Noname manuscript No.
(will be inserted by the editor)

Multi-device Application Middleware: Leveraging the Ubiquity of the Web with Webinos

Heiko Desruelle · Simon Isenberg · John Lyle · Frank Gielen

Received: date / Accepted: date

Abstract The broad range of connected devices has turned the Internet into a ubiquitous concept. In addition to desktop and laptop PCs, the Internet currently connects mobile devices, home entertainment systems, and even in-car units. From this ubiquitous evolution towards sensor-rich devices, the opportunity arises for various new types of innovative software applications. However, alongside rises the issue of managing the increasing diversity of device characteristics and capabilities. As device fragmentation grows, application developers are facing the need to cover a wider variety of target devices and usage scenarios. In result, maintaining a viable balance between development costs and market coverage has turned out to be an important challenge when developing applications for a ubiquitous ecosystem. In this article, we present the webinos platform, a distributed Web runtime platform that leverages the Web for supporting self-adaptive cross-device applications. In order to enable the development of such immersive ubiquitous applications, we introduce and evaluate the concept of a context-aware federated overlay architecture.

Keywords Ubiquitous Web · Context-awareness · Multi-device applications · Webinos

1 Introduction

The diversity of personal computing devices is increasing at an incredible pace. People are nowadays using a multitude of consumer electronic devices that

H. Desruelle · F. Gielen
Department of Information Technology, Ghent University – iMinds, Ghent, Belgium
E-mail: heiko.desruelle@intec.ugent.be

S. Isenberg
BMW Forschung und Technik GmbH, Munich, Germany

J. Lyle
Department of Computer Science, University of Oxford, Oxford, UK

have the ability to run third-party developed applications. Such devices currently range from desktop PC, to mobile and tablet devices, to home entertainment and even in-car units [27]. However, the fragmentation of devices and usage contexts makes it for applications particularly difficult to target a broad segment of devices and end-users. From a development perspective, the greatest common denominator amongst the available multi-device approaches is the Web [11]. By adopting the Web as an application platform, applications can be made available whenever and wherever the user wants, regardless of the device type that is being used.

Nevertheless these clear 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 [16]. 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, the webinos platform has been proposed. 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.

The remainder of this article is structured as follows. Section 2 discusses background and related work regarding the evolution and challenges in Web engineering. Furthermore, we explore the complexity and various dimensions of context-aware application engineering in a ubiquitous environment. Section 3 provides a general overview of our proposed federated application platform. We introduce the concept of context-aware Personal Zones, through which webinos aims to enable developers to create applications that transcend a device's physical boundaries. Section 4 elaborates on the platform's structures and models for setting up secure context-awareness support. Section 5 covers a use case on exploiting the capabilities of the proposed platform. We present the case of developing an immersive ubiquitous application that enables users to access the internal state of their vehicle, regardless of the device that is being used to run the application. In Section 6 we subsequently cover the implementation and evaluation of both the platform and the proof-of-concept application. Finally the conclusion and future work are outlined in Section 7.

2 Background and Related Work

2.1 The Web as an Application Platform

The development and deployment of ubiquitous applications introduces an important series of resource consuming requirements [4]. The combinations of existing hardware characteristics, operating systems, software frameworks, etc. are virtually endless. For software developers, this diversity has turned out to be a double-barreled asset. It provides consumers the freedom to execute

applications at will across various devices. On the other hand, the device diversity asset heavily fragments the application's delivery targets. By the absence of a general native development solution, developers often have no alternative than to create and maintain a set of device-dependent versions of their applications. Hence, ensuring a viable balance between development costs and an application's market coverage will more than ever become a challenging issue. Against this backdrop, the use of Web technologies for application development purposes has proven to be a viable long-term solution [25]. Through years of standardization efforts and the wide adoption of languages such as HTML, CSS, and JavaScript, the Web can be deployed as a powerful foundation for universal application development and delivery. Moreover, running on top of the Internet infrastructure the Web-based application paradigm is rapidly gaining momentum amongst developers.

Web-based application development approaches have been explored from various perspectives. Developers can opt for pure Web applications, running in a standard browser environment. However, due to the sandboxed execution of browsers this approach drastically limits the available APIs to the underlying device. In turn, a hybrid Web application approach was introduced providing developers access to a richer API set, whilst still maintaining most of the cross-platform advantages of pure Web applications. This type of application is still built using Web technology, but no longer uses the browser as its client-side runtime environment. A separate client-side Web runtime framework is deployed to bridge the gap between native and Web applications by granting the application access to most device APIs. Hybrid Web applications are currently being developed using Web widget engines such as provided by the BONDI/WAC initiatives [44], device-independent frameworks such as the PhoneGap [8] and Appcelerator [2] application wrappers, and even completely Web-centered operating systems such as Chromium OS [40] and Open webOS [43].

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 project aims to open up the ecosystem by deploying light-weight sensor probes on mobile devices [28]. 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 [18]. Furthermore, the Gibraltar framework adds up to Munin's approach with security-related design measures and resource usage monitoring [26].

These existing hybrid Web application solutions,however, only partially succeed in enabling a ubiquitous user experience [36]. Their main focus lies with porting traditional API support and operating system aspects to the Web. Applications built upon these old principles result in virtual silos, unable to truly cross the physical boundaries of a device. By neglecting the evolution towards distributed user interfaces and adaptive context-aware application behavior, the true immersive nature of ubiquitous computing is mostly left

behind[14,15]. The absence of elaborate context-awareness is a key element driving this issue. In order for ubiquitous applications to adaptively support various contextual situations, the underlying application platform needs to provide structured, as well as secure and up-to-date access to the user's contextual setting. This requirement is not just limited to providing access to a detailed description of the target delivery context (screen size, interaction methods, available sensors, etc). Moreover, structured access to details regarding the user context (personal preferences, social context, disabilities, etc.) and the physical environment (location, time, etc.) ought to be supported.

