Mobile Information Systems 3 (2007) 165–185
IOS Press

Mobile P2P Web Services using SIP

Guido Gehlen, Fahad Aijaz, Yi Zhu and Bernhard Walke
RWTH Aachen University, Faculty 6, Communication Networks, Kopernikusstr. 16, 52074 Aachen, Germany
E-mail: {guge,fah}@comnets.rwth-aachen.de

Abstract. Telecommunication networks and the Internet are growing together. Peer-to-Peer (P2P) services which are originally offered by network providers, like telephony and messaging, are provided through VoIP and Instant Messaging (IM) by Internet service providers, too. The IP Multimedia Subsystem (IMS) is the answer of the telecommunication industry to this trend and aims at providing Internet P2P and multimedia services controlled by the network operators. The IMS provides mobility and session management as well as message routing, security, and billing.

This article is a comprehensive version of a conference publication [1] and introduces a Mobile P2P Web Services framework which enables the creation of arbitrary P2P application on mobile devices within the IMS. The terminals are sharing and publishing a Web Service interface, which provides the capability of coupling arbitrary applications using Web Service technologies and the support of the IMS infrastructure. The concept, the necessary Web Service adaptations, and extension as well as the interworking of Web Services, IMS nodes, and protocols are described and proofed by an exemplary mobile chess game application. In addition, the performance of mobile P2P Web Services is evaluated by means of an analytical analysis.

Keywords: Mobile Web Services, IMS, SIP, SOAP, mobile P2P

1. Introduction

The success of the Internet and Web Services along with the success of mobile networks in the last few years let forecast a high economical potential of the combination of both, called Mobile Web Services. From technological point of view many challenges have to be coped to combine technologies from the Internet and Web domain with technologies from the telecommunication domain.

The IMS is from the network infrastructure point of view a promising system to enable the interworking of the telecommunication and Information Technology (IT) domain. On the network layer the convergence has already been realized by the introduction of Internet Protocol (IP) service through the General Packet Radio Service (GPRS). This line is further continued by introducing IPv6.

Telecommunication services are naturally P2P services, e.g. a telephony/video call session, or the exchange of messages. But these are only a narrow subset of P2P services. In future, also “sharing” applications, like sharing the personal photo album, a whiteboard, or a multiplayer games will be normal for users of mobile terminals. In general, arbitrary services could be shared with the contacts of the buddy list. To enable such applications, the application developers need strong support in order to cope with

- the mobility of the users
- the changing networks and network conditions
- management of fluent application sessions
- changing environmental and user contexts

– variety of device types and operating systems
– variety of network types and protocols

The proposed Mobile Web Services framework is hiding application developers the complexity of the overall system and providing services, which transparently take the above listed challenges into account. The IMS is used for session/mobility management in order to enable a seamless transparent sessions even if the Radio Access Network (RAN) or the currently used terminal changes. For example, a user playing a chess game with a friend on a PC can switch without restarting the game to his mobile terminal keeping the chess session running while the user changes his location. In addition, new network types and conditions do not cause a session termination (mobility support, roaming and message routing are applied). Furthermore, the IMS enables the billing of arbitrary services, either online or offline, dependent on the business model.

Application developers are supported by the concept of the Service Oriented Architecture (SOA) realized by Web Services technologies to focus more on the actual application than on dealing with device or communication specific features. Using these technologies, a distributed application is as simple to handle as a local software component. The Web Services are provided and published like conventional services, e.g. a laundromat, a post office, or a translator. The application developer has the ability to search and investigate services of his interest by accessing a registry, like somebody is looking up the yellow pages.

The Mobile Web Service framework described in the following focuses on the contact point of the IMS and Web Service technologies, their interworking, and enhancements. These adaptations enable a seamless integration of Mobile Web Service in the IMS (combination of Simple Object Access Protocol (SOAP) and Session Initiation Protocol (SIP)) and an enhanced Web Service communication over mobile networks using alternative protocol bindings.

An important aspect of Mobile Web Services is their performance, since mobile networks are offering frequently changing and in general bad Quality of Service (QoS). For Mobile Web Service applications the latency of Remote Procedure call (RPC) calls is the most important QoS parameter, which has to be reduced to a minimum in order to guarantee fluent interactive applications. The results of an analytical Mobile P2P Web Service performance analysis is given in Section 6.

2. Mobile Web Services

In this work the term “Mobile Web Services” is used in any case where mobile networks and devices are involved in Web Service interactions. In the following, the Mobile Web Service domain is further divided into 3 sub-domains.

A “Mobile Web Service” (Mob-WS) is according to the definition in [2] a self-contained software component identified by an Uniform Resoure Identifier (URI), i.e. an Internet address or a SIP URI, which is deployed on a mobile device in a wireless/mobile network. In short, a Mobile Web Service is a Web Service, which is deployed on a mobile and published within a wireless/mobile network.

Basically, Mobile Web Services are classified in this work into three classes, classified by means of the Web Service requestor (WS-R) and provider (WS-P) deployment (see Fig. 1). The deployment of the Web Service broker (WS-B) can be ignored.

The Mobile Web Service classes are

1. Mobile Web Services Request (mobile WS-R/fixed WS-P)
2. Mobile Web Services Provisioning (mobile WS-R/fixed WS-P)

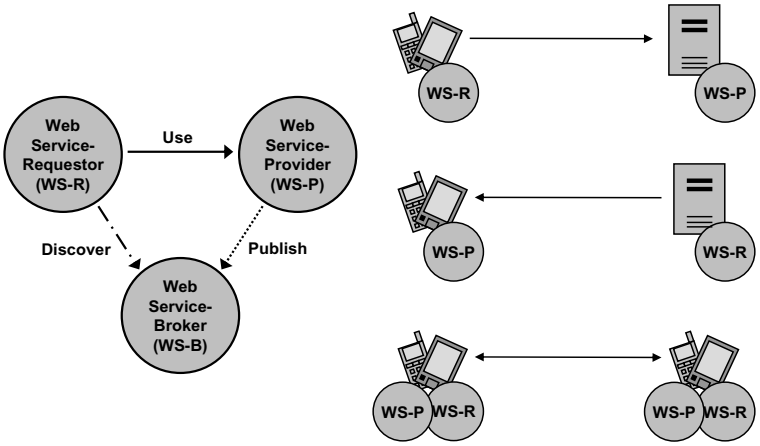

Fig. 1. Classification of Mobile Web Services.

3. Mobile P2P Web Services (both peers are WS-R and WS-P)

Web Services access from mobile devices works with no or with slightly changes of Web Service technologies, e.g. the improvement of Web Service communication over mobile networks using alternative protocol bindings. The provisioning of Web Services and Mobile P2P Web Services demand more efforts and changes on the default Web Service technologies, i.e. mobility support to enable Mobile Web Service mobility and session management by using Mobile Web Services in a P2P manner. There, the cooperation with the IMS is vital.

According to the Mobile Web Service definition and the classification of the last sections, a Mobile P2P-Web Service contains the following components form the software architecture point of view:

- SOAP node and SIP UA
- Mobile Web Service Proxies (SOAP client)
- Mobile Web Services (SOAP server)

3. Mobile P2P Web Services

Conceptually, P2P computing is an alternative to a centralized client-server model, but there is no clear border between them. A P2P environment has no fixed assignment of client and server nodes. All peer nodes are equal concerning their role to act as both, server and client. Popular P2P applications are sharing applications, like exchange of files (e.g. music or video files).

Generalizing this particular P2P file sharing application to Mobile P2P Web Services, each terminal publishes its Web Services and enables other terminals to access and use them. For large-scale networks, like cellular communication systems with Internet access, no technology specification is present. In the home network domain, Universal Plug and Play (UPnP) is one possible technology to enable P2P services in an ad-hoc manner. The next evolution step from UPnP to a Device Profile for Web Services [3] provides an aligned UPnP-like technology, which is fully aligned with the Web Service domain.

This article presents an extension for the basic Mobile Web Service (MobWS) framework that enables Mobile P2P Web Services for cellular mobile communication systems. Since the IP Multimedia Subsystem (IMS) is going to be the next generation service management system for cellular systems, the Mobile P2P Web Service framework couples to the IMS.

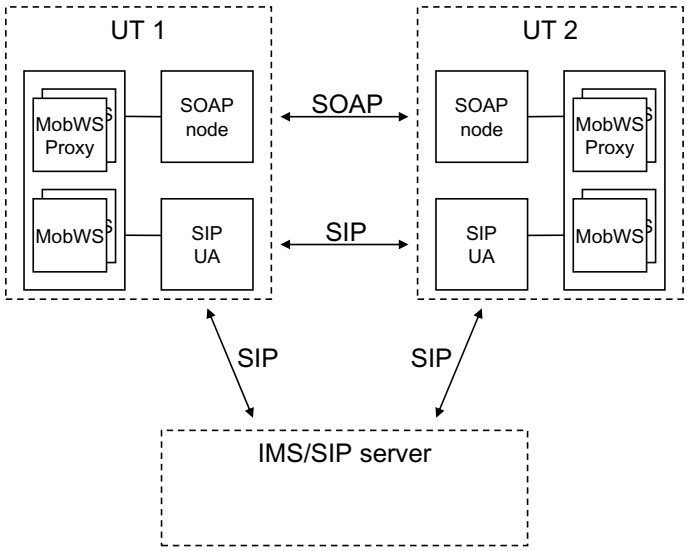

Fig. 2. Mobile Web Service and SIP UA integration.

