Towards Industrial Internet of Things: An Efficient and Interoperable Communication Framework

Jens Eliasson, Jerker Delsing, Hasan Derhamy
Dept. of Computer Science, Space and Electrical Engineering
Luleå University of Technology
Luleå, Sweden
jens.eliasson@ltu.se

Zoran Salcic, Kevin Wang
Dept. of Electrical and Computer Engineering
The University of Auckland
Auckland New Zealand
z.salcic@auckland.ac.nz

***Abstract*— Interoperability between shop floor devices and upper layer systems is a key challenge for enabling Internet of Things in industrial applications. Standardized protocols such as IPv6, CoAP, and XML can be used to address this issue. Widely used XML-based technologies such as SenML, EEML, OPC-UA as well as others rely on XML to be able to support a wide range of sensor and actuator applications. However, this approach results in high communication overhead due to the verbose nature of plain text messages encoded in XML. When devices are communicating using 6LoWPAN over IEEE 802.15.4, it is important to keep the messages small enough to fit into one MAC-layer frame to avoid fragmentation and hence conserving bandwidth and transmission energy. One possible solution is to integrate differential binary delta-encoding with a service-based framework based on CoAP, SenML and EXI. The proposed efficient communication approach for service-based architecture can compress a series of events up to 90-95%. The proposed framework is a holistic approach for enabling distributed monitoring and control applications and a move towards realizing the vision of *Services of Things*.**

Keywords—Data compression; Internet of Things, Cyber-physical systems; SOA; Smart Objects

I. INTRODUCTION

Small embedded systems equipped with a combination of sensors and/or actuators, as well as with networking capabilities, are the core building blocks of an Internet of Things (IoT) [12]. These nodes typically create IP-based low-power wireless networks as presented by Dunkels et al. [13], which, combined with gateways, middleware and cloud computing, perform distributed data acquisition, aggregation and analysis. Individual nodes gather data, perform local processing, and transmit their results to other nodes in the network using IPv6 on top of wireless technologies such as Bluetooth, 6LoWPAN or Wi-Fi or wired solutions such as Ethernet and Power-Line Communication (PLC). The nodes collaborate in a distributed fashion to implement various kinds of monitoring and control tasks. Since a network may consist of a large number of nodes, individual node breakdown does not cause total network failure. Gateways are used to enable low-power nodes to communicate with external networks, such as cellular networks and the Internet.

Organizations such as IEEE, IETF, IPSO, ZigBee Alliance, etc. are currently developing standards for communication in the field of Internet of Things. Many protocols, for example 6LoWPAN over IEEE 802.15.4, Bluetooth Low Energy, PLC, and RPL are starting to become de-facto standards for low-power and low-cost devices [5]. The use of SOA (Service-Oriented Architecture)-enabled protocols like CoAP [2], MQTT [21] and OPC-UA [9] enables machine-to-machine (M2M) communication as well as interactions between end-users and sensor and actuator platforms. Even though CoAP addresses issues such as low-power access to resource-constrained devices, and powerful scripting frameworks for service composition targeted CoAP (see for example Kovatsch et al. in [1]).

The use of common and standardized data models and semantics, e.g. XML-based technologies such as SenML, EEML, SensorML, OPC-UA, DPWS and others, results in a very high overhead in terms of communication and processing. The expressiveness of XML enables meaningful data exchange in M2M communications. Many existing systems for data collection, storage and visualization, such as Xively [19], rely on either XML or JSON for data exchange. However, on constrained networks meaningful information must be efficient and simple. Data compression for SOA-enabled systems, which is one enabling technology for addressing this issue, has also been investigated by other researchers, e.g. Kyusakov et al. [4], Caputo et al. [18], and others.

This paper presents another approach, using an efficient communication method designed specifically for event-based communication and SOA over the CoAP protocol. The long-term aim of the method is to enable efficient distributed service composition and orchestration [14] on resource-constrained embedded systems such as wireless sensor and actuator platforms.

