An enhanced UDP SOAP-Binding for a mobile Web Service based Middleware

Guido Gehlen, Fahad Aijaz, Bernhard Walke
RWTH Aachen University
Communication Networks
Kopernikusstr. 16, 52074 Aachen
{guge, fah}@comnets.rwth-aachen.de

***Abstract*— A Web Service based Middleware for mobile applications is a promising platform to accelerate the application development for mobile systems. Currently, the Web Service protocol stack is using by default the Simple Object Access Protocol (SOAP) on top of HTTP. Since HTTP is using TCP, the round-trip time of a Web Service call is long due to the connection establishment of Transport Control Protocol (TCP)(three-way-handshake) and the ARQ protocol (slow-start phase).**

Some applications do not require delivery guarantees of TCP. For these applications an unreliable transport of SOAP messages over the User Datagram Protocol (UDP) is a natural choice. If an application needs a reliable transport of SOAP messages a simple Automatic Repeat Request (ARQ) mechanism on top of UDP is better choice. This reliable UDP binding can be used as a supplement for the default HTTP binding in order to deal with the mentioned lacks of TCP. The session management of HTTP is substituted by introducing Web Services Addressing properties within the SOAP header.

This paper will present the architecture and the realization of an unreliable and reliable UDP SOAP binding. Both bindings will map two basic MEP to UDP services. The reliable binding encapsulates a selective-repeat (explicit request) ARQ protocol on top of UDP which ensures even the transmission of SOAP messages over several datagrams. Some use cases and examples motivate the work and stress the importance according to the mobile middleware.

I. Introduction

Service Creation for mobile distributed applications is a challenging area of activity due to the heterogeneous environment (multitude of devices and communication systems), lack of a consistent service creation environment and middleware.

The term middleware is not uniformly defined, but for a good discussion see [1]. In the context of this work middleware is a software layer between the communication protocols and the application and programming environment. For application developers it hides the complexity of the communication and distributed system. For the application it provides services which enable distributed computing and especially for a mobile middleware it additionally provides context information.

Web Services gain more and more in importance as a middleware for applications which are loosely coupled over the internet. The escalating technological advancements in the era of distributed computing and communication systems utilize Web Services as a core of interaction. The Service Oriented Architecture (SOA) as the main concept behind Web Services aims to achieve the same success as the World Wide Web (WWW)[2].

A Web Service based Middleware for mobile applications is a promising platform to enable a platform independent way of distributed computing and accelerate the application development for mobile environments. Section II-A summarizes the Web Services basics which are relevant for this work and II introduces the overall architecture of the developed Web Services based middleware. Fundamental features of this middleware are additional protocol bindings, policy driven object monitors, and support for future Plug-and-Play services in ad-hoc networks.

In all three extensions, the use of UDP is a natural choice. UDP provides an unreliable transmission of message, which is used to send out real time data, such real-time events, or probe messages. Real time messages have to be send out as fast as possible to ensure their relevance at receive time. A retransmission of such messages would be unnecessary, since they got obsolete in the meanwhile. Probe messages have to be sent to a broadcast address or to a multicast group. UDP enables broadcast and multicast communication. In section III-A the motivation for a SOAP binding to UDP is elucidated in detail.

In addition, this paper will show that UDP can be used to reliably transport messages, if a simple ARQ protocol will be implemented on top of UDP. The paper will introduce the design and realization of the binding. A further paper will show that these extensions perform better than using the default Web Service transport Hypertext Transport Protocol (HTTP) over TCP [3]. All extensions are developed on the internet application layer and do not require changes of internet protocols.

The presented realization is geared to the technical specification [4]. The terminology and the basic SOAP-Binding framework is taken from [5], [6] and the protocol layer description leans against the ISO reference model [7].

II. Mobile Web Services based Middleware

In order to provide applications and application developers an Application Programming Interface (API) to a mobile

distributed environment, the Mobile Web Service based middleware has been developed. Although the complex Web Services technologies seemed to be too heavyweight for a mobile environment, the flexibility and extensibility of Web Services facilitate the definition of a mobile middleware. The advantages are:

- Compliance to the Extensible Markup Language (XML)
- Linkable to arbitrary XML Schema Document (XSD) ontology
- Bindable to any arbitrary messaging/transmission protocol
- Platform and programming language independent
- Support from a multiplicity of development environments
- Existing security specification (WS-Sec.)

