On the Challenges of Building a Web-based Ubiquitous Application Platform

Heiko Desruelle
Ghent University – IBBT
heiko.desruelle@intec.ugent.be

John Lyle
University of Oxford
john.lyle@cs.ox.ac.uk

Simon Isenberg
BMW Forschung und Technik
simon.isenberg@bmw.de

Frank Gielen
Ghent University – IBBT
frank.gielen@intec.ugent.be

ABSTRACT

People use an increasing number of consumer electronic devices to access their mobile apps. To enhance the applications' immersive user experience, these devices often expose APIs for accessing a wide array of sensors and domain-specific capabilities. Existing mobile application environments, however, only provide limited support for cross-device access of such APIs. To address this limitation, the Webinos platform was designed. Webinos is a virtualized Web-based application platform, aiming to support the collaboration of multiple devices within a single mobile application. In this paper we elaborate on the Webinos platform design. We discuss the encountered design challenges regarding portability, scalability, and privacy, and how these were mitigated.

Author Keywords

mobile applications; ubiquitous web; distributed application platform.

ACM Classification Keywords

D.2.2 Software Engineering: Design Tools and Techniques; H.5.2 Information Interfaces and Presentation: User Interfaces

General Terms

Design; Human Factors.

INTRODUCTION

The diversity and availability of personal computing devices is growing rapidly. In their everyday life, people already use a multitude of consumer electronic devices that are able to run third-party applications [6]. Such devices currently range from desktop PC, to mobile and tablet devices, to home entertainment and even in-car units. From an application development perspective, the greatest common denominator amongst all these devices is the Web. By adopting the Web as an application platform, mobile apps can be made available whenever and wherever the user wants, regardless of the type of device that is being used. Nevertheless these advantages, existing Web application platforms are generally founded on the principles of porting traditional API support and operating system aspects to the Web. The evolution towards large-scale distributed service access and sensor usage is often not supported [2]. In result, the true immersive nature of ubiquitous web applications is mostly left behind.

To enable developers to set up Web applications and services that fade out the physical boundaries of a device, we propose the Webinos platform. Webinos is a virtualized application platform that spans across the various Web-enabled devices owned by an end-user. Webinos integrates the capabilities of these devices by seamlessly enabling the distribution of API requests. We briefly introduce the platform and elaborate on the encountered challenges during the design and prototype implementation phases. The paper is structured as follows. Related work is covered in Section 2. In Section 3, we discuss a vehicular use case scenario that emphasizes the need for a ubiquitous application solution. Furthermore, Section 4 provides a high-level introduction to the Webinos platform and its concepts. Section 5 highlights a number of pertinent challenges faced during the design of Webinos. Finally, our conclusions are presented in Section 6.

RELATED WORK

Despite the prevalence of mobile applications that dynamically use features of multiple devices and sensors, existing work focuses on integrating Web applications more tightly with the local device's resources. Early work goes back to the Lively Kernel experiments for leveraging the Web as a full-fledged application platform [9]. More recently, a number of stable approaches have emerged, exposing a rich set of local APIs through injected JavaScript interfaces and the HTML Document Object Model (DOM). In this context, application developers can turn to Web widget engines such as specified by BONDI/WAC [14], device-independent wrapping frameworks such as PhoneGap [1], and even completely Web-centered operating systems such as Mozilla Boot to Gecko and HP webOS [17].

For distributed application solutions, existing work is still largely confined to specifically targeted platforms and vendors (e.g., Connected TV platforms supporting second screen applications via smartphone devices). As a counter, the Funf

UbiComp12, September 5-8, 2012, Pittsburgh, USA.
ACM 978-1-4503-1224-0/12/09.

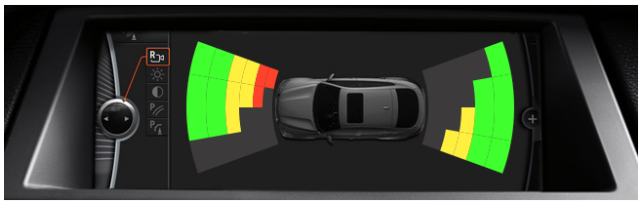

Figure 1. Vehicle use case application. A ubiquitous park distance control (PDC) application able to access a vehicle's internal state

project aims to open up the ecosystem by deploying lightweight sensor probes on mobile devices [4]. The probes' states are aggregated within the cloud. In result, Funf mainly focuses on unidirectional state extraction from the probed devices. The Munin toolkit broadens this scope with a more flexible peer-to-peer design for distributed mobile applications over the Internet [3]. Furthermore, the Gibraltar framework adds up to Munin's approach with security-related design measures and resource usage monitoring [7].

