Efficient XML Interchange in Factory Automation Systems

Rumen Kyusakov, Henrik Mäkitaavola, Jerker Delsing and Jens Eliasson
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology S-97187 Luleå, Sweden

Abstract—**The advent of Service-Oriented Architecture (SOA) in the automation domain has made possible the cross-layer vertical integration of devices, manufacturing systems and business processes. However, the use of standard web service technologies is not always possible in an industrial environment with high real-time requirements and limited hardware resources due to the overhead connected to XML processing. The work presented in this paper analyses the opportunities, advantages and challenges when applying the newly emerged Efficient XML Interchange (EXI) standard for XML encoding to the factory automation systems. The two major SOA-based automation middleware architectures, namely OPC Unified Architecture (OPC UA) and Devices Profile for Web Services (DPWS), were investigated. Furthermore, we present an EXI-based approach for extending the reach of the service technology covering deployments on resource constrained embedded devices.**

I. Introduction and related work

The transition from static, single-purpose manufacturing systems to multi-functional, highly flexible and configurable industrial processes requires support for complex interactions and data exchange among numerous components within the automation system. This induces the extensive application of object-oriented technology, distributed computing and service oriented approach in the development of today's middleware automation solutions. Many successful protocols and concepts used in the domain of enterprise systems and the Internet infrastructure are directly applied to the automation systems while others must be adapted to the real-time, robustness and safety critical requirements found in the industrial environment. Example of this is the utilization of the SOA approach and web service protocol stack in particular. Unification of the communication infrastructure i.e. the use of Industrial Ethernet, IP on the network layer, TCP/UDP as a transport and SOA on the application layer, is seen as a way to provide better interoperability and hence lower system complexity, deployment and maintenance costs. A motivation, broader overview and technology survey on the application of Service-Oriented Architecture in industrial automation are given by Jammes et al. [1].

Today's service technologies are often making use of XML for structuring the content and meta-data of the service messages. As an example, web service protocol called Simple Object Access Protocol (SOAP) [2] can be used as a means for defining the formatting of service requests and responses and specifying the interchange patterns. Each SOAP message is represented as a XML document with a predefined structure (XML schema). RESTful web services [3] are another SOA approach that does not rely on SOAP. Instead, the HTTP protocol capabilities are used to carry the service messages directly in the HTTP body. Although the data in the RESTful HTTP payload can be represented in different ways, in practice XML, JSON or similar encodings are applied. The verbose structure of XML and the high resource requirements for its processing is, in many cases, the bottleneck when applying the web service technology to the embedded systems often found on the plant floor. More detailed discussion and quantification of the performance overhead of using web services on deeply constrained networked embedded devices is provided in [4].

Different binary structured data representations, which are compatible with XML, have been suggested as an alternative to the text-based XML format. In [5], Sakr presents a survey on that with comparison of the most widely used compression techniques for XML. The binary XML working group of the World Wide Web Consortium (W3C) has also performed measurements and comparison between different structured data formats available in [6]. According to these studies the Efficient XML Interchange (EXI) standard [7] provides highest compactness ratio and fastest processing than other representations of the XML Information Set [8]. Based on that and a set of other evaluation criteria such us interoperability, compatibility, efficiency and compact implementation among others, the W3C standardization body released the EXI specification as a recommendation on 10th of March 2011. While there are some general discussions on applying EXI (see EXI Best Practices [9] and EXI Impacts [10] for overview), an analysis on the direct impact, applicability and possible benefits of using EXI in factory automation systems has not been performed yet. A study by Moritz et al. [11] on compression techniques for DPWS [12] is related to the work we present here as it considers the use of EXI for representing the messages defined in DPWS specification. However, its focus is rather narrow as it only investigates the applicability of EXI for DPWS from the perspective of compactness of the messages. The investigation we outline here is broader including processing efficiency along with compactness. In addition, we consider more aspects of the EXI application in the automation middleware systems - not only DPWS but also OPC UA [13] which is the latest most widely adopted industrial specification for factory automation. Integration of legacy systems is also covered and is essentially extension to the SOA for devices system presented in [14] while taking the

EXI into account.

An overview of the web service technology and a state-of-the-art survey on its application for embedded devices is presented by Shelby in [15]. Therefore, our focus is on presenting the web service technology exclusively in connection to its usage in DPWS and OPC UA.