2.2 Modeling Ubiquitous Context-awareness

The availability of detailed and reliable metadata regarding a user's contextual situation provides an important driver for enabling rich ubiquitous applications. The exact entities represented by this contextual information can be of a very dynamic nature, potentially affecting the consumer's expectations towards the application's user interface, behavior, content, etc. In general, context-aware computing identifies five contextual categories: the device context, user context, environment context, time context, and historical context [9,37].

The device context describes the characteristics of the target device that is being used to access the application. A ubiquitous ecosystem covers a diversity of screen sizes, interaction methods, software support, etc. In Web-based environments, the device capabilities are generally retrieved through Resource Description Framework (RDF) devices profiles, i.e., User Agent Profile (UAProf) [29] and Composite Capability/Preference Profiles (CC/PP) [24]. The necessary device identification step in this process is handled through HTTP header user agent matching. In order to facilitate the collection and aggregation of these device profiles, the W3C Mobile Web Initiative (MWI) standardized the Device Description Repository specification (DDR). The specification provides an API and its associated vocabulary for structured access to context providers services [33]. In essence, a DDR thus provides a standardized means for retrieving contextual information about a-priori knowledge on the characteristics of a particular target device or Web runtime. Various open as well as proprietary DDR implementations are actively being maintained. Most notably OpenDDR [30], WURFL [31], and DeviceAtlas [17].

In a ubiquitous setting, the end-user's profile description gains more and more importance. Besides exposing information on user preferences and specific experience, this model should also comprise knowledge regarding the user's specific abilities and disabilities. E.g., enabling accessibility requirements for providing support to elderly people, and people with disabilities. From this perspective, Heckmann proposed the GUMO formalism as a general user model ontology for representing generic user descriptions using the Web Ontology Language semantics (OWL) [21]. The current challenge in this domain is modeling the enormous amount of parameters and relationships

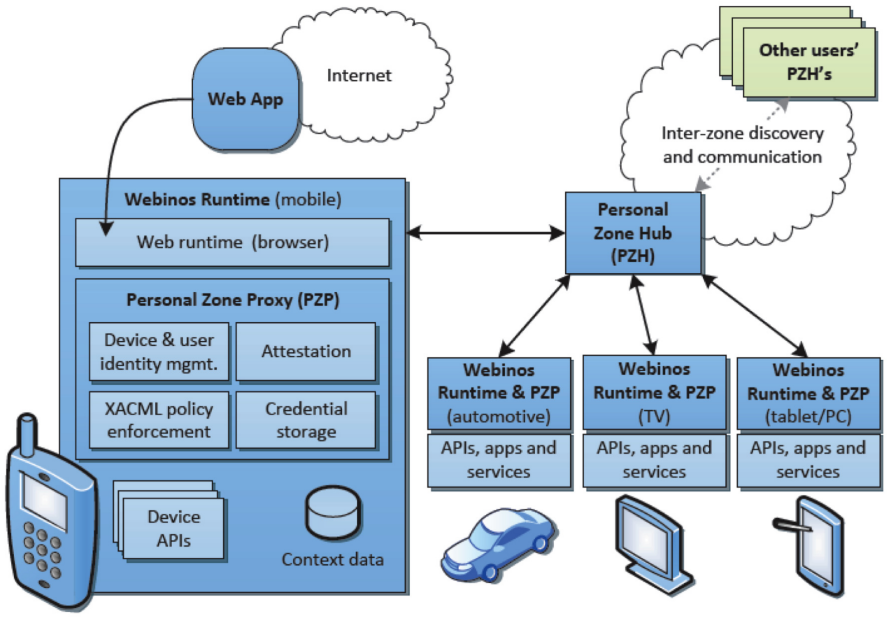

Fig. 1 High-level overview of webinos ubiquitous application platform

that characterize the user context [35]. To overcome this issue, forces are being joined with other ontology-driven projects such as Linked Data [20] and UbisWorld [22].

The environment-, time-, and historical context aspects define where, how, and when the interaction between the user and an application is exactly taking place. The environment context is specified by observing the numerous sensors available on the user's device (e.g., location, temperatures, network service discovery, the level of background noise, etc.). Furthermore, the notion of time and historical context is not to be neglected. As context is a dynamic concept, support for temporal patterns recognition and management is needed. The W3C Ubiquitous Web Domain is currently in the process of standardizing the Delivery Context Ontology specification (DCO) [6]. The DCO provides a formal model of the characteristics of the environment in which devices, applications, and services are operating.

3 The Webinos Application Platform

To enable application developers to set up services that fade out the physical boundaries of a device, we propose the webinos platform. Webinos defines a federated Web application platform and its runtime components are distributed over the devices, as well as the cloud. Fig. 1 depicts a high-level overview of the platform's structure and deployment. The system's seamless

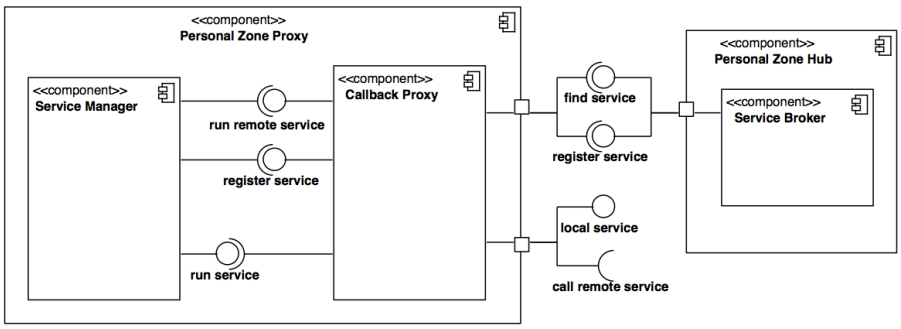

Fig. 2 Callback broker system architecture for the discovery and management of remote services within a Personal Zone

interconnection principle is cornered around the notion of a so called Personal Zone.

