AFit: Adaptive Fitness Tracking by Application Function Virtualization

Harini Kolamunna,[1]
Yining Hu,
Diego Perino,[2,4,6]
Kanchana Thilakarathna,[1]
Dwight Makaroff, [3,5,6]
Xinlong Guan,
Aruna Seneviratne,[1]
Data61/CSIRO-Australia,
[1]University of New South Wales,
[2]Telefonica Research,
[3]University of Saskatchewan,
Email:(firstname.lastname)@data61.csiro.au,
[4]diego.perino@telefonica.com,
[5]makaroff@cs.usask.ca
[6]This work was done while the
author was at Data61.

Ubicomp/ISWC'16 Adjunct , September 12-16, 2016, Heidelberg, Germany
ACM 978-1-4503-4462-3/16/09.
http://dx.doi.org/10.1145/2968219.2971364

Abstract
The popularity of wearables is exponentially growing and it is expected that individuals will utilize more than one wearable device at a time in the near future. Efficient resource usage between the devices worn by the same person has not yet been effectively addressed by the current wearable applications. In this paper, we demonstrate the feasibility of application function virtualization by utilizing common capabilities of multiple wearables on the body through a cross-platform Android application - *AFit*. AFit is developed in a way that it can opportunistically leverage the resources of smartphone, smartwatch and smartglass depending on the context of the user, which is user activity. In this demonstration, AFit shows that it is possible to adaptively select the device to track the user movement for fitness tracking, rather than using randomly selected device or all devices, utilizing the common sensor of accelerometer on all devices.

Author Keywords
adaptation; smart wearable devices; context monitoring; fitness tracking

ACM Classification Keywords
C.5.3 [Microcomputers]: portable devices; C.2.4 [Distributed Systems]: Distributed applications

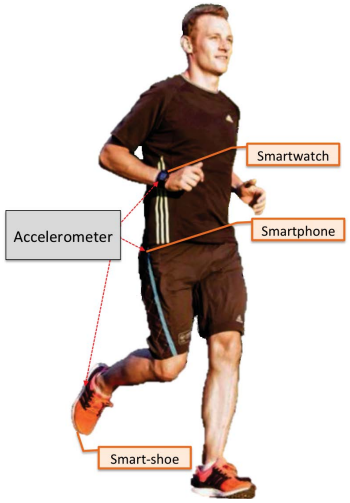

Figure 1: Example scenario.

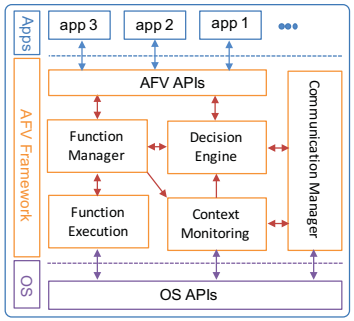

Figure 2: An overview of AFV framework.

Introduction

In the near future, we expect that individuals will have numerous smart wearable devices that form a local network within the human body capable of hosting distributed applications (a wearable personal area network, or PAN). Some of these devices are equipped with a rich-set of sensors and there could be many computation capabilities and sensing activities that can be performed by more than one device in this type of PAN. As an example, multiple devices (e.g. smartwatch, smartphone, smart-shoe) contain the same type of sensor (e.g. accelerometer) as shown in Figure 1. In addition, many devices are capable of performing the processing functions of compression and encoding as well as security functions of intrusion detection and encryption.

Moreover, the user and device context changes can be used in common functionalities utilization. As an example, let's consider the context change in the smartwatch that the battery status changes to low-battery. A fitness tracking application in this low-battery smartwatch can then utilize the fully-charged smartphone's accelerometer function, despite using the smartwatch's accelerometer. There are several implementations of context monitoring exist in the literature [1, 2], which can be adapted in context aware designs.

The question in such an environment is whether these common functionalities are efficiently used by today's applications. An analysis of several popular fitness and tracking applications that has both smartphone and smartwatch versions (e.g. UP, MyFitnessCompanion, Wear Run) suggests that the efficient collaboration between the smartphone and smartwatch are not adopted [3]. Motivated by this, we have proposed AFV [3], a framework which takes user and device context into account and effectively utilize common functionalities available in the PAN through application function virtualization.

In this demo, we show the feasibility of AFV through an implementation of a cross-platform Android fitness and tracking application - AFit - that leverages common resources available in the devices in a PAN. AFit takes user and device context information to give the optimal information by adapting to the context of the user.

AFV Framework

Our proposed AFV framework automatically leverages the available common functionalities in a PAN and user/device context information, in order to simplify application development and improve the quality of experience of the users. Specifically, the AFV framework has the following key features:

- **Optimization:** carries out function placement to maximize the efficient functionality usage via a heuristic greedy approach;
- **Adaptation:** dynamically configures the system at runtime according to changes in the context, applications and user needs;
- **Usability:** reduces the complexity of providing adaptable features for application developers and enables straightforward selection of these features via user configuration.

Figure 2 provides an overview of the main tasks and modules of the AFV framework [3]. The *Function Manager* module manages the functions registration requests coming from the applications via *AFV APIs*. The *Context Monitoring* module periodically monitors device/user context, providing inputs to the *Decision Engine*. The *Function Execution*

module manages function invocation on the device identified by the *Decision Engine*. The *Communication Manager* maintains efficient communication among the AFV-enabled devices.