II. Background

This section provides short overview of the major concepts and technologies which are central for our discussions presented in the subsequent parts of the paper.

A. OPC UA

OPC Unified Architecture is an evolutionary upgrade to the classic OPC (Object Linking and Embedding (OLE) for Process Control) suite for data exchange in industrial automation systems. It provides a complete protocol stack for plant floor, operation and control networks by specifying the communication infrastructure including security mechanisms, transport protocols and exchange patterns as well as data representation and information models as presented in [16]. The first OPC specifications (classic OPC) are based on the proprietary Component Object Model (COM) and Distributed COM (DCOM) technologies which are now deprecated in favor of the Microsoft .NET Framework. Although the classical OPC was successful and widely used, its major drawback was the platform dependency stemming from the use of COM/DCOM. In order to address this issue and enhance the classic OPC with SOA support, the OPC foundation released the XML Data Access (XML-DA) specification which is entirely based on web service technology. It uses SOAP for communication, Web Service Description Language (WSDL) for defining the service messages and XML Schema for specifying the data type hierarchy in the OPC information models. However, the performance of this SOA-based solution is not adequate for use in time-critical systems mainly due to the XML processing overhead. As a result, the OPC UA framework, released after XML-DA, provides an additional transport protocol and encoding for the predefined service messages described in [17]. A proprietary transport protocol called UA TCP is included and it operates directly on top of TCP. UA TCP is designed to substitute the resource expensive HTTP/SOAP in an environment with real-time requirements and hardware constrains. In such an environment, the text-based XML representation of the service payloads does not provide the required performance for parsing, serialization and compactness as well. Therefore, OPC UA defines additional encoding scheme called UA Binary that is proprietary binary representation of the OPC data models. Figure 1 depicts the relation of these encoding schemes (XML and UA Binary) and the two transport schemes (HTTP/SOAP and UA TCP) with respect to their intended usage throughout the industrial enterprise. As shown in the figure, the service calls on the plant floor are executed over UA TCP with UA Binary as encoding scheme. The target platforms on this level are controllers, wired/wireless HMIs and process visualization and reporting consoles (dashboards). For the interactions on the operations layer, the standard HTTP/SOAP transport is better suited for interconnecting the variety of higher level applications such as Manufacturing Execution Systems (MES) and monitoring and control panels. The timing in this layer is important in many cases - delivering of alarms and other critical events for example, so fast and efficient encoding of the messages is preferable. On the other hand, the fully interoperable web service representation based on HTTP/SOAP with XML is the best choice when connecting business systems such as Enterprise Resource Planning (ERP), Enterprise Asset Management (EAM), Accounting etc.

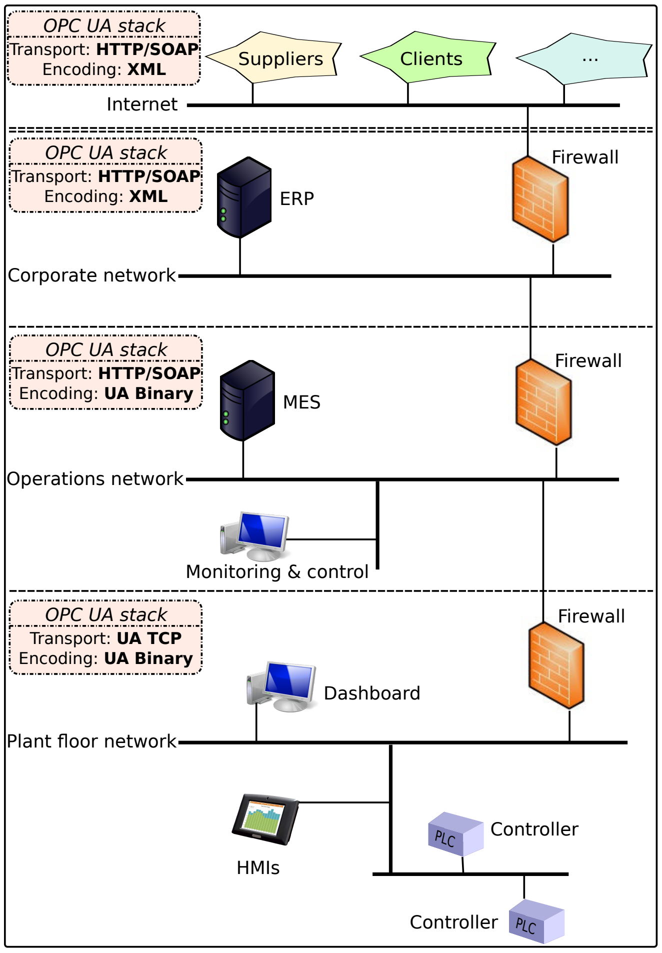

Fig. 1. OPC UA transport protocols and encodings used in different layers in an industrial enterprise: plant floor, operational, corporate and inter-enterprise

