SPOT: A Smartphone-based Control App with a Device-Agnostic and Adaptive User-Interface for IoT Devices

Mohammad-Mahdi Moazzami[a]
Michigan State University
East Lansing, MI
moazzami@msu.edu

Wei-Pen Chen
Fujitsu Labs. of America
Sunnyvale, CA
wchen@us.fujitsu.com

Daisuke Mashima[a]
Advanced Digital Sciences Center, Singapore
daisuke.m@adsc.com.sg

Guoliang Xing
Michigan State University
East Lansing, MI
glxing@msu.edu

Ulrich Herberg[a]
Independent Researcher
Santa Clara, CA
ulrich@herberg.name

[a]Work is done while the author is with Fujitsu Labs of America

Ubicomp/ISWC'16 Adjunct , September 12-16, 2016, Heidelberg, Germany ACM 978-1-4503-4462-3/16/09.
http://dx.doi.org/10.1145/2968219.2968345

Abstract
The recent progress of IoT technologies, including broad penetration of smart appliances such as remotely controllable lights, thermostats and cameras, have changed the way we interact with the appliances in our homes and perform our daily activities. However, the significant heterogeneity in the emerging IoT devices has led to fragmented smart-home systems in which each single appliance vendor provides proprietary solution for appliance specific connectivity and user experience. One of the desired solutions in smart-home systems is to have a unified smartphone app that can control any arbitrary IoT appliances. In this extended abstract we focus on the design of such an app only from the user interaction viewpoint. In particular, we present SPOT app, a smartphone-based platform for multi-vendor smart-home appliances, that features an adaptive and device-agnostic user interface enabled by a novel device driver mechanism. To validate the flexibility and feasibility of our design, we have built a SPOT prototype based on 8 real IoT devices and present the quality of generating such adaptive graphical user interface by the measure of screen smoothness in the prototyped android app.

Author Keywords
IoT appliances; Smart-home; Dynamic User-Interface; Device-agnostic GUI; Dynamic Device Drivers

Figure 1: Heterogeneity in today's smart-home IoT systems. Each appliance requires its own app as shown here that requires an appliance-specific user interface for control

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous.

Introduction

The vision of connected, smart homes has been around for decades and smart-homes have been regarded as one of the promising realizations of Internet of Things (IoT). In this vision, users easily perform tasks involving diverse sets of devices in their home without the need for painstaking configuration and custom programming. While modern homes have many network-capable devices, applications that coordinate them for cross-device tasks have yet to appear in any significant numbers.

The lack of dominating standards for IoT communication, control, and data management results in highly fragmented smart-home systems consisting of proprietary solutions provided by each device vendor. Thus, users are required to use different control interfaces, e.g., mobile apps, to interact with smart appliances in their homes, or are forced to use devices sold by a single vendor to acheive the best user experience. For instance a user needs to separately operate Philips HUE [1] app and WeMo [2] app to control lighting and smart power plug even in the same room. Although major players, including Apple and Google, started to provide solutions [3, 4] for connected smart-home, these solutions are not inter-operable and set of supported devices are often disjoint. Recently cloud-based solutions for multi-vendor and cross-device interaction, such as [5, 6, 23], have also appeared, but the users' choice of appliances is still limited to what the service providers or device vendors offer. We argue that lack of interoperability and heterogeneity among these efforts are hindering technological innovation in Internet of Things. One of the most important heterogeneity aspects is the diverse appliance control apps (cf. Fig. 1). Currently each modern smart appliance comes with its own smartphone app. These smartphone apps do not provide similarity in user interface nor any interaction with other apps. Consequently users have to switch between multiple apps when they want to operate different devices, which deteriorates the user experiences. Users need to learn specific apps' functionalities in addition to appliance functionalities. Such diversity in apps for appliance control presents barriers for interoperability and adoption of smart appliances.

Our study of the existing smart-home appliances and smart-home systems [7, 8, 9, 10, 2, 11, 12], as well as several studies from both academia and industry sectors [13, 14, 15], suggests that the main trend in this IoT application domain is toward to have unified system platforms. Following this direction, we design SPOT as an open, community-based, extensible platform to accelerate adoption and device support. Although our work is motivated by smart-home use case, the technology is generic and is applicable to other contexts. One crucial aspect of having a unified system platform, is to have a unified device-agnostic user interface. SPOT features such user interface by providing a dynamic device driver that adaptivly generate a device agnostic user interface for arbitrary IoT devices. The specification of the SPOT driver is publicly available so that device vendors, open-source community, or even end users can create or modify the drivers. [1]

