Tap to Interact: Towards Dynamically Remixing the Internet of Things

Darren Carlson
Felicitous Computing Institute
National University of Singapore
13 Computing Drive
Singapore, 117417
carlson@comp.nus.edu.sg

Max Pagel
Felicitous Computing Institute
National University of Singapore
13 Computing Drive
Singapore, 117417
pagel@comp.nus.edu.sg

ABSTRACT

The number of networked Smart Devices available in everyday environments is rapidly increasing; however, many current devices adopt mutually incompatible networks, protocols, and application programming interfaces. As such, creating mobile applications that dynamically discover and integrate ambient functionality across multiple vertical markets remains challenging. In this paper, we introduce a novel integration technique that enables commodity mobile devices (e.g., mobile phones) to mediate control messaging *between* incompatible Smart Devices situated in the user's environment. The approach enables a variety of control capabilities and protocol translation services to be dynamically installed into a user's mobile device on-demand using plug-ins. The approach features an intuitive "Tap to Interact" workflow that allows a user to tap nearby Smart Devices with a smartphone to install required interaction plug-ins and automatically "wire" them together in interesting and potentially unforeseen ways. In our demonstration, we show how this approach enables a Sphero Robotic Ball to be utilized as a physical interface for controlling media playback on an Apple TV, interacting with networked-enabled lighting equipment, and flying a Parrot AR Drone helicopter – by leveraging a commodity smartphone as a plug-and-play Smart Gateway between mutually incompatible devices.

Categories and Subject Descriptors

D.2.11 [Software]: Software Architectures – Domain-specific architectures; Data abstraction. D.2.12 [Software]: Interoperability – Distributed objects. D.2.13 [Software]: Reusable Software – Domain engineering; Reusable libraries; Reuse models.

General Terms

Management, Design, Experimentation, Human Factors.

Keywords

Ubiquitous Computing; Internet of Things; Control Protocols; Plug-and-play; Smart Gateways.

MOBIQUITOUS 2014, December 02-05, London, Great Britain
Copyright © 2014 ICST 978-1-63190-039-6
DOI 10.4108/icst.mobiquitous.2014.258014

1. INTRODUCTION

As the Internet of Things (IoT) expands, the number of networked Smart Devices available in many everyday environments is rapidly increasing. Indeed, the estimated 10 billion current Smart Devices is forecast to eclipse 26 to 100 billion units by the year 2020 [1, 2]. Unfortunately, no universally adopted application-layer interoperability standards have yet emerged. As a consequence, it is often prohibitively complex for developers to create mobile applications capable of dynamically discovering and interacting with Smart Devices across heterogeneous IoT system boundaries[3].

To address the challenges of heterogeneous IoT environments, we are developing Ambient Dynamix [4], a plug-and-play middleware framework that enables mobile apps and Web apps [5] to perform rich context sensing and fluid Smart Device interactions through plug-ins that can be dynamically installed into the user's mobile device (e.g., smartphone or tablet). Dynamix runs as lightweight background service, leveraging the device itself as a sensing, processing and communications platform. Dynamix comes with a growing collection of ready-made plug-ins[1] and provides open software developments kits (SDKs) and a scalable repository architecture, which enable 3rd party developers to quickly create and share new plug-in types with the community. An overview of the Dynamix Framework is shown in Figure 1.