II. Related WORK

There are many frameworks attempting to enable IoT applications and realize the business potential of connected devices and applications. Frameworks such as Xively, Cumulocity and ThingWorx offer data centric or application

978-1-4799-7800-7/15/$31.00 ©2015 IEEE

centric platforms ideal for aggregating data and integrating applications. AllJoyn, the software framework from AllSeen Alliance is a distributed software bus which provides device and service discovery, security and ad-hoc networking [31]. It is device centric which enables machine to machine communication, required for industrial monitoring and control systems. However it has yet to be proven on constrained networks and devices while providing adequate Quality of Service (QoS). An eXtendable Component-based Interoperable Open Model-driven Architecture (AXCIOMA) uses Object Management Group based standards to create an architecture which allows for complete system modeling and automated deployment through Interface Definition Language (IDL) and standardized language mappings. A shared dependency of these frameworks and platforms is the usage of XML and/or JSON for communication.

Below are also two state of the art approaches for the design of distributed monitoring and control systems that are applicable to the targets of this paper. They are (1) multi-agent systems [20, 21, 22] and (2) approach using a formal language, SystemJ, for distributed systems [23]. Both approaches require skilled system designers, where SystemJ excels due to outstanding expressiveness of constructs available for designing and programming concurrent software behaviors.

The concepts of agents and multi-agent systems (MAS) gained popularity over the last decades in implementing distributed control systems. Each agent is a uniquely identifiable software entity that can perform certain services and multiple agents collectively achieve the overall system operations. Agents are loosely coupled to each other and thus failure of an individual agent has limited effects on the overall system functionality. The similarity between MAS and SOA makes MAS a popular way in implementing customized SOA systems, in addition to the common web service based SOA systems. MAS have been deployed in many distributed applications such as intelligent environment [24], remote monitoring and control applications [25] and collaborative robotics [26]. Despite the support of dynamic instantiation/removal of agents, mobility, and resilience to faults, there are a number of challenges in design distributed control systems using MAS. First, MAS typically requires heavy runtime support for proper agent life cycle and execution. Therefore, MAS may not always be applicable with resource constrained devices. Second, majority of the existing MAS do not follow any formal models of computation and hence the overall system behavior and the individual service behavior cannot be verified and guaranteed. Tremendous efforts are spent to examine the system functionality and reliability when interactions among distributed components are involved.

III. THE SYSTEMJ APPROACH

Alternatively, a formal language (i.e. SystemJ) based approach follows the formal Globally Asynchronous Locally Synchronous (GALS) Model of Computation (MoC) is proposed [23]. This approach targets a network of nodes that are Java-enabled. It is based on the use of SystemJ programming language that enables designing complex distributed systems with many concurrent software behaviors that, when composed into the SystemJ program, create a GALS system. A system is composed of multiple asynchronous behaviors called clock domains which are fixed during system runtime and hence such system is considered to be a static system. Clock domains are interconnecting and interacting with each other through a powerful abstract object called channel and collectively define the system topology and overall behavior. Clock domains communicate with physical environment through another abstract object called signals. Channels and signals can be any Java primitive type or Java object. Furthermore, clock domain behavior can be further divided into synchronous behaviors called reactions which execute in locked step to guarantee deterministic execution. They fully comply with the formal synchronous/reactive model of computation and even allow formal verification. Clock domains and reactions are inspired with hardware processes in sequential digital circuits. An example of graphical representation of a SystemJ program is given in Figure 1. This example illustrates a model of a small sensing and control system. Two clock-domains, Sensor and Controller, are executed asynchronously to each other. Sensor clock-domain gathers information from the external environment, processes the information, and transfers results to the Controller clock-domain. The Controller clock-domain then generates appropriate control signals, which are emitted back to the external environment. The synchronous parallel operator (||) is used to build clock-domains from multiple synchronous reactions. Reactions can be composed hierarchically, by forking child reactions from the parent reactions, effectively implementing behavioral hierarchy.

This example demonstrates the possibility of considering clock domains as components that communicate with other clock domains using channels and with their external environment through signals with corresponding input and output ports. In terms of implementations on wireless networks, SystemJ programs can execute on commercial grade nodes that have either 6LoWPAN and/or IEEE 802.15.4 compliant protocol stack. Clock domains can be considered as service entities with clear role of input ports for requesting services and output ports to services provision according to their functionalities.