An overview of the middleware is given in section II-B, but before a brief overview of Web Services and the Simple Object Access Protocol (SOAP) is given. Further information about this middleware can be found in [8], [9].

A. Web Services and SOAP

XML Web Services are already widely used as a middleware for interacting internet applications. The Web Services middleware is based on the SOA [2] specified by the World Wide Web Consortium (W3C). To achieve high interoperability, all SOA entities use a common language for service description, messaging, and service registration. Web Services are using the XML as a common language. For messaging the SOAP [10], [5], [6] is used.

The SOAP envelope is structured in XML as well and will be delivered by an arbitrary protocol, by default HTTP. Interfaces are described in an XML subset, the so called Web Service Description Language (WSDL) [11], [12]. This description includes all the information needed in order to invoke service methods from other nodes and is used to build automatically client service-proxy (stub) code. The service-proxy represents the remote service, i.e. all published remote service methods are methods of the local proxy object. This architecture bridges the native messaging inside the client environment to the platform independent messaging in the SOA environment.

In addition to the basic Web Services elements (SOAP and WSDL), several extra specifications, such as Web Services Security, Web Services Addressing, and Web Service Choreography complete the Web Services middleware [1].

In conclusion, the Web Services specifications can be classified as a platform and programming language independent middleware framework. Thus, and due to its flexibility it is possible and reasonable to implement and extend a Web Services middleware for mobile systems. In the next section a brief overview of such a Mobile Web Services based middleware is presented. The developed UDP binding is part of this middleware, but can also be used separately.

[1]Web Services Activity - The goal of the Web Services Activity is to develop a set of technologies in order to lead Web services to their full potential, see http://www.w3.org/2002/ws/

B. Mobile Web Services based Middleware Architecture

The Web Services based middleware architecture is depicted in figure 1. The middleware provides upper layers services which support on the one hand the development of applications, on the other hand the application will be assisted at runtime. For instance, the middleware provides the application developers the possibility to invoke remote services, publish local services, monitor remote data, enable the monitoring of local data, and to manage context information.

At runtime the middleware manages the object exchange between services, discovers new services, and reacts on context changes.

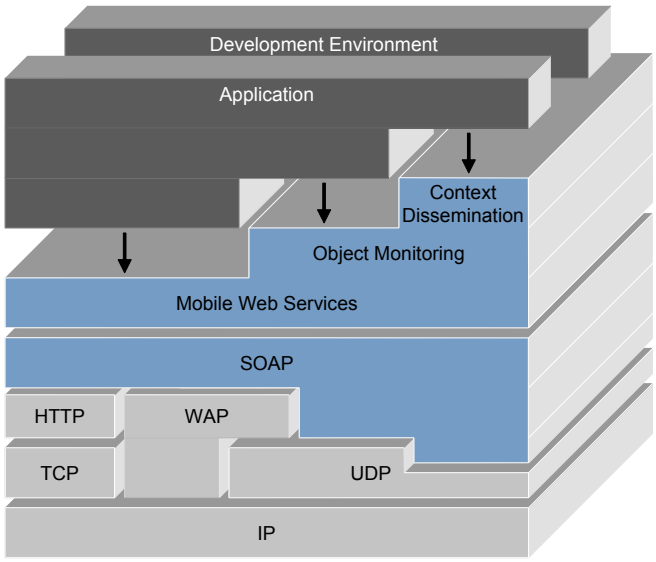

Fig. 1. Mobile Web Service based Middleware Architecture

The middleware is capable to couple to different underlying protocols, either to a session layer protocol, like HTTP, BEEP, or WSP, or to a transport layer protocol, like TCP or UDP. On top of these protocols a conversion layer, the SOAP layer, affiliates the communication domain to the computing domain. The main focus of this layer in respect of the ISO/OSI reference model is the ISO layer 6, the presentation layer, but with the absence of a session management and reliability mechanisms, SOAP takes over these functionalities. Depending on the application the SOAP layer also includes aspects of the application layer, since SOAP facilitates a basic object exchange application.

