DOCTORAL THESIS

Efficient IoT Framework for Industrial Applications

Pablo Puñal Pereira

Industrial Electronics

LULEÅ UNIVERSITY OF TECHNOLOGY

Efficient IoT Framework for Industrial Applications

Pablo Puñal Pereira

EISLAB
Luleå University of Technology
Luleå, Sweden

Supervisors:

Jens Eliasson and Jerker Delsing

Printed by Luleå University of Technology, Graphic Production 2016

ISSN 1402-1544
ISBN 978-91-7583-665-2 (print)
ISBN 978-91-7583-666-9 (pdf)

Luleå 2016

www.ltu.se

To my family

Abstract

The use of low-power wireless sensors and actuators with networking support in industry has increased over the past decade. New generations of microcontrollers, new hardware for communication, and the use of standardized protocols such as the Internet Protocol have resulted in more possibilities for interoperability than ever before. This increasing interoperability allows sensors and actuator nodes to exchange information with large numbers of peers, which is beneficial for creating advanced, flexible and reusable systems.

The increase in interoperability has resulted in an increase in the number of possible attacks from malicious devices or users. For this reason, the use of encryption techniques to protect client and server communications has become mandatory. However, even with state-of-the-art encryption mechanisms, there is no protection that can control access to each particular service with fine-grained precision. The nodes within an industrial network of wireless sensors and actuators are resource-constrained embedded devices, and increasing interoperability therefore requires a higher level of computation capabilities. The nodes' intrinsic limitations of memory and processing exert an adverse effect on power consumption and communication delays, resulting in a shorter battery lifetime. Therefore, the standard computing solutions for Internet communications are not directly applicable, and new mechanisms to achieve security, scalability, dependability, interoperability and energy efficiency are needed.

Sensor and actuator networks can transmit sensed data, but they also offer access to the actuators. Such accesses, presumably provided via services, require an access protection scheme. For this reason, the use of access control mechanisms is mandatory. Access control assists in the creation of customized services and access policies. These access policies can isolate access permissions to devices with different roles, such as production and maintenance.

The main contribution of this thesis is a novel, efficient IoT framework for industrial applications, including design, implementation, and experimental validation. The framework includes features for communication protection, authentication, fine-grained access control, zero-configuration networking, and run-time reconfiguration. These technologies and their corresponding energy consumption data clearly demonstrate the feasibility of integrating a battery-operated IoT concept into a functional System of Systems. The provided data also pinpoint the most critical areas for improvement.

Contents

Acknowledgments

This thesis is the result of more than four years of continuous research, development, and learning. During this time I met many people who help me to be the researcher I am now; because without their advice this thesis would not be possible. I need to extend my gratitude to my supervisors, Associate Professor Jens Eliasson and Professor Jerker Delsing, who had trusted me for this Ph.D. position and had invested time and effort guiding me to be on the right track. I also need to manifest my gratitude to my mentor, Dr. Rumen Kyusakov, for his unconditional help, collaborations, and discussions during these long way.

I would like to thank all my colleagues at LTU, because in one way or another, they contributed to this with a comfortable working environment. Special thanks go to Dr. Miguel Castaño for his valuable guidance during my first year, to Dr. Blerim Emruli for all the time that we spend teaching together and coding discussions, to Miguel Gómez, Lara Lorna, and Emilio Rodríguez for their support and points of view, and to Dr. Arash Mousavi because he shared his experiences and perspectives of the life.

This thesis is the last step of a long education road, which started in 2002. It was a tough way that Today has come to an end. In this fourteen years, I had received support from many people, but especially from Professor Julio Martos, Martín Piazzon, Iván Leiva, Dr. Georgy Kornakov and Marisol Robles.

I would also like to thank all the Arrowhead's partners that supported this thesis, with particular mention to Per-Erik Larsson.

A challenge is always a fruitful source of knowledge and motivation, for this reason, I need to mention the Smart Rock Bolt team (Jens, Henrik, Mikael, Claudia, Joakim, and Hasan). Thanks for all the time and support, especially during the stressful moments.

The last part of this thesis has been done remotely. Therefore, I need to say thanks to Ulf Bodin, Jerker Delsing, and Jens Eliasson for trusting me. Also, I would like to thank Artemis, Arrowhead, EMC^2, and CASTT for funding, and thereby making my Ph.D. possible.

Last but not least, I would like to say thanks to my family for all the time together, even being thousands of miles away. With especial mention to my wife Dr. Paloma Díaz Fernández, who kept me alive after spend weekends, nights, and vacations working. Thanks for all your time, discussions, suggestions, guidance, patience, and help.

Luleå, September 2016
Pablo Puñal Pereira

Part I

Chapter 1

Introduction

"Never send a human to do a machine's job."

- Agent Smith

Analyzing the physical environment is something that humanity has been doing for thousands of years, including measuring distance, time, temperature, etc. At first, rudimentary methods based on references such as the sizes of body parts or the positions of the Sun were used. However, with the standardization of measurement units, the first mechanical systems capable of measuring certain physical variables began to appear; these were the first sensors. At present, humanity is in what is known as the Silicon Age, and thanks to the electronic revolution, we can measure any physical variable using electronic sensors. In 1950, the United States Army introduced the capability of communication with a group of sensors as part of the Sound Surveillance System (SOSUS) project, as described by Silverstein [1], which was a network of submerged microphones (hydrophones) for detecting Soviet submarines in the Atlantic and Pacific Oceans. Thirty years later, in 1980, the United States Defense Advanced Research Projects Agency (DARPA), also under the umbrella of the United States Army, developed the Distributed Sensor Network (DSN), as described by Chong et al. in [2]. The DSN project explored the implementation of distributed wireless sensor networks, as the predecessor to the Wireless Sensor Network (WSN).

An actuator is an artifact that is able to modify a physical variable, such as an LED, motor, heater, or valve. The incorporation of actuators and sophisticated mechanisms into a WSN turns it into a Wireless Sensor and Actuator Network (WSAN). Today, with the use of the Internet Protocol (IP), each node in a WSAN can be transformed into an Internet of Things (IoT) device. IoT technology maximizes interoperability, which enables the possibility of connecting any device at any time to another device somewhere in the world. IoT technology for computers and big data centers is not new, but when the area of application is a Wireless Sensor and Actuator Network (WSAN), the scope of the problem changes. The definition of an IoT device used in this thesis is as follows: "An IoT device is a resource-constrained embedded system with the capability to perform a number of well-defined tasks, such as sensing, signal processing, and networking. It

usually has wireless communication capabilities and is powered by batteries." Therefore, according to this definition, an IoT device must be energy efficient.

Currently, the IoT concept is applied not only for industrial usage but also in many examples of domestic applications, such as smartwatches and GPS-based pet trackers. This thesis focuses on the growing area of the Industrial IoT (IIoT), which has many potential applications; however, the complexity of and the requirements for industrial applications are greater than in the case of domestic applications. Therefore, research in this field requires deeper knowledge and the development of more sophisticated technology to achieve these requirements. The IIoT requires communication among hundreds of devices on the same wireless network, creating issues of scalability, and the transferred data require a higher level of security to prevent data leaks and data injection. Interoperability requires appropriate control of the access to IoT devices. Therefore, a fine-grained access control mechanism is needed. Other requirements include robustness and stability.

The use of the IoT concept for industrial applications increases the complexity of the problem, and at first glance, the efficiency of this approach may be questionable. The goal of this thesis is to design and analyze an efficient framework for the Industrial IoT, providing a state-of-the-art approach for industrial applications.

1.1 Problem formulation

The use of IoT technologies in industrial Wireless Sensor and Actuator Networks enables a high level of interoperability, maybe the highest possible level if the network is connected to the Internet. According to statistics from Cisco [3] and Gartner [4], there are approximately 6 billion connected IoT devices in the world today (2016). Therefore, this level of interoperability enhances the ultimate utility of a WSAN; the possibility to consume data produced by a sensor using a mobile phone on the other side of the world was unthinkable only a few years ago. The IoT for industrial applications offers great potential for research over the coming decades, as shown by Khan et al. [5].

However, IoT devices are subject to resource constraints regarding their memory and processing capabilities. Hence, the use of standard protocols leads to increased overheads for delay and energy consumption, and in some cases, these overheads are a technological barrier. Delays can break communications connectivity, and energy overheads can drastically reduce battery life, making a given application impossible in a realistic scenario. This problematic situation gives rise to the first research question addressed in this thesis:

1. *Is it feasible to use IoT-SOA technology in WSANs for industrial applications?*

The answer to this question is not simple, and it encompasses two further questions:

1.1. *What are the benefits of adding IoT technology to industrial WSANs?*

1.2. *Is it possible to increase interoperability while mitigating performance impact?*

Interoperability is an obvious benefit, but an increase in the number of possible connections also increases the number of possible malicious users. Thus, the second research question arises:

2. *How can access to exposed IoT nodes be protected and controlled while maintaining performance?*

   The goal is to analyze the feasibility of using IoT technology in industrial applications, in which the need to manually configure each device should be avoided. This requirement leads to the third research question:

3. *How can zero-configuration operation be achieved for an IoT node?*

To answer all these questions requires a complete analysis of time and energy consumption in industrial WSANs as well as a study of the communication overheads and memory footprints. The answers to questions two and three can be obtained through two independent analyses, but answering question one requires a more detailed analysis of a complete IoT framework. Such an analysis should highlight all factors that create larger overheads in the framework, either to solve them directly or to recommend them as topics for future work.

1.2 Methodology

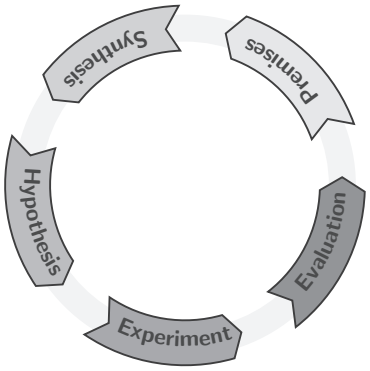


Figure 1.1: Research methodology

Experimental science is "a science that requires the use of tests or prototypes under controlled conditions to demonstrate a known truth, examine the validity of a hypothesis, or determine the efficacy of something previously untried". In 1991, at the Workshop

on Research in Experimental Computer Science (Palo Alto, California), Bob Taylor presented the following principle for a good experimental study: "you should build what you design and use what you build, as only through the extensive use of an artifact do you truly understand the implications of your work" [6].

The methodology used in the work described in this thesis is based on the iterative process illustrated in Figure 1.1. This iterative process begins with a real-world problem to be solved, for which a set of premises is provided that enables the formulation of the initial research questions (research question 1 in this thesis). After a few iterations, with deeper knowledge of the problem, additional research questions can emerge (research questions 1.1, 1.2, 2 and 3). Iteration continues until the evaluation step is successful.

1.3 Thesis scope

The Internet of Things is a research area that has undergone two decades of constant evolution. It is, therefore, a broad area of study that involves a combination of many disciplines, such as wireless communications, networking protocols, machine learning, sensors, actuators, hardware design, information security, cloud computing, and big data. The multi-disciplinary nature of IoT requires collaborative efforts from people with different backgrounds; such collaboration also contributed to the work described in this thesis, which is the fruitful result of a collaboration with many other researchers and industry partners.

This thesis focuses on a feasibility study of IoT technologies for industrial applications. In greater detail, the thesis investigates, proposes, and analyzes an efficient IoT framework that enables the use of cutting-edge IoT technology for industrial applications, thereby updating the previous prevailing design for industrial Wireless Sensor and Actuator Networks. The aspects of the research that are focused on the application layer also overlap with other research areas that are not considered in this thesis, such as encryption and link-layer protocols.

This thesis represents improvements in the IoT field in aspects such as scalability, dependability, security, interoperability, and energy efficiency. This thesis offers two relevant contributions. The first involves research on novel mechanisms for the control of access to IoT resource-constrained devices, resulting in the proposal of an energy-efficient access control scheme that enables fine-grained access control for CoAP-based networks. The second relevant contribution involves research and analysis on all of the mechanisms involved in the efficiency of an IoT network with regard to energy and delays.

1.4 Thesis outline

This is a compilation thesis that consists of two parts. Part I serves as an introduction to the research area, research methodology, and research questions. It also includes descriptions of the solutions proposed to answer the research questions, the experimental evaluation of the proposed solutions, and a discussion of and conclusions obtained from

the experimental results. Part II consists of four peer-reviewed papers that have been published in the proceedings of various conferences, one published journal paper, and two submitted journal papers. All articles fall under the umbrella of the research performed as part of this thesis work; they have been reformatted to follow the thesis layout, but their contents have not been modified.

The remaining chapters in Part I are organized as follows. Chapter 2 offers a brief introduction to the Internet of Things and its historical evolution up through its integration with Wireless Sensor and Actuator Networks. It discusses the benefits and provides examples of the application of this technology; it also describes the new issues that must be addressed for IoT technology to be applicable in industry. Chapters 3 and 4 attempt to solve these problems to make IoT technology feasible for industrial application, focusing on issues of security and efficiency, respectively. Chapter 5 describes the research contributions of this thesis and explains the evolution of how the research questions were addressed during the thesis work. Chapter 6 summarizes the results presented in this thesis, answers the research questions, and describe newly identified issues and directions for future studies.

Chapter 2

Internet of Things

Defining the concept of the 'Internet of Things' is a difficult task, considering that this concept varies from one research area to other. The IEEE Internet of Things group compiled definitions from various Internet associations and research groups in the publication "Towards a definition of the Internet of Things." [7]. The following are the most relevant definitions for this thesis:

> "The basic idea is that IoT will connect objects around us (electronic, electrical, non-electrical) to provide seamless communication and contextual services provided by them. Development of RFID tags, sensors, actuators, mobile phones make it possible to materialize IoT which interact and co-operate each other to make the service better and accessible anytime, from anywhere."
>
> – Internet Engineering Task Force (IETF), 2010

> "A network of items—each embedded with sensors—which are connected to the Internet."
>
> – Institute of Electrical and Electronics Engineers (IEEE), 2014

> "The Internet of Things refers to the unique identification and 'Internetization' of everyday objects. This allows for human interaction and control of these 'things' from anywhere in the world, as well as device-to-device interaction without the need for human involvement."
>
> – HP, 2014

In this thesis, the following definition of the IoT concept is adopted: "An IoT device is a resource-constrained embedded system with the capability to perform a number of well-defined tasks, such as sensing, signal processing, and networking. It usually has wireless communication capabilities and is powered by batteries."

The IoT concept, by definition, changes with the evolution of hardware and software (as do many other concepts related to electronics and/or computation). For this reason, this chapter provides a brief introduction to the historical development of IoT devices. It also presents a description of Wireless Sensor Networks and an overview of possible application areas.

2.1 Historical (r)evolution

With the creation of the Internet Protocol regarded as the beginning of the evolution of the IoT concept, this section analyzes the evolution of all technologies involved in IoT hardware and software up to the present time, 2016.

Internet of Things has been a hot research topic and a number of projects like SOCRADES, IoT-A, SODA, and IMC-AESOP were all addressing a more industrial usage of IoT.

2.1.1 Software

The IoT concept involves several software components, but some of the most important progress has been made in application and link-layer protocols and in operating systems (OSs). This section presents an overview of a few of the most relevant advances.

Application protocols

In the OSI model, the application layer is the abstraction layer responsible for interfacing between communications and the application running on the host. The following is a chronological list of the most representative application protocols for IoT technology:

1996 **RESTful HTTP** The first acknowledged IoT protocol was the Hypertext Transfer Protocol [8], which is a Request/Response protocol for a client-server model and is mainly used to deploy web-based services. The transport layer used is TCP. The general usage of XML makes it overly complex and inefficient for low-power purposes. The recent changes made in HTTP/2 [9] enable header compression to improve the performance of the HTTP protocol, but it is still not suitably efficient for a resource-constrained device.

1999 **MQTT** IBM created the MQ Telemetry Transport protocol based on a client-broker-server architecture, with two types of communication procedures: Request/Response, as in HTTP, and Publish/Subscribe. This protocol is more efficient than HTTP but still uses TCP as the transport layer.

1999 **Jabber** An open-source community developed this protocol for real-time instant messaging (IM). Communication is based on XML, and similarly to MQTT, it supports both Publish/Subscribe and Request/Response communications over a client-server model. This protocol also uses TCP as the transport layer.

2004 **XMPP** The IETF decided to modify the Jabber project by adding TLS for communication encryption and SASL for authentication, renaming the protocol to the Extensible Messaging and Presence Protocol (XMPP) [10].

2007 **MQTT-SN** IBM created a new, more efficient UDP-based version of MQTT named MQTT for Sensor Networks (MQTT-SN) [11].

2011 **WebSockets** This protocol was designed to improve communications between web browsers and web servers, but it can also be used as an independent client-server application protocol. It also uses TCP as the transport layer [12].

2014 **CoAP** The Constrained Application Protocol [13] was created to optimize the efficiency of communications in Wireless Sensor Networks. This RESTful-based protocol is allowed to deploy services (resources) directly on the network nodes. Based on a client-server model, it allows the use of Request/Response and Observe methods. In contrast with previous protocols, CoAP was designed to use a UDP transport layer.

Link-layer protocols

The use of wireless technologies such as Bluetooth and WiFi is quite common for the creation of Wireless Local Area Networks and may be the best solution for mobile sensing platforms because any smartphone can serve as an Internet gateway for both technologies. The barrier hindering the use of both Bluetooth and WiFi is their power consumption. At present, hardware implementations are available that can reduce this power consumption, such as the CC3000 from Texas Instruments [14], which has a transmission consumption of 936 mW and a reception consumption of 331 mW, or the ESP8266 from Adafruit [15], which consumes 561 mW for transmission and 185 mW for reception. Wireless communications represent a large percentage of the total power consumption of an IoT device, meaning that its battery life will directly depend on the selected wireless technology.

In 2007, the IETF created 6LoWPAN, a link-layer protocol with encapsulation and header compression, to allow the use of IPv6 networks over IEEE 802.15.4 [16]. This improvement was probably the greatest step forward for IoT implementation in WSNs, enabling the creation of IP wireless networks for low-power devices (Section 4.4 presents an empirical analysis of the power consumption for this protocol).

Operating systems

An IoT device can function without an operating system; it requires only a functional IP stack. However, to create smart and sophisticated IoT devices, the usage of operating systems is highly recommended. The following is a chronological list of the most common open-source operating systems for resource-constrained devices:

2000 **TinyOS** [17] Not considered an OS in the traditional sense, it used to be defined as a framework for embedded systems with a set of components to enable the deployment of IoT applications, as shown by Levis et al. [18]. It is programmed in NesC

and is widely used for scheduled applications for extremely resource-constrained devices. It requires less than 1 kB of RAM and less than 15 kB of ROM.

2002 **FreeRTOS** [19] Its name stands for Free Real-Time Operating System; this OS, with only 3 C files, is extremely easy to port, read and maintain. FreeRTOS provides a full set of tools for the creation of complex applications with multiple threads, semaphores, and timers. It is supported for more than forty different microcontrollers. A simple application requires less than 2 kB of RAM and less than 12 kB of ROM.

2003 **Contiki** [20] This OS allows multitasking with the use of protothreads, and it is officially supported for more than fifty different microcontrollers. Contiki includes a built-in full IP stack with UDP and TCP support, and it is able to use wireless low-power communications by means of ContikiMAC and 6TiSCH. The OS includes several applications for easily creating servers and clients. Contiki can run in Cooja, a simulation environment, for the easy testing and debugging of applications and communications. A simple application requires less than 2 kB of RAM and less than 30 kB of ROM. The programming language is C. The feasibility of Contiki was presented by Dunkels et. al [21].

2006 **Embedded Linux** [22] This OS is a lightweight version of the Linux kernel that is intended for use on hardware with clear limitations; however, it is not suitable for use on resource-constrained devices because it requires approximately 1 MB of RAM and 1 MB of ROM, specifications that a low-power resource-constrained device cannot meet. It can run any programming language, including Java and Python. Moreover, it can use most of the available programs for desktop versions of Linux.

2011 **OpenWSN** [23] It is not an OS by itself, but it must be included on this list. OpenWSN is an open-source implementation that provides a complete protocol stack based on IoT standards, supporting both UDP and TCP connections. It runs on top of OpenOS, FreeRTOS, and RIOT. A simple application requires approximately 14 kB of RAM and 50 kB of ROM. The programming language is C.

2013 **RIOT** [24] The newest OS on this list, it was designed to improve real-time operation, modularity, and multithreading. It focuses on the use of CoAP and CBOR, thereby reducing memory usage and allowing simple applications to require less than 2 kB of RAM and less than 6 kB of ROM. RIOT supports the C and C++ programming languages.

2.1.2 Hardware

Over the past decade, the explosion in the use of embedded devices for industrial purposes and in many commercial products, such as mobile phones, smartwatches, and miniaturized computers, has motivated the development of many different types of microcontrollers, sensors, radio modules, systems-on-a-chip, etc. This section describes the evolution of some of these hardware technologies.

Microcontrollers and microprocessors

The technological development of both microcontrollers and microprocessors is the same; their evolution involves improving their computational power while expending less energy and reducing their size. The differences between microprocessors and microcontrollers are not clear-cut. Usually, a microprocessor is an integrated circuit that includes only a processing unit, whereas a microcontroller also incorporates RAM and ROM memories and many input/output interfaces.

IoT devices have historically used microcontrollers because their power consumption is lower than that of microprocessors and their computing capabilities have been sufficient for the intended applications. Companies such as Atmel, Microchip Technology, Texas Instruments, ARM and Intel produce some of the most widely used microcontrollers for IoT applications, namely, the AVR32, PIC32, MSP430, Cortex-M7, and Quark, respectively. Figure 2.1 shows the historical and forecasted market for microcontrollers, which shows a continuous increase over the coming years.

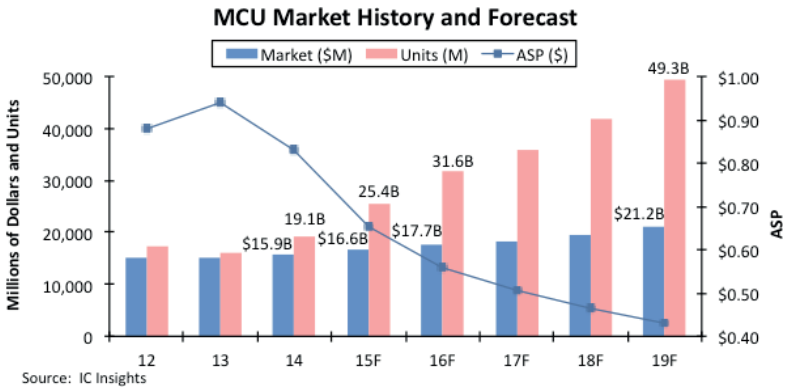


Figure 2.1: Evolution of demand for MCUs based on IC insights

Recent improvements in energy consumption reduction have led to the introduction of microprocessors with low power consumption and high performance, such as the Intel Atom and the ARM Cortex-M73. More powerful devices are suitable for use in applications in which the nodes require a high level of processing power, mitigating the overhead

for communications.

Wireless technologies

Communication in wireless environments requires more energy than in wired environments; in other words, the use of wireless communications increases the power consumption of a system. This limitation has motivated extensive research in this area, leading to the creation of several different wireless communication technologies. The selection of one of these technologies usually depends on the bandwidth, range and power consumption requirements, of which power consumption is typically the most critical factor.

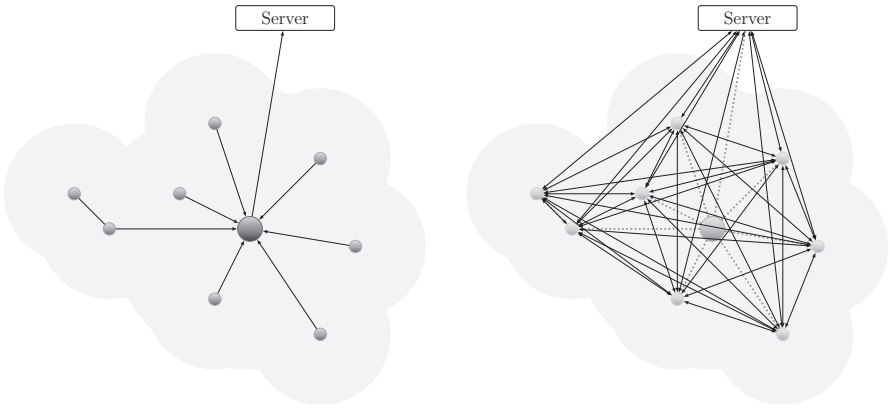
Sensors

Sensor technologies have been under constant development in recent years, and at present, sensors are available for almost every conceivable purpose, such as temperature, proximity, acoustic, chemical, position, and optical measurements, among many others. However, the feasibility of these sensors for IoT applications depends on their power consumption, and currently, some of the greatest improvements are primarily motivated by smartphones. Smartphones are embedded platforms that require energy efficiency, and they include many sensors, such as GPS units, microphones, accelerometers, gyroscopes, and magnetometers. Moreover, with the emergence of smartwatches, this application area is witnessing even greater expansion.

2.2 Wireless Sensor and Actuator Networks

Since the first Wireless Sensor Network was developed in 1950, a WSN has been understood to consist of a group of embedded nodes with connected sensors that are able to measure physical variables, perform data analysis, and communicate with a centralized data collector, or server, for data transmission (see Figure 2.2). The benefit of this architecture is that the nodes do not require a high level of complexity to function; generally, data are communicated from the nodes to the data collector. Implicit acknowledgment mechanisms are not suitably energy efficient for battery-powered devices; the use of an explicit acknowledgment mechanism instead can solve this problem, as shown by Blagojevic et al. [25] and Gonzalez et al. [26]. To simplify the tasks performed by the nodes, WSNs are pulling-based systems, which means that the measurement and transmission processes run periodically, usually with static timeouts.

The term Wireless Sensor and Actuator Network (WSAN) was born with the incorporation of actuators into industrial and domestic WSNs. However, the introduction of actuators requires significant structural changes to a WSN. An actuator requires information about the action to be performed. Therefore, an actuator node needs to be able to receive that information, and to implement this feature, the architecture must be able to communicate in both directions (node to server and server to node). To increase the benefits of using actuators, a WSAN node can use sensor information from



(a) WSN architecture (b) WSAN architecture

Figure 2.2: Comparison between the WSN and WSAN architectures

one or multiple nodes to specify actions for its actuator; thus, a WSAN also requires the implementation of a Machine-to-Machine (M2M) communication capability.

The incorporation of the Internet Protocol into a WSAN turns each node into an IoT device; however, according to some researchers, even without the Internet Protocol, a WSAN node can be regarded as an IoT device.

2.3 Constrained Application Protocol

The IETF Constrained Application Protocol (CoAP) [13] is an application-layer protocol designed to provide web services that work with resource-constrained devices. It is efficient for devices with microcontrollers with small amounts of ROM and RAM and can run over 6LoWPAN network stacks (see Figure 2.3) with high packet error rates. The protocol is designed for low-power networking, allowing nodes to switch into sleep mode to extend their battery life.

CoAP provides a Request/Response interaction model between application endpoints. It supports the built-in discovery of services (resources) and includes key Web concepts such as URIs [27], RESTful interactions [28], and extensible header options. CoAP can easily interface with HTTP for integration with the Web while meeting specialized requirements for constrained environments, such as multicast support, very low overhead and simplicity. CoAP runs over UDP (see Figure 2.3), unlike HTTP, which uses TCP. Currently, there are several groups of researchers porting CoAP to run over TCP, including Bormann et al. [29].

Several relevant features of CoAP are as follows:

- Two types of request messages. A Confirmable Message (CON) is retransmitted

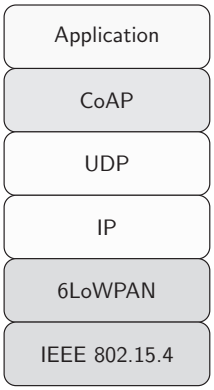


Figure 2.3: OSI model for a CoAP-based application

(a maximum of four times) with an exponential timeout while waiting for an Acknowledged Message (ACK) or the correct response from the server. By contrast, a Non-confirmable Message (NON) is sent without any expected response.

- The URI format allows the use of both standard and specialized service endpoints. One example is the resource discovery scheme defined in RFC 5785 [30], which uses the `/.well-known/core` path and the CoRE Link Format.
- CoAP also allows the sending of large messages with a stop-and-wait mechanism called "blockwise transfers".

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
V		T		TKL				Code								Message ID															
Token (if any, TKL bytes) ...																															
Options (if any) ...																															
1	1	1	1	1	1	1	1	Payload (if any) ...																							

Figure 2.4: CoAP packet format

The packet format of CoAP (see Figure 2.4) includes several parameters that are relevant for the understanding of Chapter 3. These parameters are as follows:

- Version: Indicates the CoAP version number.
- Type: Indicates whether the message is of the Confirmable, Non-confirmable, Acknowledgment or Reset type.

Option Number	Policy [31]
0...255	IETF Review or IESG Approval
256...2047	Specification Required
2048...64999	Expert Review
65000...65535	Experimental use (no operational use)

Table 2.1: CoAP option policy

- Token length (TKL): Indicates the length of the variable-length Token field.
- Code: Indicates the Request Method or the Response Code.
- Message-ID: Unique ID to prevent duplication.
- Option: Indicates the options declared in the message.

The option field allows more information to be included in each CoAP communication, such as a Max-Age for Observe, Uri-Host, or Content-Type. The value of the option field can be empty, an opaque group of bytes, unsigned integers or strings. There are many possible option numbers, as specified in the CoAP option policy (see Table 2.1). Currently, only fewer than twenty option numbers are standardized.

Resources

CoAP allows each device to consume and provide resources. A resource is defined as a simple service that requires the resource provider to perform simple tasks, e.g., transmit a sensor value or turn on an LED. This method of using CoAP resources is supplemented by the IPSO Alliance and its Smart Object Guidelines [32], in which a resource's URI is used as a tag to identify the type, instance, and value of a sensor or actuator.

In this thesis, the terms resource and service have the same meaning given above. A CoAP resource that requires processing or other external resources to provide a response can be considered a service.

2.4 Service Oriented Architecture (SOA)

A Service Oriented Architecture (SOA) is a design architecture for the creation and development of a system based on distributed subsystems. Each system can offer and consume one or more services to perform its tasks. The main benefits of this design are scalability, reusability, and flexibility. A complex system can be divided into many simple subsystems, thus making development and testing faster and easier because each subsystem can be developed individually. Expanding or updating a system simply requires increasing the number of subsystems.

A Service Oriented Architecture for the Industrial IoT makes the best possible use of interoperability and scalability. Industrial monitoring and control is a complex task that

can be divided into simple subsystems (IoT devices), with each IoT device performing a simple task, such as measuring a variable or offering a service to an actuator. Critical components for an industrial application can be duplicated or, in case of failure, replaced.

A Service Oriented Architecture requires many complex mechanisms to function, including Quality of Service (QoS), orchestration, and configuration. The work reported in this thesis was conducted under the auspices of the Arrowhead project, a European project responsible for researching, developing and improving interoperability in complex industrial environments.

Chapter 3

Security

In industrial applications, interoperability is an advantage. Interoperability reduces the costs of operation and maintenance because upgrading a framework with a high level of interoperability requires less investment and effort than upgrading a non-interoperable framework. Two different yet interoperable platforms can be integrated; they can share resources, data, and services without the need for duplication. An interoperable framework that supports various device types can exploit the best features of each device for each situation, e.g., collecting data from a resource-constrained device and processing it on a high-performance server. All these beneficial characteristics are useful in industry, but in environments in which the devices are transmitting sensitive data or offering access to actuators, interoperability poses increased risk. Security is therefore a key concern when deploying an IoT framework in industry. It is especially critical for resource-constrained devices, particularly battery-powered devices. Implementing a security mechanism will inevitably increase power consumption; therefore, for applications in which the battery life is a concern, the design must strike a suitable balance between security and power consumption. In an SOA architecture, each node acts as a service provider, and services are accessible to anyone on the network. To protect these services against malicious access, certain mechanisms are needed. For these reasons, security issues can be divided into two different areas: secure communication and access control.

This chapter presents an overview of current methods of securing communications between IoT devices and proposes a new mechanism to enable efficient, fine-grained access control.

3.1 Secure communications

All computer communications require protection against many different types of attacks, such as packet injection, eavesdropping, replay attacks, and DoS attacks. The framework presented in this thesis is based on CoAP as the application protocol and is especially suitable for 6LoWPAN networks (see Figure 4.2). To maintain interoperability, the security mechanisms must be standardized; otherwise, interoperability will be reduced.

An analysis of the possible standard technologies that can be used over a 6LoWPAN-CoAP stack is therefore needed; such an analysis performed by Hennebert et al. [33] is summarized in Table 3.1.

Layer	Security mechanism	Header overhead	Requirement achieved	Attack
Physical	CSMA-CA	None	Availability	Jamming / collision / flooding
	Secure firmware	None		Node tampering
	Secure element	None		Cloning
Link	MIC	6-26 bytes	Authentication and integrity	Packet injection
	AES encryption only	7-15 bytes	Confidentiality	Eavesdropping
	AES-CCM Nonce	11-29 bytes	Authentication, integrity, confidentiality and freshness	Replay attack
	Address filtering	None	Energy efficient	DoS / battery exhaustion
Adaptation	Hash chain	8 bytes	Integrity	Fragmentation attack
	Split buffer	None	Availability	DoS / buffer saturation
Network	IPsec AH	16 bytes	Authentication of the emitter and network integrity, resiliency, robustness, and resistance	Packet injection, replay attacks
	IPsec ESP	28 bytes	Confidentiality between two peers	Eavesdropping, replay attacks
	Secure routing	-	Availability	Routing attacks
	Secure neighbor discovery	-	Protection of network services	Intrusion
Application	Compressed DTLS ciphered layer	16 bytes	Authorization through a token and authentication of the emitter and integrity and confidentiality between two peers using a given application network, resiliency, robustness, and resistance	Aggregation, data peeking, packet injection
	IDS	-	Network services	Every intrusion

Table 3.1: Security mechanisms for 6LoWPAN networks

3.1.1 Standard end-to-end security mechanisms

Interoperability enables communication among many devices. Usually, such communication requires the use of other intermediary devices, such as routers, switches, and servers. Therefore, the priority is to ensure end-to-end communications, and according to Table 3.1, the main families of mechanisms that are able to provide end-to-end protection are IPsec and DTLS.

Internet Protocol security (IPsec)

Internet Protocol security (IPsec) [34] is the secure evolution of the Internet Protocol (IP). It consists of a collaboration among several different protocols and supports various types of encryption [35]. IPsec includes two mechanisms, Authentication Header (AH) and Encapsulating Security Payloads (ESP). AH provides data origin authentication, protection against replay attacks and connectionless data integrity, whereas ESP provides confidentiality. If guaranteeing confidentiality is a priority, then IPsec-ESP is a reasonable choice. ESP encrypts the original IP packet into the payload of a new IPsec packet, which can be decrypted only using the correct previously deployed or negotiated keys. To negotiate these keys, IPsec supports the Internet Key Exchange protocol, version 2 (IKEv2) [36]. This protocol is useful for avoiding the use of static and long-term keys, thereby increasing security.

Datagram Transport Layer Security (DTLS)

Datagram Transport Layer Security (DTLS) is primarily a UDP evolution of Transport Layer Security (TLS), which runs over TCP. It provides data origin authentication, authorization, data integrity and confidentiality. This protocol initially consisted of two phases, the first being a handshake between the two communicating machines, during which both must authenticate themselves and validate the other using certificates, and the second phase being the transmission of the encrypted information. However, the use of the default version of DTLS is not efficient for resource-constrained devices because the overhead and the use of certificates degrade low-power performance. These problems motivated the development of compressed DTLS, as reported by Raza et al. [37], and the replacement of the certificates with keys, as proposed by Fossati et al. [38], to create a standard and efficient version of DTLS.

3.1.2 Access control analysis

The standard mechanisms described above for securing (end-to-end) communication provide several features for controlling access to each device. The problem is the lack of granularity of these mechanisms. Access can be achieved at several different levels: address (IP address), ID, service, and method. Table 3.2 summarizes which types of access control are provided by IP, IPsec, Black Lists, IKEv2, DTLS, and CoAP. All of these technologies enable control at the service and method levels.

Technology	Access control			
	Address	ID	Service	Method
	coarse-grained		fine-grained	
IP	✗	✗	✗	✗
Black Lists	✓	✗	✗	✗
IPsec	✓	✗	✗	✗
IPsec+IKEv2	✓	✓	✗	✗
DTLS	✗	✓	✗	✗
CoAP	✗	✗	✗	✗

Table 3.2: Access control comparison for different security technologies

The ability to control access by address and ID also provides control over *who* can communicate with the service provider. Access control at the service level also provides the ability to create custom services for each user and user type. Finally, control over access based on method enables the provision of services with different functionalities depending on user type, e.g., a time service that a regular user can access to obtain the time, whereas the same service can be accessed to update the time only by administrators. Therefore, the use of fine-grained access control mechanisms is required.

3.2 Access control

Access control is a mechanism for monitoring service requests issued to a service provider and managing when a communication must or must not be approved. Access control can also enable the identification of a consumer of a service and the provision of relevant information about that consumer to the service, enabling the possibility of providing customized services.

As in the previous section, the use of standard mechanisms in recommended to maintain interoperability. However, an exception arises in this case because the standard solutions have shortcomings that require the implementation of a new, more efficient access control mechanism for IoT applications. This section describes the most common access control standards and their shortcomings and proposes such an efficient access control mechanism.

3.2.1 Standard solutions

The two most popular standard protocols that provide access control functionality are Kerberos and RADIUS. The working principles of the two are different, each offering certain benefits and disadvantages, which will be used as the basis for the creation of an efficient access control method in the following sections.

Kerberos

Kerberos is an access control mechanism that runs over UDP; it uses ticket granting to validate a service consumer to a service provider. The access control process requires a service provider (SP), a service consumer (SC), and a Key Distribution Center (KDC). Each entity has its own key, with the exception of the KDC, which possesses all keys. The process begins with the SC requesting a ticket. To do so, it sends a partially encrypted message with accessible information regarding ID, time and other parameters. The KDC can use the SC's key to decrypt the message; if decryption is successful, the KDC trusts the SC and sends back a packet with timeouts and other parameters that is completely encrypted using the SC's key. The KDC must store that packet and use it again to request a valid ticket to contact the SP. In such a request, the SC generates the ticket that will be utilized by the KDC to access the SP; the content of this ticket includes information about the SC, timeouts, etc., and it is completely encrypted using the SP's key. Then, the SC uses this ticket to request access, and the SP can use its key to decrypt it and extract all information about the SC and the access control policies.

Kerberos provides password protection. There is no password communication, which protects each particular SC and the SPs accessible by that SC. In other access control mechanisms, each SC must have a database of credentials or passwords for its accessible SPs. Kerberos requires the use of a centralized KDC, which provides the convenience of database maintenance. For an IoT device, however, Kerberos is not optimal. Kerberos does not require communication between an SP and the KDC, which is a clear advantage for reducing power consumption, but Kerberos tickets contain all access control information in an encrypted form, which poses in a limitation because of the ticket size and processing complexity.

Remote Authentication Dial-In User Service (RADIUS)

The Remote Authentication Dial-In User Service (RADIUS) is a UDP-based centralized Authentication, Authorization and Accounting (AAA) protocol for the management of users that connect to and use a particular service. The access control process requires a service provider (SP), a service consumer (SC), and a RADIUS server (RS). The SC requests a service from the SP, the SP claims an exchange of authentication information, and this information is then used by the SP to check the access status with the RS through an access request. There are three possible responses from the RS: access-accept, access-reject, and access-challenge. In the case of the third response, a challenge request response from the SC is required to provide more information to the RS via the SP.

RADIUS authentication and authorization processes do not require complex cryptographic operations, and the communications require the transfer of a negligible amount of data; the RADIUS packet size is 20 bytes plus the attributes (see Figure 3.1). For an IoT device, this mechanism is suitably efficient with regard to processing and complexity, but it requires the exchange of several communications, especially for the SP, which consequently compromises the low-power criterion.

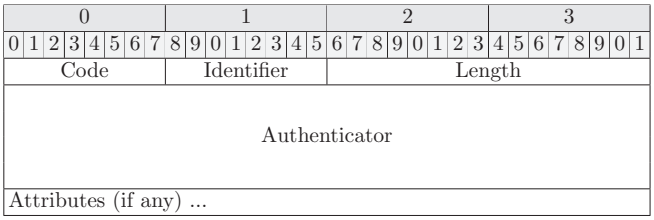


Figure 3.1: RADIUS packet format

Lack of efficient standard solutions for IoT devices

IoT devices are resource-constrained embedded devices, and their processing capability is limited. An increase in processing results in an increase in power consumption, and for battery-powered devices, this is a critical limitation. Moreover, the use of wireless communications further increases the power consumption, which means that if a mechanism requires additional communications, it will represent additional power consumption.

Kerberos requires the use of large encrypted tickets (containing information about the client, time, services, etc.). The transmission and processing of such a ticket requires a large amount of energy, and in a framework with a low rate of data transmission, this type of solution is not efficient. On the other hand, RADIUS requires the exchange of many messages, especially on the service provider's side, also increasing the power consumption of the device.

Thus, neither standard solution is suitably efficient for deployment in an IoT framework. Therefore, a new, more efficient access control mechanism is required to provide energy-efficient and fine-grained access control.

3.2.2 Ticket-based access control

The granularity of an access control mechanism depends on the level(s) at which it is capable of controlling access. In this thesis, the application protocol used is CoAP, which offers services (resources) that can be accessed using a variety of possible methods, such as GET, POST, PUT, and DELETE (see Section 2.3). Therefore, in this context, a fine-grained access control scheme must allow accesses to be controlled by user, service and method. As shown in Table 3.2, this level of control is not possible with current technology. To address this issue, I started working on a new solution [39], which is also described in the Arrowhead book [40] and summarized in this section.

The goal of this access control scheme is to limit the additional communication overheads of the original CoAP protocol, which degrade low-power performance or increase communication delays. To this end, the ticket design of Kerberos and the authentication/authorization mechanisms of RADIUS together are the keys to designing a new mechanism for access control over CoAP. CoAP supports many packet options; the idea is to use one of these options to send the ticket information. The ticket, unlike in Ker-

beros, does not include any additional information; it consists of a group of bytes to identify each identity in the network (clients and servers). The size of the ticket depends on the final application and represents a compromise between the level of security and the power consumption performance.

The use of tickets over CoAP allows the framework to centralize authentication and ticket verification for distributed services. This implementation allows either multiple access control mechanisms for individual systems or a centralized mechanism to prevent inconsistencies, thereby improving its scalability. The authentication and authorization processes are implemented specifically as CoAP services to reduce the overheads on the IoT devices.

Requirements

The proposed access control method assumes the existence of the following information for the IoT devices and the AAA server:

- Secret key (SK): a group of 16 bytes that both an IoT device and the AAA server know.
- ID: the ID of the IoT device.
- Password: the password must be a shared property of the IoT device and the AAA server, and it is never sent during either the Authentication or the Authorization process.

Description

The proposed access control mechanism consists of two different steps, namely, Authentication and Authorization, and each step is implemented via a different service on the AAA server. The Accounting function can characterize access to a service in terms of time duration or number of accesses; this Accounting mechanism is described in this section as an attractive feature for business models, but it has not been implemented or evaluated. All data are presented in JSON [41] format for human readability during development, but for enhanced performance, they can also be encoded in CBOR [42].

Authentication The Authentication process must be performed for every IoT device that must be managed by the access control mechanism, including both service (resource) providers and consumers. This step must guarantee the identification of each IoT device by the AAA server. The server must provide a single ticket to the IoT device, which will be used as an identification tag in all subsequent communications between other entities and the AAA server.

As Figure 3.2 shows, the IoT device acting as a client starts the process with a GET request to the AAA server; then, the server possesses the necessary IP and MAC addresses, and the Challenge Request/Response process begins. The AAA server sends back an authenticator generated specifically for those parameters (a group of

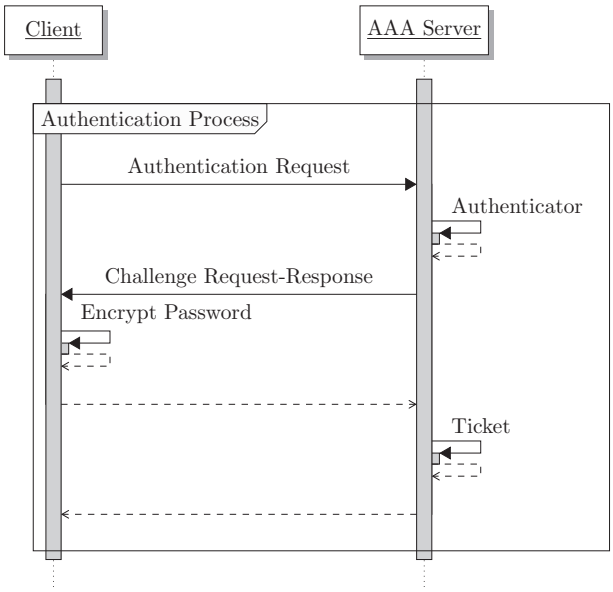


Figure 3.2: Authentication process

16 bytes), expecting a response in the next 15 seconds; otherwise, the authenticator is not valid (see Code 3.1). When the client receives the authenticator, it must encrypt the password using the same Challenge Request/Response process used in RADIUS (described below).

Code 3.1: GET response to initiate the Challenge Request/Response process

```
{
  "version": 1,
  "timeout": 15000,
  "authenticator": "00112233445566778899AABBCCEEDDFF"
}
```

JSON: 93 bytes - CBOR: 69 bytes

The secret key (SK) and the authenticator (A) must each have 16 bytes; if the length of the SK is smaller, then the remaining variable values must be filled with zeros. The password must be split into 16-byte chunks, p_1, p_2, etc., with the last one filled with zeros to maintain the chunk size.

$$b_1 = \text{MD5}(\text{SK} + \text{A}) \qquad c_1 = p_1 \oplus b_1$$
$$b_2 = \text{MD5}(\text{SK} + c_1) \qquad c_2 = p_2 \oplus b_2$$
$$\vdots \qquad \vdots$$
$$b_n = \text{MD5}(\text{SK} + c_{n-1}) \qquad c_n = p_n \oplus b_n$$

The encrypted password can then be expressed as c_1+c_2+...+c_n, where + denotes concatenation. It is sent back to the AAA server together with the entity ID (see Code 3.2); then, the AAA server repeats the process and compares the two results.

Code 3.2: Response with the encrypted password

```
{
  "name": "example name",
  "password": "00112233445566778899AABBCCEEDDFF"
}
```

JSON: 78 bytes - CBOR: 62 bytes

If the encrypted passwords are the same, then the AAA server creates a ticket with a timeout and sends it back to the entity, completing the Authentication process (see Code 3.3).

Code 3.3: AAA server response with the ticket in a successful authentication

```
{
  "name": "example name",
  "password": "00112233445566778899AABBCCEEDDFF"
}
```

JSON: 56 bytes - CBOR: 38 bytes

Authorization Authorization is a process that must be implemented for a service provider to recognize a service consumer as a valid entity or to perform double authentication, in which a consumer also uses this process to verify that the provider is valid and trustworthy.

Before beginning to process a request or response, entity A must ask the AAA server about the validity of the ticket from entity B. This request includes the IP address and the ticket from B in the payload and the ticket from A in the CoAP options (see Figure 2.4). An Authorization request is illustrated in Code 3.4.

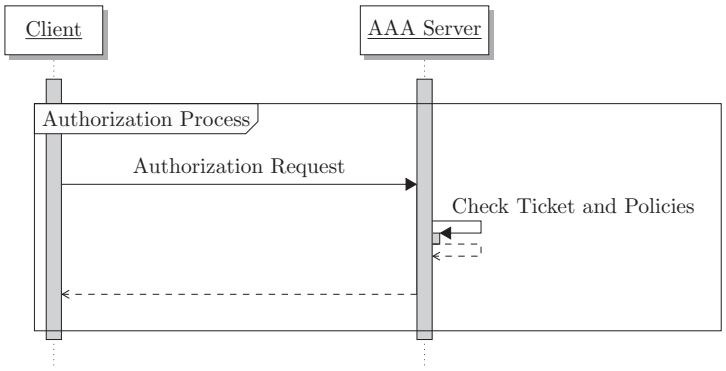


Figure 3.3: Authorization process

Code 3.4: Authorization request

```
{
  "remote_address": "fdfd::AB",
  "remote_ticket": "0011223344556677"
}
```

JSON: 73 bytes - CBOR: 56 bytes

If the request succeeds, then the AAA server will send back a validity confirmation along with B's name, last login, timeout, protocol and ticket expiration time, as shown in Code 3.5. At this point, there are two possible means of addressing the permissions: different types of users with different privileges may be defined, or the relevant policies may be included in the Authentication request.

Code 3.5: Authorization response

```
{
  "valid": true,
  "name": "example_name",
  "login": 1468521292,
  "expire": 1468522292,
  "protocols": "CoAP",
  "timeout": 60000
}
```

JSON: 135 bytes - CBOR: 75 bytes

An implementation based on the definition of users with different privileges has been tested, as discussed in the results section (Section 3.2.2 - Figure 3.8). This solution is more efficient but less flexible than a policy-based implementation. An example of the use of policies is provided in Code 3.6.

Code 3.6: Authorization response with policies

```
{
  "valid": true,
  "name": "example_name",
  "login": 1468521292,
  "expire": 1468522292,
  "protocols": "CoAP",
  "timeout": 60000,
  "policies": [
    {
      "service": "service_name1",
      "allow": [
        "GET",
        "POST",
        "PUT",
        "DELETE"
      ]
    },
    {
      "service": "service_name2",
      "allow": [
        "GET"
      ]
    },
    {
      "service": "service_name3",
      "allow": [
        "GET",
        "POST",
        "PUT"
      ]
    }
  ]
}
```

JSON: 480 bytes - CBOR: 212 bytes

Accounting The Accounting mechanism operates in two different modes: accounting by time and accounting by access instances. Accounting by time allows the service provider to provide services to a consumer for a certain amount of time, after which the access authorization expires and the service provider must notify the AAA server. An example of the transferral of such accounting information from the AAA server during the authorization process is shown in Code 3.7.

Code 3.7: Accounting by time

```
{
"accounting": {
  "type": "time",
  "timeout": 60000}
}
```

JSON: 57 bytes - CBOR: 34 bytes

Accounting by access instances limits the number of accesses to a service that can be made during a particular time window, e.g., access to a service may be allowed forty times in one hour. This accounting method requires a report to the AAA server either when the number of accesses reaches the limit or when the timeout window expires. An example of the transferral of such accounting information from the AAA server during the authorization process is shown in Code 3.8.

Code 3.8: Accounting by access instances

```
{
"accounting": {
  "type": "access",
  "timeout": 6000000,
  "accesses": 40}
}
```

JSON: 79 bytes - CBOR: 49 bytes

Ticket information

The purpose of using tickets is to reduce the communication overhead and power consumption as much as possible by using non-complex processing methods. For this reason, the current implementation of a ticket is essentially a randomized 64-bit number provided by the AAA server. Each ticket must be unique; a duplication (two clients with the same ticket) on the network could compromise the authorization process. Each entity in the network can be identified by the information on its ticket. In this thesis, the ticket content is represented as a hexadecimal number to make it human readable. Every time that an entity requests a new ticket from the AAA server, if the Challenge Request/Response process succeeds, the server responds with the ticket and a timeout. This timeout represents the valid lifetime of the ticket; in other words, it describes for how long all other devices will provide services to the ticketed device or regard it as a trusted entity in the network.

The Authentication mechanism requires the use of an encrypted channel (IPsec, DTLS, TLS, etc.). Thus, the use of a static ticket is not a problem; moreover, the reduced number of communication messages and the timeout related to the ticket help to protect it. However, if the final application were to require increased security, then the ticket could be dynamic. This means that in each communication between a service consumer and the provider, the ticket would need to consist of the hashed data of

the original ticket and another variable parameter, such as the CoAP message ID; see Figure 2.4. All applications considered for the work reported in this thesis assume trust of the confidentiality at the IPsec level; for this reason, dynamic tickets have not been implemented.