The integration of Mobile Web Services into the IMS entails the design of interfaces between Web Service and IMS/SIP components. Figure 2 shows the components of two user terminals, UT1 and UT2, which are relevant for Mobile P2P Web Services. Generally, each terminal is able to provide and use Mobile Web Services at the same time and within the same SIP session. Thus, each device covers MobWS and MobWS Proxies.

In addition, the framework is enriched with a SIP User Agent (UA). The Mobile Web Service endpoint is now a SIP URI, which is resolved by the IMS infrastructure to the actual URI based on the terminal's actual IP address. A unique Web Service ID is used for each Mobile Web Service in order to easily identify and register Mobile Web Services. This ID is used in the session establishment of a Mobile P2P Web Service session (inside the Session Description Protocol (SDP) session description) and by registering the Mobile Web Service to the network infrastructure (SIP/IMS) (Fig. 2).

The used protocols to enable Mobile P2P Web Services are presented the Mobile Web Service Protocol Architecture description in Section 4.2. There, the registration to the IMS/SIP server, the mobility management, and the session management are explained. A case study is presented in the following that illustrates the functionality and the potentials of the technology.

3.1. Mobile P2P Chess game example

In order to proof the concept of Mobile P2P Web Services, a chess game for Kava2 Micro Edition (J2ME) has been extended with a MobWS interface in order to enable a game session between two users over the network. MobWS methods are defined, which move the chessmen. The Mobile Web Services framework is providing functionalities to easily expose these methods as a Web Service using either the default Hypertext transfer Protocol (HTTP) binding or a reliable or unreliable User Datagram Protocol (UDP) binding. The exposed methods are included in an Web Service Description Language (WSDL) file that covers the methods, container object definitions, the SIP endpoint, and information related to the HTTP and UDP bindings.

The MobWS game session establishment is illustrated by screenshots of the terminal's user interface shown in Fig. 3. The application runs on J2ME terminals or terminal emulators that support the J2ME

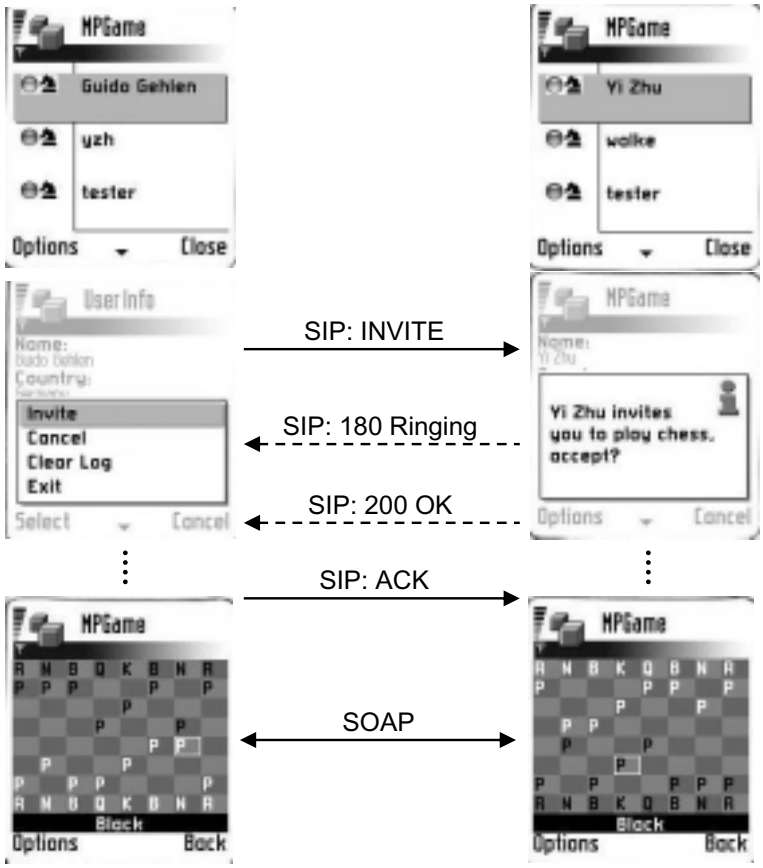

Fig. 3. Mobile Chess session establishment sequence.

SIP library JSR180 [5]. A SIP server has been modified in order to enable advanced presence and MobWS management.

The application is started from a conventional buddy list where the user's contacts and related present states are listed, as depicted in Fig. 3. The contact list and the presence state management is defined in the IMS specification resp. the SIP presence event package [6]. In addition to the presence state, the intersection of own and the contact persons MobWS are indicated by a symbol. In this example, the user's terminal and all of his contacts are providing the same chess game MobWS interface that is indicated by a "knight" chess symbol in the screenshots of Fig. 3.

On the left hand side the terminal (UT1) of the user "Yi Zhu" and on the right hand side the terminal (UT2) of the user "Guido Gehlen" are depicted. UT1 invites UT2 to a chess game by sending an SIP INVITE message with a MobWS session description using SDP. Before UT2 accepts the request, the SIP agent sends a provisional response (180 Ringing) back. After accepting the invitation, a final response (200 OK) is sent to UT1. The acknowledgement ACK of UT1 to UT2 finalizes the session establishment and the actual Web Service communication (SOAP) starts. The SOAP messages are containing the user data (chess maneuvers). If the terminal or the access network changes during the session, SIP mobility management signalling takes place and the SOAP nodes will be reconfigured.

For sure, these applications are realizable also without Web Service and IMS technologies, but the programming efforts are significantly reduced. The reuse of common network functionalities, existing

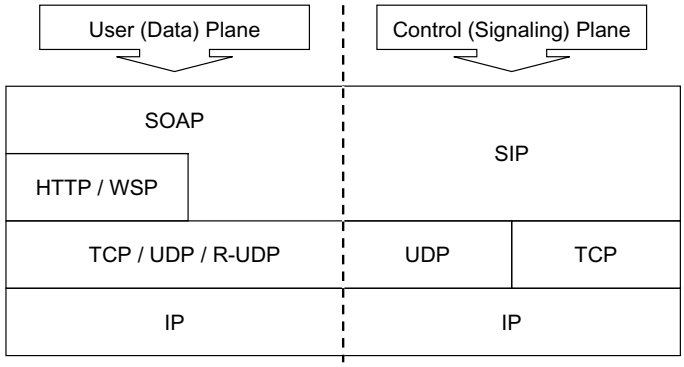

Fig. 4. Mobile Web Service user and control plane.

protocol bindings, and code generation tools ease the application development process, save time, cut down knowledge acquisition for software developers, and consequently save expenses.

4. Protocol architecture for Mobile P2P Web Services

The Web Service specifications define Web Service ontologies and service creation processes, the protocols are not in scope. The framework aims at being flexible in order to bind to arbitrary protocols. For simplicity and compatibility, the specification proposes at least the protocol stack SOAP on top of HTTP and TCP. Using Web Services in a mobile environment, the default protocol stack is not adequate.

A Mobile P2P Web Service protocol framework has to cope with following challenges:

- Mobility of users and their terminals
- Low bandwidth and high-delay connectivity
- Changing network conditions and infrastructure

The protocol stack depicted in Fig. 4 is proposed in order to counter the mentioned challenges and to keep compatibility with the default Web Service standards. The stack is divided into a user and control plane. The user plane is carrying the user data and the control plane the signalling messages.

On the user plane SOAP has been selected with various underlying protocol bindings. SOAP acts as a common RPC interface independent of the underlying communication protocols mainly on ISO/OSI layer 6 (the presentation layer). In case that no session management functions (layer 5) are provided by underlying protocols, SOAP can utilize WS-Addressing properties for that purpose. On transport layer, different protocols are possible, like TCP, UDP, Wireless Transaction Protocol (WTP), or the Reliable UDP (R-UDP). R-UDP has been used in the context of the article avoid HTTP/TCP communication, especially in the the Mobile P2P Web Service scenario.

For signalling, SIP is used due to reasons described in Section 4.2. SIP can generally be operated on top of UDP or TCP, but the Mobile Web Service (WS) prototype concentrates on SIP over UDP only.