Implementation of AFit

AFit is implemented as an Android application and installed in the smartphone. A lightweight, specific version of the AFV framework for AFit is implemented as a standalone application and installed in smartwatch and smartglass also. In this version of the AFV implementation, only the accelerometer functionality is considered with the application-specified context of user activity changes. This context is monitored by AFV, and the rules-based decision engine performs the adaptation in the system. Both the smartwatch and smartglass have the standalone applications that execute the AFV framework activities installed. The version of the AFit application installed in the smartphone contains the AFV framework implementation as well.

There is a predefined set of rules for the decision-making engine, configured by the developer. It is possible for an application to provide features for the user to add to the set of rules by checkboxes or other user interface mechanism, but this has not been implemented in the prototype. Once the application is started in the smartphone, it broadcasts the message "START" via WiFi broadcast where all the devices are connected to the same WiFi network. The Android class `WifiManager` is used to get broadcast address and `Datagram packets` are used for packet delivery. Smartglass registers its accelerometer to the OS immediately after receiving the "START" message.

At the same time, the smartphone starts operating its accelerometer for activity recognition. Activity recognition is done using the `onSensorChanged(SensorEvent event)` method from the Android system, and the sensor is registered with `TYPE_ACCELEROMETER`. The smartphone continuously monitors the accelerometer in order to detect any considerable movement (i.e., distance more than 3).

Once the smartwatch receives the message "START", it starts operating its accelerometer for activity recognition as the same way the smartphone does. If smartphone or smartwatch detects any activity from these devices, the device that detects activity broadcasts the message "watch_detect_activity" or "phone_detect_activity". Once any of these messages is received by the smartglass, it unregisters its accelerometer. If both the smartwatch and the smartphone detect activity, the smartphone is given priority for registering its accelerometer with AFit, based on the default rule set hard-coded into the decision-making engine. If the smartglass receives both "watch_no_activity" and "phone_no_activity", when there are no activities detected by both smartphone and smartwatch, smartglass registers its accelerometer again.

Demonstration

Our demonstration depicted in Figure 3 involves three smart wearable devices that are highly likely for an individual to use simultaneously in the near future (i.e., smartphone, smartwatch and smartglass). We consider a fitness tracking application installed in the smartphone that requires accelerometer data and that an accelerometer is available in all three devices. For the efficient provision of functionality, AFV uses the movements of body parts and the application's preference of device selection as context information. We then let the user wear three devices and show them the context-aware placement of accelerometer sensing function depending on the user's activity.

For instance, assume a scenario where the user will first

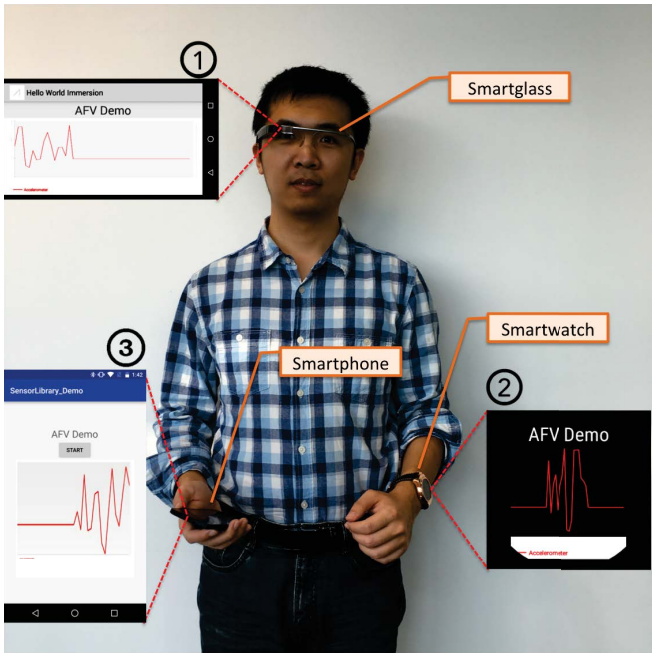

Figure 3: Demonstration.

wear the smartglass and start to move. The user will see the accelerometer trace in the smartglass's screen. Subsequently, the user may wear the smartwatch and move their hand. The accelerometer will then stop functioning in the smartglass (trace goes to zero) and at the same time accelerometer trace is shown with fluctuations in the smartwatch's screen. Moreover, the user will feel a vibration from the smartwatch when the accelerometer function is moved to the smartwatch. As the final step, the user may hold the smartphone and move their hands. As in the previous case, the accelerometer function moves from the smartwatch to the smartphone and indicates the function movement via a smartphone vibration. If both the smartphone and the smartwatch stop moving, the accelerometer will start functioning in the smartglass again.

Conclusion

Leveraging the common functionalities in different smart devices are not yet being adopted by the application developers due to the added complexity of and requirement of additional knowledge on the total system. Motivated by this, we developed a framework that enables developers to use these common functionalities optimally without additional knowledge. Here we demonstrate a developed smartphone application which adapts to user's physical activities by using a lighter implementation of the proposed framework.

REFERENCES

1. A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada. 2010. Fusing Mobile, Sensor, and Social Data to Fully Enable Context-aware Computing. In *Proc. HotMobile*. ACM, New York, NY, 60–65.
2. S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. 2008. SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-rich Mobile Environments. In *Proc. MobiSys*. ACM, New York, NY, 267–280.
3. H. Kolamunna, Y. Hu, D. Perino, K. Thilakarathna, D. Makaroff, X. Guan, and A. Seneviratne. 2016. AFV: Enabling Application Function Virtualization and Scheduling in Wearable Networks. In *Proc. UbiComp*. ACM, New York, NY, to appear.