Distributed access control

The simplest scenario for the access control mechanism is a network with a service provider (CoAP Server), a service consumer (CoAP Client) and the AAA Server.

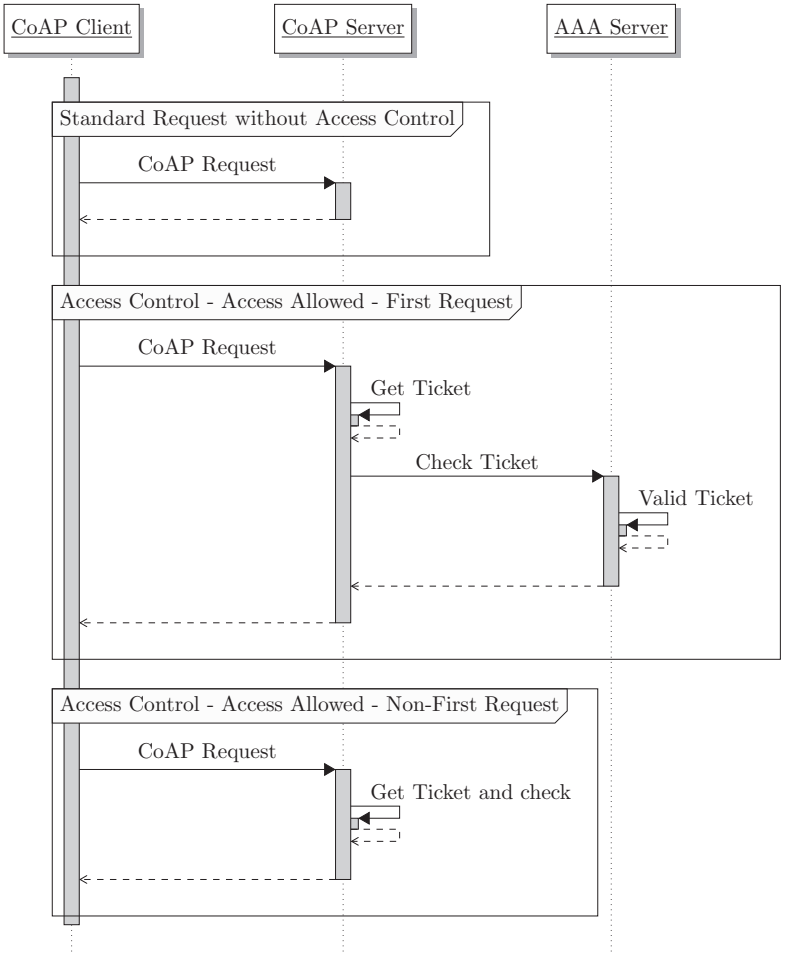


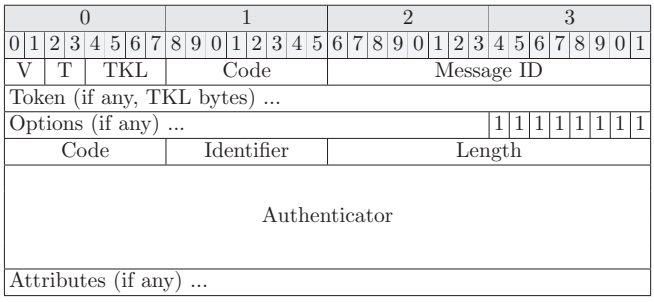
Figure 3.4: All possible access control scenarios

Figure 3.4 shows the three possible scenarios for a successfully requested service. The first case is a service request for a service without access control, in which the service provider provides the service without any other additional process.

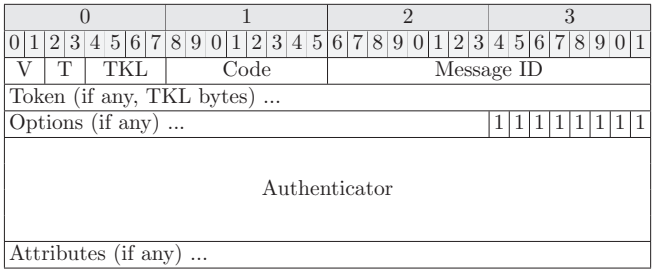
The second case represents a situation in which the service consumer has never attempted to consume the service before or its ticket has expired. Then, the service provider must validate the consumer's ticket with the AAA Server before providing the service.

The third case is similar to the first one and corresponds to a service request in which the provider already holds the consumer's ticket and that ticket is still valid. Then, the provider needs only to check the ticket's timeout and provide the service.

Compatibility with existing standardized access control solutions



(a) Without compression



(b) With compression

Figure 3.5: Proposed packet solutions for the integration of the RADIUS protocol with CoAP

The possibility of integrating the RADIUS protocol with the CoAP protocol gives the proposed framework a flexible authentication method that can be used with a standard RADIUS server. This approach requires no support for the RADIUS protocol on the client. The overhead and the required resources are smaller compared with the use of

both protocols at the same time. This is especially important for resource-constrained sensor nodes. In fact, conversion between a RADIUS packet and a CoAP-RADIUS packet is possible. In this thesis, two solutions are proposed: a CoAP packet with a RADIUS payload (see Figure 3.5(a)) and a compressed CoAP-RADIUS packet. The compression omits redundant information such as the Code, Identifier and Length fields, which can be integrated directly into the CoAP ID and Code fields (see Figure 3.5(b)).

Multi-protocol support

The research presented in this thesis was motivated primarily by the Arrowhead project [43], a project focused on maximizing interoperability for industrial applications. The goal is to offer a smart approach to exchanging services between devices with different characteristics, communication protocols, semantics, etc. in a transparent way. For example, in the case of a high-performance machine consuming services from a resource-constrained device, one may communicate with MQTT, whereas the other may communicate with CoAP. Using current standards, communication between these two devices is not possible, and the available access control mechanisms are not effective for both technologies. The problem of translation between protocols or semantics is not addressed in this thesis, and it is not relevant to the results presented herein. Therefore, a translator is treated as a black box that is able to translate from one protocol to another. See the appended paper D [44] for more information.

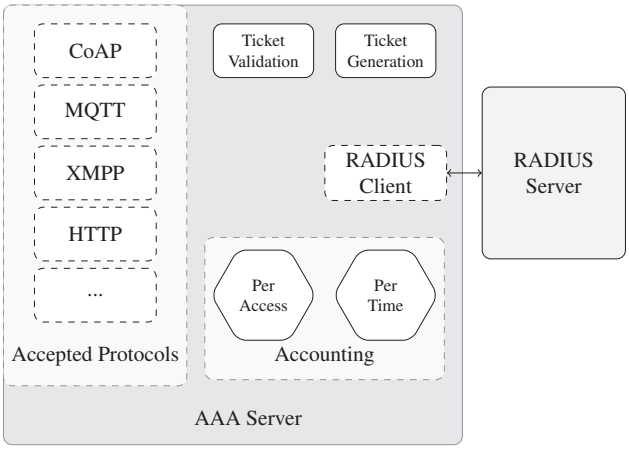


Figure 3.6: Authentication, Authorization, and Accounting server architecture for multi-protocol communications

The access control mechanism presented in this thesis is suitable for application in

multi-protocol communication with the use of a translator, where the translator is a trusted third party. The proposed AAA server is designed to be able to handle requests from different protocols, as shown in Figure 3.6. In an instance of communication between two entities, one using protocol A (in blue color) and the other using protocol B (in red), the access control mechanism functions as in the case of single-protocol communication, as explained in the previous sections, but with the additional use of a translator in direct communications between the two entities, as shown in Figure 3.7. The implementation and development of multi-protocol communications will be addressed in future work, along with research on mechanisms for preventing man-in-the-middle attacks when translators are used.

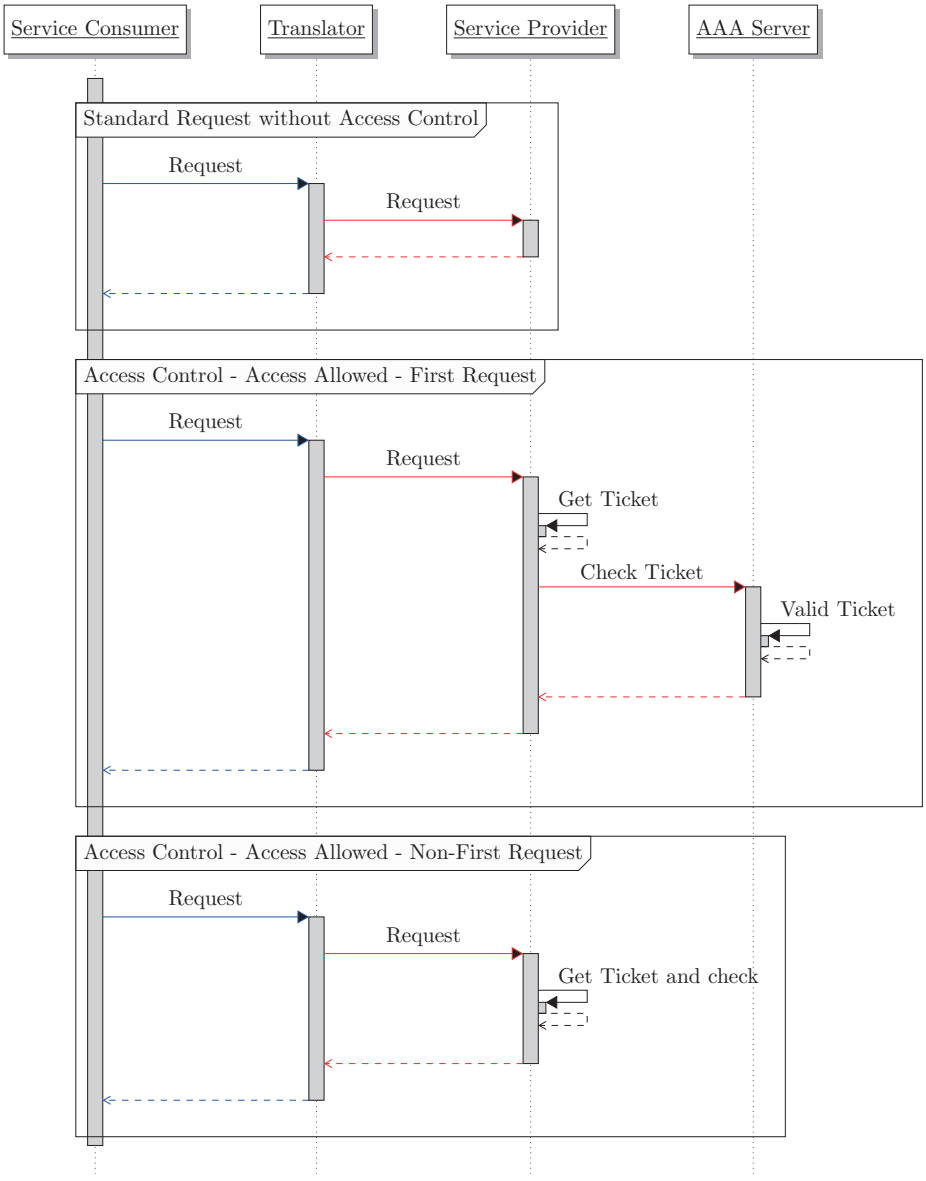


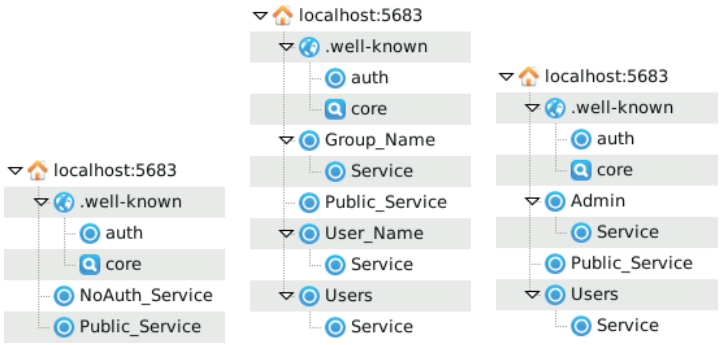
Figure 3.7: All possible access control scenarios for multi-protocol communication

Results

Implementations of this access control mechanism have been proven and tested in many different scenarios and for various purposes, such as mobile machine monitoring (Arrowhead), smart rock bolts (IPSO Challenge), and mining conveyor belts.

The code developed for this access control mechanism includes versions for libcoap 4.1.1 [45], Copper 1.0.0 [46], Erbium [47] and Californium 1.0.4 [48]. All of these are implementations of CoAP RFC 7252 [13] for servers and clients except for Copper, which is only a client implementation.

To demonstrate this fine-grained access control mechanism, the test setup included a resource-constrained device and a laptop. The selected IoT platform for the experiment was a Mulle mk4 from Eistec AB, which is equipped with a 100 Hz ARM Cortex-M4 microcontroller and an IEEE 802.15.4 module capable of communicating via 6LoWPAN. It has 2 MB of flash memory onboard and runs Contiki OS. The Mulle was configured to offer various services subject to access control, including public services ("Public_Service"), services for non-authorized users ("NoAuth_Service"), services by user type ("Group_Name"), services for specific users ("User_Name") and services for administrators ("Admin"). The laptop used a Copper client to display the results via an intuitive GUI. During the tests, various user types were tested, as shown in Figure 3.8. In all cases, a ".well-known/core" request was made, which returned different lists of services for different types of users, namely, for (a) a user requesting access without authorization, (b) a normal user, and (c) an administrator.



(a) Non-authorized user (b) Authorized user (c) Administrator

Figure 3.8: Screenshots of the services available to different types of users, taken during an experiment when performing a CoAP Discovery process

The use of the proposed access control method produces only a small overhead on the message size because of the use of tickets (T). This size can be modified, but during the

work conducted for this thesis, it was fixed at 64 bits. Table 3.3 shows the overheads for each CoAP message type in two scenarios: simple access control and access control with dual authentication. For example, in a GET request process with a confirmable response, three messages are exchanged, namely, the request, the response and the acknowledgment. The overhead size for all messages in the proposed access control mechanism is 64 bits (a single ticket), whereas that for dual authentication is 192 bits (three tickets).

	RFC 7252		Access Control		Dual Auth	
	request	response	request	response	request	response
GET	N		N+T	N	N+T	
POST	N		N+T	N	N+T	
PUT	N		N+T	N	N+T	
DELETE	N		N+T	N	N+T	
OBSERVE	N		N+T	N	N+T	
ACK		N		N		N+T
RST	N		N+T	N	N+T	
.well-known/core	N		N+T	N	N+T	

N: Normal size
T: Ticket size (64-bits)

Table 3.3: Message sizes for normal CoAP vs. CoAP with access control vs. CoAP with access control and dual authentication

3.2.3 Alternatives under development

The IETF Authentication and Authorization for Constrained Environments (ACE) group is an active group that is developing access control solutions for resource-constrained devices. This group is responsible for the development of OAuth 2.0, an access control solution, and OSCOAP, an end-to-end security mechanism. To reduce the size of messages, ACE uses CBOR as a semantic protocol. Both solutions are based on CBOR Object Signing and Encryption (COSE) [49], a specification for creating and processing signatures, message authentication codes, and encryption. COSE also specifies how cryptographic keys must be represented in CBOR format.

OAuth 2.0

OAuth 2.0 [50] is an Authentication and Authorization framework that allows a client to obtain restricted access to a particular resource offered by a resource provider. The framework requires a trusted third-party server that provides two resources, “token” and “intraspect”. These have many similarities to the “authentication” and “authorization” resources for ticket-based access control. One notable difference is the ticket concept

itself; OAuth 2.0 uses access tokens instead of tickets. An access token contains encrypted data that are readable only by the resource provider and the AA server; for this reason, is also called a Proof-of-Possession (PoP) token. Another significant difference is that no communication between the resource provider and the AA server is needed because the permissions are encoded in the token. The client requests an access token for the particular resource to be accessed and the access type; if the access is valid, the AA server generates the access token, which is encrypted using a key known by the resource provider. Code 3.9 shows an example of the information encoded in such an access token request.

Code 3.9: Example request for an access token bound to an asymmetric key

```
Header: POST (Code=0.02)
Uri-Host: "server.example.com"
Uri-Path: "token"
Content-Type: "application/cbor"
Payload:
{
    "grant_type" : "token",
    "aud" : "lockOfDoor0815",
    "client_id" : "myclient",
    "token_type" : "pop",
    "alg" : "ES256",
    "profile" : "coap_oscoap"
    "cnf" : {
        "COSE_Key" : {
        "kty" : "EC",
        "kid" : h'11',
        "crv" : "P-256",
        "x" : b64'usWxHK2PmfnHKwXPS54m0kTcGJ90UiglWiGahtagnv8',
        "y" : b64'IBOL+C3BttVivg+lSreASjpkttcsz+1rb7btKLv8EX4'
        }
    }
}
```

Object Security of CoAP (OSCOAP)

Object Security of CoAP (OSCOAP) [51] is a mechanism that adds and an additional layer of protection to CoAP communications. It provides end-to-end security, encryption, and replay protection and also checks the integrity of messages.

The idea of OSCOAP is to encapsulate the CoAP payload, header, and various options into a COSE object. This COSE object corresponds to the payload of a new CoAP packet, all content of which is encrypted.

Chapter 4

Efficient Industrial IoT Framework

Industrial applications of IoT technology usually require long battery lifetime, in many cases as long as years. These low-power requirements are especially stringent for wireless devices. Battery replacement is usually not easy in the industrial environment. To reduce the number of replacements or to avoid them entirely, the IoT devices must be efficient. Domestic IoT applications do not have the same high requirements as industrial IoT applications; the key differences are the following:

- Scalability - Industrial applications can include tens of thousands of entities.
- Security - A security breach in a factory can result in damage to the environment and/or human personnel as well as enormous costs.
- Interoperability - Industrial applications most often use multiple different systems and technologies, which complicates information exchange and necessitates the use of mediators or translators.

Today, there is no common, widely used standardized solution for networks of this type that require low-power mechanisms. The Arrowhead project, which provided funding for this thesis, is focused on improving the use of IoT for industrial applications. Other organizations, including OMA, IPSO, the IETF (T2TRG, ACE, etc), and the ZigBee Alliance, are also investigating some of these issues. Some companies also focused on the creation of IoT cloud-based platforms like Cumulocity, Xively, ThingWorx, Microsoft Azure Cloud or VxWorks (Intel). Other companies chose a peer-to-peer approach like IPSO, Thingsquare, IzoT, AllJoyn, or IoTivity. A good comparison of these platforms is presented by Derhamy et al. [52]. This thesis addresses this technological gap to make IoT SOA-based technologies feasible for industrial applications. This chapter presents new results in the following areas: a) device life-cycle management, including bootstrapping and configuration; b) efficiency of security mechanisms; and c) a feasibility of the use of standard IoT technologies in industrial applications.

4.1 Network architecture

The proposed framework was designed for resource-constrained IoT Wireless Sensor and Actuator Networks (see Figure 4.1). The communication between each gateway and its node is described by a tree with the minimal number of hops. This limitation is imposed because sometimes the wireless range between a gateway and a node is insufficient and it is better to have an additional hop between them. Each hop-node increases the power consumption because it must handle its own traffic and the traffic of other dependent nodes.

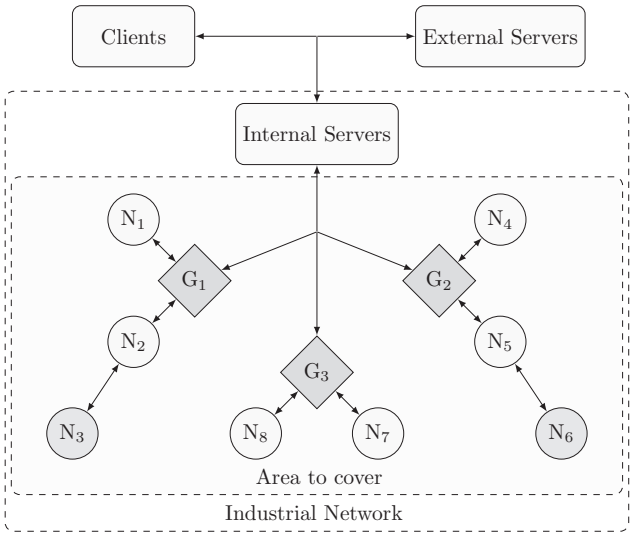


Figure 4.1: Network architecture

The communication between gateways and nodes is achieved through a wireless network based on the network stack shown in Figure 4.2. It consists of a 6LoWPAN layer over IEEE 802.15.4 to enable the IPv6 protocol; it supports IPsec to protect communications, but it may also include other security mechanisms such as TLS, DTLS, or MAC encryption. The application protocol is CoAP, which provides the means for service (resource) deployment on each node. The communication between the gateways and the servers is not addressed in this research; in the proposed network, it can be achieved through either wireless or cable technology because the power consumption is not relevant at the gateway level.

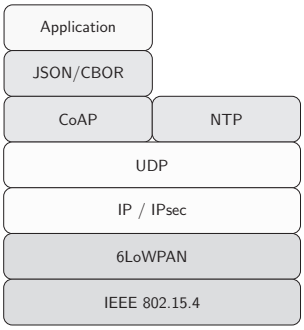


Figure 4.2: Network stack

Each type of element in the topology has a different role; thus, an individual description of each is needed.

Nodes are resource-constrained embedded devices with wireless connectivity and are usually powered by batteries. The wireless connectivity provided by the gateway allows them to communicate with other nodes, servers or clients. The essential task of a node is to sense a physical variable and use an actuator to produce a change in the physical environment. With advancements in microcontrollers, such devices have become capable of performing complex tasks, such as analyzing and evaluating data using filters, finding relevant profiles, and applying adaptive triggers. Manually configuring each node is not feasible in an industrial environment. Therefore, each node can ask for its configuration during boot time; even if the configuration changes during run time, the node must be able to reconfigure itself.

Nodes are not merely clients. In fact, with the use of CoAP, each node can dynamically create services and provide customized services. To do so, they must use an access control mechanism.

In the proposed platform, a single node can provide services based on aggregate data, such as the average temperature of nearby nodes, or take actions based on data from other nodes. In other words, the nodes can create systems of systems. To this end, direct machine-to-machine (M2M) communication is mandatory.

Gateways are embedded computers with two main tasks: providing wireless connectivity to the nodes (thus acting as a standard gateway) and running a Supervisory Control and Data Acquisition (SCADA) system. The SCADA system is responsible for registering each connected node with the Device Manager (DM) and for providing bootstrapping and configuration services as well as the authentication and authorization services for access control. All these services are replicates of the services hosted on the internal server, to decentralize the system and ensure that it continues to function even if the connection to the internal server goes down. Each service is described in detail in the following sections.

Gateways create 6LoWPAN connections to the nodes and Ethernet, WiFi, 4G or 5G connections to the internal server. Therefore, the communication between a gateway and the internal server is conducted using the HTTP/HTTPS protocol. Gateways also extend the IPv6 network of the nodes to the internal network, allowing the internal server to also communicate directly with each node.

Servers are not addressed in the research reported in this thesis, but a basic definition of their functionality is needed for a detailed understanding of the platform. There are no differences between internal and external servers (see Figure 4.1); even an external server can act as an internal server with the use of a VPN connection to the internal network. The names are simply labels to distinguish between servers that are exposed or not exposed to external links such as the Internet. Servers act as gateways with greater computational power and memory and more connections. The greatest difference between a server and a gateway is that a server can provide communication between nodes corresponding to different gateways.

4.2 Services

This section describes all of the mandatory services that the proposed framework must provide to cover all requirements for an IIoT platform: bootstrapping, configuration, device management and access control.

4.2.1 Bootstrapping

The bootstrapping service is compliant with LWM2M OMA Bootstrapping [53], which provides information to IoT devices regarding instances of essential services such as access control, configuration, and the LWM2M server.

The bootstrapping service must run on the gateway and should use a predefined port because this port is information that a node is required to have during its first boot. In other words, the bootstrapping service must be readily available to any IoT device that joins the network.

When a device initiates a bootstrapping request, it can include additional information in the request, such as its serial number, MAC address, and internal software name or version. Using this information, the framework can distribute the IoT devices among different access control, configuration or LWM2M servers. This can help to balance the loads among different servers, ensure a variety of devices with different versions, and improve availability.

Bootstrapping increases the stability and robustness of the framework. If a service is down, it can be replaced by another simply by changing the IP address or port in the bootstrapping response. It supports multiple endpoints for the same service, and when one is busy, the device can use the next. Moreover, in the case that a node is connected to another wireless network with different services or different network routes, it ensures

that everything will function as usual. Code 4.1 shows an example of a bootstrapping response.

Code 4.1: Bootstrapping example

```
{
  "auth": {
    "ip": "fdfd::0A",
    "port": 5683,
    "v": 1,
    "res": "/Authentication",
    "resAlt": "/Authorization"
  },
  "conf": {
    "ip": "fdfd::0B",
    "port": 5682,
    "v": 1,
    "res": "/Conf"
  },
  "dev": {
    "ip": "fdfd::0C",
    "port": 5681,
    "v": 1,
    "res": "/rd"
  }
}
```

JSON: 305 bytes - CBOR: 147 bytes

This service can also be used for simple time synchronization or to deploy other relevant information.

4.2.2 Configuration

The role of a sensor is to take measurements of a physical variable, and the role of an actuator is to take actions on physical variables. The configuration service sets the parameters for how those measurements must be performed, how the data should be analyzed, and which actions must be taken under certain conditions. For example, this service sets sampling rates and triggers, sets filters dynamically, sends alerts to other devices, and manages collaborative analysis, among other tasks.

The configuration service also sets the services that must be active on each device depending on the task to be performed. This on-demand creation of services improves the performance and reusability of each node. In addition, it can be used as a security countermeasure and as a way to reduce power consumption.

The configuration service is a CoAP observable resource, which means that a device can observe the configuration service and, during run time, receive a new configuration to optimize its performance or battery life.

The configuration service has several benefits, most of which are focused on optimizing each device and creating collaborative applications. The power consumption of each device can be

reduced, but at the cost of increasing the complexity of how the services are programmed, which has a direct negative impact on the program's size.

Code 4.2: Configuration example for an IoT temperature measurement device

```
{
  "Services": [
    {
      "name": "TempService",
      "type": "temperature",
      "source": "sens1",
      "interface": {
        "GET": {
          "active": true,
          "return": "sens1"
        },
        "POST": {
          "active": false
        },
        "PUT": {
          "active": true,
          "receive": "trigger",
          "return": "trigger"
        },
        "DELETE": {
          "active": false
        },
        "OBSERVABLE": {
          "active": true,
          "period": 120,
          "return": "sens1"
        }
      }
    }
  ],
  "Actuators": [],
  "Sensors": [
    {
      "name": "sens1",
      "period": 60000,
      "triggered": "yes"
    }
  ]
}
```

JSON: 692 bytes - CBOR: 268 bytes

4.2.3 Device management

In a large network, having all entities registered in a server is useful for management and administration. The idea of having a Device Manager is that during the boot process, each device registers itself with the Device Manager service, which will notify the SCADA system and start the acquisition process. The Open Mobile Alliance (OMA) has proposed the OMA Lightweight Machine to Machine (LWM2M) protocol [54] for standardized device management. At present, there are two widely used solutions: Leshan [55] and Wakaama [56]. Both support a broad range of standard LWM2M features, and both were tested during the research described in this thesis; however, the presented framework uses a customized version based on Leshan.

4.2.4 Authentication and authorization

The authentication and authorization services are part of the access control mechanism presented in the previous chapter (see Chapter 3).

4.3 Case studies

The work presented in this thesis was tested during the development of various projects directly related to industry. For this reason, a description of each of these study cases is needed.

4.3.1 Mobile machinery monitoring

High-load working vehicles are expensive to maintain. A common problem is the maintenance of the ball bearings: changing them too soon results in a useless expenditure of money, whereas changing them too late can cause damage to the wheels, engine, and other parts. Therefore, find the perfect moment at which to change them before they break saves time and money. This project was conducted as part of the Arrowhead project, in collaboration with SKF.

To address this problem, each wheel is assigned a node, with an accelerometer on the axis and a temperature sensor in the lubricant oil of the ball bearing. Using these sensors, we can measure the number of rotations per wheel, the direction, impacts, changes in the temperature of the ball bearing, etc. The nodes are connected to a gateway on the vehicle, with 4G Internet access for communication with the Arrowhead servers. This gateway can collect and analyze data and can transmit the results to Arrowhead when it has connectivity.

Figure 4.3 shows accelerometer data collected for one wheel in a wheel loader field test. Data analysis can be performed to extract the speed of the vehicle, any impacts or vibrations, and, in combination with the Z-axis data, even the angle of the wheel.

4.3.2 Smart rock bolts

Historically, there have been many dramatic accidents related to the mining Industry. For example, 29 workers died in a coal mine in New Zealand in 2010, 33 men were trapped for over two months in a coal mine in Chile in 2013, and 301 workers died in a coal mine in Turkey in 2014. In all these situations, the mine partially collapsed. Rock bolts are metal bars of 3 meters in length that are placed in the wall of a mine to reinforce its structure. However, in some situations, the rock bolts can bend or even break because of the work in the mine or

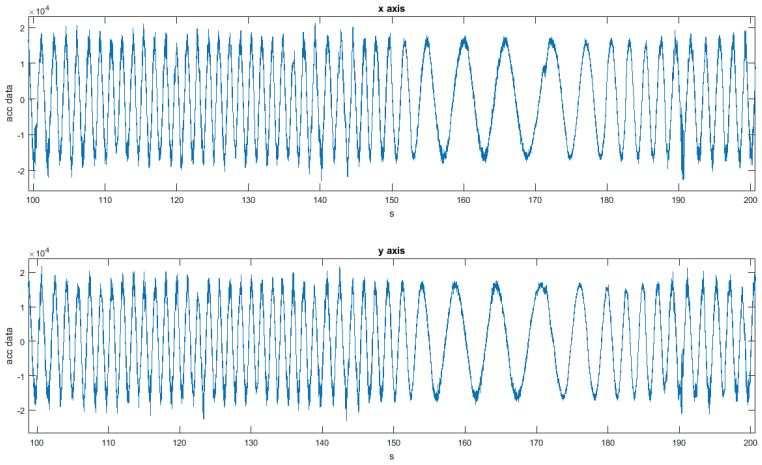
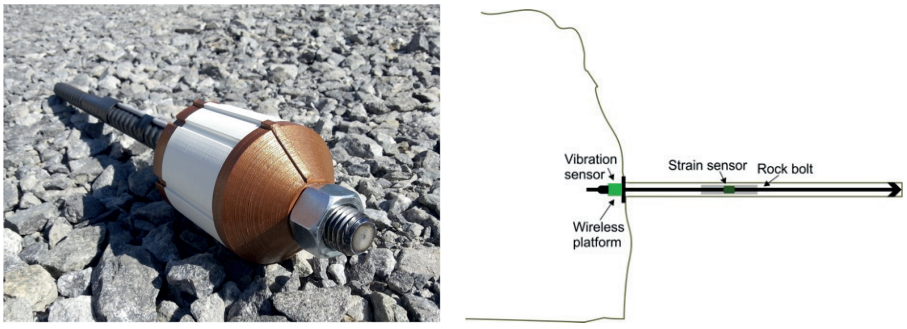


Figure 4.3: Acceleration of one wheel on the X and Y axes during a test

seismic vibrations; under these conditions, the rock bolts can lose their reinforcing properties. Detecting such situations is critical because they can easily turn into collapse scenarios, which pose a high risk to workers and cause economic problems for mining companies.



(a) Smart Rock Bolt prototype

(b) Position and sensor distribution in a tunnel installation

Figure 4.4: Smart Rock Bolt

The objective of the Smart Rock Bolt project is to add sensors and electronics to a standard rock bolt (see Figure 4.4), endowing it with sensing and communication capabilities. The sensors can measure the pressure on the bar and detect a breakdown situation; moreover, they can also detect small vibrations in the walls. The data are analyzed locally, and if a dangerous situation

is detected, the data are transmitted to a gateway, which collects the data and analyzes all possible alarms in the mine. The system can alert the corresponding authority to take actions such as evacuating or reinforcing the tunnel.

This project was presented as part of the IPSO Challenge 2015 [57] and won first place in the contest.

4.4 Experiments and results

This section provides a description of the experimental setup for all performed experiments and an overview of the obtained results.

4.4.1 Test setup

The experimental scenario was a realistic environment outside of laboratory, with a gateway running the SCADA software and multiple nodes connected to it. The experimental results could therefore be affected by radio transmissions from other nodes and by network traffic effects. The goal of the experiments was to determine the energy consumption and delays generated by the use of the various elements of the proposed framework, including secure communications, the access control mechanism, and the bootstrapping and configuration services. The experimental configuration relied on measurements of battery current and voltage that were performed externally to the device by using a 16-bit ADC operating at 1840 Hz to capture rapid events such as radio signals, wakeups, etc. All these measurements were combined into 8 digital inputs that could be used to gain detailed information on the power consumption of each software component. The selected IoT platform was a Mulle (as previously described) with the Contiki OS; all recorded measures were affected by running the OS at the same time, thereby yielding data that were as realistic as possible. The effect of running an OS generates peaks in internal processing queues, communications, internal timeouts, events, etc. which can increase the error levels of the measurements.

4.4.2 Results

IPsec

Security is a crucial feature for IoT communication. To analyze the energy consumption and delays of IPsec-ESP, various configurations were tested. The ESP settings were AES128-CTR and AES-XCBC. To obtain relevant data, the analysis was performed at two CPU speeds (96 MHz and 48 MHz), with and without hardware acceleration for AES128, and for different payloads. Between 20 and 40 measurements were obtained for each payload. Figure 4.5 compares the energy consumption (a) and delays (b) with and without security.

A detailed analysis of the data revealed the overheads in terms of energy and delays for transmission (Figure 4.6) and for reception (Figure 4.7), which demonstrate the feasibility of using this technology for communication protection.

Other services

The configuration of each service is as follows:

Internet Key Exchange v2 (IKEv2) The key exchange process has two steps: initialization and authentication. Therefore, both were analyzed separately. The IKEv2 configuration is AES128-CTR + AES-XCBC + SHA1 + ECP192.

Bootstrapping The bootstrapping analysis included the overheads for the bootstrapping request and the parsing of the response, yielding information about the Device Manager, access control, and configuration services (see Code 4.1).

Configuration The configuration analysis included the configuration request, the parsing of the response, the configuration of a sensor and the creation and deployment of a new service (see Code 4.2).

Authentication The authentication analysis included the authentication request, the Challenge Request/Response process, and the parsing of the ticket and attributes.

Authorization The authorization analysis included the first authorization request and the parsing of timeouts and permissions.

Device Manager The Device Manager analysis included the registration of a node in the OMA LWM2M Server.

Figure 4.8 shows the energy consumption and delays for each service at two CPU speeds (96 MHz and 48 MHz), and Table 4.1 presents the values obtained in the experiments in more detail.

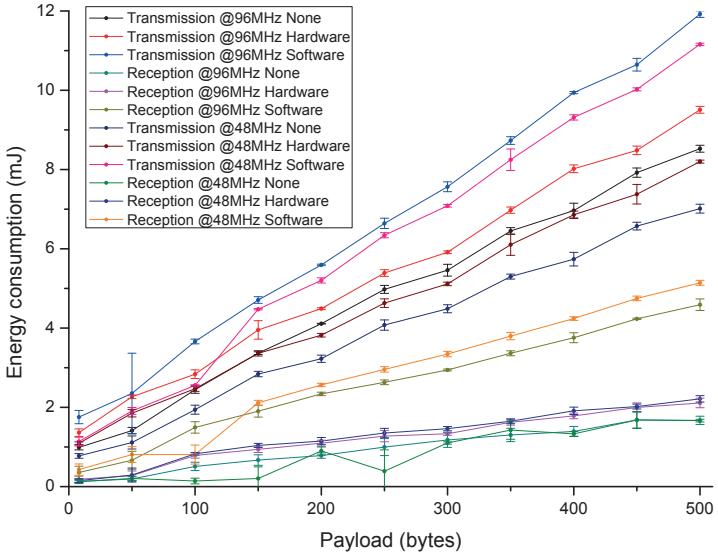
Service	Power (mW)		Delay (ms)		Energy (mJ)	
Speed (MHz)	96	48	96	48	96	48
IKE_INIT	195.7	158.6	2145.6	3789.2	419.9	601.1
IKE_AUTH	209.8	168.1	3916.5	10650.1	821.8	1791.2
Bootstrapping	68.0	65.2	56.4	55.8	3.8	3.6
Configuration	170.9	134.7	81.9	84.2	14.0	11.3
Authentication	197.9	158.6	188.4	232.8	37.3	36.9
Authorization	74.8	71.7	56.9	56.4	4.8	4.0
Dev. Manager	113.7	90.8	72.8	71.8	8.2	6.5

Table 4.1: Analysis of power consumption, delays and energy overheads for each service

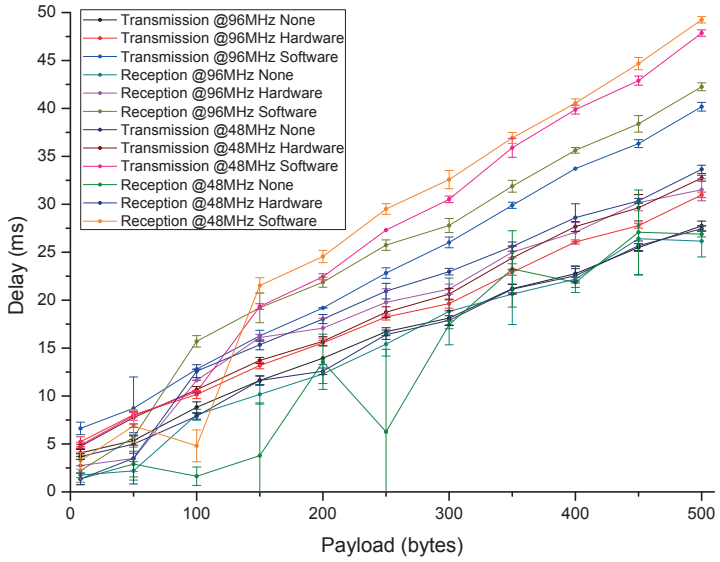
4.4.3 Summary

For smaller payloads, communication incurs lower power consumption and shorter delays. The effect of having the Contiki OS running on the device during the measurements has a larger impact for smaller payloads. For this reason, the data in Figure 4.6 and Figure 4.7 appear

inconsistent, with high overheads and large error bars on small payloads. However, the overhead values for payloads over 200 bytes seems to be consistent and stable. Therefore, to compare the overheads between different configurations, those are the values that must be used.

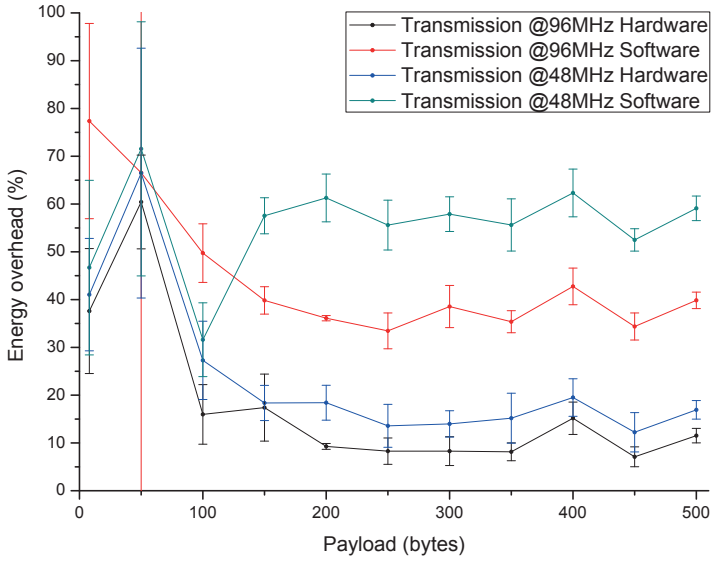


(a) Energy consumption for transmission and reception at 96 MHz and 48 MHz

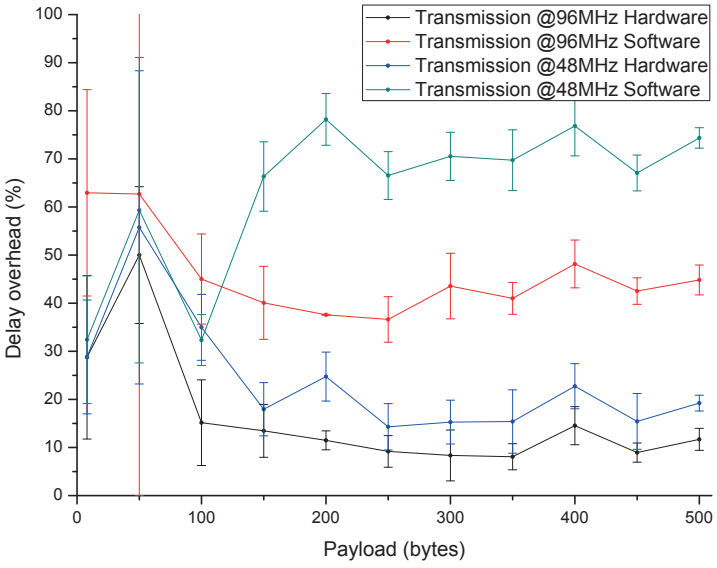


(b) Delays for transmission and reception at 96 MHz and 48 MHz

Figure 4.5: Analysis of IPsec-ESP communication in terms of energy consumption and delays

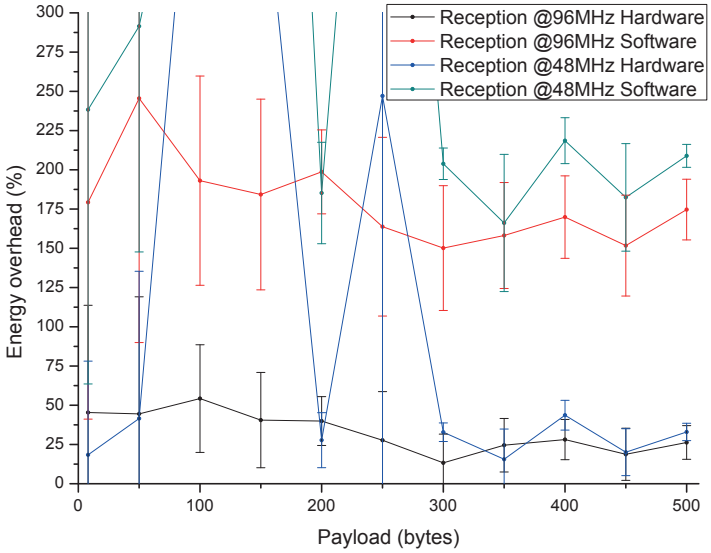


(a) Energy overheads for transmission at 96 MHz and 48 MHz

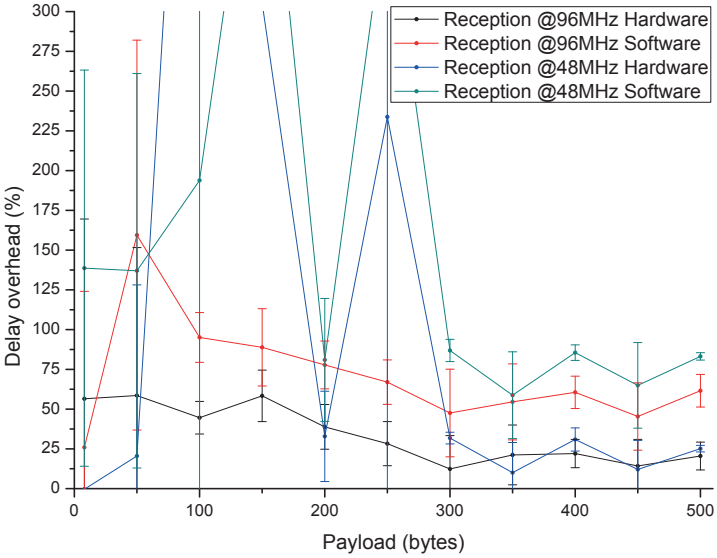


(b) Delay overheads for transmission at 96 MHz and 48 MHz

Figure 4.6: Analysis of IPsec-ESP communication in terms of overheads for transmission

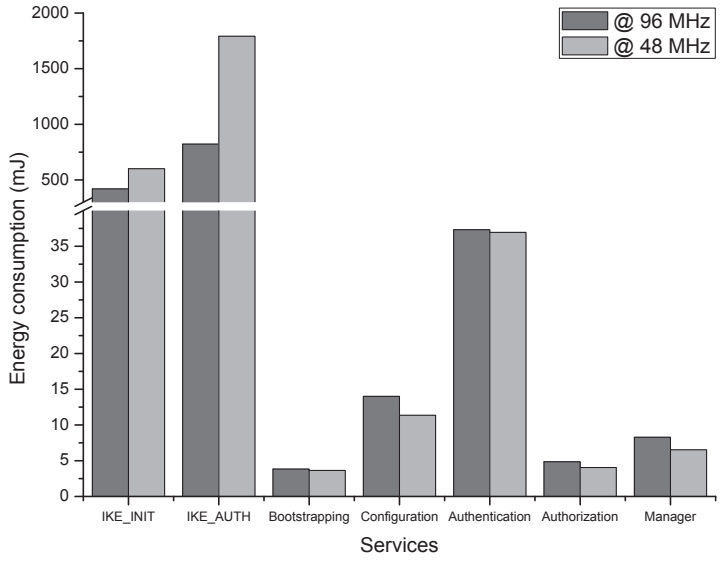


(a) Energy overheads for reception at 96 MHz and 48 MHz

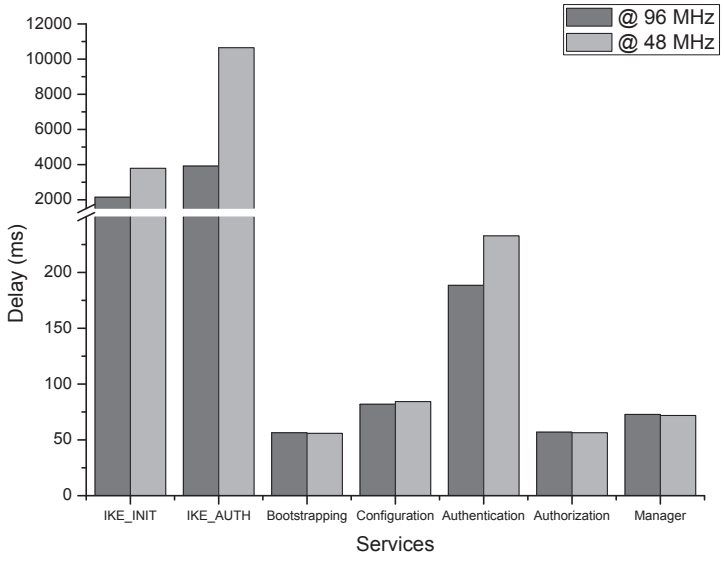


(b) Delay overheads for reception at 96 MHz and 48 MHz

Figure 4.7: Analysis of IPsec-ESP communication in terms of overheads for reception



(a) Energy consumption for each service at 96 MHz and 48 MHz



(b) Delays for each service at MCU frequencies of 96 MHz and 48 MHz

Figure 4.8: Analysis of each service in terms of energy consumption and delays

Chapter 5

Contributions

This work was performed within the framework of the Arrowhead project, a European project to improve interoperability in industrial environments. Some of the focuses of this project include service orchestration, system orchestration, multi-protocol communications, zero-configuration operation, and Quality of Service.

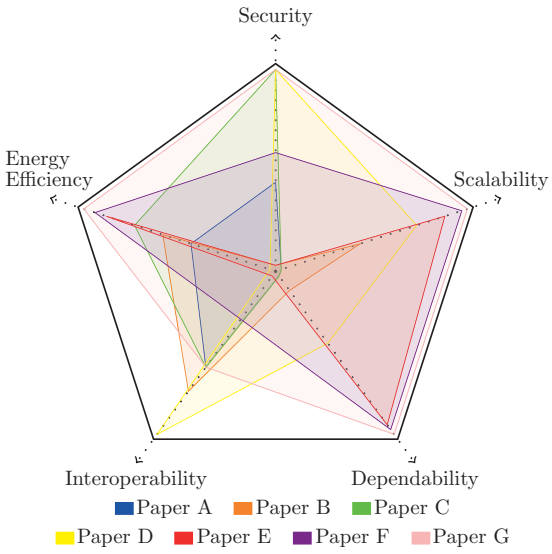


Figure 5.1: Contribution of each paper to each IoT research area

The author has contributed to various areas of research related to IoT-WSNs, all of which are interconnected, in developing this thesis: interoperability, scalability, dependability, efficiency

and security (see Figure 5.1). These areas have been described and discussed in previous chapters (see Chapter 2, Chapter 3 and Chapter 4).

This chapter presents a summary of the appended papers and an outline of the author's main contributions to each paper.

Paper A: A Feasibility Study of SOA-enabled Networked Rock Bolts

Authors: Jens Eliasson, Pablo Puñal Pereira, Henrik Mäkitaavola, Jerker Delsing, Joakim Nilsson and Joakim Gebart

Published in: Proceedings of 2014 IEEE 19th International Conference on Emerging Technologies & Factory Automation (ETFA 2014): Barcelona, Spain

In this paper, research concerning the use of IoT rock bolts in mines is presented. Each rock bolt can measure strain and seismic activity; each node provides its data as a service over a wireless mesh network. Using the collected data it, the ability to detect falling rocks and the presence of mobile machinery is demonstrated.

The author's main research contribution to this paper was a study of the possible systems that could be used to protect the network against potential attacks.

This paper was presented at the IEEE 19th International Conference on Emerging Technologies & Factory Automation (ETFA) in Barcelona, Spain, in September 2014.

Paper B: EXIP: A Framework for Embedded Web Development

Authors: Rumen Kyusakov, Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Published in: Proceedings of ACM Transactions on the Web, 2014

This paper presents the design and implementation techniques of the EXIP framework for embedded Web development. The framework consists of a highly efficient EXI processor, a tool for EXI data binding based on templates, and a CoAP/EXI/XHTML Web page engine. A prototype implementation of the EXI processor is presented and evaluated herein. It can be applied to Web browsers or thin server platforms using XHTML and Web services for supporting human-machine interactions in the Internet of Things.

This paper presents four major results: (1) theoretical and practical evaluations of the use of binary protocols for embedded Web programming; (2) a novel method for the generation of EXI grammars based on XML Schema definitions; (3) an algorithm for grammar concatenation that produces normalized EXI grammars directly, consequently reducing the number of iterations during grammar generation; and (4) an algorithm for the efficient representation of possible deviations from the XML schema.

The author's main research contributions to this paper were the integration of the EXIP library with CoAP in an embedded system and the demonstration of direct interaction between a mobile IoT device and a web browser, with the direct provision of EXIP data from the resource-constrained device.

This paper was published in ACM Transactions on the Web in October of 2014.

Paper C: An Authentication and Access Control Framework for CoAP-based Internet of Things

Authors: Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Published in: Proceedings of the 40th Annual Conference of the IEEE Industrial Electronics Society (IECON 2014), Dallas, USA

The necessity of a fine-grained access control method for CoAP networks is described and justified in this paper. It presents an analysis of other security mechanisms that can be useful in combination with CoAP in constrained embedded systems, identifying the shortcomings of these mechanisms and the reasons to create a new access control mechanism for CoAP systems. The author's main research contribution to this paper was the design of a fine-grained access control mechanism consistent with the power-efficient CoAP concept for small devices to reduce overhead and the implementation of a small network to demonstrate its performance.
This paper has been accepted for publication in the Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON) in Dallas, USA, November 2014.

Paper D: Translation Error Handling for Multi-Protocol SOA Systems

Authors: Hasan Derhamy, Jens Eliasson, Jerker Delsing, Pablo Puñal Pereira and Pal Varga

Published in: Proceedings of the IEEE 20th International Conference on Emerging Technologies & Factory Automation (ETFA 2015): Luxembourg, Luxembourg

The problem of networks using multiple protocols is addressed in this paper. In an attempt to increase interoperability in networks of this type, this paper proposes a solution based on protocol translation and a study of how to handle specific protocol error messages.
The author's main research contribution to this paper was the analysis of the security aspects involved in translating between protocols.
This paper was presented at the IEEE 20th International Conference on Emerging Technologies & Factory Automation (ETFA) in Luxembourg, Luxembourg, in September 2015.