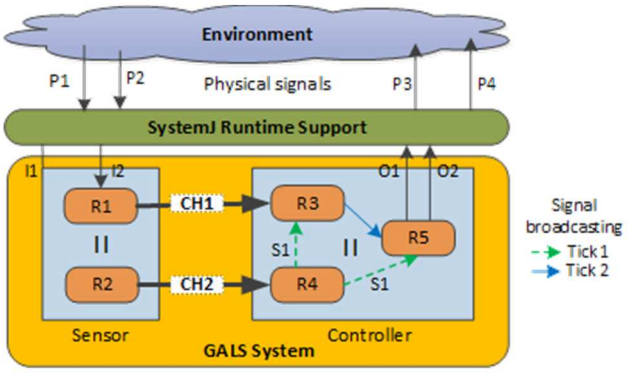

Figure 1: Graphical illustration of a SystemJ program

Both approaches rely on XML and/or JSON based message format which introduce overheads in communication, especially with low bandwidth and low rate wireless sensor networks. By enabling resource-constrained sensor and actuator platforms to communicate using standardized protocols and technologies

such as XML and the Java-based CoAP Californium toolkit by Kovatsch et al. [10], seamless integration with collaborative systems such as SystemJ is made feasible. The technique proposed in this paper can support existing tools to achieve efficient communication with analyzable system implementation.

IV. Efficient Standards-based communication

When using verbose message formats such as JSON and XML it is important to support data compression in order to be efficient. When using a standard IEEE 802.15.4 compliant radio transceiver, the maximum payload is 127 bytes, including 6LoWPAN, IPv6 and UDP headers. When using compressed IPv6 over 6LoWPAN, the effective payload can be as little as 70-100 bytes depending on compression, security and address options. Transmitting a message of almost 400 bytes would require four to five frames. This would of course increase the power consumption, latency and put more load on the network thereby limiting scalability.

Therefore, a vital property for maximizing performance is that each application layer message can be sent over the network in one MAC layer frame. The use of XML-based message format enables existing tools to be used and provides a future-proof way of exchanging information that is web-compliant.

A. *Differential Text-XML compression mode*

One feature of the proposed framework is the use of differential XML. Differential XML is a compression method where only the fields that have changed are transmitted, thus mitigating the need to send the full XML+SenML header with each subsequent data packet. When a client enables the differential XML mode, it will receive a complete SenML packet in the first GET request, and following packets are sent as plain text differential packets. An example of a differential message exchange (218 bytes) is shown below (non-constant fields marked in bold for clarity).

```
<?xml version="1.0" encoding="UTF-8"?>
<senml xmlns="urn:ietf:params:xml:ns:senml"
  bn="urn:dev:mac:0024befffe804ff1/"
  bt="$ts" ver="1" bu="A">
 <e n="temp" u="Cel" v="$t" />
 <e n="humidity" t="%rh" v="$h" />
</senml>
```

Here it is obvious that the timestamp, temperature and humidity values can change between messages. In all events coming after the first subscription response, all variables will be sent as plain text, consuming only 23 bytes, in the compressed form:

```
$ts=1276020076
$rh=22.5
$h=55
```

The client must when receiving a differential response re-create the XML description, and run the output through an XML schema validator in order to validate the output. This approach results in a compression ratio of approx. 90% compared to the original uncompressed plain-text SenML message which is 229 bytes. However, the use of differential plain-text messages only offers limited functionality, especially when more complex and structured messages must be used.

```
<?xml version="1.0" encoding="UTF-8"?>
<senml xmlns="urn:ietf:params:xml:ns:senml"
  bn="urn:dev:mac:0024befffe804ff1/"
  bt="1276020076" ver="1" bu="A">
 <e n="temp" u="Cel" v="22.5" />
 <e n="humidity" t="%RH" v="55" />
</senml>
```