B. DPWS

As opposed to OPC UA, DPWS is an open and freely available OASIS specification. Moreover, there are several open source tools available (see [18], [19] for example) for developing DPWS applications in C/C++ and Java. The core building block in the DPWS standard is the SOAP-based web service technology where a subset of the WS-* family of protocols are employed. The usage of these protocols is further constrained or augmented in order to fit into the requirements posed by the target domain for DPWS - smart networked embedded devices. This formally specified set of constrains and usage restrictions is known as web service profile. As shown in Figure 2, the profile relies on WSDL 1.1, SOAP 1.2, SOAP-over-UDP and several WS-* protocols.

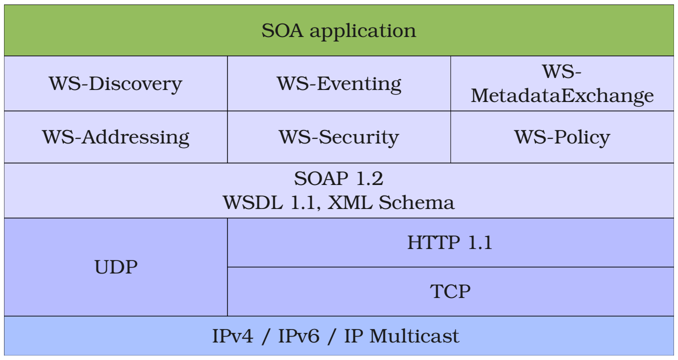

Fig. 2. DPWS protocol stack

The problems that DPWS is designed to address are connected to enabling vendor independent and secure SOA-based communication with plug-and-play capabilities, automatic configuration, remote management and deployment. DPWS, however, is not specially targeting automation system applications and thus cannot provide a rich domain specific information model as the one available in OPC UA [20]. The predefined services in DPWS for example are limited to describing the properties and functionality of a generic device i.e. device model, name and manufacturer properties and generic services such as service discovery, listing of hosted services and resources. Nevertheless, the application of DPWS in industrial environment has been studied by researchers and it is shown to be feasible and beneficial for certain use-cases with low real-time requirements. Several demonstrators and case studies were implemented within the SIRENA project [21] for example. The lack of automation-specific information model in DPWS can be overcome by using external standardized domain data models. For example, Business to Manufacturing Markup Language (B2MML) can be used to link business systems such as ERP and supply chain management systems with manufacturing systems such as control systems and MES. The Senor Web Enablement framework [22] of the Open Geospatial Consortium (OGC) is another data model applicable in industrial environment for unification of the sensor data representation and semantics.

C. *Efficient XML Interchange*

EXI is a binary representation of the XML Information Set that is designed for compactness and high performance parsing and serialization. The theoretical foundation of EXI format is derived from information and formal language theories such that each XML Info Set document can be constrained by a set of formal grammars in Greibach normal form [23]. Moreover, each terminal symbol in these grammars i.e. a XML element, attribute, element content, closing element etc., is represented with variable length codes such that the most likely to occur symbols are encoded in fewer bits. Each EXI document consists of a header and a body. The header defines different encoding/decoding parameters that affect the compactness, processing efficiency and memory usage when processing the document. For example, the document size can be further reduced by using the EXI compression option indicating the application of DEFLATE algorithm for data compression [24]. Another important parameter is the use of a XML schema that sets constraints on the structure and content of the document. The schema information must be available before the EXI encoding/decoding, and it should be either statically set or communicated in advance using schema languages such as Document Type Definition (DTD), W3C XML Schema or RELAX NG. Schema-enabled EXI processing is much faster and results in smaller EXI documents compared to schema-less processing. For more detailed discussions on the EXI format and a primer see [25].