VEHICULAR USE CASE APPLICATION

A Parking distance control (PDC) system enables the assistance of a driver whilst maneuvering a car. Figure 1 depicts a typical PDC application interface. The system relies on monitoring a car's parking sensors and gear status to provide both visual and auditive feedback. Moreover, to avoid distraction, the PDC's application interface is only activated when the car is put in reverse gear (R) and remains active until the gear is put into neutral (N), parking (P), or above second gear.

When bringing this use case to a ubiquitous ecosystem context, various new opportunities arise. From this perspective, PDCs could cover cross-device parking assistance applications. Any mobile device should, e.g., be able to represent a car's dashboard interface by remotely accessing the vehicle's internal state and its sensor data. To ensure maximal interoperability, this type of dashboard application should run on a variety of devices (smartphone, tablet, car head-unit, etc.). In this use case we can distinguish three different setup requirements that need to be supported by the underlying application platform:

- *Local setup:* The PDC application runs within the vehicle's own runtime environment. All data and required APIs are offered by the local application platform. There is no need to access any external device for the application to correctly work.
- *Peer-to-peer setup:* The PDC application is being executed on a secondary device. The executing device needs to access the vehicle's application runtime to retrieve up-to-date information regarding the car's sensor data and gear status. Both devices are connected via a local area network (LAN). The application platform should provide local discovery mechanisms to detect the remote vehicle's runtime and the services it offers. The communication between the two devices is set up through a peer-to-peer connection.
- *Proxy mediated setup:* As with the previous setup, the PDC application is being executed on a secondary device. However, in some cases the devices' runtimes will not be able to establish direct communication (e.g., due to firewall and network address translation (NAT) boundaries). In this case, a trusted proxy should be available to mediate the communication setup between the two devices.

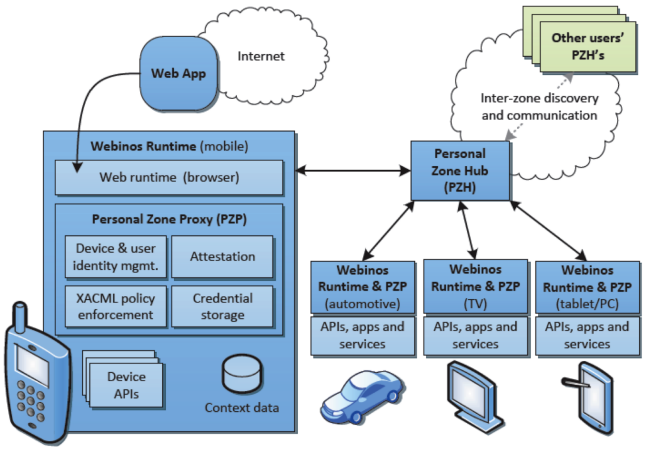

Figure 2. High-level Webinos platform overview. Enabling immersive ubiquitous application support via the federation of context-aware Personal Zones

WEBINOS APPLICATION PLATFORM

Webinos is a service platform project under the European Union's FP7 ICT Programme. The Webinos project specifies a distributed Web runtime, organizing all of a user's devices within a federated hierarchy. The runtime components are distributed over the device, as well as the cloud. Webinos offers developers access to a common set of device APIs. To leverage the availability of device-specific resources and sensors, every local API call can be dispatched off-platform. In case an application's API call can't be served by the executing device, the request is forwarded to a better-suited device. Webinos handles this process seamlessly. The interconnection principle is cornered around the concept of a so called Personal Zone. Figure 2 depicts a high-level overview of the platform's structure and deployment. The Personal Zone constitutes a secure overlay network, virtually grouping a user's personal devices and services.

To enable external access to and from devices in a zone, the Webinos platform defines a centralized proxy component: the Personal Zone Hub (PZH). Each user has his own trusted PZH instance running in the cloud. The PZH keeps track of all personal devices and services in the zone and provides functionality to coordinate their remote discovery and communication. Moreover, PZHs have the ability to request interaction permissions. PZH components are designed to dynamically join a federation of trusted peers, allowing applications to more easily discover and share data and services residing on other people's devices.

On the device-side, a Personal Zone Proxy (PZP) component is deployed. The PZP manages the exposure of local services and handles the direct communication with the zone's PZH. As all external communication goes through the PZP, this component is responsible for acting as a policy enforce-

ment point and authorizing access to the device's available resources. In addition, the PZP is a fundamental component in upholding the Webinos platform's offline usage support. Although the proposed platform is designed with a strong focus on taking benefit from online usage, all devices in the Personal Zone have access to a locally synchronized subset of the data being maintained by the PZH. The PZP can thus temporarily act in place of the PZH in case no reliable Internet connection can be established. This allows users to still operate the basic functionality of their applications, even while being offline and unable to access the Internet. Through communication queuing, all data to and from the PZP is again synchronized with the PZH as soon as the device's Internet access gets restored.