By default the Web Services middleware framework uses the HTTP binding, thus, session management is handled by HTTP and the reliability of the message transport by TCP. By using UDP we have to differentiate between different use cases. SOAP messages could be transmitted without awaiting a response or to route the message to a further node. In this case no session management is necessary. If no guarantee of the message delivery is mandatory, no reliability mechanism is necessary. By using the UDP binding as a replacement for the HTTP SOAP binding, session management and reliability

has been realized by SOAP, see section III.

III. UDP SOAP Binding

All SOAP bindings have to follow the SOAP Protocol Binding Framework [5]. The Binding framework defines general rules for a valid SOAP-Binding. The specification says:

"SOAP enables exchange of SOAP messages using a variety of underlying protocols. The formal set of rules for carrying a SOAP message within or on top of another protocol (underlying protocol) for the purpose of exchange is called a binding."[13]

In general, a SOAP binding specification has to enable one or more Message-Exchange Patterns (MEPs). Thus, MEPs are selected and implemented in the UDP binding based on the services provided by the underlying User Datagram Protocol (UDP).

In the following sections the UDP binding is motivated (section III-A), a brief overview of the UDP services is given (section III-B), the Web Services Addressing specification is introduced and mapped to the binding (section III-C), the supported MEPsIII-D are outlined, and a description of the realized binding, separated in an unreliable and a reliable one, is presented (section III).

A. Motivation for a SOAP Binding to UDP

The use cases for an unreliable and a reliable UDP binding are different. The unreliable binding primarily enables the use of Web Services within ad-hoc networks where services are discovered using multicast "Probe" messages, services announce themselves using "Hello" messages, or services disband by sending a "Bye" message. The specification "Device Profile for Web Services"[14] is following this approach. There, specifications like "Web Sevices Dynamic Discovery"[15] and "Web Services Eventing"[16] are used which themselves are based on a UDP binding. SOAP messages over UDP are used for service discovery and announcement as well as for sending events to a multicast group. The use of the unreliable transport of SOAP messages is mandatory due to the multicast or broadcast communication.

In addition, the unreliable binding can be used in cellular networks, for instance if real-time data (e.g. a position or sensor data) is transmitted. In the case that such real-time data messages are going lost a retransmission makes no sense, since this data possibly is already obsolete at a later time.

The reliable binding can be used in particular as a supplement for the default HTTP binding in mobile networks in order to avoid the mentioned disadvantages of TCP in mobile networks. Due to the high transmission delay of mobile networks, the TCP connection establishment (3-way handshake) and the slow-start phase in the ARQ mechanism entail a high latency of transmitting SOAP messages using the default HTTP binding. The connection establishment has a negligible effect on the performance, if SOAP sender and receiver communicating for a long duration via one TCP connecting without closing this connection, but usually SOAP messages are send only within one HTTP session and, thus, for each message a new TCP connection is established.

A context-aware Instant-Messageing (IM) system is one example application using this middleware, and thus, the UDP binding. The instant messages are transmitted with the reliable binding, the context changes are transmitted by using the unreliable binding. There is not only an added value regarding the performance in mobile networks, in addition application developers benefit, since they do not have to think about practical communication protocols. Application developers only have to decide, if the service methods should be accessible in a reliable or unreliable manner. Application developers who like to invoke remote methods only need to import the Web Service using the WSDL description. The protocol conversion is done automatically.

B. User Datagram Protocol

The User Datagram Protocol (UDP) [17] provides higher layer protocols an unreliable and connectionless service to send datagrams to remote hosts by using the Internet Protocol (IP) [18]. Entities which communicate via UDP are addressed by a port number of 16 bit. Thus, the UDP header consists of a source and a destination port in the first 4 octets of a UDP packet. The next 2 octets contain the length of the UDP packet including the header information. The last two octets of the header cover a checksum, see figure 2.

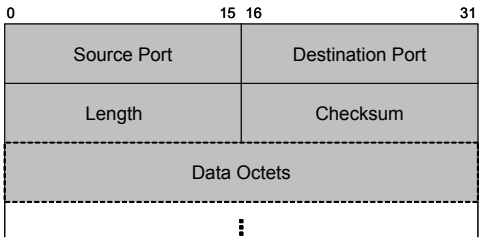

Fig. 2. UDP Packet Structure

From the ISO/OSI reference model point of view, UDP provides one service named "DATA" with two service primitives, UDP-DATA request and UDP-DATA indication. Before the upper layers can use these service primitives, the uper layer have to establish a service access point addressed by a UDP port.