The Personal Zone represents a secure overlay network, virtually grouping a user's personal devices and services. To enable external access to and from the devices in this zone, the webinos platform defines a centralized Personal Zone Hub (PZH) component. Each user has his own PZH instance running in the cloud. The PZH is a key element in this architecture, as it contains a centralized repository of all devices and contextual data in the Personal Zone. The PZH keeps track of all services in the zone and provides functionality to enable their discovery and mutual communication. This way, the PZH facilitates cross-device interaction with personal services over the Internet [1]. Moreover, PZHs are federated, allowing applications to easily discover and share data and services residing on other people's devices. Webinos achieves this structure by incorporating two service discovery abstraction mechanisms. On a local level, webinos supports various fine-grained discovery techniques to maximize its capability to detect devices and services (e.g., through multicast DNS, UPnP, Bluetooth discovery, USB discovery, RFID/NFC, etc.). Secondly, on a remote level, the local discovery data is propagated within the Personal Zone and with authorized external PZHs. Based on webinos' aim for flexible Personal Zones in terms of scalability and modifiability, the overlay network is designed in line with the callback broker system pattern [5] (see Fig. 2). With this pattern, the availability of locally exposed services is communicated throughout the Personal Zone via a service broker in the PZH. Moreover, the platform's high-level communication infrastructure is founded on a direct handle tactic via JSON-RPC (JavaScript object notation - remote procedure call), which is invoked over HTTP and WebSockets [23]. This in order to bypass the risk of the service broker becoming a communication bottleneck (see use case description in Section 5 for more details on the communication handle).

On the device-side, a Personal Zone Proxy (PZP) component is deployed. The PZP abstracts the local service discovery and handles the direct communication with the zone's other devices and PZH. As all external communication goes through the PZP, this component is responsible for acting as a policy en-

forcement point and managing the access to the device's exposed resources [12]. 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 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. In result, the synchronized data allows the local PZP to avoid the constant need for PZH-mediated communication between devices via its ability to set up a peer-to-peer (P2P) based connections with PZPs in the same local network. 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 an extension to traditional Web rendering engines (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 much more powerful than traditional browser-based application environments. Through this binding, applications running in the WRT are able to securely interface with the APIs and services offered by devices throughout the user's Personal Zone.

4 A Context-aware Personal Zone

The innovative nature of the proposed platform lies with webinos' capability to establish a cross-device, cross-service, cross-user overlay network. For this Personal Zone concept to be successfully adopted by ubiquitous application developers, the platform needs to provide these developer access to a rich at-runtime overview of the user's contextual setting. As stipulated in Section 2.2, elaborate platform support for transparent context management is vital. In this section we provide more detail on the available developer tools for setting up secure context-awareness within a webinos Personal Zone.

4.1 Delivery Context Model

The webinos delivery context model is defined to span all the platform's contextual knowledge within the user's Personal Zone. The model builds upon the W3C's Delivery Context Ontology (DCO) specification [6] and the Context of Use (COU) model proposed by the NEXOF-RA Project [7]. The webinos context delivery model can be represented as the quad-tuple

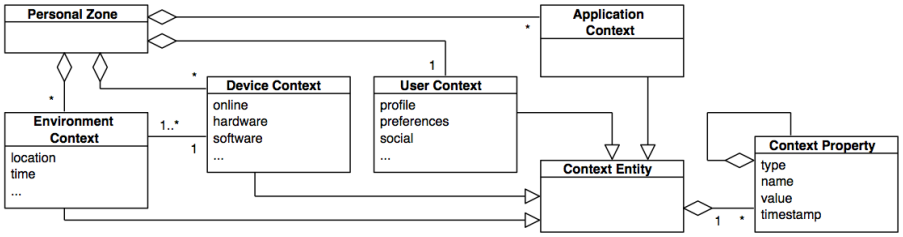

Fig. 3 Simplified webinos Personal Zone context model

$$C = \{U, D, E, A\}\,. \tag{1}$$

The context model thus comprises four sub models: U denotes the user context, D denotes the device context, E denotes the environment context, and the application context is denoted by A (see Fig. 3). The first three sub models are internally managed and updated by the webinos platform, whilst the application context model for additional application-specific knowledge is to be maintained by the application developer. The contextual information regarding the Personal Zone's owner is described by the user context model. This model consists of an aggregation of user profile data, user preferences, social context information, etc. Furthermore, each device and its physical environment are described by a separate instance of respectively the device context model and the environment context model. A device context model comprises knowledge regarding the corresponding device's availability and status in the Personal Zone, hardware characteristics, supported software, etc. The environment context model contains a description of a certain device's location, surrounding noise levels, etc. Lastly, the application context model provides developers the freedom to store a number of contextual properties, describing a situation from the perspective of their application.

Each of the above described models consists of a dynamic set of context properties

$$\forall(\Phi \in C) : \Phi = \{p_{a_1}, p_{a_2}, \ldots, p_{a_n}\}\,, \tag{2}$$

where the properties p_{a_i} keep track of the historical evolution of context attributes a_i by maintaining a timestamped list of their associated context values

$$p_{a_i} = (\tau_1 : v_1, \tau_2 : v_2, \ldots)\,. \tag{3}$$

The values v represent the state of attribute a_i within the context model at a certain time τ. The historical values are stored in support of the context framework's mechanisms for historical evaluation, pattern detection and reasoning, and the implementation of conflict resolution strategies.

4.2 Context Aggregation and Retrieval

The webinos context framework is built on top of the above described context models. As depicted in Fig. 1, this framework is one of the core services provided by webinos. The context framework provides functionality for acquiring, storing, inferring new knowledge, and granting external access to contextualized data. Web applications running within the webinos WRT, as well as other background webinos services, can rely on the context framework to support their at-runtime need for contextualized information.

