encDPWS - Message Encoding of SOAP Web Services

Compression and Encoding for Devices Profile for Web Services

Guido Moritz[1], Dirk Timmermann[1], Regina Stoll[2]
University of Rostock
[1] Institute of Applied Microelectronics and CE
[2] Institute of Preventive Medicine
18055 Rostock, Germany
guido.moritz@uni-rostock.de

Frank Golatowski
Center for Life Science and Automation
CELISCA
18119 Rostock, Germany
frank.golatowski@celisca.de

***Abstract*—Current efforts in real world deployment of wireless sensor networks end up in the development of the 6LoWPAN protocols. While big industrial consortia develop new technologies, architectures, and concepts, 6LoWPAN is heading towards improvements and adaptations of existing and matured network protocols like IP, TCP, and UDP. Because 6LoWPAN uses the low power WPAN standard IEEE 802.15.4 in lower layers, a standardized and seamless connectivity of low power wireless devices with existing networking infrastructure is achieved. But further research on suitable application layer protocols on top of 6LoWPAN is still required. This paper investigates on the applicability of the Devices Profile for Web Services (DPWS) in 6LoWPAN networks. Main scope is the necessary optimized message encoding, to reduce the overhead of this SOAP based protocol. Therefore, the paper introduces in current developments of encDPWS (encoded DPWS), a new encoding for DPWS.**

SOAP; Compression; Encoding; Devices Profile for Web Services; 6LoWPAN

I. INTRODUCTION

High research and development efforts have been made, to allow a seamless connectivity of emerging wireless sensor networks and low power ad-hoc devices with existing network infrastructures. While big industrial consortia like ZigBee and Bluetooth are developing new protocols, 6LoWPAN is heading to universal cross domain solutions based on existing and matured network technologies and protocols. 6LoWPAN is an acronym for IPv6 over Low power Wireless Personal Area Networks and the name of the corresponding active working group in the IETF [1]. The basis of 6LoWPAN is the adaption of IPv6 to allow the deployment of IPv6 in IEEE 802.15.4, a low power radio MAC and PHY specification, networks. But further developments are made to adapt also transport layer protocols like UDP.

Based on these 6LoWPAN protocols, the applicability of existing application layer protocols, which fit the resource constraints of the deeply embedded low power devices, is still missing. Existing and emerging middleware and architectural concepts like REST (Representational state transfer), Service-oriented Device Architectures (SODA) [2], and technologies like UPnP (Universal Plug and Play), JINI, and DPWS (Devices Profile for Web Services) are proper candidates to provide the required functionalities for low power networking devices. While UPnP and DLNA are focusing specific application scenarios like networked home and small office environments, DPWS is a cross domain technology and is widely used in the automation industry at device level [3] and it has been shown that they are also applicable for enterprise integration [4], [5].

Based on the current specifications of the OASIS "Web Services Discovery and Web Services Devices Profile" (WS-DD) [6] technical committee, this paper presents research efforts on the reduction of the message overhead, implied by the usage of SOAP [7] and thus XML data representation. The paper bases on experiences presented by Moritz et al. in [9]. The measurements of message sizes of a realistic scenario in [9] show clearly the partly significant compression rate of existing XML compressors. However, core requirement of application layer protocols in 6LoWPAN networks should be the size limitation of a single IEEE 802.15.4 frame, to avoid sending multiple frames for one application layer message. This requirement is difficult to meet the existing solutions as presented in [9]. Therefore, this paper describes ongoing efforts in a new encoding named encDPWS (encoded DPWS).

II. COMPRESSION OF DPWS MESSAGES

In [9], Moritz et al. presented an overview about compression and encoding schemes for DPWS specific messages. This section discusses further DPWS specific improvements and the concept of the new encoding scheme encDPWS.

A. Assumptions and Requirements

Based on our experiences with compressors and encoders, experiences from analyzing different scenarios, and experiences in our ongoing implementations of XML processing applications on resource constrained devices, we have investigated on developing an own, more DPWS specific encoding.

The encoding is performed on the border routers of a 6LoWPAN network and is only used for communication inside the 6LoWPAN network, to allow a transparent compression without changes in existing implementations. Border routers are assumed to be applied with sufficient resources for message re-encoding. By moving parsing and encoding tasks to the border routers, the highly constraint nodes with typically tens of kB RAM and ROM are relieved. The developments of encDPWS are driven by six principles:

- Compact message format and size
- Development of an encoding specific for well formed DPWS and SOAP and not for general XML compression
- One message including network and transport headers should fit into one IEEE 802.15.4 MAC frame (max. 102 Byte)
- Parsing can be done directly on the data and no conversions and caching are required
- Permit streaming parsing (i.e. starting processing the message even if it is not received completely, to avoid memory consuming storing of fragments)
- Permit vendor/application specific extensions (i.e. DPWS only provides basis functionalities and might be enhanced for dedicated scenarios)
- Stateless encoding (Communication with 6LoWPAN networks is transparent for external devices and implementations and compression is only applied in the 6LoWPAN network to omit message overhead. Nodes might change routes and/or subnets in mobile scenario, which requires stateless encoding on the border/edge routers.)