Since UDP is connection-less it is possible to broadcast datagrams or to send datagrams to a multicast address. What is missing in UDP in order to bind it directly to SOAP is a session management and an enhanced addressing schema. In case reliability is required, a reliability mechanism is required, too.

C. Web Services Addressing

Web Services are designed to be flexible and independent of any transmission protocol. Indeed, addressing of Web Service nodes is dependent on the addressing mechanism of the used transmission protocol, e.g. the use of HTTP implies that

Web Services are addressed by means of Uniform Resource Locators (URLs).

The Web Services Addressing working group aims at defining transport-neutral mechanisms to address Web Services and messages exchanged by Web Services. The working draft of this specification [19] defines abstract properties which have to be integrated within other Web Services technologies, such as WSDL and mapped to concrete Web Services protocols, such as SOAP. Two constructs, message addressing properties and endpoint references have been designed in order to be extensible and re-usable.

A Web Service endpoint reference is a entity to which Web Service messages can be addressed. It supports the dynamic generation and customization of service endpoint descriptions, referencing of specific services instances, and dynamic exchange of endpoint information. The structure of a Web Service endpoint reference is illustrated by the XML infoset:

```
<wsa:EndpointReference>
    <wsa:Address>xs:anyURI</wsa:Address>
    <wsa:ReferenceParameters> ... </wsa:ReferenceParameters>
    <wsa:Metadata> ... </wsa:Metadata>
    <xs:any/>
</wsa:EndpointReference>
```

Message Exchange properties convey message characteristics including addresses for source and destination `<wsa:From>`, `<wsa:To>` and message identities `<wsa:MessageID>` as shown in the following XML infoset:

```
<wsa:MessageID> xs:anyURI </wsa:MessageID>
<wsa:RelatesTo RelationshipType="...">
    xs:anyURI
</wsa:RelatesTo>
<wsa:To> xs:anyURI </wsa:To>
<wsa:Action> xs:anyURI </wsa:Action>
<wsa:From> endpoint-reference </wsa:From>
<wsa:ReplyTo> endpoint-reference </wsa:ReplyTo>
<wsa:FaultTo> endpoint-reference </wsa:FaultTo>
```

The unique message ID `<wsa:MessageID>` can be used i.a. to enable the session management at the sender and the receiver. In typical enterprize applications, this message ID may be generated by setting up a database infrastructure that maintains the state of the unique ID. This state is then used by the application for synchronization and for the generation of new IDs.

The need of a shared resource, that is, the database, makes this scheme difficult to implement as it may require portability across different databases [20]. This affirms that the scheme is not suitable for the applications running on mobile devices in mobile communication and ad-hoc network environments.

Another non-database scheme for unique message ID generation can be to keep a singleton (an object with the existence of just one instance across an entire application) that only resides on the server application and acts as a synchronization point for the clients requiring unique message IDs [21]. This approach creates the scalability problem and may become a bottleneck [20].

The failure risk may become higher in mobile environments, which does not make this approach a natural choice. Therefore the UDP SOAP Binding has adapted an approach to generate these IDs in the memory by creating Universally Unique Identifier (UUID) that combines enough system information to make it unique across time and space, no matter when and where it was generated [20]. The UUID is a string-based message ID encoded in hexadecimal form. The string consisting of 32 HEX digists is composed of the unique current time in milliseconds (digits 1-8), the IP address encoded in hexadecimal (digits 9-16), the hex representation of the object hash code (digits 17-24) and an integer random number coded in hexadecimal form (digits 25-32), see figure 3.

The UUID generated with this algorithm is guaranteed to be unique across all devices in the network [20]. The algorithm performs faster than other schemes already mentioned as the entire process is executed in the memory without requiring any synchronization from one point or access to some shared resource such as databases. The freedom to avoid the databases and synchronization makes it simple to implement and a convincing choice for mobile and ad-hoc environments [20]. The layout of UUID used in UDP SOAP Binding is presented in figure 3.