The context acquisition process is autonomously managed by the context framework and operates completely transparent for both the user and the application developers. As discussed in Section 3, a broker-based communication and synchronization architecture has been put in place between the device PZPs and the centralized PZH. The webinos context framework utilizes this structure to retrieve contextual knowledge from within the Personal Zone. The context framework hooks into the PZP's synchronization mechanism. These events are intercepted by the framework's context acquisition component and subsequently filtered for relevant data through rule-based reasoning. To physically store the extracted knowledge, the framework maintains a mapping between the webinos context models and a triplestore infrastructure. Moreover, the context framework is closely coupled with the PZP's security and policy enforcement framework. As will be discussed in Section 4.3, this binding ensures the secure handling of context data that is being stored as well as accessed.

Listing 1 Webinos context retrieval API

```
if ("webinos" in window)
{
   // request service broker a handle for the context API
   webinos.findServices(
      "http://webinos.org/api/context.query",
      function(service) {
           // execute SPARQL context query
      }, null);
}
```

For application developers seeking to create context-aware ubiquitous applications, the context framework provides an API for accessing Personal Zone wide contextual information. The API supports two access modes for retrieving context information: a generic query mode, and a change subscription mode. The generic query mode allows applications to execute targeted queries for specific context data in the storage system. The change subscription mode, on the other hand, enables an application to subscribe for specific context update events. These events are triggered by the context framework when new contextual knowledge is acquired or inferred through reasoning. The code example in Listing 1 demonstrates an application's access to the context retrieval API via webinos' service broker mechanism. The application can then query start

querying the context store once a handle for the API has been returned. The context API supports W3C's standardized SPARQL RDF query language for unambiguously querying context knowledgebases [32]. Context queries can be structured along the knowledge represented by the four webinos context models. Listing 2 shows an example query for accessing all vehicular sensor data within the user's Personal Zone. All context API requests are passed to a query processor component. The processor parses the request and checks its execution rights in collaboration with the PZP's policy enforcement framework. In case the request is granted by the PZP, the query is optimized and dispatched for execution.

Listing 2 Context query vehicular sensor status

```
PREFIX webinos: <http://www.webinos.org/api/context/>

SELECT ?sensor ?reading
WHERE {
   ?device a webinos:Vehicle.
   ?device webinos:sensor ?sensor.
   ?sensor webinos:reading ?reading.
}
```

4.3 Policy Enforcement Support

The webinos platform aims to meet the security and privacy requirements of applications and end users primarily by means of an access control policy system. Every access to a webinos API is mediated by policies and is enforced by the PZP on each device as well as in the central PZH. This action follows the principle of minimal privilege, granting applications only the permissions they require. Furthermore, the policy system allows webinos to securely dispatch API access to specific devices within the Personal Zone. The policy system is derived from the BONDI/WAC architecture [44] and uses XACML (eXtensible Access Control Markup Language) including a number of mobile extensions developed by the PrimeLife project [39]. XACML is a general-purpose access control language for defining policies based on subjects, resources, action and conditions [34]. By including the PrimeLife XACML extensions, webinos' policy enforcement framework allows users to specify detailed situation-specific access and dispatching control policies. This is a significant advantage over current Web runtime solutions and native application platforms. Once an application has been granted access to a particular asset this access can easily be reused without further control.

Context data is often privacy-sensitive, as its analysis might reveal a user's history of actions or the people and devices that have been interacted with [10]. The webinos platform aims 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 most context data by default, and providing the

user a simple interfaces to turn it on again, complete with feedback about the kind of data that is being shared and stored. Where possible, data is filtered to remove unnecessary personal data. The main advantage of the webinos platform is that context data remains within the 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, and applications will have to request specific access to this information. The policy example in Listing 3 shows a filter to restrict vehicular information access to devices within the same Personal Zone (lines 1-14). In case an external device tries to access this data, the Personal Zone owner will first be prompted to grant the request (lines 16-23). Alternatively, the owner can configure the policy to reject external devices by default access without any user intervention.

Listing 3 Webinos XACML vehicular API access policy

```
1  <!--Accept requests from devices within zone-->
2  <policy combine="first-applicable" description="owner">
3     <target>
4        <subject>
5           <subject-match attr="user-id" value="owner"/>
6           <subject-match attr="device-id" value="*"/>
7        </subject>
8     </target>
9     <rule effect="permit">
10       <condition>
11          <resource-match attr="api-feature" value="http://webinos.org/api/
                vehicle"/>
12       </condition>
13    </rule>
14 </policy>
15
16 <!--Prompt owner for all other requests from external devices-->
17 <policy combine="first-applicable" description="untrusted">
18    <rule effect="prompt-oneshot">
19       <condition>
20          <resource-match attr="api-feature" value="http://webinos.org/api/
                vehicle"/>
21       </condition>
22    </rule>
23 </policy>
```

5 Use Case: Vehicle Interaction

To further elaborate on the application possibilities of the webinos platform, we present an application use case that has been built with the platform. The use case covers a cross-device parking assistance application that is able to access the user's internal vehicle status through the webinos Personal Zone. Within the ubiquitous ecosystem, vehicles can be valuable data providers. As elaborated in the previous sections, this type of data source can enable developers to build a completely new breed of distributed applications. Fig. 4 depicts the webinos park distance control (PDC) application. The application

