Realizing Mobile Web Services for Dynamic Applications*

Sonja Zaplata, Viktor Dreiling and Winfried Lamersdorf

Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
[zaplata|5dreiling|lamersdorf]@informatik.uni-hamburg.de

Abstract. Use of web services also on mobile devices becomes increasingly relevant. However, realizing such mobile web services based on the standard protocol stack is often inappropriate for resource-restricted mobile devices in dynamic networks. On the other hand, using specialized alternative protocols restricts compatibility with traditional service applications. Thus, existing approaches often do not allow to integrate heterogeneous service instances dynamically, as it is, e.g., required for executing mobile service-based business processes.
In order to adequately support such more complex and dynamic applications, this paper presents a lean and flexible system architecture which supports both mobile web service consumers and providers by allowing to integrate multiple protocols depending on their capabilities and to dynamically access suitable service instances at runtime. As a proof of concept, the paper shows an exemplary combination of practically relevant protocols for resource-limited devices based on WSDL, ASN.1 and overlay transport and presents its integration in a prototype scenario for supporting decentralized mobile business processes.

1 Introduction

Mobile web services currently form one of the most promising approaches to apply well-established service-oriented concepts to mobile environments. Especially the emergence of respective mobile middleware systems leads to a rather ubiquitous availability of information and enables new personalized and context-based services for private consumers as well as for business applications. Considering the provision and consumption of such service functionality in stationary networks, web services have proved to be a successful integration technology. Based on the standardized *Web Service Description Language (WSDL)*, the message encoding format *SOAP* and the *Hypertext Transfer Protocol (HTTP)* as specified by the W3C [4], a web service typically defines an interface between two or more software applications. As web services are self-describing and enable the

* The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

development of loosely-coupled distributed applications, they are - in general - also very well suited to integrate mobile service providers and consumers. Nowadays, standard web service technologies can be applied to several mobile devices almost without any problems, e.g. considering notebooks or the newest generation of mobile phones using relatively reliable wireless networks such as WLAN or UMTS. However, the conventional web service communication framework is mostly inappropriate for small mobile devices in decentralized networks, e.g. for wireless sensors or active RFID tags, which still have very restricted resources with respect to computing power, memory capacity and communication bandwidth (cp. [1]). Several drawbacks of standard web service protocols have already been investigated in previous research works: As the most important point, the textual representation of XML-based descriptions as used in WSDL and SOAP leads to a low information density and thus to an inefficient use of communication bandwidth. As another example, the synchronicity of HTTP results in intolerance to network failures and excludes typical mobile network technologies such as Bluetooth or IrDa. Concerning the discovery of mobile services, centralized systems such as *UDDI (Universal Description, Discovery and Integration)* can hardly be applied in decentralized networks and prove to be inefficient in systems with changing network addresses (cp. [3, 14]).

The emergence of manifold and more decentralized applications have therefore triggered the development of alternative web service protocols dealing with some of the before mentioned problems. Being specific to a concrete network or addressing particular drawbacks such as messaging overhead, these protocols focus on the requirements of resource-limited mobile systems and respectively use less complex communication protocols and description languages (e.g. [2, 17]). Such alternative protocols enable mobile devices to consume specially adapted web services running on stationary servers, e.g. in order to outsource business logic or tasks which are computationally intensive. Since mobile devices are also able to provide web services themselves, also novel applications such as sharing resources and functionality in mobile ad-hoc networks can be realized. For example, a built-in car navigation system could be used to transfer the current position to a local mobile phone using Bluetooth. Nevertheless, it could also be accessed from remote (e.g. by a desktop PC) to find a stolen car by using a standard HTTP connection. Other application areas involve the provision of context information about the user or its device, or act as a replacement of physical things, e.g. by simulating a wallet by an automatic payment service [3].

Besides such specialized monolithic applications, (mobile) web services can also be part of more complex and dynamic applications, such as, e.g. business processes running on mobile devices (e.g. [8, 13]). Due to the prevailing diversity of protocols in the area of mobile web services, most of such distributed applications use rather abstract descriptions of services, avoiding to specify concrete protocols, network addresses and other specific technological details. In contrary to stand-alone applications, the execution of mobile business processes therefore requires a dynamic discovery, selection and binding of available services and thus requires to support more than one specific protocol. At the same time the

processes' functionality is provided as an aggregated service itself. This means that there is a need for a dynamic mobile web service architecture embracing functionality for service consumption as well as for service provision, considering heterogeneous devices, networks and protocols.

