Reinventing the Share Button for Physical Spaces

Darren Carlson, Bashar Altakrouri, Andreas Schrader
Ambient Computing Group / Institute of Telematics
University of Luebeck
Luebeck, Germany
{carlson, altakrouri, schrader}@itm.uni-luebeck.de

Abstract—**Imagine if your favorite social networking website provided a "Share to Screen" button that automatically dimmed the room lights and streamed your web-based photos and videos to a nearby television, projector or photo frame. Unfortunately, such fluid interactions between the Web and the physical environment remain challenging, since conventional Web browsers are still unable to directly interact with the broad range of context sources and physical actuators available in highly heterogeneous mobile environments. In our previous work, we presented our plug-and-play context framework, called Ambient Dynamix (Dynamix), which enables advanced context sensing and acting capabilities to be deployed on-demand to mobile devices as plug-ins; enabling entirely new interaction techniques for the Web. To illustrate some of these possibilities, we demonstrate a fully functional Web application that leverages Dynamix plug-ins to stream shared photos and videos from a popular online social network to nearby media rendering devices discovered in the user's physical environment. For a better viewing experience, the demo also adjusts the room's ambient light dimmers through ad-hoc discovery and control of home automation equipment. The demo highlights how Dynamix enables Web applications to discover rich, high-order contextual information, perform context-aware adaptations, and directly influence the physical environment – all from within unmodified mobile Web browsers, such as Google Chrome and Firefox.**

Keywords- Internet of Things; Context-aware Web; Ubiquitous computing; Context middleware; OSGi; REST.

I. INTRODUCTION

Over the last decade, advances in high-speed data networks, consumer electronics and client/server technologies have enabled mobile Web applications (Web apps) to increasingly rival the functionality of conventional mobile applications (installable apps). Although Web apps are bound to the capabilities of the hosting Web browser software (browser), these capabilities are increasingly formidable. Indeed, improved adherence to Web standards, advances in high-performance JavaScript engines and broad support for Asynchronous JavaScript (Ajax) allow Web apps to render dynamic user interfaces, process complex application logic and request additional data asynchronously.

Given the familiarity of Web technologies, many developers are interested in creating context-aware Web apps that are able to fluidly adapt to the needs and circumstances of their users. Contextual information extracted from the user's environment can be used to enable an app to adapt its runtime behavior and capabilities to better fit a user's changing situation and requirements [1]. Due to the complexity of context sensing and acting, middleware is often used to orchestrate context-aware adaptation in mobile applications [2]. Unfortunately, existing mobile context frameworks are often difficult to use, lack appropriate security features, support few context types, and are unable to integrate new (or updated) capabilities at runtime [3]. Further, the security restrictions of modern browsers often prevent (or restrict) Web apps from accessing local system resources, such as sensors, cameras, etc. Finally, hybrid approaches, such Apache Cordova (PhoneGap) [1], cannot interoperate with common browsers (e.g., Chrome) and do not support dynamic runtime extension.

To address these challenges, we are developing Ambient Dynamix (Dynamix), a plug-and-play context framework for Android [4]. Dynamix is a community-based approach for context-aware computing, where advanced context sensing and acting capabilities are deployed *on-demand* to mobile devices as plug-ins, and are made available to installable apps and Web apps (running in common browsers) through only a few lines of code. Dynamix runs as lightweight background service on the user's mobile device, leveraging the device itself as a sensing, processing and communications platform. Dynamix comes with a growing collection of ready-made plug-ins 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[2]. Mobile applications request context support from Dynamix using simple application programming interfaces (APIs). Dynamix automatically discovers, downloads and installs the plug-ins needed for a given context sensing or acting 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 application or framework.

As shown in Figure 1, a Dynamix Service is situated between a device's local hardware and (potentially many) Dynamix apps, which execute in their own runtime processes. Apps communicate with the 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).