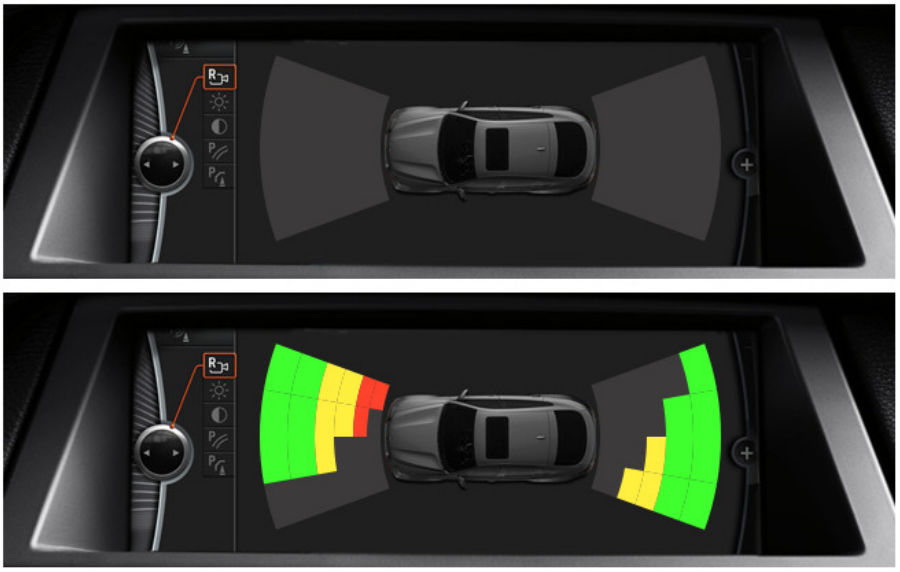

Fig. 4 Vehicle interaction use case. A ubiquitous park distance control (PDC) application able to access a vehicles internal state

is capable of assisting a driver whilst maneuvering a car by monitoring its parking sensors and gear status. The PDC application interface is 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 2nd gear. The application provides visual feedback based on the car's parking sensor status. This Web application can be run on any device in the user's Personal Zone (smartphone, tablet, car headunit, etc.) and is able to retrieve its sensor data by accessing the webinos context API. In this use case we distinguish three different device setups supported by the webinos platform:

- Local setup: The PDC application runs within the vehicle's own webinos WRT. All required APIs and data are offered by the local PZP, without webinos having to access an external PZP instance.
- Peer-to-peer PZP setup: The PDC application is being executed on a secondary webinos device. The executing device's PZP needs to access the vehicle's PZP in case its own context knowledgebase does not contain up-to-date data on the car's sensor data. Both devices are connected via a local area network (LAN). Webinos' local discovery mechanisms are able to detect the vehicle's PZP and the services it offers. The communication between the two PZPs is set up through a direct peer-to-peer connection.
- PZH mediated setup: As with the previous setup, the PDC application is being executed on a secondary webinos device. However, in some cases the devices' PZPs will not be able to establish direct communicate, due to firewall and network address translation (NAT) boundaries. In this case, the PZ is used to mediate the communication setup between the two PZPs.

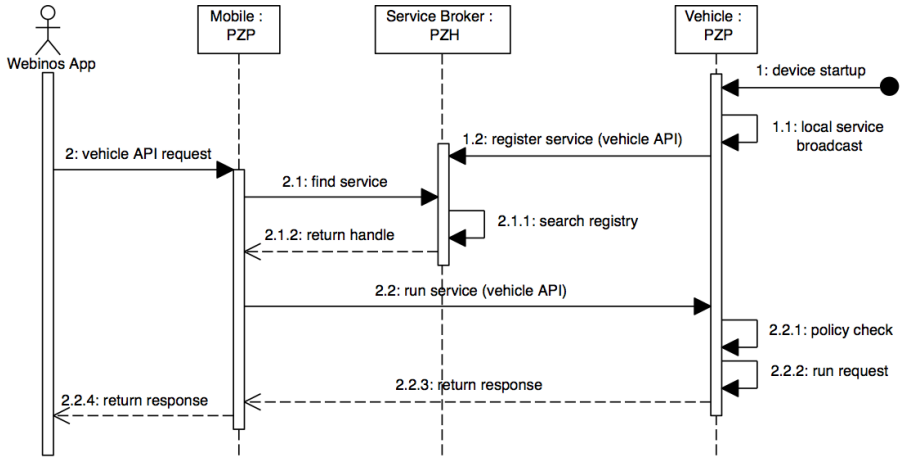

Fig. 5 Webinos' transparant overlay network for accessing remote API capabilities via the platform's callback broker system mechanism

The UML sequence diagram in Fig. 5 demonstrates how the PZH mediated setup process for dispatched vehicular API calls operates. At startup time, the vehicle's PZP starts by broadcasting its exposed APIs and services to devices within local network reach. Moreover, the PZP registers these services with the central PZH in order to ensure synchronization over the entire Personal Zone. Next, the user can use his/her mobile device to launch the webinos-enabled PDC application. As webinos offers location transparency with regards to APIs and services, the application does not need handle the communication with remote devices in order to access external services (such as the vehicular API). The mobile PZP will identify a vehicular API call as a remote request and ask the Service Broker in the PZH for a device within the Personal Zone with matching capabilities. Based on the returned handle, the mobile PZP will set up a connection with the vehicle's PZP and dispatch the application's API call. The vehicle's PZP will subsequently check its configured access policies whether or not to exectute the incoming request (see Listing 3). In case the request is granted, the vehicle PZP will execute the call locally and return results to the mobile PZP, which are in turn passed to the PDC application.

6 Evaluation

A prototype of the proposed platform has been implemented and made available as part of the webinos open source project [42]. Based on the project's extensive analysis of the current ubiquitous ecosystem [41], the following prototype platforms have been selected: PC (Linux, Windows, MacOS), mobile (Android), vehicles (Linux), home entertainment (Linux), machine-to-machine (Arduino). The PZP as well as PZH prototypes all extend NodeJS, an high-performance evented runtime for Google's V8 JavaScript engine [38]. For the

