Retrieving Monitoring and Accounting Information from Constrained Devices in Internet-of-Things Applications

Oleksiy Mazhelis, Martin Waldburger, Guilherme Machado, Burkhard Stiller, Pasi Tyrväinen

Retrieving Monitoring and Accounting Information From Constrained Devices in Internet-of-Things Applications

Oleksiy Mazhelis[1], Martin Waldburger[2], Guilherme Sperb Machado[2], Burkhard Stiller[2], and Pasi Tyrväinen[1]

[1] Department of Computer Science and Information Systems, University of Jyväskylä, FI-40014 Jyväskylä, Finland
[oleksiy.mazhelis|pasi.tyrvainen]@jyu.fi
[2] Department of Informatics (IFI), Communications Systems Group (CSG) University of Zürich, CH-8050 Zürich, Switzerland
[waldburger|machado|stiller]@ifi.uzh.ch

Abstract. Internet-of-Things (IoT) is envisioned to provide connectivity to a vast number of sensing or actuating devices with limited computational and communication capabilities. For the organizations that manage these constrained devices, the monitoring of each device's operational status and performance level as well as the accounting of their resource usage are of great importance. However, monitoring and accounting support is lacking in today's IoT platforms. Hence, this paper studies the applicability of the Constrained Application Protocol (CoAP), a lightweight transfer protocol under development by IETF, for efficiently retrieving monitoring and accounting data from constrained devices. On the infrastructure side, the developed prototype relies on using standard building blocks offered by the AMAAIS project in order to collect, pre-process, distribute, and persistently store monitoring and accounting information. Necessary on-device and infrastructure components are prototypically implemented and empirically evaluated in a realistic simulation environment. Experiment results indicate that CoAP is suited for efficiently transferring monitoring and accounting data, both due to a small energy footprint and a memory-wise compact implementation.

1 Introduction

Internet-of-Things (IoT) represents a vision for the future of information and communications technology wherein a variety of real-world "things" will be interacting and communicating with other virtual and physical entities through a global Internet infrastructure [1]. It is envisioned that up to 50 Billion of devices will be Internet-connected by 2020 [2]. Many of these devices will be so-called constrained devices, *i.e.*, devices with constraints on their memory size, computing power, communication capabilities, and/or available power [3].

In some application scenarios, *e.g.*, as envisioned in the A^4-Mesh project[3], constrained devices are deployed in remote areas, where they run unattended for

[3] https://a4-mesh.unibe.ch/

months or years, possibly without mains power. Users of these devices should have the possibility to *monitor* their operational status – *e.g.*, available memory, remaining energy level, or detected errors – without the need to physically visit the deployment site. Resource usage may need to be *accounted*, too, both for network management purposes and, especially in case of multiple organizations requesting information from devices, for any potential charging and billing purpose. These information needs necessitate the implementation of an infrastructure to support an efficient monitoring and accounting for constrained devices.

The resource constraints of wireless sensor nodes impose constraints on suitable monitoring and accounting approaches and communication protocols. Since monitoring and accounting involves on-device metering of a device's technical parameters and resource usage, this metering process should be conservative in memory, code space, and power expenditure, to accommodate limited capabilities and restricted battery power of the devices. This may render an application of traditional approaches – *e.g.*, those based on Simple Network Management Protocol (SNMP) [4] – suboptimal for devices with strong resource constraints.

Hence, this paper aims at extending IoT applications with monitoring and accounting functionality. Rather than introducing a new communications protocol to implement this functionality, this paper focuses on studying the applicability of a standard communication protocol for constrained devices – in particular, the Constrained Application Protocol (CoAP) that was introduced as a lightweight alternative to HTTP and optimized to work in constrained environments [5]. The research question addressed in this paper can be formulated as follows: *Is it feasible to utilize CoAP as a transfer protocol for retrieving the information needed for monitoring and accounting purposes as metered on constrained devices?*

To support the efficient processing and storage of this information, this work relies on using the accounting and monitoring infrastructure produced by the Accounting and Monitoring of Authentication and Authorization Infrastructure Services (AMAAIS) project [6] which provides components for accounting and monitoring of IT services. In response to the research question, the design and implementation of a prototype has been undertaken. Specifically, the functionality of AMAAIS is extended with (i) a metering component deployed on constrained devices in order to obtain relevant operational parameters and resource usage information, and (ii) a networked accounting application component. This networked accounting application is integrated with the core AMAAIS infrastructure and communicates with the metering component using CoAP as a transfer protocol. Finally, the performance of the prototype is evaluated regarding memory footprint and power consumption of the on-device metering process.

