Implementation of Asynchronous Mobile Web Services
Implementation and First Usage

Marc Jansen[1,4], Javier Miranda[2] and Juan Manuel Murillo[3]
[1]*Computer Science Institute, University of Applied Sciences Ruhr West, Tannenstr. 43, Bottrop, Germany*
[2]*GLOIN S.L. Cáceres, Spain*
[3]*Software Engineering Department, University of Extremadura, Badajoz, Spain*
[4]*Department of Media Technology, Linnaeus University, Växjö, Sweden*

Keywords: Mobile Devices, Web Services, Asynchronous Services.

Abstract: Mobile devices are nowadays used almost ubiquitously by a large number of users. 2013 was the first year in which the number of sold mobile devices (tablet computers and mobile phones) outperformed the number of PCs' sold. And this trend seems to be continuing in the coming years. Additionally, the scenarios in which these kinds of devices are used, grow almost day by day. Another trend in modern landscapes is the idea of Cloud Computing, that basically allows for a very flexible provision of computational services to customers. Yet, these two trends are not well connected. Of course there exists already quite a large amount of mobile applications (apps) that utilize Cloud Computing based services. The other way round, that mobile devices provide one of the building blocks for the provision of Cloud Computing based services is not well established yet. Therefore, this paper concentrates on an extension of an implementation that allows to provide standardized Web Services, as one of the building blocks for Cloud Computing, on mobile devices. The extension hereby consists of a new approach that now also allows to provide asynchronous Web Services on mobile devices, in contrast to synchronous ones. Additionally, this paper also illustrates how the described implementation was already used in an app provided by a business partner.

1 INTRODUCTION

Nowadays, the success of the Cloud Computing business model cannot be ignored. This is mainly due to the proliferation of small services deployed in the cloud that are consumed massively. Proof of that is the impressive economic growth experimented by companies like Facebook (Facebook, 2013) Instagram (Meijer, 2013), or Dropbox (Dropbox, 2013). Many studies show that, increasingly, this consumption is generated from mobile devices. As an example, Gartner predicts that mobile application development projects targeting smartphones and tablets will outnumber native PC projects by a ratio of 4:1 by 2015 (Gartner, 2012). Undoubtedly this fact is favoured by the effort expended by manufacturers to provide mobile devices with capabilities that were hardly imaginable a few years ago.

However, the growing capabilities of mobile devices and the interest they arouse in the industry and consumers contrast with the secondary role to which they are relegated in cloud applications. Cloud services are conventionally based on pure client-server architectures where mobile devices are always assumed as clients. Thus, although current mobile devices have enough power and capabilities to support architectures in which they could also behave as service providers (SaaS), not many efforts have been done in designing such infrastructures.

Thinking about mobile devices as service providers opens new horizons for the industry of small cloud services (Raatikainen, et al., 2012 and Guillén, et al., 2014). Nevertheless, the roadmap for getting mobile devices providing services is not without difficulties related to important aspects such as security, privacy or mechanisms to deploy services in them. In particular, this paper focuses on the implementation needed to deploy Web Services on mobile devices.

In an effort to get mobile devices capable of providing SaaS some works (McFadden, et al., 2003, Srirama, et al., 2006 and AlShahwan, et al., 2010) have already proposed platforms to deploy Web

Jansen M., Miranda J. and Murillo J..
Implementation of Asynchronous Mobile Web Services - Implementation and First Usage .
DOI: 10.5220/0004948906450651
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 645-651
ISBN: 978-989-758-019-2

Services in them. Although such proposals have made a valuable contribution, they still do not cover specific issues such as the continuous connections and disconnections of mobile devices with the subsequent renewals of IPs. Moreover, while the above proposals cover the essential aspects of the Web Services they still left uncovered other aspects that have proved very useful such as the support for both synchronous and asynchronous invocations.

In previous works (Jansen, 2013a, Jansen, 2013b, Guillén, et al., 2014 and Miranda, et al., 2013) the authors presented an implementation that supports the deployment of Web Services in mobile devices. The use of proxies enabled support for connection and reconnection of devices with IP renewals. Now, this paper focuses on the necessity of both synchronous and asynchronous invocations and how they can be enabled in mobile devices. This equals mobile devices with the potential offered by conventional servers for deployment and invocation of Web Services calls. Thus, mobile devices are provided with capabilities not exploited so far opening the door to a new generation of cloud applications where mobile devices gain relevance. As an example it is shown how the proposed implementation has been used to build a M2M messaging application.

