Poster: Improve Push Notification on Smartwatches

Xing Liu Yunsheng Yao Feng Qian
Indiana University Bloomington

1. INTRODUCTION

Receiving push notifications is one of the key features of smartwatches. In our recent measurement study involving 27 smartwatch users [1], we found that push notifications are used by more than 200 applications, dominated by instant messaging, emails, social media, *etc.* In this work, we propose a suite of methods to optimize the performance, energy efficiency, and usability of smartwatch push notifications, which have several salient features distinguishing them from regular notifications received on a smartphone: requiring heavy phone-watch cooperation, being delivered over short-range Bluetooth link, and incurring non-trivial energy consumption on watches with very limited battery capacity. Considering these factors, our proposed work focuses on four aspects as elaborated below.

• **To Push or not to Push.** We found that when a notification is available on the phone side, apps/OS employ naïve policies to determine whether to push or not to the watch. For example, many Android apps push a notification when and only when the phone-side app is not running in the foreground. Such a policy may cause unnecessary pushes when the user is interacting with the phone (so she can already see the message in the notification bar on the phone), or cause useful pushes to be skipped when the user puts the phone aside (with the app running in the foreground) and walks away. Ideally, in most use cases, an app should perform pushes only when the user is *not* interacting with the phone. We thus plan to design and implement an API that intelligently determines whether to push or not to push by leveraging diverse types of information sources such as the user interaction level, the application type, and the physical distance between the watch and the phone. The API needs to be reliable and lightweight.

• **When to Push.** Today's smartphones perform pushes immediately when notifications are available. Doing so minimizes the latency while oftentimes incurring energy overhead on watches, which need to fully wake up and keep the display on for several seconds when a notification arrives. Despite such short wake-up sessions that account for only 2% of the overall usage period based on our crowd-sourced measurement, their energy footprint is as high as 27% of the overall watch energy consumption. To reduce the energy consumption incurred by push, we leverage the fact that different types of notifications have different delay tolerance levels. For many delay-tolerant notifications, they can be pushed to the watch in a single bundle, or be pushed when the watch wakes up. Batching or piggybacking message delivery is not a new concept. However we face several unique challenges here such as determining the delay tolerance level for notifications, devising an energy-efficient push strategy, and designing an interface allowing users to conveniently read multiple notifications on the watch.

• **How to Push.** On most smartwatches, notifications are pushed over the Bluetooth (BT/BLE) channel established between the phone and the wearable. However, the range of a BT link is short, and some vendors intentionally reduce the BT antenna power for energy saving purpose. As a result, handovers between BT and WiFi (many smartwatches have built-in WiFi) may frequently happen. We found that on commodity Android watches, a handover may take a long time to finish, leading to long delays for time-sensitive notifications such as chat. The problem we try to address is thus to ensure good push performance during frequent handovers. We observed from our user study that the arrival of notifications exhibits a strong bursty pattern. Within a burst of notifications, the watch can predict the loss of the BT connectivity and preemptively establish the WiFi channel. Notifications with high priority will then be pushed over both channels to ensure their prompt and reliable delivery.

• **What to Push.** When a notification arrives at the watch, watch-side apps may allow users to take further actions such as launching a phone-side activity, opening a URL, or taking voice input. We found that these actions only involve a small set of homogeneous operations across a wide range of apps. Here our idea is to allow the phone not only to push regular contents such as text and image, but also to push an "applet" defining the operations that the user can take. The applet is expected to be small, simple, and cacheable. It can be easily crafted using OS-provided APIs. This approach can help reduce watch apps' development overhead or even eliminate most watch-side apps whose sole job is to handle simple interactions upon receiving notifications. The challenges include making the applets lightweight, secure, and cover common interaction options that a user may take.

On-going Work and Future Plan. We are working on designing and implementing the above four optimizations. We plan to integrate them into a holistic software framework. Evaluations will be conducted using both in-lab controlled experiments and our on-going smartwatch user study [1].

2. REFERENCES

[1] X. Liu, T. Chen, F. Qian, Z. Guo, F. X. Lin, X. Wang, and K. Chen. Characterizing Smartwatch Usage in The Wild. In *MobiSys*, 2017.