The reminder of this paper is organized as follows. In the next section, relevant terminology is introduced, and related work in the domain of constrained device management is overviewed. Key elements of the AMAAIS infrastructure are described in Section 3, along with the changes needed to make it applicable to a constrained environment. Details of the prototype implementation and its performance evaluation are presented in Section 4. Finally, in Section 5, obtained results are summarized, and directions to further work are outlined.

2 Background and related work

This section provides definitions for relevant terms, and overviews management mechanisms available for constrained devices in general and for retrieving monitoring and accounting information in particular.

Terminology. Constrained devices can be categorized into (i) Class-0 devices ($\ll$10 kByte of RAM and $\ll$100 kByte of ROM) only capable of engaging in simple communication scenarios with the help of a proxy or gateway, (ii) Class-1 devices ($\approx$10 kByte of RAM and $\approx$100 kByte of ROM) powerful enough to directly communicate with their peers in the Internet via lightweight protocols, and (iii) Class-2 devices ($\approx$50 kByte of RAM and $\approx$250 kByte of ROM) whose capabilities are sufficient to support full-fledged protocols used in conventional network nodes [3]. Among these three categories, Class-1 devices, being both inexpensive and capable of communicating with their peers in the Internet, are believed to play an important role in emerging IoT applications [5]. Class-1 devices therefore determine the main focus of this work.

The management of constrained devices requires a broad set of functionality to be implemented to efficiently configure, monitor, and control them. As these devices are managed as a part of the network to which they belong, the core operational network management functions that are conventionally grouped along the five functional areas of fault, configuration, accounting, performance and security [7], are also relevant for managing constrained devices and networks thereof [8]. Many of these functions are enabled by the so-called *metering* process whereby technical parameters of a particular resource are identified and current usage of the resource is determined. The metering process can be triggered by signaling or other external polling events, or alternatively it can be performed periodically or according to a statistical sampling scheme [9].

In the context of this paper, the focus is restrained to accounting functionality and performance management (specifically, performance monitoring) functions. Extensions to include other management functions are outside of the scope of this paper – they are left for further work.

Requirements for managing networks with constrained devices are being defined by the Constrained Management (COMAN) – a recently established IETF activity [10]. On a general level of a management architecture/system, the set of requirements relevant to monitoring and accounting functions include the following needs: (i) to minimize the state maintained on constrained devices, (ii) to support devices that are not always on-line, (iii) to support lossy and unreliable links, through in-built resilience mechanisms and through a limited data rate, (iv) to keep the encoding of management data compact, and (v) to optionally compress management data or complete messages. At the implementation-level, with the aim of minimizing communication overhead, two mandatory requirements are stated:

– Avoid complex application layer transactions with large messages, since they require large memory buffers and increase the volume of re-transmissions.

– Avoid the fragmentation and reassembly of messages at multiple protocol stack layers, *e.g.*, by limiting the size of application layer messages.

Ersue et al. [10] specify a number of monitoring functions for Class-1 devices; among these, only the monitoring of device status and energy level are considered mandatory, whereas all other are optional to implement depending on the device type and management needs. The document also mandates a number of security functions, including the need for authentication and access control (both on constrained devices and in the management system) as well as the need for a security bootstrapping mechanism.

The focus in this paper is on the acquisition of dynamic information for monitoring and accounting purposes, and an assumption is taken that static management information can be obtained out of band by other means. Specifically, this paper aims at equipping IoT applications with the mandatory monitoring functionality set in [10], including device status monitoring and energy status monitoring, while also satisfying the respective other architectural- and implementation-level requirements.

Management architecture and protocols for constrained devices. The resource constraints of wireless sensor nodes impose constraints on suited communication protocols and management methods. In order to cope with the specifics of networked constrained devices, Ruiz et al. have introduced MANNA [8], a management architecture for Wireless Sensor Networks (WSN) based on manager-agent interactions. The authors specified relevant management functions and considered the WSN information model, although an empirical evaluation of the proposed architecture has not been reported. Likewise, an agent-manager interaction is assumed in the WSN management system [11] which relies on SNMP and implements configuration, performance, and fault management functions for TinyOS devices. The system has been prototypically implemented; however, the performance of the management architecture and its overhead are not reported.