Paper E: The Arrowhead Framework Configuration Approach

Authors: Oscar Carlsson, Pablo Puñal Pereira, Jens Eliasson, Jerker Delsing, Bilal Ahmad, Robert Harrison and Ove Jansson

Published in: Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON 2016), Florence, Italy

The purpose of the Arrowhead project is to provide services for configuration, bootstrapping, and deployment to enhance dependability and zero-configuration capabilities. Several use cases for these services are presented in this paper, including building automation, the manufacturing industry, IoT devices and the process industries.
The author's main research contributions to this paper were the research, development and testing of low-power services that can be used by a resource-constrained IoT device to implement bootstrapping and configuration for sensors, actuators, and services.
This paper has been accepted for presentation at the 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON) in Florence, Italy, in November 2016.

Paper F: Using Internet of Things for Industrial Applications: A Feasibility Check

Authors: Jens Eliasson, Pablo Puñal Pereira and Jerker Delsing
Submitted to IEEE Journal of Sensors

This paper presents a condition monitoring architecture for industrial applications based on IoT devices using the OMA LWM2M protocol, IPSO Smart Objects, and the Arrowhead Framework. The paper analyzes the feasibility of applying this technology to rock bolts in the mines, with a focus on performance and lifetime.
The author's main research contributions to this paper were the technical aspects, research, develop and implementation of the architecture, with particular attention to reducing the power consumption of the IoT devices.

Paper G: An Efficient IoT Framework for Industrial Applications

Authors: Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing
Submitted to IEEE Internet of Things Journal

The use of IoT technology for condition monitoring has gained relevance over the past five years; this paper presents an Industrial IoT platform for condition monitoring with the capability to adapt itself to the power consumption of each node, improving the nodes' measurements or lifetimes when needed. The paper also studies the impact of each applied technology to analyze the power consumption and delays; these technologies include communication features such as access control and encryption as well as functional features such as zero-configuration networking, device management, and reconfiguration at run time.
The author's main research contributions to this paper were the design and implementation of the platform as well as the execution of the power consumption and delay analysis to optimize the platform performance and lifetime.

Chapter 6

Discussion

To conclude this thesis, some reflection on the obtained results and the research questions presented in the Introduction (Chapter 1) is required. As is illustrated in Figure 6.1, the thesis research began with a real-world problem that motivated these Ph.D. studies, i.e., the current lack of a mechanism for implementing efficient Wireless Sensor and Actuator Networks with high interoperability for industrial applications.

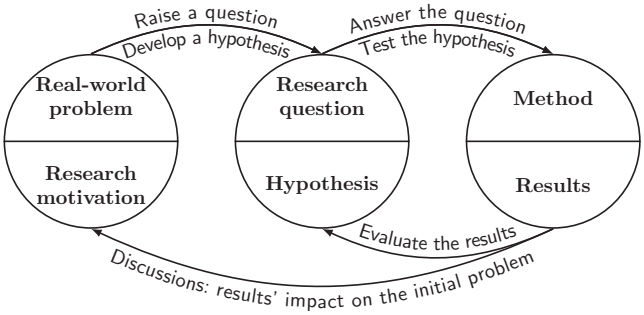


Figure 6.1: Research methodology for this thesis

This real-world problem of interoperability was addressed close to twenty years ago with the adoption of Internet technology in embedded devices, as discussed by Delsing et al. [58] and Sveda et al. [59]. However, the inefficient use of the IP protocol caused power consumption to increase, which prevented the widespread introduction of this technology into industrial WSANs, especially for battery-powered nodes. In recent years, international alliances such as the International Internet Consortium (IIC) [60], the IPSO Alliance [61], the Open Mobile Alliance (OMA) [62], and the Internet Engineering Task Force (IETF) have been investigating the standardization and promotion of the use of the Internet Protocol. The development of 6LoWPAN finally enabled the use of IP (IPv6) in wireless low-power networks, and the use of CoAP as an application protocol allows an HTTP-like protocol to be used to deploy services

on resource-constrained devices.

The initial hypothesis of this thesis was that CoAP could be used over 6LoWPAN to create a Service Oriented Architecture for industrial WSANs. The subsequent research, based primarily on experimental work, yielded the results reported in Paper A. Evaluation of this research motivated the following question: "Is it feasible to use IoT-SOA technology in WSANs for industrial applications?" considering all of the requirements of IoT devices, i.e., bootstrapping, zero-configuration networking, access control, etc. The hypothesis-results-evaluation loop required several iterations to evaluate the impact of the results on the initial problem. This final evaluation is the reason for this chapter.

The accumulated results demonstrate that enabling IoT technology with services for bootstrapping, configuration, access control, device management, etc. increases the computational complexity on the nodes but allows their constrained resources to be optimized to reduce inefficient overheads as much as possible. As demonstrated in Chapter 4 (see Figure 4.5), the highest power consumption occurs when a node is using wireless communication, especially during transmission. Thus, at this point, a comparison between polling-based and event-based WSANs, as considered in this thesis, is mandatory.

Communication efficiency

There are two perspectives from which communication efficiency can be evaluated: *data efficiency* and *energy efficiency*.

- *Data efficiency* (see equation 6.3) is the capability of a system to acquire only valuable data; in other words, is a measure of how well a system reacts when the measured value of a source changes. If the data efficiency is 1, the system can obtain all valuable data. If it is lower than 1, this means that the system is losing some relevant data.

  $$\text{Data}_{\text{acquired}} = f \cdot t$$

  where f is the frequency at which the system acquires data.

  $$\text{Data}_{\text{relevant}} = f_{source} \cdot t$$

  where f_{source} is the frequency of the measured source, or how rapidly the measured physical variable can change. A change to a measured physical variable is considered a relevant datum; for example, if the temperature in a room is stable, then only one of all recorded measurements is a relevant datum, whereas if the temperature is not stable, all varying values are relevant data.

$$\text{Data}_{\text{acquired and relevant}} = \begin{cases} f \cdot t, & \text{if } f < f_{\text{source}} \\ f_{\text{source}} \cdot t, & \text{if } f \geq f_{\text{source}} \end{cases} \tag{6.1}$$

$$\text{Data}_{\text{non-acquired and relevant}} = \begin{cases} (f_{\text{source}} - f) \cdot t, & \text{if } f < f_{\text{source}} \\ 0, & \text{if } f \geq f_{\text{source}} \end{cases} \tag{6.2}$$

$$\mu_{\text{data}} = \frac{\text{Data}_{\text{acquired and relevant}}}{\text{Data}_{\text{relevant}}} = \begin{cases} f/f_{\text{source}}, & \text{if } f < f_{\text{source}} \\ 1, & \text{if } f \geq f_{\text{source}} \end{cases} \tag{6.3}$$

- *Energy efficiency* (see equation 6.5) is the capability of a system to acquire only the important data and discard all irrelevant values, i.e., a measure of how good the system is at obtaining relevant values. If the energy efficiency is 1, the system acquires only appropriate values. If it is lower than 1, this means that the system acquires some irrelevant values.

$$\text{Data}_{\text{acquired and non-relevant}} = \begin{cases} 0, & \text{if } f < f_{\text{source}} \\ (f - f_{\text{source}}) \cdot t, & \text{if } f \geq f_{\text{source}} \end{cases} \tag{6.4}$$

$$\mu_{\text{energy}} = 1 - \frac{\text{Data}_{\text{acquired and non-relevant}}}{\text{Data}_{\text{acquired}}} = \begin{cases} 1, & \text{if } f < f_{\text{source}} \\ f_{\text{source}}/f, & \text{if } f \geq f_{\text{source}} \end{cases} \tag{6.5}$$

The total efficiency μ must therefore include both energy and data efficiency, as shown in equation 6.6.

$$\mu = \mu_{\text{data}} \cdot \mu_{\text{energy}} = \begin{cases} f/f_{\text{source}}, & \text{if } f < f_{\text{source}} \\ f_{\text{source}}/f, & \text{if } f \geq f_{\text{source}} \end{cases} \tag{6.6}$$

An energy-efficient system is usually a data-inefficient system and vice versa, i.e., an energy-efficient system acquires data at a low rate to save energy, meaning that the system will likely lose some relevant data, whereas a system with a high data acquisition rate can measure all relevant data but will also likely measure some data that are not relevant and is therefore an energy-inefficient system. The source of the relevant data is usually a physical variable that the system can sense, and with the exception of time, there is no physical magnitude that exhibits a constant frequency of change. Thus, upon adding temporal variability to equation 6.6, the following result is obtained:

$$\mu(t) = \begin{cases} f/f_{\text{source}}(t), & \text{if } f < f_{\text{source}}(t) \\ f_{\text{source}}(t)/f, & \text{if } f \geq f_{\text{source}}(t) \end{cases} \tag{6.7}$$

In a polling-based system, the polling rate may be either constant or even adaptive[1], but in either case, the polling frequency can be considered as a constant in time. Therefore, such a system cannot be perfectly efficient.

$$\boxed{\mu(t)|_{\text{polling}} \neq 1}$$

As illustrated in the Figure 6.2, polling-based systems are not efficient even for small changes in f_{source} over time. The efficiency is reduced by more than half when $f_{source} \leq (f_{polling}/2)$ or $f_{source} \geq 2 \cdot f_{polling}$.

By contrast, an event-based system acquires data only when the source changes; thus, its frequency of acquisition is variable over time, and an event-based system can potentially be fully efficient.

$$\boxed{\mu(t)|_{\text{event}} = 1}$$

[1] An adaptive polling-based system can modify its polling rate depending on external conditions. For example, if the system is measuring wind and the wind is more stable during the night than during the day, the polling rate may be different between day and night.

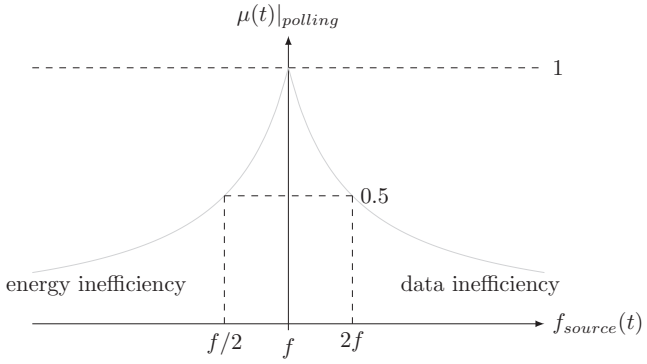


Figure 6.2: Evolution of the efficiency between $f_{source}(t)$ and f

The inefficiency of polling-based systems is a strong reason to conclude that the WSNs must implement event-based systems, especially when the nodes use wireless technology and are powered by batteries.

The crypto layer of communication has not yet been discussed. There are two standard solutions for encrypting CoAP communications: IPsec and DTLS. IPsec was used for the work presented in this thesis; the main reason for this choice was the performance of IPsec implementations compared with that of DTLS implementations four years ago (as shown by De Rubertis et al.[63]). At present, the performances of both are similar, and thus, the work presented in this thesis could be repeated using DTLS instead of IPsec. The differences between the two have been discussed by many authors, such as Hennebert et al.[33] and Alghamdi et al.[64].

The presented methods of authentication and authorization have been successfully tested in several IoT applications as part of the Arrowhead project as well as in the IPSO Challenge 2015. The results presented in this thesis prove the efficient of the proposed method as a fine-grained access control mechanism; it can be used to control access at the service and method levels and can even provide control with regard to a parameter of the CoAP URI. Concerning overheads, the method does not increase the power consumption by more than 13.3% in the worst case (a CoAP message without a payload). Thus, it is an efficient access control method.

6.1 Conclusions

The work presented in this thesis aims to advance the state of the art in Industrial IoT-WSNs, and it represents a step forward in the implementation and expansion of IoT technology in the industrial world. The approach that is presented in this thesis can be used as a guide for creating new configuration, security, access control, and management mechanisms and for improving industrial technology.

This thesis answers the main research questions presented in the Introduction (Chapter 1):

1. *Is it feasible to use IoT-SOA technology in WSANs for industrial applications?*

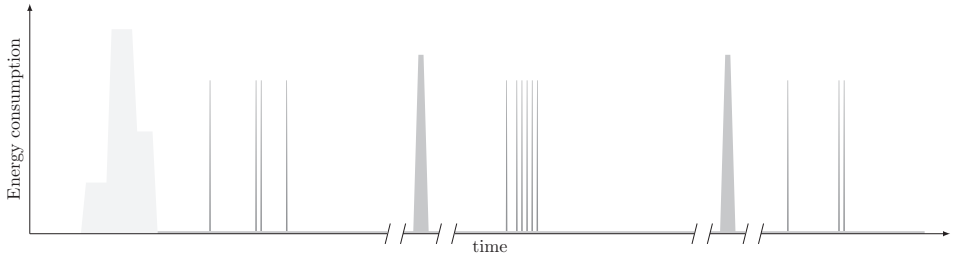


Figure 6.3: Energy consumption profile of a typical IoT device

The results of this thesis show that with SOA approach it is feasible, it requires more sophisticated techniques than WSANs, which increases the computational overhead. But this complexity can reduce the average power consumption and improve the data acquisition, with the implementation of smart resources i.e. services. Therefore, an IoT-SOA framework can be more efficient than today's WSAN.

1.1. *Which are the benefits of adding IoT technology to Industrial WSANs?*

The use of IoT technology maximizes interoperability, enables machine-to-machine (M2M) communications, makes systems easier to upgrade, allows zero-configuration networking, and increases the ease of maintaining and replacing system components.

2. *How can access to exposed IoT nodes be protected and controlled while maintaining performance?*

The Authentication and Authorization solution presented in this thesis enables fine-grained access control with very low penalties in terms of performance and power consumption. This solution, based on the use of tickets, can be adapted for use with standard solutions such as RADIUS and DIAMETER.

3. *How can zero-configuration operation be achieved for an IoT node?*

In this thesis, services such as bootstrapping, authentication, authorization, and configuration are used to demonstrate how a node can receive a customized configuration from scratch. Hence, an optimized and specific configuration is realized for each node. This thesis also provides a route toward the creation of dynamic configurations to adapt the behavior of each node to its current context.

6.2 Future work

The framework presented in this thesis demonstrates the feasibility of using IoT-SOA technology in industrial applications. Therefore, the work presented herein can be used as a basis for further research. There are three different topics related to the Industrial IoT concept that must be addressed in future work: **efficiency**, **scalability** and **Quality of Service** (QoS).

Efficiency has already been considered as part of this thesis, but there are many additional aspects that could not be addressed because of time limitations, including issues that emerged as a direct result of this research. From the charts presented in Chapter 4, is easy to recognize that IKEv2 is the most inefficient component of the IoT system, reaching levels of consumption one hundred times higher than that of 500-byte encrypted communication. Hence, a new, more efficient Internet Key Exchange mechanism is needed for IoT applications.

Another area for improvement is the optimization of the wireless communications at the low-level protocols, to synchronize the radio and sleep cycles of the microcontrollers and minimize the time that each device is awake waiting for radio beacons.

The configuration process could be improved by allowing it to be performed completely dynamically while adapting the behaviors of the sensors and actuators to their current contexts, e.g., changing sampling rates, temporarily shutting off sensors or actuators, or increasing sleep cycles to maximize battery life or enhance performance.

The most recent generation of microcontrollers includes specific hardware for processing certificates; therefore, the use of micro-certificates instead of tickets could be an alternative to enable better protection in access control.

Industrial process monitoring requires the use of a large number of nodes, for which **scalability** is critical. During this thesis work, all tests and implementations were performed with only a limited number of devices; therefore, the presented framework still needs to be tested and proved for application in massive networks.

This thesis work is based on a Service Oriented Architecture, and one parameter that is widely used in modern SOA frameworks is the **Quality of Service** (QoS). For commercial applications or critical services, this feature is particularly important, e.g., to guarantee that a firmware updating service has sufficient bandwidth or priority to be completed as soon as possible or to prioritize alert services ahead of data services.

References

[1] H. Silverstein, "Ceasar, sosus, and submarines: Economic and institutional implications of asw technologies," in *OCEANS '78*, Sept 1978, pp. 406–410.

[2] C. Y. Chong, S. Mori, E. Tse, and R. P. Wishner, "Distributed estimation in distributed sensor networks," in *American Control Conference, 1982*, June 1982, pp. 820–826.

[3] Cisco, "Cisco visual networking index (vni) complete forecast for 2015 to 2020." [Online]. Available: https://newsroom.cisco.com/press-release-content?type=press-release&articleId=1771211

[4] Gartner, "Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015." [Online]. Available: http://www.gartner.com/newsroom/id/3165317

[5] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in *Frontiers of Information Technology (FIT), 2012 10th International Conference on*, Dec 2012, pp. 257–260.

[6] B. Liskov, "Report on workshop on research in experimental computer science," p. 49, 06/1992 1992. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA256874

[7] IEEE Internet of Things, "Towards a definition of the internet of things (iot)," IEEE, Tech. Rep., May 2015. [Online]. Available: http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

[8] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC 7230 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7230.txt

[9] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC 7540 (Proposed Standard), Internet Engineering Task Force, May 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7540.txt

[10] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 3920 (Proposed Standard), Internet Engineering Task Force, Oct. 2004, obsoleted by RFC 6120, updated by RFC 6122. [Online]. Available: http://www.ietf.org/rfc/rfc3920.txt

[11] A. Stanford-Clark and H. L. Truong, "Mqtt for sensor networks (mqtt-sn) protocol specification," IBM Corporation, Tech. Rep., Nov 2013. [Online]. Available: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf

[12] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455 (Proposed Standard), Internet Engineering Task Force, Dec. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6455.txt

[13] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

[14] "CC3000 wifi chip from texas instruments - datasheet." [Online]. Available: http://www.ti.com/lit/ds/symlink/cc3000.pdf

[15] "ESP8266 wifi chip from adafruit - datasheet." [Online]. Available: https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf

[16] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), Internet Engineering Task Force, Sep. 2007, updated by RFCs 6282, 6775. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[17] "TinyOS." [Online]. Available: http://www.tinyos.net

[18] P. Levis, S. Madden, J.Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Sensor Networks," Tech. Rep. [Online]. Available: http://people.eecs.berkeley.edu/~culler/papers/ai-tinyos.pdf

[19] "FreeRTOS OS." [Online]. Available: http://www.freertos.org

[20] "Contiki OS." [Online]. Available: http://www.contiki-os.org

[21] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, Nov 2004, pp. 455–462.

[22] "eLinux - Embedded Linux OS." [Online]. Available: http://www.elinux.org

[23] "OpenWSN OS." [Online]. Available: https://openwsn.atlassian.net/wiki/display/OW/Home

[24] "RIOT OS." [Online]. Available: https://www.riot-os.org

[25] M. Blagojevic, M. Nabi, M. Geilen, T. Basten, T. Hendriks, and M. Steine, "A probabilistic acknowledgment mechanism for wireless sensor networks," in *Networking, Architecture and Storage (NAS), 2011 6th IEEE International Conference on*, July 2011, pp. 63–72.

[26] R. Gonzalez and M. Acosta, "Evaluating the impact of acknowledgment strategies on message delivery rate in wireless sensor networks," in *2010 IEEE Latin-American Conference on Communications*, Sept 2010, pp. 1–6.

[27] S. Moonesamy, "The 'about' URI Scheme," RFC 6694 (Informational), Internet Engineering Task Force, Aug. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6694.txt

[28] Z. Shelby, "Constrained RESTful Environments (CoRE) Link Format," RFC 6690 (Proposed Standard), Internet Engineering Task Force, Aug. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6690.txt

[29] D. C. Bormann, S. Lemay, Z. Technologies, and H. Tschofenig, "A TCP and TLS Transport for the Constrained Application Protocol (CoAP)," Internet Engineering Task Force, Internet-Draft draft-ietf-core-coap-tcp-tls-02, Jun. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-coap-tcp-tls-02

[30] M. Nottingham and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)," RFC 5785 (Proposed Standard), Internet Engineering Task Force, Apr. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5785.txt

[31] T. Narten and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs," RFC 5226 (Best Current Practice), Internet Engineering Task Force, May 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5226.txt

[32] J. Jimenez, M. Kostert, and H. Tschofening, "IPSO Smart Objects," IPSO Alliance, Tech. Rep. [Online]. Available: http://www.ipso-alliance.org/wp-content/uploads/2016/01/ipso-paper.pdf

[33] C. Hennebert and J. D. Santos, "Security protocols and privacy issues into 6lowpan stack: A synthesis," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 384–398, Oct 2014.

[34] S. Kent and K. Seo, "Security Architecture for the Internet Protocol," RFC 4301 (Proposed Standard), Internet Engineering Task Force, Dec. 2005, updated by RFCs 6040, 7619. [Online]. Available: http://www.ietf.org/rfc/rfc4301.txt

[35] P. Hoffman, "Cryptographic Suites for IPsec," RFC 4308 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4308.txt

[36] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," RFC 6071 (Informational), Internet Engineering Task Force, Feb. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6071.txt

[37] S. Raza, D. Trabalza, and T. Voigt, "6lowpan compressed dtls for coap," in *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*, May 2012, pp. 287–289.

[38] T. Fossati and H. Tschofenig, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things," RFC 7925, Jul. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7925.txt

[39] P. Puñal, J. Eliasson, and J. Delsing, "An authentication and access control framework for coap-based internet of things," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, Oct 2014, pp. 5293–5299.

[40] J. Delsing, Ed., *Arrowhead Framework: IoT Automation, Devices, and Maintenance*. CRC Press, 12 2016. [Online]. Available: http://amazon.com/o/ASIN/1498756751/

[41] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 7159 (Proposed Standard), Internet Engineering Task Force, Mar. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7159.txt

[42] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," RFC 7049 (Proposed Standard), Internet Engineering Task Force, Oct. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7049.txt

[43] "Arrowhead project." [Online]. Available: http://www.arrowhead.eu

[44] H. Derhamy, J. Eliasson, J. Delsing, P. Varga, and P. Puñal, *Translation Error Handling for Multi-Protocol SOA Systems*, ser. I E E E International Conference on Emerging Technologies and Factory Automation. Proceedings. IEEE, 2015.

[45] "Libcoap 4.1.1." [Online]. Available: https://github.com/obgm/libcoap

[46] "Copper 1.0.0." [Online]. Available: https://github.com/mkovatsc/Copper

[47] "Erbium." [Online]. Available: https://github.com/contiki-os/contiki/tree/master/apps/er-coap

[48] "Californium." [Online]. Available: https://github.com/eclipse/californium

[49] J. Schaad, "CBOR Object Signing and Encryption (COSE)," Internet Engineering Task Force, Internet-Draft draft-ietf-cose-msg-17, Aug. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-cose-msg-17

[50] E. Wahlstroem, G. Selander, L. Seitz, H. Tschofenig, and S. Erdtman, "Authentication and Authorization for Constrained Environments (ACE)," Internet Engineering Task Force, Internet-Draft draft-ietf-ace-oauth-authz-02, Jun. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-02

[51] G. Selander, J. Mattsson, L. Seitz, and F. Palombini, "Object Security of CoAP (OSCOAP)," Internet Engineering Task Force, Internet-Draft draft-selander-ace-object-security-05, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-selander-ace-object-security-05

[52] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, *A survey of commercial frameworks for the Internet of Things*, ser. I E E E International Conference on Emerging Technologies and Factory Automation. Proceedings. IEEE, 2015.

[53] Open Mobile Alliance (OMA), "LWM2M OMA - Bootstrap Interface." [Online]. Available: http://dev_devtoolkit.openmobilealliance.org/IoT/LWM2M10/doc/TS/index.html#!Documents/bootstrapinterface.htm

[54] Open Mobile Alliance (OMA), "Lightweight Machine to Machine (LWM2M) v1.0." [Online]. Available: http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0

[55] "Eclipse Foundation - Leshan LWM2M." [Online]. Available: http://www.eclipse.org/leshan/

[56] Intel, "Wakaama LWM2M." [Online]. Available: https://projects.eclipse.org/projects/technology.wakaama

[57] IPSO Alliance, "IPSO Challenge 2015." [Online]. Available: http://challenge.ipso-alliance.org/ipso-challenge-2015

[58] J. Delsing, K. Hyyppa, and T. Isaksson, "The ip-meter, design concept and example implementation of an internet enabled power line quality meter," in *Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE*, vol. 2, 2000, pp. 657–660 vol.2.

[59] M. Sveda and R. Vrba, "An integrated framework for sensor-based embedded systems," in *Engineering of Computer-Based Systems, 2002. Proceedings. Ninth Annual IEEE International Conference and Workshop on the*, 2002, pp. 195–202.

[60] "Industrial Internet Consortium (IIC)." [Online]. Available: http://www.iiconsortium.org

[61] "IPSO Alliance." [Online]. Available: http://www.ipso-alliance.org

[62] "Open Mobile Alliance (OMA)." [Online]. Available: http://openmobilealliance.org

[63] A. D. Rubertis, L. Mainetti, V. Mighali, L. Patrono, I. Sergi, M. L. Stefanizzi, and S. Pascali, "Performance evaluation of end-to-end security protocols in an internet of things," in *Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on*, Sept 2013, pp. 1–6.

[64] T. A. Alghamdi, A. Lasebae, and M. Aiash, "Security analysis of the constrained application protocol in the internet of things," in *Second International Conference on Future Generation Communication Technologies (FGCT 2013)*, Nov 2013, pp. 163–168.

Part II

Paper A

A Feasibility Study of SOA-enabled Networked Rock Bolts

Authors:
Jens Eliasson, Pablo Puñal Pereira, Henrik Mäkitaavola, Jerker Delsing and Joakim Nilsson

Reformatted version of paper originally published in:
Conference paper, IEEE EFTA, 2014.

A Feasibility Study of SOA-enabled Networked Rock Bolts

Jens Eliasson, Pablo Puñal Pereira, Henrik Mäkitaavola, Jerker Delsing and Joakim Nilsson

Abstract

The use of rock bolts in the mining industry is a widely used approach for increasing mine stability. However, when compared to the automation industry, where the use of sensors and real-time monitoring of processes have evolved rapidly, the use rocIPseck bolts have not changed a lot during the last 100 years. What is missing are technologies for keeping installed rock bolts under real-time and online monitoring. One problem is that rock bolts can become damaged by seismic activities or movements within the rock, and thus lose their load bearing capacity. If that happens, the outer shell of a tunnel's walls or ceiling can collapse, with disaster as a result. Therefore, there is a clear need for online and real-time monitoring solutions for strain and thereby stress, as well as seismic activity.

In this paper, the current state of art in research around intelligent rock bolts is presented. An intelligent rock bolt is the combination of a traditional rock bolt with an *Internet of Things* device, i.e. a rock bolt with embedded sensors, actuators, processing capabilities and wireless communication. In the proposed architecture, every rock bolt has its own IPv6 address and can establish a wireless mesh network in an ad-hoc manner. By measuring strain and seismic activity and exposing the sensors in the form of services, large gains in terms of safety and efficiently can be achieved. A number of mining related activities such as stress on the rock bolt can be detected, falling rocks and the presence of mobile machinery can be observed. Since the network is based on standard communication protocols such as IPv6, it is vital to add security mechanisms to prevent eavesdropping and tampering of data traffic.

By utilizing the real-time monitoring capabilities of a network of Internet-connected intelligent rock bolt, it is possible to drastically improve monitoring of mining activities and thereby providing workers with a safer working environment.

1 Background and Related work

Mine activity monitoring is today mostly made with geo-phones, still the most sensitive devices to detect earth movement [1]. In mines geo-phones are now interconnected and used to gather micro seismic data which is further analyzed to provide safety predictions [2, 3].

The mining industry have over time initiated a number of smaller projects to test the function of rock bolts. This has lead to some functional rock-bolt monitoring specifications [4, 5]. Some of the most important are:

- measure static and dynamic rock bolt load of <300kN.
- Dynamics to be captured are <100 Hz, thus a sampling rate of 1kHz will be sufficient.
- true load measured with an accuracy 2 %.
- not sensitive to uneven loading on the bolt plate.
- a cable free system.
- continuous load sampling over time (with the possibility to set sampling intervals).
- life time without changing power supply >12 month (using a battery).

There are several approaches to make one shot testing of rock bolts. Ultrasound is one common approach to measured bolt load through speed of sound measurements. We do find several scientific papers and several patents in this field. One example is [6]. Some suppliers of ultrasound measurement technology for bolt load measurements are:

- USM-3 by Norbar [7]
- Hevii - US bolt load technology [8].
- Boltscope-II by Hydratight [9]

This ultrasound technology has the potential to provide the most information on the changes in the rock bolt. The technology is still rather young and much development can be expected in the future. The major drawback is the price tag. An attractive approach for strain gauges sensing applied to rock bolt load measurements is the MMT prototype found with Hitec corporation [10]. They exhibit and custom device drilled into to the head of the bolt. The major draw back is the sensitivity to non-axial loads. To our understanding the development has been halted.

The process automation industry, where the use of sensors, actuators, distributed control systems and other technologies are widely used, have responded well to the new possibilities that networked embedded devices, e.g. Internet of Things (IoT) and Cyber-physical systems, (CPS) can offer [11]. The use of IP-based networked sensor and actuator devices with vertical integration into traditional industry systems is currently being investigated in some of Europe's largest automation projects such as the R&D projects FP7 IMC-AESOP [12] and Artemis Arrowhead [13].

The COBS project [14] at Luleå University of Technology aims at developing smart conveyor belt rollers for the mining industry and logistics. By equipping a conveyor belt roller with a wireless sensor node and additional sensors, the roller is able to monitor

itself and thereby sending alarms when for example a ball bearing is getting too warm which is an indication of a ball bearing damage. The higher level system is used to alert operators of any anomalies or alarms and assist in scheduling maintenance and reduces cost from less unexpected downtime.

2 Architecture

This section outlines the core architecture of the intelligent rock bolt with its sensing and networking capabilities, the support for communication and as well as security.

Intelligent rock bolt

The current proposed design of the intelligent rock bolt is composed of several individual components. The base is a standard rock bolt, which is equipped with measurement electronics. The core of the electronic system is the Mulle platform from Eistec AB [15]. The Mulle is a low-power sensor node designed for Internet of Things applications. The current Mulle features a 16-bit microcontroller, analog and digital inputs and outputs, an 868 MHz IEEE 802.15.4 transceiver, several memories and power management circuits. To the Mulle is an interface board for the strain and vibration sensors connected, which is described in more detail in the next section. The Mulle runs the Contiki operating system from Dunkels et al. [16].

Electronics and sensors

The measurement system consists of a strain sensor and an accelerometer. The accelerometer is mounted on a printed circuit board (PCB) while the strain sensor is external to the PCB, i.e. mounted inside the rock bolt's head. Both these sensors produce a voltage which is sampled by two 24-bit analog-to-digital converters (ADCs). These ADCs are mounted on the PCB which also hosts a connector that allows the ADCs to communicate with the Mulle using a high-speed SPI port.

The measurement board hosts a high-density connector for interfacing the Mulle. It also features some LEDs for development use, power supply, etc. Figure 1 shows the circuit of the vibration sensor system with Mulle platform.

The two sensors, accelerometer and strain, have been chosen in order for the rock bolt to be able the two most important factors for mine stability. Seismic activity will cause vibrations in the rock, and forces lead to tensions in the rock which when released can result in small earthquakes. These quakes can in worst case result in the collapse of tunnels, or even portions of the mine.

Internet of Things networking stack

The current communication stack is based on previous work from several research projects. Other research projects that have been developing the Mulle architecture are EU FP7

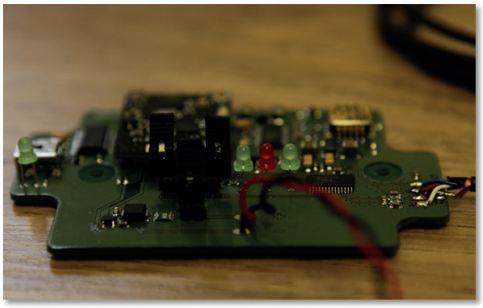


Figure 1: Sensor node electronics

IMC-AESOP and I2Mine and Artemis Arrowhead. The current version of the Mulle's communication stack is based on the IEEE 802.15.4 standard, and uses IPv6 and RPL over 6LoWPAN. Data is normally transmitted using SenML encoded using XML (with optional EXI-compression by the EXIP parser [17]) over CoAP. Figure 2 shows the Mulle's communication stack.

The software side of the strain and acceleration measurements were implemented as CoAP [18] services. A CoAP service is easily accessible through a web browser that supports it. This provides simplicity in monitoring and configuring the rock bolts as it can be done through a standard web interface over the Internet. CoAP is a protocol designed to be used on resource-constrained, low power electronic devices.

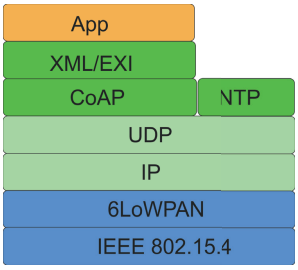


Figure 2: Rock bolt communication stack

Since Contiki, which is used on the Mulle platform and hence rock bolts, runs RPL [19] it is possible to create mesh networks, i.e. with multi-hop support. The mesh networking support has been experimentally verified on the rock bolts and the performance of RPL has been investigated by Potsch et al. in [20]. Time synchronization is performed using the NTP protocol. Wireless re-programming of Mulle devices is handled by a custom written CoAP service. The Mulles are connected to existing networks (i.e. Ethernet) using a BeagleBone based gateway, also equipped with an IEEE 802.15.4 radio

transceiver. The gateway host several services, such as RPL, NTP, and a number of CoAP services. The gateway is connected to its back-end system using an encrypted VPN solution. This ensures that sensor data is transmitted from a Mulle to database servers over encrypted channels only.

The use of a Ultra-wide Band (UWB) chip from Decawave has also been investigated. Preliminary results indicates that UWB is a viable solution for environments with severe multi-path problems. This will be studied further as well. The use of UWB in combination with distributed event detection and pattern recognition, as proposed in [21], could provide one solution for performing detection and classification of mining related activities.

Measurement software

For the strain measurements, a CoAP service was created that can retrieve a strain sample at any time. Also, a threshold value can be set that allows the user of the service to be notified when a measurement is collected that has changed a specified amount from when the threshold was set. This is realized through CoAP's *Observe*-mechanism. Moreover, the sampling interval of the notifying service can be set through another CoAP service.

For the acceleration measurements, a CoAP service was created that controls the sensor to store a given amount of acceleration samples to the internal flash memory of the Mulle. When the logging is complete, the samples can be fetched through another service. Acceleration measurements are done in this way as acceleration data must be sampled at a much higher data rate than the available bandwidth of the wireless network.

Communication security

A high level of security usually means complex methods and algorithms, therefore more CPU time and more energy consumption. For this reason on low power systems (networks) the security design is a critical task. Nowadays one of the most extended systems over 6LoWPAN is IPsec that is an extension of the IP protocol that adds security to IP and higher layers. It was developed for the "new" IPv6 standard and was later adopted to include IPv4 as well.

IPsec has two different protocols, AH and ESP, to secure the authentication, integrity and confidentiality on communication [22]. IPsec can protect completely the IP datagram (Tunneling Mode [23]) or only the protocols on higher layers (Transport Mode). In Tunneling mode the IP datagram is encapsulated completely inside a new IP datagram that uses IPsec (the final IP of the datagram could even be different). In Transport mode, IPsec only manages the content of the IP datagram, adding the IPsec header between the original IP header and the header of higher layers, shown in Figure 3.

To protect the integrity of IP datagrams, the IPsec protocol uses authentication message codes based on hash, HMAC (Hash Message Authentication Codes). To protect the confidentiality of IP datagrams, IPsec uses standard algorithms of symmetric cipher (in

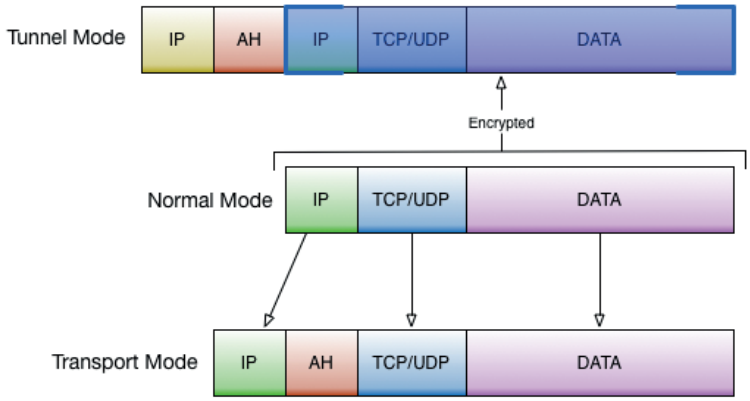


Figure 3: IPsec encryption and authentication

our case using AES-128, but could work with any other cipher such as AES-256). In order to protect against DoS (Denial of Service) attacks, IPsec uses sliding windows. Each packet receives a sequence number and only is accepted by the receiver if the number of packet is inside this window or next. Any previous packets are immediately discarded. This is an efficient protection mechanism against attacks with message repetition, especially when the attacker is using sniffed original packets to resend.

The current IPsec version, which is based based on the compressed IPsec design developed by Raza [24], is under development and does not support directional keys, this means that IPsec must use a different secret key for each direction of the communication with the same client/server, but this implementation uses the same (reducing the security level). One big step forward is the implementation of IKE - Internet Key Exchange - that is now work in progress. With IKE IPsec could change and choose the correct secret key for each communication. The use of DTLS encryption for CoAP would further increase the communication security [25].

3 Performed experiments

This section presents the tests and experiments that have been performed, and gives an overview of all tests' setup in terms of hardware and software.

Test overview

In order to investigate the performance and feasibility of the rock bolt design, several tests were performed. The first set of tests was performed indoors in a controlled laboratory environment. When it was confirmed that the sensing electronics were functioning as planned as well the integration between the electronics and the rock bolt the next step

Figure 4: Intelligent rock bolt installed in mine

was taken by performing tests using four rock bolts in an active mine. The mine test system was comprised of a total of four intelligent rock bolts, two Linux-based BeagleBone devices, interface cables, and power supplies.

Laboratory test setup

In the initial laboratory test, the strain sensor was mounted on a device constructed to simulate strain. This device was fastened to a desk and different torques were applied at the nut of the device to simulate the strain of a rock bolt. The accelerometer was also tested in a lab. setup where vibrations were measured as well. All measurement data were transmitted wirelessly using a CoAP service over a 6LoWPAN network and stored to file for later processing and visualization. A Java implementation of CoAP, Californium [26], was used to retrieve all measurements.

Mine installation

Figure 4 shows how an intelligent rock bolt is installed in a mine tunnel. The rock bolt itself is around three meters long, and the head with the Mulle and sensor interface board inside the grey plastic box. The strain sensor is located inside the stainless steel head. The two cables, one for power and one for data, are connected to the data logger and power supply, respectively. This installation is a prototype device, and not of production quality. In practice, the electronics must be protected in a better manner in order to withstand the harsh environment inside an active mine but for prototyping and testing this approach was sufficient.

Performed tests

When all four rock bolts were installed and equipped with the electronics for measuring strain and vibration, the two BeagleBone-based data loggers were time synchronized

using NTP over an 100 Mbit/s Ethernet cable. Each logger stored data from one pair of rock bolts installed on the same tunnel wall. This procedure was performed during a total of three days in order to collect as much data as possible.

Several different experiments were conducted in the mine in order to collect as much relevant data as possible. The performed experiments were:

Strain

The strain was recorded on all four rock bolts.

Tunnel wall vibration

A metal object was used as a hammer to hit the tunnel wall and the vibrations were recorded.

Top hammer drill rig

The vibrations generated by a production top hammer drill rig some 30 meters away for the rock bolts were recorded.

Falling rocks

A rock was dropped in order to simulate the event of rocks falling from a tunnel's ceiling.

Vehicle detection

A car was driven by the rock bolts and the generated vibrations were recorded.

4 Results

This section presents results from the collected data from the laboratory experiments performed in August 2013 as well as from the Kittilä mine experiments performed in October 2013. All data processing and plots were performed using Matlab. For the accelerometer, the Z-axis has been used which corresponds to vibrations along the length of the rock bolt.

Note that a 24-bit ADC has been used, together with an accelerometer that can measure static acceleration (i.e. the gravity components is visible in the signal). A DC-blocking filter could be used to remove all offset. The accelerometer will see a different offset depending on the angle the rock bolt is installed with.

Laboratory strain measurements

To test the linearity of the strain sensor, different torques were applied to the strain simulation device. Four different boards and sensors were tested, labeled 2, 3, 4 and 5

and the strain output as a function of applied torque were recorded. The measurements were taken at torques of 0, 40, 50, 70 and 80 Nm. 10 measurements were taken for each value of torque and the mean and standard deviation, respectively, of the measurements were then plotted. The resulting plots are shown in Figure 5. It can also be seen that the strain measurement sensors have good linearity properties.

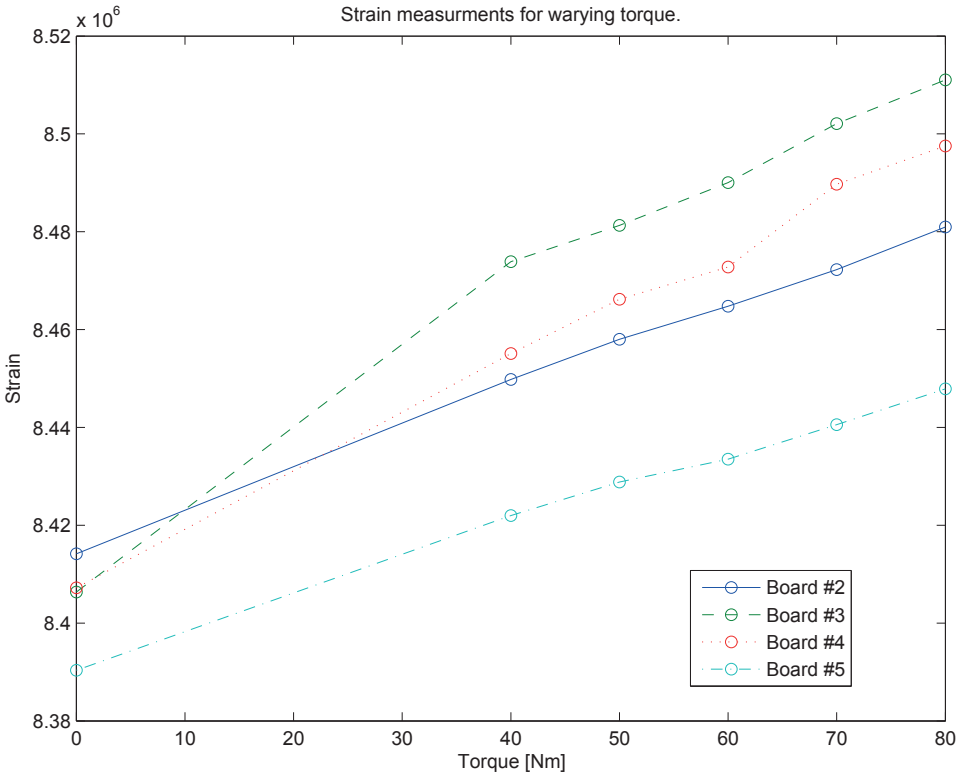


Figure 5: Strain measurements for varying torque

Steel rod

In this first mine-based test, a rock bolt rod was used as a hammer to hit the wall near one of the installed rock bolts. This was repeated eight times in order to get a better understanding of which type of signal amplitudes that could be expected from a very strong source of vibration in close proximity of a rock bolt. The vibration data collected is shown in Fig. 6.

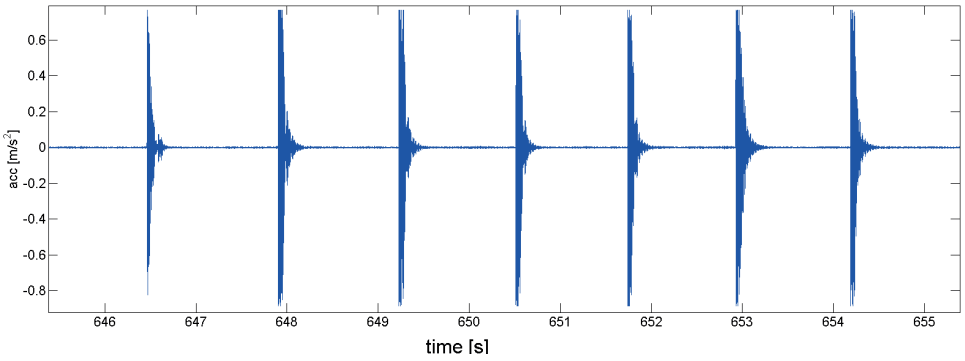


Figure 6: Steel rod hit on wall

It is clearly shown in the wave form when the rod hits the tunnel wall and generates a vibration pattern. This type of amplitudes, or even higher, would probably also be generated if a mobile machine would drive too close to a wall and brush against it. The rock bolts can therefore be used for anomaly detection around vehicles.

Drill test

The second mine-based test was performed in order to investigate if a rock bolt's vibration sensor can be used to detect mining-related activities such as drilling. A mobile top hammer production drill rig, located approximately 25-30 meters from the installed rock bolts, was used as a vibration source.

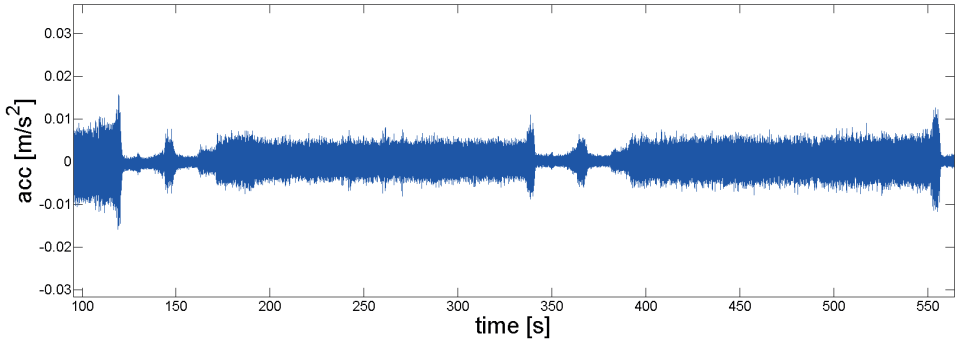


Figure 7: Drilling detection

It is clearly seen in the signal at 140 and 360 seconds in Figure 7 when the drilling machine drills, takes a short pause to insert a new rock tool, and starts drilling again.

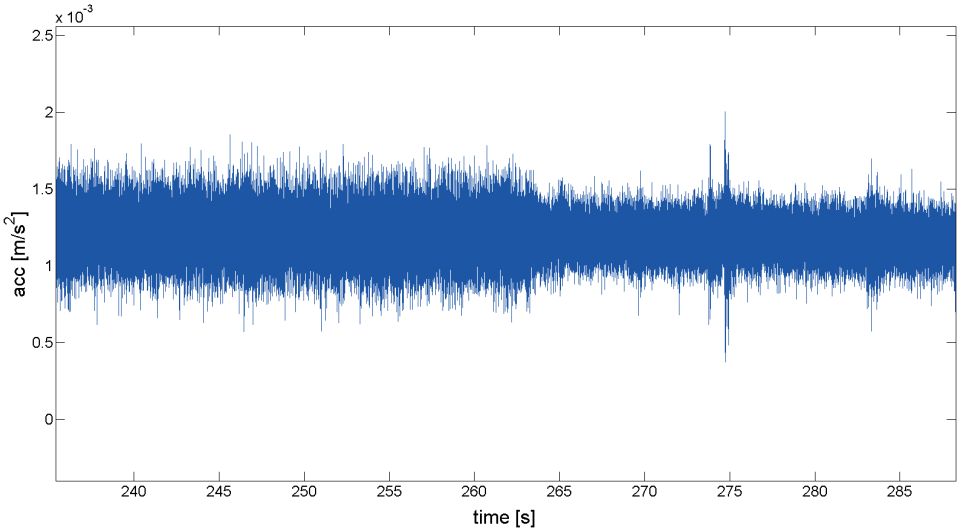


Figure 8: Vehicle detection

This indicates that a rock bolt can be used to detect drilling activity in close proximity, and even count how many drill holes that have been drilled.

Vehicle detection

One important feature that can be used to localizing vehicles is the ability for a rock bolt to monitor the presence of close by vehicles. This can be used for fine grain localization of mobile machinery such as cars, trucks, etc. Figure 8 shows the raw and unfiltered vibration signal from one rock bolt when a car was used in the vicinity. At 265 seconds into the signal, the car's engine was turned off which is clearly visible as a sharp drop of signal amplitude. The plotted signal is the raw output from the sensor, without any applied signal processing, such as filtering. By applying filtering techniques, the presence of a nearby vehicle could be detected [27].

How larger vehicles, such as loaders and trucks, will be observed is currently unknown. However, previous work performed within the iRoad project indicates that heavier vehicles generate higher amplitude levels, as shown by Hostettler et al. [28].

Falling rock detection

Rocks falling from a tunnel's ceiling are a clear indication of pending danger. When this occurs, a collapse of the tunnel could happen, or lead to larger and heavier rocks falling which could result in damage to vehicles and machinery as well as injuries on workers.

In order to see if the rock bolts could detect falling rocks, a simple experiment was

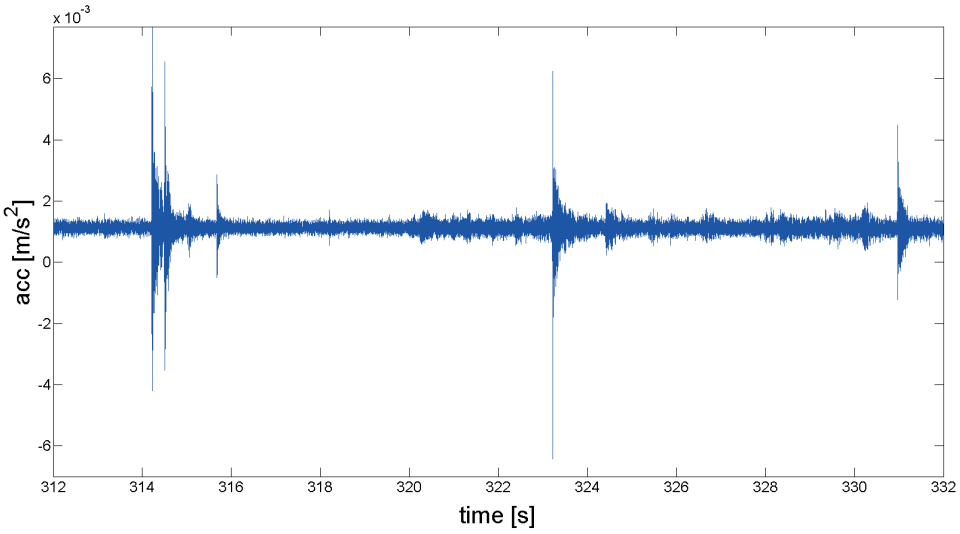


Figure 9: Falling rocks detection

performed by dropping a loose rock weighing approximately 3-4 kg from around two meters height down on the tunnel's floor around 1.5 meters from the rock bolt. At 314, 323 and 332 seconds in the signal shown in Fig. 9, three spikes are clearly visible. This indicates that an intelligent rock bolt can be used to detect falling rocks. When this feature is combined with the wireless communication capabilities, this could be used for a near real-time alarm system.

Strain test

A strain gauge sensor can be used to monitor stress in pillars, tunnels walls and ceilings. Strain can be a good indication of how strong forces that are affecting a volume of rock. The strain sensor is currently mounted at the rockbolt's head, however this will severely limit the amount of strain that can be detected by the sensor due to the fact that the shotcrete will limit the forces to propagate along the rockbolt.

The output from the strain sensor, shown in Figure 10, also concludes this. For better strain gauge sensor performance, the strain sensor must be re-designed. This is considered as future work.