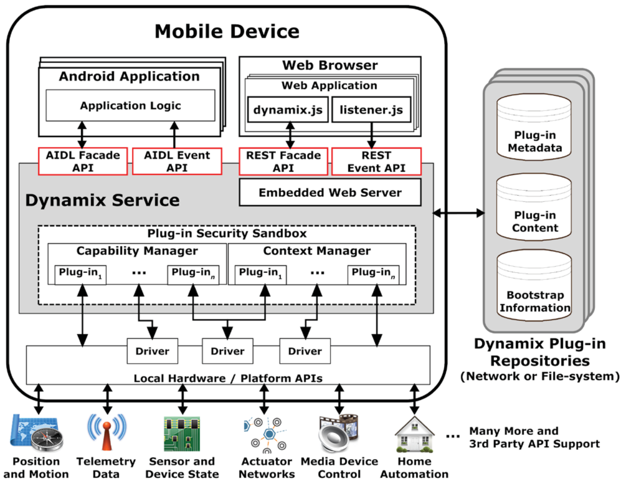

Figure 1: Overview of the Dynamix Framework

[1] http://www.ambientdynamix.org

A Dynamix Service is situated between a device's local hardware and (potentially many) Dynamix apps. Apps communicate with a Dynamix Service through easy-to-use application programming interfaces (APIs), including a Facade API (for requesting and controlling context support) and an Event API (for receiving framework notifications and context events). Dynamix automatically discovers, downloads and installs the plug-ins needed for a given sensing or control task. When the user changes environments, new or updated plug-ins can be deployed to the device at runtime, without the need to restart the framework.

In conventional IoT solutions, mutually incompatible devices are integrated through the use of *Smart Gateways* [6], which are implemented as a dedicated network appliance that serves as a communication hub and provides protocol translation services. In previous work [7], we described how Dynamix can transform a user's mobile device into an *Adaptive Smart Gateway* that moves with the user – providing network access and protocol translation services between mobile apps and encountered Smart Devices through plug-ins that can be installed on-the-fly.

In this work, we introduce an extension of our Adaptive Smart Gateway approach, which enables a Dynamix-enabled device to serve as a mediator *between* mutually incompatible Smart Devices – creating a personalized Smart Space that moves with the user. To support this functionality, we first developed a set of intra-plug-in communication features within the Dynamix Framework, which enables plug-ins to consume the services of other plug-ins. We then designed a control system library for Dynamix, called *Ambient Control (AC)*, that can be imported by plug-ins wishing to participate in control scenarios. The AC Library provides a set of *control commands* that define well-known interaction semantics and associated arguments (e.g., `MOVEMENT_LEFT`, `MOVEMENT_UP`, `DISPLAY_COLOR`, etcetera). Thirty control commands have been defined, and more are in development.

Control commands can be associated with a Dynamix plug-in through a *control profile* that describes the commands a given Dynamix plug-in can emit and consume. Control profiles are registered as XML snippets in an Internet-based registry that can be queried (e.g., using plug-in identifiers). A controllable plug-in is responsible for mapping the semantics of its associated control profile to the behavior of its underlying Smart Device. For example, the Phillips Hue plug-in changes a light's color to match the value contained in incoming `DISPLAY_COLOR` commands and the AR Drone plug-in changes the helicopter's pitch, yaw and roll orientation to match incoming inertial measurement unit (IMU) data.

A *control configuration* represents a specific arrangement of one controllable plug-in (a *receiver*) and potentially many controlling plug-ins (*controllers*). Multiple control configurations can operate simultaneously within a single Dynamix instance. The AC Library includes a Smart Wiring feature that optimally matches the inputs and outputs of the plug-ins in a given control configuration according to priority values. For example, the AR Drone plug-in prefers `SENSOR_IMU` commands, but can also accept `SENSOR_AXIS` (i.e., *joystick*) commands if `SENSOR_IMU` is not available. The AC Library also provides translation between control commands. For example, the derivative of a single value sensor (e.g., audio pitch) can be translated to `MOVEMENT_UP` and `MOVEMENT_DOWN` commands. Finally, the AC Library coordinates requested control configurations by managing required plug-in installations via Dynamix, handling the setup handshake process between plug-ins and managing full duplex communication channels between controllers and a receiver.

2. DEMONSTRATION

Using these new Dynamix features, users can uncover novel interaction possibilities in many environments by dynamically remixing encountered IoT resources on-the-fly. To showcase the potential of this approach, we developed a demonstration[2] called Tap to Interact (T2I), which was implemented as a Dynamix application that utilizes the features of the previously introduced AC Library. The T2I application enables users to tap mutually incompatible Smart Devices with a Dynamix-enabled smartphone in order to "wire" them together in interesting and potentially unforeseen ways. The plug-ins required to communicate with tapped Smart Devices are dynamically deployed into the user's mobile device during runtime, and control configurations are automatically generated based on the order of the taps.