The decision whether the information is related to user data or signalling cannot be generalized, since this dependeds on the application perspective. For instance, an Instant Messaging (IM) application can be realized by the use of the Extensible Messaging and Presence Protocol (XMPP) [7] only, or by the use of SIP and SIP extensions [8]. However, from the author's perspective it is more reasonable to use SIP for session and presence management and e.g. XMPP or SOAP for transmitting the actual user data. Thereby, user data and signalling is clearly separated.

Table 1
SOAP binding alternatives and their application within Mobile WS classes

	SOAP binding			
	HTTP	WAP	UDP	R-UDP
Mobile WS Access	x	x	–	–
Mobile WS Provisioning	x	–	x	x
Mobile P2P WS	x	–	x	x

The SIP specification [9] declares SIP as an application layer signalling protocol. This work follows the recommendation and focuses on the use of SIP on the control (signalling) plane in parallel to SOAP in the user plane, as depicted in Fig. 4. The strict separation of user and signalling plane has advantages in respect to protocol design, software design, and performance (network load on SIP servers/IMS elements).

Using SOAP on top of SIP within the user plane is also a possibility of transmitting SOAP by embedding the SOAP envelope inside of the SIP payload. This can be used in order to enrich SIP server with Web Service capabilities, but there are no advantages for mobile applications. Due to the violation of separation of user and control plane, this arrangement is not in focus. Related work is presented in [10] where SIP functions are provided by Web Services.

4.1. Mobile Web Services User Plane (*Data*)

The Web Service specifications dictate no particular protocol binding. However, the protocol configurations SOAP over HTTP or solely HTTP are used by default. The use of SOAP has some advantages with regards to service creation and integration. SOAP in combination with appropriate Web Service tools support application developers to easily create and integrate Web Services. They can handle remote Web Service calls as easy as local calls in their favorite programming language. The SOAP engine is responsible for the translation of objects and method calls from the programming language syntax into Extensible Markup Language (XML) syntax. However, this advantage comes along with a less efficient encoding of the transmitted messages in terms of their size.

In order to reduce the overhead one can compress SOAP messages. In [11] the message size of Web Service calls using SOAP are compared to RMI-IIOP, Common Object Request Broker Architecture (CORBA), and Remote Method Invocation (RMI). These alternatives require up to 70% less bandwidth than SOAP calls. Different SOAP compressions are analyzed with the result that Wireless Binary XML (WBXML) produces the best compression rate.

In addition to the mentioned advantages for application developers, SOAP provides the flexibility to bind to any arbitrary transmission protocol. This facilitates enhancements on transport layer. The characteristic of Web Service traffic in combination with the transmission properties of mobile communication systems result in performance degradations if the default HTTP SOAP binding is used. This article proposes alternative protocol bindings, which provide performance advancements in terms of reducing the RPC latency.

Three alternative SOAP bindings to HTTP, listed in Table 1, have been designed, implemented, and analyzed. A Wireless Application Protocol (WAP) binding, especially suitable for the Mobile Web Service Access class [12] and two UDP bindings qualified for Mobile Web Service Provisioning and Mobile P2P Web Services (Sections 4.1.2). The basic UDP binding realizes an unreliable transmission of SOAP messages, the other one (R-UDP) introduces in addition an error control mechanism (Selective-Repeat explicit request Automatic Repeat Request (ARQ) with cumulative ACKs) in order to protect against transmission errors. Details of the UDP SOAP binding are presented in Section 4.1.2.

```
POST /soaprpc HTTP/1.1
Content-Type: text/xml
Content-Length: 699
Connection: close
SOAPAction: urn:ChessService#Chess
User-Agent: kSOAP/1.0
Host: p900.comnets.rwth-aachen.de:9092

<SOAP-ENV:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

 ...
</SOAP-ENV:Envelope>
```

Fig. 5. SOAP RPC carried in an HTTP POST Request.

4.1.1. SOAP over HTTP

The default protocol binding to SOAP is HTTP. The Mobile Web Services framework uses this binding unchanged to the original definition. In addition to a client HTTP binding for mobile terminals a server binding has been developed, which enables the provisioning of Mobile Web Services through HTTP [13].

In this protocol configuration, SOAP covers solely presentation layer functionality. HTTP correlates request and response messages synchronously by sending both messages within one TCP connection. Thus, HTTP is taking the responsibility of the session layer (Fig. 4) and SOAP does not mandatorily require to include WS-Addressing [14] properties.

In the Listing 5 a Mobile Web Service request using the HTTP SOAP binding is displayed. The request is part of a Mobile P2P WS session and directed to a mobile terminal (`Host: p900.comnets.rwth-aachen.de:9092`). The requesting mobile client application uses the *kSOAP*[1] engine (`User-Agent: kSOAP/1.0`).

The corresponding SOAP response, depicted in Listing 6, is embedded in an HTTP response message. The mobile SOAP server answers with the fingerprint (`Server: CNSOAP/0.8` (`ComNets Mobile WS framework`) `kSOAP/1.0`) specifying the server version and used Web Service and SOAP engine.

SOAP over HTTP is almost the only used Web Service transport mechanism, since existing Web Server technologies are using solely HTTP. The introduction of new protocols in an existing infrastructure is very difficult to realize or afflicted with high costs. The MobWS framework includes therefore the HTTP binding even if one have to assume bad performance of applications in low-bandwidth and high-delay networks. However, enhancements are possible without changing the server infrastructure. This is presented in the following section by introducing a SOAP binding to the User Datagram Protocol (UDP).

4.1.2. SOAP over UDP

The User Datagram Protocol (UDP) provides upper layers a connectionless and unreliable datagram transmission service. There are mainly to different reasons to bind SOAP to UDP. First, SOAP is used as a presentation layer for originally UDP based services, like broadcast or multicast announcements or event messages. Secondly, UDP can be used as supplement for HTTP if no reliability is needed. If reliability is needed, a reliability mechanism has to be implemented on top of UDP.

[1] http://ksoap.objectweb.org/.

```
HTTP/1.1. 200 OK
Server: CNSOAP/0.8 (ComNets Mobile-WS framework) kSOAP/1.0
Content-Type: text/xml
Content-Length: 579
Connection: close

<SOAP-ENV:Envelope xmlns:n1="urn:UnsubscribeNameSpace"
  xmlns:n0="http://soapinterop.org/xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

 ...
</SOAP-ENV:Envelope>
```

Fig. 6. SOAP RPC return embedded in HTTP Response indicating successful completion.

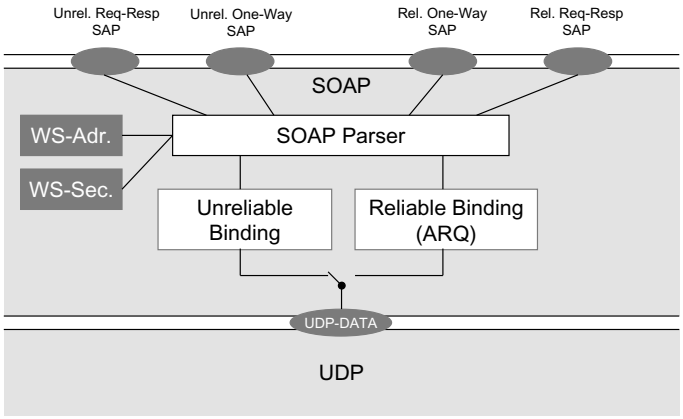

Fig. 7. SOAP layer concerning the binding to UDP.

These mentioned reasons provide a rationale the design of a SOAP binding to UDP, which enables application developers to have the option of using a UDP binding with optional reliability. The decision has been made to realize the reliability protocol on application layer, within the SOAP binding. In Fig. 4 the protocol is denoted as R-UDP. R-UDP provides reliability by means of a Selective Repeat explicit request ARQ with cumulative ACKs and limited to one SOAP message on top of UDP.

In general, a SOAP binding specification has to enable one or more Message-Exchange Patterns (MEPs). The presented UDP binding enables the One-Way and the Request-Response SOAP MEP. Both MEPs can be used not only unicast, but also a broadcast or multicast transport is possible by using the corresponding IP addresses. In case of a multicast/broadcast transport only the request is sent out to a multicast/broadcast address, the responses have to be unicast.

From ISO/OSI reference model point of view, depicted in Fig. 7, the core SOAP parser provides upper layers four Service Access Points (SAPs) by using WS-Addressing [14] and optionally WS-Security. It couples either to an unreliable or a reliable binding. These four SOAP SAPs result from all variations of the two supported MEPs and two bindings. The unreliable binding orients on the specification [15], which has been proposed in order to enable WS-Eventing [16], WS-Discovery [17], which are the basis for the Device Profile for Web Services [3]. The Device Profile for Web Services aims at defining a UPnP like service framework based on Web Service technologies.