B. *Differential binary-XML compression mode*

Since XML is very verbose it is not suitable for use in wireless Internet of Things networks over low-bandwidth links. One technology that can be used to mitigate the performance impact of plain-text messages while retaining the benefits of using XML is Efficient XML Interchange (EXI) as shown by Kyusakov et al. [6]. EXI is a pure binary technology that enables both compaction and compression of plain text XML messages into a binary form. The use of EXI can compress a plain text XML message between 30-90% in most cases. The use of differential operation is actually mentioned in the SenML specification but only for local usage. A very resource–constrained device can use a pre-compiled EXI message and only change certain byte with the EXI byte stream thereby eliminating the need of running a full-fledged XML to EXI processor. However, this approach assumes that the EXI message cannot change size or even the position of the bytes to update. This of course limits the usability. In the example above, the timestamp field which is 10 bytes long must always be sent in whole even if only 1 bit has changed. This issue can be eliminated by using a differential binary approach. When representing the value 1276020076 in binary representation only four bytes are required. If for example only the fourth byte would change (in the case of a new time stamp of value 1276020077) it would be beneficial to only transmit the one byte that is actually changed.

In the proposed framework, this is achieved by first compressing the plain text XML file into its binary EXI representation, and then calculating the difference compared to the previous EXI message. The binary delta is then transmitted to the receiver which takes the previous EXI message and re-computes the new message using the delta information. By using this approach, an event can have its size reduced from potentially hundreds of bytes to tens of bytes. There are of course cases when the differential information is actually larger than the EXI message itself, in that case the pure EXI message is transmitted. This guarantees that the delta compression never causes larger messages than the original EXI version.

C. *Lightweight Interoperable Message Exchange*

With the use of CoAP, it would be very beneficial to merge binary delta encoding, differential transmission and event-based communication. When a client requests to *Observe* [16] a CoAP resource, it can treat that initial response as the base

and all following messages as differential data. This is how the proposed message exchange framework operates. The client uses a CoAP query option with an Observe request to indicate to the remote CoAP resource that it would like to communicate using differential binary delta encoding. The server responds with an EXI-compressed SenML message to the GET request with the Observe option. All following events are first compressed to an EXI stream which is passed through a differential binary encoder. The encoder returns a very small data set of bytes which represents the delta between the initial message and the current. The delta information is sent to the client with an Option set to indicate that the payload is differential. When the client receives the message, it updates the base EXI stream's bytes according to the new information, and then decodes the updated EXI stream into an XML message and passes that to the application layer. Figure 2 shows the currently supported network stack.

The proposed framework supports these two proposed compaction methods by extending CoAP-based services with URI-query options that a client use to inform the remote service which output semantics (XML or JSON), and which compaction method (none, EXI, differential text, or binary delta encoding).

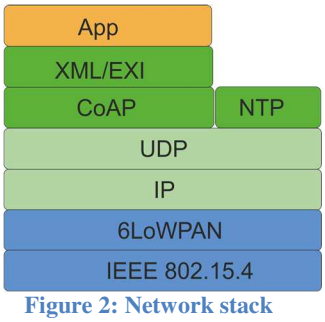

Figure 2: Network stack

One interesting approach to extend the applicability of the proposed features would be to add support for these compaction schemas directly in the CoAP protocol. This is however considered as future work. Note that differential mode can only be used in event-based communication and cannot be used when using ordinary GET requests.

V. TEST APPLICATION - WHEEL LOADER MONITORING

In order to test the proposed method in a real world monitoring application, subtask 1.8 in the Arrowhead project was chosen. Arrowhead is a European R&D project with the aim to develop SOA-based interoperable systems [7]. Arrowhead's Task 1.8 is a research and development activity aimed at delivering hardware and software for ball-bearing monitoring of a wheel loader. Task 1.8 is a joint collaborative effort conducted by Luleå University of Technology, SKF and Eistec AB [11]. The aim of Task 1.8 in Arrowhead is to be able to monitor each wheel of a wheel loader (or other vehicles as well), and to transmit an alarm if for example a ball bearing has been damaged. See Figure 3 for a layout of the network architecture of Arrowhead Task 1.8.