Addressing such challenges, the following section analyzes existing work in the area of mobile web services and identifies research gaps with respect to dynamism, flexibility and interoperability of mobile service providers and consumers. To overcome the identified restrictions, section 3 introduces a lightweight architecture to both use and provide web services based on arbitrary protocols, as well as to publish, find and bind such services dynamically. Section 4 presents an example combination of protocols suitable for smaller and medium mobile devices. The prototypical implementation is evaluated in section 5, integrating the proposed architecture and its reference configuration into an existing mobile process execution system. The paper concludes with a brief summary and an outlook on future work.

2 Existing and Related Work

Due to the large amount of work in the area of web services and mobile computing, this section abstracts from individual approaches, but classifies them with respect to the strategies used to provide and use web services in heterogeneous mobile environments. Thus in general, respective previous and ongoing research can be distinguished into three main areas: Application and adaptation of standard web service technology; integration of alternative protocols, description languages and registries; and use of additional mediator components.

Adaptation of Standard Web Service Technology As introduced above, in some cases standard web service technologies (i.e. WSDL, SOAP, HTTP and UDDI) can directly be applied to mobile systems (as e.g. shown by [15]) – assumed that these are relatively powerful, use reliable network connections and are able to provide adequate addressing mechanisms. Smaller and more restricted mobile systems however often omit dynamic components which need a large amount of resources or which cannot be realized due to decentralized infrastructures. Two examples are summarized in the following:

- Considering the consumer side, web services can be bound statically as a fixed part of the mobile application. This relieves mobile devices from service discovery and from generating and integrating web service proxies at runtime. However, this simple approach is very inflexible as services cannot be exchanged at runtime and thus it does not support applications which require to pick service instances dynamically.
- Mobile service providers can optionally abstain from publishing their services in a registry and assume that potential service consumers are aware of the services' existence and syntax. Obviously, this strategy is rather restrictive as service providers can hardly expand the number of users if the service cannot be discovered dynamically.

Alternative Protocols, Description Languages and Registries As standard web service protocols do not adequately meet the needs of resource-restricted mobile computing infrastructures, alternative technologies have evolved. These address – among others – the overhead of XML in service descriptions and messages, the synchronicity of communication and the centralization of registry information. Examples to exchange (in part or in total) the standard combination of HTTP, SOAP, WSDL and UDDI are sketched in the following:

- Universal (e.g. ZIP) or XML-specific (e.g. XMILL) compression mechanisms can efficiently be used to minimize the size of XML messages (e.g. [17]). Nevertheless such algorithms are quite resource-intensive as they require a relatively large amount of computing power to encode and decode the messages.
- To reduce complexity in another way, the use of XML can be avoided by alternative description languages, such as *JSON (JavaScript Object Notation)* or *ASN.1 (Abstract Syntax Notation Number One)* (cp. [12]).
- A more appropriate asynchronous communication can be realized by using alternative protocols such as SMTP and POP/IMAP, decoupling sender and receiver and thus allowing disconnected operation of web services (cp. [18]).
- The overhead of HTTP can alternatively be eliminated by performing message exchange over TCP or UDP directly (cp. [18]).
- Registries for decentralized infrastructures allow service providers to describe their services locally (e.g. WS-Inspection) or to save service information in a distributed way (e.g. *Konark* presented by [9]).
- The emergence of advanced addressing mechanisms such as *MobileIP* will probably facilitate the access of mobile (web service) resources.

Mobile Web Service Architectures and Use of Mediator Components While the use of traditional web service technologies does not consider specific characteristics of mobile computing systems, the restriction to specialized alternative approaches leads to an incompatibility with traditional web service applications. Therefore, current research considers the challenges arising from the diversification of above mentioned technologies and protocols. Primarily, approaches which are similar or related to this work focus on the *use of additional mediator components*. The most important examples are presented below:

- In order to address the exclusion of local services and personal area networks, *proxy components* can be applied both to service consumers and providers. As an example, the approach presented by [2], presents an architecture which allows web service invocation over Bluetooth by wrapping SOAP messages to bind them to the Bluetooth transport protocol. More general approaches establish an overlay network to completely abstract from technological details of the underlying transport layer (e.g. [6]).
- To consider limitations of mobile systems and allow proprietary protocols, a *mediation framework* can act as a broker between the mobile device and stationary web service providers or consumers (e.g. [5, 7]). In this case, the

mediator is responsible for the transformation and the routing of web service messages. Furthermore, peer-to-peer mediator approaches have also successfully been applied to mobile service providers and consumers [16]. However, if mediators are not accessible, this component represents a hazardous single point of failure in centralized as well as in decentralized infrastructures.