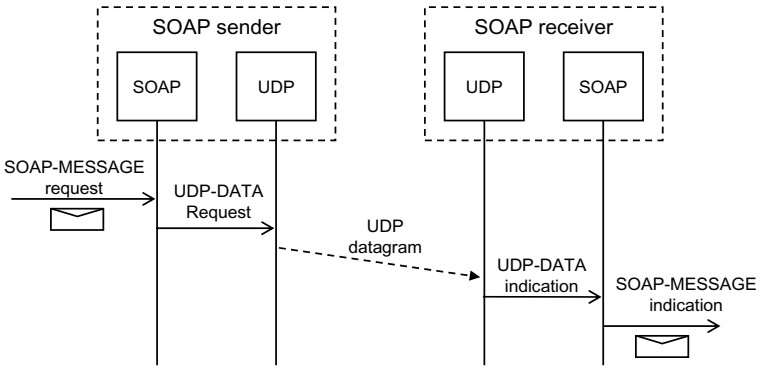

Fig. 8. Sequence diagram of an unreliable One-Way SOAP message transmission over UDP.

4.1.3. One-way MEP support

The One-Way MEP representation is naturally supported by UDP, since UDP already defines a one-way transmission of datagrams. Therefore, the SOAP One-Way service can be mapped directly to a UDP-DATA service. The SOAP sender uses the "UDP-DATA request" service primitive and in case of an error free transmission the SOAP receiver will be informed by a "UDP-DATA indication" service primitive. The sequence chart of an error free One-Way SOAP message transmission over UDP is depicted in Fig. 8.

Attention have to be paid to the maximum length of the payload of a datagram. If the length of the SOAP message exceeds the max. length of a datagram payload, the SOAP message has to be segmented into several datagrams and sent to the receiver. The receiver has to assemble the datagrams to the original message.

The use of a WS-Addressing elements within the SOAP header is not mandatory, but they are recommended in order to identify and de-multiplex requests at the server. Thus, only one UDP port assignment is sufficient to accept requests for different services.

4.1.4. Request-response support

The Request-Response SOAP MEP mapping to UDP is more complex, because UDP does not provide any comparable service. Since the Internet protocol stack retains unchanged, all additional features are implemented within the SOAP layer. The correlation between the request and response message is realized by using Web Service Addressing (WS-Addressing). Each message can be identified by a Universally Unique Identifier (UUID) that resides within the `<wsa:MessageID>` tag. The reply `<wsa:ReplyTo>` and destination address `<wsa:To>` specify the message flow between the service requestor and provider.

Figure 9 shows the sequence diagram of a successful SOAP over UDP Request-Response communication. As in the One-Way case, only the UDP-DATA service is used. The Request-Response session consists of two separate datagram transmissions. The SOAP node is responsible for correlating transmitted and received SOAP messages. The required WS-Addressing information is included within the SOAP header.

The example illustrated by the Listings 10 and 11 show the SOAP envelopes for a Request-Response session between two mobile terminals *s55* and *p900*. The request with the message ID `51F6DC39-1A5372A9-62087DF1-C34F5921` is sent from the *s55* to the *p900* terminal (Listings 10).

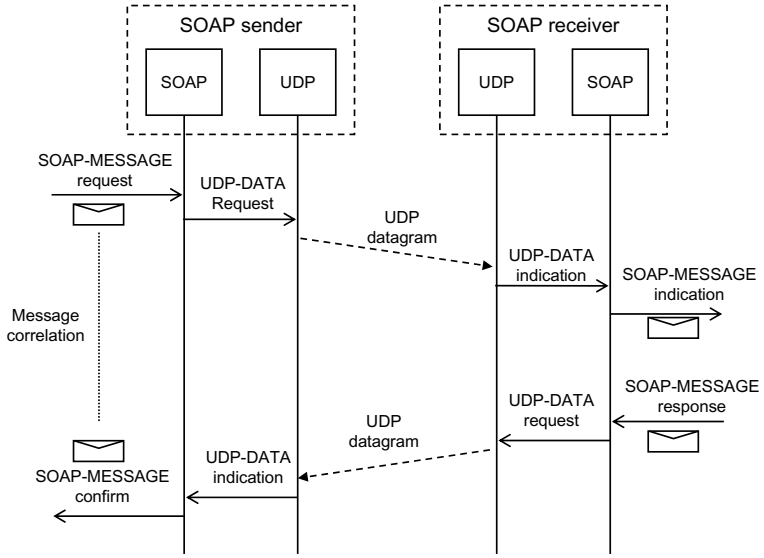

Fig. 9. Sequence diagram of an unreliable Request-Response SOAP message exchange.

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/...
    xmlns:wsa="http://www.w3.org/2005/03/addressing">
    <S:Header>
        <wsa:MessageID>
            51F6DC39-1A5372A9-62087DF1-C34F5921
        </wsa:MessageID>
        <wsa:ReplyTo>
            <wsa:Address>
            datagram://s55.comnets.rwth-aachen.de/soaprpc
            </wsa:Address>
        </wsa:ReplyTo>
        <wsa:To S:mustUnderstand="1">
            datagram://p900.comnets.rwth-aachen.de/soaprpc
        </wsa:To>
        <wsa:Action>
            http://comnets.rwth-aachen.de/Contacts/Delete
        </wsa:Action>
    </S:Header>
    <S:Body>
        <f:Delete
        xmlns:f="http://comnets.rwth-aachen.de/Contacts">
            <ContactID> 2383-145 </ContactID>
        </f:Delete>
    </S:Body>
</S:Envelope>
```

Fig. 10. SOAP request, using a binding to UDP with WS-Adr. properties.

The response SOAP message from *p900* to *s55* with the message ID `3F9202A6-D72B94f2-92F2A572-B19632D3` contains the message ID of the request message within the `<wsa:RelatesTo>` tag of the response message SOAP header.

From the use of WS-Addressing for managing Request-Response sessions the advantage appears that

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/...
    xmlns:wsa="http://www.w3.org/2005/03/addressing">
    <S:Header>
    <wsa:MessageID>
        3F9202A6-D72B94f2-92F2A572-B19632D3
    </wsa:MessageID>
    <wsa:RelatesTo>
        51F6DC39-1A5372A9-62087DF1-C34F5921
    </wsa:RelatesTo>
    <wsa:To S:mustUnderstand="1">
        datagram://s55.comnets.rwth-aachen.de/soaprpc
    </wsa:To>
    <wsa:Action>
        http://comnets.rwth-aachen.de/Contacts/DeleteAck
    </wsa:Action>
    </S:Header>
    <S:Body>
        <f:DeleteAck
        xmlns:f="http://comnets.rwth-aachen.de/Contacts"/>
    </S:Body>
</S:Envelope>
```

Fig. 11. SOAP response, using a binding to UDP with WS-Adr. properties.

it is possible to mix different SOAP transmission alternatives within one Request-Response session. For instance, a short SOAP request message could be sent by using the unreliable UDP binding and the response by using HTTP, since the response is too large for the unreliable UDP transport. For that purpose, the SOAP server has to manage all SOAP bindings jointly.

4.1.5. Unreliable binding

The unreliable binding simply transfers SOAP messages within one user datagram. Since the datagram payload size is theoretically limited to 65507 bytes (in practice this value is dependent on the implementation), the size of a SOAP message is limited too. A fragmentation of SOAP messages into many user datagrams is not useful, since one lost segment causes the loss of the whole SOAP message. Messages of a size greater than the max. payload size should be transmitted using the reliable binding described in the next paragraph. The R-UDP SOAP binding supports message fragmentation and retransmissions of lost datagrams.

If the requirements regarding the SOAP message size is fulfilled, both MEPs can be supported. By using the Request-Response MEP the client side changes its role to a SOAP receiver after transmitting the SOAP request. Thus, the client has to open a UDP server socket and to specify its address within the `</wsa:ReplyTo>` tag of the SOAP request message.

4.1.6. Reliable binding

Although the default HTTP binding provides a reliable message transmission, a reliable UDP binding is reasonable. HTTP and TCP are typically used to download web pages or files. For these long-lived data flows of several Kbyte or Mbyte TCP is meaningful, since it provides a reliable transmission in combination with congestion control. Thus, the network capacity is shared between different entities and network congestions are avoided.