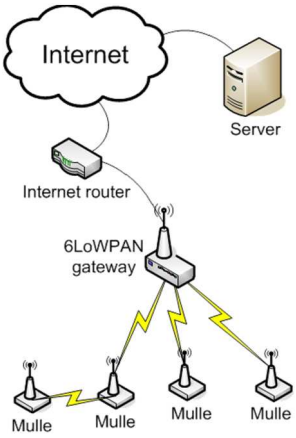

Figure 3: Wheel loader monitoring architecture

In this task, Eistec's wireless sensor and actuator platform Mulle mk4 is used. The Mulle features a low-power Freescale ARM Cortex-M4 microcontroller, wireless communication using IEEE 802.15.4 and several sensors. In the current setup, the Mulle runs the Contiki operating system and communicates using CoAP over 6LoWPAN. To support hierarchical messages as well as structured data, the SenML message format was selected.

```
<?xml version="1.0"?>
<senml xmlns="urn:ietf:params:xml:ns:senml"
    bn="urn:dev:mac:0024befffe804ff1"
    bt="1414860576"  ver="1">
  <e n="pktid" u="count" v="36350"/>
  <e n="rpm" u="r/m" v="353"/>
  <e n="rpm_av" u="r/m" v="353"/>
  <e n="totrounds" u="count" v="375230"/>
  <e n="totrounds_r" u="count" v="0"/>
  <e n="bearingtemp" u="Cel" v="45.4"/>
  <e n="rssi" u="%" v="90"/>
</senml>
```

The example shown above has a size of 384 bytes. In EXI format, the converted message is 184 bytes when the EXI compressor (OpenEXI is this case) is used in strict Schema mode with the Byte-aligned EXI option enabled. As shown earlier in this section, a message larger than the maximum effective payload of an IEEE 802.15.4 frame will need several frames to be transmitted. A message size of 184 bytes will require two frames, thereby doubling the required power consumption and latency.

A. *Differential message exchange*

By investigating the information that is actually transmitted, it is easy to see that there is a lot of redundant information that is transmitted in each packet, for example the SenML header, MAC-address, name of all tags, etc. A much more efficient way of transmitting the information would be to only transmit the changes from a previous message. Many video encoders, such as MPEG, operate in this way, using a base frame and then only encoding the difference between following frames and the base frame. In RFC 3229 [20], a method for utilizing a similar approach for differential HTTP is proposed. However, since

HTTP does not support event-based communication, no widely used webserver is currently supporting this RFC.

The SenML specification also mentions that resource-constrained nodes do not require a full-fledged EXI compressor in order to generate an EXI stream. Instead, a base version can be generated at compile time, and the node can update individual bytes within the stream at run time, thereby eliminating the need to compress an entire XML message. Even though this is a very efficient method for generating EXI streams, it will still require the entire EXI message to be transmitted over a wireless medium. A much more efficient method would be to combine the mentioned methods by only transmitting the difference between a base message and following messages and the use of pre-compiled EXI messages and only updating certain bytes within the EXI stream.

VI. RESULTS

In order to validate the proposed method of differential binary compression several test cases were implemented and tested. In the following subsection, the characteristics of each implementation are presented, together with implementation details, status of the implementation and test results.

A. *Test A*

This test uses the example SenML message found in [15], section-7. EXI was used in strict schema mode with Byte-aligned compaction. Four tests were compared: standard human-readable XML (including indention and white spaces), M2M-XML (with all whitespaces removed), JSON-encoded, and binary-delta encoded.

B. *Test B*

This test also uses the example SenML message found in [15], section-7 but also adds a timestamp field and a temperature reading as integers. EXI was used in strict schema mode with Byte-aligned compaction. Four tests were compared: standard human-readable XML (including indention and white spaces), M2M-XML (with all whitespaces removed), JSON-encoded, and binary-delta encoded.

C. *Test C*

This test is also on an example SenML message from [15], section-7, with six current readings and one voltage reading. EXI was used in strict schema mode with Byte-aligned compaction. Four tests were compared: standard human-readable XML (including indention and white spaces), M2M-XML (with all whitespaces removed), JSON-encoded, and binary-delta encoded.