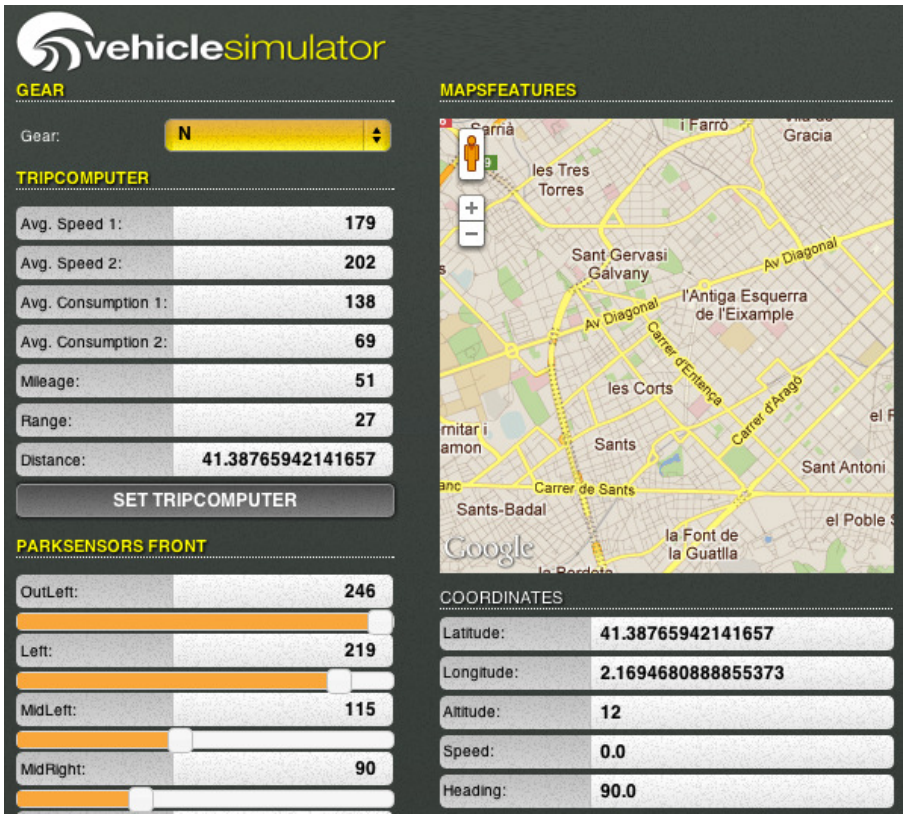

Fig. 6 Webinos PZP vehicle simulator. Supporting application developers to easily simulate vehicle data through a Web interface

vehicle implementation, the prototype platform is developed on Pandaboard hardware, as it reflects the resource limitations of in-car headunits and is indented for mobile and embedded software development. The headunit hosts a car's navigation system, as well as portable media interaction, and connects to the built-in GSM/CDMA modem, the rear-seat entertainment system, etc. Application access to the system internals is controlled through the webinos context API and policy framework. To enable the easy evaluation of webinos-enabled in-car applications, a vehicle simulator was added to the project. The simulator emulates a vehicle PZP and allows an application developer to simulate the various internal states and parameters of a car (see Fig. 6).

The context framework is implemented as a platform-independent NodeJs module. The designed context models are specified using W3C's Web Ontology Language (OWL) [13]. Moreover, the framework integrates Jena [3], a toolkit for semantic storage, rule-based reasoning, and SPARQL query access on the Web. By applying the RETE algorithm for executing rule-based reasoning within the context framework as well as the policy enforcement point, the worst case computational complexity of this process remains linear

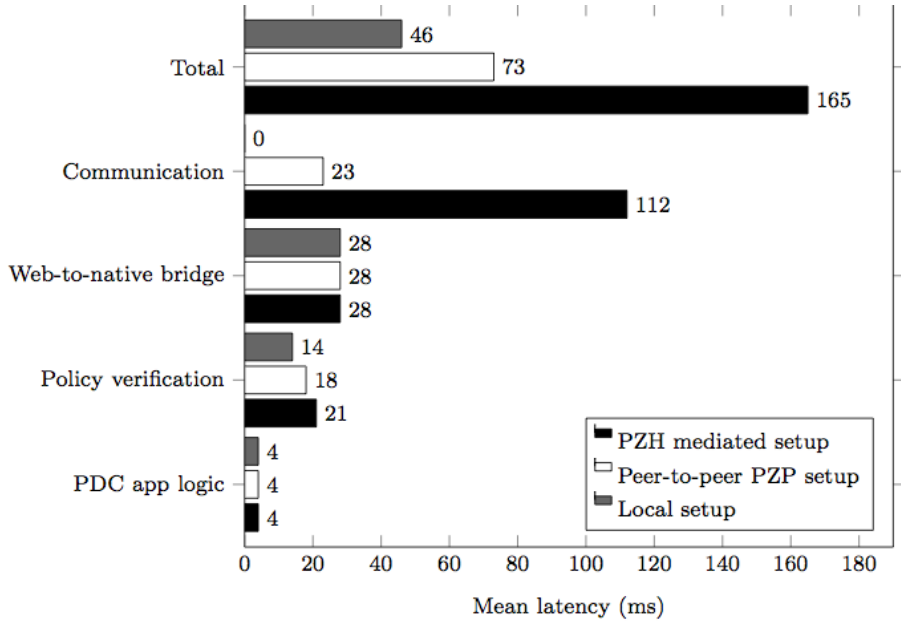

Fig. 7 Comparing the mean latency breakdown for sensor data access requests initiated by the PDC application

$$O\left(P \cdot W^{C}\right) , \tag{4}$$

with P the average number of rules or policies, W the number of facts in the knowledgebase, and C the average number of conditions in each rule [19]. Our performance analysis on the actual implementation confirm these theoretical predictions. For a policy base containing 50 policies, the policy framework is able to reach conclusions within 20ms (see Fig. 7). For context reasoning, however, the knowledgebase is generally much larger. Our context framework prototype is designed to run a context store containing between 1.000 and 5.000 context facts. Reasoning time about this amount of data increases linearly and takes between 500 and 2.500ms.

