Constrained Application Protocol for Low Power Embedded Networks: A Survey

Berta Carballido Villaverde
and Dirk Pesch
Nimbus Centre for Embedded Systems Research
Cork Institute of Technology
Cork, Ireland
Email: {berta.carballido, dirk.pesch}@cit.ie

Rodolfo De Paz Alberola,
Szymon Fedor,
and Menouer Boubekeur
United Technologies Research Centre Ireland
Cork, Ireland
Email: {depazar, fedors, boubekm}@utrc.utc.com

***Abstract*—IPv6 will make it possible to provide Internet connectivity to any device. In the same line, Web technologies will make managing, communicating and visualizing any information provided by these devices attractive to the end users and application developers. Most of the new devices connected to this Web of Things (WoT) will be embedded and wirelessly connected. However, current Web technologies, developed with powerful devices in mind, will not be suited for this kind of environment. In order to make the WoT a reality for low power embedded networks, specialised protocols that consider the energy, memory and processing constraints of these devices must be designed. The IETF recently created the CoRE group whose first goal has been developing a RESTful application layer protocol for communications within embedded wireless networks referred to as Constrained Application Protocol (CoAP). The year 2011 has seen a big push with regards to research in this area, indicating a growing interest in the community towards RESTful interactions in low power wireless embedded networks. This paper surveys current research efforts on the Constrained Application Protocol for low power embedded networks.**

I. Introduction

In the foreseeable future every object will have a unique way of identification, thanks to the vast address space offered by IPv6, which will allow interconnecting and addressing every smart object on an individual manner. The Internet will become the Internet of Things (IoT) and, in this scenario, interactions among smart objects, machines and users will be common. This will require universal application layer protocols and user friendly technologies in order to be a success.

Web services have demonstrated to be essential in enabling interoperable communications between computers on the traditional Internet. Web services can be realized using two different approaches: a RESTful approach, where for instance resources may be managed using HTTP interactions; or a remote procedure call (RPC) type approach using, for instance, the Simple Object Access Protocol (SOAP). Nevertheless, standard Web service technologies such as XML, SOAP, and HTTP are too costly in terms of complexity and overhead to be used in constrained environments. However, resource constrained embedded devices hosting machine-to-machine applications such as energy monitoring, building automation, security systems, etc. will be common in the IoT.

In this line, the RESTful paradigm has many advantages over RPC style interactions for low power embedded networks as it introduces less overhead, less parsing complexity, statelessness, and allows tighter integration with HTTP [1]. However, although RESTful web services are simple in concept, the protocols and payload formats used to realize them are still not completely suitable for embedded wireless networks and devices. Problems with these protocols include: (i) HTTP headers add too much overhead; (ii) TCP has performance problems over lossy links and no multicast support (iii) Pull model is not inappropriate for sleeping sensors; (iv) XML has a very high parsing complexity if used as payload.

Therefore, in order to fully realize the embedded Web services vision, the RESTful Web service paradigm needs to be extended into the constrained domain. To do this, the IETF Constrained RESTful Environment (CoRE) group is developing an application protocol for embedded web services referred to as Constrained Application Protocol (CoAP) [2]. CoAP has been designed to offer simplicity, low overhead and machine-to-machine communications that is required to enable the interaction and management of embedded devices. CoAP is a platform independent approach that facilitates the integration of the embedded network with existing Web technologies which makes it highly attractive for users and developers. The year 2011 has seen a big push with regards to research in this area, indicating a growing interest in the community towards RESTful interactions in low power wireless embedded networks. Moreover, some companies are already offering products with the CoAP/IPv6 framework [3], [4].

This paper aims to evaluate ongoing research efforts in the CoAP area in an attempt to provide a clear idea on where CoAP research stands.

The rest of the paper is organised as follows: Section II provides a general overview of the Constrained Application Protocol; Section III analyses ongoing research efforts in the CoAP field; Section IV details available implementations of CoAP for different Operating Systems and Languages. Section V analyses where current research stands and which should be the future focus of research. Finally, Section VI concludes this paper.