The outline of the paper is as follows. Section 2 presents the context of the work. In section 3 the necessity and benefits of both synchronous and asynchronous Web Services invocations in mobile devices is discussed. Section 4 proposes an implementation of both invocation methods. Section 5 presents an experience of using the proposal for the implementation of a M2M messaging app. Finally, section 6 summarizes the conclusions and future works.

2 STATE OF THE ART

The idea of providing Web Services on mobile devices was probably presented first by IBM (McFaddin, et al., 2003). This work presents a solution for a specific scenario where Web Services are hosted on mobile devices. More general approaches for providing Web Services on mobile devices are presented in (Srirama, et al., 2006) and (AlShahwan, et al., 2010). In (Li and Chou, 2011), another approach, focusing on the optimization of the HTTP protocol for mobile Web Services provisioning, is presented. Importantly, none of the mentioned approaches manages to overcome certain limitations of mobile devices, as demonstrated in the next section.

Yet, these approaches do not overcome certain limitations of mobile devices, e.g., permanently changing networks, IP addresses from networks with network address translation (NAT) or the fact that mobile devices are usually not designed to be always online (might be switched off, might have not network connection, …).

An additional approach that covers these problems, is presented in (Jansen, 2013a). This approach utilizes a central proxy infrastructure, that allows on the one hand to cover the mentioned problems and on the other hand establishes a stable infrastructure for mobile device to provide standardized Web Services. As we will see in the following sections, this central proxy infrastructure could easily be extended in order to provide also asynchronous Web Services on mobile devices.

Additionally, the work presented in (Jansen, 2013b) argues for a new perspective to Web Services especially if those services are deployed to mobile devices.

An alternative proposal for deploying services in Mobile devices has been presented in (Guillén, et al., 2014 and Miranda, et al., 2013). This work focuses not only on technical issues but also on the kind of services that could be served from mobile devices and the new generation of applications they would enable.

All the above works let the authors see the potential of having Web Service in mobile devices and the necessity of having them fully featured, including asynchronous calls.

After contextualizing the presented work in the context of the current research, the next section will start describing the basic difference between synchronous and asynchronous service requests.

3 ASYNCHRONOUS VERSUS SYNCHRONOUS WEB SERVICES

In both, programming and the theory of distributed systems, a differentiation between synchronous and asynchronous tasks is well defined. This section provides a short overview about this differentiation in the first subsection.

Furthermore, based on the description of the differentiation synchronous and asynchronous calls, the necessity of asynchronous calls with respect to Web Services provided on mobile devices is argued.

3.1 Asynchronous versus Synchronous Services

In distributed systems, the differentiation between an asynchronous and a synchronous call to a certain service is well defined. In case of a synchronous service call, the service consumer waits for the answer of the service provider, so the service result, before the service consumer continues its tasks. Figure 1 shows the flow of calls for a synchronous service request between a process p1 (server provider) and a process p2 (service consumer).

Figure 1: A synchronous service request between a service provider (p1) and a service consumer (p2).

Therefore, the more time the calculation of the service result takes for the provider of the service, the more time the service consumer waits until it continues with its current tasks.

In contrast, by performing an asynchronous service request, the service consumer is not waiting until the service provider answers the request, but just sends the request for the execution of the service to the service provider and immediately continues working on remaining tasks. In a later step, when the service provider has finally calculated the outcome of the service request, the service provider sends the result back to the service consumer. The information flow for an asynchronous service requests between a service provider (p1) and a service consumer (p2) is shown in Figure 2.

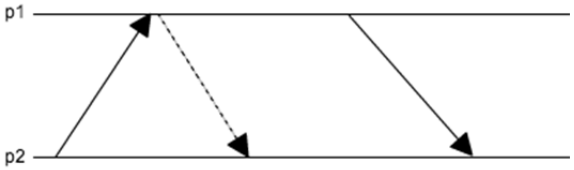

Figure 2: An asynchronous service request between a service provider (p1) and a service consumer (p2).