- To integrate alternative transmission protocols dynamically, the preferred message representation can be subject of negotiation. As an example, the *Handheld Flexible Representation (HHFR)* [14] optionally determines which part of a SOAP message is omitted when invoking a service. The approach is characterized by a very flexible architecture and is able to adapt to the requirements of mobile devices dynamically. Considering the repeated invocation of the same service, the following data exchange can be reduced considerably. In case of single service invocations, however, the negotiation itself causes a considerable overhead.

Requirements Summary As an interim result, it seems that there is no perfect combination of traditional and alternative technologies, but that the use of a specific approach is determined by the capabilities of the mobile system and its applications. Although web services have originally been intended to integrate heterogeneous resources, the diversification of protocols resulting from necessary adaptations leads to another integration problem. On the one hand, heterogeneous capabilities and characteristics of mobile devices with respect to network connection and protocol support have to be considered. On the other hand, interoperability with traditional applications and industry standards should be preserved. Finally, dynamic applications such as ad-hoc mobile business processes require the executing mobile device to adapt to available service instances and protocols at runtime – a system software characteristic which is hardly supported by current mobile web service architectures.

These observations lead to the need of a flexible web service architecture which is able to adapt to the prevailing technology at runtime – provided the respective (mobile) device is able to support one or more (to some extent) established protocols. The next section therefore presents the basic idea of a flexible web service architecture for such dynamic mobile applications.

3 A Flexible Mobile Service Architecture

As presented in the previous section, developers of mobile web service providers and mobile web service clients can select from a large range of protocols and technologies to adjust their application to the requirements and capabilities of the mobile device. To enable a customized design of mobile web service applications, to allow interoperability with more than one service consumer or provider and to access services dynamically, this section presents an adaptable web service architecture for mobile devices.

Figure 1 shows a coarse overview of the *decentralized* mobile service-oriented architecture. It consists of one or more (possibly mobile) service providers and

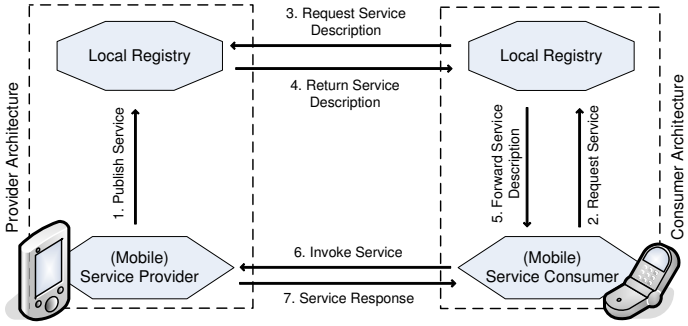

Figure 1. Overview of the mobile service architecture

consumers which both integrate an individual local registry. In case of the service provider, the registry holds and manages the service descriptions of the service instances provided by the mobile device itself. For the service consumer, the registry is responsible to search for required services by exchanging information with the registries of service providers in the local environment. Because the local registry only acts as a proxy to its environment, also centralized stationary registries (e.g. UDDI) or distributed decentralized registries (e.g. Konark by [9]) can participate if they are in communication range of the mobile service requester.

The detailed architecture for mobile web service consumers and providers is characterized by a modular design. The resulting basic architecture for both roles is depicted in figure 2. Due to potential resource restrictions, basic functionality such as communication, message handling and service registration is shared by consumer and provider components. Functionality exclusive to service providers involves a lightweight service runtime environment which manages respective service instances. Exclusive to the client side, a proxy generator is responsible for generating and assigning local proxies to invoke a mobile web service. The proxy represents a local interface of the remote service, handles the work of mapping parameters to the elements of the description language and prepares the respective message contents to be send over the network.

Depending on the capabilities of the mobile system and on the requirements of the application(s), this abstract architecture can be instantiated with one or more adapters realizing a concrete technology. Alternative technologies can be assigned to service description, to message encoding and to transport protocols. For example, to be compatible to industry standards, services can be described using a WSDL adapter for the local registry and for proxy generation, the message handling can use SOAP format and finally, the communication component can include an HTTP adapter to send the message. To be compatible to resource-restricted mobile devices, alternative configurations can be realized, e.g. as the combination of WSDL, ASN.1 and overlay network transport which is presented in section 4.