However, the Web Service communication behavior is different. SOAP RPC calls result in short-lived data flows of approx. 100 byte to 10 Kbyte of data per SOAP message. Especially in low-bandwidth and

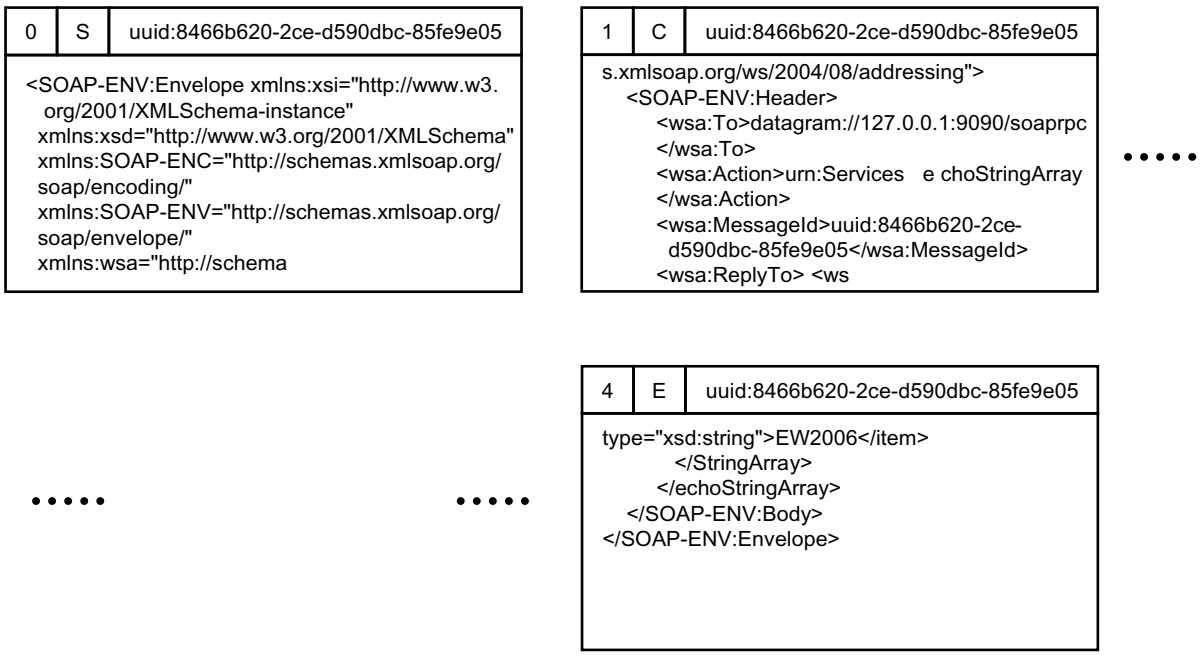

Fig. 12. Segmentation of a SOAP envelope to UDP datagrams with an additional R-UDP header.

high-delay mobile communication systems the transmission of short-lived data flows by HTTP and TCP is very inefficient [18]. For each RPC call a new TCP connection has to be established. In the majority of cases the transmission is proceeded within the TCP slow-start phase.

This problem has been already treated in the past by defining several alternative transport layer protocols, like e.g. the Transactional TCP (T/TCP) [19], which aims at providing an efficient transaction-oriented (request/response) service, or the Reliable UDP (RUDP) [20,21] that is intended to provide a UDP based protocol with guaranteed-order packet delivery and less complexity/overhead than TCP. In fact all alternative transport layer protocols can be used for Mobile Web Services and performance advances are expected. However, a different protocol, referred to as Reliable UDP (R-UDP) is introduced that has the additional advantage to be strongly aligned to Web Services protocols and traffic properties.

The aim of the R-UDP binding is to remedy the disadvantages of HTTP and TCP in order to increase the performance of Mobile Web Services calls respective their RPC delay. For a Mobile Web Service Access this approach is not reasonable, but Mobile Web Service Provisioning and especially Mobile P2P Web Services benefit from a reliable UDP binding. In the P2P case both terminals are using the alternative R-UDP protocol, changes of core network or third party service provider components are not needed.

The reliable SOAP binding to UDP is using a Selective Repeat ARQ (SR-ARQ) mechanism with explicit requests and cumulative ACKs restricted to one SOAP message. Thus, the ARQ window size is fixed during the transmission of one SOAP message. The protocol introduces an additional header, including a sequence number, message type flag, and the WS-Addressing Message-ID as depicted in Fig. 12.

The sender splits the SOAP message in M datagrams and transmits each datagram sequently. The unique Message-ID of the additional header is equal across all datagrams corresponding to the same SOAP message. The first datagram has an START flag (S), the following datagrams a CONTINUE

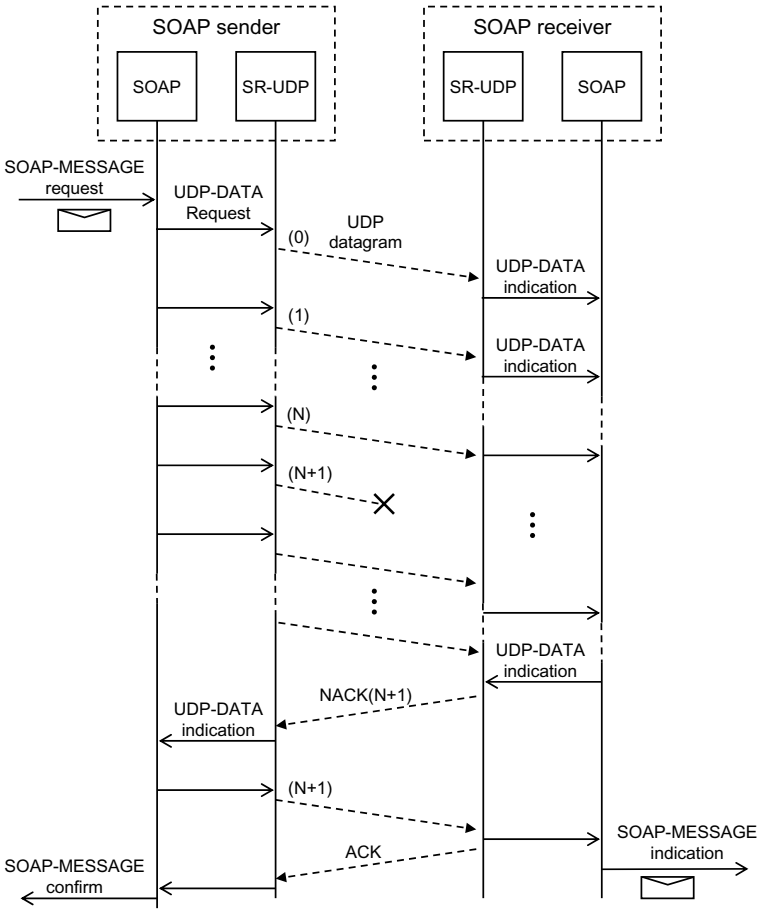

Fig. 13. Sequence diagram of a reliable One-way SOAP message exchange using a selective-repeat-ARQ.

flag C, and the last one an END flag E. The sequence number of the datagrams is increased with each datagram by one, starting from 0. Figure 13 shows an exemplary sequence diagram of transmissting M datagrams, where the datagram with sequence number $(N + 1)$ gets lost.

After sending the last datagram the sender waits T seconds for a final acknowledgement and indicates the successful transmission of the message if the ACK is received. In case no acknowledgement has been received during the time T, the SOAP sender either can retransmit the whole message or indicate an error to the application.

The SOAP receiver opens a new buffer for each received datagram when ever a new Message ID has been received. All datagrams with this message ID are saved according to their sequence number in the corresponding position of the buffer. After writing the datagram with the END flag into the buffer or waiting T seconds, the receiver requests the missing datagrams with one cumulative NACK(..,..). If the message is completed, a final ACK message is sent to the SOAP sender, otherwise the Selective-Repeat explicit-request procedure will be repeated.

The advantage of the SR-ARQ mechanism is the less amount of acknowledgements and retransmissions especially if the datagram loss rate is low. In the error free case only one additional ACK message is sent compared to a transmission using the unreliable binding. Besides, all lost segments within one retransmission phase are entail only the actual retransmissions and one additional ACK or NACK.

4.2. *Mobile Web Services Control Plane* (*signalling*)

Mobile Web Services signalling is used to manage changes of Web Service properties, i.e. changing Web Services endpoints (Uniform Resource Locator (URL), IP address) or the availability of the services themselves. Additional information could be included as well, like QoS or reliability of the service.

The Mobile Web Service signalling is divided into two tasks, mobility management and session management. The mobility management is responsible for signalling events related to the mobility of the user/terminal/service. These events can be e.g. the change of the RAN (changing IP address), handover to an alternative terminal, and the deployment and un-deployment of services. The session management enables the establishment, update, and release of Mobile Web Service sessions. The session is defined as a logical relation between at least two border nodes (terminals or network elements). Mostly a session key or ID is used to uniquely identify a session.

For the purpose of the MobWS mobility and session management SIP is used due to the following reasons:

- SIP is IP based.
- SIP follows the same notation as HTTP
- SIP provides rich application layer control functionality
- SIP is the main protocol of the IMS

Therefore, a Web Service framework can be easily extended by SIP towards a Mobile Web Service framework supporting mobility and session management. The integration of this Mobile Web Service framework to the network operator's IMS is enabled as well as the class of Mobile P2P Web Services.

4.2.1. *Mobility management for mobile web services*

A distinguishing characteristic of Mobile Web-Services (MobWSs) compared to conventional Web Services is their mobility, since users with their terminals regular change their location and, thus, the RAN they are connected to. In addition, "equal" MobWSs are may deployed on different terminals. The term "equal" mean that the offered interfaces are equal, the implementation and even the QoS can be different. This enables the user to start a MobWS session on one terminal and to continue the session with another terminal later (terminal handover).