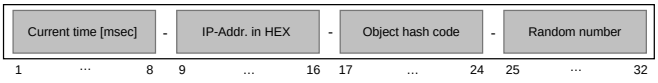

Fig. 3. Layout of Message UUID

The Web Services Addressing SOAP Binding specification defines the binding of these abstract WS-Adressing properties to SOAP and the Web Services Addressing WSDL Binding specification the binding to WSDL.

D. Supported MEPs

In general a SOAP binding must enable one or more MEPs. This section will select and specify the MEPs which are supported by the UDP SOAP binding and the mapping to SOAP and WSDL MEPs.

At SOAP level there are currently two specified MEPs, the SOAP request-response and SOAP response MEP. The SOAP response is not of interest, since a none SOAP request, as e.g. a HTTP GET request will be responded by a SOAP message.

This paper will concentrate on the request-response MEP and the very basic one-way MEP (this is no real MEP, since only one message is involved).

These two SOAP MEPs can be mapped to the WSDL MEP, which are defined in the [12], following table I.

SOAP MEP	WSDL 1.1 MEP	WSDL 2 MEP
SOAP request-response	Request-Response, Solicit-Response	In-Out, Out-In
SOAP one-way	One-way	In-Only, Out-Only

TABLE I
MAPPING OF SOAP, WSDL 1.1, AND WSDL 2MEPS

The mappings of these MEPs are important for the developer of sevice-proxy (stub) generators, since the WSDL

information regarding the new UDP binding and MEPs have to be represented in the sevice-proxy code.

The WSDL 2 ”MEPs In-Optional-Out, Out-Optional-In, Robust In-Only, Robust Out-Only” are similar to the MEPs listed in table I and will be not mentioned separately. The meaning of these MEPs is described in [12].

Both MEP, One-way and request-response, can be send out 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 send out to a multicast/broadcast address, the responsesf have to be unicast.

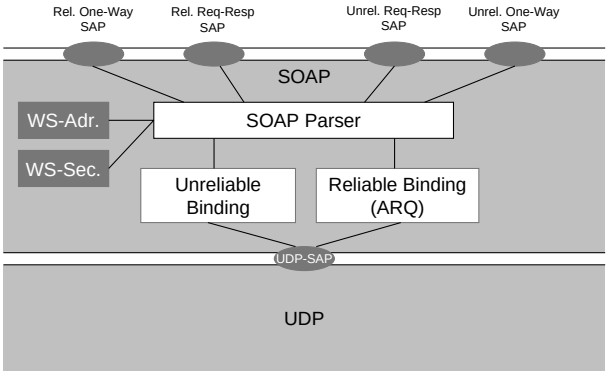

Fig. 4. SOAP layer concerning the binding to UDP

From OSI communications layer point of view, see figure 4, the core SOAP parser provides upper layers four Service Access Points (SAPs) by using Web Services Addressing properties and coupling either to an unreliable or an reliable binding. The four SOAP SAPs result from the variation of the two supported MEPs and bindings.

1) One-Way: The one-way MEP representation for a UDP binding is trivial, 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 request” service primitive.

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. This procedure is illustrated in section III-F about the reliable binding.

2) Request-Response: The Request-Response SOAP MEP mapping to UDP is more complex, since UDP does not provide any comparable service. Since we will not change the current internet protocol stack, all additional features are implemented within the SOAP layer. The correlation between the request and response message is realized by using Web Services Addressing properties [19]. Each message can be identified by a unique message identifier `<wsa:MessageID` and a source and reply address can specify the message flow between the service requestor and provider.