The main advantages of the EXI format can be summarized as follows:

- Highly compact representation comparable to a hand-optimized application specific encodings.
- Capability to preserve all features found in a XML document - even comments and processing instructions. This property guarantees that proven technologies, such as web services, XHTML, SVG and many others that are based on XML, are amendable to EXI representation.
- Efficient processing algorithms available which are also generic enough to work on documents with or without schema information or in a mixed mode where only part of the document is structured according to the schema. In addition, these algorithms can be coded in a small footprint implementation.

III. Analysis Results

This section presents the results of our analysis on the application of Efficient XML Interchange format as an encoding technique for OPC UA and DPWS. The evaluation of the opportunities, benefits and challenges is covering three different aspects: compactness of the service messages, processing efficiency and legacy systems integration. We based our study on the OPC UA performance measurements presented in [26], the W3C EXI evaluation studies [6], [27] as well as the data for size reduction of DPWS messages with EXI encoding [11].

A. *Compactness*

In this work, the compactness of the EXI encoding is given as a percentage of the EXI document size compared to the text-based XML representation. The compactness is highly dependent on the document characteristics (size and content density) as well as EXI encoding options (usage of compression, schema information etc.). As shown in [6], the EXI compactness span from 1% for the large documents with compression and schema encoding enabled till up to 95% for schema-less encoding of very small documents. Therefore, in order to assess the compactness of the EXI format for

OPC UA and DPWS, it is important to take into account the characteristics of the service messages and the targeted EXI options. In OPC UA and DPWS the service messages are relatively small, ranging from 225 bytes to more than 10 kB in XML representation. Schema information is available, yet schema-less encoding can be used for the service payload in OPC UA so both schema and schema-less metrics are important. As indicated in [11], the DEFLATE compression has little effect on the small messages used in DPWS and it also impacts the processing speed. For that reason, the use of DEFLATE compression option in EXI is not considered in our analysis.

The average compactness for the EXI encoded DPWS messages given in [11] is 20.6% for the schema encoding and 59.6% in schema-less mode. We performed similar measurements for the OPC UA services [17]. The description and formating of the service messages are taken from the OPC UA service descriptions[1] freely available for non-members. The average compactness achieved for auto-generated service messages including the SOAP envelope is given in Table I.

TABLE I
AVERAGE COMPACTNESS OF THE OPC UA SERVICE MESSAGES

Encoding type	Average compactness in %	Average size [bytes]
XML (text)	100.0	2547
EXI (schema-less)	32.5	523
EXI (schema)	3.2	80

As shown in Table I, the compactness achieved for OPC UA is much better than the one for DPWS mainly due to the highly structured messages used in OPC UA. It can be concluded that Efficient XML Interchange encoding for DPWS and OPC UA provides huge improvements in messages size compared to text-based XML.

B. Processing efficiency

Based on the measurements presented in [26], the use of HTTP/SOAP transport in OPC UA decreases the service execution speed by 50% on average compared to UA TCP. However, the negative impact on the performance is much higher if a text-based XML encoding is enabled instead of UA Binary. The use of XML accounts for 50% increase in latency for small single variable messages and up to 1800% for large messages with 1000 variables. According to the authors, the test setup for this measurements consisted of two PCs running .NET UA Stacks with a 100 MBit network link in between. The performance degradation is caused by the XML processing as well as the network latency as a function of the throughput and the size of the messages. If the use of EXI for OPC UA service calls provides a performance gain in the same rank as the one observed from the use of UA Binary, then enabling EXI encoding in OPC UA can serve as a single data representation providing flexibility, efficiency and interoperability at the same time. The use of a single data format will lower the complexity and the size of the OPC UA implementations as there is no need to support multiple parsers and serializers for XML and UA Binary. On the other hand, the smaller program footprint of the OPC UA stack could enable the use of SOA-based communications on highly resource constrained controllers and sensor nodes - a concept further explored in the next section.

Measurements comparable to the above ones can be found in [6], where the EXI processing efficiency for a similar setup is given. The tests were made on a PC and a server machine connected over 100 MBit network link using Java EXI parsers and serializers. In both encoding and decoding cases, the performance gain compared to plain XML is in similar magnitude as the one seen for the UA Binary encoding above: the approximate average value in the 95% confidence interval being between [250% - 350%] for all test cases with schema-less processing and between [430% - 520%] in the schema-enabled mode. It is also expected that the performance gain for DPWS will be close to the one for OPC UA due to the similarities in size and structure of the messages.

