Web Services and Data Caching for Java Mobile Clients

Alexandru Caracas, Iulia Ion, Mihaela Ion
{*alexandru.caracas*|*iulia.ion*|*mihaela.ion*}*@i-u.de*
International University in Germany, 76646 Bruchsal, Germany

January 27, 2007

Abstract

Web services are becoming more and more pervasive and are used by a number of information system clients. However, mobile clients still have limited computing and network resources. Mobile information systems require client side caching of data for various reasons. Caching issues are discussed in many scientific publications that mostly lack an usable implementation. Nearly all mobile devices support Java's mobile information device profile (MIDP). This paper presents the implementation of real-time train schedule mobile application for MIDP-capable mobile phones using web services architecture and data caching.

1 Introduction and Related Work

Mobile information systems clients pose many challenges. One of these challenges is the wireless connectivity which could be slow, expensive, or not widely available. Another issue is that mobile devices have limited computational and memory resources. To alleviate these two problems we propose mobile applications to have: a web services architecture to lessen computation and memory requirements, and data caching to reduce network traffic.

Web services for mobile clients are scarcely discussed in literature and implementations for mobile phones are nascent. There are several advantages of using a service oriented architecture: loose coupling of services and adaptability for changes. There are many publications on caching, hoarding or replicating data in mobile information systems. However, almost all of them lack an usable implementation and evaluations are mostly of a purely scientific nature based on system simulations. In this paper, we present the implementation of a Java-based train schedule mobile application using web services as well as client side caching for storage and offline usage of data.

Our paper touches various research areas. The main two topics are web services for mobile devices and client side data storage in mobile information system. Based on the classification of replication, hoarding and caching in [4] one can identify major caching issues in our work. We implicitly store data on the mobile client that is explicitly queried and reuse it.

There are some projects that aim to implement a similar system to our application presented in Section 3. *Deutsche Bahn* offers a static solution for train schedules[1]. Mobile users have the possibility to download a personal schedule between two stations by specifying the start and destination. However, there is no guarantee that the data is up to date, moreover the user is presented only with the trains to one particular destination.

Another approach is by *Actuan Mobile* who offers the complete dynamic schedule for the New Jersey Transit rail system. The application[2] is offered both as a mobile browser solution and J2ME application. The mobile application available for download offers extensive features. However, the schedule information is obtained directly by querying on line servers for each request using proprietary protocols.

[1]`http://persoenlicherfahrplan.bahn.de`
[2]`http://www.actuanmobile.com/documentation/train\_schedule\_manual-online-version-v0\_1.pdf`

2 Caching and Web Services on Java Mobile Devices

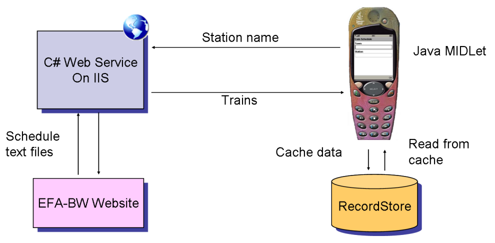
In the following we briefly present how to access an information system server via web-services from an MIDP-capable mobile device and introduce the Record Management Store (RMS) that was used for caching data on the client.Web services are accessed from mobile devices (J2ME clients) using the standard JSR 172: J2ME Web Services Specification [1]. The specification provides two new capabilities to the J2ME platform: (a) access to remote SOAP/XML based web services, and (b) parsing XML data.

Method invocation follows the syncronous request-response model of client-server interaction. A mobile client invokes a request to a remote web service. The web service processes the request and sends back the result. The client then receives the data. The JSR 172 stub and runtime handle the encoding of the method and arguments, serialization, sending the request and receiving the response, decoding, and de-serialization - transparent to the application.

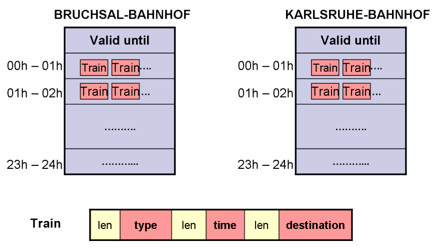
The RMS provided by MIDP is ideal for storing data on the client. A record store is identified by its name. A record store contains a number or records accessible by their index. A MIDlet application can persistently store data using the provided framework. A more detailed description of the MIDP record management store is given in [2, 3]. Our Java caching implementation address the following caching issues: coherency, replacement and look-up.