The vehicle use case application is implemented on top of the prototype webinos platform. The park distance control application is designed as a regular Web application, using HTML, CSS, Canvas, and JavaScript. In addition, the APIs with the webinos platform are provided through JavaScript interfaces by the local PZP. Fig 7 depicts the mean latency breakdown for the PDC application to access vehicular sensor data. The graph covers the three device setup scenarios discussed in Section 5. The Personal Zone overlay abstracts the communication between devices. API calls by the PDC applications are automatically dispatched by the platform based on decisions made by the policy framework and the available knowledge within the context framework. The webinos platform manages the inter-device communication, which is im-

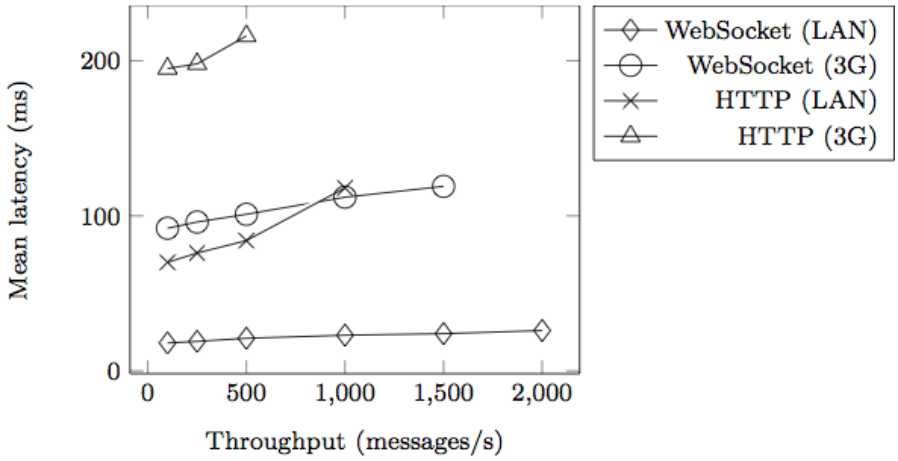

Fig. 8 Webinos inter-device communication over WebSocket interfaces. Mean latency in terms of 512b message throughput and connection type

plemented to use the JSON-RPC protocol running over WebSockets. Fig. 8 visualizes the communication latencies for 512b messages between the webinos PZPs on a Android smartphone and a Pandaboard vehicle instance. For reference, the latencies for the same type of communication over HTTP are included, as WebSockets are optimized to reduce communication overhead to a minimum, with a header of only 2 byte compared to the 8 Kbyte header for conventional HTTP messages. Especially on a mobile connection the implementation of WebSocket communication shows its use. The median latency for a typical Personal Zone containing 5 devices stays well below 25ms in a LAN environment and below 125ms over a 3G mobile network connection. Even at a throughput rate of more than 100 messages/s, e.g., for communicating sensor data updates.

7 Conclusion and Future Work

In this article we presented the webinos application platform, aiming to enable immersive ubiquitous software applications by leveraging the cross-platform possibilities of the Web. The proposed platform utilizes the Internet infrastructure to establish its Personal Zone concept, a virtual overlay network for grouping all of the user's devices and available services. Through the federated structure of Personal Zones, webinos is able to provide application developers access to elaborate at-runtime context data regarding the current user, his devices, and the surrounding environment. The availability of this information allows developers to more accurately anticipate to a user's contextual situation. The webinos platform's context-awareness enables numerous applications that make full use of the diversity and interconnectivity of devices. 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.

Although the webinos project addresses challenging issues in the ubiquitous application development domain, the current architecture only represents a first milestone in the pursuit of true ubiquitous application convergence. Whilst the webinos platform provides structured access to rich contextual knowledge, 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, driven by the platform's rich context-awareness. Regarding the privacy and security impact of such an application runtime, there will undoubtedly be a need to further experiment with user interfaces. This in order to strike an acceptable balance between the advantages that context sensitivity can offer, as well as privacy and user and developer convenience.

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) and number 258348.

References

1. Aikebaier A, Enokido T, Takizawa M (2011) Trustworthy Group Making Algorithm in Distributed Systems. HCIS 2011, 1(6). doi: 10.1186/2192-1962-1-6
2. Allen S, Graupera V, Lundrigan L (2010) Pro Smartphone Cross-Platform Development. Apress, New York
3. Apache Software Foundation (2012) Apache Jena project. `http://incubator.apache.org/jena`. Accessed 10 December 2012
4. Banavar G, Bernstein A (2004) Challenges in design and software infrastructure for ubiquitous computing applications. Advances in computers 62(1):179-202. doi: 10.1016/S0065-2458(03)62004-8
5. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M (2001) Pattern-oriented software architecture: A system of patterns. John Wiley & Sons, West Sussex
6. Cantera JM, Rhys L (2010) Delivery Context Ontology. W3C Working Group Note. `http://www.w3.org/TR/dcontology`. Accessed 10 December 2012
7. Cantera JM, Tsouroulas N (2010) Context model and universal APIs. NEXOF-RA. `http://www.nexof-ra.eu`. Accessed 10 December 2012
8. Charland A, Leroux B (2011) Mobile application development: web vs. native. Communications of the ACM 54(5):49-53. doi: 10.1145/1941487.1941504
9. Chen G, Kotz D (2000) A Survey of Context-aware Mobile Computing Research. Technical report TR2000-381, Dept. of Computer Science, Dartmouth College
10. Chuan D, Lin Y, Linru M, Yua C (2011) Towards a Practical and Scalable Trusted Software Dissemination System. JoC 2(1):53-60
11. Cozza R, Zimmermann A, Milanesi C, De La Vergne HJ, Sato A, Lu CL, Glenn D, Nguyen TH, Shen S, Gupt A (2011) Market Share: Mobile Communication Devices by Region and Country. Gartner inc.
12. Dafir Ech-Cherif El Kettani M, En-Nasry B (2011) MIdM : an Open Architecture for Mobile Identity Management. JoC 2(2):25-32
13. Dean M, Schreiber G (2004) OWL Web Ontology Language Reference. W3C Recommendation. `http://www.w3.org/TR/owl-ref`. Accessed 10 December 2012