D. *Test D*

This test is based on the minimal EEML example found at http://www.eeml.org/xml/0.5.1/minimal.xml. EXI was used in standard schema mode with Byte-aligned compaction. Three tests were compared: standard human-readable XML (including indention and white spaces), M2M-XML (with all whitespaces removed), and binary-delta encoded. All tests were performed using EXI files generated by the OpenEXI EXI processor. All binary files were manually verified using the Bless hex editor on Linux Mint 17 (64-bit). The binary delta compression were tested using two original documents with a number of readings and timestamps slightly modified in order to reflect a change of sensor reading(s) which would cause an event to be transmitted to any CoAP client *Observing* a resource. Figure 4 shows the results when using different semantic models and compression techniques. It is easily seen that the proposed approach of using a delta-encoded EXI message exchange outperforms the standard XML, EXI-encoded XML and JSON models.

This proposed method has shown to be able to compress a series of messages from the same resource to up to 90-95%, as shown in Figure 4. The differential binary delta encoding shown here is the key component in the proposed lightweight message exchange protocol. This protocol is targeting resource-constrained networked embedded systems communicating using CoAP, SenML and EXI. However, any XML-based format can be compressed using this approach. In order to verify that no error has occurred during compression, delta compression or merge, a 32-bit checksum (Adler32) is currently used to verify the integrity of the messages. The option to use Adler32 adds an additional four bytes to each differential message.

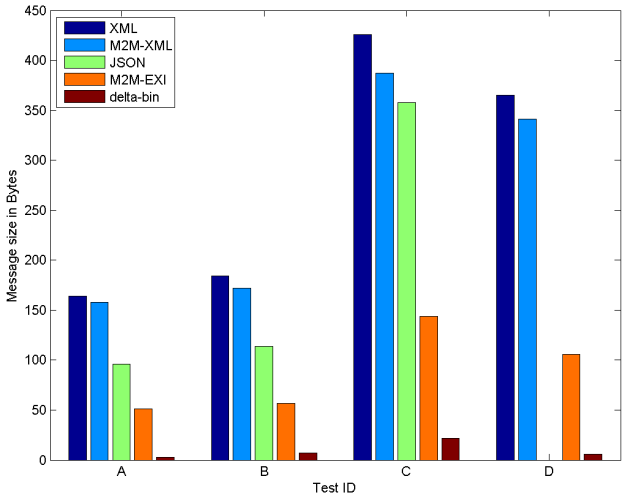

Figure 4: Message compaction results

VII. FUTURE WORK

This paper has presented advancements on efficient service-based communication for low-power Internet of Things devices and networks. However, more work is needed in the following areas:

- Integration with the CoAP Message Queue [30], to enable sleepy nodes to use the proposed delta compression and having the Message broker convert into the standard EXI/text XML representation. This would make the use of delta encoding transparent for end clients.
- Improved integration with existing monitoring and control tools and languages such as SystemJ.
- Investigate power consumption reductions for each communication method.
- Real-time/low-latency CoAP operation, see for example the proposed solution by Ludovici et al. [17].

- Support for high data rate sensors such as microphones and accelerometers and variable-sized differential messages.
- Perform extensive test with SenML, CBOR [29], EEML, and OPC-UA industrial real-world messages.

VIII. CONCLUSIONS

This paper has presented a framework for highly efficient event-based communication suitable for use when integrating resource-constrained devices with standards-based frameworks such as SystemJ, IPSO Smart Objects, Xively and the Arrowhead framework. The message exchange method presented in this paper is designed specifically for resource-constrained embedded systems using low-bandwidth wireless links.

In order to achieve a very high level of efficiency and thereby enable low latency communication, a binary delta compression technique has been integrated into the method. The binary delta compression operates on EXI-compressed XML documents and can compact XML-based documents in for example SenML from several hundred bytes down to tens of bytes when using event-based communication in CoAP (Observe). Currently, there are two reference implementations available: one based on the Contiki operating system on the Mulle sensor and actuator platform, and one based on the Californium CoAP library on Linux.

The implemented data compaction functionality shows that a data reduction of up to 90-95% is possible when streaming EXI data. The proposed method is well suited for use with SystemJ, the Arrowhead Framework and the IPSO Smart Objects model and thus the OMA LWM2M standard.