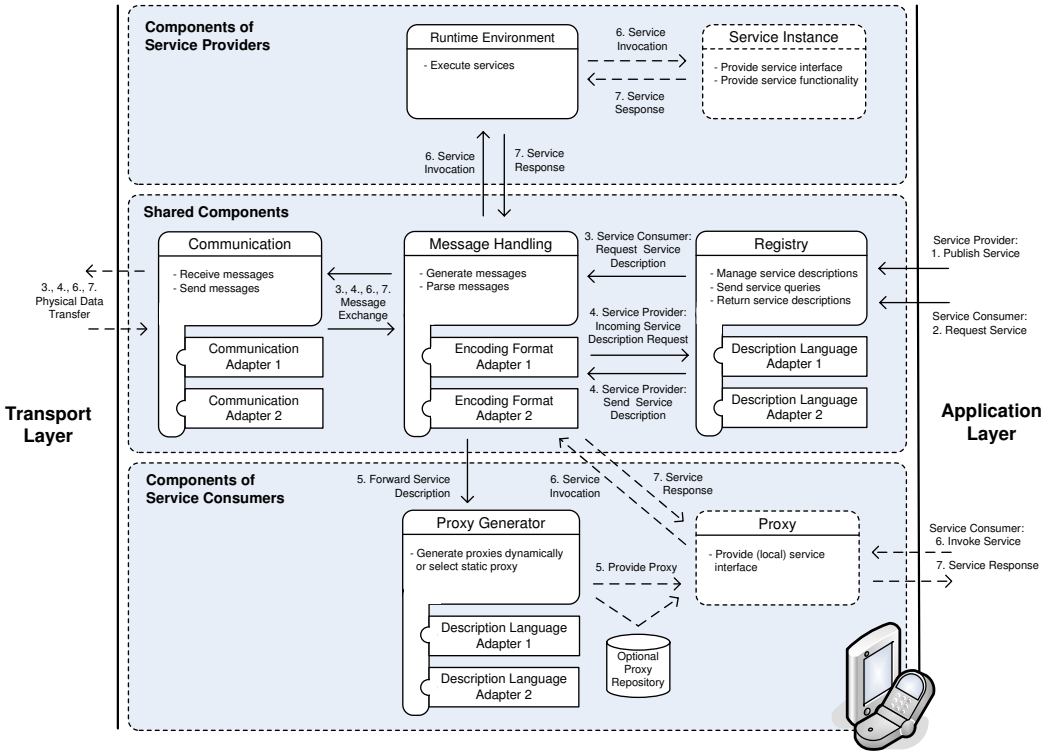

Figure 2. Mobile service architecture component model

The overall procedure of providing and consuming web services is realized as follows: First, the service provider publishes its services to the local registry (Step 1 in figure 1 and 2). As the deployment of adapters and services is performed at design-time, each published service can be associated with one or more descriptions determined by the configuration of supported protocols.

Potential service consumers are now able to find these services by performing an abstract service request to their local registry (step 2). The abstract service request contains the search parameters of the respective application, e.g. the required functionality of the service and optionally non-functional criteria. The consumer's registry first checks if the required service can be accessed locally, e.g. in case the service is provided by the device itself. Otherwise it forwards the request to other devices in its environment making use of the type(s) of encoding format and communication protocol it supports (step 3). The environment of the device is therefore determined by the capabilities of the communication adapter, e.g. resources on the Internet can be accessed via HTTP, whereas local networks can only be accessed via alternative communication protocols. The resulting request now involves at least the identifier of the service's functionality (e.g. a simple *Uniform Resource Identifier (URI)*, a *Universally Unique Identifier*

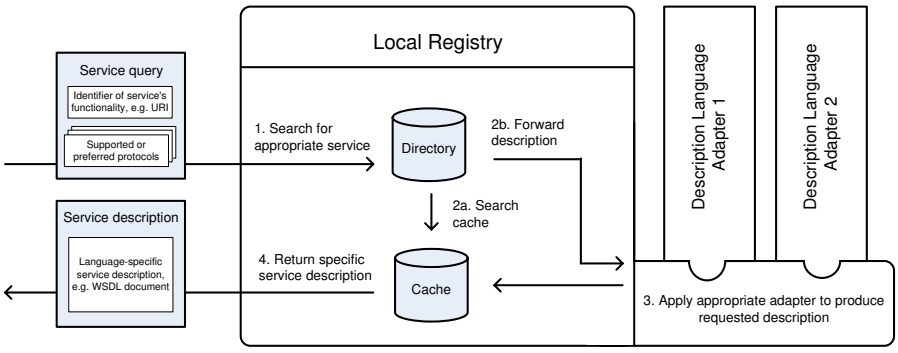

Figure 3. Service discovery: receiving a remote service request

(UUID) or a link to external semantic resources such as an *Resource Description Framework (RDF)* document) and optionally a list of supported or preferred protocols (cp. figure 3).

