectors can affect medical alert devices as well. Based on our case study results and analysis, the physical proximity of the connected devices has almost no effects on call notification push. However, our research suggests that even most newer smartphone performed efficiently, few older devices displayed delay behavior in their notification push. Phone stored data volume and phone carrier service also seemed to play a small role in notification push delay. These vectors might not seem too significant, however, it can easily affect a medical wearable device and prevent the alert notification push from performing in a timely manner. For future work, a tool will be developed to test large numbers of smartphones with smartwatches to study call notifications, text notifications, and other social media notifications as well. The future studies will help with identifying true time delay in these notifications processes and to determine the vectors that influence the notifications process and speed. Another area also to look at in our future research to see the correlations between a smartphone's stored data volume (i.e. images, document files, etc. vs. free memory) and the speed of push notifications. The overall research highlights some of the influencing vectors in the push notification process. More study and research in this area will provide the developers with a "lookout factors" when developing wearable medical alert devices.

REFERENCES

[1] T. Abdelkader, K. Naik, A. Nayak, N. Goel, and V. Srivastava. 2016. A performance comparison of delay-tolerant network routing protocols. *IEEE Network* 30, 2 (March 2016), 46–53. https://doi.org/10.1109/MNET.2016.7437024

[2] M. Al-Sharrah, A. Salman, and I. Ahmad. 2018. Watch Your Smartwatch. In *2018 International Conference on Computing Sciences and Engineering (ICCSE)*. ICCSE, Kuwait City, Kuwait, 1–5. https://doi.org/10.1109/ICCSE1.2018.8374228

[3] F. Z. Benhamida, A. Bouabdellah, and Y. Challal. 2017. Using delay tolerant network for the Internet of Things: Opportunities and challenges. In *2017 8th International Conference on Information and Communication Systems (ICICS)*. IEEE, Irbid, Jordan, 252–257. https://doi.org/10.1109/IACS.2017.7921980

[4] Yong Li, Pan Hui, Depeng Jin, and Sheng Chen. 2015. Delay-Tolerant Network Protocol Testing and Evaluation. *IEEE Communications Magazine* 53 (2015), 9. Issue 1.

[5] Xing Liu, Tianyu Chen, Feng Qian, Zhixiu Guo, Felix Xiaozhu Lin, Xiaofeng Wang, and Kai Chen. 2017. Characterizing Smartwatch Usage in the Wild. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '17*. ACM Press, Niagara Falls, New York, USA, 385–398. https://doi.org/10.1145/3081333.3081351

[6] Xing Liu, Yunsheng Yao, and Feng Qian. 2017. Poster: Improve Push Notification on Smartwatches. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '17*. ACM Press, Niagara Falls, New York, USA, 154–154. https://doi.org/10.1145/3081333.3089298

[7] G. Muhammad, S. M. M. Rahman, A. Alelaiwi, and A. Alamri. 2017. Smart Health Solution Integrating IoT and Cloud: A Case Study of Voice Pathology Monitoring. *IEEE Communications Magazine* 55, 1 (January 2017), 69–73. https://doi.org/10.1109/MCOM.2017.1600425CM

[8] Adam Roegiest, Luchen Tan, and Jimmy Lin. 2017. Online In-Situ Interleaved Evaluation of Real-Time Push Notification Systems. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*. ACM Press, Shinjuku, Tokyo, Japan, 415–424. https://doi.org/10.1145/3077136.3080808

[9] J. She and X. Bai. 2016. Caching strategy in Mobile Delay Tolerant Network. In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, Beijing, China, 497–500. https://doi.org/10.1109/ICSESS.2016.7883117

[10] Shachar Siboni, Asaf Shabtai, Nils O. Tippenhauer, Jemin Lee, and Yuval Elovici. 2016. Advanced Security Testbed Framework for Wearable IoT Devices. *ACM Transactions on Internet Technology* 16, 4 (Dec. 2016), 1–25. https://doi.org/10.1145/2981546

[11] Liam D. Turner, Stuart M. Allen, and Roger M. Whitaker. 2015. Push or Delay? Decomposing Smartphone Notification Response Behaviour. In *Human Behavior Understanding*, Albert Ali Salah, Ben J.A. KrÃűse, and Diane J. Cook (Eds.). Vol. 9277. Springer International Publishing, Cham, 69–83. https://doi.org/10.1007/978-3-319-24195-1_6

[12] Yi Yang and Guohong Cao. 2017. Characterizing and optimizing background data transfers on smartwatches. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, Toronto, ON, 1–10. https://doi.org/10.1109/ICNP.2017.8117536

[13] Thomas Zachariah, Noah Klugman, Bradford Campbell, Joshua Adkins, Neal Jackson, and Prabal Dutta. 2015. The Internet of Things Has a Gateway Problem. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15)*. ACM, New York, NY, USA, 27–32. https://doi.org/10.1145/2699343.2699344