For retrieving the result of an asynchronous service request, the service consumer usually defines a function that retrieves the result of the requested service. This function then controls also the further actions necessary for handling the result of the service request. The function responsible for handling the result of an asynchronous service request is usually called a callback function.

In programming languages, the difference between an asynchronous and a synchronous call to a method or a function is also well known. If a program performs a usual (synchronous) call to a method, the program continues working in this method and the rest of the program is only executed after the program finished the method call and provided the return value. Of course, programs can outsource computational expensive method calls in sub processes/threads, so that the program continues with the rest of its tasks while the method is executed in parallel.

On the other hand, many programming languages (or frameworks based on usually synchronous programming languages) allow asynchronous requests to methods/functions. One of the probably most prominent examples for this approach is JavaScript. Especially for the development of Web 2.0 applications, the idea of AJAX (Asynchronous JavaScript and XML) (Powell, 2008) came up. Here, method calls to JavaScript methods/functions are performed in an asynchronous manner that allows the browser to continue with its current task. By using this approach, the AJAX application running in a browser, behaves pretty much the same way as a usual desktop application with respect to performance and user feedback. For similar reasons, also the currently most prominent server side implementation for JavaScript, NodeJS (Tilkov and Vinoski, 2010), also integrates an asynchronous approach for method calls. This allows a non-blocking and single threaded approach for the server side implementation of services, that is known for providing increased performance (Tilkov and Vinoski, 2010) in comparison to multi threaded approaches, e.g., this is the major reason that proposed solutions for the C10K problem (Kegel, 2013) almost completely rely on asynchronous approaches.

3.2 The Necessity of Asynchronous Mobile Web Services

Synchronous mobile Web Services represent a valid approach when the required time by a mobile device to process each request is low enough to be solved and responded back to the client before it gets a timeout exception message. In this sense, there are several use cases that could be implemented using this type of mobile Web Service, like those that involve simple information retrieval or processing,

that is, when such information is available immediately (usually, it has been previously calculated and stored in the device, like an image, a contact name, or a text message).

Unfortunately, mobile devices are exposed to continuous context changes that may affect the time lapse required for solving specific types of request, especially those that depend on information provided by external resources or sensors. This implies that the response time for such requests is unpredictable at design time, and a proper behaviour at run-time cannot be ensured using the synchronous approach.

In order to illustrate the necessity of asynchronous mobile Web Services, we have considered, at least, the following potential use cases of mobile Web Services we could deploy in the current state-of-the-art of mobile applications:

- Services that offer information fetched from an external device/sensor. Gathering information from sensors (like GPS or Bluetooth) is usually implemented by mobile operative systems using asynchronous methods with their callbacks.
- Services that depend on device's user interaction. Mobile devices could serve human-based services that allow asking users about anything, including OCR or captcha-like challenges. They can be answered at any time, so the callback response with the user's choice would be sent asynchronously.
- Services offering information that needs to be evaluated gathered or processed during a long period of time. We can include in this category use cases related with measurement and tracking, like "determining the distance the user will travel in the next five minutes" using the accelerometer sensor information for tracking it, or "taking the temperature average for the next 2 hours" tracking it from a temperature sensor.
- Services that depends on a particular context to be processed. This category could include services offering information resulted from some interaction with an external device or resource, like: getting information from the car handsfree system or a domotic interface; heavy processing task that requires the device to be plugged-in to a power supply; or a service that require higher connectivity bandwidth to be properly executed.

These use cases that involve mobile Web Services and require an asynchronous behaviour suggest that synchronous mobile Web Services are not enough for supporting more complex services, and it motivates the proposal of this approach, which implementation is described on the next section.

4 IMPLEMENTATION

Following the requirements formulated in the last section, the approach described in (Jansen, 2013a) was extended by the possibility to also provide asynchronous calls to Web Services deployed to mobile devices. This section describes the implementation of these asynchronous mobile Web Services.