The potential service provider receives the incoming service request and – assumed it has at least one suitable adapter – forwards it to the registry which picks an adequate format to return the description of the requested service instance (step 4). As the description is received by the consumer, it is forwarded to the proxy generator (step 5). Depending on the implementation, the proxy can either be picked from a proxy repository holding a number of static proxies or can be generated automatically according to the received service description. The service consumer is now able to invoke the service by calling the provided proxy object (step 6). The proxy uses the message format and communication protocol as specified in the service description to send the required input parameters, and if any, receives the service's return values (step 7). If the service is going to be invoked again later, the proxy can optionally be added to the proxy repository.

To address scalability, the presented architecture supports complex applications acting as service providers and service consumers at the same time as well as both roles individually. As the role-specific components are completely optional, unneeded provider/consumer functionality can be omitted to save resources. Furthermore, the type and number of adapter components can be tailored to the capacity and performance of the mobile device. However, if the number of adapters is rather small or the applied protocols are too exotic, the compatibility will be restricted to special application areas and therefore influence the number of suitable service consumers or providers.

4 An Example Configuration for Mobile Web Services

Since actual web service standards WSDL, SOAP and HTTP do not meet the requirements of mobile systems particularly well, alternative technologies for the realization of mobile web services can be considered. This section presents a

proposal on technologies that can be integrated into the presented architecture to realize web services on more resource-restricted mobile devices. The configuration reduces the overhead of the message description by using a non-XML description language and provides mechanisms for compensation of connection resets by creating an overlay network between the mobile participants. However, this configuration only shows *one* example of several (arbitrary) combinations which can be composed depending on the mobile devices' actual capabilities. Other combinations and their interplay can be found in section 5.

Service Description The example configuration presented here uses WSDL 2.0 to be compatible to established web service based systems and only differs in the use of an alternative message description language and an alternative transport protocol. WSDL allows the integration of both alternative message description languages and transport protocols without violating the WSDL standards of W3C (cp. [4]). Listing 1 shows an example of a WSDL binding that contains the URIs associated with the alternative technologies used in this example configuration.

```
<wsdl:binding name="ExampleConfiguration"
     interface="TestServiceInterface"
     type="http://vsis-www.informatik.uni-hamburg.de/projects/demac/asn1der"
     protocol="http://vsis-www.informatik.uni-hamburg.de/projects/demac/overlay">
     <wsdl:operation ref="testOperation"/>
</wsdl:binding>
```

Listing 1. WSDL Binding

Encoding Format The example configuration uses ASN.1 and DER encoding to describe the communication messages containing the payload and the protocol data. The approach is based on the specifications X.694 [11], X.690 [10] and X.892 [12] of ITU-T and, in comparison to XML-SOAP, results in a reduced description overhead, which has also been shown in [12].

The basic idea of the message exchange is to use a predefined set of data types which are known to all participants (X.694 and X.892) followed by a binary encoding of the values according to their types (e.g. UTF8 encoding of strings) and a substitution of the data types by binary constants which are – due to the standardization – also known to other participants (X.690).

Listing 2 shows an example of an XML schema describing the structure of a message, whereas listing 3 shows the respective ASN.1 instance. Listing 4 shows the resulting DER encoding of this instance representing the actual payload of a communication message. As to see, the encoded value only contains information about the structure of the original complex value, the values of the elements it consists of and their types, but it does not contain additional identifiers.

The complete message is encoded similarly to the presented example. The X.892 specification of ITU-T describes the structure of an ASN.1 SOAP message and defines the obligatory fields. Among other attributes, each instance representing the payload of the message has an ID attribute to denote the schema of

the instance, particularly its URI (namespace) and its name. Since provider and client possess the WSDL document of the web service, both can understand the information that is encoded as an ID, assign the identifiers to the values and interpret the messages correctly.

```
<xsd:complexType name="integerSequence">
  <xsd:sequence>
    <xsd:element name="elem1" type="xsd:integer"/>
    <xsd:element name="elem2" type="xsd:integer"/>
  </xsd:sequence>
</xsd:complexType>
```

Listing 2. Message structure defined in XML Schema

```
integerSequence SEQUENCE ::=
{ elem1 2,
  elem2 3 }
```

Listing 3. Message structure defined in ASN.1

```
00110000 | binary constant associated with a sequence
00000110 | length of the binary representation of
           that sequence (number of octets, 6 in this example)
00000010 | binary constant associated with an integer (elem1)
00000001 | length of the binary representation of that integer
00000010 | value 2
00000010 | binary constant associated with an integer (elem2)
00000001 | length of the binary representation of that integer
00000010 | value 3
```

