This work-in-progress paper was presented as part of the main technical program at IEEE ETFA'2011

A CoAP based SOAP Transport Binding

Guido Moritz, Frank Golatowski and Dirk Timmermann
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18055 Rostock
Email: {guido.moritz,frank.golatowski,dirk.timmermann}@uni-rostock.de

Abstract

A huge momentum towards IP enabled Wireless Sensor Networks (WSN) appeared through the emerging 6LoWPAN protocols (i.e. IPv6 over Low power Wireless Personal Area Networks). By usage of existing cross domain open standards in contrast of proprietary solutions, deployments of WSNs are not tailored too tight for specific applications. Nevertheless, 6LoWPAN is only the first step for the usage of internet protocols in WSNs. Still efforts on higher layers on top of 6LoWPAN are an urgent need to provide seamless connectivity and interaction of highly resource constrained devices with higher valued services. This paper describes a new approach to bind SOAP to the emerging Constrained Application Protocol (CoAP) protocol. Thereby CoAP provides a lightweight but reliable transport binding for SOAP based protocols. Compared to the widespread TCP based HTTP binding, round trip times can be reduced by 43% in an exemplary scenario. Combined with dedicated XML compressors like the Efficient XML Interchange format (EXI), existing heavy weight SOAP based protocols become also applicable in WSNs.

1 Introduction

Research on wireless sensor networks (WSN) culminates in (re-)usage of existing internet technologies and protocols also in such highly resource constrained environments. While existing solutions for WSNs are often bound to specific link layers, internet technologies are cross domain applicable by design and built on top of IP. IP hides lower link layer capabilities and provides the required abstraction. A huge momentum towards IP enabled WSNs appeared through the emerging 6LoWPAN (i.e. IPv6 over Low power Wireless Personal Area Networks) protocols [2, 5]. But still efforts on higher layers on top of 6LoWPAN are an urgent need to integrate WSNs in higher valued applications.

The Devices Profile for Web Services (DPWS) is the current state of the art for Service-oriented Architectures (SOA) in distributed networking device centric environments and is already widely used in the automation industry [3]. To apply the SOAP based DPWS in resource constrained environments, the size of the XML SOAP envelopes can be reduced significantly by using dedicated XML compressors and encodings like for example the Efficient XML Interchange format (EXI) [10] as presented in [7]. For transport of the compressed or uncompressed XML documents, usually the HTTP binding or the UDP binding are used. This paper describes a new approach for a binding of SOAP to the emerging Constrained Application Protocol (CoAP) [8] of the IETF Constrained RESTful Environments (CoRE) working group. Even if CoAP is specified for interaction with and manipulation of resources in RESTful environments, it can also be used as transport mechanism for SOAP envelopes as described in this paper.

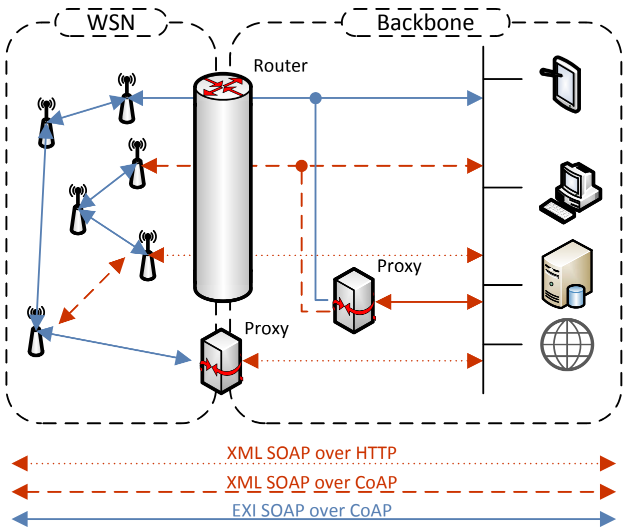

Figure 1. Messaging scenarios

Figure 1 depicts a few possible messaging scenarios. Sensor nodes inside a WSN are likely to use only highly optimized mechanisms and data representation like EXI encoded SOAP envelopes carried by the CoAP binding. For external communication, also this EXI encoded SOAP-over-CoAP can be used. If sensor nodes are capable of parsing both EXI and compliant XML, SOAP en-

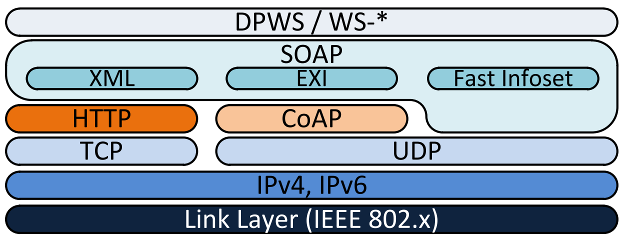

Figure 2. WS-* Stack with CoAP Binding

velopes may be encoded in compliant XML but still carried over CoAP. Finally also the usage of default XML encoded SOAP-over-HTTP messaging is still possible.