To keep message size as small as possible, the SOAP-over-HTTP binding is not applicable. Thus, encDPWS bases on the discussion in [9] and omits HTTP headers in general. Required HTTP header fields are carried in the message itself. On the border router of a 6LoWPAN, the required HTTP header fields can be included in the message without deep knowledge about semantic of the carried SOAP envelope. To shape encDPWS as stateless as possible, additional information are required in the message, to define the dedicated message as HTTP request or response.

Because current 6LoWPAN efforts do not define a compression for TCP headers, UDP is a reasonable alternative for the best possible compression of network and transport layer and to leave as much frame size as possible for application layer. But DPWS and WS-* implementations in general may depend the reliable nature of TCP, even if they should not and errors are indicated and resolved in upper layers. Thus, a comprehensive definition of a mapping of external TCP based message exchange and 6LoWPAN internal UDP message exchange is required, but out of scope of this paper.

B. encDPWS

In figure 1, the scheme for the encoding of encDPWS (encoded DPWS) is depicted. For clarity and due to size limitations, not the complete scheme can be presented here. The encDPWS encoding is based on the Tag-Length-Value (TLV) format, which is used extensively in lower layer (e.g. IPv6 extension header chaining) and is applicable on resource constraint devices. Required buffers are known before parsing starts and not supported fields can be left out without further need to parse their end by end-identifiers. Carrying length information for every data unit inline differentiates encDPWS from most known solutions, but increases parsing performance significantly.

```
<Message>       ::=  <Preamble> <Length> <StatusCode>
                     [<StringDef>] [<Tags>]

<StatusCode>    ::=  <ByteValue> <ByteValue>

<Preamble>      ::=  "D6"
<Length>        ::=  <ByteValue> <ByteValue>
<ByteValue>     ::=  0x00 | 0x01 | ... | 0x255

<StringDef>     ::=  <FreeIndicator> <ByteValue>
                     [<String>]
<FreeIndicator>::=   0x00 | 0x01 | ... | 0x1E

<Tags>          ::=  <Tag> | <Tag> <Tag>
<Tag>           ::=  <TagIndicator> <AttrLength>
                     [<Attributes>] <ValueLength>
                     [<Value>]

<AttrLength>    ::=  <ByteValue>
<Attributes>    ::=  <Attribute> |
                     <Attribute> <Attribute>
<ValueLength>   ::=  <ByteValue> |
                     <HierarchicalTagIndicator>
<Value>         ::=  <String> | <WellDefinedString>
```

Figure 1. encDPWS schema in Backus–Naur Form

The new encDPWS scheme does not differentiate between HTTP header information, SOAP header, and SOAP body, but is a straight forward cross layer design and includes all required information to be converted at the border routers into compliant formats.

encDPWS header. Each messages starts with a fixed preamble (magic bytes) to identify the protocol, followed by a two byte value indicating the overall length of the message, and the status code for the latter HTTP header. The StatusCode field may weather include the HTTP status code in binary format or 0x00 for requests. Own string definitions and the tags itself are carried in the encDPWS payload body.

Structur *indicators*. For string definitions and the tags, *indicators* are used to identify the contained data. These *indicators* are similar to events in SAX parsers or the events used in the EXI format and follow design principle of structure knowledge based encoding. Required namespaces and tag names are limited and known at compile time and depend only on the applied version of DPWS. Additionally, specific tags in DPWS SOAP envelopes always occur at defined order and hierarchy in the XML tree structure (e.g. wsa:MessageID in /soap:Envelope/soap:Header). Thus, *indicators* are well-known and base on the Web services specifications used by DPWS. They point to specific tags in the given XML tree structure. Up to now, we detected 125 different *indicators* to be defined, to identify all DPWS related tag names including namespaces. But additional Web services specifications and vendor specific extensions might be used in applications scenarios. For generic creation of a complete list, we are working on possibilities to generate the *indicators* tables out of the XML schema files of the Web

services specifications, similar to the schema-informed mode in EXI and Fast Infoset (FI). In contrast to existing solutions, encDPWS *indicators* define a complete path in the XML tree structure. Most existing encodings point only to tags in single hierarchical layers and the complete path is a composed list. Other XML non-specific compressors have no knowledge about XML structure and rearrange and separate strings and structural information in string tables, independent of syntactic or semantic meaning. However, current efforts concerning encDPWS also investigate on possibilities for a hybrid solution, to combine tag identification within a single layer with complex XML structure path *indicators*. But, the encoding is also aware of unsupported tags and values at runtime.