ACKNOWLEDGMENT

The authors would like to thank Michael Koster and Jamie Jimenez for interesting discussions regarding CoAP and OMA LWM2M. The authors would also like to express their gratitude towards the European Commission, Artemis and the I2Mine and Arrowhead projects for funding.

REFERENCES

[1] Kovatsch, M.; Lanter, M.; Duquennoy, S., "Actinium: A RESTful runtime container for scriptable Internet of Things applications," Internet of Things (IOT), 2012 3rd International Conference on the , vol., no., pp.135,142, 24-26 Oct. 2012. doi: 10.1109/IOT.2012.6402315

[2] Bormann, C.; Castellani, A.P.; Shelby, Z., "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," Internet Computing, IEEE, vol.16, no.2, pp.62,67, March-April 2012. doi: 10.1109/MIC.2012.29

[3] Gruian, F.; Roop, P.; Salcic, Z.; Radojevic, I., "The SystemJ approach to system-level design," Formal Methods and Models for Co-Design, 2006. MEMOCODE '06. Proceedings. Fourth ACM and IEEE International Conference on , vol., no., pp.149,158, 27-30 July 2006. doi: 10.1109/MEMCOD.2006.1695918

[4] Kyusakov, R.; Makitaavola, H.; Delsing, J.; Eliasson, J., "Efficient XML Interchange in factory automation systems," IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society , vol., no., pp.4478,4483, 7-10 Nov. 2011. doi: 10.1109/IECON.2011.6120046

[5] Moritz, G.; Golatowski, F.; Lerche, C.; Timmermann, D., "Beyond 6LoWPAN: Web Services in Wireless Sensor Networks," Industrial Informatics, IEEE Transactions on , vol.9, no.4, pp.1795,1805, Nov. 2013. doi: 10.1109/TII.2012.2198660

[6] Kyusakov, R.; Eliasson, J.; Delsing, J., "Efficient structured data processing for web service enabled shop floor devices," Industrial Electronics (ISIE), 2011 IEEE International Symposium on , vol., no., pp.1716,1721, 27-30 June 2011. doi: 10.1109/ISIE.2011.5984320

[7] Blomstedt, F, Lino Ferreira, L, Klisics, M & Eliasson, J 2014, 'The Arrowhead Approach for SOA Application Development and Documentation', IECON 2014, Dallas, USA.

[8] Chatzigiannakis, I.; Hasemann, H.; Karnstedt, M.; Kleine, O.; Kroller, A.; Leggieri, M.; Pfisterer, D.; Romer, K.; Truong, C., "True self-configuration for the IoT," Internet of Things (IOT), 2012 3rd International Conference on the , vol., no., pp.9,15, 24-26 Oct. 2012. doi: 10.1109/IOT.2012.6402298

[9] Candido, G.; Jammes, F.; de Oliveira, J.B.; Colombo, A.W., "SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications," Industrial Informatics (INDIN), 2010 8th IEEE International Conference on , vol., no., pp.598,603, 13-16 July 2010. doi: 10.1109/INDIN.2010.5549676

[10] Kovatsch, M.; Mayer, S.; Ostermaier, B., "Moving Application Logic from the Firmware to the Cloud: Towards the Thin Server Architecture for the Internet of Things," Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on , vol., no., pp.751,756, 4-6 July 2012. doi: 10.1109/IMIS.2012.104

[11] Eistec AB, http://www.eistec.se (accessed 21 November 2014).

[12] Castellani, A.P.; Bui, N.; Casari, P.; Rossi, M.; Shelby, Z.; Zorzi, M., "Architecture and protocols for the Internet of Things: A case study," Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on , vol., no., pp.678,683, March 29 2010-April 2 2010. doi: 10.1109/PERCOMW.2010.5470520

[13] Dunkels, A.; Eriksson, J.; Finne, N.; Osterlind, F.; Tsiftes, N.; Abeille, J.; Durvy, M., "Low-power IPv6 for the Internet of Things," Networked Sensing Systems (INSS), 2012 Ninth International Conference on , vol., no., pp.1,6, 11-14 June 2012. doi: 10.1109/INSS.2012.6240537

[14] Erl, T.; "SOA Principles of Service Design", Prentice Hall/PearsonPTR. ISBN: 0132344823

[15] SenML specification, http://tools.ietf.org/html/draft-jennings-senml-10 (accessed 19 November 2014).