C. Legacy systems integration

The introduction of EXI encoding for DPWS and OPC UA will provide substantial improvements in compactness and processing efficiency and hence lowering the hardware requirements for the legacy systems integration approach presented in [14]. In this way, the proposed SOA gateways that expose the legacy systems functionality as services can be less powerful and put closer to the systems they wrap. Moreover, the real-time properties of the network will be easier to satisfy as the compact EXI messages are less traffic intensive.

However, the new EXI based SOA protocols for factory automation must ensure the backward compatibility not only with legacy systems using proprietary and vendor dependent protocols but also XML capable devices. We identified the following methods for EXI integration of SOA ready XML-based enterprise and middleware systems running DPWS or OPC UA:

1) Deploy software updates or plugins providing support for EXI processing - characterized by optimal performance, but raising safety issues and relatively long non-working maintenance window required
2) Introducing EXI network proxies that translate EXI to XML and vice versa - transparent and non-intrusive method requiring additional networked hardware components
3) Modification in the application protocols used such that EXI is included as an option in the encoding negotiations

IV. EXI ON WIRELESS SENSOR AND ACTUATOR NODES

The use of flexible and standard-based service messages exchanged throughout all levels of the industrial enterprise is certainly providing many benefits as already discussed in the previous section. In such an environment, the seamless and cost-effective integration of small ubiquitous sensor and actuator devices spread across the shop floor is further providing new opportunities for process monitoring and control by

[1]Services.wsdl - http://opcfoundation.org/UA/2008/02/Services.wsdl

supplying timely and accurate measurements for the executed workflows. A promising and already established standard for low-cost, low-power and limited bandwidth wireless communications is IEEE 802.15.4 [28]. By means of this standard, battery powered embedded devices called wireless sensor and actuator nodes can be easily deployed to provide close to the source measurements and events information. This information can be filtered, aggregated and enriched on different levels: on the sensor nodes as well as on intermediate and higher level systems. The network layer integration of such devices is largely addressed by the 6LoWPAN specification [29] for IPv6 compatible communications over IEEE 802.15.4 based networks. However, the application layer protocols are mostly proprietary and specially optimized for use over 6LoWPAN as the frame size of IEEE 802.15.4 is only 127 bytes and fragmentation is required when the application layer payload is bigger than between 76 and 116 bytes (depending on the addressing and security options). The approach presented here is based on the use of the same application protocols already used on the plant floor. The motivation for this approach is the simpler and cost-effective configuration and deployment as the need for middleware gateways is diminished. The use of efficient encoding and processing of the application data is also required and we suggest the usage of standard EXI encoding executed on the sensor nodes. In order to verify the proposed extension of the SOA communications down to the wireless sensor nodes, we performed a number of performance tests. For our experimental setup, we assume an OPC UA environment similar to the one depicted in Figure 3.

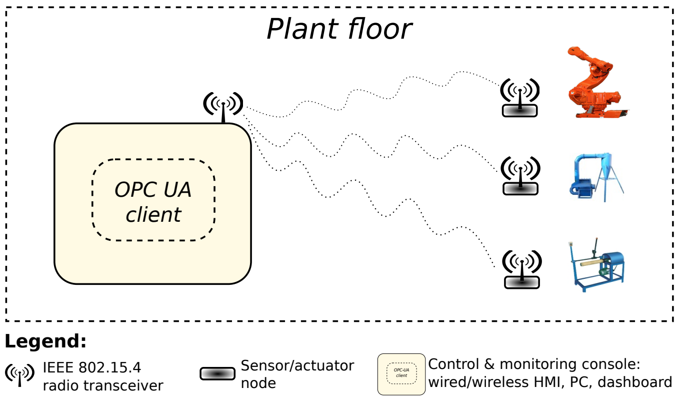
Plant floor
OPC UA client
Legend:
IEEE 802.15.4 radio transceiver
Sensor/actuator node
Control & monitoring console: wired/wireless HMI, PC, dashboard

Fig. 3. Example OPC UA setup: integration of wireless sensor nodes through the use of UA TCP as a transport with EXI encoded payload

In our simplified testing configuration we used a PC and two Mulle v6.2 sensor platforms [30]. The Mulle nodes were equipped with a Renesas M16C/65 microcontroller running at 10 MHz with 47 kB RAM and 512 kB programming memory using 2.4 GHz IEEE 802.15.4 radio transceiver for wireless communications. The software running on the nodes included TinyOS environment and Berkeley Low-power IP stack (BLIP) as a 6LoWPAN network layer implementation.