String tables. Own strings in encDPWS can be defined using the StringDef *indicator*. The StringDef *indicator* is followed by an arbitrary own *indicator* in the range of 0x00-0x1E. The third byte of a string definition specifies the length of the following string to be defined. String definitions are made before the tags itself, to allow applications buffer allocations and string table creation before start parsing itself. These string definitions can be used further as tag names and/or attribute values.

Tag attributes. Attributes carried in the XML tags are indicated by a nonzero attribute length value. If present, the values itself are carried inline immediately after the attribute length value. Because tag names may contain known attributes, their order and occurrence is previously defined additionally. This omits inline definition of attribute names and only their values are carried in the payload. Possible reordering of attributes is performed on the border routers also. Not known attributes may be carried inline fully after the well-defined attributes. After attributes, the length of the values of the tags is encoded in one byte. DPWS defines values to be carried by some fields. This concerns particularly discovery and eventing specific messages, where the wsa:Action carries well-known URIs. Thereby, 0x255 for the value length field is reserved as it specifies that the tag value is one byte pointing to previously known URIs, stored in permanent string tables on the nodes.

Message-id agreement. The wsa:MessageID fields in SOAP envelopes are major bandwidth consumers in the new encoding (45 bytes each in the specific scenario presented in section III). These message ids are used to decouple transport process from message exchange and allow asynchronous messaging independent of changing transport addresses or routes. But constraint sensor nodes not need to differentiate between a huge number of asynchronous requests and responses. Thus, a more lightweight representation for message ids can be used for internal communication. The format can be as compact as possible, to allow border routers to associate internal traffic and messages with external traffic and the corresponding valid message id. For simple deployments, the endpoint transport and network address might be sufficient, but may be extended to usage of few sequentially numbered bytes as message ids. This specific functionality is defined as *message-id agreement*.

TABLE I. ENCODING SCHEMES FOR SOAP ENVELOPES

Compressor	Average in Byte	Average in %	Minimum in Byte	Maximum in Byte
encDPWS	148,72	18,89	47,00	371,00
EXI [1]	153,72	19,60	66,00	354,00
EXI [2]	167,22	20,60	55,00	452,00
EXI [6]	196,17	24,13	66,00	533,00
EXI [3]	205,00	26,25	119,00	414,00
Fast Infoset [1]	218,39	27,98	103,00	455,00
Fast Infoset [2]	242,00	30,09	97,00	563,00
EXI [4]	315,67	40,31	192,00	630,00
XMLPPM	425,22	55,16	274,00	749,00
gzip (C=9)	425,56	55,66	297,00	755,00
gzip (C=1)	437,44	56,99	300,00	799,00
Xmill (C=9)	459,39	59,78	303,00	824,00
Xmill (C=1)	463,72	60,18	304,00	852,00
EXI [5]	467,77	59,64	234,00	1118,00
bzip2 (C=1)	474,78	61,82	315,00	852,00
bzip2 (C=9)	474,78	61,82	315,00	852,00
Fast Infoset [5]	561,89	69,70	315,00	1301,00
XML	814,89	100,00	418,00	2089,00

TABLE II. COMPARISSON OF ENCODING SCHEMES FOR SOAP ENVELOPES

Message Type	encDPWS	EXI [1]	FI [1]	XML
Hello	181	186	282	1107
Probe	114	96	144	668
Probe Match	220	192	281	1144
Dirceted Probe	47	77	116	567
Directed Probe Match	219	192	283	1145
Resolve	86	110	160	692
Resolve Match	148	175	258	973
Bye	96	124	189	757
Get Device Metadata	86	101	144	464
Get Device Metadata Response	371	354	455	2089
Invoke 1-Way	107	142	186	453
Invoke 2-Way	149	156	209	576
Invoke 2-Way Response	177	137	199	586
Event Subscribe	154	191	255	920
Event Subscribe Response	158	168	238	829
Event Delivery	173	152	219	622
Event Unsubscribe	144	148	210	658
Event Unsubscribe Response	47	66	103	418
Average	149	154	218	815
Maximum	371	354	455	2089
Minimum	47	66	103	418

[1]schema-informed (optimized) / compression with Deflate
[2]schema-informed (optimized) / without compression
[3]schema-informed (default) / compression with Deflate
[4]schema-less / compression with Deflate
[5]schema-less / without compression
[6]schema-informed (optimized) / without compression / byte aligned

Implied knowledge encoding. encDPWS also includes further specific built-in functionalities defined as *implied knowledge encoding.* Specific examples are:

- wsdp:Device type is supported by all DPWS devices and must not be carried in the discovery messages.
- Transport specific addresses of device and client carried in the SOAP envelopes are derived from lower layers.
- Default addressing scheme for endpoint negotiation is HTTP (due to SOAP-over-HTTP binding).