A number of studies were aimed at applying standard IP tools, such as SNMP and NETCONF [12] – a protocol used for manipulating the configuration of a network device – for device management purposes. Under the assumption that standard SNMP tools require a large Management Information Base (MIB) and result in noticeable message overhead, attempts are made at optimizing the SNMP architecture and tailoring these protocols, *e.g.*, by limiting the set of functionality, or by introducing a gateway to mediate between standard protocols and tailor-made monitoring agents on constrained devices. In particular, this approach is followed by the LoWPAN Network Management Protocol (LNMP) [13], 6LoWPAN-SNMP [14], and EmNetS Management Protocol (EMP) [15].

Kuryla and Schönwälder [4] have studied the feasibility of implementing SNMPv1 and SNMPv3 message processing models as an SNMP agent for Contiki OS. To meet the hardware constraints, only Get, GetNext and Set operations were implemented, and simplifications to the modular SNMP architecture were made. The implementation reportedly has a relatively modest memory footprint and a short processing delay of 40–120 ms; however, it does not support notifica-

tions (Trap and Inform), and thus implies the need for the manager component to explicitly request information from an on-device agent. Sehgal et al. [16] have empirically compared SNMP against a lightweight version of NETCONF, without subtree filtering and the edit-config operation, in the context of constrained devices. As NETCONF relies on the exchange of relatively large XML messages, its use in constrained devices results in longer processing time, as well as in an increased memory footprint [16].

Instead of tailoring traditional protocols designed for non-constrained devices, an alternative approach is to re-use CoAP for device management purposes. CoAP is a new transfer protocol introduced as a lightweight alternative to HTTP. It is optimized to work in constrained environments [5, 17]. CoAP relies on UDP as a transport, and offers a simple in-built stop-and-wait reliability mechanism. It uses a compact four-byte binary header with a total header size of 10 to 20 Byte, and defines four methods – GET, POST, PUT, and DELETE – enabling a RESTful architectural style. Importantly, the protocol supports an asynchronous retrieval of information by using the "Observe" option: by issuing specially crafted GET requests, clients subscribe to the updates of a resource of interest; after that, the device asynchronously notifies its observers about resource changes, without the need for explicit polling requests. All that makes messages compact, minimizes the overall volume of the transferred data, and reduces the complexity of implementation – all crucial characteristics in constrained environments.

To the best knowledge of the authors, the use of CoAP for device management purposes has not been tried in practice yet, though this is considered in the LightweightM2M standard under development by OMA[4]. Sehgal et al. [16] suggest the suitability of CoAP for accessing on-device configuration data, while also indicating the lack of practical experience. For these considerations, the applicability of CoAP for the purpose of retrieving metering information from constrained devices is investigated in this paper.

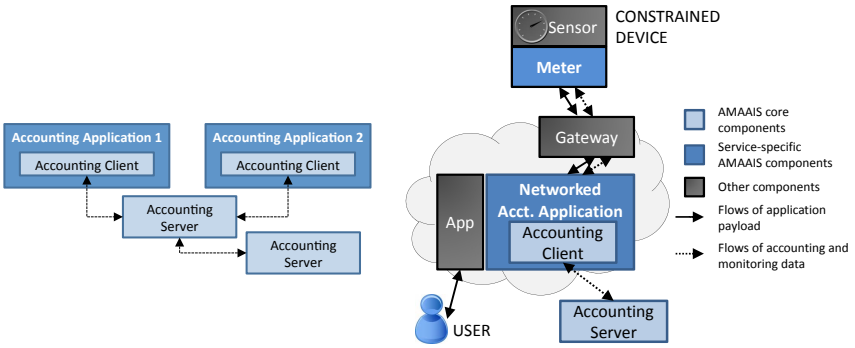
3 Adapting AMAAIS infrastructure to IoT applications

The AMAAIS project aims at supporting the accounting and monitoring of IT services by offering a set of enabling components and facilitating their integration with an authentication and authorization infrastructure [6]. The infrastructure produced by AMAAIS is well suited for federatively aggregating, processing, and storing accounting and monitoring information. This infrastructure, however, has not been designed for constrained devices, and hence needs to be tailored to be suitable in constrained environments. This section overviews the AMAAIS infrastructure and considers the changes that are needed for making it work in applications with Class-1 devices.