Here, the basic idea for the implementation of these asynchronous mobile Web Services was to be as close as possible at the solution for synchronous Web Services, which basically relies on two major parts:

1) JSR 181 compatible annotations that allow to marker a class as an implementation of mobile Web Services and methods as mobile Web Methods.
2) A proxy infrastructure that dynamically generates implementations of a façade (Gamma, et al., 1995) for the mobile Web Services in order to overcome certain limitations of mobile devices, with respect to the provision of services.

Additionally, from a technical point of view, another requirement for the implementation was to re-use already existing possibilities and not to invent new techniques whenever possible. Therefore, the architecture for synchronous mobile Web Services, as shown in Figure 3, was analysed.

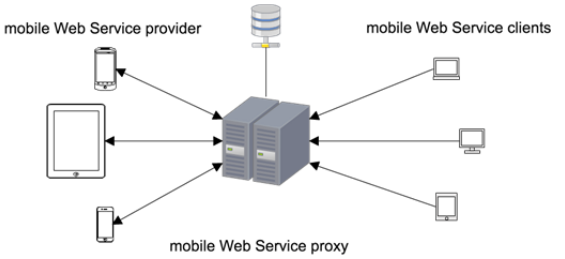

Figure 3: Architecture for providing synchronous mobile Web Services.

In the centre of Figure 3, the mobile Web Service proxy could be seen, which basically provides the façade for the Web Services actually running on the mobile devices in the left part of Figure 3. Therefore, the requests to the performed by the Web Service clients are not directly addressed to the mobile devices, but to the proxy infrastructure.

After the detailed analysis of this architecture, the following strategy was developed for the implementation of asynchronous mobile Web Services. First of all, a new annotation was integrated in the existing framework (@AsynchronousMobileWebMethod) for the

implementation of a method that should be executed on a mobile device as an asynchronous mobile Web Method.

Additionally, the proxy was modified so that it is able to handle also asynchronous requests, next to the synchronous ones. Here, the idea was to provide a callback for every request to an asynchronous mobile Web Service, as shown in Figure 4.

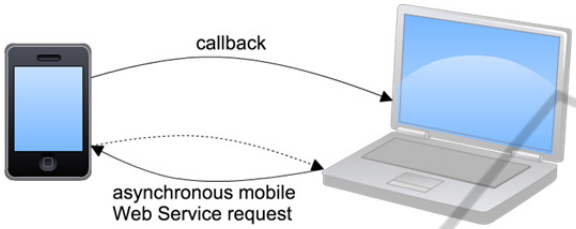

Figure 4: An asynchronous mobile Web Service request.

This callback itself was implemented as a reference to a Web Service that was responsible for handling the result of the asynchronous request to the mobile Web Service. Therefore, the reference to the callback is usually a URL to a WSDL (Web Service Description Language) of the service that handles the result of the asynchronous request.

By this, the method/function handling the result could either be implemented on a mobile device (using the implementation for mobile Web Services described in (Jansen, 2013a)) or it could be implemented as a usual Web Service running on a usual computer/server.

Interestingly, from a developers point of view, the implementation of an asynchronous mobile Web Method does not differ from the implementation of a synchronous mobile Web Method, especially no reference to a callback needs to be provided. All the necessary extensions for the handling of asynchronous requests could automatically be applied to the façade created by the central proxy.

The next section provides a description about a mobile application that already utilizes the described approach for asynchronous mobile Web Services deployed to mobile devices. By the successful integration of this implementation, the mobile application was able to achieve the requirements discussed earlier in this paper.

5 INTEGRATION OF ASYNCHRONOUS MOBILE WEB SERVICES INTO AN EXISTING APPLICATION

The proposed implementation of asynchronous mobile Web Services has been tested in *beeFun*, a particular messaging application.

This mobile app treats to reinvent the traditional messaging concept into a new, different and funny experience. In order to make it attractive for users, *beeFun* combines a simpler interface for sending different preconfigured (or not) messages very quickly with several gaming experiences like weekly-renewed scores and badges obtainment. Different templates of message are available to be sent, including simple text, text with images, location-based messages and surveys. Messages can be sent to a given group of known users, or a specifying set of criteria that define such receivers, like their current location, age, or gender, among others. Unlike other similar apps, this information is stored exclusively on each device, so those criteria are evaluated in every device when this kind of message is sent. This design, in addition to bringing a different perspective on mobile devices as service providers similar to the Software-as-a-Service model (but in a mobile context), allows sending geo-located messages without forcing users to be tracked neither updating their information to a centralized server continuously, solving certain privacy issues in a very elegant way.