III. MEASUREMENTS AND COMPARISSON

To evaluate the different existing compression and encoding schemes for SOAP and thus for DPWS messaging, we implemented a test scenario by using our WS4D gSOAP DPWS toolkit available at [12]. Because measuring other performance parameters than message sizes is out of scope of this paper, we recorded all message types of a real scenario and made an offline evaluation. Deeply embedded devices are not expected to provide complex services. Thus, the exemplary implementation realizes basic functionalities that are required for sensor integration in networking device infrastructure. The used messages are the same as already used by Moritz et al. in [9], which makes the results comparable. The results presented here do not include *message-id agreement* functionalities. This example and the measured message sizes are only one possible realization and are only used to evaluate the compression schemes. Other implementations and measurements may vary.

Table 1 presents averages of compression of a set of 18 different message types of the scenario. Even if the main scope of encDPWS is to provide an efficient processing on deeply embedded devices, it results also in the smallest average message size due to *implied knowledge encoding*. However, the requirement of fitting on one IEEE 802.15.4 frame cannot be achieved for all messages, but might possible by applying *message-id agreement.* Table 2 compares particularly EXI, FI, and encDPWS and presents separated values for the 18 message types. All values only show the SOAP envelopes excluding HTTP headers.

IV. CONCLUSION AND FUTURE WORK

This paper presents encoding and compression of SOAP based Devices Profile for Web Services (DPWS) deployments for deeply embedded devices. Therefore, the new encDPWS (encoded DPWS) approach is introduced. Based on the design principles of encDPWS, no compression or decompression of the encoded message must be performed and no additional memory is required for parsing the messages, which is a major constraint on deeply embedded devices.

On the basis of measurements of different encodings and compressors, qualitative comparisons are possible. While the compression rates and possible improvements of existing compressors are known, further developments on encDPWS are required. This includes integration of version control, fault message integration, and validation of the encoding.

The results show, that it is possible to compress SOAP based Web service down to fit size requirements of low power deeply embedded devices. This allows DPWS deployments in 6LoWPAN networks and thus a seamless connectivity of low power sensors and actors with higher-value services in networking device infrastructure or the internet with one comprehensive cross domain technology. The next steps must include resource requirement analysis and performance evaluations and comparison with existing schemes.

ACKNOWLEDGMENT

This work has been achieved in the ITEA2 project uSERVICE and OSAmI and has been funded by the German Federal Ministry of Education and Research under contract numbers 01|S0902F and 01|S08003I.

We would like to thank Noemax Technologies Ltd. (http://www.noemax.com) for their considerable contribution regarding Fast Infoset.

REFERENCES

[1] IETF, IPv6 over Low power WPAN (6lowpan), Technical report, http://tools.ietf.org/wg/6lowpan/, online, 2008.

[2] Scott de Deugd, et al., "SODA: Service-Oriented Device Architecture," IEEE Pervasive Computing, vol. 5, no. 3, 2006, pages 94-C3.

[3] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - Service Infrastructure for Realtime Embedded Networked Devices: A service oriented framework for different domains," International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06), Washington, DC, USA, 2006, page 43.

[4] S. Karnouskos, et al., "Integration of SOA-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure," Emerging Technologies and Factory Automation (ETFA2007), Patras, Greece, 2007, pages 293-300.

[5] Elmar Zeeb, et al., "A context aware service-oriented maintenance system for the B2B sector," 3rd International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE 2008), Ginowan, Okinawa, Japan, 2008, pages 1381-1386.

[6] OASIS Web Services Discovery and Web Services Devices Profile (WS-DD) TC, www.oasis-open.org/committees/ws-dd/, online, 2009

[7] World Wide Web Consurtium (W3C) Recommondation, SOAP Version 1.2, Online, http://www.w3.org/TR/soap/, online, 2007.

[8] World Wide Web Consurtium (W3C), Efficient XML Interchange Working Group, Online, http://www.w3.org/XML/EXI/, online, 2009.

[9] Guido Moritz, Dirk Timmermann, Regina Stoll, and F. Golatowski, "Encoding and Compression for Devices Profile for Web Services," 5th IEEE International Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE2010), Perth, Australia, 2010, in press.

[10] W3C, Web Services Addressing (WS-Addressing), http://www.w3.org/Submission/ws-addressing/, online, 2004.

[11] W3C, Web Services Eventing (WS-Eventing), http://www.w3.org/Submission/WS-Eventing/, online, 2006.

[12] WS4D: Web Services for Devices, http://www.ws4d.org, online, 2009.

[13] IETF, RFC1951, http://www.ietf.org/rfc/rfc1951.txt, 1996.