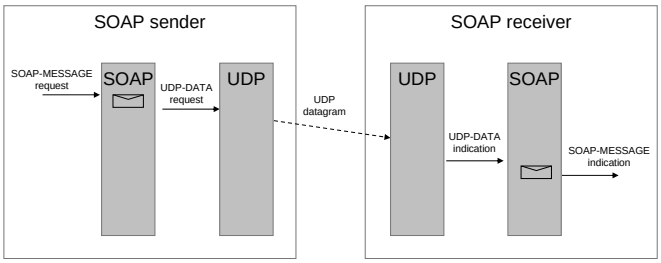

Fig. 5. Sequence diagram of a unreliable One-Way SOAP message transmission

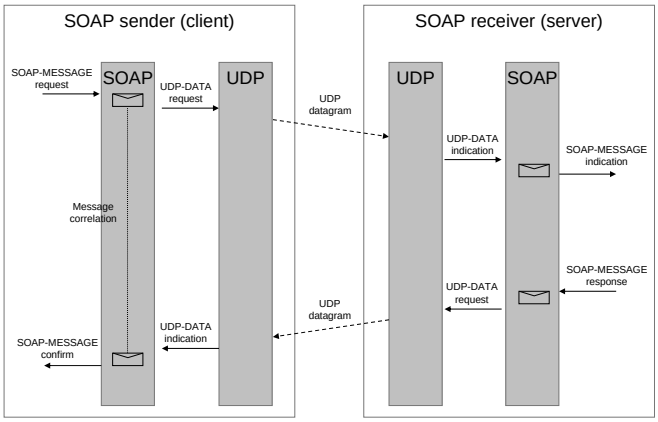

Fig. 6. Sequence diagram of a unreliable Request-Response SOAP message exchange

The Web Services Addressing information is stored within the SOAP header. The response header can be constructed from the request header as illustrated in the following example.

The request with the message ID *51F6DC39-1A5372A9-62087DF1-C34F5921* is sent from *mobile1* to *mobile2*

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/03/addressing">

    <S:Header>
        <wsa:MessageID>
            51F6DC39-1A5372A9-62087DF1-C34F5921
        </wsa:MessageID>
        <wsa:ReplyTo>
            <wsa:Address>
            datagram://mobile1.comnets.rwth-aachen.de/soaprpc
            </wsa:Address>
        </wsa:ReplyTo>
        <wsa:To S:mustUnderstand="1">
            datagram://mobile2.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> 4711 </ContactID>
        </f:Delete>
    </S:Body>

</S:Envelope>
```

The response message from *mobile2* to *mobile1* with the message ID *3F9202A6-D72B94f2-92F2A572-B19632D3* inserts the message ID of the request message into the `<wsa:RelatesTo>` tag of the response message.

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
    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://mobile1.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>
```

Thus, the session management of the request-response MEP is realized by correlating the `<wsa:MessageID>` identifiers at the SOAP sender.

E. 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 which should be transmitted by using this binding 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 section.

F. Reliable Binding

Although the default HTTP binding provides a reliable message transmission, since TCP provides the reliability on top of IP, a reliable UDP binding is useful in wireless networks. Low bandwidth and time-consuming media access and ARQ protocols of mobile communication networks cause great network delays. Thus, stop-and-wait-ARQ mechanisms should be avoided, since the transmission of a packet is initiated not till the acknowledgement of the previous packet has been received. The TCP connection establishment and slow start phase perform therefore bad in a wireless network.

For the reliable SOAP binding a selective-repeat-ARQ mechanism will be introduced which is adapted to the transmission of SOAP messages over UDP. Between UDP and SOAP a protocol will be included which enables the reliable transmission of arbitrary messages. In this paper we focus on the transmission of SOAP messages, thus, this protocol is placed within the SOAP binding. The protocol introduces an additional header, including a sequence number, message type flag, and the WS-Adressing Message-ID, see figure 7.

Fig. 7. Segmentation of a SOAP envelope to many datagrams with addition header information of the selective-repeat ARQ

The sender splits the SOAP message in M datagrams and transmits each datagram sequently. Each datagram corresponding to the same SOAP message carries in the ”Binding” header the unique Message ID, the first datagram has an *START* flag (S), the following datgrams a *CONTINUE* flag (C), and the last message an *END* flag (E). The sequence number of the datagrams is increased with each datagram by one, starting by 0, see figure 8.

After sending the last datagram the sender waits T second for a final acknowledgement and indicates the successful transmission of the message. If after this time no acknowledgement is received, the SOAP sender either can retransmit the whole message or indicate the application an error.

The SOAP receiver opens for each received datagram, containing a new Message ID, a buffer. All received datagrams with this message ID will be 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 with one NACK(..,..) datagram the missing datagrams. If the message is completed, a final ACK message is send to the SOAP sender, otherwise the selective-repeat explicit-request procedure will be repeated.

The advantage of the selective-repeat ARQ is the less amount of acknowledgements and retransmissions especially if the datagram loss rate is low. In this case only one additional ACK message is sent compared to an unreliable transmission. The delay is as long as using the unreliable binding.