Consequently, this architecture requires a fast and flexible way to connect the server-side (which orchestrates every message's request and response) with mobile devices (which evaluate the conditionals and accept or decline every message depending on their current status). Technically, this was implemented through an in-house solution that combines the use of Push Notification Services for "waking up" mobile devices when new messages arrive, and a RESTful-based server API infrastructure for handling both requests and responses to and from mobile devices. Figure 5 shows the main flow of interactions between server and devices when a location-based message is sent.

Due to the asynchronous nature of the location-based messaging of *beeFun*, it represents a useful testbed for integrating the asynchronous mobile Web Services framework. In *beeFun*, devices' location is fetched from the Location Service provided by the mobile operating system, and it is done asynchronously for avoiding deadlocks until a valid

position is obtained from sensors (GPS, WiFi, etc.). Actually, not only location but also any information gathered by *beeFun*, including history of locations, answers to surveys and so, conform the information core component of the app, and can be considered as a mobile Web Service itself with several Web Methods for getting information about the device or its owner.

Following that consideration, the integration of the framework has changed the previous design of the messaging component behaviour from the diagram described in Figure 5 to the new flow illustrated in Figure 6.

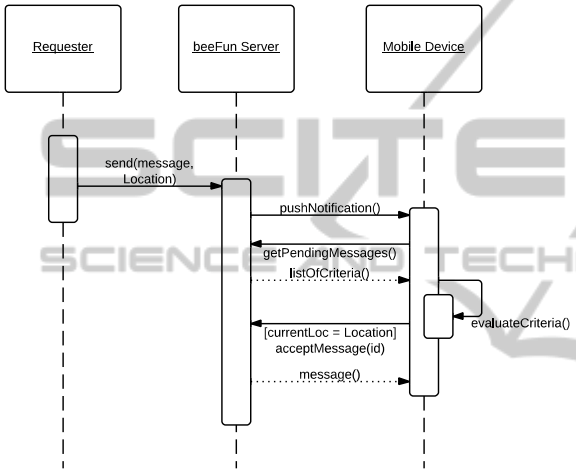

Figure 5: Sequence diagram of the location-based messaging implementation without the asynchronous mobile Web Service integration.

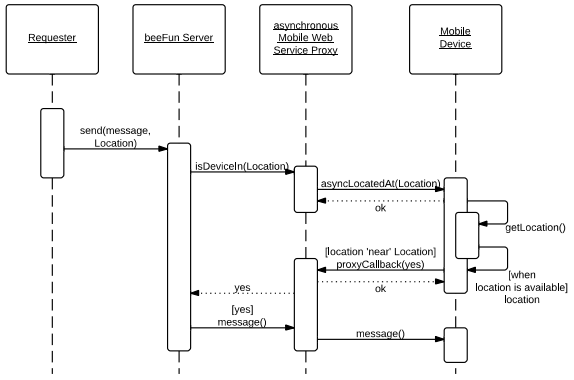

Figure 6: Location-based messaging flow using the asynchronous mobile Web Service approach.

Both figures represent the same request for sending a location-based message that is evaluated and accepted by the device. However, the integration of the asynchronous mobile Web Services framework greatly simplifies the complexity of the outgoing protocol between the *beeFun* server and mobile devices in comparison with the previous non-standardized approach.

Regarding the code, the implementation of the mobile-side is also clearer thanks to the separation of concerns derived from the adoption of the proposed class hierarchy and its annotations, both required for implementing mobile Web Services and Web Methods properly.

Additionally, the use of the asynchronous mobile Web Services platform can easily be extended to other parts of *beeFun* services with minimal effort due to the standardization of the business rules between server and mobile.

6 DISCUSSION AND OUTLOOK

This paper has presented a proposal to enable Web Services deployment in mobile devices allowing both synchronous and asynchronous calls. It has also described the experience of using the implementation to build *beeFun,* a M2M messaging application. *beeFun* is currently being commercially exploited by the startup GLOIN[1].