The sequence diagram in Fig. 14 illustrates an exemplary MobWS mobility support. In this case a MobWS has been published (e.g. through the IMS) without waiting for a session establishment. The following Section 4.2.2 explains the publication of services within a P2P session.

Two user terminals (UT1 and UT2) are registering to a SIP server. In this case the SIP server takes over the role of a REGISTRAR server, which can be deployed on a Call Session Control Function (CSCF) server of the IMS. After successful registration, the IP addresses can be resolved by the SIP server depending on the SIP addresses (see (1)–(4) in Fig. 14). The presented example is limited to the registration of the terminal's IP address, but without loss of generality arbitrary properties can be registered. The prototype, described in Section 3, registers in addition to the IP address, the current presence state of the user, properties of the RAN (type, file mode/Modulation and Coding Scheme (MCS), QoS), and Mobile Web Services published (MobWS Service IDs). Other peers are may allowed to subscribe to these values in order receive notifications about the requested events.

Before the actual SOAP call (8), the MobWS Proxy resolves the IP address of the remote MobWS by calling the SIP UA *getIPAddress* method (6). In the example illustrated in Fig. 14, the MobWS Proxy can subscribe to the remote MobWS in order receive notifications about IP address changes of UT2,

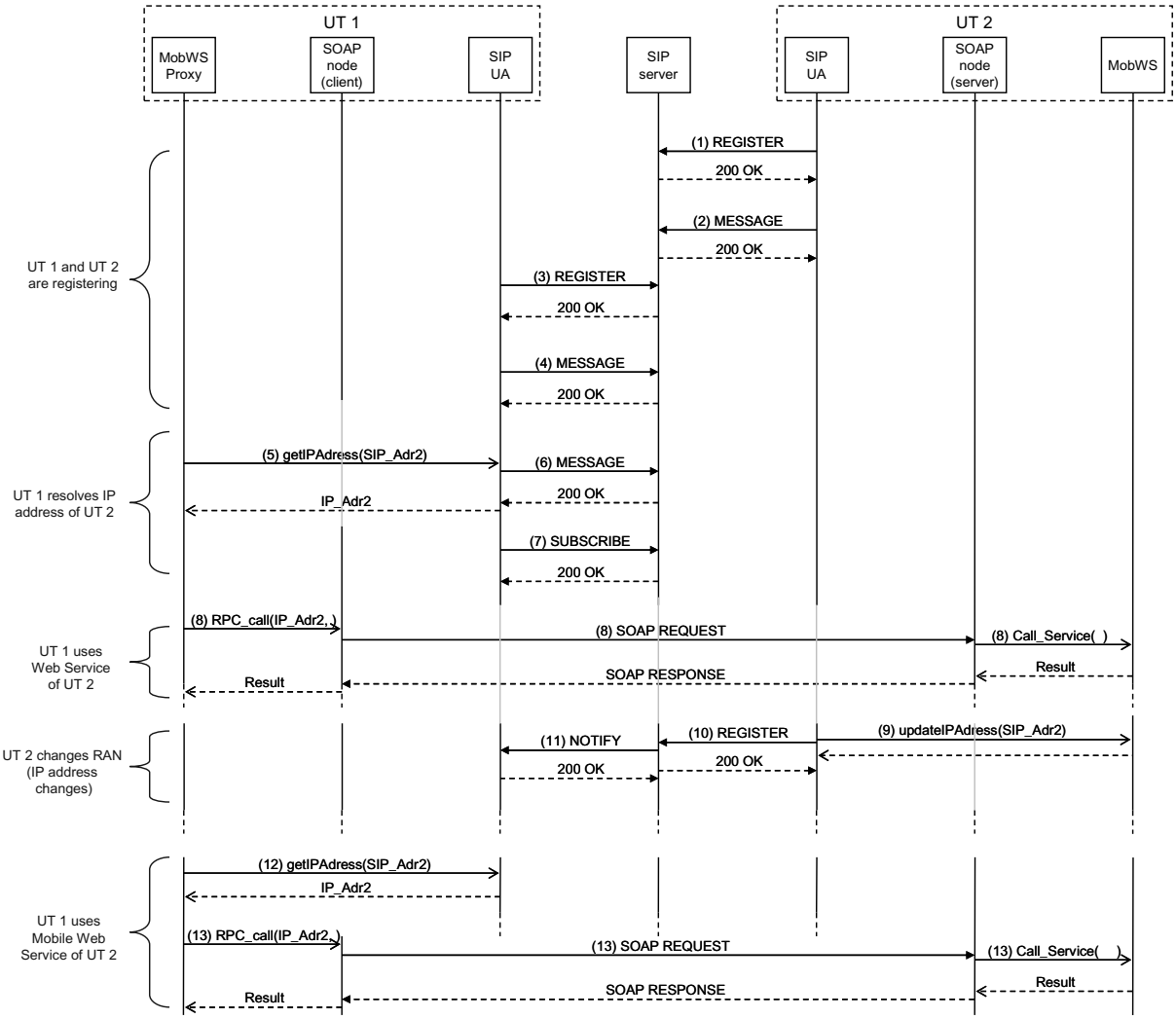

Fig. 14. Message sequence chart of SIP based Mobile Web Service mobility support.

see (10), (11) in Fig. 14. A following MobWS call (13) is therefor always addressed to the correct IP address.

Alternatively, the SIP AU could resolve the IP address of the MobWS not until a MobWS request has to be sent. However, in this approach is less beneficial for high-delay communication networks, since the overall RPC delay increases with the address resolution time.

Other mobility management approaches like Mobile IP Version 6 (MobIPv6) are not controversial to the SIP approach. They can be used in combination [22]. MobIPv6 takes over the responsibility of managing IP mobility and SIP is still responsible for service and terminal mobility.

4.2.2. Session management for mobile web services

If a Mobile Web Service should not constantly be published, but per request, a SIP session establishment procedure can be used. SIP sessions in combination with the Session Description Protocol (SDP) are used in the definition of an IMS session too. The Mobile Web Service framework make use of these

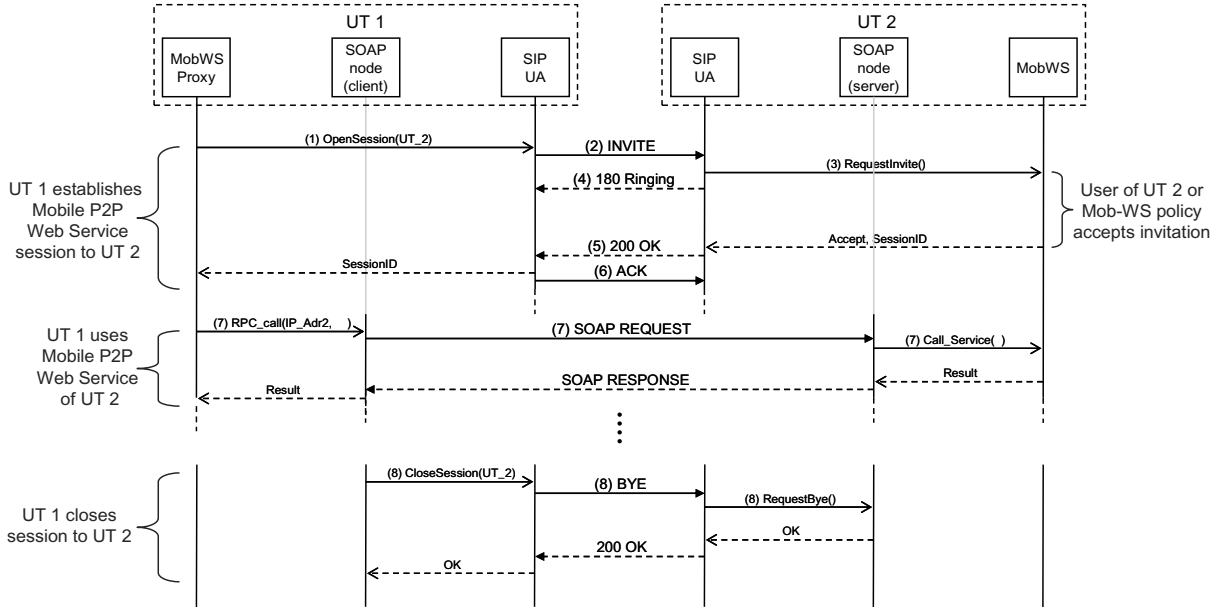

Fig. 15. Message sequence chart of SIP based Mobile P2P Web Services sessions.

session management protocols in order to enable a service publication and consumption per request. In Fig. 15 the message sequence chart of a Mobile Web Service session establishment and release is presented. Both terminals can be Web Service requestor and provider at the same time, but to simplify matters, UT1 is requesting a service of UT2 only. Thus, UT1 covers only a MobWS proxy and UT2 solely a MobWS.