IV. Implementation Architecture

As depicted in the figure 9, the server architectural implementation revolves around several components. This section

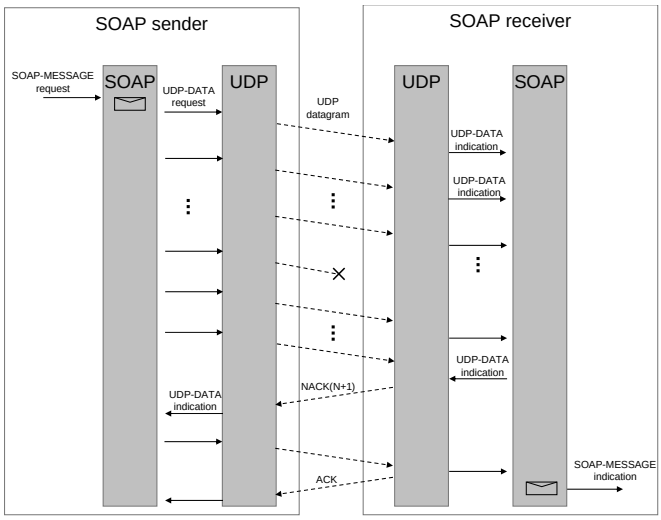

Fig. 8. Sequence diagram of a reliable One-way SOAP message exchange using a selective-repeat-ARQ

will explain how the Request-Response control flow is handled and processed by these components.

The SOAP server shares a common Graphical User Interface (GUI) among the two threads dedication for accepting either the UDP or HTTP requests. This is achieved by introducing the *ServerController* component that stands between the GUI and the listeners. This component acts as a delegate for the GUI and invokes the corresponding listener thread based on the input. Besides, the *ServerController* component also performs the logging operations for the SOAP server. In short, it controls the the other two components named as *UDPServerThread* and *HTTPServerThread*. This design makes the server architecture flexible enough to plug in additional protocol listeners with ease. We will only discuss the control flow for the *UDPServerThread* in this paper.

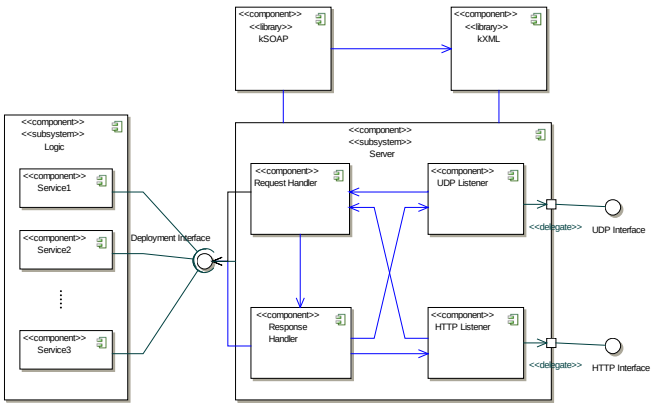

Fig. 9. Architecture of the mobile Web Services Server

Once the *ServerController* has invoked the *UDPServerThread*, which is the listener for the UDP requests, now its the job of this component to set up the initial configurations of the server by registering the hosted services, listen for the incoming requests and identify the transport mechanism which can either be receiving a complete SOAP message in a single datagram or the selective repeat mechanism, that is, receiving SOAP messages in segments. Once all these tasks has been done, the *UDPServerThread* passes on the received datagram to the *RequestHandler* for further processing.

Upon receival of the datagram by the *UDPServerThread*, the *RequestHandler* component extracts the SOAP message from the packet, parses its XML and extracts the Uniform Resource Identifier (URI) from the SOAP headers. In the next step the SOAP XML is de-serialized and passed on to the *DeploymentInterface* which invokes the requested service. Finally, the control flow from the *RequestHandler* is handed over to the *ResponseHandler* component for sending the resulting SOAP message back to the client via the UDP. The responsibility of the *ResponseHandler* is not just sending the result to the client, but also to attach the SOAP headers that conforms to the WS-Addressing [13] within the response SOAP message.

The Java2 Micro Edition (J2ME) technology due to its robust and flexible environment for the applications running on the consumer devices, has been adapted for the implementation of the SOAP server. On the other hand, *kXML* and *kSOAP*, the open source APIs for XML and SOAP parsing from Enhydra.org were utilized for manipulating the XML messages. These APIs supports Document Object Model (DOM) parsing using the ”pull” parsing technique which requires very low memory overhead.