The demo consists of a sample environment containing Smart Devices that expose heterogeneous APIs for sensing and/or control and adopt multiple network access technologies. These devices include a Sphero Robotic Ball, an Apple TV, Phillips Hue network-enabled lights, and a Parrot AR Drone helicopter. Radio-Frequency Identification (RFID) tags are affixed to each device to enable identification through tapping with a smartphone. Each Smart Device, along with a Dynamix Device (a stock Samsung S4), is connected in a home networking configuration, as shown in Figure 2.

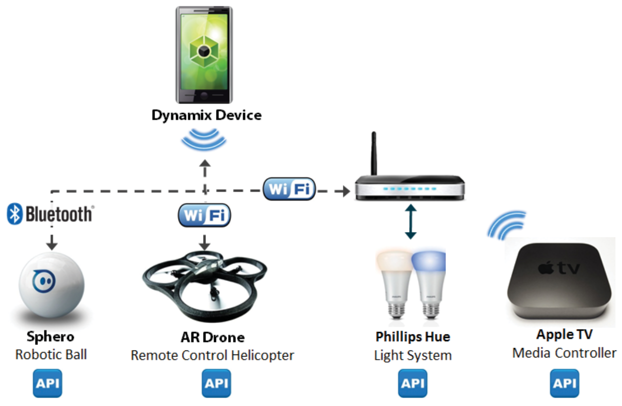

Figure 2: Demo Setup

The above Smart Devices are not inherently compatible; however, all provide network-available APIs that allow sensing and/or control. For example, the Sphero supports sensor data streaming of its IMU data over a Bluetooth connection, which can be used to determine the current orientation of the robot (the Sphero also accepts movement and light control commands). The Apple TV supports data connectivity over WiFi and allows media player discovery and media device control via the AirPlay protocol. The Phillips Hue light bulbs automatically form a mesh network using ZigBee and provide light control (e.g., color and intensity) to Internet based hosts through a hardware bridge device that connects directly to a local network switch. Finally, the AR Drone supports data connectivity over WiFi and accepts flight control commands that can be used to remotely pilot the drone.

Using the AC Library, we developed Dynamix plug-ins for each Smart Device within the demo environment. Each Dynamix plug-

[2] http://ambientdynamix.org/demos/tap-to-interact

in exposes the underlying features of its associated Smart Device and bundles all resources required for interacting with it. For example, the Sphero plug-in bundles the Orbotix Sphero native Android SDK[3] and the AR Drone includes the AR Drone SDK[4]. Each plug-in also registers a control profile with the registry, indicating which control commands it can consume and emit.

To begin the T2I demo, the user first taps the Sphero robot using the Dynamix-enabled smartphone, which registers the Sphero as a general control surface. The plug-in required to control the Sphero is installed automatically into Dynamix and its associated control profile is downloaded. Next, the user taps the Apple TV, which similarly installs its associated plug-in and control profile. The T2I application establishes a control configuration that wires the Sphero device as a controller for the Apple TV. Once the Dynamix AC Library sets up the configuration, the Sphero's IMU data are sent to the Apple TV's plug-in, which maps them to its media control system. The user can then pick up the Sphero device and use it as a controller for the Apple TV, as shown in Figure 4 (top). In this configuration, tilting the Sphero forward automatically plays a video on the Apple TV, and tilting the Sphero backwards pauses the video. Rotating the Sphero left and right shuttles the video backwards and forwards respectively. The Apple TV plug-in is also automatically wired to the Phillips Hue plug-in in a feedback configuration, sending `DISPLAY_COLOR` commands to the Phillips Hue plug-in, which dims the lights during playback to simulate a home theater scenario. An overview of this control configuration is shown in Figure 3.