The Web Runtime (WRT) represents the last main component in the Webinos architecture. The WRT can be considered as the extension of a traditional Web render engine (e.g., WebKit, Mozilla Gecko). The WRT contains all necessary components for running and rendering Web applications designed with standardized Web technologies: a HTML parser, JavaScript engine, CSS processor, rendering engine, etc. Furthermore, the WRT maintains a tight binding with the local PZP. The WRT-PZP binding exposes JavaScript interfaces, allowing the WRT to be more powerful than common browser-based application environments. Through this binding, applications running in the Webinos WRT are able to securely interface with both local and remote device services.

DISCUSSION

Privacy and Security

The Webinos platform aims to meet the security and privacy requirements of applications and end-users primarily through a access control policy system. Every access to a Webinos API is mediated by policies, which are enforced by the PZPs on each device as well as in the PZH. This action follows the principle of least privilege, granting applications only the permissions they require. Access policies are set when an application is first installed within the WRT and can be updated by the user subsequently. The policy system is derived from the BONDI/WAC architecture and uses XACML (eXtensible Access Control Markup Language). Furthermore, specific mobile-related adaptations were made, including a number of extensions developed by the PrimeLife project [13]. XACML is a general-purpose access control language for defining policies based on subjects, resources, action and conditions [11]. By including the PrimeLife XACML extensions, the Webinos policy enforcement framework enables users to specify detailed situation-specific access control policies. This is a significant advantage over current web runtime solutions and native mobile application platforms, where once an application has been granted access to a particular asset this access can be reused without further control [8].

The data and services managed by Webinos are often privacy-sensitive, as their analysis might reveal a user's history of actions or the people and devices that have been interacted with. The Webinos platform tends to follow an approach of least surprise, so that a minimum of unexpected data disclosures ought to occur. This is achieved by disabling the collection of contextualized data by default. Where possible, data and actions are filtered to remove unnecessary personal data. The main advantage of the Webinos platform is that context data remains within the Personal Zone and under the control of the end-user. This compares favorably to online user tracking, as users are able to view and manage the data stored about them. Furthermore, applications need to request specific access to this information. The policy example in Listing 1 shows a filter that restricts calendar information access to devices within the same Personal Zone. This case can intuitively be extended to first prompt the Personal Zone owner to grant the request when an external device tries to access the API. Alternatively, the owner can configure the policy to reject external devices by default access without any user intervention.

Listing 1. Webinos XACML calendar access policy

```
<!--Accept requests from within zone-->
<policy combine="first-applicable"
    description="owner">
  <target>
    <subject>
      <subject-match attr="user-id" value=
          "owner"/>
      <subject-match attr="device-id"
          value="*"/>
    </subject>
  </target>
  <rule effect="permit">
    <condition>
      <resource-match attr="api-feature"
          value="http://www.w3.org/ns/api-
          perms/calendar.read"/>
    </condition>
  </rule>
</policy>
```

Portability and Scalability

The Webinos platform's operation heavily relies on service discovery. Webinos has been designed to incorporate two layers of service discovery abstractions. On a local level, Webinos' PZPs support various fine-grained discovery techniques to maximize their capability to detect devices and services. Resources can be discovered through multicast DNS, UPnP, Bluetooth discovery, USB discovery, and RFID communication. Secondly, on a wider-area level, the local discovery data is propagated within the Personal Zone and with authorized external PZHs. Driven by Webinos' aim for flexible Personal Zones in terms of scalability and portability, the overlay network is designed based on an event-driven publish-subscribe pattern. Furthermore, its high-level communication infrastructure is founded on the Extensible Messaging and Presence Protocol (XMPP) over HTTP and WebSockets [10].

In order to validate the portability as well as the scalability qualities of our approach, a proof-of-concept of the proposed platform is implemented. The implementation is made available as part of the Webinos open source project [15]. Based on the project's extensive analysis of the current ubiquitous ecosystem [16], the following prototype platforms have been selected: PC (Linux, Windows, Mac OS X), mobile (Android), vehicles (Linux), home entertainment (Linux). Both PZP and PZH implementations are an extension of the NodeJS platform. NodeJS is an high-performance evented

runtime environment for Google's V8 JavaScript engine [12]. For the vehicular runtime environment, the prototype platform is developed to run on the Pandaboard hardware. This single-board computer is intended for mobile and embedded software development and optimally reflects the resource limitations of in-car head-units.