5 Future work

Some of the more prominent features that need more work are: efficient signal processing of captured data, sufficient low-power operation on sensing, processing and communication, and integration with back-end mine monitoring systems. Performance of the used

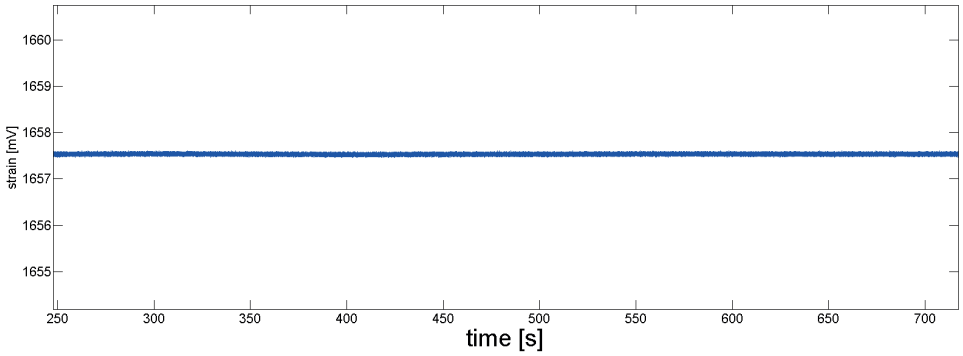


Figure 10: Strain gauge sensor output

sensors also needs more testing, especially the strain gauge which is challenging to observe in a mine due to the very high time constants and slow change rates. The mounting of the strain gauge also needs more investigation. Another key issue that needs more research is how the use of traditional Internet of Things protocols and technologies, which were originally designed for very low data-rate transmission, with no or low real-time requirement will behave when larger amounts of data must be streamed, i.e. from vibration sensors, with high requirements on low-latency transmission. The impact of scalability and security issues must also be investigated further. An interesting approach for self-learning methods for signal processing proposed in [29] would be interesting to evaluate for rock bolt usage. The fourth issue to explore is how strain and/or stress information and vibration data can be successfully integrated in today's monitoring systems.

In order to secure the communication and SOA model, the IPsec protocol must be enhanced with a key exchange mechanism like IKEv2 [30]. A system for fine-grain access control like Radius is also needed to be able to allow or deny specific clients to access services.

6 Conclusion

The use of rock bolts in the mining industry is a well known approach for increasing stability in for example tunnels, and thereby increasing safety for workers. However, what has been missing is a method of keeping installed rock bolts under constant monitoring. When compared to the process automation industry, where the use of sensors and SCADA system is a commonly used, rock bolt monitoring has not been especially improved.

This paper has presented a novel method for rock bolt monitoring, and the design of an intelligent rock bolt architecture with on-board sensing, processing and communication capabilities. The intelligent rock bolt, which comprise of a standard rock bolt, sensors and actuators, signal processing, data storage and wireless communication, can monitor itself and send alarms when seismic activities are detected, or when different mining

activities are observed. Since security is highly important, the rock bolts have also been equipped with a security framework designed to provide tamper-free and secure communication. The rock bolt can detect, at least but not limited to, the following mining related activities:

- Deviation of strain on rock bolts
- Drilling
- Usage of mining machinery
- Falling rocks

This paper has also presented concrete test results from a mine-based field test using a low-cost intelligent rock bolt as the measurement device. Results from the tests indicates that a traditional rock bolt can be equipped with sensors, and that the sensors are capable of detecting mining-related activities.

Test results also show that successful integration between low-power electronics and a standard rock bolt is feasible. When all results presented in this paper are summarized, it is clear that intelligent rock bolts can be used within the mining industry to produce a better and safer working environment.

7 Acknowledgment

The authors would like to express their gratitude toward Agnico Eagle's Kittilä mine for allowing us to perform our experiments there. We would especially like to thank Antti Pyy and André van Wageningen for their assistance with the preparations and support during the field tests. The authors would also like to thank Mikael Larsmark for his contributions to the development of hardware and software for the intelligent rock bolt. The authors would also like to thank Fredrik Sandin for fruitful discussions regarding signal processing and data analysis.

We would also like to express our gratitude towards our partners within the I2Mine and Arrowhead projects, and the European commission and Artemis for funding. We would also like to thank Gluetech AB for assisting us with the strain gauge sensors and Eistec AB for their support with the Mulle platform.

References

[1] C. E. Krohn, "Geophone ground coupling," *Journal of GeoPhysics*, vol. 49, pp. 722–731, 1984.

[2] A. T. Kunnath and M. V. Ramesh, "Integrating geophone network to real-time wireless sensor network system for landslide detection," in *Proc. First International Conference on Sensor Device Technologies and Applications*, IEEE. IEEE, 2010, pp. 167–171.

[3] B. L. F. Daku and J. Salt, "Directional performance of an algorithm used to locate microseismic events in underground mines," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, Nov 2011, pp. 2198–2201.

[4] G. Bäckblom, "Project plan migs wp3 monitoring of rock bolt load in underground openings," RTC, Tech. Rep., Dec. 2008.

[5] J. Delsing, "Migs wp3 monitoring of bolt load - review of sensor technology for bolt load measurements," EISLAB, Luleå University of Technology,, SE-971 87 Luleå, Sweden, Tech. Rep., 2009.

[6] O. G. et.al., "Us patent 4,402,222, bolt load determining apparatus," Tech. Rep., Sept. 1983.

[7] [Online]. Available: http://www.norbar.com/

[8] [Online]. Available: http://www.heviitech.com/Hevii_UT.html

[9] [Online]. Available: http://www.hydratight.com/en/products/ultrasonics/boltscope\-ii

[10] [Online]. Available: http://www.globalspec.com/Supplier/CustomProductDetail/HITEC?Comp=10\&QID=13910091\&ExhibitId=42329

[11] S. Karnouskos, O. Baecker, L. de Souza, and P. Spiess, "Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure," in *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, Sept 2007, pp. 293–300.

[12] "IMC-AESOP - Architecture for Service-Oriented Process Monitoring and Control," Feb. 2013. [Online]. Available: http://www.imc-aesop.eu

[13] "Arrowhead - Enable collaborative automation by networked embedded devices." Feb. 2013. [Online]. Available: http://www.arrowhead.eu/

[14] J. Eliasson, R. Kyusakov, and P.-E. Martinsson, "An Internet of Things approach for intelligent monitoring of conveyor belt rollers," in *International Conference on Condition Monitoring and Machinery Failure Prevention Technologies - CM2013*, June 2013.

[15] "Eistec AB," Feb. 2013. [Online]. Available: http://www.eistec.se/

[16] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, Nov 2004, pp. 455–462.

[17] R. Kyusakov, H. Makitaavola, J. Delsing, and J. Eliasson, "Efficient xml interchange in factory automation systems," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, Nov 2011, pp. 4478–4483.

[18] C. Bormann, A. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *Internet Computing, IEEE*, vol. 16, no. 2, pp. 62–67, March 2012.

[19] J. Tripathi, J. De Oliveira, and J. P. Vasseur, "A performance evaluation study of rpl: Routing protocol for low power and lossy networks," in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, March 2010, pp. 1–6.

[20] T. Potsch, K. Kuladinithi, M. Becker, P. Trenkamp, and C. Goerg, "Performance evaluation of coap using rpl and lpl in tinyos," in *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, May 2012, pp. 1–5.

[21] M. Baqer and A. Khan, "Energy-efficient pattern recognition approach for wireless sensor networks," in *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, Dec 2007, pp. 509–514.

[22] V. Manral, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," RFC 4835 (Proposed Standard), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4835.txt

[23] S. Kent, "IP Encapsulating Security Payload (ESP)," RFC 4303 (Proposed Standard), Internet Engineering Task Force, dec 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4303.txt

[24] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig, "Securing communication in 6lowpan with compressed ipsec," in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, June 2011, pp. 1–8.

[25] S. Raza, D. Trabalza, and T. Voigt, "6lowpan compressed dtls for coap," in *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, May 2012, pp. 287–289.

[26] M. Kovatsch, S. Mayer, and B. Ostermaier, "Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, July 2012, pp. 751–756.

[27] R. Hostettler, W. Birk, and M. Nordenvaad, "Feasibility of road vibrations-based vehicle property sensing," *Intelligent Transport Systems, IET*, vol. 4, no. 4, pp. 356–364, December 2010.

[28] ——, "Extended kalman filter for vehicle tracking using road surface vibration measurements," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, Dec 2012, pp. 5643–5648.

[29] S. del Campo, K. Albertsson, J. Nilsson, J. Eliasson, and F. Sandin, "Fpga prototype of machine learning analog-to-feature converter for event-based succinct representation of signals," in *Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on*, Sept 2013, pp. 1–6. [Online]. Available: http://pure.ltu.se/portal/files/43648751/mlsp2013.pdf

[30] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 5996 (Proposed Standard), Internet Engineering Task Force, Sep. 2010, updated by RFCs 5998, 6989. [Online]. Available: http://www.ietf.org/rfc/rfc5996.txt

Paper B

EXIP: A Framework for Embedded Web Development

Authors:
Rumen Kyusakov, Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Reformatted version of paper accepted for publication in:
Journal paper, ACM Transactions on the Web, 2014.

EXIP: A Framework for Embedded Web Development

Rumen Kyusakov, Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Abstract

Developing and deploying Web applications on networked embedded devices is often seen as a way to reduce the development cost and time to market for new target platforms. However, the size of the messages and the processing requirements of today's Web protocols, such as HTTP and XML, are challenging for the most resource-constrained class of devices that could also benefit from Web connectivity.

New Web protocols using binary representations have been proposed for addressing this issue. Constrained Application Protocol (CoAP) reduces the bandwidth and processing requirements compared to HTTP while preserving the core concepts of the Web architecture. Similarly, Efficient XML Interchange (EXI) format has been standardized for reducing the size and processing time for XML structured information. Nevertheless, the adoption of these technologies is lagging behind due to lack of support from web browsers and current Web development toolkits.

Motivated by these problems, this article presents the design and implementation techniques for the EXIP framework for embedded Web development. The framework consists of a highly efficient EXI processor, a tool for EXI data binding based on templates, and a CoAP/EXI/XHTML Web page engine. A prototype implementation of the EXI processor is herein presented and evaluated. It can be applied to Web browsers or thin server platforms using XHTML and Web services for supporting human-machine interactions in the Internet of Things.

This article contains four major results: (1) theoretical and practical evaluation of the use of binary protocols for embedded Web programming; (2) a novel method for generation of EXI grammars based on XML Schema definitions; (3) an algorithm for grammar concatenation that produces normalized EXI grammars directly, and hence reduces the number of iterations during grammar generation; (4) an algorithm for efficient representation of possible deviations from the XML schema.

Categories and Subject Descriptors: E.4 [**Coding and information theory**]: Data compaction and compression; H.3.5[**Online Information Services**]: Web-based services

This work is supported by the EU FP7 Project IMC-AESOP and ARTEMIS Innovation Pilot Project Arrowhead.

Author's addresses: R. Kyusakov, P. Puñal, J. Eliasson, and J. Delsing are with the Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå;

General Terms: Performance, Design, Algorithms, Standardization

Additional Key Words and Phrases: Information Exchange, EXI, XML, Data Formats, CoAP, Data Processing, XHTML, Embedded Systems, Internet of Things, Web of Things

1 Introduction

Web technologies are rapidly expanding to networked embedded devices with studies showing that in 2013 there were more Web-connected gadgets than people in the U.S.[1] This process is expected to accelerate due to the increased IPv6 adoption rate and the availability of small-sized, cheap, off-the-shelf hardware that is powerful enough to execute full-featured network stacks. Already now, the number of TCP/IP connected sensor and actuator devices using low-power wireless technologies or even power-line communication is huge. The application areas cover home automation [1], energy management, and industrial process monitoring and control [2].

With the increase in the number of devices, the requirements on their interfaces are also higher. Consumers are demanding "smart" gadgets that are easy and intuitive to deploy, configure, interact with, and integrate with other devices and systems. An example from the home automation domain is a smart thermostat that can communicate with the user's smart phone to display the current temperature in the house along with energy costs as well as control settings. It is becoming more common to equip the traditionally simple sensor and actuator devices with additional diagnostics, logging, and security capabilities. This phenomenon leads to developing more complex embedded applications, which are often required to support Web connectivity for human-machine interfacing. As the code base increases, so are the product cost and time-to-market for new devices. The development and support for different hardware platforms becomes especially challenging, and thus the need for a common development platform based on established and globally adopted standards. The Web development has proved successful in leveraging a set of global standards for unification of the development for front-end tools and applications over a large number of desktop and mobile platforms. In addition, ICT research as argued by [3] suggests that embedded computing will also benefit from Web development platforms.

The trade-off between Web and native applications has been a turning point for development strategies in the mobile market. As discussed by [4], Web applications are cheaper to build, deploy, and maintain, but are often lagging behind in performance and user experience when compared to the native apps. This gap is narrowing, thanks to HTML5 and new Web toolkits such as Argos [5] which provides direct access to devices' capabilities from JavaScript code. While the app stores made the management of native applications much easier and user-friendly, their main drawback remains - supporting

[1] According to data from research firm NPD Group

different platforms often requires substantial rebuild of the code base that needs to be kept up-to-date with new versions of the different operating systems. As Charland et al. conclude, one size does not fit all, and there are use cases when it is better to use one or the other approach. While there are a number of differences between the smart phone and the embedded systems segments, it is possible to draw some similarities and list a number of applications where building Web applications is more beneficial even for resource-constrained hosts when compared to developing proprietary solutions. The simplified use case presented in Section 5, which demonstrates a human-machine interface with a sensor platform, provides an example of such application. In this scenario, the user interface is implemented as dynamic Web application based on CoAP/EXI/XHTML and using the EXIP framework.

The approach of using standard binary protocols for enabling Web connectivity for constrained hosts differs from the most common methods described in the literature. The state-of-the-art solutions to the problems of embedded Web development (e.g., memory, network, and processing constraints) can be classified into two groups. The methods in the first group rely on powerful gateway devices that translate the standard Web protocols to some lightweight messaging framework, and vice versa. An example of this approach is the work by [6], which describes a gateway architecture for providing Web connectivity to highly resource-constrained nodes. The methods in the second group focus on implementing efficient and stripped-down version of the standard text-based Web protocols. High-impact research results based on this method are the techniques for implementing an efficient HTTP server for embedded devices presented by [7] and [8], as well as the small-footprint XML Web service implementation by [9].

Using text-based protocols that rely on simple character encoding such as ASCII, was important requirement in the early days of distributed computing systems. During that time, the ability to debug the interactions between the systems with one's bare hands was crucial to the acceleration of the adoption of the protocols. Nowadays, practically all text editors and development tools support UTF-8 character encoding. The tools also parse the XML documents before printing them to the screen to support syntax highlighting. Proper tool support opens up new possibilities for efficient representation of the information on the wire. The new binary encoding schemes are transparent for the user - if, in any case, the XML documents are parsed before printing them, then it is better to use faster, binary encoding which is easier to process than text-based representation. However, implementing highly optimized binary coding schemes is much more challenging than processing text-based streams. Even more challenging, is the use of such binary processors on resource-constrained embedded devices where the memory footprint and CPU usage are crucial. As an example, a common way to compress the size of an XML document is by indexing frequently used tags and value items. Instead of encoding each occurrence in the stream, the repeated information items are represented by their index. Using more extensive indexing increases the compression, but also makes the memory footprint required to store the indexed information larger. Providing efficient methods to build and store the indexes is just one example of optimization that is needed for running binary encoding schemes on embedded hosts.

In this work, we present design and implementation strategies for running an Efficient XML Interchange processor on embedded devices for enabling Web connectivity through RESTful interface that is based on Constrained Application Protocol. The RESTful interface can be used for human-machine interactions with Internet of Things hosts as well as for implementing embedded distributed systems based on the Service Oriented Architecture, as discussed by [10].

Unlike XML, the EXI specification mandates the use of schema-specific parsing [11] when the EXI document is encoded with schema knowledge i.e. using schema mode. In order to address all possible use cases, the presented EXI processor supports both the schema and schema-less modes of operation. This is achieved by using dynamic state machine abstraction that can evolve through addition of new states and state transitions. The main benefit of using static state machines, as in the EIGEN [12] and libEXI [13] libraries, is the small footprint and hence the ability to implement highly optimized, dedicated EXI processors. In order to efficiently support a static mode of operation - in other words, strict schema processing with no deviations, the EXIP library needs to be configured to strip the code responsible for evolving the state machines. This can be done easily during compile time due to EXIP modular architecture.

One important component of EXI implementations supporting schema-enabled processing is the automatic generation of the state machines based on XML schema language definitions. These definitions are used to construct a set of formal grammars that describe a particular XML language which is then recognized by the generated state machines. EXIP includes an optimized and lightweight grammar generation utility that can be executed efficiently at run time. This allows it to support dynamic XML schema negotiations even on embedded hosts. The main contributions of this article are the grammar generation algorithms that are the core of the high performance of this utility. To the best of our knowledge, all other EXI implementations use an external library for processing the XML Schema definitions that are used for the grammar extraction. A commonly used external XML Schema library is Apache Xerces. However, its usage for embedded Web development is limited to static compile-time generation of the EXI state machines.

A prominent research work that is based on the approach of compile-time generation of the state machines is presented by [14]. The authors show that the use of EXI for embedded Web service development brings substantial benefits in hardware utilization (network, CPU, RAM and programming memory). Moreover, their work includes the design of a Web service code generator based on Simple Object Access Protocol (SOAP) and the HTTP/EXI/SOAP protocol stack. Promising future research work, as stated by the authors of that study, is to add support for CoAP RESTful Web service interface to the proposed generator. As such, the EXIP framework described herein is extending and further specifying the suggested CoAP RESTful Web service generator.

EXI is not the only possible data format that can meet the requirements of the embedded Web programming, but it has been shown to provide the highest efficiency compared to rival binary XML solutions [15]. Lightweight text formats such as JSON and Comma-separated values (CSV) or binary encoding schemes (ASN.1, BSON, Protocol Buffers, Thrift etc.) are also capable of representing very efficiently structured

information. However, the lack of formally defined mapping between these technologies and the XML Information Set [16] makes them unable to guarantee interoperability with existing Web technologies and protocols such as XHTML, Scalable Vector Graphics (SVG), Extensible Messaging and Presence Protocol (XMPP), and RSS feeds to name a few.

The initial goal of the EXIP library was only to provide efficient implementation of an EXI processor for embedded systems. Since the initial version of the prototype EXI processor, the EXIP library was used in a number of research projects and prototypes as in [17] and [18]. Based on the recurring need of higher processing efficiency and Web integration, the scope of the EXIP project has now extended, and new processing algorithms are employed. In addition to the grammar generation algorithms that are part of the EXI processor prototype implementation, this work defines the overall architecture of the EXIP Web development toolkit. The architecture consists of three main modules: the EXI processor library, EXI data binding, and the CoAP/EXI/XHTML Web page engine. Their functionality, required properties, and overall design in the context of embedded Web development are discussed in Section 2. Detailed descriptions of each of these modules and the associated research questions that are investigated are presented in Sections 3, 4, and 5, respectively.

2 Background

Optimizing the hardware utilization by the Web protocols is a key requirement for their application on embedded platforms. Very often the connected devices have limited memory (both RAM and programming memory), and use low-cost CPUs. If the device is battery powered, the communication overhead is a main contributor to the power consumption that needs to be carefully modeled in order to guarantee the intended up-time periods. Simulation tools such as PowerTOSSIM [19] can be employed to highlight areas of the protocol implementations that are mostly responsible for draining the battery. Among the use of radio duty cycling and CPU sleep modes, reducing the number of packets sent and received is another way of cutting the power consumption, especially in wireless applications.

W3C performed an extensive evaluation of the EXI format [20],[15] that shows substantial improvements in compactness compared to text encoding as well as other XML binary formats. Additionally, EXI has superior processing performance compared to plain XML. Both the compactness and processing efficiency depend heavily on the structure of the encoded documents and the options used for processing. For example, the use of XML schema information during encoding and decoding can cut the size of small documents more than 50 %, as the element and attribute qualified names are encoded as indexes instead of strings. This allows for substantial reduction of the number of packets required for communication of structured information over the network, and thereby minimizes the power consumption. Existing Web technologies that are formally described using XML schema language such as XHTML, for example, can then be efficiently represented for use in embedded applications.

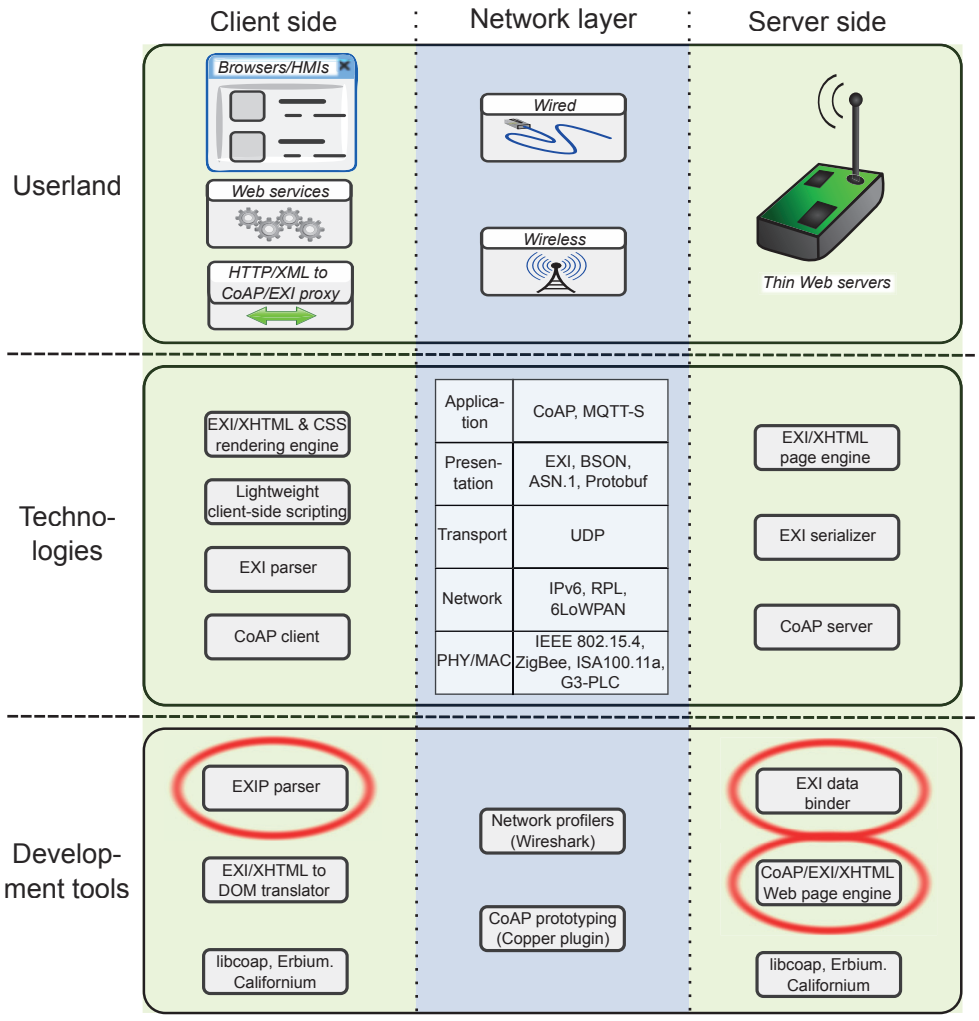


Figure 1: Overview of the tools and technologies for embedded Web development that are based on standard binary protocols. The tools that are the main focus of this work are marked with red ellipses.

Compaction and processing improvements of CoAP compared to HTTP are also significant, as reported by [21]. Moreover, the asynchronous design of the CoAP protocol makes it much more suitable for event-driven interactions. Publish/subscribe protocols are often preferred in embedded systems, as they provide better hardware and network utilization compared to polling schemes that are used by HTTP, for example.

Figure 1 provides an overview of the state of the art of embedded Web development along with a high level architectural view of the use of binary protocols for network communication and information exchange. The suggested components of the architecture are grouped depending on their role - client-side, networking layer, and server-side execution; and their application domain - user tools and applications, technologies/protocols/specifications, and development tools. The goal of this categorization is to show how the work presented in this article relates to the current technologies and applications, and to further motivate the need for this research.

As shown in Figure 1, client-side user applications of the embedded Web include browsers, graphical Web clients (HMI devices), embedded Web services, and proxy devices translating the binary Web protocols to their text-based counterparts. The technologies to implement these client-side user applications are CoAP client, EXI parser, lightweight client-side scripting engine, and EXI/XHTML/CSS rendering engine. The concrete development tools that can be used for implementing these technologies are the EXIP parser library, which is the primary objective of this work, EXI/XHTML to DOM translator, and CoAP libraries such as libcoap [21], Erbium [22] and Californium [23].

Similarly, the networking layer shows different wired and wireless network stacks and protocols grouped according to the OSI model along with developing tools used for debugging.

The server-side is represented by resource-constrained embedded devices that are conforming to the *thin server architecture* suggested by [23]. The server technologies include CoAP server, EXI serializer, and EXI/XHTML page engine. The proposed development server-side tools are the EXI data binder and CoAP/EXI/XHTML Web page engine that are described in detail in Sections 4 and 5 of this work. Other server-side tools for embedded Web development are again, the CoAP libraries libcoap, Erbium, and Californium.

EXI

EXI data format significantly reduces the size of XML when stored on disk or transferred over the network and also speeds up the parsing and serialization. According to [15] the compression level varies between 1 % of the original size for large and sparse documents with compression and schema options enabled to 95 % for schema-less encoding of very small and dense documents. Nevertheless, EXI format has few drawbacks that are inherited from XML and must be taken into account in the discussions that follow in this article. XML notation and semantics are perceived as complex both for humans to understand but also for machines to process which stems from the design goal of the format to be flexible and easily extendable for application in variety of use cases. This

flexibility creates a lot of special cases and exceptions that must be specifically handled with if-then-else statements during serialization and parsing. While EXI is very efficient in removing the redundancy in the XML syntax, it does not simplify the processing - it merely speeds it up. Besides, EXI adds another level of flexibility by introducing encoding options that can be used to influence the level of compression, processing speed and RAM usage during parsing and serialization. Providing support for all possible EXI options requires large and complex code base that can hardly fit into the programming memory of a highly resource constrained embedded device. Therefore, the application of EXI on such platforms often requires defining a profile of the EXI specification which restricts the supported EXI options to particular values and predefines the XML Schema. Different EXI profiles and how they are supported by EXIP are further discussed in Section 3.

Selecting the values for the EXI options is often a trade-off between memory usage, processing speed and level of compression (for example when setting the values of valuePartitionCapacity, valueMaxLength and compression options). Furthermore, as these parameters heavily depend on the structure of the documents and even on the schema design (as shown by [20],[15]) it is difficult to predict the level of efficiency when applying EXI on a particular set of XML documents without performing an extensive empirical study.

EXI theoretical foundations

The goal of this section is to provide the necessary background information for supporting the discussions on the EXI processor architecture and algorithms for embedded processing that follow without going into details of the inner workings of the EXI specifications. For in-depth overview of the EXI format, the reader is advised to refer to the W3C specification [24] and white paper [25].

An EXI stream is a sequence of events that describe the content of the XML document. These events are analogous to the streaming XML events and denote the start of an element or attribute, value items, closing tags and so on. For achieving higher compactness, the events are represented by a simplified Huffman coding [26] scheme. The occurrence of each event in the EXI stream is controlled/described by a set of formal grammars. The EXI specification very broadly identifies the formal grammars used as being in restricted Greibach normal form [27]. Support for the theoretical fitness of the discussed grammar generation algorithms is given in the next paragraph. It provides more concrete classification of the EXI grammars.

Unlike Greibach grammars, the EXI grammars have at most one non-terminal symbol on the right-hand side of the grammar productions. Therefore, all EXI grammar rules are in one of the following two forms: 1) $Z \rightarrow aY$ or 2) $Z \rightarrow a$,where Z and Y are intermediate (non-terminal) symbols and a is a terminal symbol. As all grammar rules are in one of these two forms, the EXI grammars are also *regular* and in particular *right linear grammars* as they require exactly one terminal on the right-hand side and at most one non-terminal which is at the end of the grammar rule. The regular grammars are strict subset of the context-free grammars according to the Chomsky hierarchy, and as

every context-free grammar can be represented in Greibach normal form [27], they are also a subset of the Greibach grammars.

Identifying the EXI grammars as regular grammars provides much more insight into their properties. For example, context-free grammars define very broad class of languages and are equivalent to pushdown automaton (PDA), while regular grammars are equivalent to nondeterministic finite automaton (NFA). Moreover, the EXI grammars are *simple* a.k.a. *s-grammars* [28] as each pair $Z \rightarrow a...$ appears only once in each EXI grammar. Based on this constraint, the EXI grammars are also *unambiguous* and support linear parsing time by deterministic finite automaton (DFA).

The process of converting a set of XML Schema definitions to EXI grammars includes four steps:

1. Create a set of proto-grammars that describe the content model according to the schema. The EXI proto-grammars are strictly context-free grammars that are neither regular nor in Greibach normal form as they allow unit productions: $Z \rightarrow Y$ where both Z and Y are intermediate (non-terminal) symbols.

2. Normalize the proto-grammars to EXI grammars. The normalization includes simplification of the proto-grammars by removal of the unit productions. This creates regular grammar that can be *ambiguous*, in other words, lacking unique leftmost derivation tree for every input. In this case a second simplification is performed in which the *ambiguous* regular grammars are transformed to unambiguous *s-grammars*.

3. Assign event codes to grammar productions

4. Extend the EXI grammar with additional productions that describe the possible deviations from the XML Schema

Section 3 describes an extension to the algorithm for creating proto-grammars from schema definitions [step (1)] that guarantees that the resulting grammars are regular *s-grammars*. This allows for avoiding the normalization of the proto-grammars as a separate second-step process.

Section 3 describes a modified version of the algorithm for augmenting the EXI grammars for handling schema deviations [step (4)]. The new version of the algorithm allows the removal of redundant grammar productions that are otherwise required by the approach described in the EXI specification.

Related work for XML grammars

The formal grammars used in the EXI specification express the constraints defined in the XML Information Set [16] and are not specific to EXI format itself. As such, the formal models and theoretical results developed for XML are also valid for EXI. There are two main theoretical models for studying the properties of XML languages and XML schema languages. The first model treats XML instances as strings and schema languages as formal languages that define particular sets of strings representing the possible XML

instances that are valid according to a certain schema. This model is based on context-free (word) grammars and their more restricted forms such as parenthesis and balanced grammars as presented by [29].

In the second model, the XML instances are treated as trees and the schema languages as formal languages defining sets of trees representing the valid instances according to a certain schema [30], [31]. The nested structure of the XML forms ordered unranked trees i.e. trees with nodes allowed to have any number of ordered child nodes. The theoretical foundation of this model are regular tree grammars which can be seen as a generalization of regular word grammars. The tree model is appropriate when studying the expressive power of different XML schema languages as shown by [32]. In this work, Murata et al. present a formal classification and comparison between DTD, W3C XML Schema, and RELAX NG based on the regular tree grammar theory.

Context-free word languages and regular tree languages are closely related. For example, it is proven that the set of derivation trees for a language defined by a context-free word grammar forms a regular tree language [33]. In addition, Brüggemann-Klein et al. show that tree grammars, and even more generally hedge grammars, are effecively identical to balanced grammars and that balanced languages are identical to regular tree languages, modulo encoding [34]. These results demonstrate that the two models are equally expressive and can be used interchangeably when studying or characterizing languages based on XML Information Set.

The discussions in this paper are following the first model, because the EXI specification defines the XML content with a set of regular word grammars as already presented in Section 2. For that reason, all grammars in this work are assumed to be *word* grammars even if not explicitly stated.

Instead of defining the terminal alphabet in terms of ASCII or UTF-8 characters, which is commonly used in word grammars, the EXI grammars use XML events (start element, attribute definition, end element etc.) as terminal symbols. This provides high level description of the XML content model without affecting the theoretical results developed for regular grammars. As XML Information Set defines context-free language parsed by pushdown automaton, a single regular grammar (a single DFA) is, in general, unable to represent (parse) the content of a whole XML document. Using a single regular grammar (or a single DFA) for describing (parsing) the whole content of an XML is possible when certain restrictions on the document structure are met by the XML/EXI instances. For example, this approach is used for efficient processing of SOAP Web services that are *ordered* XML documents with predefined schema [35]. A less restrictive form of schema-specific XML parsing that uses an extended version of PDA is presented by [11]. Unlike these approaches, the EXI specification defines the parsing and serialization of XML Information Set documents based on a stack of regular grammars. Each regular grammar in the stack describes the content of particular XML element. The stack of grammars is used to model the nesting of elements (e.g. parsing a nested element equals adding its regular grammar on the stack) similarly to the role of the stack in the PDA.

For illustrating how the grammar stack is used during processing in EXI it is conve-

nient to represent the XML Information Set in terms of extended context-free grammars (ECFG) which describe exactly the context-free languages and are the basis for DTD schema language [36]. In an extended context-free grammar each right-hand side of a production consists of a regular expression which is in turn equivalent to regular grammar or finite automaton. Consider the example XML instance and its corresponding ECFG shown in Table 1:

Table 1: Extended context-free grammar for a sample XML instance where element <notebook> can have zero or more <note> elements with optional <subject> and mandatory <body>. The following operators are used in the regular expressions in ECFG: **.** - denotes concatenation, ***** - Kleene star operator (zero or more occurrences), **?** - zero or 1 occurrence and [ ] - matches a single character from the specified set within the brackets. The non-terminal symbols are in uppercase letters.

Sample XML	**Corresponding ECFG**
<notebook> <note> <subject>Sample</subject> <body>XML Instance</body> </note> </notebook>	NOTEBOOK → <notebook>.(NOTE)*.</notebook> NOTE → <note>.(SUBJECT)?.BODY.</note> SUBJECT → <subject>.[UTF-8 characters]*.</subject> BODY → <body>.[UTF-8 characters]*.</body>

The set of regular grammars, used during processing of EXI documents, corresponds to the set of regular expressions in ECFG which describe the content of all possible elements. At every step the EXI processor uses the regular grammars on top of the grammar stack to process the content of the current element. Starting of a nested element involves pushing its grammar to the stack and closing an element pops its grammar from the stack. In this way, parsing the XML document shown in Table 1 involves: (1) parse the content of <notebook> element according to the regular grammar for that element which is initially the only grammar in the stack; (2) the start of the nested <note> element requires pushing its regular grammar on the stack and parsing its content according to that grammar; (3) on start of the nested <subject> element its grammar is pushed to the stack and used for parsing; (4) When all the content of <subject> element is parsed and there are no more nested elements at this level pop its grammar from the stack and continue processing according to the <note> grammar that is currently on the top of the stack; (...) the same procedure is repeated for the rest of the elements in this example.

Unlike DTD which defines a *local* language, the language defined by the set of regular grammars in EXI is a *single-type* language that corresponds to the expressive power of W3C XML Schema [32]. This essentially means that two or more elements sharing the same name but having different types are evaluated using different regular grammars that match their type. This differs from DTD where the name of an element uniquely identifies its content model (or, equivalently, the regular expression or the regular grammar of its

content).

CoAP

The Constrained Application Protocol [37] is specially designed for use with resource-constrained hosts over low-bandwidth network links. CoAP functionality resembles the HTTP request/response interaction model, and is based on the Representational State Transfer (REST) architecture of the Web [38]. CoAP also supports well established concepts of the Web such as URIs and Internet media types. This allows for transparent translation between CoAP and HTTP traffic while enabling Web interactions with embedded systems.

CoAP fulfills the requirements of the embedded domain such as providing support for asynchronous message exchange, multicast capabilities, lightweight discovery mechanism, very low overhead, and implementation simplicity. This is possible by using UDP as a transport protocol with optional reliable unicast support and Datagram Transport Layer Security (DTLS) instead of TCP and TLS. The use of UDP enables the implementation of CoAP lightweight publish-subscribe mechanism [39] supporting dynamic content exchange between embedded servers and Web clients. The built-in asynchronous exchange of events encoded with EXI provides features similar to the AJAX framework, but with much lower cost in terms of network bandwidth and hardware requirements for the hosts.

Application areas that would greatly benefit from an open and standard way to connect embedded hosts to the Web include various Internet of Things and machine-to-machine (M2M) applications such as home automation and energy management.

3 EXI Processor Design and Implementation

Deploying EXI-based RESTful Web services on resource-constrained hosts requires a modular implementation of the EXI processor library that can support different compile-time configurations depending on the application scenario. For example, some target platforms can make use of hash tables for fast lookups in the string tables, while others have too little RAM for that. In other cases, certain EXI options (e.g., compression, random access, etc.) are not allowed, and hence the code for processing them can be pruned from the library.

In this section, we present the modular design of the EXIP library [40] that enables compile-time profiling of the code base. As shown in Figure 2, by using fine-grained components that have low interdependencies, it is possible to define different profiles of the library that support a variety of use cases. Such profiles can be application-specific (e.g., full-featured, most-restricted, etc.), or defined as part of different communication standards - EXI Profile for limiting usage of dynamic memory [41], Vehicle to grid communication interface (ISO 15118), or other energy management standards [42] such as Smart Energy Profile 2.0 [43], and OpenADR, for example.

The encapsulation of the components' source code is done with the standard mechanisms available in the C programming language - splitting the code into different header

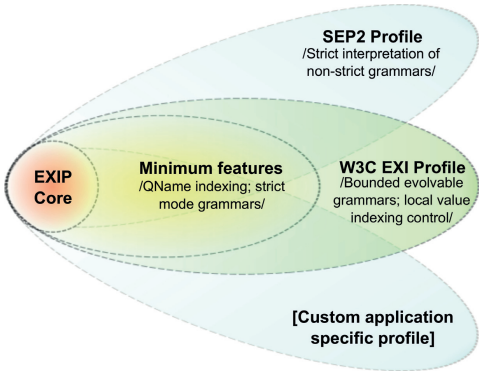


Figure 2: EXIP modular architecture and application profiles

and source files, and hiding the implementation in static functions, strictly avoiding the use of global variables and, where needed, using conditional C preprocessor macros. This enables the implementation of a simple and easy-to-maintain *Makefile* build system which can track the dependencies between the components. With this build system in place the developers can cherry-pick only the components that are needed during compile time which allows using the EXI Processor library for different application profiles or contexts.

Problem Formulation

The first step in supporting the requirements of the EXI-based embedded Web programming is to provide efficient Application Programming Interface (API) to encode and decode EXI streams. Already established XML APIs such SAX, DOM, and StAX are widely used in Java processors, but are shown to provide less than optimal efficiency for resource-constrained devices [44]. Other requirements of the EXI processor implementation include a small footprint and an easy-to-use code base that executes quickly, and consumes as little RAM as possible while being portable across a wide range of embedded platforms. Although the main goal of the EXIP library directly follows from these requirements, detailed description and evaluation of the degree to which these requirements are met is out of the scope of this paper. The reason for excluding these discussions is the low research value of the implementation technicalities that are involved in writing efficient and portable C code, a subject which is better presented by the EXIP library developers' documentation[2]. Instead, this section is focused solely on the grammar generation functionality that is an essential part of a number of use cases connected to dynamic/runtime exchange of schema information. The need for such runtime negotiation of the document structure is evident in supporting versioning of the schema documents and implementing generic Web services such as information logging and archiving, data visualization of

[2]Available at http://exip.sourceforge.net/

uncategorized information, dynamic Web service composition, and peer-to-peer services. A concrete example where the XML Schema documents are processed during runtime to generate EXI grammars is the specification draft for using EXI over Extensible Messaging and Presence Protocol (XMPP) [45].

The dynamic processing of XML schema information can also be employed in cases where no schema information is available to describe a particular set of XML documents. In such cases the XML schema can be inferred from the set of available XML examples, and used to enable more compact EXI encoding. Both the schema inference and the generation of EXI grammars can be done at runtime. Example approaches for schema inference include learning of deterministic regular expressions [46], as well as learning chain regular expressions, in the case of the Trang open source software library [47].

Efficient EXI Grammar Generation

The standard way of generating EXI grammars from XML Schema is to rely on a generic XML Schema parser/validator such as the Apache Xerces library. The role of the XML Schema parser is to load the schema definitions into appropriate structures in the memory. These structures are then converted to EXI grammars based on the algorithms specified in the EXI specification. The EXIP library takes a different approach by including a dedicated EXI grammar generator without external dependencies on schema parsers, which uses a modified version of the algorithms described in the EXI specification.

Many embedded targets use EXI because XML processing is too heavy to support. In such cases, the dynamic generation of the EXI grammars cannot be achieved in a standard way, as it requires processing text-based XML schema definitions. One possible solution is to use proprietary encoding for the EXI grammars, which is against the principles of the Web, and will still require some loading code that expands the programming memory footprint.

The dedicated EXI grammar generator solves this problem by using two simple ideas. First, the XML Schema document is itself an XML document that can be represented in binary using EXI, thus reducing its size and improving the loading time. Second, once represented in EXI, the XML Schema document can be parsed by the EXI parser itself without the need of an external library for that; in other words, the EXI decoder code is reused to extract the XML schema definitions.

EXI Grammar Concatenation and Normalization

The EXI specification defines an algorithm for building a set of context-free grammars that directly correspond to the definitions in the W3C XML Schema specification. These grammars are called proto-grammars as they are intermediate representation which is only used during EXI grammar generation. The process of building proto-grammars is roughly as follow:

1. a set of simple proto-grammars are defined that describe the content model for each atomic XML schema definition (attributes, simple types, element terms, wildcard

terms)

2. the proto-grammars for composite schema definitions are built by using the proto-grammars of their sub-components. For example, the <*sequence*> compositor equals to concatenation of the proto-grammars of its child elements and <*choice*> compositor equals to the union of the proto-grammars of its children.

The next step in the process of building EXI grammars is to normalize the proto-grammars such that all unit productions ($Z \rightarrow Y$ where both Z and Y are intermediate symbols) are removed and there are no ambiguities in the grammars. This essentially converts the proto-grammars to EXI grammars that are then used for processing EXI documents conforming to a schema.

The review of the algorithm for creating EXI proto-grammars from XML Schema definitions in section *8.5.4.1 EXI Proto-Grammars* of the EXI specification leads to the conclusion that the only way for creating proto-grammars that contain unit productions, and hence are not regular, is as an output of the grammar concatenation operator (see *8.5.4.1.1 Grammar Concatenation Operator* of the specification). However, all atomic grammars used as an input to the concatenation operator are regular and from the closure property of the regular languages under concatenation [48] we know that the resulting output grammar can also be presented in a regular form.

This section defines an extended grammar concatenation operator that produces regular EXI grammars, thereby removing the need for additional normalization of the grammars by removal of unit productions. The extended operator depends on the following recursive definition:

DEFINITION: Weak equality of grammar productions *The grammar production* $A : Z_1 \rightarrow a_1 Y_1$ *and the grammar production* $B : Z_2 \rightarrow a_2 Y_2$ *are* ***weakly equivalent*** *if:*

1. $a_1 \equiv a_2$ *and* $Y_1 \equiv Y_2$
   OR

2. $a_1 \equiv a_2$. *Let the sets of productions in the EXI grammar that have* Y_1 *and* Y_2 *as a left-hand side be denoted as* $\{Y_1\}$ *and* $\{Y_2\}$ *respectively. The two sets have the same cardinality, and each production* $P \in \{Y_1\}$ *is* ***weakly equivalent*** *to a production in* $\{Y_2\}$.

The grammar concatenation operator defined below is very similar to the one in the EXI specification in the sense that it creates a new grammar given two input grammars. The new grammar accepts any set of symbols accepted by the left operand followed by any set of symbols accepted by the right operand of the concatenation operator. The main difference is that the operator defined here produces regular EXI grammars, given its operators are also regular grammars.

DEFINITION: Extended grammar concatenation operator *Given two EXI Grammars* $L(N_l, T, S_l, P_l)$ *and* $R(N_r, T, S_r, P_r)$ *where* N_l *and* N_r *are finite sets of non-terminals,* T *is the set of terminal symbols representing the EXI events,* $S_l \in N_l$ *and* $S_r \in N_r$ *are both designated initial symbols, and* P_l *and* P_r *are the sets of grammar productions in* L *and* R *respectively. All grammar productions in* P_l *and* P_r *are in one of the following two forms:* $Z \rightarrow aY$ *where* $a \in T$ *and* $a \neq EE$ *or* $Z \rightarrow EE$ *where* $EE \in T$ *is the terminating* end element *EXI event.*

The result of applying the grammar concatenation operator to L *and* R, $L \bigoplus R$, *is a new grammar* $C(N_l \cup N_r, T, S_l, P_c)$ *where the set of productions* P_c *is defined as follows: each production* $l \in P_l$, *where* $l \neq Z \rightarrow EE$ *for every* $Z \in N_l$, *is part of* P_c*; each production* $r \in P_r$, *where* $r \neq S_r \rightarrow aY$ *for every* $a \in T$, *and* $Y \in N_r$ *is part of* P_c*. For each production* $e_l \in P_l$, *where* $e_l \equiv Z \rightarrow EE$ *for every* $Z \in N_l$, *the following set of productions is also part of* P_c*: the set* $\{Z \rightarrow aY\}$ *where a production* s_r *of the form* $S_r \rightarrow aY$ *exists in* P_r, *and* s_r *is not* ***weakly equivalent*** *to any production in* P_l *that has* Z *as a left-hand side non-terminal symbol. There are no other productions in* P_c *besides those defined with these rules.*

When the extended concatenation operator is used for XML Schema ⟨*sequence*⟩ definitions, the resulting regular grammar might contain productions with duplicate terminal symbols i.e. the result can be an ambiguous regular grammar. In this case the algorithm in section *8.5.4.2.2 Eliminating Duplicate Terminal Symbols* of the EXI specification should be further applied to the resulting concatenated EXI grammar. It is worth noting that these cases are extremely rare and can only occur when optional element particles are allowed to repeat more than once. Example content model that contains duplicate terminal symbols and leads to the creation of ambiguous regular grammar is the following:

```
<sequence maxOccurs="2">
  <element name="a" maxOccurs="3"/>
  <element name="b" minOccurs="0"/>
</sequence>
```

Efficient Representation of Schema Deviations

The EXI specification defines an algorithm that augments the EXI Grammars with additional grammar productions which are used to handle possible deviations from the XML schema. Such deviations are often used to add extensions to a particular protocol or handle cases that require additional information in the XML documents. Furthermore, certain XML events that are not explicitly declared in the schema may also occur in the instance documents without making them invalid (e.g. comments, processing-instructions, type casts using *type* attribute from `http://www.w3.org/2001/XMLSchema-instance` namespace).

One constraint that must be followed when adding productions to the normalized EXI grammars is that addition of productions allowing attribute deviations must only occur before the element content - otherwise the grammars describe a document which is not

well formed. The algorithm as described in the EXI specification (see *8.5.4.4.1 Adding Productions when Strict is False* [24]) depends on a set of redundant productions in the normalized EXI grammars in order to fulfill this requirement. The redundant productions are a copy of the productions describing the possible states for starting the content of an XML element that has wildcard attributes or a mixed-content model. An example of such redundant productions is the EXI grammar describing element fragments (see *8.5.3 Schema-informed Element Fragment Grammar* [24]).

The algorithm described in this section augments the EXI grammars for accepting schema deviations without having a dependency on redundant productions in the input EXI grammar. The algorithm is presented by highlighting only the modifications and differences with comparison to the algorithm in the EXI specification. An example of applying the modified algorithm is given in Appendix A.

The algorithm depends on the definition of a *content* non-terminal symbol, and an index called *content index* for each input EXI grammar. The assignment of *content index* and *content* to a non-terminal symbol is identical to the process defined in the EXI specification, and a prose description of it is given below:

DEFINITION: content non-terminal symbol *The **content** non-terminal symbol is the symbol that indicates that all attributes (AT terminal symbols) are already encoded. The **content** non-terminal symbol represents all the states for starting the encoding of the content of a particular XML element.*

DEFINITION: content index *Assign index numbers to all non-terminal symbols such that the designated initial symbol of the EXI grammar has index 0 and all other indexes are larger than 0. The index of the **content** non-terminal symbol, in other words, the **content index**, is then the smallest index that is larger than the indexes of all non-terminal symbols that are used as a left-hand side in grammar productions with AT terminals.*

DEFINITION: Grammar augmentation for schema deviations *Create a copy of all grammar productions that have the **content** non-terminal on the left-hand side if and only if there are AT productions that have the **content** non-terminal symbol on their right-hand side or the **content index** is 0. The copy of the **content** non-terminal symbol - **content2** if available, is inserted just before the **content** i.e. it has index of (**content index - 1**). In the case when the **content index** is 0, that would mean that the **content2** is now the entry non-terminal symbol of the grammar. After the copying, there should be no productions with **content2** non-terminal on the left-hand side that have **content2** on their right-hand side - instead they should have only **content**. All AT productions that have a **content** non-terminal symbol on their right-hand side are changed to point towards **content2** instead.*

*Apply the procedure in **8.5.4.4.1 Adding Productions when Strict is False** [24] while applying the following modifications to the algorithm:*

- *The designated initial symbol of the EXI grammar is changed to* ***content2*** *when* ***content index*** *is 0.*
- *Change each occurrence of* ***content*** *with* ***content2*** *and vice versa, that is, each occurrence of* ***content2*** *with* ***content****.*
- *If there is no* ***content2*** *non-terminal, then do not perform the procedure for it and assume the* ***content2 index*** *is smaller than the* ***content index****, but larger than the indexes of all non-terminals that are used in AT productions.*

Performance Evaluation

The goal of this section is to evaluate the performance of the dedicated EXI grammar generator implemented as part of the EXIP library. The grammar generator accepts EXI encoded XML Schema definitions as an input, and uses the extended grammar concatenation operator and the algorithm for efficient representation of schema deviations. The measurements in this section are indicative and aim to compare the execution time and memory usage of grammar generation on real-world data. As the core contribution of this work is in the grammar generation utility, this section does not evaluated the overall EXI processing performance. Measurements of the EXI parsing speed are included only to the extent needed to put the grammar generation evaluation in context.

Description of the test setup A set of 5 XML schema documents were used for decoding 15 instances (XML examples that conform to the schema; 3 instances per each schema document) by 3 different EXI processors. Decoding in this experiment refers to converting a binary EXI file to its text-based XML representation. The EXI processors are EXIficient v0.9.1 Java [49], OpenEXI v1.0238.0 Java [50], and EXIP v0.5.3 C [51]. At the time of writing this article - June 2014, there is one more open source EXI parser - WS4D-uEXI[3]. WS4D-uEXI is written in C and is designed for constrained embedded devices. It is not included in this comparison as it uses EXIficient library for building the EXI grammars at compile time and therefore does not support runtime grammar generation [52]. Moreover, WS4D-uEXI implements a subset of the EXI specification and its current version (SVN r2) is unable to decode some of the EXI instances in this evaluation due to missing features.

The evaluation uses the following XML schema documents: netconf.xsd[4], SenML.xsd[5], sep.xsd[6], OPC-UA-Types.xsd[7], and XMLSchema.xsd[8]. All of them were accessed from the local hard-drive, including the imported XML schema files, so there were no dependencies on the network performance.

[3]http://code.google.com/p/ws4d-uexi/

[4]Network Configuration Protocol: https://www.iana.org/assignments/xml-registry/schema/netconf.xsd

[5]Sensor Markup Language: http://tools.ietf.org/html/draft-jennings-senml-10

[6]SEP2: http://www.zigbee.org/Standards/ZigBeeSmartEnergy/SmartEnergyProfile2.aspx

[7]OPC-UA: http://opcfoundation.org/UA/2008/02/Types.xsd

[8]Schema for XML Schema: http://www.w3.org/2001/XMLSchema

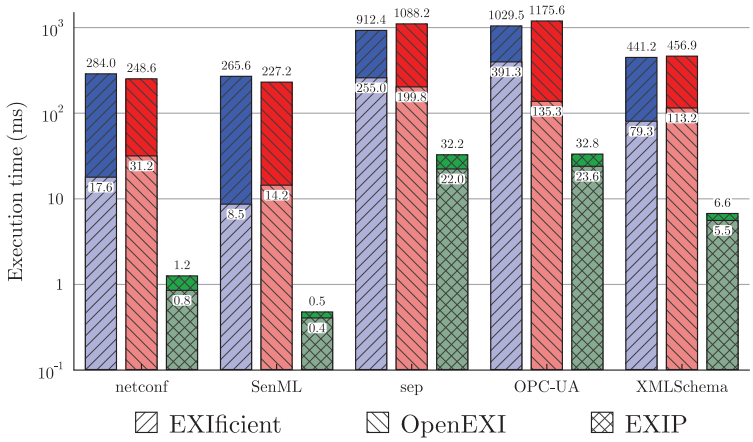


Figure 3: Grammar generation execution times for each XML Schema test case. The averaged times per XML schema are given on the logarithmic Y axis for each of the tested EXI processors - EXIficient (leftmost column, forward slash hatching), OpenEXI (middle column, backslash hatching) and EXIP (rightmost column, grid hatching). Each bar in the chart represents the execution times when explicit optimizations are applied (lighter colored part of the bar) and when no optimizations are applied.