The interplay of AMAAIS accounting applications and AMAAIS core components is visualized in Figure 1(a). Any service-specific Accounting Application

[4] `http://member.openmobilealliance.org/ftp/Public_documents/DM/LightweightM2M/`

(AA) is responsible for generating events, *e.g.*, by parsing system log files and extracting any information of interest from them. Once a new event is created, the Accounting Client (AC) API is called. Both AA and AC are running on the same host as a daemon. Once an event generated by an AA is pushed to the AC, it enters the AC pipeline. The event will pass along the AC's pipeline, will be then transmitted to one or multiple Accounting Servers (AS), pass along the respective AS pipeline(s), and will be persisted to database eventually. The communication between AC and AS is based on exchanging Security Assertion Markup Language (SAML) messages. AC and AS pipelines are constituted from so-called Sources and Sinks. Sinks are used to receive and process events, and Sources are components that produce events. AMAAIS provides purpose-specific sub-types of Sinks and Sources.



(a) AMAAIS components (b) Adaptations for constrained devices

Fig. 1. AMAAIS components and adaptations for constrained environments

In principle, it is possible to enable monitoring and accounting functionality on constrained devices by deploying AA and AC components directly on the device. However, for Class-1 devices (which determine the main focus in this paper), this is not feasible due to memory constraints and the lack of Java support, not to mention a large computational and data communication overhead while being engaged in an exchange of SAML messages.

Hence, in order to cope with the limitations of constrained devices, it is reasonable to implement the AA and AC outside of any constrained device, for instance, on a gateway between constrained and unconstrained networks, or on another unconstrained node, as shown in Figure 1(b). This way, the functionality to be kept on constrained device can be limited to a lightweight agent – a *meter* – responsible for executing the metering process and transmitting the metered information to the AA executed on a remote host by using a suitable communication protocol (thus, called Networked AA in Figure 1(b)). CoAP is used as a data transfer protocol for this part of the communication.

4 Empirical evaluation

Prototype implementation. For prototyping purposes, existing AMAAIS components have been extended by a networked AA and an on-device meter, as described below.

The meter software agent has been implemented for TmoteSky, a sensor mote platform featuring an MSP430 16-bit 3.9 MHz CPU, a CC2420 radio chip, as well as 48 kByte of program flash and 10 kByte of RAM. A ready set of software components, including Contiki OS and networking libraries were utilized in the implementation. For evaluation purposes, the meter software has been deployed in the Cooja simulation environment, which emulates TmoteSky motes and enables a simultaneous simulation at network level, operating system level, and machine code instruction set level [18].

As the CoAP engine for Contiki OS, the Erbium CoAP implementation [19] has been used. The meter agent is responsible for gathering and transmitting the following attributes: event identifier, remaining battery level, operational status of the device, available memory, as well as the name of the device and the uptime at which the metering process has been performed. The number of packets sent, received, or dropped has not been metered in the prototype, as these attributes are considered optional for constrained devices. Along with the management attributes listed above, also temperature and light sensor readings are transferred. Integrating such application payload with the accounting and monitoring information within a single message allows data communications overhead to be minimized and hence battery lifetime at a device to be prolonged.

In order to avoid packet fragmentation and reassembly, the size of a CoAP message shall not be longer than 80 Byte [20]. This is also important, since CoAP's "Observe" option assumes short notification messages, fitting into a single packet. Therefore, to keep messages compact and to simplify the implementation, a simple comma-separated value (CSV) format was used for message encoding. While the Efficient XML Interchange (EXI) representation defined by SenML [20] may offer a more compact encoding that allows messages to be compressed to as little as 3% of the original size [17], the CSV encoding was also found sufficiently compact, allowing the message size to be below 80 Byte.

The metering process was implemented to be executed periodically, with a configurable *metering interval*. Both subscribing to periodic metering updates and setting of the metering interval are done in a RESTful manner by issuing GET (with the "Observer" option set) and PUT requests, respectively.

In order to reduce energy consumption by the transmitter, the meter uses the ContikiMAC radio duty cycling (RDC) protocol [21]. Two versions of the meter were implemented:

- With RDC enabled all the time. This version is aimed at application scenarios where a device needs to be constantly available, *e.g.*, for retrieving metered information or for device configuration. For instance, a weather monitoring station may need to be ready to deliver instant measurements of wind strength and direction whenever requested.