UT1 opens a session, as shown in Fig. 15 (1), by sending from its SIP agent an INVITE (2) directly to UT2. The SIP/IMS infrastructure resolves and routes the specified address to the terminal. In addition, an IMS application server could be triggered, if user 2 has specified a filter rule concerning an INVITE message of user 1. After a provisional SIP response 180 Ringing and in case UT2 accepts the incoming session request, a final SIP 200 OK is received by UT1 and answered by a SIP ACK message.

During the SIP session establishment, the unique Mobile Web Service IDs are shared in terms of the OFFER/ANSWER model of SDP [23]. The SDP OFFER is carried inside of the *SIP:INVITE* message. The SDP ANSWER could already be transmitted with the provisional response. The following example illustrates an SDP OFFER for a Mobile Web Service session using a UDP SOAP binding. The unique MobWS ID labels the service and is used to register the service within the IMS resp. to a SIP REGISTRAR server.

In this case the OFFER from UT1 indicates to be the initiator of a SOAP over UDP session. Therefore the direction attribute is set to active. Since UT1 does not know where the answering party likes to connect to, the port number offered in the "$m =$" line is set to 9 (the discard port) and there is no contact attribute.

The respective ANSWER message is presented in Fig. 17. The "$m =$" statement corresponds to the "$m =$" statement of the OFFER. There the port for the UDP session is set to 112. The contact attribute carries the Web Service endpoint URL and the direction attribute confirms that UT2 will accept the connection.

In case, UT1 is initiating a Mobile P2P Web Service session (e.g. a chess game), the SDP OFFER within the SIP INVITE message contains the offered Web Service interface. The respective SDP OFFER

```
v=0
o=- 289083125 289083125 IN IP4 comnets.rwth-aachen.de
s=-
t=0 0
c=IN IP4 0.0.0.0
m=data 9 UDP SOAP
m=Mob-WS-ID:http://www...de/Mob-WS/IDs/GPS-Service
a=direction:active
```

Fig. 16. SDP OFFER for a Mobile Web Service session request.

```
v=0
o=- 2890844526 2890844526 IN IP4 comnets.rwth-aachen.de
s=-
t=0 0
c=IN IP4 0.0.0.0
m=data 112 UDP SOAP
a=contact:http://ut2.comnets.rwth-aachen.de/soaprpc
a=direction:passive
```

Fig. 17. SDP ANSWER for a Mobile Web Service session.

```
v=0
o=- 28908365 289088316 IN IP4 comnets.rwth-aachen.de
s=-
t=0 0
c=IN IP4 0.0.0.0
m=data 9 UDP SOAP
m=Mob-WS-ID:http://www.comnets...de/Mob-WS/IDs/MChess
a=contact:http://ut2.comnets.rwth-aachen.de/soaprpc
a=direction:active
```

Fig. 18. SDP OFFER for a Mobile P2P Web Service session establishment.

description is shown in Listing 18. This information is used by UT2 to contact UT1's Web Service. The SDP ANSWER of UT2 remains unchanged to the previous case.

During the session, both peers (terminals) are using the Web Service of their counterparts. After finishing the session, both peers are able to close the session by sending a SIP:BYE message. In order to enable a terminal handover or the freezing of a MobWS session, the application state has to be captured and submitted either to the terminal that continues the session or to the IMS resp. SIP infrastructure. A re-establishment of the session, identified by the SIP session ID (SIP dialog), is possible if the SIP dialog can be located within the session history of the IMS.

5. P2P service creation

Mobile P2P Web Services can be used in order to simplify the sharing of arbitrary application interfaces with other peers. These services have to be well defined, specified, and identified by an unique Mobile Web Service ID. If there are different Mobile Web Service interfaces with the same functionality a standardization have to take place or application developers have to implement for one application different interfaces in order to enable the interworking of nearly all kind of applications of this domain.

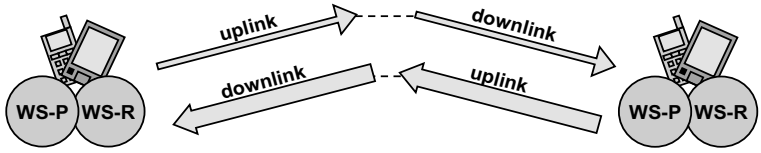

Fig. 19. Mobile P2P WS performance analysis scenario, 1 kbyte requests and 2, 4, 6, 8, 10 kbyte responses.

For example, somebody has designed a chess game Web Service interface, one can load the WSDL description and generate with common Web Service tools the needed server and client components in the programming language of the specific device. Using this middleware as a basis, the application developer can implement the Graphical User Interface (GUI) and logic, which is appropriate to the device capabilities. Thus, the same Mobile Web Service interface definition of the exemplary chess game can be implemented for instance on a mobile game console with a 3D graphic environment in C++ on Symbian OS and for a simple mobile phone under J2ME and rudimentary graphical capabilities. The interworking of these application is always warranted in case the Mobile Web Service interface definitions has been used. The automated process of the interface integration in the programming environment guarantees the standard conform implementation.

6. Mobile P2P Web Services performance

For the Mobile P2P Web Service delay performance analysis, a analytical model of the mobile link (GPRS, multislot class 10, Coding Scheme CS2 has been modeled), the network layer (TCP, R-UDP), and the session and presentation layer has been used. The analysis is based on a Hidden Markov Model (HMM) channel that simulates radio block errors and their correlation. ARQ mechanisms on link and transport layer correct these errors and entail additional delays.

The analyzed scenario (Fig. 19) take the delays of four wireless transmission per Mobile P2P Web Service session into account. The SOAP request is transmitted in uplink and downlink direction to the Mobile Server and the SOAP response in uplink and downlink direction, too. The request has a size of 1 kbyte and the responses vary from 2 kbyte to 10 kbyte. The HMM channel has a mean Block Error Ratio (BLER) of $p_2 = 0.1$ and a transition probability of $p_{22} = 0.8$. The segment loss probability if set to $p_{SL} = 0.001$.

In Fig. 19 the round-trip delay Complementary Distribution Function (CDF) of a P2P RPC is depicted. The delay gain of R-UDP compared to HTTP is always around 7% and independent from the response message size. In total the delay is quite high, between 5 s and 15 s. However, taking the Mobile P2P Chess game as an example the request and response messages do not exceed 1 kbyte and a delay of 3 or 4 seconds is acceptable in that application case. More interactive and realtime applications are not useful over GPRS. Even the R-UDP will not change this. Only a higher bandwidth and faster channel access procedures enable realtime P2P application between mobile users.

7. Conclusion

The paper has presented a novel P2P service creation framework based on Mobile Web Services and a SIP infrastructure, like the IMS. The service Creation is based on a Mobile Web Service middleware

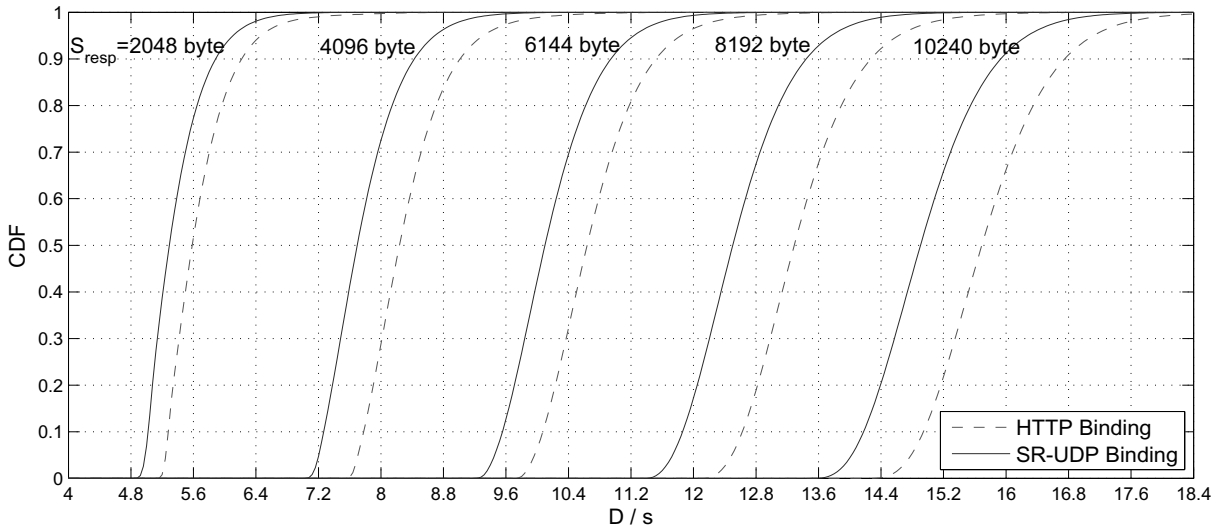

Fig. 20. Mobile P2P WS RPC delay, 1kbyte request, 2, 4, 6, 8, 10 kbyte response using HTTP or R-UDP, $p_{SL} = 0.001$, GPRS multislotclass 10, CS2, canonical HMM channel with $p_{22} = 0.8$, $p_2 = 0.1$.

supporting HTTP and UDP client and server bindings, which are coupled with SIP user agents. The framework has been proofed by an exemplary mobile chess game for J2ME devices.