But both EXI and CoAP are designed to map seamless, transparent and stateless to XML and HTTP in dedicated proxy servers. Thus, endpoints are not necessarily required to use one specific encoding or one specific binding exclusively to conform other endpoints. The proxies can be used to re-encode for example external XML SOAP-over-HTTP to internal EXI SOAP-over-CoAP messaging and vice versa as depicted in figure 1. Thereby, existing SOAP deployments are not required to be changed. But the integration of WSNs in higher valued applications is possible with one comprehensive technology.

2 SOAP-over-CoAP Binding

Most SOAP deployments are using the existing SOAP-over-HTTP [6] binding for transport of SOAP envelopes. The SOAP-over-HTTP binding does not use all HTTP functionalities extensively, but uses HTTP for transport and e.g. for bypassing firewalls. But SOAP is not bound to HTTP exclusively. For example OASIS has defined the SOAP-over-UDP [4] binding. The intention of SOAP-over-UDP is to send SOAP envelopes over IP multicast.

Both the HTTP and the UDP binding are not optimal to be used in highly resource constrained networks like 6LoWPANs. HTTP is bound to TCP. The overall performance of TCP in 6LoWPAN networks is poor due to expensive handshakes for establishing and closing connections. UDP datagram messaging instead is lightweight at the expense of reliability.

The IETF Constrained RESTful Environments (CoRE) working group[1] is developing a protocol suite around the CoAP (i.e. Constrained Application Protocol) protocol [8]. Even if not identically, CoAP is leaned to HTTP. In contrast to HTTP, CoAP is based on UDP. Due to the unreliable nature of UDP, CoAP ads mechanisms for transport reliability and separates between the request/response layer (similar to HTTP) and the messaging layer. The messaging layer realizes transport reliability on application layer without the need for heavyweight handshakes.

Additional to the usage of reliable UDP instead of TCP, the advantage of the CoAP binding is a more compact message format, because the CoAP header is encoded binary and no string values are used like in HTTP. Figure 2 shows the protocol stack with the three feasible SOAP bindings.

Because CoAP is leaned to HTTP, CoAP defines a detailed mapping procedure of CoAP to HTTP in dedicated proxies. This CoAP/HTTP proxy feature can also be used to map the CoAP to the HTTP binding and additionally to re-encode optimized EXI data representation to compliant XML and vice versa as discussed above.

2.1 Usage of CoAP

Most of the required protocol mechanisms are already embedded in SOAP Web services through the appropriate usage of corresponding WS-* specifications. Similar to the existing SOAP-over-HTTP binding, the CoAP binding is not intended to fully exploit the features of CoAP. Among these not required features for this binding are certain discovery and eventing (c.f. publish/subscribe) features of CoAP that are already covered by e.g. WS-Discovery and WS-Eventing.

Nevertheless, the SOAP-over-CoAP binding makes appropriate usage of CoAP as application layer protocol. For example, successful and failure responses are indicated by the corresponding CoAP response codes (e.g. 2.xx, 4.xx, 5.xx).

2.2 Messaging Patterns

DPWS deployments support in general two different message exchange patterns (MEP). A one way MEP consists only of one SOAP message for a request without a SOAP envelope in the response. The two way MEP contains a SOAP envelope in both request and response. Thus, the SOAP-over-CoAP binding must cope with these SOAP request/response patterns additionally to the CoAP request/response patterns.

In addition to the CoAP request/response layer, CoAP defines the messaging layer to decouple request/response and transport. The messaging layer differentiates between confirmable (CON) and non-confirmable (NON) messages. CON messages must be acknowledged, while NON only consist of a single message without acknowledgement. If using a CON message for a request, the response can either be embedded synchronously in the acknowledge or send in an additional further message exchange. This asynchronous additional message exchange to transmit the response to the origin sender of the request can be again either CON or NON.

In summary, five typical message exchanges based on the messaging mechanism of CoAP exist. In a one-way MEP the request carries a SOAP envelope and the response consists only of the CoAP response header. In a two-way MEP the CoAP response header is additionally supplemented by a SOAP response envelope.