– With radio disabled between communication sessions and only being switched on for a relatively short duration of time (10 s) to acquire and transmit metered information or re-configure the device. This version is suitable for scenarios where energy efficiency is of prime concern, and where periods of unavailability can be tolerated.

The networked AA (NAA) has been implemented in Java and integrated with the AMAAIS core components[5]. For communication with constrained devices, the NAA employs the Californium Java framework[6] that implements CoAP communication primitives. Once launched, the NAA connects to each of the constrained devices and subscribes to the metering information updates by issuing GET requests with the "Observe" option.

From that point on, the NAA periodically receives messages with metered information, parses them, and forms accounting events which are then dispatched, through the AC API, to a pre-defined processing pipeline. For prototypical purposes, a simple pipeline was configured consisting of a filtering Sink and two persistence Sinks as well as the respective communication channels between them. The filtering Sink allows individual attributes to be excluded from further processing; for simplicity, the filter in the prototype was configured to let all the event attributes pass. In turn, the persistence Sinks are responsible for storing newly created events in two separate databases.

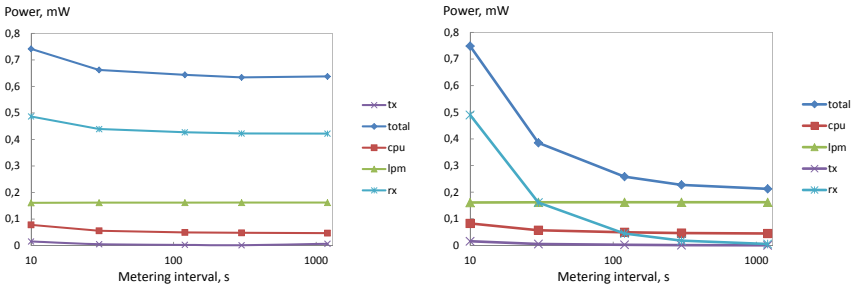
Performance evaluation. As was discussed in Section 2, in applications dealing with constrained devices, the management functionality shall acquire and expose at the very minimum information about device status and about device energy parameters. Further, this functionality should require a minimal state to be maintained on the devices, support lossy and unreliable links, and keep the encoding of management data compact. Finally, authentication, access control, and security bootstrapping mechanisms shall be provided.

These requirements have been taken into account when designing and implementing the prototype. In particular, the prototype enables acquiring and delivering mandatory information, *i.e.*, device status and energy level, as well as the optional information about available memory. By means of CoAP resilience mechanisms, support for lossy and unreliable network connection is provided. The use of CSV as a data encoding format allows the encoding to be compact and hence helps avoid any need for fragmentation and message reassembly. A compact representation, along with the RESTful methods of CoAP, also enable simple and small application level transactions. It shall be noted that security mechanisms were omitted when implementing the prototype; implementing them has been postponed until the CoAP DTLS security specifications are finalized.

The implementation of the meter is quite compact and requires only 1286 Byte of ROM and 158 Byte of RAM. Together with the operating system and other necessary libraries (RPL, uIP, ContikiMAC, Erbium, etc.), the overall application memory footprint is 48638 Byte of ROM and 8732 Byte of RAM.

[5] Available at `http://www.csg.uzh.ch/research/amaais.html`
[6] Available at `http://people.inf.ethz.ch/mkovatsc/californium.php`

Both versions of the meter were experimented with. The power consumption at a constrained device has been estimated with the help of the Contiki OS' Energest module [22], which traces the time a device spends in different modes of operation. Five different metering intervals were used: 10 s, 30 s, 120 s, 300 s, and 1200 s. For each metering interval value, the operation of the meter was simulated over a duration of 50 metering intervals, and the medians of the obtained power consumption estimates have been evaluated, as shown in Fig. 2.



(a) RDC always on (b) RDC on periodically

Fig. 2. Power consumption at a constrained device, both total and according to modes of operation including CPU active (cpu), low power mode (lpm), transmission (tx), and reception (rx): (a) RDC always on; (b) RDC on periodically. Note that a logarithmic scale is used for the metering interval shown on the x-axis.