Listing 4. DER encoding of the example message

Communication Protocol The communication interface can be realized by one of the individual alternative protocols presented in section 2, e.g. TCP/IP, Bluetooth or IrDA. To also show the applicability for more complex solutions, the communication adapter used in the example configuration abstracts from specific transport protocols, but relies on a peer-to-peer overlay network with its own addressing scheme and an asynchronous message transport (as e.g. proposed in [6]). To detect other devices in the environment, participating devices use their communication adapter to send short broadcast messages in periodic intervals. Within these messages, they encode their UUID – a identifier that is universally unique for every device. When a device receives such a message, it saves the UUID and its source address. This information is updated or complemented in case the same UUID is received with a different source address. As a result, the participating devices have basic up-to-date information about other devices in the (local) environment and the current protocols and addresses that can be used to contact them.

In order to communicate with a particular device, the sender selects an address associated with the UUID of the receiver. This (virtual) address is then translated into a concrete protocol specific address and the message is sent using the respective protocols and endpoint information. If the device is reachable by different protocols, more than one address can be associated with a UUID. The

participants are therefore able to select the most appropriate protocol – or change the communication interface in case a connection is temporarily interrupted.

5 Prototype Implementation and Use Case Scenario

In order to demonstrate the feasibility of the approach, the flexible architecture and its example configuration have been prototypically implemented and integrated into the *DEMAC (Distributed Environment for Mobility Aware Computing)* project. DEMAC realizes the idea of mobile (business) processes migrating several stationary and mobile devices in order to share their resources and functionality (cp. [13]). A typical application scenario for such processes is e.g. the context-based collection and processing of information in mobile environments, involving data from wireless sensors, mobile users or traditional web service resources. Since devices which are able to execute mobile processes can be considered to be relatively powerful (e.g. notebooks or PDAs), the presented architecture can be used to aggregate a set of protocols in order to integrate web services from several heterogeneous devices and networks. As required service functionality is specified in a technology-independent way, the process execution engine can use the presented architecture to search for adequate service instances and integrate them at runtime.

The resulting use case scenario is depicted in figure 4. The described example configuration has been applied to a wireless sensor (device 1) which provides temperature data. The application of the example configuration using ASN.1 reduces the size of communication messages considerably (cp. also last row in figure 5) and achieves even better results if the number of long identifiers that occur in the message payload is getting larger. The ASN.1 type library is implemented as a small set of structures which can be combined to create a complete message. The instance of each structure calls the encoding procedure responsible for the associated ASN.1 type and saves its result into a collective output container. Thus the messages do not have to be parsed, but can be encoded directly by passing the respective values to the encoder. In consequence, the implementation is very fast and efficient and can be considered to be quite suitable even for latest technologies such as e.g. active RFID tags which have a very restricted communication bandwidth.

The standard web service configuration is provided by a stationary server (device 2) transforming the temperature data into another representation (e.g. Celsius to Fahrenheit). Device 3 is responsible to execute the mobile process integrating both of these functionalities as a simple sequential service composition. Using adapters for the presented reference configuration addressing small mobile devices (cp. section 4) and adapters for the standard set of web service technologies (i.e. WSDL, SOAP and HTTP), the executing mobile device is able to access the wireless sensor as well as the traditional stationary web service. It is further able to dynamically generate the respective proxies and thus involve the required functionality to fulfill the mobile processes' activities at runtime. The integrated services are re-offered as a composed service functionality using