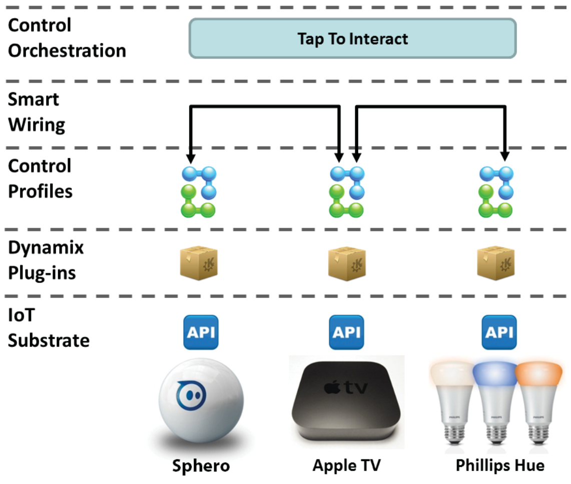

Figure 3: The Sphero/Apple TV/ Hue Control Configuration

Next, the user taps the AR Drone helicopter to install its associated plug-in and control profile. In this case, the T2I application establishes a control configuration that wires the Sphero device as a controller for the helicopter. Once the Dynamix AC Library sets up the configuration, the Sphero's acceleration and IMU data are sent to the AR Drone plug-in, which maps them to its flight control system. The user can then pick up the Sphero device and use it as an intuitive flight controller for the helicopter, as shown in Figure 4 (bottom). In this configuration, when the user shakes the Sphero gently (causing an impact acceleration), the AR Drone plug-in initiates flight and hovers the helicopter approximately one meter above the ground. As the user changes the pitch, yaw and roll of the Sphero, its IMU data are sent to the drone plug-in, which maps them directly to the flight control system of the helicopter – changing the orientation of the helicopter to match the Sphero. Gently shaking the Sphero again automatically lands the helicopter. As shown in Figure 4, Dynamix bridges the network access technologies and application-layer protocols of all Smart Devices connected to the smartphone, enabling them to operate together in ways not previously possible.

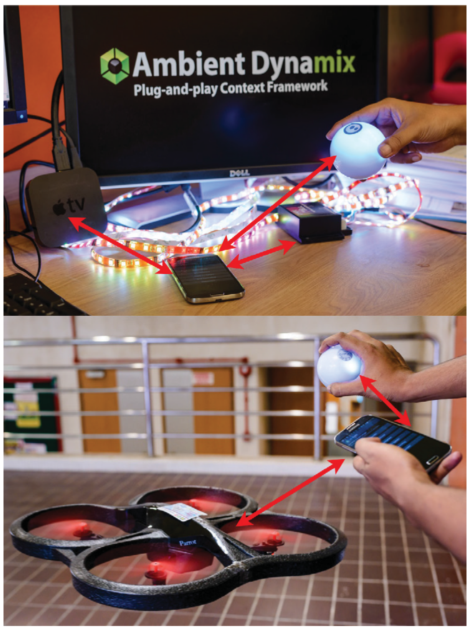

Figure 4: A Sphero Controlling an Apple TV (Top) and a Drone Helicopter (Bottom) using a Dynamix-enabled Mobile Phone as an Adaptive Smart Gateway

3. REFERENCES

[1] P. Middleton, P. Kjeldsen and J. Tully, *Forecast: The Internet of Things, Worldwide, 2013*, Gartner, Inc., 2013.

[2] C. MacGillivray, V. Turner and D. Lund, *Worldwide Internet of Things (IoT) 2013–2020 Forecast: Billions of Things, Trillions of Dollars*, IDC Corporate, 2013.

[3] M. Blackstock and R. Lea, "IoT Mashups with the WoTKit," *Proc. International Conference on the Internet of Things (IoT 2012)*, IEEE, 2012.

[4] D. Carlson and A. Schrader, "Dynamix: An Open Plug-and-Play Context Framework for Android," *Proc. International Conference on the Internet of Things (IoT 2012)*, 2012.

[5] D. Carlson, B. Altakrouri and A. Schrader, "AmbientWeb: Bridging the Web's Cyber-physical Gap," *Proc. International Conference on the Internet of Things (IoT 2012)*, 2012.

[6] D. Guinard, V. Trifa and E. Wilde, "A Resource Oriented Architecture for the Web of Things," *Proc. International Conference on the Internet of Things (IoT 2010)*, IEEE Computer Society, 2010, pp. 1 - 8.

[7] D. Carlson, B. Altakrouri and A. Schrader, "An Ad-hoc Smart Gateway Platform for the Web of Things.," *Proc. IEEE International Conference on Internet of Things (iThings 2013)*, IEEE Computer Society, 2013.

[3] https://github.com/orbotix/Sphero-Android-SDK

[4] https://projects.ardrone.org/projects/show/ardrone-api