3 Mobile Train Application Architecture

The mobile application (see Figure 1(a)) connects to a web service to download the latest schedules for a given station. The web service sends the schedule which is serialized as an XML document which includes trains with their respective type, departure time, final destination, etc. The web service is responsible for data consistency, updates and proper format. This transfers the computation and storage intensive tasks to an 'always' on-line server machine.



(a) Train schedule application architecture



(b) Train schedule data structure

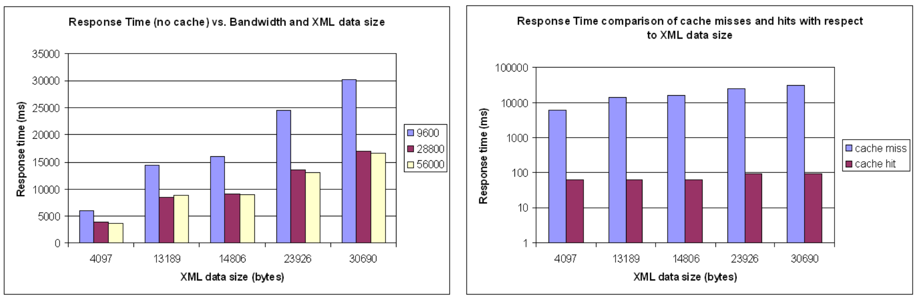
Figure 1: Architecture and implementation details

The MIDlet application caches the data into the persistent storage of the Record Store. If the user queries for the next trains for a given station, the application first checks whether the information is already present in cache. If the information is cached and the schedule still up-to-date the next departing trains are presented to the user. Otherwise, the MIDlet queries the Web Service for the latest schedule for the given station and caches the received response.

Web services involve marshaling of XML documents. However, saving XML data on a mobile phone is not feasible due to the limited amount of storage. Nevertheless, XML data is hierarchical and can be mapped to the RMS while keeping an efficient index structure. The pragmatic method is to map each binary serialized XML data record to a record and use the name of the document as the key for the record store. Figure 1(b) graphically depicts the data structure used for caching the train schedule information for two example train stations, Bruchsal and Karlsruhe. Each of the train station is saved in a separate record store.

4 Evaluation

For measuring the performance of our application we used the mobile emulator present with the J2ME Wireless Toolkit. The reason is that support for JSR 172 is nascent. Many devices, especially mobile phones and smart phones lack the required implementation. The tests represent the performance of the application for various parameters of the VM emulator speed and network throughput emulation.



(a) (b)

Figure 2: Cache performance

Figure 2(a) illustrates for various XML data sizes the response time for answering a query if no caching is used. In fact, this means that all data needs to be accessed from the server. Figure 2(b) shows the response time in case of a cache miss to the case of a cache hit. Answering a query locally is much faster than receiving the data over network. It is interesting to see, that the influence of the size of nearly requested data disappears if data can be provided locally.

5 Summary, Conclusions and Outlook

We presented a service oriented train schedule application using data caching for MIDP-capable mobile device. We discussed the architecture and implementation, illustrated the cache related issues and presented first evaluation results. This paper is only a first step in terms of caching for mobile clients. Our implementation substituted nearly all caching issues that are currently discussed in journals and on conferences by pragmatic approaches. Moreover, the web services technology is currently not widely deployed and supported by Java mobile clients. Nevertheless, the results show that web services and caching on mobile phones are indeed possible and have a huge potential to support the integration on mobile devices into distributed information systems.

The next steps of our work are to investigate further the availability of web service specifications for mobile devices, and implement more efficient caching strategies as well as more detailed evaluations.

References

[1] John Ellis and Mark Young. *J2ME Web Services 1.0.* Sun Microsystem, Inc., Santa Clara, CA, USA, final draft edition, October 2003. Available at `http://jcp.org/aboutJava/communityprocess/final/jsr172/index.html`.

[2] Soma Ghosh. J2ME record management store – Add data storage capacities to your MIDlet apps. Online article at IBM developerWorks, May 2002. Available at `http://www-128.ibm.com/developerworks/library/wi-rms/`.

[3] Eric Giguere. Databases and MIDP, Part 1: Understanding the Record Management System. Online article at Sun Developer Network, February 2004. Available at `http://developers.sun.com/techtopics/mobility/midp/articles/databaserms/`.

[4] Hagen Höpfner, Can Türker, and Birgitta König-Ries. *Mobile Datenbanken und Informationssysteme — Konzepte und Techniken.* dpunkt.verlag, Heidelberg, Germany, July 2005. in German.