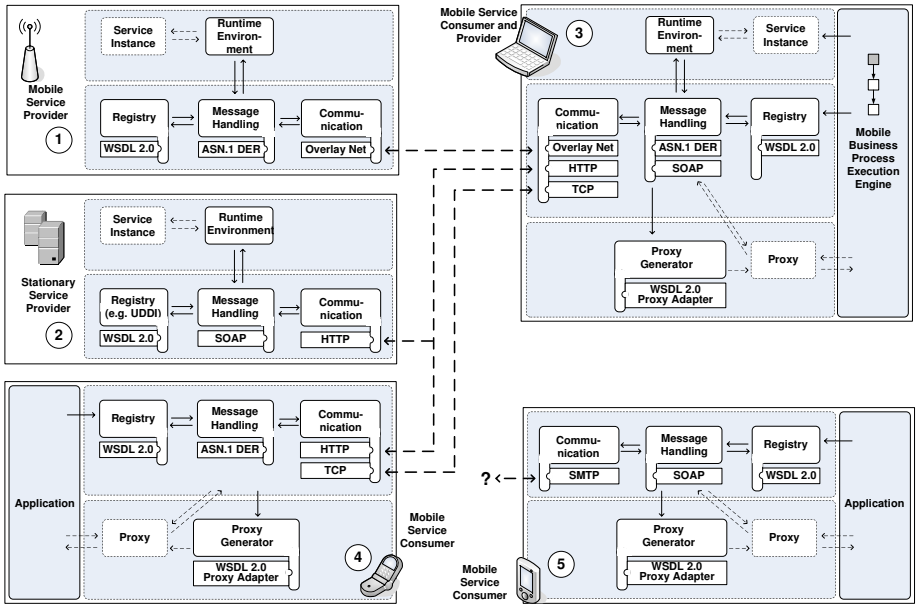

Figure 4. Mobile web services test configuration

either the example configuration, the standard web service technologies or even another mix of protocols, as exemplary represented by another web service consumer (device 4). However, if the set of supported protocols does not match any other configuration (as indicated by device 5) the required services cannot be accessed. Due to its mobility, the incompatible device is however still able to potentially find adequate services in another environment.

The number and size of the messages exchanged to execute the presented scenario are depicted in figure 5. To allow a proper comparison of message sizes, all services used in the test share a similar message structure (i.e. a request-response message exchange pattern with one input and one output parameter) as well as a similar service description (in WSDL). The italic font indicates that the respective value is variable and results from the parameters used in the test scenario.

The experimental evaluation of the prototype shows that the load of finding the proper configuration only affects devices which are able to cope with different protocols and adapters - and thus can be regarded to be more powerful. If more than one adapter for communication is available, the device can start service discovery with its preferred protocol and fall back to other protocols in case there is no positive response. For instance, in the worst case, device 3 would have to send the service discovery message over all of its three communication protocols. It is obvious that the more adapters are available on a mobile device, the higher is the probability of finding an adequate service. Less powerful devices will simply ignore the messages which cannot be interpreted and only respond

Device Number in scenario	1		2		3				4	
Device	Sun SPOT Wireless Sensor		Intel Pentium 4 Desktop PC		ASUS Eee PC 1000H Netbook				Nokia 6131 NFC Cell Phone	
Properties (Processor, RAM)	180 MHz		3,2 GHz		1,6 GHz				229 MHz	
	512 KB RAM		1 GB RAM		1 GB RAM				26 MB RAM	
Role Type	Mobile service provider		Stationary service provider		Mobile service consumer and provider				Mobile service consumer	
Communication Protocol	Overlay Network		HTTP		HTTP	TCP	Overlay Network		HTTP	TCP
Header Size (Bytes)	*86* (+20)		*123* (+20)		*123* (+20)	20	*86* (+20)		*123* (+20)	20
Messages for Service Discovery	Service Queries received	Descriptions sent (WSDL)	Service Queries received	Descriptions sent (WSDL)	Service Queries received	Descriptions sent (WSDL)	Service Queries performed	Descriptions received (WSDL)	Service Queries performed	Descriptions received (WSDL)
Message Exchange for Service Discovery	1	1	1	1	1	1	max. 3	1	1	1
Message Size (Bytes)	*86*	*1547*	*86*	*1547*	*86*	*1547*	max. *258*	*1547*	*86*	*1547*
Service Message Description Language	ASN.1		SOAP		ASN.1		SOAP		ASN.1	
Service Message Type	Request	Response	Request	Response	Request	Response	Request	Response	Request	Response
Service Message Size (Bytes)	*114*	*24*	*914*	*758*	*114*	*24*	*914*	*758*	*114*	*24*
Message Exchange for Service Execution	Received: 1	Sent: 1	Received: 1	Sent: 1	Sent: 1 Received: 1	Sent: 1 Received: 1	Sent: 1	Received: 1	Sent: 1	Received: 1
Total Message Size for Service Execution (Bytes)	*138*		*1672*		*1948*				*138*	

Figure 5. Overview of message exchange and size within the scenario

to those which will lead to a correct service invocation. The configuration of adapters and thus protocols can be installed in a way which fits the device's capabilities and performance best, leading to an reduced message description overhead as exemplary shown by the total message size of device 1 in the last row of figure 5: The message overhead is only 138 Bytes, which constitutes only 8.25 percent of the respective traditional technology (e.g. the message size of device 2: 1672 Bytes).