The tests were executed on a desktop PC (Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz, 4GB RAM @ 1067 MHz) running 32-bit Linux Ubuntu 13.10. The version of the Java Virtual Machine (JVM) used for running EXIficient and OpenEXI is Java HotSpot(TM) Server VM 1.7, and the C compiler used for EXIP is GCC 4.8.1.

Two distinct measurements of the execution time were performed for each EXI processor: (1) the time it takes for loading an XML Schema and converting it to EXI grammars, and (2) the time it takes to generate the EXI grammars as well as decode a sample XML instance. The time was measured using *System.nanoTime*() in Java and *clock_gettime*() in C, in other words, we measured wall-clock time which can vary depending on the external load of the system. In order to get comparable results, the tests were executed ensuring similar conditions on the system load, and taking the mean value of 300 measurements. Moreover, the mean value is calculated for two distinct runs of the test framework - one with optimizations and one without applying optimizations. In the unoptimized case the Java processors run on a "cold" JVM i.e. the code is executed for the first time on the VM and hence the classes for grammar generation and instance decoding are loaded at runtime. Also the "cold" JVM has smaller chance for applying run-time optimizations such as Just-In-Time (JIT) compilation. Conversely, the optimized case uses "warmed-up" JVM where the tests are run 5 times on the JVM before the measurement are taken. The EXIP processor is compiled with $-O0$ flag for

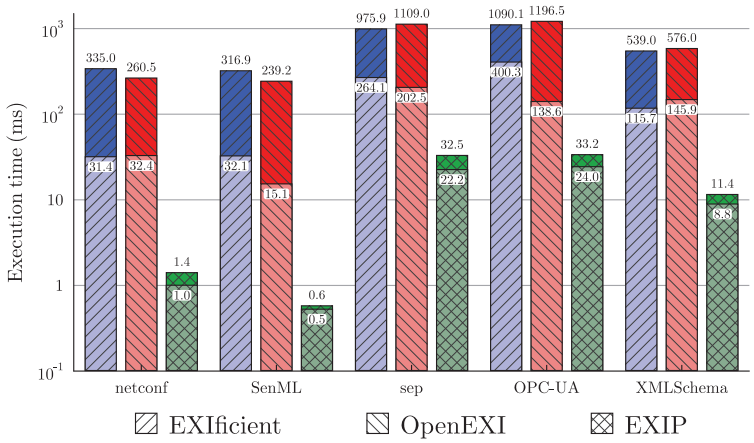


Figure 4: Grammar generation and instance decoding execution times for each XML Schema test case. The averaged times per XML schema are given on the logarithmic Y axis for each of the tested EXI processors - EXIficient (leftmost column, forward slash hatching), OpenEXI (middle column, backslash hatching) and EXIP (rightmost column, grid hatching). Each bar in the chart represents the execution times when explicit optimizations are applied (lighter colored part of the bar) and when no optimizations are applied.

unoptimized case and with $-O3$ for the optimized run.[9]

Figure 3 and Figure 4 show the averaged execution times per each XML schema test case with enabled and disabled optimizations. In Figure 3 the times are for grammar generation only while Figure 4 shows the execution times for both grammar generation and instance decoding. In both charts, the execution times on the Y axis are represented in logarithmic scale for enhancing the visual representation.

Table 2: Averaged execution times (ms) for all XML Schema test cases

EXI Processor	Optimized		Unoptimized	
	Grammar	Grammar+Instance	Grammar	Grammar+Instance
EXIficient	150.3	168.7	586.5	651.4
OpenEXI	98.8	106.9	639.3	676.2
EXIP	10.5	11.3	14.7	15.8

On average, among all test cases, the execution times for grammar generation and

[9]The automated test framework for configuring and executing the evaluation is available open source at http://github.com/kjussakov/exip-eval

instance decoding are given in Table 2. As shown in the table, EXIP generates the grammars about 9 times faster than OpenEXI and 14 times faster than EXIficient when compile time optimizations for the C code and run-time JVM optimizations for the Java code are enabled. This cannot be attributed solely to the performance difference in native code versus Java byte code execution where on average Java programs are somewhere between 50 % faster to 4 times slower than their C counterparts[10].

The superior performance of EXIP grammar generation is mainly due to the use of EXI-specific XML Schema parser that accepts EXI encoded XML Schema definitions as opposed to the use of general purpose XML Schema parser. By using the extended grammar concatenation operator (see Section 3), EXIP has to perform one less iteration over the set of all grammar rules which has noticeable benefits mainly in large XML Schemas such as SEP2 (sep.xsd). The grammar augmentation algorithm presented in Section 3 has no effect on processing efficiency, but slightly improves memory usage. Code optimizations, in terms of avoiding unnecessary loops and selecting appropriate searching and sorting algorithms (for example the use of a hash table for mapping element definitions to their globally defined types instead of iteration), have impact on the performance as well but are harder to quantify.

Memory usage

This section provides some insight into the memory consumption of EXIP, and EXI in general, as memory is often a bottleneck in embedded system applications. Section 2 already discussed that the dynamic memory usage for EXI processing can be controlled by some of the parameters defined in the EXI header. This is done by adjusting the extent of the content indexing used to detect and reduce redundancy in the data which also affects the compactness and processing speed. However, the mechanisms provided in the EXI specification cannot guarantee bounded run-time memory usage when deviations from the XML schema are present. For that purpose, an extension to these mechanisms are developed in a complementary specification called EXI Profile for limiting usage of dynamic memory [41]. A subset of this profile is supported by EXIP but its impact on the memory consumption is not evaluated in this section as the tests presented here are restricted to a schema valid instance of the SenML standard. Table 3 shows the size and memory usage during encoding and decoding for a sample instance document borrowed from the SenML specification[11].

The size and memory consumption are given for different encoding options. The platform used for testing is Raspberry Pi embedded computer with ARM-based system on chip including 700 MHz processor with 512 MB of RAM. The memory usage presented in Table 3 shows only the amount of dynamic memory (heap) usage for statically compiled EXI grammars and is measured using DHAT (dynamic heap analysis tool) that is part of the code profiling library Valgrind.

An interesting observation is that although the document is relatively small, turning

[10]Source: http://benchmarksgame.alioth.debian.org/u32/java.php
[11]Available at: http://tools.ietf.org/html/draft-jennings-senml-10#section-7

Table 3: Size of a SenML instance for different encoding modes and memory usage for EXIP and the light-weight XML parser library MiniXML on a Raspberry Pi system. The rows are ordered by document size.

Encoding mode	Size (bytes)	RAM/heap usage (kB)			
		EXIP		MiniXML	
		Encoding	Decoding	Encoding	Decoding
Plain XML	387	-	-	1.36	1.55
EXI Schema-less byte aligned	248	7.95	8.26	-	-
EXI Schema-less no value indexing	237	6.93	6.79	-	-
EXI Schema-less default options	200	7.90	8.26	-	-
EXI Schema mode no value indexing	137	1.93	2.27	-	-
EXI Schema mode default options	100	2.87	2.21	-	-
EXI Schema mode strict	98	2.89	2.23	-	-

off the indexing of repeating values (i.e. setting *valuePartitionCapacity* parameter to 0) substantially inflates the size of the resulting EXI representation. This is due to the high redundancy in the attribute values which has profound affect even in schema mode encoding. This simple example shows the high variation of compression and dynamic memory usage depending on the content of the documents and the encoding options in use.

The compile-time allocated RAM used by the EXIP library (calculated as the sum of .rodata, .data and .bss sections in the Executable and Linking Format (ELF)) is 23 kB (of which 8 kB EXI grammar definitions used for schema mode cases) while the light-weight XML parser MiniXML v2.8 requires only 3 kB. EXIP SenML parser uses 79 kB programming memory while MiniXML uses only 16 kB. Additionally, as shown in Table 3, MiniXML is more efficient in the use of dynamic memory compared to EXIP. These results indicate that EXI processing, and EXIP library in particular, require more RAM compared to highly optimized XML processing. The main reason for this is the use of content indexing and grammar information during EXI processing. Further optimizations of the RAM usage in EXIP are possible both for the size of the content index as well as the in-memory grammar representation. It should also be noted that schema-based EXI processing implicitly performs partial schema validation while MiniXML is a non-validating parser.

Enabling run-time EXI grammar generation from the SenML schema additionally requires 57 kB of dynamic memory and 37 kB of programming memory. These memory requirements show that the run-time grammar generation module fits easily in embedded devices such as Raspberry Pi but is too heavy for the most constrained platforms. As an example, the popular Stellaris LM4F120H5QR 32-bit ARM Cortex-M4F microcontroller (80 MHz CPU frequency, 256 KB flash and 32 KB SRAM) does not have enough RAM for supporting run-time EXI grammar generation. Nevertheless, by using static grammars the EXIP library is capable of running on such platforms with averaged total RAM usage of about 20 kB[12] and 60kB of programming memory for the SenML sample instance.

[12]The RAM usage in schema mode is 20 kB (1 kB stack size + 2.5 kB heap + 16.5 kB .data and .bss)

4 EXI data binding

The information contained in an XML/EXI document is often loaded into the memory for further processing and mapped to a hierarchy of data structures or objects that are maintained by the applications. For example, a status report by a device can include various hierarchal information such as network status (which in turn contains parameters like RTT, signal strength, connected peers etc.) or resource utilization (storage space, battery level etc.) that is mapped to a corresponding hierarchy of programming objects. The process of generating an XML/EXI document from a hierarchy of objects and vice versa is known as *XML/EXI data binding*. The process of building objects from an XML/EXI input document is called *unmarshalling* and the reverse - the generation of XML/EXI output document from objects, is called *marshalling*. The *unmarshalling* is implemented as a software module that connects to the parser API, and generates memory structures that correspond to the structure and content of the XML document. The *marshalling* is implemented as a module that transforms a set of objects in the memory to a sequence of calls to the serialization API.

The *XML/EXI data binding* code can be complex to write and maintain manually. For that reason, it is often automatically generated. There are two main approaches when generating the code and keeping it in sync with the XML/EXI documents - direct, and indirect mapping. In direct mapping, the source code is generated based on XML schema definitions or vice versa - the XML schema can be built based on the existing source code definitions. When no schema information is available or needed, the XML tree can be directly mapped to a memory representation, as in the case of the Document Object Model (DOM). The data binding frameworks that are based on direct mapping of the XML Information Set and the memory representation, are widely adopted in desktop and enterprise applications - examples include DOM, JAXB, XMLBeans, and others [53]. Their main advantage is that it is very easy to build and maintain the XML-to-source code mapping. An example of a pure XML direct mapping framework for embedded systems development is the gSOAP toolkit [54]. A similar approach, but applied to EXI and targeted at highly resource-constrained embedded devices is the automatic EXI Processor generation reported by [55].

The indirect mapping is a more flexible approach that allows discarding the unnecessary XML structures or reusing existing objects in the memory by defining a layer of indirection between the XML Information Set and the memory representation. Example libraries in this category include Castor and JiBX [56] - both only available in Java, and targeted at server/desktop applications. A comparison between the two approaches i.e. direct and indirect mapping, along with performance measurements, are presented by Sosnoski in *IBM developerWorks* article on data binding tools for Java/XML [57].

The EXI binding presented in this section falls into the category of indirect mapping, and it is targeted at embedded systems development. Its design is based on the following requirements:

---

while
the RAM usage in schema-less mode is 19 kB (1 kB stack size + 7.5 kB heap + 10.5 kB .data and .bss)

- The mapping rules should have intuitive syntax and semantics.
- The binding definitions should be independent from the programming language in use - the same binding definition should work for programs written in C, Java, Python, and so on.
- The EXI binding should be efficient to use on embedded platforms.
- The mapping layer should allow for loading the binding definitions and building the objects in memory dynamically at run-time.

To optimally fulfill these requirements, we propose template-based binding definitions that are written in XML and converted to EXI before being used for code generation or loading at runtime. The binding templates are very similar to other frameworks for dynamic content delivery based on templates such as JavaServer Pages (JSP) technology. An in-depth overview of template-based code generation is presented by [58] where the authors describe the theoretical foundations of template systems and include comparison with other code generation techniques. The proposed EXI template framework is a heterogeneous code generator that follows the model-view-controller design pattern as suggested by [58].

Figure 5 shows a comparison of this approach to what is a commonly used method for defining such binding definitions. As depicted, the mapping between dynamic EXI content and programming constructs is done using a special character @ and semicolon notation. As such, the definitions are intuitive to define as well as simple to process by the loading code. As with other such approaches based on templates, these special characters must be escaped when used in a static content. As an example, the value for a static attribute *email* within an EXI binding definition should be defined as *example*@@*com* to escape the special character that indicates the beginning of dynamic content mapping.

5 CoAP/EXI/XHTML Web page engine

This section presents a prototype implementation of a dynamic Web interface for an embedded sensor platform based on CoAP/EXI/XHTML technologies. The implementation is developed using the EXIP framework, and consists of an experimental Java browser running on a laptop PC that connects to a wireless sensor device (Mulle version 3.2 [59]) over Bluetooth. The laptop user can navigate to the device Web interface using mDNS/DNS-SD or CoAP built-in discovery capabilities - multicast service discovery [37], or CoRE Resource Directory [60]. In our simplified test setup, the network address of the sensor device is predefined so the discovery process was not implemented.

The EXI encoded XHTML page is dynamically generated on the sensor platform on a CoAP GET request, and it contains an *iframe* tag with a link to an external observable CoAP resource:

```
...
<p>Current temperature is:</p>
```

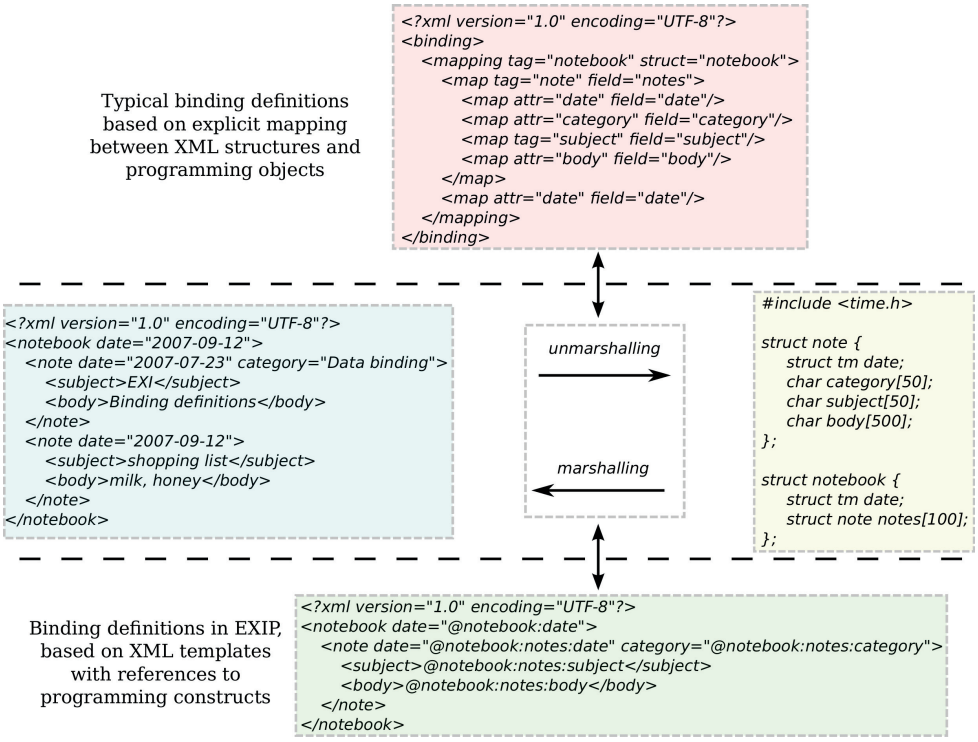


Figure 5: Comparison between typical binding definitions and the EXIP templates

```
<iframe src="coap://192.168.150.10:5683/temp"/>
...
```

The observable CoAP resource is linked to a temperature sensor on the wireless device, and is updated whenever there is a change in the measured air temperature.

As shown in Figure 6, upon user request, the Java browser[13] sends a CoAP GET request to the sensor device using the *Californium* library. The wireless device receives the request and generates a CoAP response using the *libcoap* library. The CoAP version 13 payload is a dynamically generated EXI/XHTML Web page using the EXIP framework. Once the packet is transmitted back to the Java browser, the EXI document is decoded by the *OpenEXI* library, and the *iframe* link is resolved by initiating an additional CoAP request to fetch the temperature. By using CoAP Observe [39], the Java client subscribes to changes in the temperature without requiring additional periodic polling requests. The temperature is represented in plain text, and visualized on the browser window each time a CoAP notification is received.

[13]Based on Flying Saucer XHTML/CSS 2.1 renderer: https://code.google.com/p/flying-saucer/

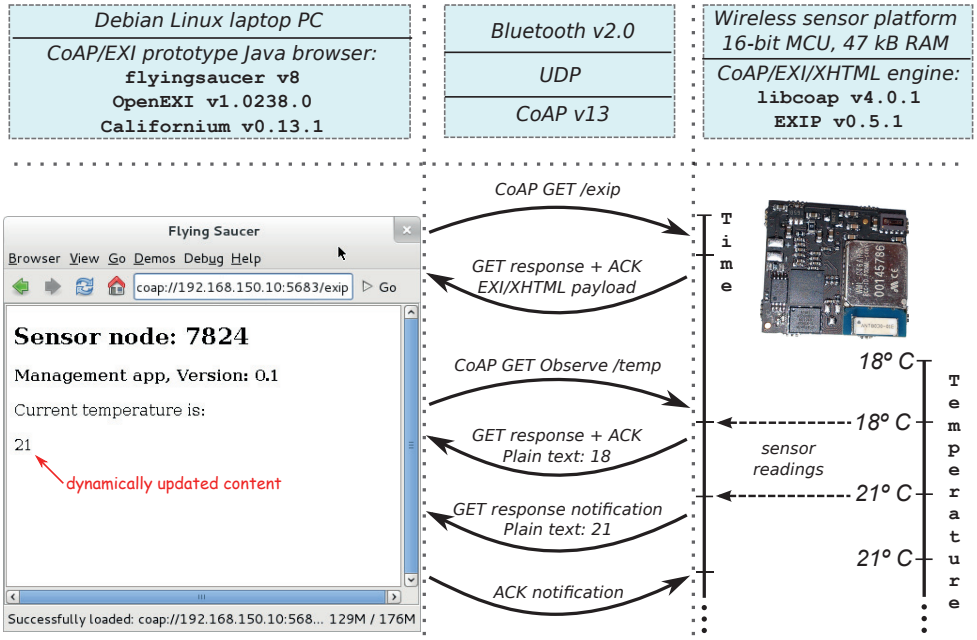


Figure 6: CoAP/EXI/XHTML dynamic Web interface demonstration

This prototype demonstrates how the newly emerging binary Web protocols can be employed to enable a dynamic Web interface for highly resource-constrained embedded devices. The Web interface can be used in a wide range of mobile applications, as suggested by [61]. The approach of using the *iframe* tag with CoAP Observe enables very lightweight event-based content delivery that is suitable for low-power radio communications such as IEEE 802.15.4 (6LoWPAN, ZigBee), Z-Wave, or Bluetooth low energy. Example application domains for the EXIP framework include, but are not limited to: industrial process monitoring and control, eHealth and elderly care, wearable electronics, home automation, and energy management.

Data visualization technologies based on XML encoding such as SVG[14] and X3D[15] can be readily included in the CoAP/EXI/XHTML engine to efficiently represent graphical indicators (e.g., battery level, signal strength) and visualize measurements and configuration parameters. An evaluation of EXI encoding for SVG in rich media applications for embedded systems presented by [62] shows that EXI significantly increases the efficiency of the SVG format. Also shown in this work is an approach using the EXI header option *datatypeRepresentationMap* to further optimize the compression of graphics formats for embedded web applications.

[14]Scalable Vector Graphics (SVG): http://www.w3.org/TR/SVG11/
[15]X3D Specification for 3D Graphics: www.web3d.org/x3d

The presented CoAP/EXI/XHTML Java browser always tries to subscribe to the *iframe* CoAP links - if the resource is not observable, the subscription is not established. When the resource is observable but should be treated statically for display in the browser (for example representing a snapshot of dynamic data), the embedded server should reject the subscription request by the browser. This approach can be too limited in certain scenarios, in which case different ways to indicate whether the browser should subscribe to changes on the *iframe* resource can be employed - adding an extra boolean argument *observe* to the *iframe* tag as an XHTML schema deviation, or requesting the resource description in CoRE Link Format before sending the subscription request. Similarly, in more complex scenarios the use of plain text encoding for the *iframe* resource might be too limiting. In such cases a structured format such as EXI can be used instead of plain text. The definition of the data format (including parameters and schema if available) for particular *iframe* can be defined as XHTML schema deviation or read from the CoRE Link Format as suggested for the observe use case.

Implementation details The information provided hereafter gives more insight into the actual implementation, and is useful for reproducing the test setup. The Mulle sensor platform has a 16-bit Renesas M16C/62P microcontroller and Mitsumi Bluetooth 2.0 wireless module. The application runs on bare metal, in other words, without an OS, on top of a port of *lwIP* TCP/IP stack and a *libcoap* v4.0.1 library. The EXI/XHTML generation is done in schema-less mode using *EXIP* v0.5.1.

The laptop PC is running Debian Linux, and is equipped with a USB Bluetooth 2.0 adapter. Debian packages *bluez-compat v4.99-2*, *bridge-utils v1.5-6*, and *isc-dhcp-server* were installed and configured on the system to enable TCP/IP communication over Bluetooth.

The size of the EXI/XHTML Web page is 239 bytes, and is generated directly in binary (EXI) form without transition to plain XML. If converted to text XHTML, the size is 427 bytes. The temperature notifications are in plain text, and account for 14 bytes of CoAP packet size (UDP payload) in total, assuming 2 bytes for the plain text temperature value.

6 Conclusions

The newly emerging transport and data representation protocols based on binary encoding - CoAP and EXI - provide an efficient way to connect embedded systems to the Web across scenarios as diverse as mobile computing, home automation, and smart grid. As the translations between CoAP $\Leftrightarrow$ HTTP and EXI $\Leftrightarrow$ XML are well defined, the integration of these binary protocols to the existing Web infrastructure is standardized and conforms to the well-established programming interfaces. For example, EXI processors often provide the same API as XML processors, and CoAP/HTTP proxies are simple to deploy and are transparent for the Web users.

The work presented in this paper shows that the use of CoAP/EXI stack and the EXIP Web development toolkit enables reuse of the existing pool of Web technologies

and developers' skills, even on very resource-constrained embedded platforms. The development process and especially the integration with existing systems is much faster and easier to maintain as compared to the use of handcrafted communication protocols.

Moreover, the presented EXI processor design, and the EXI grammar generation algorithms in particular, provide superior processing performance compared to the methods described in the EXI specification with order-of-magnitude speed-up in some of the test cases. This could enable exchange patterns supporting dynamic XML schema negotiations even for embedded hosts. The use cases for such an approach include support for schema versioning, generic Web services, and runtime service composition.

Finally, the presented prototype of dynamic Web interface for sensor platforms demonstrates the possibility to use event-based Web content delivery with a very low overhead in terms of network bandwidth and processing power. The development of the Web interface or Web service exchange can be automated by using the template-based EXI data binding. As the data binding creates indirect mapping between the EXI document and the programming constructs, the memory structures and programming objects can be reused when generating or decoding the EXI streams.

Possible extensions of this work include in-depth memory consumption evaluation and trade-off analysis as well as developing a formal specification of the EXIP data binding, and implementing prototypes in C and Java to evaluate the proposed approach against existing XML data binding frameworks. Providing support for light-weight client-side scripting as part of the CoAP/EXI/XHTML embedded Web programming is also an interesting and important topic for future investigation. It is also worth analyzing the application of CoAP/EXI, and the EXIP framework in particular, for mobile platforms and even for desktop applications that are not resource-constrained. Lowering the network traffic and CPU cycles for Web content delivery on mobile phones and tablet PCs could potentially increase the battery life for these devices, lower the networking cost for both operators and users, and even lead to energy savings if applied on a global scale.

*

APPENDIX A: Grammar Augmentation Algorithm Example This appendix gives an example of how the augmentation procedure is applied to the wildcard XML schema type *anyType* which is the base type definition for all other XML schema types. A minimal (without redundant productions) EXI grammar that describes the content model according to the process of creating proto-grammars is:

```
anyType-0:
        AT(*)   anyType-0
        SE(*)   anyType-1
        EE
        CH      anyType-1

anyType-1:
        SE(*)   anyType-1
        EE
        CH      anyType-1
```

There is no need to copy the content grammar productions (the ones with anyType-1 on the left-hand side) because there are no AT productions that points to it and the *content index* is 1.

Then apply the procedure in *8.5.4.4.1 Adding Productions when Strict is False* [24]: As there is EE production already do not add additional one. Adding the AT(xsi:type) and AT(xsi:nil) productions produces:

```
anyType-0:
        AT(*)           anyType-0
        SE(*)           anyType-1
        EE
        CH              anyType-1
        AT(xsi:type)    anyType-0
        AT(xsi:nil)     anyType-0

anyType-1:
        SE(*)           anyType-1
        EE
        CH              anyType-1
```

"For each non-terminal $Element_{i,j}$, such that $0 \leq j \leq content$ ..."
becomes:
"For each non-terminal $Element_{i,j}$, such that $0 \leq j \leq content2$ ..."
Because there is no *content*2 we apply that rule only to anyType-0:

```
anyType-0:
        AT(*)                   anyType-0
        SE(*)                   anyType-1
        EE
        CH                      anyType-1
        AT(xsi:type)            anyType-0
        AT(xsi:nil)             anyType-0
        AT(*)                   anyType-0
        AT(*)[untyped value]    anyType-0

anyType-1:
        SE(*)                   anyType-1
        EE
        CH                      anyType-1
```

After adding the NS and SC productions:

```
anyType-0:
        AT(*)                   anyType-0
```

```
            SE(*)                   anyType-1
            EE
            CH                      anyType-1
            AT(xsi:type)            anyType-0
            AT(xsi:nil)             anyType-0
            AT(*)                   anyType-0
            AT(*)[untyped value]    anyType-0
            NS                      anyType-0
            SC Fragment

anyType-1:
            SE(*)                   anyType-1
            EE
            CH                      anyType-1
```

"Add the following productions to each non-terminal $Element_{i,j}$, such that $0 \leq j \leq content$."
becomes:
"Add the following productions to each non-terminal $Element_{i,j}$, such that $0 \leq j \leq content2$."
The result of applying this rule is:

```
anyType-0:
            AT(*)                   anyType-0
            SE(*)                   anyType-1
            EE
            CH                      anyType-1
            AT(xsi:type)            anyType-0
            AT(xsi:nil)             anyType-0
            AT(*)                   anyType-0
            AT(*)[untyped value]    anyType-0
            NS                      anyType-0
            SC Fragment
            SE(*)                   anyType-1
            CH[untyped value]       anyType-1

anyType-1:
            SE(*)                   anyType-1
            EE
            CH                      anyType-1
```

"Add the following productions to $Element_{i,content2}$ and to each non-terminal $Element_{i,j}$, such that $content < j < n$, where n is the number of non-terminals in $Element_i$."
becomes:
"Add the following productions to $Element_{i,content}$ and to each non-terminal $Element_{i,j}$, such that $content2 < j < n$, where n is the number of non-terminals in $Element_i$."
The final grammar is:

```
anyType-0:
            AT(*)                   anyType-0
            SE(*)                   anyType-1
            EE
            CH                      anyType-1
            AT(xsi:type)            anyType-0
            AT(xsi:nil)             anyType-0
            AT(*)                   anyType-0
            AT(*)[untyped value]    anyType-0
            NS                      anyType-0
            SC Fragment
            SE(*)                   anyType-1
            CH[untyped value]       anyType-1

anyType-1:
            SE(*)                   anyType-1
            EE
```

```
CH                      anyType-1
SE(*)                   anyType-1
CH[untyped value]       anyType-1
```

2A Acknowledges

The authors would like to thank Takuki Kamiya, Yusuke Doi, Daniel Peintner, and the other members of the W3C EXI working group for the fruitful discussions and valuable feedback. We are very grateful to the contributors and the users of the EXIP project for their feedback, code review, and source code contributions. We acknowledge the valuable discussions and support by the partners of the EU projects IMC-AESOP and Arrowhead.

References

[1] M. Kovatsch, M. Weiss, and D. Guinard, "Embedding internet technology for home automation," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, 2010, pp. 1–8.

[2] G. Bumiller, L. Lampe, and H. Hrasnica, "Power line communication networks for large-scale control and automation systems," *Communications Magazine, IEEE*, vol. 48, no. 4, pp. 106–113, 2010.

[3] G. Borriello and R. Want, "Embedded computation meets the world wide web," *Commun. ACM*, vol. 43, no. 5, pp. 59–66, May 2000. [Online]. Available: http://doi.acm.org/10.1145/332833.332839

[4] A. Charland and B. Leroux, "Mobile application development: web vs. native," *Commun. ACM*, vol. 54, no. 5, pp. 49–53, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1941487.1941504

[5] R. Gossweiler, C. McDonough, J. Lin, and R. Want, "Argos: Building a web-centric application platform on top of android," *Pervasive Computing, IEEE*, vol. 10, no. 4, pp. 10–14, april 2011.

[6] V. Trifa, S. Wieland, D. Guinard, and T. M. Bohnert, "Design and implementation of a gateway for web-based interaction and management of embedded devices," in *Proceedings of the 2nd International Workshop on Sensor Network Engineering (IWSNE 09)*, Marina del Rey, CA, USA, Jun. 2009.

[7] Q. Kang, H. He, and H. Wang, "Study on embedded web server and realization," in *Pervasive Computing and Applications, 2006 1st International Symposium on*, aug. 2006, pp. 675–678.

[8] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, "Smews: Smart and mobile embedded web server," in *Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on*, march 2009, pp. 571–576.

[9] R. Van Engelen, "Code generation techniques for developing light-weight xml web services for embedded devices," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 854–861.

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

[11] K. Chiu and W. Lu, "A compiler-based approach to schema-specific xml parsing," in *First International Workshop on High Performance XML Processing (May 2004)*, 2004.

[12] Y. Doi, Y. Sato, M. Ishiyama, Y. Ohba, and K. Teramoto, "Xml-less exi with code generation for integration of embedded devices in web based systems," in *Internet of Things (IoT), 2012 3rd International Conference on the*, oct. 2012, pp. 76 –83.

[13] A. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi, "Web services for the internet of things through coap and exi," in *Communications Workshops (ICC), 2011 IEEE International Conference on*, june 2011, pp. 1–6.

[14] S. Käbisch, D. Peintner, J. Heuer, and H. Kosch, "Optimized XML-based Web service generation for service communication in restricted embedded environments," in *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, 2011, pp. 1–8.

[15] 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/

[16] J. Cowan and R. Tobin. (2004) XML Information Set (Second Edition). W3C. [Online]. Available: http://www.w3.org/TR/xml-infoset/

[17] R. Kyusakov, H. Mäkitaavola, J. Delsing, and J. Eliasson, "Efficient XML Interchange in factory automation systems," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, nov. 2011, pp. 4478 –4483.

[18] D. Caputo, L. Mainetti, L. Patrono, and A. Vilei, "Implementation of the EXI Schema on Wireless Sensor Nodes Using Contiki," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, 2012, pp. 770–774.

[19] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 188–200. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031518

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

[21] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg, "Implementation of coap and its application in transport logistics," *Proc. IP+ SN, Chicago, IL, USA*, 2011.

[22] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A low-power coap for contiki," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*. IEEE, 2011, pp. 855–860.

[23] M. Kovatsch, S. Mayer, and B. Ostermaier, "Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things," in *Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), Palermo, Italy*, 2012.

[24] J. Schneider, T. Kamiya, D. Peintner, and R. Kyusakov, *Efficient XML Interchange (EXI) Format 1.0*, W3C Std., February 2014. [Online]. Available: http://www.w3.org/TR/exi/

[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] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[27] 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

[28] A. J. Korenjak and J. Hopcroft, "Simple deterministic languages," in *Switching and Automata Theory, 1966., IEEE Conference Record of Seventh Annual Symposium on*, Oct 1966, pp. 36–46.

[29] J. Berstel and L. Boasson, "Xml grammars," in *Mathematical Foundations of Computer Science 2000*. Springer, 2000, pp. 182–191.

[30] B. Chidlovskii, "Using regular tree automata as xml schemas," in *Advances in Digital Libraries, 2000. Proceedings. IEEE*, 2000, pp. 89–98.

[31] F. Neven, "Automata theory for xml researchers," *ACM Sigmod Record*, vol. 31, no. 3, pp. 39–46, 2002.

[32] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, "Taxonomy of xml schema languages using formal language theory," *ACM Transactions on Internet Technology (TOIT)*, vol. 5, no. 4, pp. 660–704, 2005.

[33] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, "Tree automata techniques and applications," Available on: http://www.grappa.univ-lille3.fr/tata, 2007, release October, 12th 2007.

[34] A. Brüggemann-Klein and D. Wood, "Balanced context-free grammars, hedge grammars and pushdown caterpillar automata." in *Extreme Markup Languages®*, 2004.

[35] R. A. Van Engelen, "Constructing finite state automata for high performance web services," in *IEEE International Conference on Web Services*, 2004.

[36] D. Wood, "Standard generalized markup language: Mathematical and philosophical issues," in *Computer Science Today*. Springer, 1995, pp. 344–365.

[37] Z. Shelby, K. Hartke, and B. C., *Constrained Application Protocol (CoAP)*, IETF Std., Dec. 2013. [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-coap-18

[38] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology*, vol. 2, pp. 115–150, 2002.

[39] K. Hartke, *Observing Resources in CoAP*, IETF Std., February 2013. [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-observe-08

[40] R. Kyusakov, "EXIP User Guide," EISLAB, Tech. Rep., 2013. [Online]. Available: http://exip.sourceforge.net/exip-user-guide.pdf

[41] Y. Fablet and D. Peintner, *Efficient XML Interchange (EXI) Profile for limiting usage of dynamic memory*, W3C Std., May 2014. [Online]. Available: http://www.w3.org/TR/exi-profile/

[42] R. Kyusakov, J. Eliasson, J. van Deventer, J. Delsing, and R. Cragie, "Emerging energy management standards and technologies - challenges and application prospects," in *Emerging Technologies Factory Automation (ETFA), 2012 IEEE 17th Conference on*, 2012, pp. 1–8.

[43] A. Petrick and S. V. Ausdall. (2013, April) Smart Energy Profile 2.0. ZigBee Alliance, HomePlug Powerline Alliance. [Online]. Available: http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Version20Documents.aspx

[44] R. Kyusakov, J. Eliasson, and J. Delsing, "Efficient structured data processing for web service enabled shop floor devices," in *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, june 2011, pp. 1716 –1721.

[45] P. Waher and Y. Doi, *XEP-0322: Efficient XML Interchange (EXI) Format*, XMPP Standards Foundation Std., Rev. 0.3, July 2013. [Online]. Available: http://xmpp.org/extensions/xep-0322.html

[46] G. J. Bex, W. Gelade, F. Neven, and S. Vansummeren, "Learning deterministic regular expressions for the inference of schemas from xml data," *ACM Trans. Web*, vol. 4, no. 4, pp. 14:1–14:32, Sep. 2010. [Online]. Available: http://doi.acm.org/10.1145/1841909.1841911

[47] J. Clark, "Trang - multi-format schema converter based on relax ng," 2013.

[48] J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1969.

[49] D. Peintner. (2013) EXIficient. [Online]. Available: http://exificient.sourceforge.net/

[50] T. Kamiya. (2013) OpenEXI. [Online]. Available: http://openexi.sourceforge.net/

[51] R. Kyusakov. (2014) Efficient XML Interchange Processor. LTU. [Online]. Available: http://exip.sourceforge.net/

[52] G. Moritz, F. Golatowski, C. Lerche, and D. Timmermann, "Beyond 6lowpan: Web services in wireless sensor networks," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 4, pp. 1795–1805, Nov 2013.

[53] F. Simeoni, D. Lievens, R. Conn, and P. Mangh, "Language bindings to XML," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 19 – 27, 2003.

[54] R. A. van Engelen and K. A. Gallivany, "The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks," in *2nd IEEE International Symposium on Cluster Computing and the Grid*, 2002, p. 128.

[55] S. Käbisch, D. Peintner, J. Heuer, and H. Kosch, "Efficient and Flexible XML-Based Data-Exchange in Microcontroller-Based Sensor Actor Networks," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*, 20-23 2010, pp. 508 –513.

[56] D. Sosnoski. (2014) JiBX: Binding XML to Java Code. [Online]. Available: http://jibx.sourceforge.net/

[57] ——, "Xml and java technologies: Data binding," *developerWorks-XML or Java Technology. IBM*, 2003.

[58] J. Arnoldus, M. van den Brand, A. Serebrenik, and J. Brunekreef, *Code Generation with Templates*, ser. Atlantis Studies in Computing. Atlantis Press, 2012. [Online]. Available: http://books.google.se/books?id=UvC0MJHSqjkC

[59] J. Eliasson, P. Lindgren, and J. Delsing, "A bluetooth-based sensor node for low-power ad hoc networks," *Journal of Computers*, vol. 3, pp. 1–10, 2008.

[60] Z. Shelby, S. Krco, and C. Bormann, *CoRE Resource Directory*, IETF Std., Sep. 2013. [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-resource-directory-00

[61] P. Puñal Pereira, J. Eliasson, R. Kyusakov, J. Delsing, A. Raayatinezhad, and M. Johansson, "Enabling cloud connectivity for mobile internet of things applications," *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, vol. 0, pp. 518–526, 2013.

[62] D. Peintner, H. Kosch, and J. Heuer, "Efficient XML Interchange for rich internet applications," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, 282009-july3 2009, pp. 149–152.

Paper C

An Authentication and Access Control Framework for CoAP-based Internet of Things

Authors:
Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Reformatted version of paper accepted for publication in:
Conference paper, IEEE IECON, 2014.

An Authentication and Access Control Framework for CoAP-based Internet of Things

Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Abstract

Internet of Things (IoT) and Cyber-physical Systems (CPS) are two very hot research topics today, and more and more products are starting to appear on the market. Research has shown that the use of Service Oriented Architecture (SOA) can enable distributed application and devices to device communication, even on very resource constrained devices, and thus play an important role for IoT and CPS.

In order to realize the vision of Internet of Things, communication between devices must be secured. Security mechanisms for resource constrained devices has attracted much interest from the academic community, where research groups have shown solutions like IPsec, VPN-tunnels, (D)TLS, etc. are feasible to use on this type of networks. However, even though the use of well-known security mechanisms are vital for SOA-based IoT/CPS networks and systems to be protected, they do not provide any fine-grain access control.

In this paper, a CoAP-based framework for service-level access control on low-power devices is presented. The framework allows fine grain access control on a per service and method basis. For example, by using this approach a device can allow read/write access to its services to one group of users while only allowing read access to another group. Users without the right credentials are not even allowed to discover available services. To demonstrate the validity of the proposed approach, several implementations are presented together with test results.

The aim is to provide a holistic framework for secure SOA-based low power networks comprise by resource constrain devices.

1 Introduction

The use of Service-Oriented Architecture (SOA) on resource-constrained devices has gain a lot of interest from both the academy as well from the industry in recent years as shown by [1, 2]. Service-oriented Architecture is based around the notion of services, formal interfaces and standardized protocols. A service is the core building block of SOA, and is piece of software performing some task, encapsulated with a formal interface described using some standard description format such as WSDL, and WADL in the case of Web services. A service must hold certain properties, such being discoverable, composable and loosely coupled from any operating system, programming language and other services. Since services are distributed in their nature, and relies on communication channels to

exist to function, they are inherently vulnerable for issues such as hackers, malware and other network based intrusion threats, for example denial of service (DOS) attacks. Services must therefore be protected using communication protocols using strong encryption and authentication, such as IPsec [3], SSL and other mechanisms. If Internet of Things and Cyber-physical Systems are to become mainstream technologies, used by millions of users, there must exist strong communication security and reliable authentication mechanisms. For example [4], Kasinathan et al. investigated mechanisms for detecting denial-of-service (DOS) attacks, which can be used to disrupt a network. 6LoWPAN is especially sensitive of DOS attacks due to the low bandwidth.

Regarding security, the utilization of IPsec (IP Security) over low-power networks was increased during the last years. IPsec has two modes of operation: the Transport Mode, which adds a Authentication header between the IP header and the UDP/TCP header, which allows the system to validate incoming packets but the original data is visible and accessible for all other devices on the network. The second mode is called Tunnel Mode, which is similar to the first one except that the IP header, the UDP/TCP header and the payload are encapsulated and encrypted (typically using AES) as payload of a new IP packet. This mode protects packets against eavesdropping attack, discards data modification and adds the possibility to detect Denial of Service attacks.

IPsec needs a shared password to encrypt and decrypt properly all incoming and outgoing messages. If these passwords are static could be compromised after some thousand messages. To solve this problem the IKE (Internet Key Exchange) and IKEv2 protocols were created. These protocols guarantee a safety communication between two devices and are able to create new shared passwords using circling derivative methods.

To protect UDP packets (even over IPsec), there is another protocol that can be used to add an extra layer of protection called Datagram Transport Layer Security (DTLS) [5]. This protocol uses a initial handshake to set the passwords. After that the content of the UDP packet is encrypted (usually with TLS_PSK over AES) and a header of 13 bytes is added, together with the initialization Vectors (IV) (over 8 bytes for AES128), integrity values (8 bytes) and the padding required by the cipher suite. DTLS increases the size of the packet, but this is a consequence of the packet encapsulation. This protocol is fully integrated in the CoAP protocol [6].

However, even though the use of well-known security mechanisms such as IPsec, VPN-tunnels, SSL, (D)TLS etc are vital for SOA-based networks and systems to be protected, they do not provide any fine-grain access control mechanisms. For example, if computers are exchanging data using a SOA-enabled protocol such as CoAP, only the packets are protected from external tampering and access. Any request from a client already inside a protected tunnel will be accepted by the server. The use of DTLS-encrypted CoAP could be one way of achieving a per service access control, but this would require a large number of different key-pairs to be in use in order to enable a true access control where a client can have different security access to different services, and even access to different operations on one service. For example, a client can have GET permissions to an actuator in order to view the status, but might not be allowed to actually perform a change of state using PUT or POST on that service. If only a few DTLS keys would

be used, then fine-grain access in not possible. If a very large number of keys are used, then fine-grain access control is possible but the management and administrating of key exchange mechanisms would be difficult.

A better approach is to separate the access control from the communication security. This would increase the security since another layer of protection is added. A client would also only need a few keys for the IPsec ad DTLS encryption, and then use a authentication service to gain access to other services. This makes administration easier since all access rights are centralized in the authentication service. Two access control protocols that use this approach are Kerberos [7] and RADIUS [8]. Kerberos is an authentication protocol which works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. RADIUS is often used for network authentication in wireless domains, and supports Access control, Authentication and Accounting (AAA).

This paper proposes a CoAP-based framework that solves the problem of a fine grain access control, which is not possible with other connection control system like IPsec and DTLS. The framework is focused on low overhead on resource-constrained devices that are commonly used in network for Internet of Things and Cyber-physical systems. The proposed solution uses ideas for other access control systems like Kerberos and RADIUS , and merge the two with the CoAP protocol to get a reliable access control framework for IoT.

This paper is structured as follows: Section 2 presents the background and related work, followed by a presentation of the proposed architecture in Section 3. After comes Section 4 which provides a detailed presentation of the authentication process, followed by a security analysis in Sec. 5. Section 6 outlines the performed experiments and results. Finally, future work and the paper's conclusions are presented in Sections 7 and 8, respectively.

2 Background and Related work

In this section, the background and the reason of this work are described, with a CoAP protocol description and some authentication protocols, and methods that were a base for the proposed framework.

Industrial networked devices and security

Industrial usage of networked for automation has been around for a long time. The industry is healthy with expected growth of 7% or more [9]. The projected big numbers of connected devices [10] indicate an even more rapid growth in networked devices for industrial usage automation. The discovery of the Stuxnet virus [11] and the information from the whistle blower Edward Snowden opened the eyes of the industry and general public about that any connected device might be vulnerable to Internet and electronics security issues. Currently much effort is devoted to prevent and protect against cyber attacks on networked devices. This certainly is true for Internet of Things.

In the field of networked resource constrained devices certain protocols are gaining popularity. Some of the most interesting ones, from the security point of view, are briefly reviewed below.

CoAP (Constrained Application Protocol)

The IETF Contrained Application Protocol is an application-layer protocol designed to provide web services working with constrained nodes. The protocol is designed for low-power networking. CoAP provides a request/response interaction model between application end-points, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs, RESTful interaction, extensible header options, etc. CoAP easily interfaces with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead and simplicity for contrained environments. CoAP uses UDP unlike HTTP. Some features of CoAP are:

- Two types of request messages: Confirmable Message (CON) - the message is retransmitted (four times maximum) with an exponential time out waiting for an Acknowledged Message (ACK) or the correct response form the server. The second type is the Non-Confirmable Message (NON) - the message is sent without any kind of response.

- The URI format allows the use of standard and specialized service endpoints. One for example is the resource discovery defined in RFC 5785 [12] that uses the *.well-known/core* path and the CoRE link format.

- CoAP also allows to send very big messages with a stop-and-wait mechanism called "blockwise transfers" (splitting messages and sending them with a reference order).

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
V		T		TKL				Code								Message ID															
Token (if any, TKL bytes) ...																															
Options (if any) ...																															
1	1	1	1	1	1	1	1	Payload (if any) ...																							

Figure 1: Original CoAP packet format.

The CoAP packet format (see Figure 1) has a maximum length of 1400 bytes, but the header has a length of 32 bits (2 for the version control, 2 for message type, 4 for token length, 9 for the message code and 16 for the message ID).

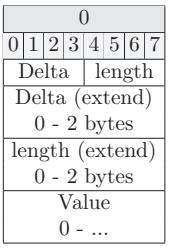


Figure 2: CoAP option format.

Kerberos

Kerberos [7] is a protocol that uses a primary communication between the client and the Authentication Server (AS) to generate a valid ticket. This ticket will be used for future accesses to Service Servers (SS). There are Kerberos implementations that run over UDP or TCP. Also the Ticket generation process could include different encryption methods, everything is flexible and configurable by the network administrator.

RADIUS (Remote Authentication Dial In User Service)

RADIUS [8] is a networking protocol to provide Authentication, Authorization and Accounting management centralized in a single server. This protocol offer the possibility to configure a single Network Access Server (NAS) into a user specific NAS. The use of it is widely used. It was designed in 1991 and allows many different types of configurations, but always work over UDP. This protocol supports Challenge responses (as PAP and CHAP) increasing the security against Eavesdropping attacks and also supports the use of certificates like X.509 [13]. The RADIUS packet format is shown in Figure 3.

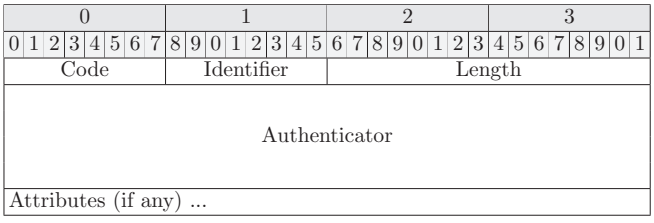


Figure 3: Original RADIUS packet format.

There are only five possible fields: Code field (to identify request/response type), Identifier (to identify each packet), Length (to know the packet size), Authenticator and Attributes. The first four are always required, but the last one is optional. The

Authenticator is a field of 16 bytes that is used to send encrypted data only to trust the communication between server and client. The encryption type is not predefined, originally this protocol was designed to use MD5 [14] but nowadays is not sufficient, therefore any other HASH generator could be used instead of MD5, like SHA-512 [15] or PBKDF2 [16]. After a request, there are three types of responses:

- Access Reject: the user has not access to any network resources. The reason could be a failure or a wrong identification.
- Access Challenge: to increase the security, the server could request extra information before trust on that user.
- Access Accept: The user has granted access.

The RADIUS protocol and format could run over CoAP protocol (see section 4), because all connection features can be possible also over CoAP. Also it is useful to authenticate devices over 6LoWPAN and enable the access to the network (see future work at section 7). The overhead of this protocol affects only to the authentication process (see section 4 and Fig. 6). Therefore it is reasonable to use in low-power and low band width networks.

Diameter

Diameter [17] is the evolution of RADIUS, the biggest modification is the use of TCP/SCTP instead of UDP. This change increase the security performance as much as TCP allows error handling, capability to negotiation and alive connections. But it also makes Diameter non-backwards compatible with RADIUS, and also with CoAP.

3 Framework

In this section all components of the proposed architecture are explained, with their ability to provide robust low power authentication and access control mechanisms.

Requirements

In order to get sufficiently fine grain access control, the framework must be designed optimal in terms of computation and overhead for access control and security over low power devices and networks with low width band. The communication to the access control service must be enable for all network devices, at least for valid devices. Is possible to filter the connection to access control service by IPsec and DTLS. The access control must be independent of other devices (there are no relation between two different accesses) and must be easy to automatize, due to most of these will be done by machines. The access rules and methods could be different depending on authentication polices, witch could change in base of local conditions, like time, power source, position, sensor

information, etc. The final objective is reach a fine-grain access control per service (that includes access methods and specific tags).

Scope

The aim of the proposed framework is to provide fine-grain and low-power authentication and access control mechanisms for resource-constrain devices, however the framework does not focus on communication security but relies on existing secure protocols such as IPsec and DTLS (see Fig. 4).

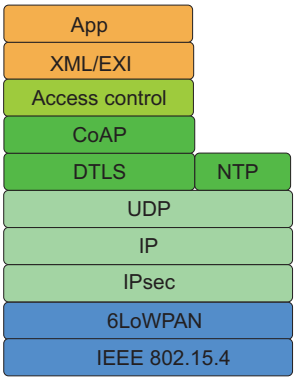


Figure 4: Communication layers

The lowest communication security level is based on IPsec (see Figure 4). The implementation of this IPsec layer is based on the open source implementation by Raza[18].

Access Control and Authentication Aspects

The proposed framework uses the best benefits of CoAP, Kerberos and RADIUS solutions to create a low-power platform for AAA. The propose architecture is shown Figure 5.

The Client must be a CoAP client ready to get the ticket after a login and use it in each future CoAP request. The AAA server checks the authentication requests and communicates the result to the CoAP server. CoAP server has the following requirements:

- Must check all the incoming packets looking for a CoAP Option [#100] called Ticket (Kerberos approach). This Ticket must validate the user and must give permissions to use the specific service by the user. All incoming request without Ticket will be discarded except Reset (RST) and Acknowledge (ACK) messages. Optional the [.well-known/core] could be also discarded to protect the network against automatic attacks.

- The authentication service [.well-known/auth] must exist on the CoAP server.

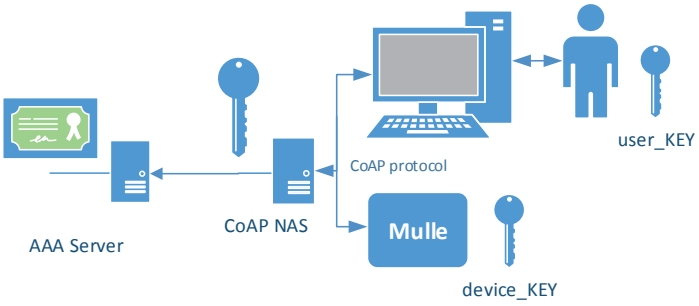


Figure 5: Simplified architecture

- Must have two different options [.well-known/auth?login] to do the login and [.well-known/auth?logout] to do the logout.
- If the login process fails, the server must send an error message. If the login process succeeded, the server must generate and send a new ticket together with the timeout.
- The logout process must delete the ticket on the server side and send a message to the user that the logout was successfull.