According to the OPC UA specification, a session establishment is needed before the data exchange between an OPC UA client and an OPC UA server is possible. The session establishment is accomplished through a handshake service messages which are not considered in our setup in order to reduce the implementation efforts required. That way, we assume the presence of already available and active session between one of the sensor nodes and the PC. The studied exchange pattern is a reading of a value from a OPC UA server instance - in our case it is one of the sensor nodes, by a OPC UA client instance i.e. the PC. The second node is only used to deliver the IEEE 802.15.4 packets to and from the PC i.e. it provides access to the IEEE 802.15.4 wireless medium. This is done by installing the *IP BaseStation* TinyOS utility application to that node such that it transmits the received packets from the serial port over the radio and sends the received packets over the radio to the serial port connected to the PC. The service messages are standard *Read* service request and response for a single variable value. Table II shows the size of the request and response messages with different encodings applied. In our experimental setup, we used the EXIP open source library[2] to parse and serialize the EXI encoded OPC UA *Read* service messages. We used the EXIP low-level API [31] to map the message parameters encoded in an EXI binary stream to a C structure. As the current version of the EXIP library does not provide support for schema-enabled EXI processing, the messages were encoded without schema information using the default EXI header options.

TABLE II
SIZE OF THE OPC UA *Read* SERVICE REQUEST AND RESPONSE MESSAGES FOR PLAIN XML, EXI SCHEMA-LESS AND EXI SCHEMA-ENABLED

	Message size [bytes]	
Encoding type	Request	Response
XML (text)	912	712
EXI (schema-less)	362	251
EXI (schema)	56	31

In our experiment, the PC initiated a read request over TCP consisting of UA TCP protocol header (28 bytes) followed by the EXI encoded message (362 bytes). The receiving sensor node parsed the request, then serialized the response with the value of the requested variable and sent it back to the PC. The response again included a 28 bytes UA TCP header and an EXI payload of 251 bytes. We measured the time for the whole service call that includes: TCP connection establishment, serialization of the service request by the PC, transmission of the request, parsing of the request by the sensor node, serialization of the response by the sensor node, transmission of the response back to the PC and parsing of the response by the PC. The average time for complete service execution in this setup was measured 1.2 seconds. While the PC parsing and serialization time for the EXI encoded messages can be neglected (approximately 0.5 ms) these operations for the sensor node are quite time consuming as they accounted for around 534 ms. However, these measurements can be seen

[2]Efficient XML Interchange Processor - http://exip.sourceforge.net/

as an upper limits as the EXIP library does not employ the most optimal processing algorithms. We envision significant improvements in the EXI message processing when more efficient representation of the EXI grammars is employed. Furthermore, the utilization of the available schema information for the service messages will substantially improve not only the processing, but also the transmission time as the size of the messages is several times smaller (see Table II). The use of more compact messages reduces the number of 6LoWPAN packages required and hence the transmission time.

Along with the improvements in service execution time, memory consumption must also be addressed. The EXIP library and the simplified service engine used in our scenario occupy almost 57 kB of programming memory. The Random Access Memory (RAM) consumed during EXI parsing and serialization is around 8 kB. Both programming memory and RAM footprints are expected to decrease when schema information is employed in the EXI processing.

V. Conclusion and future work

Based on the presented analysis, the application of Efficient XML Interchange in factory automation systems could bring significant improvements in resource utilization (network throughput, computational power and memory footprint) when compared to plain XML encoding used in DPWS and OPC UA. Similar improvements could be achieved with other binary encodings (UA Binary, Fast Infoset, BSON, esXML etc.) so more in-depth studies are needed to determine the benefits of EXI while taking into account non-functional properties such as interoperability and backward compatibility.

The concept of extending the SOA approach to embedded devices through the use of EXI was also discussed and proof of concept measurements were presented. The first results in this direction showed the feasibility of this approach while pointing out the need for further investigation of efficient processing algorithms for EXI. Finally, considering the available room for improvements, we can conclude that the Efficient XML Interchange encoding is a viable solution for extending the applicability of the SOA approach across small wireless sensor and actuator devices.

Acknowledgment

The authors would like to thank the European Commission and the partners of the EU FP7 project IMC-AESOP (www.imc-aesop.eu) for their support.

References

[1] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62 – 70, 2005.

[2] *Simple Object Access Protocol (SOAP) 1.1*, W3C Std. [Online]. Available: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[3] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, 2002.