Dynamic and Device-agnostic GUI Creation

Dynamic GUI creation based on the device driver definition is a salient feature provided by SPOT. The system utilizes the XML driver and extracts the fields, access controls, and message formats to communicate with the device. Similarly, upon receiving the XML driver, the system reads the appro-

[1] SPOT is an award-winning smartphone app in the MobiCom 2015 app contest. The driver specification and some supporting material is available at https://github.com/SPOT-SMARTHOME/SPOT

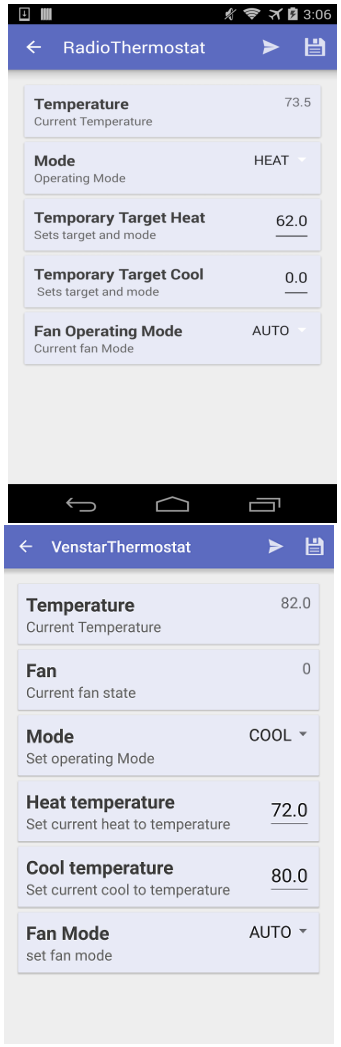

Figure 2: Example screenshots of dynamically generated GUI for two different IoT appliances: Venstar and Radio Thermostats.

priate XML tags indicating what type of GUI components has to be generated and appear in the associated section in the mobile app for any device that user wants to control. Relevant information is obtained from the user-interface (UI) related tags, *uiType, uiHelperTex, uiCaption*, defined in the variable driver unit. The GUI components in the Android are all sub-classes of a parent class called *View*, e.g., *Button, TextView,* and *ProgressBar*, and by intelligently implementing the system, the user-interface is rendered at runtime. This feature not only enables the users to control different variables from different appliances without requiring to design separate apps, but also provides the appliance vendors with the ability to indicate what variables and how they want users to have access to (i.e., access control). Moreover, by changing the values in UI-related fields in the XML driver, the GUI changes accordingly in the next run of the app without any need to change in the design of the app. All the information about UI components set in the XML are polled out by the driver parser component and then is mapped to our defined JAVA annotations (as used in the SPOT JAVA Library as well) during the process of generating the GUI at runtime. Fig. 3 shows a snippet of a SPOT XML driver.

Underlying Supporting System

The generated GUI requires an underlying system that connects it to the appliances. SPOT enables unified read/write interface for multi-vendor appliances. For this purpose, SPOT implements a novel driver model, for abstracting heterogeneity in communication and data models, and implements dynamic driver loading mechanism. The driver can be designed easily by open source community and even end users using SPOT's web portal[2]. The detail of the driver and how it handles the communication is beyond the scope of this manuscript. The details of the driver mechanism and the drivers of several IoT appliances used in our study along with the drivers schema can be found in [16] and [17], respectively. The architecture of the underlying system in high level is similar to [18, 19, 20, 21, 22, 23, 24, 5].

[2]http://spot-smarthome.github.io/SPOT/

Evaluation

The smoothness of displaying dynamic GUI: As an mobile app that provides a dynamic user interface for the users, drawing screen frames with a regular rhythm is essential for good performance and use experience. We analyze this by using an Android system tools, Systrace [25], which is particularly useful in analyzing application display slowness or pauses in rendering the UI components. Typically the analysis of display components (UI threads) by Systrace is reported under *SurfaceFlinger* process [26] as shown in Fig. 4. Having a regular rhythm for display ensures that UI components are smoothly appearing on the screen [25]. Fig. 4 illustrates the execution pattern of display component in SPOT. The regularity of *SurfaceFlinger* process suggests a smooth GUI rendering in the app. Moreover, the regular pattern in the CPU state in the upper panel of Fig.4 indicates that there is no other threads in the app, e.g., network communication, disk operation for DB access or loading the UI components like images, which may interrupt the rendering of the user-interface. This validates the efficient architecture of SPOT that achieves the smoothness in displaying the UI.