4 Authentication Process

The Authentication and Access Control consists of two different steps. The first step is a Authentication. In this step the system must recognize user as a valid user (with a password, a shared key or some other validator). This process will inform the CoAP-NAS about who the user is, permissions, group, time out of the ticket, etc. At the end of this process the CoAP-NAS must send the user a valid ticket but only if the authentication process is successful. In other case the server must inform the user that the authentication is not valid for the authentication process (incorrect user, incorrect password, black list, etc.). The second step is the Access control. On each client request, a valid ticket must be included to identify this request to the valid user, then the server will recognize the user and will respond with the correct message. If the client does not send the ticket or a valid one, the server will respond with an error message. The complete process is shown in a scheme in Figure 6.

Authentication Method

On the authentication process the server must recognize the user as a valid user and communicate that to the CoAP-NAS. This process needs to be flexible and compatible

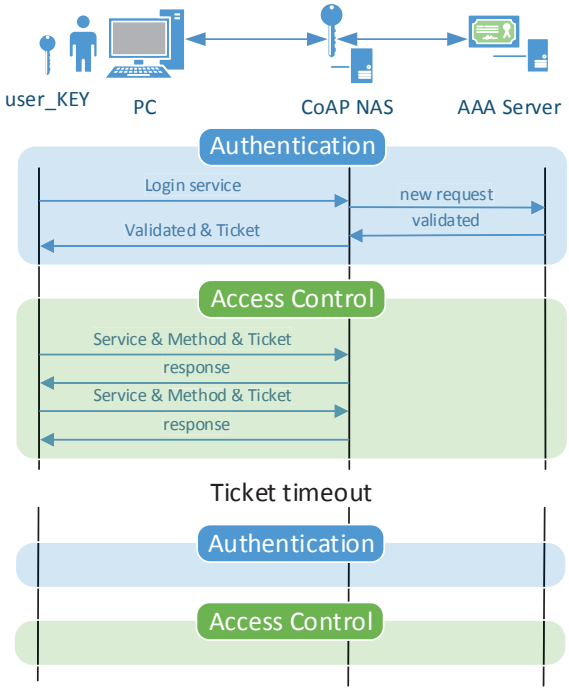


Figure 6: Authentication process

with other standards and with this goal the propose framework creates a public login CoAP service on the CoAP-NAS. This login service must receive a PUT request with one of the following contents as a payload:

- User name and password as plain text. This option is only recommended during testing, debugging and development phases.
- User name and password hash. This is easy to implement and could be authenticated directly on the CoAP server (without RADIUS).
- A RADIUS packet (future work).

The possibility to run RADIUS protocol over CoAP (see section 2) gives to the framework a flexible authentication method usable with a standard RADIUS server. This appaorach requires no RADIUS protocol on the client, then the overhead and the required resources will be smaller compared with the use of both protocols at the same time. This is especially important for resource-constrained sensor nodes. Therefore,

the conversion between RADIUS packet and CoAP-RADIUS packet is simple. In this framework there are proposed two alternatives, the first one is the most compatible with RADIUS standard, due to all of a RADIUS packet will be in the CoAP payload packet (Figure 7). The alternative option is omit any redundant information as much

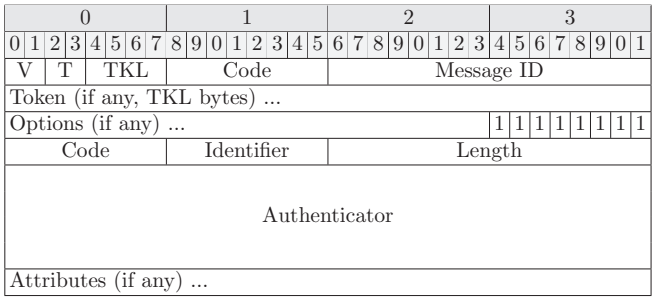


Figure 7: RADIUS over CoAP packet

as possible, thus deleting the Code, Identifier and Length in the RADIUS packet. This could be directly translated from the CoAP ID and Code (shown in Figure 8).

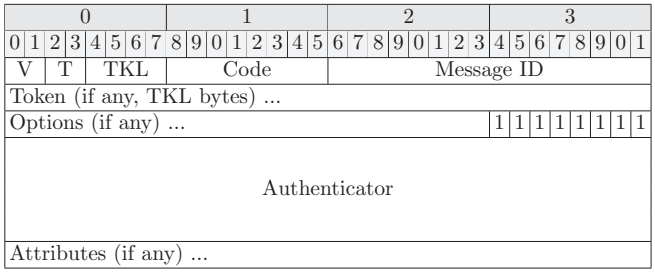


Figure 8: Compressed RADIUS over CoAP packet

Control Method

The servers could manage a lots of connections per second and CoAP protocol is focused on communication with low overhead packets. Then, the access control process must not increase too much the normal size of the packets. For this reason the propose framework would add a new option into the CoAP standard option set. This option called Ticket has the following properties:

- Unique per user and session. When a session expires the user must authenticate again and will receive a new one. The use of a time out will further increase the security.
- The ticket length could be configurable from 32 to 128 bits according to the requirements.
- Alternative to increase the security level, the first Ticket could be used as a seed to create a new one on each communication. This dynamic generations must be done with the use of hashing techniques.

On each incoming packet, the server must know from the CoAP packet the Message ID, the Ticket, the Service Name or URI to access and the Method. From the IP packet the server knows the IP address and the port number. For each authentication process the server must keep on memory the user name, the user password (or a hash), the IP address, the port number, the generated Ticket and the time stamp for time out. With all this information the server is able to recognize the user and check the permissions in a database. This process must done with the highest priority to detect a valid or non-valid ticket. At this point, if the ticket is valid, the server must send a normal response (according to the CoAP specifications). If the ticket is wrong or there is no ticket on the packet, the server must send an error message with the error code 406 (Not Valid) to inform the user that has not permissions to use that service. The propose framework is able to detect if a client is sending wrong tickets to ignore it.

5 Security Analysis

This section shows the security features of the proposed platform against different attacks.

1. Eavesdropping attack: IPsec and DTLS protect the system against this type of attacks. But to increase the security, the proposed platform could generate a new ticket for each message (plus hashing it with other parameters like Message ID), thus increasing the difficulty for a malicious user to predict a valid ticket.
2. Data modification attack: IPsec layer protects against this type of attacks. If the data is modified the Authentication Header of IPsec will detect that.
3. Man in the middle attack: IPsec will encrypt the data and will use time outs for each IP connection (with IKE), increasing the difficulty to guess the valid password to decrypt all packets.
4. Identity Spoofing attack: the use of false IP addresses is not going to work over the IPsec layer.
5. Denial of Service attack: there is no protection against this type of attacks. It is possible to decrease the overload discarding all dirty packets directly on the IP layer

(using IPsec), but finally if the attack is is severe the system will finally collapse (see [4]).

6. Application-Layer attack: this is considered the most difficult type of attack. In this case, the IPsec layer and (D)TLs are not going to protect the system. Here, only services with are under access control will be protected.

7. Replay attack: IPsec layer protect those attacks with the use of sliding windows.

The security features depend directly on the complexity of the encryption method and also on the length of the keys. The proposed framework do not need a specific cipher suite or a hash generator, but the security performance will depend directly of this selection.

6 Experiments and results

To perform the experiment, the server is based on libcoap 4.1.1 [19]. The source code of the server was modified to integrate the management of the users and the groups on each service with the permissions. The Copper (Cu) Firefox add-on [20] was used as a client. This enables the latest version of CoAP (draft 18) to be used, and has a user-friendly Graphical User Interface. The Copper plugin needed to be modified in order to integrate the ticket management and to add a visual user login menu (shown in Fig. 9).

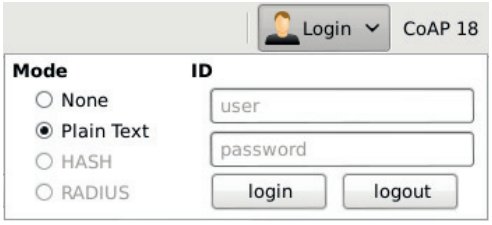


Figure 9: Copper login menu

To test the proposed fine-grain access control, several services were created with different permissions according to methods, user names and group names. Like other common access control systems, three users groups were define: unknown user, normal user and administrator user.

Screenshots taken during the experiments (see Fig. 10) show that the proposed access control mechanism work as expected. Services for non authorized users could be public (accessible for everybody) or could be some specific service for non authorized, like guest services that are not going to be accessible by authorized users. Services for authorized users could be exclusive for the user (User_name services), exclusive for the user group (Group_name services) or accessible services for authenticated users.

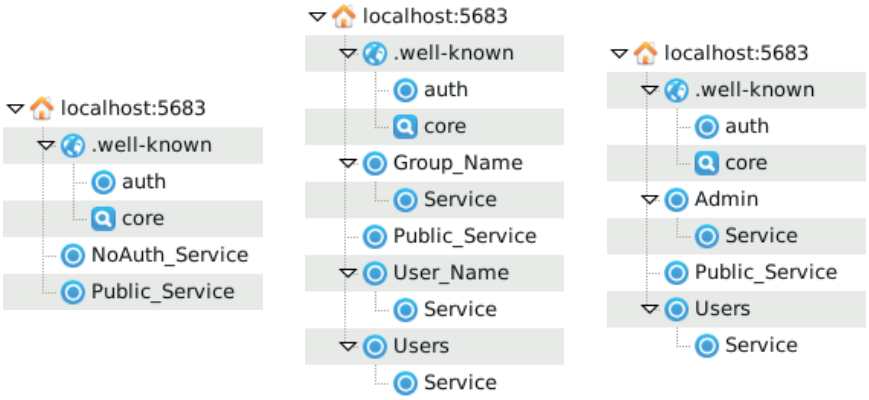


Figure 10: Screenshots of the available services for a non authorized user / authorized user / administrator taken during the experiment

Table 1 shows which packet types where an overhead is caused by the access control mechanism proposed in this paper. Depending on ticket size, the overhead (Ticket size) is in normal cases only 6-10 bytes.

	CoAP		CoAP+AA	
	request	response	request	response
GET	N	N	N+T	N
POST	N	N	N+T	N
PUT	N	N	N+T	N
DELETE	N	N	N+T	N
OBSERVE	N	N	N+T	N
ACK	N	N	N	N
RST	N	N	N	N
.well-known/core	N	N	N+T	N
.well-known/auth?login	-	-	I	N+T
.well-known/auth?logout	-	-	O	N

N: Normal size
T: Ticket size
I: Login request size (I>S)
O: Logout request size (O>=S+T)

Table 1: Normal CoAP message size vs CoAP+AAA

7 Future work

The use of a more Kerberos style access control mechanism would be beneficial, therefor the proposed framework will be improved with distributed mechanisms like Kerberos for service access control. The use of RADIUS for 6LoWPAN will also be investigated, i.e. to adopt the same type of mechanisms used by for example WiFi to 6LoWPAN to further increase a networks's security.

The use of a two-way ChallengeResponse mechanism such as CHAP for authenticating a client would improve the security even more since no password, or even a hash of the password, would need to be transmitted. This would never compromise a client's credentials, even in the case when a service has been compromised. By also enabling real-time monitoring of access attempts to services, it is possible to detect when a certain IP address or user name is trying to gain access with incorrect credentials. This intrusion detection system could then be used to notify network administrators about suspicious activity and even blacklist an IP address in for example a firewall. The use of a firewall on each sensor node would of course also be beneficial since rules can be dynamically adjusted to provide an extra security layer by e.g. only allowing certain IP address or address ranges certain access to services.

8 Conclusion

The use of the CoAP protocol has increased in popularity during the last years, especially over low-bandwidth links such as 6LoWPAN over IEEE 802.15.4. To secure the type of communication, there is the possibility to use IPsec and/or (D)TLS. These two protocols are able to protect the communication channel against some sorts of attacks, but regarding access control both IPsec and DTLS are not sufficient. That is, controlling the access to services by IP and/or session level only does not enable sufficient fine-grain access control. A user on one IP address might have full read and write access to one service, but only read access on another. By using existing methods there is no good way on achieving this. For that reason the proposed framework suggests a new CoAP Option to be used. This framework proposes uses a type of Ticket as well as defines packet formats to add the possibility to use standard protocol for authentication such as Kerberos and RADIUS. With these definitions this framework's CoAP extension enables a fine grain access control to CoAP-based servers and services.

In the performed experiments, a modified version of the CoAP C-library libcoap [19] was running as a server and a customized version of the Firefox plugin Copper [20] as a client, demonstrating that the proposed CoAP extension works and that fine-grain access control is able to know which IP address, user, services and method that are involved in a request, and send the correct response depending on the client's permissions.

The framework's access control mechanisms have been defined and successfully tested. In the next step, this concept will be converted into a more distributed mechanism like Kerberos, and to test the performance impact in terms of memory usage, power-consumption and packet overhead. Plans are also to add full support using decentralized

AAA servers.

9 Acknowledgment

The authors would like to express their gratitude towards our partners within the Arrowhead project, the European commission and Artemis for funding.

References

[1] S. Karnouskos and A. Colombo, "Architecting the next generation of service-based scada/dcs system of systems," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, Nov 2011, pp. 359–364.

[2] G. Candido, F. Jammes, J. de Oliveira, and A. Colombo, "Soa at device level in the industrial domain: Assessment of opc ua and dpws specifications," in *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, July 2010, pp. 598–603.

[3] S. Frankel and S. Krishnani, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," Internet Engineering Task Force (IETF), RFC Editor, RFC 6071, February 2011. [Online]. Available: http://tools.ietf.org/rfc/rfc6071.txt

[4] P. Kasinathan, C. Pastrone, M. Spirito, and M. Vinkovits, "Denial-of-service detection in 6lowpan based internet of things," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, Oct 2013, pp. 600–607.

[5] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," Network Working Group, RFC Editor, RFC 4347, April 2006. [Online]. Available: https://tools.ietf.org/rfc/rfc4347.txt

[6] Z. Shelby, K. Hartke, and C. Bormann, "Constrained application protocol (coap) draft-ietf-core-coap-18," Tech. Rep., June 2013. [Online]. Available: http://tools.ietf.org/search/draft-ietf-core-coap-18

[7] L. Zhu, K. Jaganathan, and S. Hartman, " The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2," Network Working Group, RFC Editor, RFC 4121, July 2005. [Online]. Available: http://tools.ietf.org/rfc/rfc4121.txt

[8] Y. Rekhter and T. Li, "Remote Authentication Dial In User Service (RADIUS)," Internet Requests for Comments, RFC Editor, RFC 1654, July 1995. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1654.txt

[9] "Monitoring and control: Today's market and its evolution till 2020," 2008. [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/necs/20081009-smart-2_en.pdf

[10] L. Gide, T. Koljonen, J. Lohstroh, A. ten Berg, and A. Foster, "Artemis strategic reseach agenda," Artemisis, June 2011.

[11] D. Kushner, "The real story of stuxnet," February 2013. [Online]. Available: http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet/

[12] M. Nottingham and E. Hammer-Lahav, "" defining well-known uniform resource identifiers (uris)"," Internet Engineering Task Force (IETF), RFC Editor, RFC 5785, April 2010. [Online]. Available: https://tools.ietf.org/rfc/rfc5785.txt

[13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Network Working Group, RFC Editor, RFC 5280, May 2008. [Online]. Available: http://tools.ietf.org/rfc/rfc5280.txt

[14] R. Rivest, "The MD5 Message-Digest Algorithm," Network Working Group, RFC Editor, RFC 1321, April 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1321.txt

[15] D. Eastlake and T. Hansen, "The MD5 Message-Digest Algorithm," Network Working Group, RFC Editor, RFC 4634, July 2006. [Online]. Available: https://tools.ietf.org/rfc/rfc4634.txt

[16] S. Josefsson, "PKCS 5: Password-Based Key Derivation Function 2 (PBKDF2) Test Vectors," Internet Engineering Task Force (IETF), RFC Editor, RFC 6070, January 2011. [Online]. Available: https://tools.ietf.org/rfc/rfc6070.txt

[17] E. V. Fajardo, J. Arkko, J. Loughney, and E. G. Zorn, "Diameter Base Protocol," Internet Engineering Task Force (IETF), RFC Editor, RFC 6733, October 2012. [Online]. Available: http://tools.ietf.org/rfc/rfc6733.txt

[18] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt, "Secure Communication for the Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN," *Security and Communication Networks, Wiley*, Jan. 2012.

[19] O. Bergmann. (2014, February) libcoap 4.1.1: C-implementation of coap. [Online]. Available: http://sourceforge.net/projects/libcoap/

[20] M. Kovatsch. (2013, November) Copper (cu) v0.18.2, firefox add-on. [Online]. Available: https://github.com/mkovatsc/Copper

Paper D

Translation Error Handling for Multi-Protocol SOA Systems

Authors:
Hasan Derhamy, Jens Eliasson, Jerker Delsing, Pablo Puñal Pereira and Pal Varga

Reformatted version of paper originally published in:
Conference paper, IEEE EFTA, 2015.

Translation Error Handling for Multi-Protocol SOA Systems

Hasan Derhamy, Jens Eliasson, Jerker Delsing, Pablo Puñal Pereira and Pal Varga

Abstract

The IoT research area has evolved to incorporate a plethora of messaging protocol standards, both existing and new, emerging as preferred communications means. The variety of protocols and technologies enable IoT to be used in many application scenarios. However, the use of incompatible communication protocols also creates vertical silos and reduces interoperability between vendors and technology platform providers. In many applications, it is important that maximum interoperability is enabled. This can be for reasons such as efficiency, security, end-to-end communication requirements etc. In terms of error handling each protocol has its own methods, but there is a gap for bridging the errors across protocols. Centralized software bus and integrated protocol agents are used for integrating different communications protocols; however these do not fit well in all Industrial IoT application scenarios.

This paper investigates error handling challenges for a multi-protocol SOA-based translator. A proof of concept implementation is presented based on MQTT and CoAP. Experimental results show that multi-protocol error handling is possible and furthermore a number of areas that need more investigation have been identified.

1 Introduction

The Internet of Things (IoT) has assisted in breaking down application domain silos and promoting horizontal integration between application domains. The Arrowhead framework, presented by Blomstedt et el. in [1] is looking to improve interoperability and integrability of services provided by networked embedded devices. Cisco has estimated that there will be 50 billion devices connected to the Internet by 2020 [2]. This is a staggering number of devices and managing the differing communication standards is not trivial.

The IoT area has seen many existing and new communications protocols emerging as preferred standards. The adoption of the varied communication protocols can be linked to specific application vertical requirements and is likely to stay this way as the IoT further develops. Thus in order for the Arrowhead framework to provide interoperability between application verticals, methods and technologies for communication protocol translation are required. Some of the IoT protocols used in SOA based applications and systems are; REST over HTTP, XMPP, MQTT, COAP and OPC-UA. Each of these

protocols offers benefits in particular application requirements, such as low-power operation, verbose headers and semantics, connection oriented messaging, decoupling producer from consumer, discovery, bootstrapping, real-time or reactiveness, and statelessness.

Today, there is a variety of commercial IoT platforms which support interaction between different communication protocols. They offer an API for either; translation agents [3][4] running embedded on the device or in gateways or a cloud based software bus [5][6] for each protocol. This indicates that there is a need to integrate different communications protocols.

However, these platforms either confine applications to using adapters integrated into their solutions, or require all communications be routed through a central server. Both approaches reduce flexibility for application designers and integrators, introduce security vulnerabilities with untrusted third-party clouds. This creates inefficiencies in the communications path and bandwidth usage for localized applications. Enabling protocol interoperability by the use of Service Oriented Architecture (SOA) will increase design flexibility, enable local applications and remove dependency on third-party translators. But Quality of Service (QoS), end-to-end connectivity, robustness and error handling become challenges which need to be addressed. A literature search did not reveal much research in multi-protocol translation. This indicates the need for more research in this area.

This paper investigates the question of error handling in a multi-protocol translation for SOA systems. While in a single protocol system, errors are propagated according to protocol specification. In the case of multi-protocol systems error handling becomes more complex. In designing a SOA based translator error handling and considerations becomes critical to robust communication. An error in one protocol must be translated to be understood by other protocols. While a SOA based translator must also address other aspects such as QoS, control messaging, security and semantic translation, these are not considered in this paper and are considered future work.

This paper is structured as follow: Section 2 provides background and related work, followed by problem definition in Section 3 and proposed solution in Section 4. An example application scenario and implementation details and results are presented in Sections 5 and 6. Finally, Conclusions are presented in Section 7, with suggestions for future work presented in Section 8.

2 Background and Related Work

A SOA-based architecture presented by Karnouskos et el. in [7] shows a shift towards SOA paradigm for Industrial IoT (IIoT). Development in the area of SOA is driven by the need for collaboration within Ultra-Large Scale systems [8]. SOA has been used to great effect in web based systems to create an ecosystem of collaborative parts.

The IoT is seeing growth in new application spaces and new application requirements with an evolving ecosystem of platforms, frameworks, protocols and devices [9]. This means system integrators are presented with the challenge of evolving their legacy systems, and technologies, to satisfy the new requirements and make use of new technologies.

By tightly coupling translation agents into the systems the cost of upgrading the system is increased. Relying on centralized cloud based software bus also limits the ability to leverage the native benefits of using new technologies.

Collina et el. in [10] have proposed an MQTT to REST bridge. This architecture exposes MQTT topics as REST resources. This allows MQTT clients and REST clients to interact through the new centralized QEST broker. There are two terms that appear in this field: protocol translation and protocol conversion. There is no clear differentiation between these terms, although "translation" is used generally in computer networking (especially and almost exclusively for network layer translation) - whereas "conversion" is used more widely in the industrial automation domain.

The traditional networking ISO-OSI terminology for nodes dealing with translation are that switches, routers and gateways work at the data link, network and transport layers, respectively [11]. The most widely known functionality is Network Address Translation, although its protocol translation version working between IPv4 and IPv6 (NAT-PT) was suggested for historic status [12] due to a series of serious issues.

There are no general guidelines or standards for higher level translation - these are seemingly all legacy solutions. Such translators do merely parameter mapping between two protocols, although sometimes also deal with the issues of the transport layers.

For this study two protocols used in similar application spaces were selected. This investigation helped to refine challenges error handling and proposal of solutions to address some of these challenges. CoAP and MQTT were selected as they are both intended to be used in highly efficient industrial applications.

CoAP

The CoAP protocol [13] has been developed by the IETF for use in extending Internet capability down to resource constrained devices. It applies the request-response communication pattern to a client-server network model. CoAP is targeting sleepy and lossy networks in which supporting TCP becomes inefficient and power consuming [14]. It is based on UDP and provides an optional retry mechanism at the CoAP layer. It has a RESTful API with the GET, PUT, POST and DELETE verbs supported with the addition of the OBSERVE function. It creates a publisher-subscriber session between a CoAP server and client, sending notifications either when resource state changes or periodically [13]. Having this flexibility makes it an ideal choice for machine-to-machine interaction.

MQTT

The MQTT protocol has been developed for enabling efficient communication between data sources and data sinks. It applies the publisher-subscriber pattern to a client-server network model. It has recently been standardized by OASIS but has a long history with IBM being used in sensor networks. Some of its features [15] are decoupling data producer from data consumer through the centralized broker system; reduced header size and event based publishing enable highly efficient communication; QoS levels with

message delivery; and, simple centralized security model with connection initiation by clients enabling useful firewall and NAT traversal features.

3 Problem Definition

Dependent on how much the protocols overlap in the OSI layers there can be much complexity in the translation process. Errors which can be detected and monitored need to be handled in a manner which will enable adequate debugging and issue resolution, either automated or by manual intervention.

In this section some of the challenges with handling errors of multi-protocol translation are elaborated and specifically, the case of translation between MQTT and CoAP studied. The error cases defined may not be exhaustive; however they represent some of the most common and in some cases challenging errors.

Error cases

Connection errors can occur when trying to establish new connections, having a current connection lost for some reason, or inability to close a connection gracefully. These kinds of errors need to be detected and translated appropriately to ensure efficient use of resources (at the end points as well as at the translator). Also information connection error events need to be made known for analysis on network performance and candidates for possible improvements.

Lossy communication errors are a real problem in wireless sensor networks (WSN), which make up a good proportion of the future IIoT. The problem with lossy communication is made more complex with layers handling the issue at different layers of the OSI stack. A higher level protocol may rely on a lower level layer to guarantee transport while the target protocol may perform such transport checking itself. This means that handling lossy communication at may need to go across layers or provide informational error alert which will then rely on application layers to monitor and perform corrective action if needed.

Response related delays and application introduced delays are two such categories of delay related errors. Miss-matched timeouts at the communications or application layers can lead to one sided timeouts. To handle one sided timeouts the channels need to be re-synched, how will the translator deal with this? Application delays could be found on resource constrained devices not being able to service a request or publish an update within the time limit expected by the other party.

Application layer packetization which relies on ordered delivery by underlying layers is a special case of moving between ordered delivery protocols such as TCP and un-ordered delivery such as UDP. Is this something which can be handled by the translator? Is this something which is required by the translator?

Invalid messages arriving at the translator requires the translator to be able to detect and take appropriate action. In request-response protocols an error message can be sent to the origin. However in many publish-subscribe protocols it is not possible to send

an error back The translator must be responsible for providing some level of confidence to the end points about how much of the protocol dependencies are still valid and how much cannot be support.

Errors produced as part of the protocol procedure such as authentication problems, resource availability or others can generate legitimate error codes. These error codes while represented on one side of the translator need to be passed to the other side. To address this first the error codes of each protocol must be listed and then mapped based on the cause. Often the mapping will not be symmetric with many error codes being mapped to a lesser number, or even some error codes not being able to be mapped at all. This is described in detail in Section 4.

Transient Error Cases

Transient error cases can occur at end points but also in the translator. This class of error cases relate to errors which can occur at any time and will generally 'heal' in a short period of time. They require special treatment in terms of handling and translation, in terms of maintaining protocol behavior.

Transient errors due to resource usage occur when there is an increase in translation demand to such an extent that the translator is no longer capable of handling the throughput. It must take remedial action in accordance with the protocols which are being translated. These actions are highly dependent on the source of the increased demand and the capability for the translator to influence this demand. This can be illustrated where two end-points have a contract for delivery of notifications on periodic updates. If the size of the messages begins to increase while the rate of the messages remains at a high frequency, the translation of these message packets may consume resources which are not available. In this case the source of the data cannot be asked to reduce the frequency as the contract is between the end points. In such a case the translator must become a negotiating party and reduce the frequency or the size of the messages. In some cases the protocol does not allow such freedom for negotiation and in fact requires the end-point to drop the connection in such situations, as is the case for MQTT [15].

The second transient error case is from buffer problems. Buffer problems can occur when resource usage increases without correct remedial actions. But in this case we are referring to miss matches in buffer sizes between protocols and end-points. Whilst one high performance end-point, such a REST based web application, may be able to handle large verbose messages, once this is translated for a constrained end-point, such as CoAP, an error will occur between the end-points and once again remedial action will need to be taken and the event will need to be logged.

Lastly, transient errors can occur due to miss-matches in protocol or application sensitivity to jitter or delays. An application or protocol which has been designed to be sensitive to a specific range of jitter or delay may find that this is not always possible to be met either because of the jitter and delays introduced at the translator, or because the other protocol cannot guarantee the same jitter and delay bands. This can impact either the protocol behavior or the application behavior and thereby generate errors in

the end-to-end process.

4 Proposed Solution

There is much work to be done in order to address the challenges in the previous section. As each challenge is tackled there is likely to be an increase in the solution complexity. This section looks to address error code translation and discuss QoS and Error reporting.

Mapping Error codes

In this paper the error mapping was done a single pair of protocols. This is the basis for the development of an error code mapper which would produce a generic interface which individual protocols must use to define their error codes to the map. In MQTT, protocol error codes are only reported from the broker to the client [15]. The protocol does not specify error code reporting by the client to the broker. This is illustrated in the Figure 1. CoAP in turn also only reports errors from the server to the client [13]. This unidirectional use of error responses means that in certain configurations the error codes cannot be passed to the end-points. This is discussed further below.

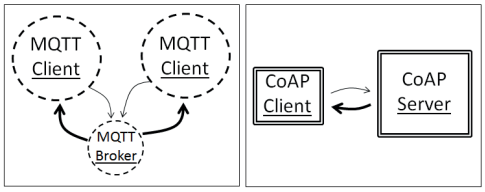


Figure 1: Direction of error reporting in MQTT and in CoAP protocols

The MQTT protocol defines error codes for initial connection and subscription control packets, other control packets do not have associated error codes [15]. The decoupled nature of MQTT networks means that clients have much less visibility of errors occurring in other clients.

Below are two tables with the error codes producible in MQTT and in CoAP. Table 1 shows the error cases which are generated by MQTT and mapped to CoAP. The mapping in this case is used when a CoAP client is attempting to initiate a subscription to an MQTT broker through the translator. This is illustrated in Figure 2.

In this case error codes generated by the MQTT broker can be translated and passed to the CoAP client. For example the CoAP client will be awaiting the response to its GET request and if the MQTT broker does not allow the connection or the subscription, then this error can be passed to the CoAP client.

In Table the CoAP error codes are mapped to MQTT. These represent the case when an MQTT client subscriber is attempting to retrieve data from a CoAP server. This case

MQTT error case	MQTT code	CoAP
Failure – Topic filter not accepted	SUBACK 0x80	Error 4.04
Not Authorized	CONNACK 0x05	Error 4.01
Bad username or password	CONNACK 0x04	Error 4.01
Server not available	CONNACK 0x03	Error 5.03
Identifier rejected	CONNACK 0x02	Error 4.00
Unacceptable protocol version	CONNACK 0x01	Error 4.06

Table 1: Error code mapping from MQTT broker to CoAP client

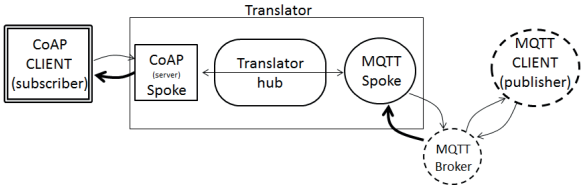


Figure 2: Configuration one allows direct translation of some error codes

is illustrated in Figure 3.

CoAP error case	CoAP code	MQTT
Bad Request	Error 4.00	Not implemented
Unauthorized	Error 4.01	
Bad Option	Error 4.02	
Forbidden	Error 4.03	
Not Found	Error 4.04	
Method Not Allowed	Error 4.05	
Not Acceptable	Error 4.06	
Request Entity Too Large	Error 4.13	
Unsupported Media Type	Error 4.15	
Internal Server Error	Error 5.00	
Not Implemented	Error 5.01	
Bad Gateway	Error 5.02	
Gateway Timeout	Error 5.04	

Table 2: Error code mapping from MQTT broker to CoAP client

In this case there is no path for the error codes to be transferred to the subscribing MQTT client. This is due to the nature of the MQTT protocol; and so the CoAP error codes do not have a mapping to MQTT clients.

However, the error codes can be used by the translator and can be mapped to transla-

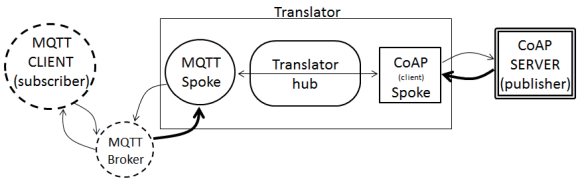


Figure 3: Configuration two does not allow direct translation of error codes

tor behaviors. That is, for translating between two different protocol pairs, there will be different behavior by the translator to take corrective actions or logging. This could be translating the error code as in Table 1, or other remedial actions as defined in the translator. For the use case implemented in this paper the translator actions are described in Section 6.

Quality of Service aspects

Since the translator is in the path between the service producer and the service consumer application systems, its performance affects the end-to-end QoS. The translator, as a physical entity has resource limitations for memory and processing power.

Furthermore, it can serve very different application needs and very many of those. Which means it will handle many queues, which fragment those limitations further: allocating memory for the various queues, for each entity in the queue; and handling the processing overhead due to the handle queuing mechanisms (i.e. scheduling).

The translator has similar types of QoS-related issues as a network-level processing node (i.e. router); although these are somewhat enlarged. This is due to the differences in information volume: translator needs to process application payload, whereas a router merely processes the network layer header.

It is not only that the translator (that handles various types of service needs) should handle QoS profiles, but these should describe further detailed metrics than those well-known at the network level.

Besides handling loss, delay, delay and utilization metrics, their more specified versions [16] should be kept under control: one-way and two way throughput and delay, as well as their variance. Availability as a QoS metric is hard to address other than binary terms - either it is available, or not. Loss as a quality metric gets another meaning here - loss in translation - where not the whole message, but its parts get lost. Depending on the context and the parameters that weren't able to be mapped, may lead to QoS degradation - or it may have no noticeable effect.

Error reporting aspects

One of the most challenging aspects of distributed systems is error diagnosis. The IoT promises massively distributed systems and with the introduction of cross protocol trans-

lation the error cases not only increase in number but also in complexity. Therefore error logging and reporting is critical to the success of a translation system.

Systems can take an active approach towards error monitoring which will mean that the error notification may have soft real-time requirements. While other systems may take a reactive approach to error monitoring which will mean that they will require persistent log of the error events leading up to the final event which caused the investigation. This is true for all distributed systems with or without translation, but as the translation is a third party to the two end points involved in most interactions it becomes pertinent to discuss the implications. This means that the error stream reported by the translator must maintain a link with the end points it is translating for.

Either one of the end-points being translated or a third application will need to be able to securely and efficiently query the error log. This introduces challenges in authenticating and authorizing the interested parties to access the logs and to then find the relevant error records. If the error logs need to be exposed to a third-party which is providing support, then how can they be authorized to access the required logs?

5 Application Scenario

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 [1]. 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. The translation scenario selected for demonstrating the challenges of error handling within the framework of Arrowhead is a CoAP based sensor network monitoring the condition of the wheel loader's ball bearings and providing this data as a service within the Arrowhead framework. See Figure 4 for a layout of the network architecture of Arrowhead Task 1.8.

An MQTT based service consumer is behind a firewall and initiates a session with the broker consuming the sensor data from the CoAP based sensor. The MQTT service consumer could be a head office system which does not allow incoming UDP packets or a hand held device running a VPN which again does not allow incoming UDP packets. In this case CoAP is not suitable as a service consumer. A high level diagram of this scenario can be seen in Figure 5.

The translator must connect the MQTT broker and the CoAP server to allow data flow. There are many error conditions possible, as stated in Section III. As a proof of concept the authors have chosen to detect a sensor disconnect at the wheel loader. In an industrial environment such as mining, road works or construction sites a disconnection error is something which the translator must be able to handle. The interaction diagram between the different components is shown in Figure 6.

Of particular interest in this scenario is the nature of CoAP running on UDP which is connectionless and therefore requires an agreed timeout based approach to connection loss. While on the MQTT side TCP is used and therefore a connection state is maintained

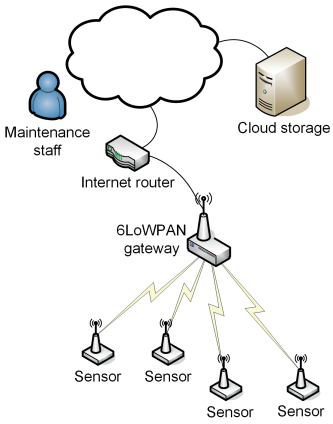


Figure 4: Wheel loader monitoring architecture

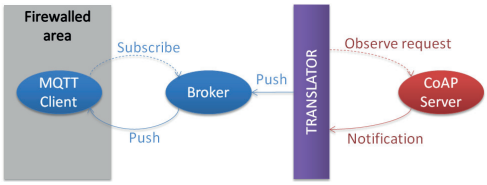


Figure 5: Application scenario requiring translation

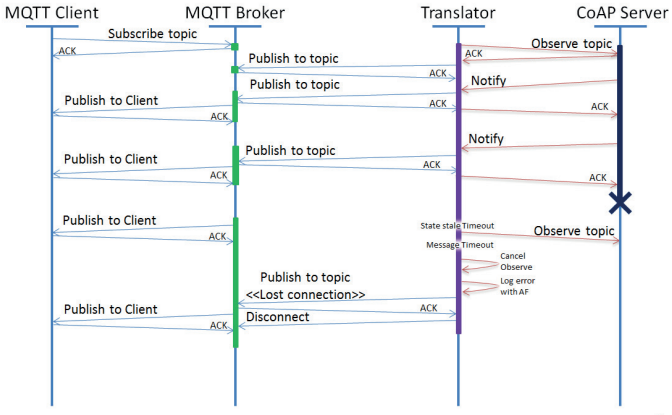


Figure 6: Error condition interaction diagram

between the end points. Even so not all TCP disconnections can be identified and so if the detection of a disconnection is desired, a heartbeat or a keep-alive timeout is also

required.

6 Implementation and Results

In order to validate the error case assumptions and begin the process of identifying limitations in error handling of the translation process an error translation scenario has been implemented. This section will describe the implementation setup and the results of running the error cases.

Eclipse Paho MQTT client library has been used for developing a simple visualization of data and events. This was connected to a Mosquitto MQTT broker running on a standard Windows computer on a loop back network. The wheel loader sensor was taken from the Arrowhead task 1.8 pilot project and it uses Contiki OS 2.7 and Erbium. The sensor connected to the translator through a Contiki border router and a BeagleBone Black gateway. The translator was implemented in Java and uses a Californium [17] CoAP client to initiate an observe on the state of the wheel loader sensor and using a hub and spoke architecture passes the resource notifications to an MQTT Paho client which then publishes the notification to the corresponding MQTT topic in the Mosquitto broker. This setup is shown in Figure 7 below.

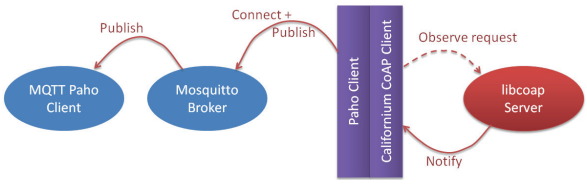


Figure 7: Implementation technologies mapped to system components

The translator itself is not the core of this paper and so the implementation has been kept to a simple transfer of payload from CoAP to MQTT. Semantics and other protocol procedures have not been considered. Using a hub and spoke architecture for implementation, results in a decoupled component based translator with only simple object method calls being setup by the hub between the spokes. This can be seen in the Figure 8.

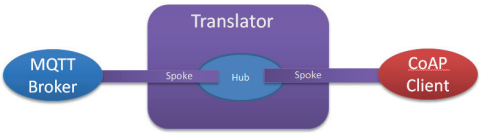


Figure 8: Translator architecture diagram

Running the experiment uncovered several error cases which were in addition to the disconnect error case that was to be modeled. This unintended error case was very useful as it shows a real world use case. The wheel loader sensor is a resource constrained device running on a low power network and therefore notification timeout due to late delivery or packet loss was common. In these cases the CoAP spoke would follow the CoAP specification and on max-age expiry, would attempt to re-register the observation. This was in almost all cases successful and would re-establish the periodic notifications. However in the MQTT specification there is no mechanism within the protocol to pass information regarding update timeout except by disconnection. The keep-alive timer was controlled by the Paho library and so would keep the connection alive even when no data was being sent. This means that a non-standard message would need to be sent from the CoAP spoke to inform the MQTT client that the sensor has had a timeout. Processing of this message would be at the discretion of the MQTT client application.

However, in the event of a disconnection error which does have protocol procedures in both MQTT and CoAP there is still special behavior required. So the CoAP spoke monitors the max-age of the last resource state update. If this max-age is exceeded then a timeout is noted and the CoAP spoke will attempt to re-establish the observation, as described earlier. However if the re-establishment is not successful then the CoAP spoke cancels the observation and an error passes an error event to the translator hub. In a normal setup an ungraceful disconnection detected by the MQTT system, would result all subscribing clients being delivered a last will message, if it is available. However in this case the MQTT system does not have visibility of the CoAP disconnection. It is required to translate and notify the MQTT system of this disconnection event. The proposed translator maps the CoAP disconnection to a last will message in the MQTT side. This mapping takes place in the translation hub. Figure 9 shows the interaction between internal components of the translator system.

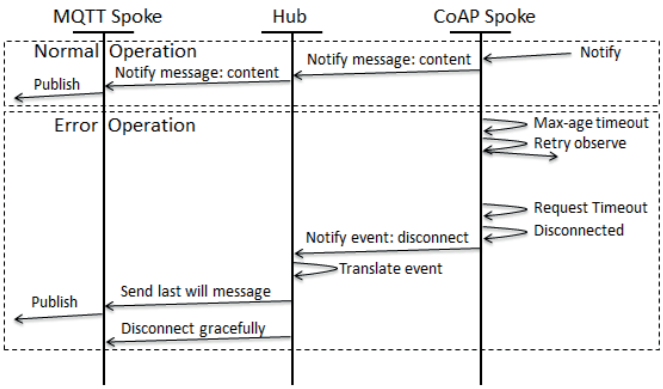


Figure 9: Internal object interaction of the translator behavior

In this way, the translator has made use of the protocol procedures of both sides to make sure both protocols are aware of the error event. For interoperable use of the translator between systems the definition of both the timeout event and the disconnect event signals is a must. For this implementation two event signals were defined and encoded in XML. These tags are presented below.

Implementation of the rudimentary good path payload translation between CoAP and MQTT was relatively trivial. However implementing the mapping of disconnection and timeout errors introduced a lot of complexity to the code. By refining the implementation the error handling and mapping were moved to the central hub. This reduced the complexity of the translation effort immensely. Each translation spoke did not need to have knowledge of the other. This means that once a spoke is developed it can be connected through the hub to any other spoke. This reduces the effort required to develop translation services between the ever changing array of protocols.

By decoupling the protocol specific handling to spokes and translation aspects to the translation hub has meant that the solution is extensible for new spokes and allows interesting potential for a multi-spoke translator. It has also meant that error cases can be handled in a standard manner within the translation hub and new spoke development need only use available hooks in the translation hub in order to pass error conditions.

The results were promising with key advantages to the use of a hub and spoke SOA based translator. Its active participation in the network, its simplicity for handling errors and potential for extension to being orchestrated and also into semantic translation are the main advantages.

7 Conclusion

This paper has presented the challenges and solution for error handling in multi-protocol translation scenarios for SOA systems. This work is motivated by the creation of new systems-of-systems that are composed of application domains with different communications requirements. Current protocol translation solutions use tightly-coupled software components or integrated middleware that reduces flexibility and increases cost of change. Moreover, utilizing centralized software bus for translation increases round trip time, bandwidth usage and introduces further dependencies (i.e. on cloud platforms, often operated by third parties). In both of these cases intermediary protocols are used and this limits the benefits of the native communications protocol.

On the other hand, SOA-based translation systems provide the opportunity to decouple the translation components from the application development and also create flexibility in deciding execution location of the translation service.

This paper discussed the challenges of error handling in the case of loosely coupled SOA translators. Some of the investigated error cases are connection errors, lossy communication, application introduced delays and protocol error code mapping. Beside these, the transient errors in the translator and at end-points need to be handled. This means that not just message parameter mapping, but the protocol procedures of one side needs to be reflected on the other side. Transient errors can occur, when resource requirements

in terms of memory and processing power do not scale as usage demand increases, a mismatch in buffer requirements between a protocol pair, or a mismatch in jitter and delay sensitivity. These transient errors require the translator to be capable of self-monitoring and also negotiation capabilities with the protocol pairs.

The proposed solution uses translator behaviors, which are then mapped to a protocol procedure or error code. The translator's overall behavior depends on the protocols being translated. This is realized in the implementation by the use of a hub and spoke architecture with the hub containing the possible behaviors of each spoke.

The proof of concept implementation provided error handling for translation between CoAP and MQTT. To accomplish this, it was required to pass error cases generated on the CoAP side of the connection to be communicated to the MQTT side. It was decided to define a set of xml tags and attributes, which would communicate the errors at the application level. There were two such events defined for this proof of concept, they were the disconnect event and the timeout event.

8 Future Work

In the future, the architecture of the multi-protocol translator needs to be defined and refined. Use case extension to other SOA protocols such as XMPP and REST will be needed.

Challenges are seen in orchestration and co-ordination of the translator end-points, managing resource requirements, providing security in terms of privacy, confidentiality and authenticity, and proving performance and flexibility gains.

Performance metrics, evaluation and bench-marking will be needed in order to prove the advantages of a multi-protocol SOA translator.

Further development of the semantics used to send error information and signals should be looked into.

Error logging and diagnosis has much work to be done. Logging encompasses a larger scope than just error events should enable SOA based applications to create an end to end stream of events. An API must be developed with the ability for machine query and manual query of the logs.

9 Acknowledgment

The authors would like to express their gratitude towards the European Commission and Artemis for funding, and our partners within the Arrowhead project.

References

[1] F. Blomstedt, L. Ferreira, M. Klisics, C. Chrysoulas, I. Martinez de Soria, B. Morin, A. Zabasta, J. Eliasson, M. Johansson, and P. Varga, "The arrowhead approach for

soa application development and documentation," in *Industrial Electronics Society, IECON 2014 - 40th Annual Conference of the IEEE*, Oct 2014, pp. 2631–2637.

[2] D. Evans, "The internet of things how the next evolution of the internet is changing everything," in *The Internet of Things How the Next Evolution of the Internet Is Changing Everything*, April 2011.

[3] "Cumulocity," http://www.cumulocity.com/guides/concepts/interfacing-devices/, April 2014, interfacing devices.

[4] "Iotivity," https://www.iotivity.org/documentation/iotivity-services/protocol-plug-manager/, April 2014, protocol plug-in manager.

[5] "Ptc," http://www.thingworx.com/, April 2014, thingworx platform a ptc business.

[6] "Xively," https://personal.xively.com/dev/docs/api/, April 2014, xively rest api.

[7] S. Karnouskos, A. Colombo, T. Bangemann, K. Manninen, R. Camp, M. Tilly, P. Stluka, F. Jammes, J. Delsing, and J. Eliasson, "A soa-based architecture for empowering future collaborative cloud-based industrial automation," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 5766–5772.

[8] R. P. G. J. G. R. L. T. L. R. K. M. K. D. S. K. S. L. Northrop, P. Feiler and K. Wallnau, "Ultra-large-scale systems - the software challenge of the future," http://www.sei.cmu.edu/Iibrary/assets/ULS_Book20062.pdf, Juny 2006, software Engineering Institute, Carnegie Mellon, Tech. Rep.

[9] P. Suresh, J. Daniel, V. Parthasarathy, and R. Aswathy, "A state of the art review on the internet of things (iot) history, technology and fields of deployment," in *Science Engineering and Management Research (ICSEMR), 2014 International Conference on*, Nov 2014, pp. 1–8.

[10] M. Collina, G. Corazza, and A. Vanelli-Coralli, "Introducing the qest broker: Scaling the iot by bridging mqtt and rest," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, Sept 2012, pp. 36–41.

[11] B. Singh, *Data Communications And Computer Networks 2Nd Ed.* Prentice-Hall Of India Pvt. Limited, 2006. [Online]. Available: https://books.google.se/books?id=RsocJmhDBcIC

[12] C. Aoun and E. B. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status," IETF RFC 4966, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc4966.txt

[13] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

[14] C. Bormann, A. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *Internet Computing, IEEE*, vol. 16, no. 2, pp. 62–67, March 2012.

[15] "Mqtt version 3.1.1," http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, 2014, oASIS Standard.

[16] P. Varga and l. Moldovan, "Integration of service-level monitoring with fault management for end-to-end multi-provider ethernet services," *Network and Service Management, IEEE Transactions on*, vol. 4, no. 1, pp. 28–38, June 2007.

[17] M. Kovatsch, S. Mayer, and B. Ostermaier, "Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things," in *Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012)*, Palermo, Italy, Jul. 2012.

Paper E

Configuration Service in Cloud based Automation Systems

Authors:
Oscar Carlsson, Pablo Puñal Pereira, Jens Eliasson, Jerker Delsing, Bilal Ahmad, Robert Harrison and Ove Jansson

Reformatted version of paper originally published in:
Conference paper, IEEE IECON, 2016.

Configuration Service in Cloud based Automation Systems

Oscar Carlsson, Pablo Puñal Pereira, Jens Eliasson, Jerker Delsing, Bilal Ahmad, Robert Harrison and Ove Jansson

Abstract

Current challenges in production automation requires the involvement of new technologies like Internet of Things (IoT), Systems of Systems and local automation clouds. The objective of this paper is to address the actual process of defining a cloud based automation system. The objective of this paper is to address one of the challenges involved in establishing and managing a cloud based automation system. Three key capabilities have been identified as required to create the expected benefits of local automation clouds; 1) capturing of plant design 2) capturing and distributing configuration and deployment information 3) coordinating information exchange

This paper addresses the capturing and distribution of configuration and deployment information. For this purpose a system service is proposed, the ConfigurationStore, following the principles of the Arrowhead Framework. The service is accompanied by a deployment methodology and a bootstrapping procedure. These are discussed for several types of automation technology, e.g. controllers, sensors, actuators. A qualitative evaluation of the proposed approach is made for four use cases; Building automation, Manufacturing automation, Process automation and IoT devices. Concluding the usability for large-scale deployment and configuration of Industrial Internet of Things.

1 Introduction

High level topics in today's society are sustainability, flexibility, efficiency and competitiveness. These in turn are driven by big societal questions like environmental sustainability, availability of energy and raw material, rapidly changing market trends. We find several trends that in different ways are addressing these topics. One is the move from large monolithic organisations towards multi-stakeholder cooperations where cooperation are fostered by market requirements. Another is the learning from previous products, other parts of the value chain, the life cycle of the product and the product or service production process itself.

These trends are creating new requirements for the technology used to support product and service production, causing a drive for digitisation of production. Digesting this reveals a number of gaps regarding technology, organisation, cooperation structure, operational management and related business models that have to addressed.

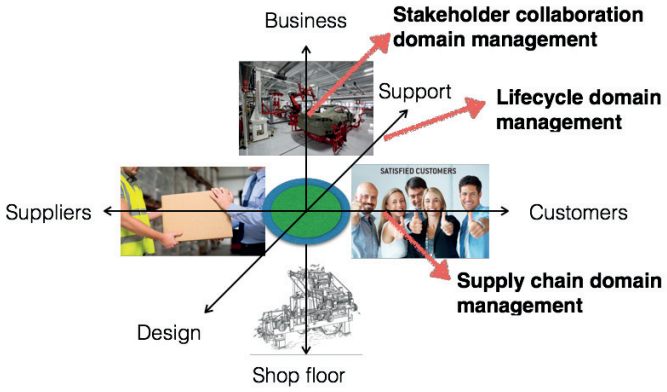


Figure 1: Three important axes for collaborative production, product life cycle, supply chain and stakholder integration management

Around organisation, cooperation structure and operational management the high profile key aspects are related to three domains, see Figure 1:

- Product life cycle management
- Supply chain management
- Stakeholder integration management

With the move from large monolithic enterprises towards multi-stakeholder cooperation the management is changing towards distributed multi-stakeholder collaboration with distributed responsibilities and decision making. The flexible collaboration along and in-between the three domains also opens for the possibility of dynamic learning. A further aspect is that these domains tend to become wider (longer) involving more stakeholders with diverse objectives and more details and variations of the service or product to meet customer diversity and service and product quality.

These ideas are currently emerging but regarded as very important to address the high level topics of flexibility, efficiency, competitiveness and with suitable incentives also supporting sustainability. To support these developments there are a number of technology gaps which seemingly can not be addressed by current state of the art. Because of this, a number of new technologies are emerging to fill these gaps. Some current big buzz technologies are:

- Internet of Thing, IoT
- System of Systems, SoS
- Cyber-Physical Systems, CPS

- Service Oriented Architecture, SOA

Despite all the new operational and organisational ideas and emerging technologies the automation fundamentals captured by today's state of the art automation technology have to maintained. Thus, next generation of automation and digitalisation technology has to meet a large set of requirements and involve a wider scope of actors and stakeholders. This is the big challenge for technology suppliers of the future, in this field.