V. Conclusion and Outlook

The paper has presented an enhanced SOAP binding to UDP and the realization for J2ME mobile phones. The binding follows and extends the technical specification [4] and uses the Web Services Addressing specification [19]. The extensions are mainly related to the reliable transport of SOAP messages over UDP and the segmentation of large messages into multiple datagrams. For this purpose, a Selective-Repeat protocol has been added to the reliable binding on top of UDP. The advantage is that no adaptation of the protocols are needed, the enhancements are realized on ”application” layer within the Simple Object Access Protocol (SOAP).

A further paper aims at evaluating the presented binding compared to the default HTTP binding. Latency measurements of remote procedure calls using the UDP and the HTTP binding will prove the strength of the UDP binding in mobile communication networks, like GPRS and UMTS.

References

[1] B. Aiken, J. Strassner, B. Carpenter, I. Foster, C. Lynch, J. Mambretti, R. Moore, and B. Teitelbaum, “Network Policy and Services: A Report of a Workshop on Middleware,” RFC 2768 (Standard), Feb. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2768.txt

[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/2004/NOTE-ws-arch-20040211/

[3] J. Postel, "DoD standard Transmission Control Protocol," RFC 761, Jan. 1980. [Online]. Available: http://www.ietf.org/rfc/rfc761.txt

[4] 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

[5] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen, "SOAP Version 1.2 Part 1: Messaging Framework," Published on the internet, June 2003, w3C Recommendation. [Online]. Available: http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[6] ——, "SOAP Version 1.2 Part 2: Adjuncts," Published on the internet, June 2003, w3C Recommendation. [Online]. Available: http://www.w3.org/TR/2003/REC-soap12-part2-20030624/

[7] ISO/IEC, "Information Technology - Open Systems Interconnection - Basic Reference Model: Conventions for the Definition of OSI Services," International Telecommunication Union, ITU-T Recommendation X.210, Nov. 1993. [Online]. Available: http://msdn.microsoft.com/library/en-us/dnglobspec/html/soap-over-udp.pdf

[8] G. Gehlen and G. Mavromatis, "Mobile Web Service based Middleware for Context-Aware Applications," in *Proceedings of the 11th European Wireless Conference 2005*, vol. 2. Nicosia, Cyprus: VDE Verlag, Apr 2005, pp. 784–790. [Online]. Available: http://www.comnets.rwth-aachen.de/guge-pub.html

[9] 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

[10] N. Mitra, "SOAP Version 1.2 Part 0: Primer," Published on the internet, June 2003, w3C Recommendation. [Online]. Available: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[11] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," Published on the internet, May 2005, w3C Recommendation. [Online]. Available: http://www.w3.org/TR/2005/WD-wsdl20-20050510

[12] R. Chinnici, H. Haas, A. Lewis, J.-J. Moreau, D. Orchard, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts," Published on the internet, May 2005, w3C Recommendation. [Online]. Available: http://www.w3.org/TR/2005/WD-wsdl20-adjuncts-20050510

[13] M. Gudgin and M. Hadley, "Web Services Addressing 1.0 - SOAP Binding," Published on the internet, Mar. 2005, w3C Working Draft. [Online]. Available: http://www.w3.org/TR/2005/WD-ws-addr-soap-20050331

[14] 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/

[15] 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

[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. Postel, "User Datagram Protocol," RFC 768 (Standard), Aug. 1980. [Online]. Available: http://www.ietf.org/rfc/rfc768.txt

[18] ——, "DoD standard Internet Protocol," RFC 760, Jan. 1980, obsoleted by RFC 791, updated by RFC 777. [Online]. Available: http://www.ietf.org/rfc/rfc760.txt

[19] M. Gudgin and M. Hadley, "Web Services Addressing 1.0 - Core," Published on the internet, Mar. 2005, w3C Working Draft. [Online]. Available: http://www.w3.org/TR/2005/WD-ws-addr-core-20050331

[20] F. Marinescu, Ed., *EJB Design Patterns, Advanced Patterns, Processes, and Idioms*. John Wiley & Sons Inc., 2002.

[21] P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace," RFC 4122, July 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4122.txt