Conclusion and Future Work

The SPOT design benefits *users* by providing one single gateway to their smart-home appliances and applications. It eases device setup and management and provides a central place for users to trace their activities e.g., track their energy usage. Moreover, users can benefit from accelerated device support as the open XML-based device driver and the JAVA-annotation based driver framework are expected to encourage contribution from open-source developer community. In addition, the dynamic, adaptive user interface provided by

```
<variablesList>
  <variable>
       <name>ip</name>
       ...
       <showOnUi>true</showOnUi>
       <uiType>EditText</uiType>
       <helperText>
            e.g. 192.168.19.10
       </helperText>
       <caption>IP</caption>
  </variable>
  <variable>
       <name>on</name>
       ...
       <showOnUi>true</showOnUi>
       <uiType>TgleButton</uiType>
       <helperText>
            The On/Off State
       </helperText>
       <caption>On/Off</caption>
  </variable>
       ...
</variablesList>
```

Figure 3: The snippet of a SPOT XML driver including GUI related fields

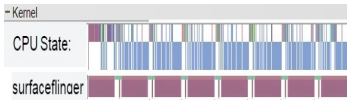

Figure 4: The smoothness of displaying GUI: The regular rhythm in *SurfaceFlinger* process indicates the smooth display rendering. The regular rhythm in the CPU state in the same period of time indicates no interference between the threads in the app.

SPOT improves the users' experience and decreases their learning curve. Our future plans include extending SPOT (as a whole system) and release it as a commercial platform for large-scale home automation deployments. Beyond smart-homes, we will investigate how to apply the SPOT approach to other IoT domains like smart vehicular systems or health-care systems and even across multiple IoT domains, that encounter exact or similar kinds of heterogeneity issues. Future plans also include extending the GUI and bring its adaptivity and dynamic aspect to the next level based on learning the latent users' preferences based on how they interact and use the app's GUI.

REFERENCES

1. Philips hue light. **http://www.lighting.philips.com**.
2. Wemo home automation. **http://www.wemothat.com/**.
3. Apple homekit. **http://www.apple.com/homekit**.
4. Physical web. **https://google.github.io/physical-web**.
5. Alljoyn. **https://allseenalliance.org**.
6. If this, then that. **https://ifttt.com/**.
7. Nest smarthome. **https://nest.com/**.
8. Ecobee thermostats. **http://www.ecobee.com/**.
9. Smart-things. **http://www.smartthings.com/**.
10. Wink smarthome. **http://www.wink.com/products/**.
11. Venstar. **http://www.venstar.com/Thermostats/**.
12. Radio thermostat. **http://www.radiothermostat.com/**.
13. Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7).
14. Richard Harper. *Inside the smart home*. Springer Science & Business Media, 2006.
15. A Sinclair. Vision of smart home the role of mobile in the home of the future. *GSMA, London*, 2011.
16. Ulrich Herberg and Mohammad-mahdi Moazzami. Dynamic device drivers, April 7 2016. US Patent 20,160,098,284.
17. Spot's github. **https://github.com/SPOT-SMARTHOME/SPOT**.
18. Mohammad-Mahdi Moazzami, Dennis E Phillips, Rui Tan, and Guoliang Xing. Orbit: a smartphone-based platform for data-intensive embedded sensing applications. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*.
19. M. M. Moazzami, D. E. Phillips, R. Tan, and G. Xing. Orbit: A platform for smartphone-based data-intensive sensing applications. *IEEE Transactions on Mobile Computing*, 2016.
20. Mohammad-Mahdi Moazzami. Droidsense: A toolkit for smartphone-based data-intensive embedded sensing systems. *PhD Forum of the 12th International Conference on Information Processing in Sensor Networks (IPSN)*, 2013.
21. Wei-peng Chen, Mohammad-mahdi Moazzami, and Daisuke Mashima. Internet of things (iot) device configuration construction, March 3 2016. US Patent 20,160,065,653.
22. Mohammad-Mahdi Moazzami, Dennis E Phillips, Rui Tan, and Guoliang Xing. A smartphone-based system platform for embedded sensing applications. Technical report, MSU-CSE-13-11, Department of Computer Science, Michigan State University, 2013.
23. Openhab. **http://www.openhab.org**.
24. Colin Dixon, Ratul Mahajan, Sharad Agarwal, A. J. Brush, Bongshin Lee, Stefan Saroiu, and Victor Bahl. The home needs an operating system (and an app store). In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, 2010.
25. Display and performance analysis in android apps. **http://developer.android.com/tools/debugging/systrace.html**.
26. Android graphics. **https://source.android.com/devices/graphics**.