[4] C. Groba and S. Clarke, "Web services on embedded systems - a performance study," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, march 2010, pp. 726 –731.

[5] S. Sakr, "XML compression techniques: A survey and comparison," *J. Comput. Syst. Sci.*, vol. 75, pp. 303–322, August 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1530897.1531090

[6] G. White, J. Kangasharju, D. Brutzman, and S. Williams, "Efficient XML Interchange Measurements Note," W3C, Tech. Rep., 2007. [Online]. Available: http://www.w3.org/TR/exi-measurements/

[7] *Efficient XML Interchange (EXI) Format 1.0*, W3C Std. [Online]. Available: http://www.w3.org/TR/2011/REC-exi-20110310/

[8] J. Cowan and R. Tobin. XML Information Set (Second Edition). W3C. [Online]. Available: http://www.w3.org/TR/xml-infoset/

[9] M. Cokus and D. Vogelheim, "Efficient XML Interchange (EXI) Best Practices," W3C, Tech. Rep., 2007. [Online]. Available: http://www.w3.org/TR/2007/WD-exi-best-practices-20071219/

[10] J. Kangasharju, "Efficient XML Interchange (EXI) Impacts," W3C, Tech. Rep., 2008. [Online]. Available: http://www.w3.org/TR/2008/WD-exi-impacts-20080903/

[11] G. Moritz, D. Timmermann, R. Stoll, and F. Golatowski, "Encoding and Compression for the Devices Profile for Web Services," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, 2010, pp. 514 –519.

[12] *Devices Profile for Web Services Version 1.1*, OASIS Std. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf

[13] *OPC Unified Architecture*, OPC Foundation Std., 2009. [Online]. Available: www.opcfoundation.org/ua/

[14] S. Feldhorst, S. Libert, M. ten Hompel, and H. Krumm, "Integration of a Legacy Automation System into a SOA for Devices," in *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, 2009.

[15] Z. Shelby, "Embedded web services," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 52 –57, 2010.

[16] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Springer, 2009, ch. Introduction, pp. 1–16.

[17] ——, *OPC Unified Architecture*. Springer, 2009, ch. Services, pp. 125–189.

[18] (2011, April) Service-oriented architecture for devices. [Online]. Available: http://forge.soa4d.org/

[19] (2011, April) Web services for devices (ws4d). University of Rostock, University of Dortmund and MATERNA. [Online]. Available: http://ws4d.e-technik.uni-rostock.de/

[20] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Springer, 2009, ch. Standard Information Models, pp. 107–122.

[21] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains," in *International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006*, 2006.

[22] (2011, April) Sensor web enablement. Open Geospatial Consortium. [Online]. Available: http://www.opengeospatial.org/projects/groups/sensorweb

[23] S. A. Greibach, "A new normal-form theorem for context-free phrase structure grammars," *J. ACM*, vol. 12, pp. 42–52, January 1965. [Online]. Available: http://doi.acm.org/10.1145/321250.321254

[24] P. Deutsch, *DEFLATE Compressed Data Format Specification version 1.3*, Internet Engineering Task Force Std., May 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1951.txt

[25] D. Peintner and S. Pericas-Geertsen, "Efficient XML Interchange (EXI) Primer," W3C, Tech. Rep., 2009. [Online]. Available: http://www.w3.org/TR/2009/WD-exi-primer-20091208/

[26] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Springer, 2009, ch. Performance, pp. 305–309.

[27] C. Bournez, "Efficient XML Interchange Evaluation," W3C, Tech. Rep., April 2009. [Online]. Available: http://www.w3.org/TR/exi-evaluation/

[28] *IEEE 802.15.4-2006*, IEEE Computer Society Std., 2006. [Online]. Available: http://standards.ieee.org/findstds/standard/802.15.4-2006.html

[29] *IPv6 over Low power WPAN (6LoWPAN)*, IETF Std., August 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4919.txt

[30] J. Johansson, M. Völker, J. Eliasson, Å. Östmark, P. Lindgren, and J. Delsing, "Mulle: A minimal sensor networking device - implementation and manufacturing challenges," in *Proceedings IMAPS Nordic*, 2004, pp. 265–271.

[31] R. Kyusakov, J. Eliasson, and J. Delsing, "Efficient structured data processing for web service enabled shop floor devices," in *20th IEEE International Symposium on Industrial Electronics*, 2011.