6 Conclusion and Future Work

Due to the heterogeneity of current mobile systems, it seems that there is no generally applicable combination of web service technologies, but that the use of a specific approach is determined by the capabilities of the specific mobile device. For enabling also more complex and dynamic applications such as ad-hoc mobile business processes, this paper proposes a flexible mobile web service architecture which supports accessing the functionality of multiple heterogeneous devices. By use of a customized configuration of protocols and technologies, this architecture can be tailored according to the requirements of the respective (mobile) application and its users, allowing to preserve interoperability with industry standards while also respecting the restrictions of resource-limited devices.

However, as also to see in figure 5, the exchange of WSDL descriptions takes a significant amount of the overall data transfer. As recommended, a possible so-

lution is to integrate alternative description languages, such as e.g. JSON which reduces the overhead of XML of about 20 percent. If this is still unsatisfying, the presented architecture could be enhanced to optionally provide compression mechanisms for service descriptions and service invocation messages. Furthermore, mobile service requesters capable of carrying multiple adapters may (in the worst case) produce unnecessary messages which could be inadequate for networks with a small bandwidth. This problem can be addressed by an increased network-awareness, enabling the mobile service requester to prioritize more lightweight protocols. Future work therefore involves the integration of context information to adapt not only to the capabilities of mobile devices but also to specific network characteristics.

References

1. F. Adelstein, S. K. Gupta, G. Richard III, and L. Schwiebert. *Fundamentals of Mobile and Pervasive Computing*. McGraw-Hil, 2005.
2. V. Auletta, C. Blundo, E. D. Cristofaro, and G. Raimato. A Lightweight Framework for Web Services Invocation over Bluetooth. In *Proceedings of the IEEE Int. Conf. on Web Services (ICWS06)*, pages 331–338. IEEE Computer Society, 2006.
3. S. Berger, S. McFaddin, C. Narayanaswami, and M. Raghunath. Web Services on Mobile Devices – Implementation and Experience. *IEEE Workshop on Mobile Computing Systems and Applications*, 0:100, 2003.
4. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web Services Architecture. Technical report, W3C, 2004.
5. C. Chong, H.-N. Chua, and C.-S. Lee. Towards flexible mobile payment via mediator-based service model. In *Proceedings of the 8th Int. Conf. on Electronic Commerce (ICEC06)*, pages 295–301. ACM, 2006.
6. D. Doval and D. O'Mahony. Overlay Networks: A Scalable Alternative for P2P. *IEEE Internet Computing*, 7(4):79–82, 2003.
7. P. Farley and M. Capp. Mobile Web Services. *BT Technology Journal*, 23(3):202–213, 2005.
8. G. Hackmann, M. Haitjema, C. D. Gill, and G.-C. Roman. Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices. In *Int. Conf. on Service-Oriented Computing (ICSOC 2006)*, volume 4294, pages 503–508. Springer, 2006.
9. S. Helal, N. Desai, V. Verma, and C. Lee. Konark – A Service Discovery and Delivery Protocol for Ad-hoc Networks. volume 3, pages 2107–2113. IEEE Computer Society, 2003.
10. ITU-T. ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). Technical report, International Telecommunication Union, 2002.
11. ITU-T. ASN.1 Encoding Rules: Mapping W3C XML Schema Definitions into ASN.1. Technical report, International Telecommunication Union, 2004.
12. ITU-T. Generic Applications of ASN.1: Fast Web Services. Technical report, International Telecommunication Union, 2004.
13. C. P. Kunze, S. Zaplata, M. Turjalei, and W. Lamersdorf. Enabling Context-based Cooperation: A Generic Context Model and Management System. In *Business Information Systems (BIS 2008)*. Springer, 2008.
14. S. Oh. *Web Service Architecture for Mobile Computing*. PhD thesis, Indiana University, Indianapolis, USA, 2006.

15. S. N. Srirama, M. Jarke, and W. Prinz. Mobile Web Service Provisioning. In *Proceedings of the AICT and ICIW 2006*, page 120. IEEE Computer Society, 2006.
16. S. N. Srirama, M. Jarke, and W. Prinz. Mobile Web Services Mediation Framework. In *Proceedings of the 2nd Workshop on Middleware for Service Oriented Computing (MW4SOC07)*, pages 6–11. ACM, 2007.
17. M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller. Performance Considerations for Mobile Web Services. *Elsevier Computer Communications Journal*, 27:1097–1105, 2004.
18. C. Werner, C. Buschmann, and T. Jacker. Enhanced Transport Bindings for Efficient SOAP Messaging. In *Proceedings of the IEEE Int. Conf. on Web Services (ICWS05)*, pages 193–200. IEEE Computer Society, 2005.