With the version of the meter with the RDC always on, the radio receiver periodically awakes, by default with a frequency of 8 Hz, to check for radio activity. This is valuable when a device needs to be always available for metering and/or reconfiguration. However, as a result, even for longer metering intervals, the incurred reception-related energy footprint remains significant ($\tau \to \infty \Rightarrow P_{\text{rx}} \to \text{const} > 0$, where τ is the length of the metering interval), as visible in Fig. 2(a).

On the other hand, for the second version of the meter, which disables the radio completely between metering and communications sessions, the reception-related energy drainage is minimized ($\tau \to \infty \Rightarrow P_{\text{rx}} \to 0$, *cf.* Fig. 2(b)). Due to this, the power consumption at the device decreases rapidly as the metering interval grows, at the expense of the device being available only periodically.

Assuming that a constrained device is powered by a pair of AA Zync-carbon batteries, the expected battery lifetime has been estimated and is reported in Table 1. As the table indicates, even with short metering intervals of 10 s, the battery lifetime is approaching half a year. With RDC always enabled, the battery lifetime increases insignificantly with the metering interval, extending the battery lifetime maximum by less than a month. On the other hand, when RDC is enabled periodically, the battery lifetime can be extended to 1.5 years and above, thus, approaching the self-discharge time of a battery.

Table 1. Estimated constrained device's battery lifetime (in days)

Radio mode	Interval, s				
	10	30	120	300	1200
RDC is always enabled	169	189	194	197	196
RDC is on periodically	167	324	484	550	588

It shall be noted that energy consumption at a constrained device is affected by multiple factors, including the overhead of the routing protocol, the type of low power mode used, the time needed for sensors to initialize after being turned on again, etc. Therefore, the absolute numbers of energy consumption estimates reported above shall be used with care, as they are likely to differ to a certain extent depending on case-specific implementation details.

5 Summary and concluding remarks

The management of constrained devices in general, and in particular, the monitoring of their operational status and performance level, as well as the accounting of resource usage of these devices are of great importance for applications in the emerging IoT field. For organizations implementing or deploying IoT applications, the availability of monitoring and accounting information offers numerous benefits, such as better granularity of charging and billing information, support for device status monitoring and troubleshooting, and support for performance optimization and network planning.

This work has studied the applicability of CoAP, a lightweight IETF protocol under development, for acquiring and transmitting management information from the constrained devices. As a part of the study, a meter component responsible for acquiring and transmitting the management information has been prototypically implemented. On the infrastructure side, for storing, processing, and distributing management information retrieved from constrained devices, the meter has been integrated with AMAAIS core components, which provide standard building blocks for the accounting and monitoring of IT services.

Based on results obtained from experiments with the implemented prototype, CoAP has been found suitable for transferring monitoring and accounting information from constrained devices. Its strong sides include the simplicity and compactness of the implementation, as well as a small energy footprint attributed to a small size of messages being transferred and the use of the observer design pattern. The protocol is especially suitable in application scenarios where it is also utilized for exchanging application-specific information, since in this case the protocol implementation code can be reused and hence the size of memory required can be minimized.

The impact of the metering process on device battery lifetime depends on whether a constrained device needs to be constantly available. In case periods of unavailability can be tolerated, estimated battery lifetime is rather long, reaching 1.5 years in the experiments conducted. On the other hand, making a constrained

device available all the time shortens the battery lifetime considerably, with the maximum of circa six months according to estimates obtained in the experiments.

The use of CoAP also imposes some constraints on the management functionality implementation. Specifically, it restricts the volume of information that can be retrieved without making the implementation more complex and/or less energy efficient. This is due to the fact that the use of the "Observe" option of the protocol assumes that information is transmitted within a single CoAP package, which, to avoid packet fragmentation and reassembly, effectively limits the size of a message to circa 80 Byte – leading in the prototype to a reduced number of messages transmitted, implying a positive effect on energy consumption.

The use of the "Observe" option also brings the need to confirm successful message reception, as otherwise, according to CoAP specifications, the subscription to metered information updates is canceled. This implies that a connection loss will nullify a subscription, and its re-initiation shall be implemented by the networked Accounting Application. It should be mentioned that the CoAP standardization process is not completed yet, and therefore some changes may still be introduced, possibly making these side effects less restrictive.

While this work was, thus, able to provide an answer to its research question, it sees several limitations that shall be addressed in future work. In particular, the prototype implementation shall be expanded to allow additional ways of triggering the metering process, include additional attributes to be metered at constrained devices, and allow a reconfiguration of metering process parameters. Once the specification of CoAP DTLS security is finalized, the prototype shall be equipped with the required security mechanisms. Finally, performance characteristics of the prototype shall be empirically evaluated in field experiments.