Performance analysis show that the use of an alternative transport binding (R-UDP) to SOAP enhances the response time of the P2P application about 7%.

References

[1] G. Gehlen, F. Aijaz, Y. Zhu and B. Walke, *Mobile P2P Web Service Creation using SIP*, in Proceedings of the 4th international Conference on Advances in Mobile Computing and Multimedia (MOMM2006). Yogyakarta, Indonesia: Austrian Computer Society, Dec 2006, pp. 39–48. [Online]. Available: http://www.comnets.rwth-aachen.de.

[2] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion and C.F.D. Orchard, *Web Service Architecture*, Published on the internet, Feb. 2004, w3C Recommendation. [Online]. Available: http://www.w3.org/TR/ws-arch/.

[3] S. Chan, C. Kaler, T. Kuehnel, A. Regnier, B. Roe, D. Sather, J. Schlimmer, H. Sekine, R.D. Walter, J. Weast, D. Whitehead, D. Wright and Y. Yarmosh, *Device Profile for Web Services*, Published on the internet, May 2005. [Online]. Available: http://www-128.ibm.com/developerworks/ webservices/library/specification/ws-eventing/.

[4] Y. Zhu, *Sitzungsverwaltung von mobilen Web-Service Anwendungen mittels SIP*, Master's thesis, RWTH Aachen University, Chair of Communication Networks, June 2006.

[5] B. Godeny, *Jsr 180: Sip api for j2me*, Internet, jan 2005. [Online]. Available: http://jcp.org/en/jsr/detail?id=180.

[6] J. Rosenberg, H. Schulzrinne, C. Huitema and D. Gurle, *A Presence Event Package for the Session Initiation Protocol* (*SIP*), RFC 3856, Aug. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3856.txt.

[7] P. Saint-Andre, *Extensible Messaging and Presence Protocol* (*XMPP*)*: Core*, RFC 3515 (Standard), Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3920.txt.

[8] J. Rosenberg, H. Schulzrinne, C. Huitema and D. Gurle, *Session Initiation Protocol* (*SIP*) *Extension for Instant Messaging*, RFC 3261, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3428.txt.

[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261, June 2002. [Online]. Available: http://www.ietf. org/rfc/rfc3261.txt.

[10] W. Chou, F. Liu and L. Li, *Web Service for Tele-Communication*, in Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on, 19–25 Feb. 2006, pp. 88–88. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs all.jsp?arnumber=1602220.

[11] C. Werner and C. Buschmann, *Compressing Soap Messages by Using Differential Encoding*, in Web Services, 2004. Proceedings. IEEE International Conference on, 6–9 July 2004, pp. 540–547.

[12] G. Gehlen and R. Bergs, *Performance of Mobile Web Service Access using the Wireless Application Protocol* (*WAP*), in Proceedings of World Wireless Congress 2004. San Francisco, USA: University Aachen, Communication Networks, 05 2004, pp. 427–432. [Online]. Available: http://www.comnets.rwth-aachen.de/guge-pub.html.

[13] L. Pham and G. Gehlen, *Realization and Performance Analysis of a SOAP Server for Mobile Devices*, in Proceedings of the 11th European Wireless Confernce 2005, Vol. 2. Nicosia, Cyprus: VDE Verlag, Apr 2005, pp. 791–797. [Online]. Available: http://www.comnets.rwth-aachen.de.

[14] M. Gudgin, M. Hadley and T. Rogers, *Web Services Addressing 1.0 – Core*, Published on the internet, May 2006, w3C Recommendation. [Online]. Available: http://www.w3.org/TR/ws-addr-core/.

[15] H. Combs, M. Gudgin, J. Justice, G. Kakivaya, D. Lindsey, D. Orchard, A. Regnier, J. Schlimmer, S. Simpson, H. Tamura, D. Wright and K. Wolf, *SOAP-over-UDP*, Published on the internet, 2004. [Online]. Available: http://msdn.microsoft.com/library/ en-us/dnglobspec/html/soap-over-udp.pdf.

[16] D. Box, L.F. Cabrera, C. Critchley, F. Curbera, D. Ferguson, A. Geller, S. Graham, D. Hull, G. Kakivaya, A. Lewis, S.B. Lovering, M. Mihic, P. Niblett, D. Orchard, J. Saiyed, S. Samdarshi, J. Schlimmer and I. Sedukhin, *Web Services Eventing* (*WS-Eventing*), Published on the internet, Aug. 2004. [Online]. Available: http://www-128.ibm.com/ developerworks/webservices/library/specification/ws-eventing/.

[17] J. Beatty, G. Kakivaya, D. Kemp, T. Kuehnel, B. Lovering, B. Roe, C.S. John, J. Schlimmer, G. Simonnet, D. Walter, J. Weast, Y. Yarmosh and P. Yendluri, *Web Services Dynamic Discovery* (*WS-Discovery*), Published on the internet, Apr. 2005. [Online]. Available: http: //msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Discovery.pdf.

[18] S. Pack, S. Ahn, Y. Choi and S. Choe, *Wireless Tcp Model for Shortlived Flows*, in Vehicular Technology Conference, 2003. VTC 2003- Spring. The 57th IEEE Semiannual, vol. 3, 22–25 April 2003, pp. 1725–1729vol.3.

[19] R. Braden, *RFC 1644: T/TCP – TCP Extensions for Transactions Functional Specification*, July 1994, status: EXPERIMENTAL. [Online]. Available: ftp://ftp.internic.net/rfc/rfc1644.txt.

[20] D. Velten, R.M. Hinden and J. Sax, *RFC 908: Reliable Data Protocol*, July 1984, updated by RFC1151 [21]. Status: EXPERIMENTAL. [Online]. Available: ftp://ftp.internic.net/rfc/rfc908.txt.

[21] C. Partridge and R.M. Hinden, *RFC 1151: Version 2 of the Reliable Data Protocol* (*RDP*), Apr. 1990, updates RFC908 [20]. Status: EXPERIMENTAL. [Online]. Available: ftp://ftp.internic.net/rfc/ rfc1151.txt.

[22] S. Zeadally, F. Siddiqui, N. DeepakMavatoor and P. Randhavva, *Sip and Mobile Ip Integration to Support Seamless Mobility*, in Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on, vol. 3, 5–8 Sept. 2004, pp. 1927–1931Vol.3.

[23] J. Rosenberg and H. Schulzrinne, *An Offer/Answer Model with the Session Description Protocol* (*SDP*), RFC 3264, June 2002. [Online] Available: ttp://www.ietf.org/rfc/rfc3264.txt.

Guido Gehlen (guge@comnets.rwth-aachen.de) received his diploma degree in Electrical Engineering from the RWTH Aachen University, Germany, in 2001. In 2002 he joined the Chair of Communication Networks (ComNets), RWTH Aachen University, as a Research Engineer where he was involved in a German logistics project and in the European research project IST-MYCAREVENT. He has finished in Oct. 2007 his PhD thesis entitled “Mobile Web Services – Concepts, Prototype, and Traffic Performance”. Since March 2007 he is working for Ericsson Eurolab as a project leader of the German research project “Cooperative Cars”.

Fahad Aijaz (fah@comnets.rwth-aachen.de) received his Bachelors in Computer Science from Mohammad Ali Jinnah University in Karachi before he earned his Masters degree in Media Informatics from RWTH Aachen University in 2006. Since then he is serving the Chair of Communication Networks (ComNets) of RWTH Aachen University, where he is working toward his Ph.D. degree. He participates in an IST project “MYCAREVENT”, and is responsible for Mobile Web Services research projects within UMIC excellence cluster (http://www.umic-aachen.de). Some of his research interests include Mobile Web Services, middleware for mobile applications, ubiquitous computing, asynchronous mobile Services.

Bernhard H. Walke (walke@comnets.rwth-aachen.de) is an internationally recognized expert (numerous publications, including several internationally published books) in the field of communication networks. His groupfls research work focuses on formal specification, modeling, mathematical and simulative analysis, and pilot implementation of protocols and services for communication networks. Having worked at AEG-TELEFUNKEN Research Institute, and later at Fern University of Hagen (Chair for Data Processing Techniques), he has held the chair of Communication Networks at Aachen University since 1990. He is member of various national and international expert groups and coordination boards (e.g., ITG, AG Mobikom, WWRF) and has frequently served as an advisor to the German national regulation body (RegTP).