ISA-95, standardised through ISA [1], is today's standard architecture for automations systems [2]. Accompanying the ISA-95 standard are related standards like ISA-99 and IEC 62443 [3, 4], which address control system security.

In 2011, the concept of Industry 4.0 [5] was born in Germany. This concept builds upon the last generation of industrial monitoring and control systems, but enables an even finer level of interaction between shop-floor devices and high-level enterprise systems. In industry 4.0, state of the art technologies like Internet of Things (IoT) and Cyber-physical Systems (CPS) are utilized in order to be able to break the classical, strictly hierarchical, approach of ISA-95 [2] with a more flexible approach without fixed barriers and closed systems.

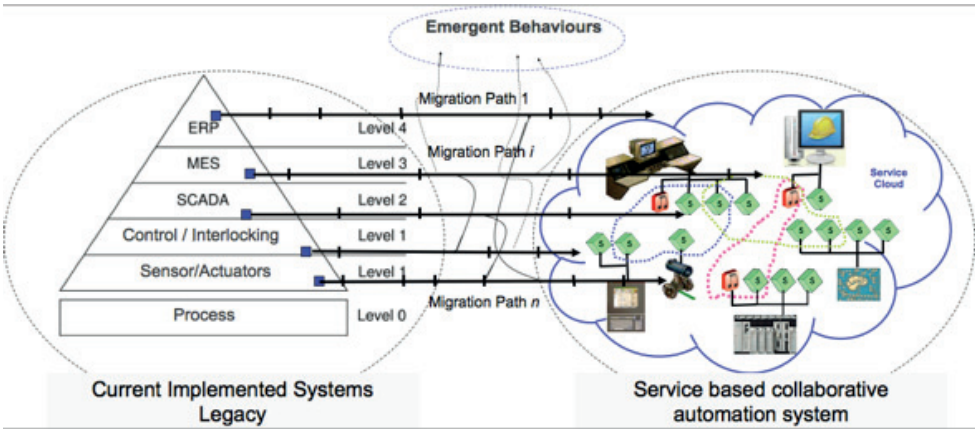


Figure 2: Transferring the ISA-95 automation pyramid architecture to a cloud. This has been investigated by several larger EU projects like e.g. SOCRADES and IMC-AESOP

The trends and perspectives put forward indicates that the current solutions use to build automation systems are not sufficient. Further the cost connected with the engineering and building of larger automation systems involving multiple stakeholders seems to be prohibitively high.

For the last 10 years, discussion on the next generation SCADA, DCS and MES systems has been around. A multitude of research projects on the topic have been executed. Some more prominent such are SOFIA, SOCRADES [6], and IMC-AESOP

[7]. All of them were investigating a move from a hierarchical ISA-95 approach to a more cloud-like approach. A well-known illustration from the IMC-AESOP project is illustrating such move from the pyramid to the cloud in Figure 2. This is currently a rapidly changing landscape but some other efforts touching this field are e.g. FiWare [8]. In addition there are a growing number of cloud offerings on the market. An analysis of the different approaches found in 2015 can be found in [9].

In all these cases the key technology for creating the integration within and in-between different levels of the ISA-95 architecture is Service Oriented Architecture, SOA [10]. SOA was originally developed by IBM to enable data/information exchange between different lines of computer systems.

For the transfer of the ISA-95 architecture to a cloud based approach we do find some important published work regarding, system architecture [11, 12], suitable technologies[13], real time [14, 15] migration from legacy systems [16, 17], engineering for cloud based automation [18, 19].

A parallel discussion is ongoing for the MES and ERP levels. Some important publications on cloud approaches for the MES and ERP level are [11, 20].

Recently the concept of local automation clouds have been introduced via the Arrowhead project the newly released Arrowhead Framework [21].

Most of the recent developments are adopting Service Oriented Architecture, SOA, as the main approach to enable plant automation using cloud technology.

The objective of this paper is to take a step beyond the current cloud automation technology for production. The ambition is to address the actual process of producing a cloud based automation system. It is here argued that there are at least three important capabilities of the automation clouds which are critical to create expected benefits from local automation cloud approach. The capabilities are:

- A way of capturing a plant design: physical devices, components and systems, geographical layout and controlled interaction between physical devices, components and systems
- A way of capturing and distributing configuration information to the entities involved in the plant design
- A way of coordinating information exchange between different entities within a plant

This paper will address the second point: capturing and distribution of configuration information.

The outline is; Section 2 outlines related work, followed by Sections 3 and 4, which presents the proposed approach and gives examples of a few different use cases with experimental results. Section 5 gives a theorethical discussion on the findings, followed by the conclusions in Section 6. The paper finishes by stating ideas for future work in Section 7.

2 Related work

The configuration has long been a significant part of commissioning automated systems in all parts of society. As different domains and applications form, using different approaches and technology, the configuration procedure has developed in somewhat different directions in different areas.

Hodek and Schlick [22] describe the typical operations for integration of field devices in a state-of-the-art industrial automation system. The device profiles that they present could very well be used as a standardized baseline configuration for field devices. However, they also note that there are several other components to a quick and simple commissioning, such as the programming of Programmable Logic Controllers (PLC's) and integration into enterprise software systems. Additionally one of the suggested further investigations is on technology independent Plug& Play features of industrial Ethernet solutions.

Cachapa et al. [23] show how an engineering tool based on a Service-Oriented Architecture (SOA) can help by simplifying the process of designing and applying a production line layout, although in their case the configuration still contains manual configuration for each device.

Dürkop et al. [24] present a solution for a reconfigurable automation system where a Programmable Logic Controller (PLC), using Real-time Ethernet (RTE) connected IO-devices, is equipped with a Web-Service Interface that allows configuration of both the PLC and the RTE network.

Perera et al. [25] present a model to help users configure IoT middleware, primarily for middleware used to collect and process data from IoT-enabled sensors. The issue described here illustrates a situation that can be expected to become more common as automation systems become more dynamic and reconfigurable, where users or operators without detailed knowledge in IT or automation are tasked with the reconfiguration of a system and how tools and methods can aid this process.

In conclusion, there are several approaches and solutions that can help improve the engineering and configuration of both existing and future automation systems in many different areas.

3 Proposed approach

The approach presented in this paper attempts to define certain methods, structures and interaction patterns that will fit in the many areas of society that Arrowhead aims to address. As this will encompass a large number of different technologies and very different external requirements, the general approach is not detailed on a technical level but instead some more detailed scenarios are discussed in the following sections.

Arrowhead Configuration store

The Configuration Store service is one of the Arrowhead automation support core services.

The purpose of the Configuration Store is to provide a uniform way for Arrowhead compliant system to manage distribution of configurations. The extent of the configurability of a system ultimately depends on the system and therefore the design of this service is intended to allow different levels of configurations to be transferred using the same interface, from changes in system parameters to full firmware updates that may change which services a system is able to produce and consume.

Through this design the decision of how configurable a system should be is left to the system provider, and not imposed by the framework, while still allowing a uniform method for configuration management across diverse systems of systems containing systems with different levels of configurability.

In this design the actual configuration file is not necessarily provided by the Configuration Store, this design was selected to accommodate difference scenarios where accessibility, storage or file transfer ability may otherwise be limiting the distribution of configurations. There is however the possibility to have the same system providing both the Configuration Store service and the appointed file storage.

General deployment procedure

As the Arrowhead Framework is intended to allow cost effective deployment of a very wide array of devices, all general methods and procedures need to allow for some flexibility and adaptation to the specific use case. Still, many Arrowhead compatible devices are expected to be deployed in large numbers using low-cost hardware manufactured identically in very large numbers. Under these conditions it becomes even more important to keep the costs for engineering, deployment and commissioning at a minimum.

For the general procedure, the devices are assumed to have identical hardware and identical software preloaded from a factory, workshop or back office, the only differences between devices are their network Media Access Control address (MAC address) and their Serial Number (S/N), or some other individual identifier that has been assigned to them automatically during the device manufacturing process. The preloaded software contains the required security measures and to allow the device to connect to the Arrowhead Framework Core systems and to use a minimal set of services. The basic security may, for example, consist of a certificate installed as part of the manufacturing process which certifies that the device is of the brand and type that it claims to be.

To organize the large number of devices into a productive system of systems each device needs to be assigned a specific task and be configured to perform the task according to a larger plan. This information describing the different tasks and configurations is store in the Configuration Store, as described in section 3. In order to allow some flexibility in network structure, without increasing the engineering or deployment time for each device, a specific procedure has been developed.

Step-by-step general deployment procedure:

1. Device is physically connected to the network and turned on.

2. Device connects to the network using DHCP, or a similar standardized technology applicable for the network in question.

3. Device looks for the Arrowhead core systems [26] at predefined locations. (E.g. on the local network or a cloud service hosted by the device supplier)

4. Core systems authenticate device as factory configured device with basic authorization.

5. Human (operator, electrician, engineer or similar) performing the installation associates the physical/logical location (location as registered in the core systems referring to the point where the device is installed) with the MAC address, S/N or other agreed upon identifier of the device.

   This can be achieved through a mobile interface (e.g. laptop, smart-phone or tablet) where the installer selects the correct location and either reads the identifier from a bar-code, RFID or similar tag on the device or transfers the identifier from the mobile interface to the installed device using NFC or similar communication.

6. Device registers with the core systems providing its identifier for identification.

7. Once the identifier is available at both the installed device and at the core systems a service connection can be orchestrated between the configurable device and the appropriate configuration store.

8. Once the connection is made between the configuration repository and the identified device, the complete configuration containing the assigned task is transferred to the device and installed/executed.

9. Using the received configuration the device is automatically able to start performing its assigned tasks.

10. A message of success/failure is sent as an event to subscribed user interfaces.

Bootstrapping of Resource Constrained Devices

As an example of how the general procedure may need to be specialized to fit certain requirements, a more limited procedure has been designed and tested. The purpose of this is to show how a general procedure may be of use even though the application field is very diverse and it may be difficult to apply the original procedure explicitly in every case.

A zero configuration approach for IoT devices requires the use of Bootstrapping techniques. From the point of view of a wireless sensor and actuator (WSAN) node, the first time that a new node connects to the wireless network it only knows its own IP address

and the IP address of the gateway; it has normally no information about other services in the network.

Bootstrapping is a pseudo-configuration service which provides a basic configuration of the mandatory and essential services that a node requires and needs during the boot process, examples are; configuration services, authentication services or device manager service.

As the only information that a node has after connect to the wireless network is the IP of the gateway, the Bootstrapping service must run on it and should use a predefined port (this is the unique predefined information). The bootstrapping request can include information about the IoT device to create a customized and optimized response for the device; this information can contain a serial number, a predefined ID, MAC address, internal software name, version, or other information able to identify the device or at less the device type.

After this process, the device should store all the information in a non-volatile memory, and use it in case the connection to the bootstrapping service goes down.

The penalty for the utilization of this technology regarding communication is a single request per boot but also it requires the implementation of a parser on the device, which can consume valuable memory on resource-constrained devices.

The following example is a bootstrapping profile encoded with JSON (Code E.1), but it can be encoded with CBOR to reduce the packet size.

Code E.1: Example of Bootstrapping for an IoT node encoded in JSON

```
{
  "auth": {
    "ip": "fdfd:0:0:0:0:0:0:0A",
    "port": 5683,
    "v": 1,
    "res": "/Authentication",
    "resAlt": "/Authorization"
  },
  "conf": {
    "ip": "fdfd:0:0:0:0:0:0:0B",
    "port": 5682,
    "v": 1,
    "res": "/Conf"
  },
  "dev": {
    "ip": "fdfd:0:0:0:0:0:0:0C",
    "port": 5681,
    "v": 1,
    "res": "/rd"
  }
}
```

Intended areas of configuration

To illustrated how the configuration approach may provide different possibilities for different areas of application, this section provides a few examples from some of the domains where Arrowhead is intended to be used.

Control code to PLC's and IoT controllers

Writing control code is one of the critical aspects of automation systems engineering process. The traditional approach to program control systems does not fit well within the paradigm of Cyber-Physical Systems (CPS), where physical systems need to evolve in intimate and aligned correspondence with their virtual representation [27]. To address this, the Arrowhead approach propose a data-driven method for control code deployment, deployable on a number of devices and platforms with embedded data storage and processing capabilities, for the configuration of automated manufacturing systems.

The proposed architecture is composed of three types of elements (see Figure 3): i) data model that describes the system structure and the control behaviour, ii) logic engine that orchestrate system by interpreting description of the system defined within the data model, and iii) resource specific standard library Function Blocks (FBs) acting as an interface between the hardware (i.e. sensors and actuators) and the data model.

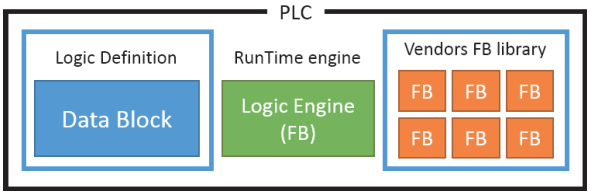


Figure 3: Control software architecture

In this software architecture, the logic engine is configuration independent as for any system configuration it remains unchanged while the data model is configuration dependent as any change in the system, such as sequence, require reconfiguration of the data model. Resource-specific FBs can remain the same for different control or hardware configuration, but may need to be changed in some cases (e.g. vendor specific configuration, actuator type/configuration etc.).

The motivation for this software architecture is to dissociate the key elements required to achieve the overall device configuration and therefore dissociate the engineering processes that support various aspect of the device configuration (e.g. device firmware/logic engine update, system specific configuration change/update, etc.). This approach also allows generating the control code using standard library components (i.e. FBs), which are driven by the control logic defined in the manufacturing process simulation tools to enable seamless transition from virtual to physical system and feedback of runtime data

to virtual system to facilitate data model calibration and analytics. As the control behaviour is defined in the data model, which can be accessed (i.e. read/write) in runtime, service visualization and process parametrisation can be attained using human-machine interface devices.

Configuration of sensors and actuators

Sensor and actuator nodes usually have two types of physical resources: sensors and actuators; and in the case of IoT systems based on Service Oriented Architecture (SOA) the configuration can also set the service's behaviour. Therefore, there are three different types of configurations.

1. Sensor
    - Sample rate
    - Inactive periods
    - Sensitivity
2. Actuator
    - Sensitivity
    - Active/Sleep
3. Service
    - Service composition
    - Filtering
    - Data compression
    - Output format
    - Triggering

Configuration of Building Automation Controllers

Building Automation Controllers (BAC's) use multiple physical resources: sensors, actuators and remote i/o units as well as being freely programmable with applications for monitor & control, communication, HMI and event handling. Therefore, there are many different types of configuration but they can be divided into functional configurations (programming, setup, services) and operational configurations (settings), the later often provided by services in a SOA environment.

4 Use cases and evaluation

The proposed approach is intended to be flexible and to provide different possibilities for scenarios likely to be encountered in different areas of automation. Some use cases have been collected to illustrate the benefits, and possible drawbacks of the approach.

Use case: Building automation

In the field of building automation it is common that many systems in one area are to use identical configurations, or that there are a few typical configurations that are used for a large number of systems. This may be the case for an area with apartment buildings that are all built during the same era and are owned by the same company, here it is beneficial for the owners from a maintenance and management point of view if the systems are as similar as possible.

In this case, and similar scenarios, a centralised Configuration store allows management and updates of all systems based on one configuration that can be rigorously tested and monitored for its first deployment. Once the initial deployment has proven successful it can be applied to all others with low risk of failure.

Additionally, a Configuration store that can keep track of configuration status will allow easier comparison of similar buildings using different configurations, in order to optimise or find flaws in the tested versions.

Use case: Manufacturing industry

Due to the growing need to manufacture highly innovative and customized products, efficient and quick adaptation of manufacturing systems to new product and production volumes is of significant importance. It has been established that one of the major obstacles in realising an efficient and reconfigurable production systems is the existing PLC control code development and deployment approach. The management of PLC devices configuration is currently completely dissociated from other engineering phases (such as process planning and mechanical engineering) and from the digital data set resulting from them.

WMG and FDS is developing a virtual engineering toolset, vueOne, and associated CPS oriented engineering methods that aim at filling this gap by providing data-driven control code generation capabilities (described in section III, D) directly from the vueOne virtual process planning module [27]. In this use case, an engineering scenario focusing on PLC devices configuration is investigated using Web service and Arrowhead Framework to enable direct deployment of control software to PLC devices to provide basis for dynamic and more distinctive configuration scenarios.

Unlike the classical method of PLC device configuration, which requires a direct connection between the PLC and a laptop running the vendor-specific programming tool, the PLC devices (or a server module linked to it) subscribes to the Configuration Store. Any changes in the control configurations are then directly passed from the Configuration Store to the subscribed PLC device when available. To capture the changes made to in the control code on the shop-floor, if any local changes are made in the control software, such as changing some parameters of the data model, the PLC device uploads the latest configuration to the Configuration Store to update the configuration database.

In an ideal configuration mechanism, the PLC device will retrieve updated configuration if/when available automatically. In practice however, the approach requires access to proprietary APIs from PLC vendors to allow download access to the PLCs. The Config-

uration Store consumer software is currently deployed on laptops that control engineers connect to PLCs in order to update or install control code. However, in case of embedded controllers the dynamic configuration can be achieved seamlessly using the Arrowhead framework due to their non-proprietary configuration mechanisms.

Use case: IoT devices

This section is based on experimental results of energy consumption and delays. The benchmark configuration relies on measures of battery current and voltage externally to the device; these measurements are done using a 16-bit ADC at 1840 Hz to capture rapid events such as radio, wakeups, etc. All these measurements are combined to 8 digital inputs that can be used to recognize in detail the power consumption of each software module.

The selected IoT platform to do the test was a Mulle from Eistec AB, which is equipped with an ARM Cortex-M4 at 100 MHz microcontroller and an IEEE 802.15.4 transceiver. It has an onboard 2 MB of flash memory and 256 KB of internal memory on the microcontroller. The Mulle runs the open- source Contiki OS; so all taken measures are affected by running an OS on the same device without any isolation to get real condition data.

As the number of devices increase, so does the need for technologies for large scale management of systems including hardware devices and life cycle management. The issue of life cycle management, in particular configuration, is even more complicated when it comes to managing a very large number of resource constrained, wireless and battery powered devices in harsh environments.

In this use case, the authors have investigated how advanced configuration can be achieved in a very efficient manner in terms of communication overhead and energy usage.

The introduction of the IP technology for Wireless Sensor Networks (WSNs), now called Internet of Things (IoT) is today a hot topic, the power consumption of application's level protocols was a barrier which hindered a massive expansion of IoT; But in 2014, the standardization of CoAP[28] helped to develop IoT systems. The application of CoAP was a step forward and changed the behaviour of the WSAN nodes, from simplistic pull-based to more complex event-based communication. Nowadays, each node can provide services (resources) to other nodes, or to any server/client at Internet. This enables deployment of smart and efficient IoT systems with advanced features such as; service composition, event detection, low-power operation, high dependability, etc.

All these new features requires either run-time zero configuration and/or static configuration; In order to provide run-time configuration which is a much better approach for deploying larger networks, the framework must implement both bootstrapping and configuration services. The validation of these services for bootstrapping and configuration are a direct comparison between the benefits they provide versus their respective performance impact in terms of power consumption, communication overhead and memory usage.

The proposed Bootstrapping and Configuration services enable the following features:

- Low-power. The configuration of the previous WSNs was static; that means that the performance of each device was the same during the life-cycle. But now with Configuration service, the performance can be adaptive. All variables like sample rate, type of processing, triggers, etc. can be modified.
- Dependability and Robustness. The bootstrapping nature is dynamic, on each request, the bootstrapping service creates a customized response; Then if a service goes down, another one can replace it, just changing the bootstrapping parameters.
- Zero Configuration. With bootstrapping there are no fixed end-points on the device, one node could be deployed at any WSNs, and it will be able to establish a correct configuration.

The consumption of these two services (Bootstrapping and Configuration) is not excessive compared to others as Authentication, Authorization and Device Manager. The values at Figure 4 can change depending on the complexity of the device configuration.

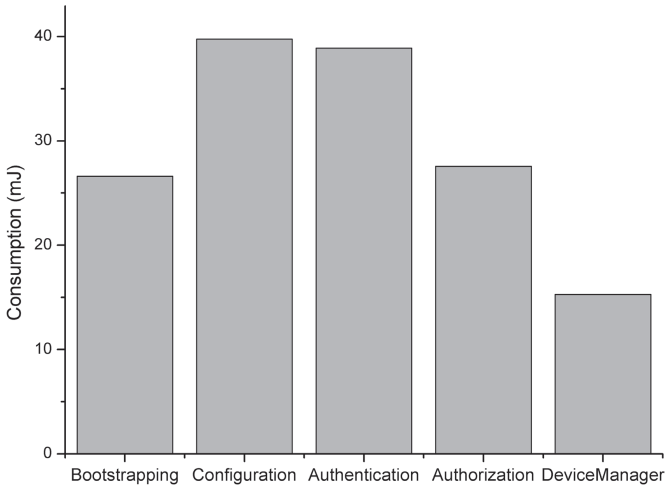


Figure 4: Comparison of energy consumption between Bootstrapping, Configuration and other common IoT-WSNs services

Regarding memory, the biggest cost is to parse the incoming configuration profile. For this experiment, the profile was encoded in JSON format, and the jsmn[29] C-API was used as the parser. The configuration memory usage represents the memory of configuring a single service. The bootstrapping memory usage also includes the JSON parser's ROM memory usage (see Table 1).

	Bootstrapping	Configuration
ROM (bytes)	1126+4382(parser)	840
RAM (bytes)	412	776

Table 1: Memory footprint of Bootstrapping and Configuration

Use case: Process industries

The process industry typically has comparatively static production lines where reconfigurability of the production, as commonly described for manufacturing industries, is not usually sought after. However, another scenario that may require significant configuration is that of replacement of devices, machines or groups of systems.

In the current environment of process industry automation systems, replacement of devices or larger systems generally takes place because of one of two reasons.

The first reason is that the existing one is broken, faulty or worn out and should be replaced with a new, but identical, item. This case is generally considered to be part of standard maintenance procedures, either reactive or proactive.

The second reason is that the existing item is too old and should be replaced with a newer part, this may be due to e.g. discontinued support from suppliers and difficulty to procure replacements, difficulty to attract and retain personnel with the required expertise in older systems or due to lack of technical features or functionality. This case is generally considered part of the long term investments in production systems and as such, replacements are generally planned carefully and in great detail.

While the conditions before the two cases are very different, especially with regards to the possibility to make preparations, as one is planned and the other is not, there are also some commonalities in that in both cases the functionality is expected to be exactly the same as before and that the time for the replacement and commissioning is usually very limited.

Traditionally these kind of replacement procedures have relied heavily on existing documentation and backup copies of control code and configurations, however these are not always kept up to date and may not be fully compatible with updated replacement devices. This leads to additional engineering work and may in unfortunate cases lead to unforeseen complications during the critical commissioning work.

As the Configuration store can be used to store the most current configuration for each device and the deployment procedure allows for a smooth introduction of new devices, an easier, quicker and cheaper replacement process is expected.

However, the Configuration store does not solve the issue with incompatible configurations or software version, but having a backup that is certainly of the most recent configuration will make engineering work easier and the re-commissioning less uncertain. If the device manufacturers can be persuaded to use open and standardized configuration files this would allow more possibilities for conversion and testing of updates ahead of the physical replacement.

5 Discussion

The use cases presented are diverse and have significantly different requirements and potential benefits. The case presented from Building automation highlights the benefits from a management and maintenance point of view. The use case from a Manufacturing industry illustrates some of the strong potential once an approach like this is fully integrated in a complete engineering tool chain, something that can be expected to happen eventually in all areas with significant engineering requirements. However, it also illustrates some of the limitations that may occur when existing hardware is not yet capable to make use of new possibilities.

The IoT case illustrates the drawbacks that can be expected from using the approach for resource constrained devices. The cost incurred in terms of energy and memory consumption is not negligible, but for scenarios where a large number of devices are to be deployed and configured it is likely to be acceptable. At least when compared to the additional engineering time required to configure all devices individually prior to deployment. The use case from a Process industry shows that a structured, centralised management of device configurations can provide benefits to diverse elements of society. In addition, keeping the records of device configurations automatically updated is likely to remove some of the additional cost incurred when a device needs to be replaced or upgraded.

6 Conclusion

In this paper, we have presented the Arrowhead Framework's approach for advanced configuration of systems and devices. The proposed approach is designed to be highly versatile while still being efficient for usage by resource-constrained devices. The approach has been implemented and tested on a resource-constrained wireless sensor and actuator platform.

Test results indicates that the overhead from performing bootstrapping and automatic configuration of a newly deployed device is negligible in terms of energy consumption and memory usage in relation to the energy spent by the device for sensing purposes during its lifetime. This shows that advanced run-time configuration is feasible even on small, battery-powered devices.

7 Future work

Among the possible paths of future advances can be found several interesting options. Along the path of further implementation lies the tasks of prototype implementations for configuration of larger, less limited devices.

Further development could include implementations more integrated with engineering tools. This could be done either by implementing methods for managing the ConfigurationStore from existing, discipline related engineering tools, or by designing an engineer-

ing tool centred around the ConfigurationStore and associated processes. The former implementation would allow for a smooth engineering and deployment process within the specific domain, while the latter has an advantage of spanning all affected domains easing e.g. management and maintenance of multi-domain facilities.

8 Acknowledgment

The authors would like to extend their gratitude towards EU ARTEMIS JU for funding within project ARTEMIS/0001/2012, JU grant nr. 332987 (ARROWHEAD). We would also like to thank the partners of the Arrowhead project for fruitful collaboration and interesting discussions.

References

[1] (2016). [Online]. Available: https://www.isa.org

[2] B. Scholten, *The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing*. ISA, 2007.

[3] (2016). [Online]. Available: http://isa99.isa.org/ISA99%20Wiki/Home.aspx

[4] K. Staggs and et.al., "ISA 62443 – 4 – 2 security for industrial automation and control systems technical security requirements for iacs components," ISA, Draft Standard Draft 2 edit 4, July 2015.

[5] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[6] (2016). [Online]. Available: http://www.socrades.net

[7] A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, "Industrial cloud-based cyber-physical systems," *The IMC-AESOP Approach*, 2014.

[8] Wikipedia, "FiWare — wikipedia, the free encyclopedia," 2016, [Online; accessed 21-April-2016]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=FIWARE&oldid=711836994

[9] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the internet of things," in *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 2015, pp. 1–8.

[10] T. Erl, *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.

[11] S. Karnouskos and A. W. Colombo, "Architecting the next generation of service-based scada/dcs system of systems," in *Proceedings IECON 2011*, Melbourne, Nov. 2011, p. 6.

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

[13] F. Jammes, B. Bony, P. Nappey, A. W. Colombo, J. Delsing, J. Eliasson, R. Kyusakov, S. Karnouskos, P. Stluka, and M. Till, "Technologies for soa-based distributed large scale process monitoring and control systems," in *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2012, pp. 5799–5804.

[14] G. Candido, A. W. Colombo, J. Barata, and F. Jammes, "Service-oriented infrastructure to support the deployment of evolvable production systems," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 759–767, Nov 2011.

[15] R. Kyusakov, P. P. Pereira, J. Eliasson, and J. Delsing, "Exip: a framework for embedded web development," *ACM Transactions on the Web (TWEB)*, vol. 8, no. 4, p. 23, 2014.

[16] J. Delsing, J. Eliasson, R. Kyusakov, A. W. Colombo, F. Jammes, J. Nessaether, S. Karnouskos, and C. Diedrich, "A migration approach towards a soa-based next generation process control and monitoring," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, Nov 2011, pp. 4472–4477.

[17] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo, and T. Bangemann, "Migration of industrial process control systems into service oriented architecture," in *IECON 2012*. IEEE, 2012.

[18] A. Jain, D. Vera, and R. Harrison, "Virtual commissioning of modular automation systems," in *Intelligent Manufacturing Systems*, vol. 10, no. 1. IFAC, 2010, pp. 72–77.

[19] N. Kaur, C. S. McLeod, A. Jain, R. Harrison, B. Ahmad, A. W. Colombo, and J. Delsing, "Design and simulation of a soa-based system of systems for automation in the residential sector," in *IEEE ICIT 2013*. IEEE, 2013.

[20] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, and T. Bangemann, "Towards an architecture for service-oriented process monitoring and control," in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2010, pp. 1385–1391.

[21] J. D. ed. (2016) Arrowhead framework wiki. [Online]. Available: http://forge.soa4d.org/arrowhead-f/wiki

[22] S. Hodek and J. Schlick, "Ad hoc field device integration using device profiles, concepts for automated configuration and web service technologies: Plug amp;play field device integration concepts for industrial production processes," in *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*, March 2012, pp. 1–6.

[23] D. Cachapa, R. Harrison, and A. Colombo, "Configuration of SoA-based devices in virtual production cells," *International Journal of Production Research*, vol. Volume 49, no. Number 24, pp. 7397–7423, 2011. [Online]. Available: http://wrap.warwick.ac.uk/54497/

[24] L. Dürkop, H. Trsek, J. Otto, and J. Jasperneite, "A field level architecture for reconfigurable real-time automation systems," in *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*, May 2014, pp. 1–10.

[25] C. Perera, A. Zaslavsky, M. Compton, P. Christen, and D. Georgakopoulos, "Semantic-driven configuration of internet of things middleware," in *Semantics, Knowledge and Grids (SKG), 2013 Ninth International Conference on*, Oct 2013, pp. 66–73.

[26] F. Blomstedt, L. L. Ferreira, M. Klisics, C. Chrysoulas, I. M. de Soria, B. Morin, A. Zabasta, J. Eliasson, M. Johansson, and P. Varga, "The arrowhead approach for soa application development and documentation," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, Oct 2014, pp. 2631–2637.

[27] R. Harrison, D. Vera, and B. Ahmad, "Engineering Methods and Tools for Cyber-Physical Automation Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 973–985, May 2016.

[28] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

[29] "Jsmn C JSON parser," https://github.com/zserge/jsmn, updated: 2016-01-15.

Paper F

Using Internet of Things for Industrial Applications: A Feasibility Check

Authors:
Jens Eliasson, Pablo Puñal Pereira and Jerker Delsing

Reformatted version of paper originally published in:
Submitted to IEEE Journal of Sensors.

Using Internet of Things for Industrial Applications: A Feasibility Check

Jens Eliasson, Pablo Puñal Pereira and Jerker Delsing

Abstract

Today's high demands on raw materials like iron ore, copper, and other minerals puts a high pressure on the mining industry to deliver a steady flow of ore, metals and minerals. However, since mines are inherently dangerous, there is a strong need for making mining operations more safe, efficient, and environmentally friendly. Rock bolts are widely utilized by the mining industry as an approach for increasing mine stability. One problem is that rock bolts can become damaged by for example seismic activities such as blasting or mine quakes.

In this paper, we present an architecture based on OMA LWM2M, IPSO Smart Objects and the Arrowhead Framework for IoT in industrial applications. The paper also presents results from two feasibility aspects: performance and lifetime. By utilizing (Industrial) Internet of Things technologies, it is possible to drastically improve monitoring of mining activities and thereby providing workers with a safer working environment.

The architecture supports low-power operation, zero-configuration networking, interoperability, security and authentication mechanisms. This shows that standardized protocols such as 6LoWPAN, IPsec, LWM2M, and IPSO Smart Objects can be used in the Industrial Internet of Things.

1 Introduction

The mining industry has been an important contribution to recent years development of new types of products and businesses. A good supply of cost-efficient raw materials has resulted in strong economic growth within the European Union and around the world. As mines are starting to operate at increased depths, mining operation is getting more and more hazardous for the workers. Last year's drop in ore prices also put higher requirements on higher operation efficiency. Safety and efficiency are two very important factors for the mining operation.

These facts show the need for low-cost, massive online monitoring of mine operations. Today, best practice in mines make use of very accurate and expensive sends for online monitoring. One problem though is that today's solutions often requires a fixed infrastructure for power supply and data communication. This is one main reason for mining companies to not being able to install sensors at the same rate is tunnels are being excavated.

In 2011, the concept of Industry 4.0 was first introduced. This concept builds upon the last generation of industrial monitoring and control systems, but enables an even finer level of interaction between shop-floor devices and high-level enterprise systems. In Industry 4.0, the state of the art technologies like Internet of Things (IoT) and Cyber-physical Systems (CPS) are utilized. This opens up for new strategies in terms of global plant optimization, minimized energy consumption, safety, and security, etc. In for example [1], IPSO Smart Objects was used to interface a lighting application. Of course, when more complex communications stacks are being deployed even on resource-constrained sensors and actuators, new challenges and problems that must be handled are introduced. One of the most critical challenges is security [2]. When more and more devices are networked, it opens up vulnerabilities for remote network-based attacks. Earlier systems could only be configured by someone actually on the factory floor, but when Internet protocols are used it can allow users to change settings and configuration from anywhere on the planet.

The use of IoT in industrial applications has been used for some time, see e.g. [3] and [4]. In [5, 6], the use of Internet-connected rock bolts was first proposed by the authors. This approach enables mining companies to monitor critical infrastructure in near real-time and to be able to detect possibly dangerous changes in tunnels and other cavities. This paper extends the work presented in [5] with an updated and holistic architecture, plus a comprehensive performance evaluation. The architecture presented in this paper makes use of the Arrowhead Framework for distributed collaborative systems [7].

In this paper, we present the state of the art in terms of rock bolt monitoring using an architecture for Industrial Internet of Things technologies. The paper also presents resulted regarded sensing performance and anomaly detection and low-power operation characteristics. The use of standardized protocols always come with a higher overhead that using customized and fine-tuned proprietary protocols. However, by basing the architecture exclusively on open standards and protocols a high level of interoperability is made possible. When integrating state of the art IIoT systems with existing industrial monitoring and control systems, a high level of interoperability is, of course, important.

This paper is structured as follows; Section 2 provides related work and a state of the art review of Internet of Things and wireless sensor and actuator networks. Sections 3 and 4 presents the proposed networked and system architectures. Following that comes Sections 5 and 6, which presents laboratory and field test experiments, results and a theoretical discussion on feasibility. Finally, the paper's conclusions and suggestions for future work are presented in Sections 8 and 7, respectively.

2 Background and Related work

This section provides an overview of how industrial process monitoring and control systems have evolved, and how the mining industry today is measuring rock stability to provide a safe working environment.

Process monitoring and control

Modern industrial production and manufacturing systems have evolved in basically four generations. The first generation that enabled the industrial revolution dates back more than 200 years or so. The use of e.g. steam-powered machines allowed mass production of goods such as clothes, equipment, and many other products starting around 1850.

In the second generation, the use of efficient pneumatic systems became a widely adopted solution for mass-production. The combined use of pneumatic valves and sensors enabled automatic production systems to be used in industrial applications.

The third generation systems used electrical motors. Sensors and actuators were now also connected to new types of monitoring and control systems like Distributed Control Systems (DCS) and Supervisory Control And Data Acquisition (SCADA). The hierarchical approach of device-level, DCS, and SCADA (known as ISA-95), soon became the de-facto architectural style for how industrial productions systems were designed and deployed. The use of IoT technologies is also starting to be widely accepted in industrial applications, see e.g. [8].

Mining monitoring

Mine activity monitoring is today usually performed using a type of sensors called *geophones*. Geophones are extremely sensitive devices that can be used to detect anomalies in for example mine tunnels [9]. Today, geophones are typically network connected and can be used for remote monitoring of mines. Data from geophones are typically processed off-site.

Rock bolts have been used for reinforcing tunnels and over cavities in mining for over 100 years. Rock bolts are widely used in the mining industry for increasing tunnel stability. Each year, between 10-100 million bolts are installed around the world. This shows that rock bolts are a vital component for underground mines. However, rock bolts can become damaged due to seismic event lick earth and mine quakes as well as by regular mining activities such as blasting. When this happens, rock bolts may loose some or all of their load-bearing capacity. This can lead to disasters in terms of injuries and even deaths as well as economic losses. Damages are not always visible from the tunnel since a large majority of a rock bolt is inside tunnel walls and ceilings. Today, there are no real technologies for low-cost, large-scale and real-time monitoring of installed rock bolts.

Internet of Things

Estrin et al. presented opportunities for wireless sensing in [10] already in 2001. Back then, the term Wireless Sensor Networks (WSN) was used to describe an ecosystem comprised of a potentially very large number of low-power, wireless sensor platform collaborating in sensing their environment. Today, we are seeing this vision being realized. Due to technological breakthroughs in battery technologies, micro-controllers, and wireless communication capabilities, the Internet of Things is now foreseen to include billions of devices the net decade or so. In [11], Breivold and K. Sandstrom compares

"traditional" Internet of Things with Industrial requirements. When comparing IIoT to more consumer-based application scenarios such as home automation, security and sports monitoring, there are many requirements that exist in the industrial domain that is non-existent or at least not a primary concern in the consumer world.

- *Scalability.* In industrial process monitoring and control, there can be up to tens of thousands of devices in e.g. a factory floor process. This adds requirements on network scalability, robustness, and plug and play operation. In industrial applications, it is most likely not the price of a device that with dominate the overall ownership costs but rather a deployment and configuration costs. In a domestic home, there is usually no costs for deployment since the house owner will install the device himself.

- *Security.* An intrusion in a home automatic scenario can be annoying for the owner. But an intrusion in a factory monitoring system there can be severe consequences in terms of economical loss and downtime. A large deployment is also more susceptible to intrusion attacks from professorial hackers compared to a household. The Stuxnet attack [12] showed that cyber warfare can cause severe consequences.

- *Configuration.* Cirani et al. showed in [13] the need for methods and tools for large-scale maintenance and especially configuration of IoT devices. As industrial installations will be of a considerate size in terms of sensor- and actuator platforms as well as gateways, there are clear needs of mechanisms for minimizing (or even eliminating) human interaction when deploying devices and maintenance and configuration during their life cycle.

- *Interoperability.* In an industrial application, there is often a multitude of different systems in operation simultaneously, often from different vendors. In order to enable seamless integration with as many systems as possible, it is important that new systems use standardized protocols and technologies. The industry today generally avoids vendor lock-in effects and proprietary solutions as they can become very costly in the long run.

- *Lifetime.* In the consumer market, devices tend to be replaced after a few years, especially if the price of new devices is low. On the contrary, in industrial applications, there is a clear need of long system lifetimes due to high installation costs. Therefore, it is important that the devices, systems and technologies tat are installed will be able to operate for many years.

Today, there are a number of protocols and technologies for Internet of Things and Cyber-physical Systems such as SigFox, DASH7, LoRA, Wireless HART, and others. There is a very large market being foreseen which drives technology development for newer and improved solutions for wireless communication, sensing, processing etc.

For more consumer-based markets, ZigBee Alliance, Thread group, and Z-wave hold strong positions. IPv6 and 6LoWPAN, which enables true IP-based communication is the

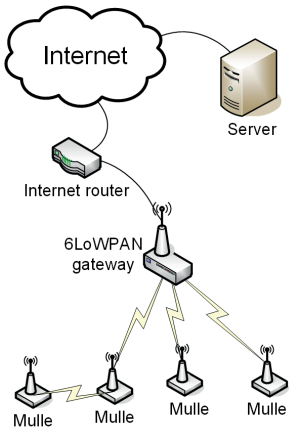


Figure 1: Communication architecture

base for Thread is supported by ZigBee has not yet been a widely adopted in consumer products. Yanzi Networks is one provider of 6LoWPAN-based devices for the Internet of Things.

3 Network infrastructure

This section presents a description of the communication architecture, including the sensor and actuator platform in use, wireless communication, used protocols and security considerations.

Wireless sensor and actuator network

The communication architecture is outlined in Figure 1. This represents a traditional setup with 6LoWPAN-enabled wireless devices connected to a 6LoWPAN border router (gateway). The gateway, in turn, is connected to a local network. The architecture also features data storage services that can either be hosted inside the local network or on the public Internet.

Figure 3 shows the communication stack that is used between the rock bolts and the gateway. This stack uses an 868 MHz IEEE 802.15.4 wireless network with 6LoWPAN, RPL, and IPv6. Currently, ContikiMAC [14] is used to provide low power consumption when the radio is on. IPsec provides encrypted communication and authentication of valid peers on the network. All sensor nodes run the Contiki operating system on the Mulle wireless sensor and actuator platform, shown in Figure 2.

Figure 2: Eistec's Mulle sensor- and actuator platform

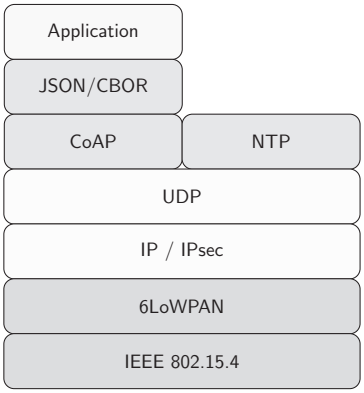


Figure 3: Communication stack

Gateways

The gateways provide several functions, such as conversion between 6LoWPAN and Ethernet, data storage, local wireless time synchronization, configuration and basic visualization. The use of monitoring and control features on the gateway also allows sensor network to be deployed even without any network connectivity. This approach also enables the entire system to operate even if there is a network failure. When network connectivity is restored, the gateway starts to replicate data against cloud data storage services.

The gateways also use encrypted VPN tunnels when establishing the connection with their data storage servers. This, together with IPsec, enables encrypted communication between rock bolts and gateway, and gateway to the cloud. The gateway currently supports solutions like OpenVPN as well as Cisco AnyConnect.

Security and access control

The network allows communicating each node to each other and external devices, under this situation and especially on wireless networks, protect the communications is one of the first tasks. The communication needs to guarantee the veracity of the data to prevent intruders to inject data (man-in-the-middle attack), and also protect each transfer to be read by third parties (sniffing attack). The use of IPsec can protect the communications at this level, encrypt the communication and adds authentication, but only at a peer-to-peer level. In fact, with the use of IPsec, the Application layer (OSI model) can not distinguish between protected and unprotected communications. A similar problem occurs with DTLS; it does not provide the fine-grained access control which is needed to customize the access to each single service on the nodes. For this reason, a new Access Control method is required.

The implemented Access Control [15] is based on the use of Tickets. A Ticket is a group of bytes, 8 bytes in this case, which have to be included in all the requests of the system. The Ticket is unique and is generated by the Authentication and Authorization Server, which is the responsible to verify them. When a node, which is acting as a server, receives a request from a new client, the node must contact to the Authentication and Authorization Server and check the Ticket validity and timeout. This Access Control mechanism, protects the alert services, preventing to none authorized nodes to use it. The capability to integrate this mechanism to other standard authentication systems, like RADIUS and DIAMETER, helps to merge an Industrial solution to this network.

4 System Architecture

The proposed architecture consists of several components. The core of the architecture is an Internet of Things-based rock bolt. Many smart rock bolts can form a low-power wireless mesh network. In order to connect the low-power, often sub-GHz, network with existing infrastructure gateways is used. Finally, in order to be able to perform data processing of large amounts if sensor data, a back-end system used. However, the work presented in this paper is focused on the lower levels of the proposed architecture, from sensors, networks and gateways to cloud-based systems for data storage, basic processing, and visualization. Mine monitoring systems are for example not considered to be a part of the proposed WSAN architecture itself.

The foundation of the proposed architecture is the smart rock bolt, shown in Fig. 4. The smart networked rock is comprised of a multitude of on-board sensors and actuators, signal processing capabilities and low-power wireless communication.

The current generation smart rock bolts used to collect experimental data uses three main sources of environmental information; namely strain, bolt breakage and seismicity (vibrations). These three sensors provide important information about the condition of each monitored rock bolt. In order to be able to warn nearby workers of imminent danger, the smart rock bolt is equipped with high-power LEDs that can be used to blink in different colors, patterns, and combinations.

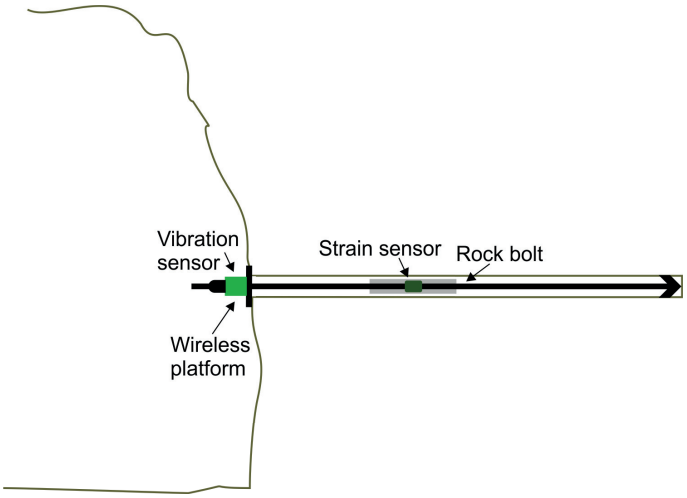


Figure 4: Smart rock bolt in tunnel wall

OMA Lightweight M2M

OMA Lightweight M2M (LWM2M) [16] is a standard for device management from the Open Mobile Alliance. LWM2M provides features for example security, bootstrapping, over the air firmware upgrades, device and resource registry, to mention a few.

In the Smart Rock bolt architecture, OMA LWM2M is used to enable plug and play capabilities, device management, etc. The Resource Directory is a key component to allow the gateway to automatically discover new devices and to read manufacturer ID, serial numbers, battery voltages, etc. To test the LWM2M implementation, both Eclipse Leshan (in Java) and Wakaama (in C) was used.

IPSO Smart Objects

In the Smart rock bolt architecture, IPSO Smart Objects is utilized for data models and CoAP communication for sensor and actuator interaction. All sensors and actuators, except the vibration sensor, are available as IPSO objects. This enables the lightweight SCADA system on the gateway to easily read and control all parameters of the rock bolt monitoring system.

The use of standardized interfaces and data models also enables a streamlined integration with other systems since there is no vendor or technology lock-in. For Internet of Things to be an interesting approach to industrial applications, interoperability is one of the key features to support. To test the IPSO implementation, Eclipse Leshan (in Java) was used.

Arrowhead Framework

The Arrowhead Framework is a comprehensive framework for IoT in the area of industrial process monitoring and control. This framework provides a Service-oriented Architecture (SOA) for features like device and service discovery, orchestration, configuration, security and data storage. The framework has been developed within the Artemis Arrowhead project.

The Arrowhead Framework is used in combination with OMA LWM2M and IPSO Smart Objects in order to provide a higher level feature such as access control, data storage, service orchestration and data processing.

5 Results

This section provides experimental data of the embedded sensors and the ability to sense phenomena in the surrounding environment, power consumption data and latencies in the communication. Since many industrial applications are time-critical, it is important for a monitoring and control application to support low-latency communication. The experiments have been divided ito two categories: mining monitoring and process monitoring.

Mining monitoring - rock bolts

Below are the results from vibration sensing in field tests and strain gauge output in a laboratory environment on a 22mm rock bolt. Some experiments have also been reported in previous work by the authors [6].

Detecting falling rocks

Figure 5 shows the vibration signature that occurs when rocks fall from e.g. a tunnel ceiling and hit the floor. These spikes in vibration happened when a rock of a weight of 2 kg. was dropped from approx. 2 meters height at three meters distance from a rock bolt.

Strain

Figure 6 shows the output from the rock bolt's strain gauge when being subject to severe load in a test rig. 4.3 minutes into the experiment, it is clear that the measured force (in red) shows a a deviation when the steel in the rock bolt reaches the plastic region. The derivative of the force drops as the steel's has reached the maximum load for it to function properly. At 6.0 minutes, the strain gauge is broken due to the severe elongation. At 25 minutes in, the rock bolt is torn in half which is clearly detected by the reference instrument (the line in blue), i.e. the force are now at 0 due to no load.

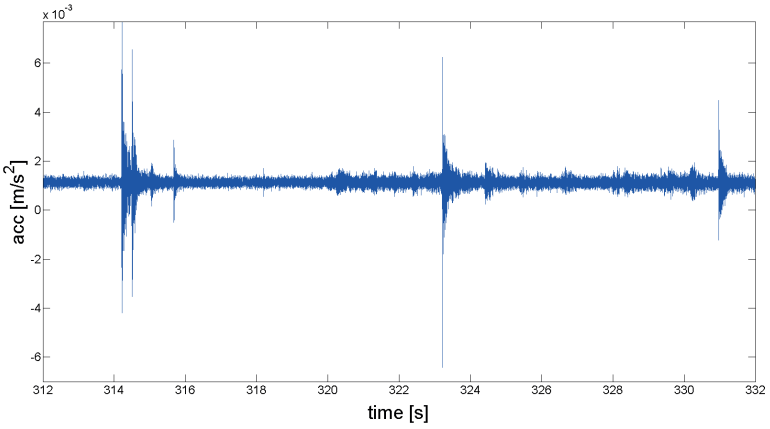


Figure 5: Falling rocks detection

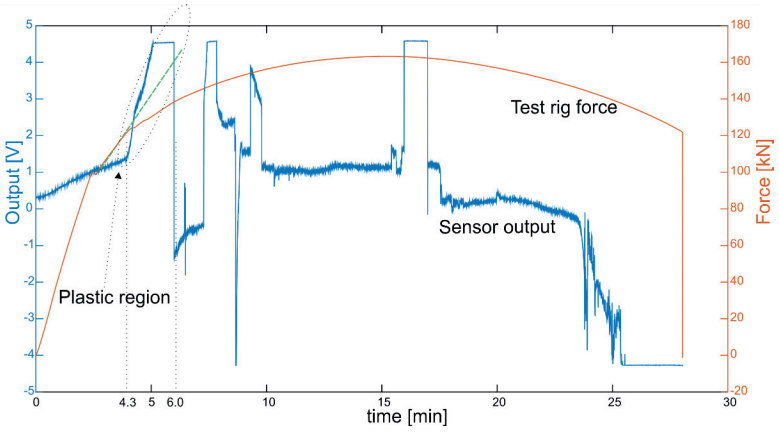


Figure 6: Strain (load) sensing

Process monitoring

This experiment was performed with the presented sensing platform in order to monitor the state of a compressor.

Compressor state monitoring

A vibration sensor was attached to the compressor and its vibrations was measured in four different state: off, starting, on, and stopping. Figure 7 shows the envelope of one vibration experiment's vibration signal when the compressor cycled through its states. from this output, a state detector was implemented in order to detect the various state

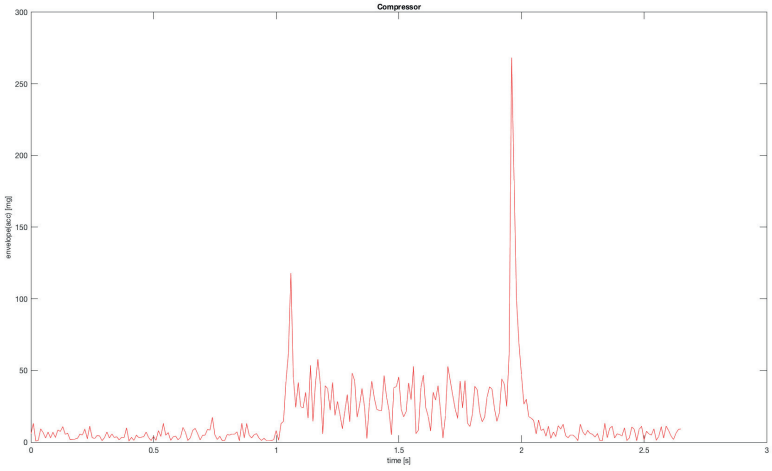


Figure 7: Compressor state monitoring

transitions.

Energy consumption

The energy consumption of a rock bolt depends on the configuration of the sampling rate and the triggers to notify a vibration. Therefore, a consumption profile of each involved part is crucial. The benchmark configuration to get that data was based a 16-bit ADC at 1840 Hz to capture rapid events combined with 8 digital inputs that can be used to recognize each software module. On vibration sensing, there are three modules: acquisition, processing and notification. The last one is called only if a vibration is detected. As an example, Figure 8 represents the data from the ADC during a vibration detection.