1. non-confirmable for request and response; provides no reliability

[1]http://tools.ietf.org/wg/core/

2. confirmable for request, response carried in message acknowledgement

3. separated confirmable messages for request and response (e.g. if processing time of message is higher than retransmission timeout); requires separated acknowledgments for both request and response with an empty message body

4. confirmable message for request, non-confirmable message for response; provides reliability only for the request

5. non-confirmable message for request, confirmable message for response; provides reliability only for the response

To achieve the required reliable message transport, the SOAP-over-CoAP binding should use the CON feature of CoAP. For unreliable messaging, the existing SOAP over UDP binding might be sufficient.

To match CON messages and acknowledgements and for duplicate detection, CoAP uses message ids which are unique for each message exchange. To match requests and responses instead, the *CoAP Token Option* is used. *WS-Addressing* defines the *message id* property to identify and match SOAP requests/responses in time and space and thus provides the same mechanism like the *CoAP Token Option*. The *CoAP Token Option* may be much more compact by providing equal functionality like the *WS-Addressing message id*. In case of using the CoAP binding, the *WS-Addressing message id* property can be empty and must contain a value only if the *CoAP Token Option* is not present or not sufficient. The intention of this adaptation is to reduce message size. The *WS-Addressing message id* property has a typical size of 45 bytes and cannot be compressed significantly because the value of this property is unique for each request/response.

2.3 Serialization

Both EXI and standard XML are based on XML-Infoset [9] and thereby interoperable. Hence, the SOAP-over-CoAP binding only requires the capability to represent XML-Infoset documents. This includes utf-8 XML for interoperability reasons with existing SOAP implementations and compressed XML like EXI.

2.4 Endpoint Identification

WS-Addressing defines an *anonymous* URI that can appear in the *address* property of an endpoint reference. If the *reply endpoint* property of a SOAP request transmitted over CoAP has an *WS-Addressing address* property with this value, the UDP source address and port are considered to be the address to which reply messages should be sent. If the *address* in the *reply endpoint* property is different from that value, the response is intended to be send to an endpoint different from the origin requester.

Table 1. Performance of SOAP-over-CoAP

	HTTP	HTTP keep alive	UDP	CoAP
No. of Messages	10	after TCP handshake: 2	2	2
Header Size Request	208	213	0	4
Header Size Response	136	122	0	6
Payload Size Request	819	794	919	843
Payload Size Response	752	848	845	841
Message Size Request	1027	1007	919	847
Message Size Response	888	970	845	775
Message Size Req & Res Sum	1915	1977	1764	1622
Round Trip in ms	2.25	1.35	1.23	1.24
Round Trip in %	100	59.9	54.6	55.2

Implementations of the CoAP binding must be aware of that difference. It might be required to send the acknowledgment of a CoAP CON message to the origin requester and use an additional CoAP message exchange to send the response to the third party endpoint indicated in the *WS-Addressing reply endpoint*.

3 Implementation

The CoAP binding is implemented to ensure comprehensive binding specification and to verify the approach. The implementation of the SOAP-over-CoAP binding is integrated in WS4D-gSOAP [11]. WS4D-gSOAP is a DPWS toolkit and is based on the well-known gSOAP stack [1]. For CoAP the libcoap[2] implementation is used. As dedicated scenario, the air conditioner example published along with the WS4D-gSOAP implementation is implemented. Because neither the HTTP nor the UDP binding are capable of featuring separated messages for request and response like the CoAP binding, the implemented scenario always includes the response payload in the CoAP acknowledgements to allow comparable performance results. As server and client, desktop computers where used connected over a laboratory Ethernet IPv6 backbone. A standard ICMP echo (i.e. ping) of 56bytes between the endpoints takes on an average 0.502ms, measured over 30.000 runs.

3.1 Performance Measurements

From the air conditioner realized to evaluate the binding, a two way MEP has been selected to be presented in this paper (i.e. setting the target temperature of the exemplary air conditioner and returning status information in the response). Table 1 presents the results.

The sizes in the table 1 are presented as follows

- The rows differentiate between request and response messages.

- *No. of Messages* refer to the number of messages on transport layer and thus also include TCP handshakes (e.g. SYN, SYN-ACK, FIN...). The keep alive

[2]http://sourceforge.net/projects/libcoap/

mechanism of HTTP causes only one TCP handshake and sending further application data and transport acknowledgements in one message instead of separating them.