[16] Ketema, G.; Hoebeke, J.; Moerman, I.; Demeester, P.; Li Shi Tao; Jara, A.J., "Efficiently Observing Internet of Things Resources," Green Computing and Communications (GreenCom), 2012 IEEE International Conference on , vol., no., pp.446,449, 20-23 Nov. 2012. doi: 10.1109/GreenCom.2012.70

[17] Ludovici, A.; Garcia, E.; Gimeno, X.; Calveras Auge, A., "Adding QoS support for timeliness to the observe extension of CoAP," Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on , vol., no., pp.195,202, 8-10 Oct. 2012. doi: 10.1109/WiMOB.2012.6379074

[18] Caputo, D.; Mainetti, L.; Patrono, L.; Vilei, A., "Implementation of the EXI Schema on Wireless Sensor Nodes Using Contiki," Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on , vol., no., pp.770,774, 4-6 July 2012. doi: 10.1109/IMIS.2012.79

[19] Xively (2914), Framework Description. Available at https://xively.com/whats_xively/ (accessed 21 November 2014).

[20] Mogul, J.; Krishnamurthy, B,; Douglis, F.; "RFC3229: Delta encoding in HTTP", http://www.ietf.org/rfc/rfc3229.txt (accessed 21 November 2014).

[21] Collina, M.; Corazza, G.E.; Vanelli-Coralli, A., "Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST," Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on , vol., no., pp.36,41, 9-12 Sept. 2012. doi: 10.1109/PIMRC.2012.63628

[22] Bellifemine, F., Bergenti, F., Caire, G., and Poggi, A., "JADE—a java agent development framework," Multi-Agent Programming, pp. 125-147, 2005.

[23] O'Hare, G. M. P., Collier, R., Dragone, M., O'Grady, M. J., Muldoon, C., and Montoya, D. J., "Embedding Agents within Ambient Intelligent Applications," Bosse, T.(ed.). Agents and Ambient Intelligence:

Achievements and Challenges in the Intersection of Agent Technology and Ambient Intelligence, 2012.

[24] Bordini, R. H., Braubach, L., Dastani, M., El FSeghrouchni, A., Gomez-Sanz, J. J., Leite, J., O Hare, G., Pokahr, A., and Ricci, A., "A survey of programming languages and platforms for multi-agent systems," INFORMATICA-LJUBLJANA-, vol. 30, p. 33, 2006.

[25] Malik, A., Salcic, Z., Roop, P. S., and Girault, A., "SystemJ: A GALS language for system level design," Computer Languages, Systems, & Structures, vol. 36, pp. 317-344, 2010.

[26] Wang, K. I. K., Abdulla, W. H., and Salcic, Z., "Ambient intelligence platform using multi-agent system and mobile ubiquitous hardware," Pervasive and Mobile Computing, vol. 5, pp. 558-573, 2009.

[27] Atmojo U. D., Salcic Z., and Wang K. I.-K., "System-Level Approach to the Design of Ambient Intelligence Systems based on Wireless Sensor and Actuator Networks," Journal of Ambient Intelligence and Humanized Computing (AIHC), in press, doi: 10.1007/s12652-014-0221-3

[28] Nourbakhsh, I., Sycara, K., Koes, M., Yong, M., Lewis, M., & Burion, S, "Human-robot teaming for search and rescue," Pervasive Computing, IEEE, vol. 4, no. 1, pp. 72-79, 2005

[29] Bormann, C.; Hoffman, P.; "Concise Binary Object Representation (CBOR)", https://tools.ietf.org/html/rfc7049 (accessed 20 November 2014).

[30] Koster, M.; Keranen, A.; Jimenez, J.; "Message Queueing in the Constrained Application Protocol (CoAP)W, https://tools.ietf.org/html/draft-koster-core-coapmq-00 (accessed 21 November 2014).

[31] AllJoyn, AllSeenAlliance (2014), Developers overview. Available at https://allseenalliance.org/developers/learn (accessed 21 November 2014).