Having the possibility of deploying web services to mobile devices raises the concept of Mobile Clouds to a higher level. It not only opens the door to a new generation of cloud applications in which mobile devices acquire the capabilities of providing SaaS. In addition, it also enables new business flows benefiting the device owners.

Currently the authors are releasing new applications based on the principles presented in the article[2]. The new application areas are allowing them to detect new features to be incorporated into the technological platform.

Additionally, the described implementation also allows to implement new scenarios, e.g., in the area of mobile health applications, in which aspects like security and privacy, but also de-central storage of sensual personal data, could be handled much easier in comparison if other technologies are used. Therefore, the approach described in this paper provides rich potential for further development of new Cloud Computing based applications.

[1] http://www.gloin.es/en/
[2] http://www.nimbees.com/

REFERENCES

F. AlShahwan, K. Moessner, “Providing SOAP Web Services and REST Web Services from Mobile Hosts”, In: *Fifth International Conference on Internet and Web Applications and Services (ICIW 2010)*, pp. 174-179.

Dropbox. Dropbox FAQ, 2013. https://www.dropbox.com/help/7/en.

Facebook, Do you know What's Up?. https:// www.facebook.com/notes/up-creative-inc/do-you-know-whats-up-check-out-these-2013-social-media-statistics/470970089631080.

E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Pattern – Elements of Reusable Object-Oriented Software, Addison-Wesley. 1995.

Gartner. Market Trends: Application Development Software, Worldwide, 2012-2016. Technical report, Gartner, 2012. http://www.gartner.com/ resId=2098416.

J. Guillén, J. Miranda, J. Berrocal, J. García-Alonso, J. M. Murillo, C. Canal, *“People as a Service: a mobile-centric model for providing collective sociological profiles”*. IEEE Software SWSI: Next Generation Mobile Computing (2014) [ACCEPTED, PENDING PUBLICATION]

M. Jansen. Analysis and Improvement of Energy Consumption for Providing Mobile Web Services. In. *International Journal of Soft Computing and Software Engineering*, DOI: 10.7321/jscse. 2013a.

M. Jansen. About the Necessity to Change the Perspective for Mobile Web Services. In: Proceedings of the 15th *IEEE International Symposium on Web Systems Evolution*, 2013b.

D. Kegel. The c10k problem, http://www.kegel.com/ c10k.html, last visited: 18.12.2013.

L. Li, W. Chou, “COFOCUS – Compact and Expanded Restful Services for Mobile Environments”, In: *Proceedings of the 7th International Conference on Web Information Systems and Technologies*, pp. 51-60, Noordwijkerhout, The Netherlands. 2011.

G. Meijer. Instagram: A Case Study for the Clouds, 2013. http://www.cloudproviderusa.com/instagram-a-case-study-clouds/.

S. McFaddin, C. Narayanaswami, M. Raghunath, “Web Services on Mobile Devices – Implementation and Experience”, In: *Proceedings of the 5th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA’03)*, pp. 100-109, Monterey, CA.

Miranda, J., Guillén, J., Berrocal, J., Garcia-Alonso, J., Murillo, J., & Canal, C. (2013). Architecting Infrastructures for Cloud-Enabled Mobile Devices. In C. Canal & M. Villari (Eds.), Advances in Service-Oriented and Cloud Computing SE - 23 (Vol. 393, pp. 277–287). *Springer Berlin Heidelberg*. doi:10.1007/978-3-642-45364-9_23.

T. A. Powell. AJAX – The Complete Reference, *McGraw-Hill Osborne Media*, 2008.

M. Raatikainen, T. Mikkonen, V. Myllarniemi, N. Makitalo, T. Mannisto, and J. Savolainen. Mobile content as a service a blueprint for a vendor-neutral cloud of mobile devices. *IEEE Software*, 29(4):28{32, 2012.

S. Srirama, M. Jarke, W. Prinz, “Mobile Web Service Provisioning”, In: *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006)*, p. 120, Guadeloupe, French Caribbean.

S. Tilkov, S. Vinoski. Node.js: Using JavaScript to Build High-Performance Network Programs, *IEEE Internet Computing*, vol. 14, no. 6, pp. 80-83, November/December, 2010.