Furthermore, the presented vehicle use case application is implemented on top of the prototype Webinos platform. The park distance control application is implemented as a regular Web application, using only HTML, CSS, Canvas, and JavaScript. The APIs to communicate with the Webinos platform are provided through JavaScript interfaces by means of the local WRT-PZP binding. As discussed in Section 4, the Personal Zone overlay abstracts the communication between devices. API calls originating from the PDC applications are automatically dispatched by the platform based on decisions made by the policy framework and the available service discovery knowledge. The Webinos platform manages all inter-device communication, which is implemented to use the XMPP protocol running over WebSockets. WebSocket are optimized to reduce communication cost over the Web to a minimum, with a header of only 2 byte compared to the 8 Kbyte header limit for most HTTP messages [5]. Especially on a mobile connection the implementation of WebSocket communication shows its use. The median latency for a Personal Zone containing 10 devices stays well below 25ms in a WiFi LAN environment, below 125ms over a 3G mobile network connection, and around 200ms over an Edge cellular connection. Even at a throughput rate of more than 100 messages/s, e.g., for communicating sensor data updates.

CONCLUSION

In this paper we presented the Webinos application platform, aiming to enable immersive ubiquitous software applications by leveraging some of the cross-platform possibilities of the Web. The platform utilizes the Web infrastructure to establish its Personal Zone concept, a virtual overlay network for grouping a user's devices and their set of available services. From this perspective, Webinos aims to be a key enabler in the realization of ubiquitous applications that are able to execute across the physical boundaries of devices. The first phase of Webinos' design has focused on ensuring specific quality requirements. A number of architectural tactics were applied in order to obtain acceptable quality attribute support for portability, scalability, and privacy.

The presented architecture only represents a first milestone in the pursuit of true ubiquitous application convergence. Whilst Webinos provides structured access to a remote-able and Web-based API platform, it is still the application developers' responsibility to incorporate the necessary logic that allows their applications to act accordingly. Therefore, future work should include research on extending the Webinos platform with (semi-) automated application adaptation mechanisms. This process should be driven by a high degree of context-awareness regarding the user, his devices, and the surrounding environment.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7-ICT-2009-5, Objective 1.2) under grant agreement number 257103 (Webinos project).

REFERENCES

1. Charland, A. and Leroux, B. Mobile application development: web vs. native. *Communications of the ACM*, 54, 5 (2011) 49-53.
2. Desruelle, H., Blomme, D., Gionis, G. and Gielen, F. Adaptive user interface support for ubiquitous computing environments. In *Proc. UIDL 2011*, Thales Research and Technology (2011), 107-113.
3. Elmqvist, N. Munin: a peer-to-peer middleware for ubiquitous visualization spaces. In *Proc. DUI 2011*, University of Castilla-La Mancha (2011), 17-20.
4. Funf Open Sensing Framework. `http://funf.media.mit.edu/`.
5. Gutwin, C.A., Lippold, M. and Graham, T.C. Real-time groupware in the browser: testing the performance of web-based networking. In *Proc. CSCS 2011*, ACM Press (2011), 167-176.
6. Holzer, A. and Ondrus, J. Mobile application market: A developers perspective. *Telematics and Informatics*, 28 (2011), 22-31.
7. Lin, K., Chu, D., Mickens, J., Zhuang, L., Zhao, F. and Qiu, J. Gibraltar: Exposing Hardware Devices to Web Pages Using AJAX. In *Proc. WebApps 2012*, USENIX (2012).
8. Lyle, J., Monteleone, S., Faily, S., Patti D. and Ricciato F. Cross-platform access control for mobile web applications. In *Proc. POLICY 2012*, IEEE Press (2012).
9. Mikkonen, T. and Taivalsaari, A. Creating a Mobile Web Application Platform: The Lively Kernel Experiences. In *Proc. SAC 2009*, ACM Press (2009), 177-184.
10. Paterson, I., Smith, D., Saint-Andre P. and Moffitt, J. XEP-0124: Bidirectional-streams Over Synchronous HTTP (BOSH). XMPP (2010).
11. Rissanen, E. (ed.). eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS (2010).
12. Tilkov, S. and Vinoski, S. Node.js: Using JavaScript to build high-performance network programs. *Internet Computing*, 14, 6 (2010) 80-83.
13. Trabelsi, S. and Njeh, A. Policy Implementation in XACML. In *Privacy and Identity Management for Life*, Springer (2011), 335-374.
14. Wholesale Application Community. `http://www.wacapps.net/`.
15. Webinos Developer Portal. `http://developer.webinos.org/`.
16. Webinos consortium. Industry landscape, governance, licensing and IPR frameworks, Tech. Rep. D2.3 (2011).
17. Weiss, A. WebOS: say goodbye to desktop applications. *Networker*, 9, 4 (2005), 18-26.