To simplify the analysis, the use of 4 digital signals help to recognize each software module and the Figure 8 can be represented as Figure 9, which is easier to understand. At this point, a more detail explanation about the internal functionality of each module helps to understand better why the consumption has that profile.

- Acquisition. The rock bolt needs to activate a 3-axis accelerometer and start to take 128 measurements at 400 Hz, which only the measurement takes 320 ms. The process does not require a high level of energy because there is no high processing cost.

- Processing. The simplest detection can be done with the use of RMS, maximum and minimum values per axis; this process requires more computation than Acquisition, but with a 100 MHz micro-controller it can be done in just a few milliseconds.

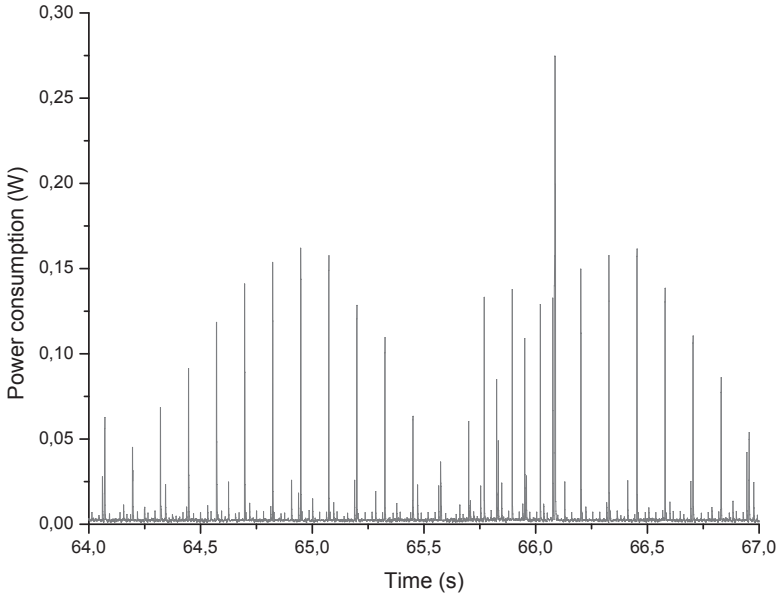


Figure 8: RAW power consumption during an vibration measurement, detection and notification

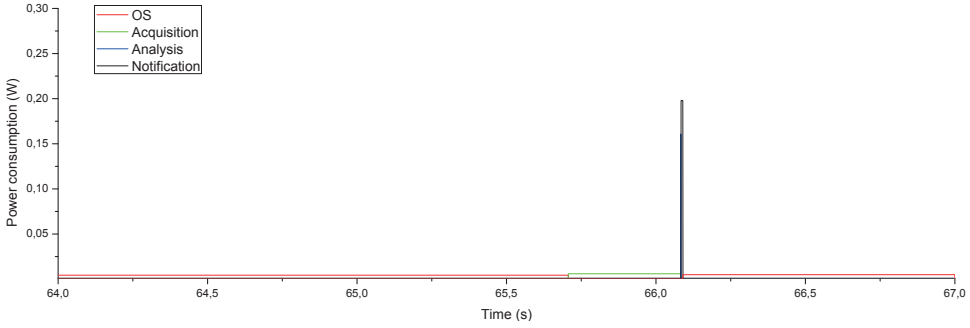


Figure 9: Average power consumption and representation of each action during the vibration detection

- Notification. This module is the responsible to, in case that a vibration is detected, send a notification to the corresponding devices. The energy consumption of Notification includes all the software to do the communication as well as the radio transceiver.

The results of the benchmark are shown in Table 1. Note that the processing is performed in pure software, utilizing the DSP features in the Freescale Kinetis MCU

would likely reduce the consumed energy even more.

Action	Power (mW)	Time (ms)	Energy (mJ)
System	0.25	-	-
Acquisition	1.21	376.1	0.46
Processing	160.8	1.16	0.19
Notification	197.6	5.83	1.15

Table 1: Consumption of phases during vibration sensing.

As shown in Table 1, the radio communication has the highest cost in terms of energy. This is well in line with similar papers on energy consumption for sensor nodes. With the assumption that each sensing period will not occur a radio notification being generated, we can conclude that by duty cycling the vibration sensing by for example 10 activations (and one notification) per minute, the average system plus sensing energy cost would be around 22 mJ per minute which equals to 0.38 mW average power consumption.

A system that does not perform any sensing or communication, the battery life is more than 3 years on a 2000 mAh battery, which gives us the upper limit of system lifetime.

IPSO Challenge 2015

The architecture presented in this paper was submitted to IPSO Alliance's competition IPSO Challenge in 2015. The submission contained a holistic platform composed on rock bolts with embedded sensors, signal processing, actuators, low-power wireless communication and a web-based monitoring system. In competition with over 50 other entries, the Smart rock bolt received first place [17].

6 Discussion

The results in terms of sensing, processing and low-power operation that has been presented in this paper show that very a very traditional and simplistic object such as a reinforcing bar (rebar) can be equipped with sensors and (Industrial) IoT communication capabilities in order to enable real-time monitoring of structural health.

From the presented graphs, as well as other experiments it is possible to conclude that the sensing performance of modern sensors like MEMS accelerometers and strain gauges is sufficient for low-power, low-cost rock bolt monitoring.

Input on lifetime requirements from experts in the mining industry indicates that there are basically three different types of lifetime requirements;

- Short-term. Short term lifetime is up to six months near tunnel faces where blasting and mining happens. This is the most dangerous place in a mine due to rock

instability, moving machinery, etc. Devices that can life for up to six months can be used here. After that, the face has been moved further away into the rock masses.

- Medium-term. This span is up to 2 years. Normally, if no problem has been detected within two years after a rock bolt has been installed, there is very little likelihood that something will happen. Most collapses occur within this time span.

- Long-term. Tunnels and other cavities are used for storage, transportation must be monitored for a very long time span. Here experts indicate that up to 20 years is desired. Today's battery technologies can achieve up to 10 years or more using state of the art chemical compositions. However, when adding the power consumption of wireless devices and self-discharge of modern batteries, it is difficult to reach 20 years of lifetime. Two solutions though are either to keep rock bolts that must operate for 20+ years powered by cables, or by manually replacing drained batteries.

7 Future work

In the work presented in this article, we have focused on developing technologies, tools and methods for low-power operation of IoT devices in a smaller network. More research is needed to enable very large scale networks with potentially thousands of devices per gateway. For example, consumers are much more willing to adopt and try new technologies, where the industry tends to rely on more field proven products and services. Another issue to address is the use of wireless communication under severe interference [18].

The radio environment in mines is also more like a thin chain than a mesh since tunnel walls cannot be penetrated by radio signals. We must, therefore, investigate new technologies for long thin mesh networks with a large number of hops. One such alternative worth investigation is 6TiSCH [19].

Another interesting approach for distributed anomaly detection, e.g. proposed by Khan in [20], could be suitable for minimizing power consumption and network traffic.

8 Conclusion

As mining is conducted on increasing depths, more sensors are needed to monitor cavities, pillars and tunnels in the search for potentially hazardous phenomena. In this paper, we have presented a system for distributed real-time monitoring of seismic activities, and strain in rock masses. The system's architecture is based on (Industrial) Internet of Things by the use of low-cost and low-power sensors, wireless communication, battery powered electronics and distributed sensing and processing. At the core of the architecture, we find the Smart rock bolt. A smart rock bolt is composed of a standard rock bolt, equipped with sensors, actuators, processing, storage and low-power wireless

communication. The smart rock bolt can monitor its own strain (load), as well as seismicity (vibrations). Any deviations from normal operation are detected using on-board signal processing. Alarms and sensor data are sent using technologies such as CoAP, the Arrowhead Framework, and IPSO Smart Objects.

Since the rock bolts are designed to be battery powered, power consumption is of utmost importance. The current generation electronics enables operation lifetimes in the range of 1-2 years depending on environmental aspects, e.g. level of seismicity, desired sample rates, network topology, etc.

When deploying wireless sensor and actuator networks in industrial applications, security is a very important feature since intrusion could lead to great economical losses. The rock bolt architecture, therefore, supports strong encryption using IPsec between wireless devices and gateways, and VPN tunnels between gateways and cloud systems.

As seen in the Results section, by duty-cycling radio, sensors, and micro-controller, it is possible to reach months of operational lifetime. Using duty-cycle levels of 10-50% would enable a rock bolt to operate for up 2 to 4 years using a 2000 mAh Lithium battery. This is well in line with the requirements of the industry regarding the required lifetime.

The target deployment level now is that between 1 and 5% of all rock bolts could be of the smart type. Having every bolt smart would be relatively expense in terms of obtained performance. Therefore, 2-10% is sufficient in order to be able to detect changes in strain and vibrations in cavities and tunnels. The use of self-monitoring rock bolts with wireless connectivity can be one enabling technology for the deep mine of the future.

With the reported low-power operation, zero-configuration networking, interoperability, security and authentication mechanisms, the proposed architecture meets identified requirements from an industrial view. This shows that standardized protocols such as 6LoWPAN, IPsec, LWM2M and IPSO Smart Objects can be used to build Industrial Internet of Things applications in different environments such as mining and processing monitoring.

9 Acknowledgment

The authors would like to extend their gratitude toward the European Commission for their financial support of this work in the FP7 I2Mine project, and the Artemis Arrowhead project. We would also like to acknowledge Vinnova for their financial support in the RBM-IoT and ACO projects.

Furthermore, we acknowledge the important support from Eistec AB, Malmfalten AB and Gluetec AB for their support in practical matters relating to the Mulle platform, rock bolts, and strain gauges. Finally, thanks to New Boliden AB and Agnico Eagle for allowing us to perform real-world experiments in fully operational mines. This provided us with an in-depth understanding of today's challenges in mining.

References

[1] M. Castro, A. J. Jara, and A. F. G. Skarmeta, "Smart lighting solutions for smart cities," in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, March 2013, pp. 1374–1379.

[2] A. R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[3] J. Eliasson, R. Kyusakov, and P.-E. Martinsson, "An Internet of Things approach for intelligent monitoring of conveyor belt rollers," in *International Conference on Condition Monitoring and Machinery Failure Prevention Technologies - CM2013*, June 2013.

[4] D. Schulz, "Fdi and the industrial internet of things," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, Sept 2015, pp. 1–8.

[5] J. Eliasson, J. Delsing, A. Raayatinezhad, and R. Kyusakov, "A soa-based framework for integration of intelligent rock bolts with internet of things," in *Industrial Technology (ICIT), 2013 IEEE International Conference on*, Feb 2013, pp. 1962–1967.

[6] J. Eliasson, P. P. Pereira, H. Mäkitaavola, J. Delsing, J. Nilsson, and J. Gebart, "A feasibility study of soa-enabled networked rock bolts," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Sept 2014, pp. 1–8.

[7] J. Delsing, *Arrowhead Framework: IoT Automation, Devices, and Maintenance*. CRC Press.

[8] "Accenture technology: Winning with the Industrial Internet of Things," https://www.accenture.com/us-en/insight-industrial-internet-of-things, accessed: 2016-05-24.

[9] X. Luo, A. King, and M. V. de Werken, "Tomographic imaging of rock conditions ahead of mining using the shearer as a seismic source #x2014;a feasibility study," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 11, pp. 3671–3678, Nov 2009.

[10] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, vol. 4, 2001, pp. 2033–2036 vol.4.

[11] H. P. Breivold and K. Sandstrom, "Internet of things for industrial automation – challenges and technical solutions," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, Dec 2015, pp. 532–539.

[12] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May 2011.

[13] S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, and L. Veltri, "A scalable and self-configuring architecture for service discovery in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 508–521, Oct 2014.

[14] A. Dunkels, J. Eriksson, N. Finne, F. Österlind, N. Tsiftes, J. Abeillé, and M. Durvy, "Low-power ipv6 for the internet of things," in *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, June 2012, pp. 1–6.

[15] P. P. Pereira, J. Eliasson, and J. Delsing, "An authentication and access control framework for coap-based internet of things," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, Oct 2014, pp. 5293–5299.

[16] "Oma lwm2m specification."

[17] "Ipso alliance names the smart rock bolt winner of the 2015 challenge."

[18] B. Al Nahas and O. Landsiedel, "Competition: Towards low-latency, low-power wireless networking under interference," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, 2016.

[19] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6tisch: deterministic ip-enabled industrial internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, December 2014.

[20] A. I. Khan and P. Mihailescu, "Parallel pattern recognition computations within a wireless sensor network," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 1, Aug 2004, pp. 777–780 Vol.1.

Paper G

An Efficient IoT Framework for Industrial Applications

Authors:
Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Reformatted version of paper originally published in:
Submitted to IEEE Internet of Things Journal.

An Efficient IoT Framework for Industrial Applications

Pablo Puñal Pereira, Jens Eliasson and Jerker Delsing

Abstract

Internet of Things devices can be a valuable complement for condition monitoring in industrial processes. These devices offer new possibilities for industrial applications due to; low-cost deployments, no wiring required and easy integration with existing systems. However, resource-constrained IoT devices also come with several drawbacks. The most prominent are the low processing capability, limited memory and storage, low bandwidth (especially on wireless networks), and battery life (on battery-powered devices). These issues impose some limitations.

When deploying IoT technologies in industrial application, there are a few significant differences compared to the most traditional usage of IoT such as home automation, etc. An industrial application requires robustness, scalability, security and feasibility. To provide that the usage of concepts like zero configuration, adaptative performance, dynamic service deployment, access control, etc. need to be included in the framework.

This paper proposes a new framework for Industrial condition monitoring with IoT technology. Analyzing in detail the impact of energy and delays of each functionality involve in the IoT devices.

1 Introduction

The introduction of Internet Protocol (IP) on Wireless Sensor and Actuator Networks (WSANs) started at the middle of 90s. There were few implementations and designs of architectures to provide IP technology to the sensor- and actuator nodes, as shown by Corson et al. [1] in 1999. The implantation of IP technology had severe limitations for the nodes of the network, which were extremely constrained resource devices. The application protocol at that point, to run over TCP/IP, was the Hypertext Transfer Protocol (HTTP). The complexity of this protocol was too high for those nodes with limited memory and processing capabilities; in fact, if the nodes could work with, the power consumption was a limitation for the battery capabilities of that age, counting the lifetime of the device in hours. Another factor was the memory consumption to handle HTTP messages.

The shortage of Application protocols to run over IP was quickly covered by RESTful HTTP [2] in 1997, MQTT [3] in 1999, Jabber in 1999 which was the precursor of XMPP [4] in 2004, WebSocket [5] in 2011 and finally CoAP [6] in 2014. At the same time, the link-layer also evolved with the introduction of 6LoWPAN [7], enabling IPv6 to low

power wireless networks. Both evolutions and a natural hardware development drove the rapid growth of IoT. Modern technologies, therefore, allow to deploy CoAP-based WSANs with Internet connectivity, where the nodes can actuate as CoAP servers, being able to offer services (i.e. resources) to other nodes, users, or other machines on the network.

This increase of interoperability, and especially on Industrial IoT network, also increases the number of issues to address like; security, access control, scalability, dependability and energy efficiency, as reported by Stankovic [8]. For this reason, new low-power mechanisms to address these problems are necessary.

Industrial applications which require the use of IoT can have different lifetime, from weeks to years, being the last one the most critical situation for a wireless and battery power device. A battery replacement in the industrial environment usually is not easy, and represent an increment of the economic cost. To reduce the number of replacements or to avoid that, the IoT devices must be efficient.

IoT systems need to protect the communications and services, and the use of cryptographic mechanism is the solution, but it can compromise the balance of performance and low-power. An access control method protects the services against malicious users but also provides the tools to create customized services. Today's standard solutions like Kerberos, RADIUS, etc. require the use of a particular protocol, and some of them are not optimal for low-power devices. For a CoAP-based system, the devices must implement an additional protocol for access control, increasing the memory and processing. The use of additional communication and big size packets is other of the problems of using modern standardized access control mechanism. Therefore, a new access control mechanism for low-power CoAP-based networks is needed. Another challenge for IoT systems in Industrial applications is the dependability. Today, there is no commonly used standardized solution for this type of network which requires low-power mechanisms.

This paper addresses this technological gap, to make IoT CoAP-based technologies feasible for Industrial applications. Presenting in this paper new results in the following areas; a) device life-cycle management: including bootstrapping and configuration, b) security, and c) a feasibility analysis of the use of standard IoT technologies in industrial applications.

This paper is structured as follows: Section 2 presents the background and related work. Section 3 presents an overview of Industrial Internet of Things. Sections 4 presents experimental results, followed by suggestions for future work. Finally, the paper's conclusions are presented in Section 7.

2 Background and Related work

This section introduces a brief background of Service Oriented Architecture (SOA) and industrial IoT aspects, concepts that are broadly used in the paper.

Service Oriented Architecture (SOA)

Service Oriented Architecture (SOA) bases on a group of entities providing services to other entities. In this architecture, each entity can be designed as a separate system, splitting all the architecture's desing into small parts (Loose Coupling). This division increases the flexibility to design each part, reducing the complexity of the implementation. It allows to reduce and locate possible problems in the complete system improving the scalability and reusability.

The introduction of CoAP into IoT enables the possibility to deploy services (resources) on each node, providing the data as service, this architectural design is considered a Service Oriented Architecture. In the past, most of the WSANs have been based on client nodes sending data to a centralized server; the use of SOA enables the possibility to the deployment of customizing services, data transmission on demand with the introduction of event-based services, etc.

One of the biggest challenges to enable SOA on resource constrained IoT devices is the security. The services require protection against malicious users to protect privacy and integrity. The use of encryption increases the processing overhead, and therefore, energy consumption and delays.

Industrial Internet of Things

The term 'Wireless Sensor Networks' (WSN) has been used for the last 15 years to describe the technology composed a potential number of low-power, wireless sensor nodes. These nodes, together with gateways, performing some sort of sensing application. As WSN:n evolved, the next natural step was to not only to connect them to the Internet using a gateway but actually to utilize Internet technologies such as TCP/IP down on the nodes [9]. Today, new generations of resource-efficient protocols such as 6LoWPAN and CoAP, EXI and CBOR are now enabling true Internet of Things networks.

The use of (wireless) sensor and actuator platforms has also be embraced by the industry. Modern technologies, for example, Wireless HART, has been widely deployed at industrial sites.

In 2013 at the Hannover Fair, the German working group Industry 4.0 presented their report for the future fourth generation production and manufacturing. Industry 4.0 includes technologies such as (Industrial) Internet of Things and Cyber-physical Systems.

Industrial usage of Internet of Things adds more requirements than the consumer market. Most notable differences are;

- Scalability - Systems for industrial process monitoring and control can be composed of tens of thousands of sensors.
- Security - A security breach in a factory can cause damage to the environment, humans, and inflict major costs.
- Interoperability - A factory system most often uses a number of different systems

and technologies, This complicates information exchange as mediators or translations must be used.

The issues above are usually not found or not as severe in smaller, consumer-based, installation for home automation, security, sports, etc. Security is of course always important, however in an industrial facility, the costs for a security breach can be much higher compared to a home-automation system.

3 Proposed Industrial IoT framework

This section provides a detailed description of the proposed framework for industrial monitoring and control applications. The framework consists of a network architecture, services with data models as well as tools. The framework also provides security mechanisms for fine-grain access control and authentication.

Network Architecture

This framework was designed for resource constrained IoT Wireless Sensor and Actuator Networks, a mesh network topology will affect the performance, bandwidth, range, power consumption, etc. At this point, the design of the architecture is a compromise between:

- The area to cover: the size and the environment will demand to deploy more access points (gateways). For wireless networks the range is a technology limitation, each access point can cover a portion of the total area, then there is a direct relation between the total area to cover and the number of access points to deploy. On the other hand, there is an environmental/physical limitation. The allocation close to walls, pipes, tunnels, water tanks, other networks, high-voltage networks, high temperature, radiation, etc. can affect the range and the quality of the network, so in these cases, a reinforce with more gateways will be needed.

- Lifetime, crucial only on battery powered devices, and expected data rate: must be a balance between these two aspects, and increment of data rate affects the duration of the battery power devices. The data rate requisites determine the type of wireless technology to use. Examples of radio technologies for wireless sensing are IEEE 802.15.5, WiFi, WirelessHART, Z-wave, etc.

The proposed network topology is represented in Figure 1, the communication between each gateway and its nodes is a tree with a minimized number of hops. This limitation is because sometimes the wireless range between a gateway and a node is not enough and is better to have a hop between both. Each hop-node increases the power consumption because it needs to address its traffic and also the traffic of other nodes.

The communication between gateways and nodes is a wireless network based on the network stack shown in Figure 2. It consists of a 6LoWPAN layer over IEEE 802.15.4

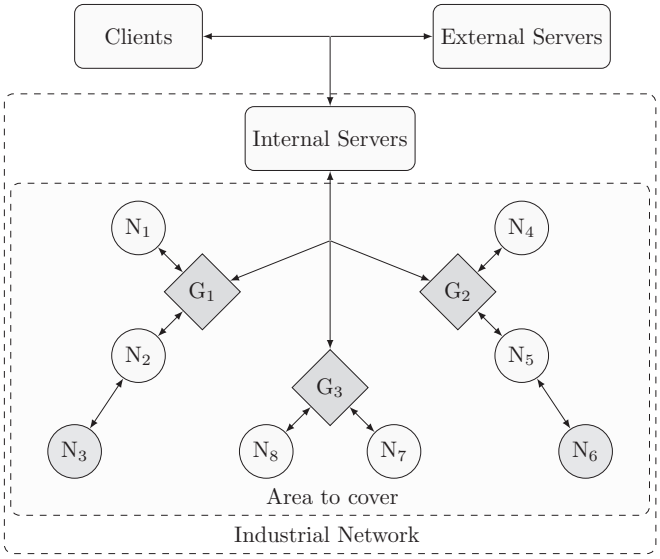


Figure 1: Proposed network topology

to enable the IP protocol; the IP protocol also supports IPsec to protect the communications, but can include other encryption mechanism like TLS/DTLS, MAC encryption, etc. The application protocol is CoAP, which provides the deployment of services/resources on each node.

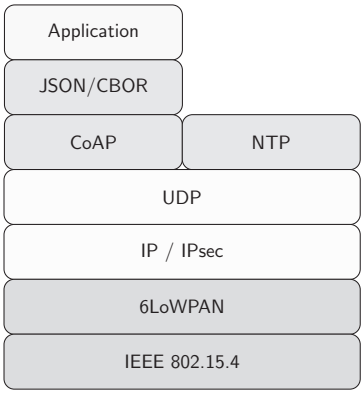


Figure 2: Proposed network stack

The communication between the gateways and the servers is not part of this research, in our proposed network it can be done by wireless or cable technologies because the power consumption is not relevant at gateway's level.

Functional Architecture

Each element type of the topology has a different role; this section describes the functionalities of each type.

Nodes

The Nodes are resource constrained devices with wireless connectivity and usually battery powered. The connectivity allows them to communicate with other nodes, servers or clients by the wireless connectivity provided by the gateway. The essential task of a node is to sense a physical variable and use an actuator to produce a physical environmental change. With the improvement of the microcontrollers, the resource constrained devices can run complex tasks, like analyze and evaluate the data with filters, find relevant profiles, apply adaptable triggers, etc.

Configure each node manually in an Industrial environment is not feasible, for that reason each node can ask for its configuration during the boot time; even if the configuration changes in runtime, the node must be able to reconfigure itself. This process is described in more detail in Section 3.

The communication usage changed from the past; the Nodes are not only clients. In fact, with the use of CoAP, each Node can create dynamically services and provides customized services, to do it the use of access control is mandatory.

On the proposed platform a single node can provide services based on collective data, like average of temperature of closed nodes, or take actions based on data from other nodes. In other words, the nodes can create systems of systems, to implement that a direct communication machine to machine (M2M) is mandatory.

Gateways

The Gateways are embedded computers with mainly two different tasks: provide the wireless connection to the nodes, acting as a standard gateway and running a lightweight Supervisory Control and Data Acquisition (lw-SCADA). The lw-SCADA is the responsible to register each connected node to the Device Manager (DM), provide the Bootstrapping, Configuration, Authentication, and Authorization, for the Access Control. All these services are a replication of the services on the Internal Server, to decentralize and work even if the connection to the Internal Server goes down. Each service is described in detail in Section 3.

The Gateways create a 6LoWPAN connection to the nodes and an Ethernet/Wifi connection to the Internal Server. Therefore, the communication between Gateway and Nodes is with CoAP and the communication between Gateway and Internal Server with

HTTP/HTTPs. The Gateways also extends the IPv6 network of the nodes to the internal network, so the Internal Server can also communicate directly with each Node.

Internal and External Servers

The Servers are not part of this paper research, but a little definition of the functionality is needed to understand better the platform. There are no differences between Internal or External Servers; even an External Server can act as Internal with a VPN connection to the internal network. The name is a tag to distinguish between exposed or not exposed Servers to external links like The Internet. Servers act as Gateways with more power of computation, memory, and connections. The light weight limitation disappears, and Servers actuate as complete SCADA systems. The biggest difference between a Server and a Gateway is that a Server can provide the communication between nodes of different Gateways.

Security

Each device with processing, memory and communication capabilities is susceptible to be an attack target. Industrial IoT is a very attractive environment, with hundred thousands of devices connected offering a high rate of relevant data flowing and access to thousands of actuators. Then, an attack can steal data and even modify some actuator, producing even bigger issues. In this context, a security breach can represent a significant economic loss for a company. For these reasons, the Security and Privacy must be one of the topics with relevance for the platform's design, as was demonstrated by Sadeghi et al. in [10] and Roman et al. in [11]. This protection can be achieved with the usage of Encryption and Access Control Systems.

Encryption

During the last two decades, Internet Protocol has been broadly used, and Nowadays there are many solutions to protect the communications; most of them designed for computers which are not resource constrained devices as IoT. Even solutions designed principally to IoT communications like [12] and [13] are not feasible for resource constrained devices because of the power consumption and delays.

Hennebert et al. analyze in [14] the standard options to protect an IoT device with a communication stack based on 6LoWPAN, and also according to Alghamdi et al. [15], the options to protect the communication are IPsec and DTLS. The performance between both was analyzed previously by De Rubertis et al. [16] with a minor overhead for IPsec, but nowadays with DTLS header's compression like proposed by Raza [17] the performance between both is similar.

For final application point of view, the usage of DTLS is taking more relevance than IPsec; it is easier to integrate with servers, and the application layer can recognize when a communication is secure, which does not happen with IPsec. In contrast, IPsec encap-

sulates all the IP packet, protecting UDP and TCP communications and it also enables the possibility of tunneling the communications.

Access Control

The use of Access Control methods is mandatory for many reasons; all the communication layers bellow CoAP are not able to implement a fine-grained access control, which is needed to enable for example different ranks of privileges per service like administrator, users, guests, etc. allowing different actions depending on the rank level. Therefore, each device must know the access policies for each new client request. IKEv2 can provide an authentication by device name to initiate the communication, but not control at service and method levels. IPsec can provide authentication of the device, but not control at service and method levels. Even the use of DTLS can not provide an access control at service and method level.

IPsec can control who access to the IoT device, but are not able to monitor the access to each single service, or even worst, it does not share information with the application layer, so the application can not recognize the user as a particular user. On the other side, DTLS shares access information about the user to the application layer. But there is no information regarding service's permissions. For these reasons, a fine-grained access control is required.

Today's solutions like RADIUS [18], Diameter [19] or Kerberos [20], require the use of other protocols, increasing the communication stack and the memory on each node. Kerberos solution is based on the use of tickets, but the complexity of a complete authentication process is too high for a low-power IoT device (high communication overhead).

Mandatory Services

This section describes all the mandatory services that the proposed platform must have to cover all the requirements of an IIoT network.

Bootstrapping

The bootstrapping service compliant the LWM2M OMA Bootstrapping [21], this provides information to the IoT devices about essential instances like Access Control, Configuration and the LWM2M Server.

The bootstrapping service must run on the Gateway and should use a predefined port, because this is the required information that a node can have during the first boot. In other words, the bootstrapping service must be accessible for any other IoT device who joins the network.

When a device starts the bootstrapping request, it can includes on the request some extra information like serial number, MAC address, internal software name or version, etc. With this information the framework can distribute the IoT devices between different Access Control, Configuration or LWM2M Servers; this can help to balance the overload

between different servers or to include devices with different versions and keep both minimizing conflicts.

The use of the bootstrapping increases the stability and robustness of the framework if a service is down it can be replaced by another just changing the IP or port on the bootstrapping response. Also, it supports multiple endpoints for the same service, and in the case that one is down the device can use the next one. And in the case that one node is connected to another wireless network, with different services or different network routes, everything will work as usual.

The penalty for the use of this technology regarding communication is a single request per booting but also it requires the implementation of a parser on the device, which consumes some memory.

The following example is done with JSON (Code G.1), but it can be implemented in CBOR reducing the packet size.

Code G.1: Bootstrapping example

```
{
  "auth": {
    "ip": "fdfd::0A",
    "port": 5683,
    "v": 1,
    "res": "/Authentication",
    "resAlt": "/Authorization"
  },
  "conf": {
    "ip": "fdfd::0B",
    "port": 5682,
    "v": 1,
    "res": "/Conf"
  },
  "dev": {
    "ip": "fdfd::0C",
    "port": 5681,
    "v": 1,
    "res": "/rd"
  }
}
```

JSON: 305 bytes - CBOR: 147 bytes

Configuration

WSANs uses sensors and actuators, the role of a sensor is to take measurements of physical variables and the role of an actuator is to take actions on physical variables. The Configuration service sets the parameters of how that measurement are done, how should be the data analysis, and decides which actions must take if some under certain conditions. For example, set the sample rates, triggers, set filters dynamically, send alert to other devices, collaborative analysis,

etc.

The configuration also sets the services that must be active on the device, based on the task to develop. This creation of services on demand improves the performance and the reusability of each node. But also, it can be used as security countermeasure and as a way to reduce the power consumption.

Several are benefits of the use of Configuration services, and mainly all of them are focused to optimize at maximum each device and to create collaborative applications. The power consumption of each device can be reduced, but the complexity of how to program the services rises, this complexity has direct negative impact on the program's size.

Code G.2: Configuration example for a Temperature IoT device

```
{
  "Services": [
    {
      "name": "TempService",
      "type": "temperature",
      "source": "sens1",
      "interface": {
        "GET": {
          "active": true,
          "return": "sens1"
        },
        "POST": {
          "active": false
        },
        "PUT": {
          "active": true,
          "receive": "trigger",
          "return": "trigger"
        },
        "DELETE": {
          "active": false
        },
        "OBSERVABLE": {
          "active": true,
          "period": 120,
          "return": "sens1"
        }
      }
    }
  ],
  "Actuators": [],
  "Sensors": [
    {
      "name": "sens1",
      "period": 60000,
```

```
36          "triggered": "yes"
37      }
38    ]
39  }
```

JSON: 692 bytes - CBOR: 268 bytes

Access Control

The implemented Access Control on the framework is based on the ticket based authentication and authorization that Puñal et al. presented in 2014 [22]. This implementation addresses the issue discussed in Section 3, providing an efficient fine-grained access control mechanism.

Required features

This section presents an overview of some of the required features that are vital for the framework to operate properly.

Device Manager

Have listed all the network devices and detect when a new device is connected are part of the task of a Device Manager. The Open Mobile Alliance (OMA) proposed the use of OMA Lightweight Machine to Machine (LWM2M) [23] protocol to address the device management. Today there are two widely used solutions: Leshan [24] and Wakaama [25]. Both are supporting a wide variety of standard LWM2M features.

Supervisory Control and Data Acquisition (SCADA)

The SCADA system running on the gateway is the responsible to detect when a new node connects, providing all the services for Bootstrapping, Configuration, and Authentication. Once a node is registered, the SCADA system recognizes the services running on the node, and it starts to subscribe to all of them. At this point, the SCADA will receive the data from the node when an event happens, and can analyze and save the data in a logger. In fact, if the SCADA detects that an action is needed, it will communicate that to the appropriate service on the corresponding node.

4 Test and Results

This section provides an overview of all performed experiments and obtained results.

Test scenario

The test scenario is a real condition test, with a gateway running the lightweight SCADA software and a single node connected to it. The test does not include the radio effects for other nodes as well as the network traffic effects.

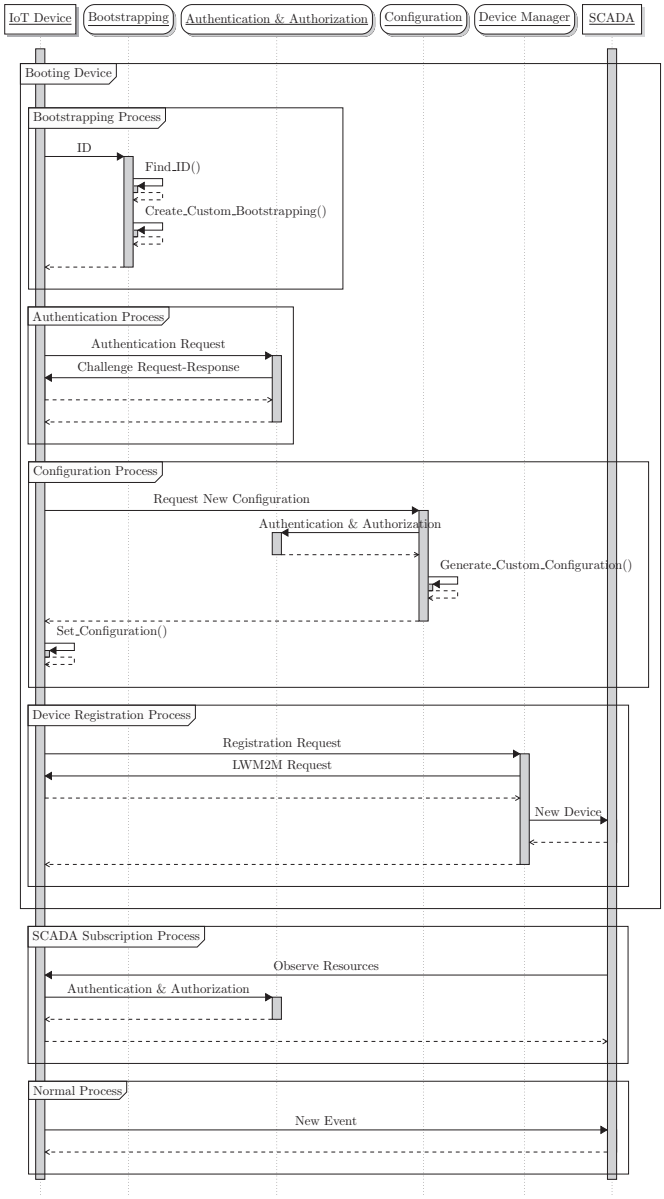


Figure 3: Framework behaviour

Test setup

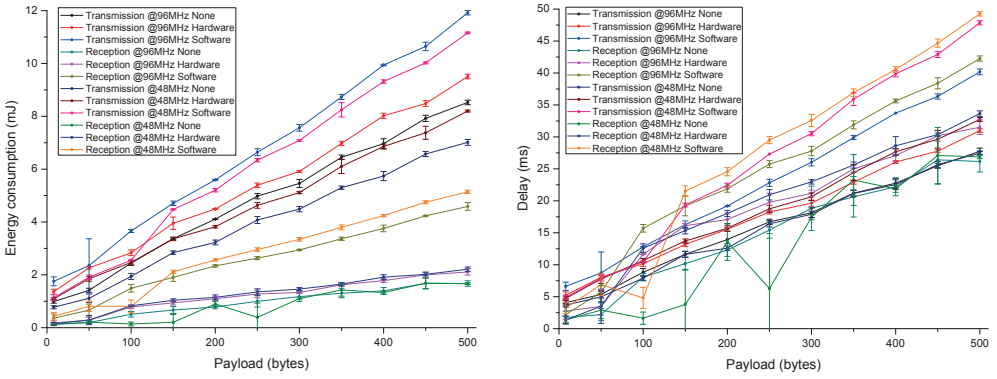
This section is based on experimental results of energy consumption and delays. The benchmark configuration relies on measures of battery current and voltage externally to the device; these

measurements are done using a 16-bit ADC at 1840 Hz to capture rapid events such as radio, wakeups, etc. All these measurements are combined to 8 digital inputs that can be used to recognize in detail the power consumption of each software module.

The selected IoT platform to do the test was a Mulle from Eistec AB[26], which is equipped with an ARM Cortex-M4 at 100 MHz microcontroller and an IEEE 802.15.4 transceiver. It has an onboard 2 MB of flash memory and 256 KB of internal memory on the microcontroller. The Mulle runs the open-source Contiki OS [27]; so all taken measures are affected by running an OS on the same device without any isolation to get real condition data. Those effects involve operating system's energy consumption with peaks of internal queues, communications, internal timeouts, and events, etc. which in turn affects the error level of the measurements.

Results

As was discussed in previous sections, Security is a crucial feature for IoT communication. To analyze in detail the energy consumption and delays of the IPsec ESP different configurations have been tested. The settings for ESP are AES128-CTR and AES-XCBC. Figure 4 compare the energy consumption (a) and delays (b) off communication with different CoAP payloads of none protected communication and IPsec with and without hardware acceleration (AES 128) for two CPU speeds (96 MHz and 48 MHz).



(a) Energy consumption of transmission and reception at 96MHz and 48MHz

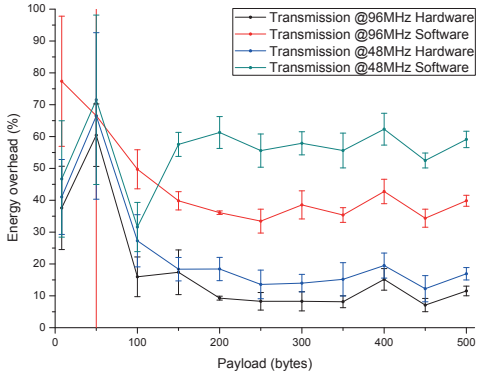
(b) Delay of transmission and reception at 96MHz and 48MHz

Figure 4: Analysis of IPsec-ESP communication in energy consumption and delays

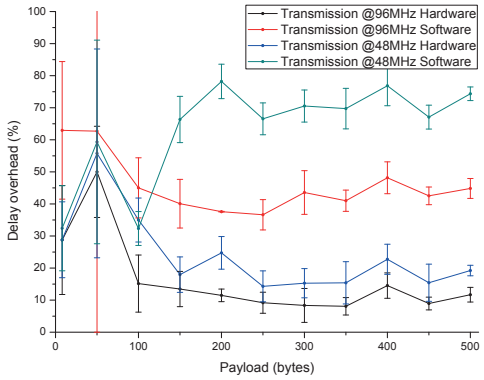
A more detail analysis of the data can provide the overhead in energy and delays for transmissions (a and b) and delays (c and d) on Figure 5.

The next step is to compare the overheads of all the services that the nodes need to use. The configuration of each service is as follow:

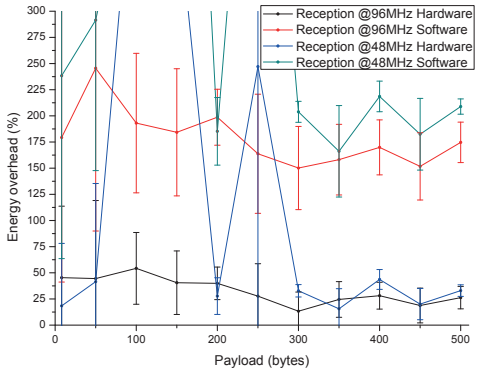
- Internet Key Exchange v2 (IKEv2). The key exchange has two steps: Initialization and Authentication, for that reason both are analyzed separately. The IKEv2 configuration



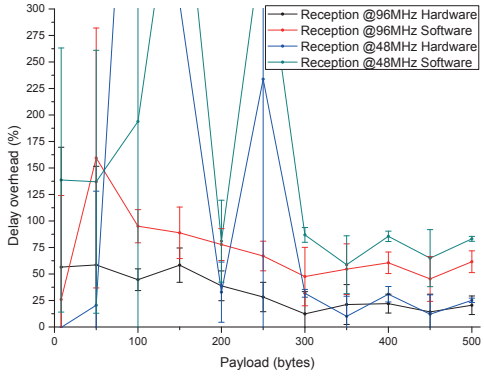
(a) Energy overhead for transmission at 96MHz and 48MHz



(b) Delay overhead for transmission at 96MHz and 48MHz



(c) Energy overhead for reception at 96MHz and 48MHz



(d) Delay overhead for reception at 96MHz and 48MHz

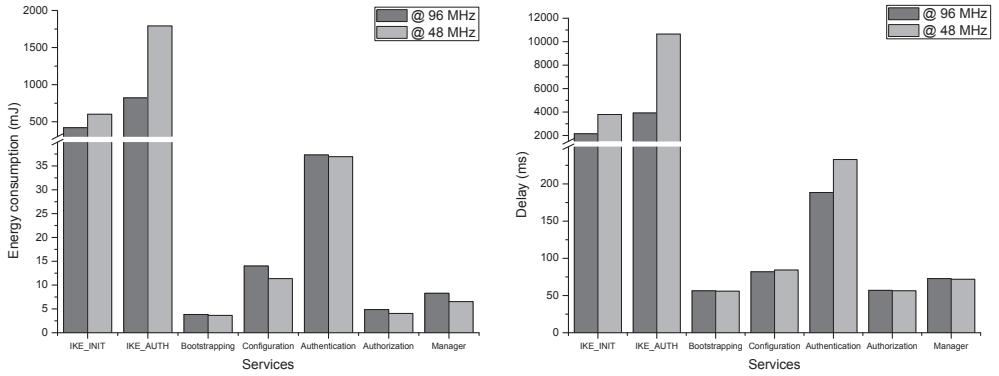
Figure 5: Analysis of IPsec-ESP communication in overheads

is AES128-CTR + AES-XCBC + SHA1 + ECP192.

- Bootstrapping. The analysis includes the overhead of the Bootstrapping request and parsing of the code (similar to Code G.1).
- Configuration. The analysis includes the Configuration request, parsing of the configuration (like Code G.2), the configuration of the sensing and service deployment.
- Authentication. The analysis includes the Authentication request, the Challenge Request-Response, and the parsing of the Ticket and attributes.
- Authorization. The analysis includes the first authorization request and the parsing of timeouts and permission.

- Device Manager. The analysis includes the registration of the node into the OMA LWM2M Server.

Figure 6 shows the energy consumption and delays of each service for two CPU speeds (96 MHz and 48 MHz). The Table 1 offers the values of the experiment.



(a) Energy consumption of each service at 96MHz and 48MHz

(b) Delay of each service at 96MHz and 48MHz MCU frequency

Figure 6: Analysis of each service in energy consumption and delays

Service	Power (mW)		Delay (ms)		Energy (mJ)	
Speed (MHz)	96	48	96	48	96	48
IKE_INIT	195,74	158,65	2145,63	3789,22	419,99	601,15
IKE_AUTH	209,85	168,19	3916,53	10650,12	821,88	1791,25
Bootstrapping	68,00	65,23	56,40	55,85	3,84	3,64
Configuration	170,90	134,73	81,94	84,29	14,00	11,36
Authentication	197,99	158,69	188,46	232,83	37,31	36,95
Authorization	74,80	71,76	56,96	56,41	4,86	4,05
Dev. Manager	113,76	90,89	72,81	71,87	8,28	6,53

Table 1: Analysis of power consumption, delays and energy overheads per service

Summary

Figures 4 and 5 show an erratic performance with small payloads; this is the effect of having the Contiki OS running on the device. With small payloads, the consumption is reduced; Therefore, a variation in the OS consumption has a bigger effect on the measurement. For payloads over

200 bytes, it is possible to see a constant overhead, which must be the number that we need to use as the reference.

5 Discussion

According to the results, this paper concludes that there are three types of energy consumption: Initial, Constant, and Event.

- The Initial consumption represents the energy consumption that a device needs to initiate all the modules, such as Bootstrapping, Configuration, initial key negotiation, etc. This overhead is static and only can be reduced deactivating those services. Based on the experimental data, the Initial consumption is practically irrelevant compared with the consumption in a long term test.
- The Constant consumption represents the energy consumption that a device needs to work properly over the time without events, in other words, it is the energy that a device needs to be alive. The Constant consumption depends directly on the final application, and it is extremely configurable, it depends on awake-sleep cycles, sensors' configuration, actuators' configuration, IKE configuration (timeouts, encryption type, etc.), authentication and authorization (ticket timeouts), etc.
- The Event consumption represents the energy consumption that a device needs to detect and notify an event to the clients (SCADA or other nodes), the Event consumption directly depends on the detection algorithm, triggers, and especially the environmental conditions.

The three types of energy consumption are represented at Figure 7, where the Initial consumption is represented by light-gray color, Constant consumption by medium-gray color and Event consumption by dark-gray color.

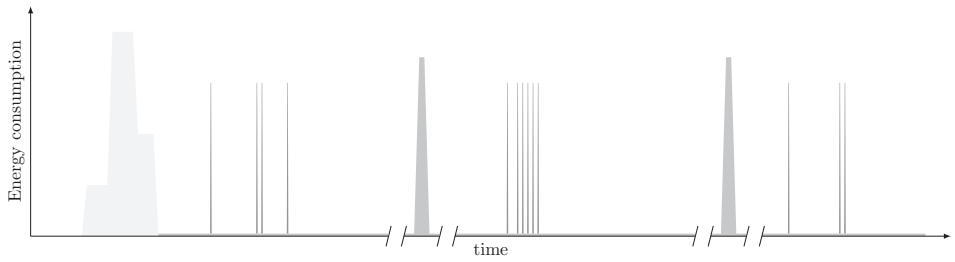


Figure 7: Energy consumption profile of a typical IoT device

As was demonstrated at Section 4 (Figure 6), the IKE negotiation is the elements with highest power consumption of all the system and it is part of the Constant consumption. So, the balance between security and power consumption is mandatory; In other words, if a system

is refreshing quite often the keys the energy consumption rise and the battery's lifetime of the device is compromised. The same balance is required for Authentication and Authorization; the timeout of the tickets is also configurable, but in this case, the consumption is not critical.

One of the biggest problems for event-based WSANs is that the events depend on the environment, and it is not predictable, so create smart nodes with high logic level to determine when a relevant event occurs is crucial. This area is a possible way for future work, to integrate for example low-power neural networks to improve at maximum the data veracity.

6 Future work

A possible continuation of this work can include an analysis of a complete system with Bootstrapping and dynamic configuration to deploy on the nodes dynamic collaborative services.

Another suggestion for future work is the design of an IoT-version of the IKE protocol. Even though the overhead of IPsec is reasonable, IKE imposes a significant problem regarding energy consumption and time-consuming key negotiation. A light-weight version of IKE designed specifically for resource-constrained embedded devices would improve the overall IPsec performance substantially. As a part of the evaluation of IoT-IKE, it would be interesting to perform a holistic performance evaluation between DTLS vs the new IPsec-based eco-system. As a part of this evaluation, we propose that application scenarios involving device to device, device to the gateway as well as device to external communication.

7 Conclusions

In this paper, an efficient framework for (Industrial) Internet of Things system has been presented. The framework includes features for low-power wireless communication, security mechanisms, bootstrapping and advanced run-time configuration.

A secure interoperability between nodes is feasible regarding power consumption and delays, and enables distributed and collaborative services in IoT networks.

Regarding power consumption and delays, there is one clear conclusion; "The use of hardware acceleration for encrypted communications should be mandatory in an IoT device". These results demonstrate that the use of IPsec for IoT is feasible, as the overhead is low, and is a recommended solution if the device needs to protect both UDP and TCP communication protocols. If not, DTLs is a better option because of the easier integration to cloud servers and because the application layer has access to the DTLS and it can distinguish between secure and unsecure communications.

The tested device (Mulle mk6) with Contiki OS consumes 0.25mW with no communication and no sensing processes. Regarding the results of this paper, with the use of a standard battery (3.7V at 2000mAh), the battery life can be up to 2 years with the following configuration:

- daily keys negotiation
- hourly authentication
- ten encrypted event reports per hour (application dependant)
- daily re-configuration

In summary, the proposed framework offers protection for communication (IPsec+IKEv2), an access control mechanism with easy integration with already standard solutions like RADIUS, zero-configuration operation, dynamic services, secure interoperability node-to-node, and reconfiguration at runtime with a multi-year battery's lifetime. This shows that standard IoT technologies are feasible for use in the Industrial Internet of Things.

8 Acknowledgment

The authors would like to express their gratitude towards the European Commission, Artemis and the Arrowhead project for funding.

References

[1] M. S. Corson, J. P. Macker, and G. H. Cirincione, "Internet-based mobile ad hoc networking," *IEEE Internet Computing*, vol. 3, no. 4, pp. 63–70, Jul 1999.

[2] J. Mogul, L. M. Masinter, R. T. Fielding, J. Gettys, P. J. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, Mar. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc2616.txt

[3] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s - a publish/subscribe protocol for wireless sensor networks." in *COMSWARE*, S. Choi, J. Kurose, and K. Ramamritham, Eds. IEEE, 2008, pp. 791–798. [Online]. Available: http://dblp.uni-trier.de/db/conf/comsware/comsware2008.html#HunkelerTS08

[4] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 3920, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc3920.txt

[5] A. Melnikov and I. Fette, "The WebSocket Protocol," RFC 6455, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc6455.txt

[6] D. C. Bormann, K. Hartke, and Z. Shelby, "The Constrained Application Protocol (CoAP)," RFC 7252, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[7] P. Thubert and J. Hui, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc6282.txt

[8] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, Feb 2014.

[9] A. Dunkels, J. Eriksson, N. Finne, F. Österlind, N. Tsiftes, J. Abeillé, and M. Durvy, "Low-power ipv6 for the internet of things," in *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, June 2012, pp. 1–6.

[10] A. R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[11] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, Sept 2011.

[12] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the iot," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 725–730.

[13] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "Enabling data protection through pki encryption in iot m-health devices," in *Bioinformatics Bioengineering (BIBE), 2012 IEEE 12th International Conference on*, Nov 2012, pp. 25–29.

[14] C. Hennebert and J. D. Santos, "Security protocols and privacy issues into 6lowpan stack: A synthesis," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 384–398, Oct 2014.

[15] T. A. Alghamdi, A. Lasebae, and M. Aiash, "Security analysis of the constrained application protocol in the internet of things," in *Second International Conference on Future Generation Communication Technologies (FGCT 2013)*, Nov 2013, pp. 163–168.

[16] A. D. Rubertis, L. Mainetti, V. Mighali, L. Patrono, I. Sergi, M. L. Stefanizzi, and S. Pascali, "Performance evaluation of end-to-end security protocols in an internet of things," in *Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on*, Sept 2013, pp. 1–6.

[17] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure coap for the internet of things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, Oct 2013.

[18] X. Luo, "The realization of the radius security authentication," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, Oct 2008, pp. 1–4.

[19] S. B. Ayed and F. Teraoka, "Diameap: An open-source diameter eap application and its evaluation," in *2010 16th Asia-Pacific Conference on Communications (APCC)*, Oct 2010, pp. 464–469.

[20] E. El-Emam, M. Koutb, H. Kelash, and O. F. Allah, "An optimized kerberos authentication protocol," in *Computer Engineering Systems, 2009. ICCES 2009. International Conference on*, Dec 2009, pp. 508–513.

[21] O. M. A. (OMA), "LWM2M OMA - Bootstrap Interface," http://dev_devtoolkit.openmobilealliance.org/IoT/LWM2M10/doc/TS/index.html#!Documents/bootstrapinterface.htm.

[22] P. P. Pereira, J. Eliasson, and J. Delsing, "An authentication and access control framework for coap-based internet of things," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, Oct 2014, pp. 5293–5299.

[23] OMA, "Lightweight Machine to Machine (LWM2M) v1.0," http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0.

[24] “Eclipse Foundation - Leshan LWM2M Server,” http://www.eclipse.org/leshan/.

[25] Intel, “Wakaama LWM2M Server,” https://projects.eclipse.org/projects/technology.wakaama.

[26] Eistec, “Mulle IoT platform,” http://http://www.eistec.se/.

[27] “Contiki OS,” http://www.contiki-os.org/.