- *Header Size* refers to the headers on application layer of HTTP or CoAP (UDP has no additional header).
- *Payload Size* refers to the SOAP envelope.
- *Message Size* refers to the sum of header and payload sizes.

All messages are smaller than the minimum MTU of IPv6 (i.e. 1280 byte), whereby IP layer fragmentation is avoided. This differences in the payload occur mainly through the usage of the *WS-Addressing message id* mechanism. This mechanism is only required in the HTTP binding with keep alive and the UDP bindings. For the HTTP binding without keep-alive this feature is not required because of the direct TCP end-to-end connection between the endpoints which is used only for one request/response. For the CoAP binding this feature is provided in a much more compact format by CoAP internal mechanisms.

While message size can be reduced for all messages, both HTTP bindings suffer from the additional required HTTP header. UDP instead requires no further header, but requires to use the *WS-Addressing message id* mechanism. The CoAP binding does not require this mechanism and is capable of providing the same functionality by using the internal messaging layer. Hence, the CoAP binding results in the most efficient overall solution to transport SOAP envelopes in resource constrained networks. Combined with the EXI encoding to reduce overhead of the payload, the advantage of smaller headers compared to HTTP is much more significant.

4 Conclusion

The SOAP based WS-* framework provides a huge number of cross domain protocol features. But the basic characteristics like data representation and transport mechanisms of SOAP Web services imply an overhead which prevents them to be applied in WSNs. Studies in [7] have shown that the data representation of SOAP can be tailored for endpoint capabilities. It can be based on compliant utf-8 XML or binary encoded and compressed while the different formats can be re-encoded stateless and transparent.

But using existing solutions also for the transport of SOAP documents in these networks is questionable. The CoAP protocol developed by the IETF CoRE working group is leaned to HTTP, but designed to fit better in 6LoWPAN limitations. This paper describes in detail an approach of a SOAP-over-CoAP binding for transport of SOAP messages in resource constrained environments. Combined with the mentioned binary encodings like for example EXI, SOAP Web services can also be deployed in bandwidth limited environments like WSNs.

Both EXI and CoAP are designed to map seamless and stateless to XML and HTTP. Existing implementations, based on compliant XML SOAP-over-HTTP, deployed on resource rich devices and used in higher valued applications not have to be changed. Dedicated proxies are responsible for the transparent mapping, whereby WSNs can be integrated in these higher valued applications and services with one comprehensive technology.

While bandwidth limitations can be met with the solutions proposed in this paper, the lower bound for memory requirements of implementations cannot be estimated clearly. The next steps include the integration of the SOAP-over-CoAP binding in the existing WS4D-uDPWS[3] implementation and development of an EXI parser for WS4D-uDPWS.

References

[1] R. A. V. Engelen. A framework for service-oriented computing with c and c++ web service components. *ACM Trans. Internet Technol.*, 8:12:1–12:25, May 2008.

[2] J. Hui and D. Culler. Ipv6 in low-power wireless networks. *Proceedings of the IEEE*, 98(11):1865 –1878, 2010.

[3] F. Jammes and H. Smit. Service-oriented paradigms in industrial automation. *Industrial Informatics, IEEE Transactions on*, 1(1):62 – 70, 2005.

[4] R. Jeyaraman. Soap-over-udp version 1.1. OASIS WS-DD TC, 2009.

[5] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), Sept. 2007.

[6] J.-J. Moreau, M. Gudgin, M. Hadley, N. Mendelsohn, Y. Lafon, A. Karmarkar, and H. F. Nielsen. SOAP version 1.2 part 2: Adjuncts (second edition). W3C recommendation, W3C, Apr. 2007. http://www.w3.org/TR/2007/REC-soap12-part2-20070427/.

[7] G. Moritz, F. Golatowski, D. Timmermann, and R. Stoll. Encoding and compression for the devices profile for web services. In *Proceedings of 5th International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE2010)*, april 2010. to appear.

[8] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained application protocol (coap). IETF Draft, 2011.

[9] R. Tobin and J. Cowan. XML information set (second edition). W3C recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-xml-infoset-20040204.

[10] W3C. *Efficient XML Interchange (EXI) Format 1.0 (W3C Recommendation)*, 2011.

[11] E. Zeeb, G. Moritz, D. Timmermann, and F. Golatowski. Ws4d: Toolkits for networked embedded systems based on the devices profile for web services. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 1 –8, 2010.

[3]http://code.google.com/p/udpws/