14. Desruelle H, Blomme D, Gielen F (2011) Adaptive Mobile Web Applications: A Quantitative Evaluation Approach. In: Proceedings of the 11th International Conference on Web Engineering (ICWE 2011). Springer, Heidelberg, pp 375-378. doi: 10.1007/978-3-642-22233-7_29
15. Desruelle H, Blomme D, Gionis G, Gielen F (2011) Adaptive user interface support for ubiquitous computing environments. In: Proceedings of the 2nd International Workshop on User Interface eXtensible Markup, Thales Research and Technology, Paris
16. Desruelle H, Lyle J, Isenberg S, Gielen F (2012) On the challenges of building a Web-based ubiquitous application platform. In: Proceedings of the 14th ACM International Conference on Ubiquitous Computing, ACM, New York, pp 733-736
17. DotMobi (2012) DeviceAtlas. `https://deviceatlas.com`. Accessed 10 December 2012
18. Elmqvist N (2011) Munin: a peer-to-peer middleware for ubiquitous visualization spaces. In: Proceedings of the 1st Workshop on Distributed User Interfaces (DUI 2011). University of Castilla-La Mancha, pp 17-20
19. Forgy F (1976) On the efficient implementation of production systems Dissertation, Carnegie-Mellon University
20. Heath T, Bizer C (2011) Linked data: Evolving the web into a global data space. Synthesis Lectures on the Semantic Web: Theory and Technology, 1:1-136. doi: 10.2200/S00334ED1V01Y201102WBE001
21. Heckmann D (2005) Ubiquitous User Modeling. Dissertation, Dept. of Computer Science, Saarland University
22. Heckmann D, Loskyll M, Math R, Recktenwald P, Stahl C (2009) Ubisworld 3.0: A Semantic Tool Set for Ubiquitous User Modeling. In: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization, Springer, Heidelberg
23. JSON-RPC Working Group (2011) JSON-RPC 2.0 Specification. `http://json-rpc.org`. Accessed 10 December 2012
24. Kiss C (2010) Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0. W3C Working Group Note. `http://www.w3.org/TR/CCPP-struct-vocab2`. Accessed 10 December 2012
25. Lawton G (2008) Moving the OS to the Web. Computer, 41(3):16-19. doi: 10.1109/MC.2008.94
26. Lin K, Chu D, Mickens J, Zhuang L, Zhao F, Qiu J (2012) Gibraltar: Exposing Hardware Devices to Web Pages Using AJAX. In: Proceedings of the 3rd USENIX conference on Web Application Development (WebApps 2012). USENIX Association, Berkeley
27. Lyle J, Faily S, Flechais I, Paul A, Goker A, Myrhaug H, Desruelle H, Martin A (2012) On the design and development of webinos : a distributed mobile application middleware. In: Proceedings of the 12th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2012). Springer, Heidelberg, pp 140-147. doi: 10.1007/978-3-642-30823-9
28. MIT Media Lab (2011) Funf Open Sensing Framework. `http://funf.media.mit.edu/`. Accessed 10 December 2012
29. Open Mobile Alliance (2001) WAG UAProf. Technical report WAP-248-UAProf-20010530. `http://www1.wapforum.org/tech/terms.asp?doc=WAP-248-UAProf-20010530-p.pdf`. Accessed 10 December 2012
30. OpenDDR (2012) Version 1.0. `http://www.openddr.org`. Accessed 10 December 2012
31. Passani L (2012) WURFL: Mobile device database. `http://wurfl.sourceforge.net`. Accessed 10 December 2012
32. Prud'hommeaux E, Seaborne A (2008) SPARQL Query Language for RDF W3C Recommendation. `http://www.w3.org/TR/rdf-sparql-query`. Accessed 10 December 2012
33. Rabin J, Trasatti A, Hanrahan R (2012) Device Description Repository Core Vocabulary W3C Working Group Note. `http://www.w3.org/TR/ddr-core-vocabulary`. Accessed 10 December 2012
34. Rissanen E (2010) eXtensible Access Control Markup Language (XACML) Version 3.0 OASIS standard. `http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cd-03-en.pdf`. Accessed 10 December 2012
35. Silva JLT, Moreto Ribeiro A, Boff E, Primo T, Viccari RM (2011) A Reference Ontology for Profile Representation in Communities of Practice. Metadata and Semantic Research. 240:68-79. doi: 10.1007/978-3-642-24731-6_7

36. Taivalsaari A, Mikkonen T (2011) The Web as an Application Platform: The Saga Continues. In: Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE Press, New York, pp 170-174
37. Teraoka T (2012) Organization and exploration of heterogeneous personal data collected in daily life. HCIS 2012, 2(1). doi: 10.1186/2192-1962-2-1
38. Tilkov S, Vinoski S (2010) Node. js: Using JavaScript to build high-performance network programs. Internet Computing, 14(6):80-83. doi: 10.1109/MIC.2010.145
39. Trabelsi S, Njeh A (2011) Policy Implementation in XACML. Privacy and Identity Management for Life. Springer, Heidelberg
40. Tsai WT, Shao Q, Sun X, Elston J (2010) Real-Time Service-Oriented Cloud Computing. In: Proceedings of the 6th World Congress on Services, IEEE Press, New York, pp 473-478
41. Webinos (2011) Industry landscape, governance, licensing and IPR frameworks. Technical report D2.3
42. Webinos (2012) Webinos Developer Portal. `https://developer.webinos.org/`. Accessed 10 December 2012
43. Weiss, A (2005) WebOS: say goodbye to desktop applications. Networker, 9(4):18-26. doi: 10.1145/1103940.1103941
44. Wholesale Application Community (2010) WAC Home. `http://www.wacapps.net/`. Accessed 10 December 2012