[1] cordova.apache.org
[2] ambientdynamix.org

978-1-4244-9529-0/13/$31.00 ©2013 IEEE

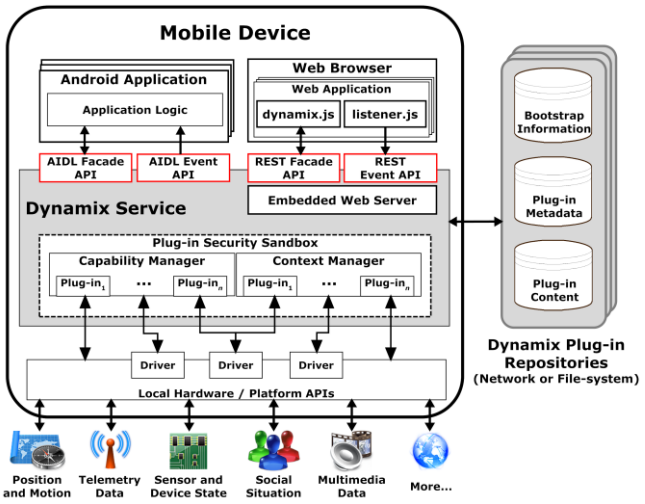

Figure 1: Overview of the Dynamix Framework

Dynamix supports two principle app types: Android apps and Web apps. Android apps are defined as installable native applications that communicate with a local Dynamix service using interfaces based on the Android Interface Definition Language (AIDL). In contrast, Web apps are defined as HTML/CSS/JavaScript-based pages that are hosted by unmodified Web browsers, such a Google Chrome and Firefox. To support communication with browser-based Web apps, Dynamix exposes two REST-based APIs using a customized Web server that is embedded within the Dynamix Service itself (see [5]). Web apps use these APIs to access the same comprehensive set of context features available to Android apps via AIDL. To overcome the same-origin policy restriction[3], Dynamix's embedded Web server is configured to support Cross-Origin Resource Sharing[4], which enables secured cross-site HTTP access between a Web app loaded from one origin (e.g., http://www.example.org) and the local Web server running within the Dynamix Service (e.g., http://localhost).

Browser-based Web apps interact with a local Dynamix Service via Ajax, using two provided JavaScript support files: *dynamix.js* and *listener.js*. Together, these files simplify AmbientWeb app development by encapsulating Dynamix binding, API interactions, data serialization/deserialization, error handling and context event processing. The *dynamix.js* file provides a complete implementation of the Dynamix Façade API, including methods for adding listeners; setting up context support; requesting context information (e.g., heart-rate, indoor positioning); and performing ad-hoc interactions with local or remote computing resources (e.g., controlling home automation equipment). The *listener.js* file provides a complete implementation of the Dynamix Event API, including callback methods for context events, context subscription updates, framework state information, plug-in installation status, etc.

[3] w3.org/Security/wiki/Same_Origin_Policy

[4] w3.org/TR/cors

II. Reinventing the Share Button

The practice of sharing content online is promoted on many social networking websites through the use of the "share button" (or "post button"), which allows users to quickly and easily broadcast content to his or her social network. Many social networking websites also support integrated media hosting, providing a natural platform for sharing personal photos and videos. While online sharing has become commonplace, sharing digital media with people in the same physical vicinity remains a common and frustrating struggle. In many cases, the user is often forced into a long process of social negotiation related to media devices, storage mediums, passwords, software incompatibilities and device cabling. Of course, the complexity of this negotiation process is dramatically increased in foreign environments, such as when visiting a friend's house or lecture hall. Although supportive technologies exist, localized content sharing remains an enduring challenge [6].

To highlight the novel interaction techniques made possible by Dynamix, we created a demo Facebook application that leverages Dynamix enable users to easily stream their shared photos and videos on nearby Digital Living Network Alliance (DLNA)[5] media renderers, such as networked televisions, projectors and photo frames. If a compatible Art-Net home automation controller is available, the demo app can also dim the room's ambient light to help improve the media viewing experience. Aside from the Dynamix Framework and a standard Web browser, the demo requires no additional software or specialized environmental configuration. As shown in Figure 2, our demo scenario includes an Android-based mobile device; an Art-Net light controller with attached light; and a DLNA-certified media renderer.

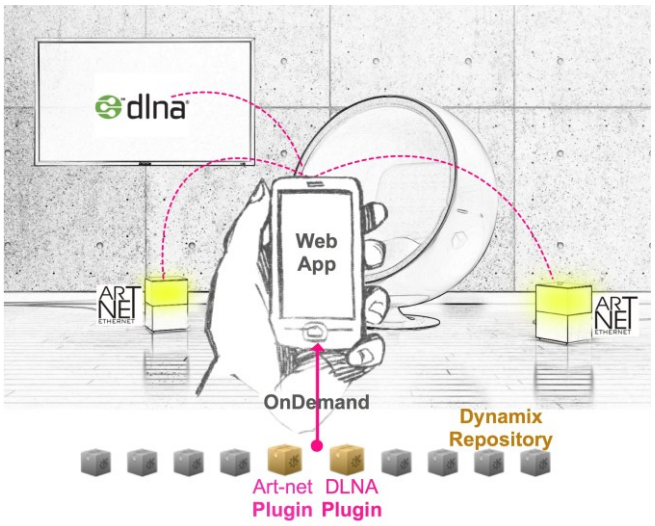

Figure 2: Demo Scenario[6]

[5] dlna.org

[6] Generic mobile and hand image ©Matt Jones, 2008. Image released under the Attribution 2.0 Generic Creative Commons License.

We implemented the demo using a combination of Web, Dynamix and Facebook technologies. The demo is constructed as a Facebook "Canvas" page, which provides an embedded `iframe` that loads necessary HTML, JavaScript and CSS from an external server. We use the Canvas page to load the app's visual content, Facebook JavaScript files, and the previously described Dynamix JavaScript files (dynamix.js and listener.js), which provide a connection between the browser and the local Dynamix Service running on the user's device. Figure 3 shows the demo app running within Facebook.