Acknowledgements

This work has been performed partially in the framework of the AMAAIS project as part of the "AAA/SWITCH e-infrastructure for e-science" programme under the leadership of SWITCH, the Swiss National Research and Education Network, and has been supported by funds from the State Secretariat for Education and Research (SER). The work has been also partially carried out within the framework of the Internet of Things Program of TIVIT Oy nominated to organize and manage the programs of the Strategic Center for Science, Technology and Innovation in the field of ICT funded by the Finnish Funding Agency for Technology and Innovation (TEKES).

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Comput. Netw. **54**(15) (October 2010)
2. Ericsson: More Than 50 Billion Connected Devices. Ericsson White Paper (February 2011)
3. Bormann, C., Ersue, M.: Terminology for Constrained Node Networks. draft-bormann-lwig-terms-00 (November 2012)

4. Kuryla, S., Schönwälder, J.: Evaluation of the resource requirements of snmp agents on constrained devices. In: Proceedings of the 5th international conference on autonomous infrastructure, management, and security. AIMS'11, Berlin, Heidelberg, Springer-Verlag (2011) 100–111
5. Bormann, C., Castellani, A.P., Shelby, Z.: Coap: An application protocol for billions of tiny internet nodes. IEEE Internet Computing **16**(2) (2012) 62–67
6. Stiller, B.: Accounting and monitoring of AAI services. SWITCH Journal **2010**(2) (October 2010) 12–13
7. ITU-T: TMN management functions. ITU-T Recommendation M.3400 (February 2000)
8. Ruiz, L.B., Nogueira, J.M.S., Loureiro, A.A.F.: MANNA: a management architecture for wireless sensor networks. IEEE Communications Magazine **41**(2) (February 2003) 116–125 ISSN 0163-6804.
9. Karsten, M., Schmitt, J., Stiller, B., Wolf, L.: Charging for packet-switched network communication-motivation and overview. Comput. Commun. **23**(3) (February 2000) 290–302
10. Ersue, M., Romascanu, D., Schoenwaelder, J.: Management of Networks with Constrained Devices: Use Cases and Requirements. Internet Draft 02, IETF (October 2012)
11. Ma, Y.W., Chen, J.L., Huang, Y.M., Lee, M.Y.: An efficient management system for wireless sensor networks. Sensors **10**(12) (2010) 11400–11413
12. Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A.: Network Configuration Protocol (NETCONF). RFC 6241 (Proposed Standard) (June 2011)
13. Mukhtar, H., Kim, K.M., Chaudhry, S.A., Akbar, A.H., Kim, K.H., Yoo, S.W.: Lnmp – management architecture for ipv6 based low-power wireless personal area networks (6lowpan). In: NOMS, IEEE (2008) 417–424
14. Choi, H., Kim, N., Cha, H.: 6lowpan-snmp: Simple network management protocol for 6lowpan. In: HPCC, IEEE (2009) 305–313
15. Chaudhry, S.A., Boyle, G., Song, W., Sreenan, C.J.: Emp: A network management protocol for ip-based wireless sensor networks. In: ICWUS, IEEE (2010) 1–6
16. Sehgal, A., Perelman, V., Kuryla, S., Schnwlder, J.: Management of resource constrained devices in the internet of things. IEEE Communications Magazine **50**(12) (December 2012)
17. Shelby, Z.: Embedded web services. Wireless Communications, IEEE **17**(6) (December 2010) 52 –57
18. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with cooja. In: Local Computer Networks, Proceedings 2006 31st IEEE Conference on. (November 2006) 641–648
19. Kovatsch, M., Duquennoy, S., Dunkels, A.: A low-power coap for contiki. In: Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011), Valencia, Spain (October 2011)
20. Jennings, C., Shelby, Z., Arkko, J.: Media Types for Sensor Markup Language (SENML). Internet draft, IETF (October 2012)
21. Dunkels, A.: The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science (December 2011)
22. Dunkels, A., Österlind, F., Tsiftes, N., He, Z.: Software-based sensor node energy estimation. In: Proceedings of the 5th international conference on Embedded networked sensor systems. SenSys '07, New York, NY, USA, ACM (2007) 409–410