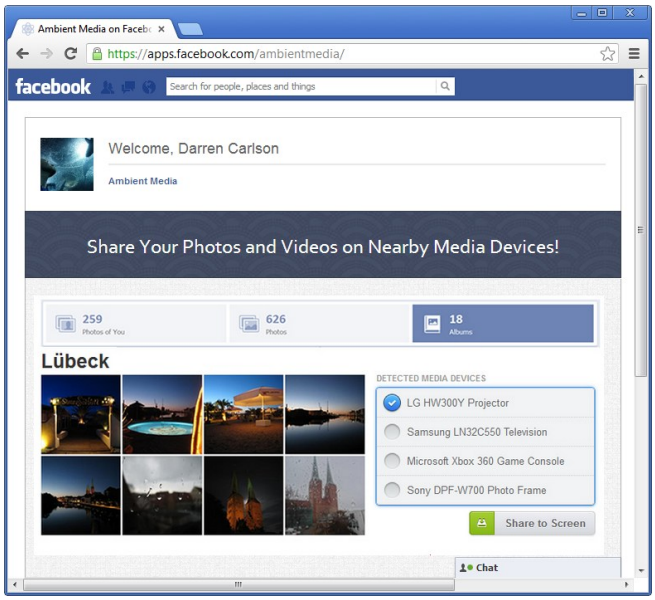

Figure 3: The Demonstration Facebook App

Once loaded into the user's browser, the demo app requests authorization for access to relevant Facebook data, such as shared photos and videos. As shown above, the demo app presents the user's shared photo and video content using a familiar thumbnail view. Next to the media thumbnails, a selectable list of discovered media devices is also displayed. Finally, a "Share to Screen" button is shown below the list of available media devices. Next, the demo app binds to the local Dynamix Service and registers for context support using two plug-ins: 1) The Universal Plug and Play (UPnP) plug-in for discovery and control of nearby media devices; and 2) the AmbientLight plug-in for discovery and control of nearby networked lighting controllers. During this initial interaction, Dynamix searches its configured set of plug-in repositories and automatically installs and activates the requested plug-ins during runtime. Plug-in installation and operation are performed automatically, without the need for user intervention. The installed plug-ins automatically begin scanning the user's local network for available media renderers and light controllers. Information about discovered devices is sent from Dynamix to the demo app using features of the listener.js file.

When the user taps the "Share to Screen" button on the demo app, the room lights are dimmed, and the currently selected media collection is automatically streamed from Facebook's content delivery network to the selected media device, allowing for the easy viewing of shared media with friends in the same room. Room lighting adjustments are accomplished using the AmbientLight plug-in, which uses features of the DMX512-A lighting protocol (not described) to influence the discovered Art-Net control station. Media stream orchestration is accomplished by the Dynamix UPnP plug-in, which sends control commands to the selected media renderer, indicating that it should load the user's media content network addresses (obtained from the Facebook API). Application logic is handled internally by the demo app using JavaScript. Importantly, the demo app does not need to know the specifics of the UPnP or DMX protocols; rather, it sends simple high-level commands to the Dynamix's REST Façade API, which routes these commands to the underlying Dynamix plug-ins.

Funded by the German Federal Ministry of Education and Research.
BMBF Grant Number: 16KT094

III. Conclusion

In this demo, we show how the Dynamix plug-and-play context framework enables the creation of Web applications capable of discovering rich sources of high-order contextual information, performing context-aware adaptation, and influencing the physical environment – all from within the browser. The demo enables users to automatically dim the room lights and easily stream shared photos and videos from Facebook to media devices located in the user's physical environment. The plug-ins required for the demo are automatically discovered and installed by Dynamix during runtime without user intervention. Aside from the Dynamix Framework and a Web browser, no additional software or specialized environmental configuration are required. To the best of our knowledge, sharing social media directly from the browser to discovered physical hardware resources has not been demonstrated before. This demo highlights the endless possibilities for creating hybrid Web applications with Dynamix that combine virtual data with physical resources to create unique and rich user experiences.

References

[1] B.N. Schilit, N. Adams and R. Want, "Context-Aware Computing Applications," Proc. Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, 1994, pp. 85-90.

[2] M. Modahl, et al., "Toward a Standard Ubiquitous Computing Framework," Proc. ACM Workshop on Middleware for Pervasive and Ad-hoc Computing, ACM Press, New York, NY, USA, 2004, pp. 135 - 139.

[3] M. Baldauf, S. Dustdar and F. Rosenberg, "A Survey on Context-aware Systems," International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, no. 4, 2007, pp. 263-277.

[4] D. Carlson and A. Schrader, "Dynamix: An Open Plug-and-Play Context Framework for Android," Proc. 3rd International Conference on the Internet of Things (IoT2012), 2012.

[5] D. Carlson, B. Altakrouri and A. Schrader, "AmbientWeb: Bridging the Web's Cyber-physical Gap," Proc. 3rd International Conference on the Internet of Things (IoT2012), IoT Challenge, 2012.

[6] H. Simo, K. Hannu and R. Jukka, "Leveraging Social Networking Services to Encourage Interaction in Public Spaces," Proc. The 7th International Conference on Mobile and Ubiquitous Multimedia, ACM, 2008.