II. CoAP Overview

The Constrained Application Protocol (CoAP) [2] is a RESTful Web transfer protocol for use with constrained networks currently being developed by the IETF CoRE working group. CoAP uses an interaction model similar to the client/server model of HTTP. Nevertheless, machine-to-machine interactions typically result in a CoAP implementation acting in both server and client roles (referred to as end-points). CoAP is being designed for constrained environments and therefore it introduces low header overhead and parsing complexity. Other main features of CoAP include:

- User Datagram Protocol (UDP) binding to avoid costly TCP handshakes.
- Four request methods similar to those of HTTP: GET, POST, PUT and DELETE. And three types of response codes: 2.xx (success), 4.xx (client error), 5.xx (server error).
- Four different message types: Confirmable, Non Confirmable, Acknowledgement and Reset (i.e. Nack). Confirmable and Acknowledgement/Reset messages are utilised to provide reliable communications when required (note that CoAP runs on top of UDP). Non Confirmable messages do not require acknowledgements.
- Unicast and multicast requests, where multicast requests can be useful to query several similar devices simultaneously (i.e. Group communications [5]). Moreover, CoAP includes resource observation [6] (i.e. publish/subscribe) where a client can subscribe to be updated when a server resource changes to avoid requesting constantly for the state of that resource.
- Resource discovery capability [7] to allow clients discover the resources hosted by servers. And subnet discovery capability [8] to allow subnetwork devices, using different link layer technologies, to discover each other. Hosted resources can also be posted to Resource Directory (RD) [9] entities that would contain descriptions of resources held by several other servers.
- Large file transfer support (e.g. firmware updates) referred to as block-wise transfers [10].
- Simple proxy and caching capabilities (catching can be beneficial in low power networks to store information for/of sleeping devices) as well as HTTP mapping for integration with existing networks and Web technologies.
- URI based resource representations (i.e. coap://server/lightinfo) and support for different payload content types.
- Security binding to Datagram Transport Layer Security (DTLS).

Figure 1 shows an example of a CoAP client/server interaction where a client requests a temperature reading from a server (in this case a sensor). In the example, since the client uses a Confirmable (CON) message to perform the request, the response is piggy-backed in the Acknowledgement message. The client uses the GET method to request the status of the temperature resource in the server. Two types of response codes are used, one for the successful interaction (2.05) and another for the failure (4.04).

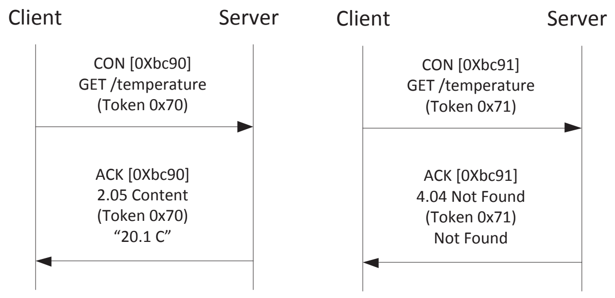

Fig. 1. Two GET requests with piggy-backed responses, one successful, one not found [2]

III. State of the Art

Up to this point, research on the Constrained Application Protocol has mainly focused on performance evaluations, user-network interaction and network autoconfiguration. Those research efforts are reviewed next.

A. *Performance Comparisons & Evaluation*

The authors in [11] provide an implementation of CoAP over 6lowPAN and the low power duty-cycling MAC referred to as ContikiMAC for the Contiki embedded OS. Block-wise transfers [10], resource observation [6] and resource discovery [7] features are included in this implementation. These features as well as their interaction with the low power MAC are evaluated in terms energy consumption and latencies. Overall, this work demonstrates that power-efficient CoAP operation can be achieved just at the radio duty cycling level without the need for specialized power-management at the application layer. Note that this contradicts some of the conclusions of [12] where the authors see the need for developing specialized communication mechanisms for CoAP in low-power environments. This is due to the fact that the authors in [12] do not consider that packets can be buffered, thus a request that can not be immediately received and processed by a server is considered lost.

Work in [13] compares CoAP performance with regards to HTTP and evaluates the suitability of CoAP for resource constrained scenarios. As expected, CoAP introduces lower response time delays and much lower overhead than HTTP due to its compressed header and the fact that it is based on UDP which avoids TCP hand-shaking mechanisms. Moreover, work in [14] describes the implementation of the libcoap CoAP library for Contiki and TinyOS embedded operating systems and analyses the suitability of CoAP for transport logistic applications. Results show again that CoAP is more suitable for this kind of application, and the involved constrained devices, than other RESTful protocols such as HTTP due to the lower overhead introduced while still maintaining reliability. The lower energy consumption shown by CoAP when compared to HTTP is as well demonstrated in [15].

On the other hand, researchers in [16] provide an implementation of CoAP and EXI data format [17] on TinyOS embedded operating system. The authors examine the successful handling of requests made to servers with regards to number or requests and serving nodes. As expected, the success probability decreases with the number of requests performed on the servers (success is above 95% up to rates of 20 requests per second and above 90% up to 50 requests per second). Results also show that using EXI compression for the payload is up to 50 times more efficient than plain XML in terms of payload size, which shows that EXI can be a good candidate for compressing payload generated (and transmitted) by constrained devices.

Finally, researchers in [18], [19] discuss the need and demonstrate the benefits of a SOAP-over-CoAP binding, as described in [20], which provides the possibility to integrate SOAP based Web services in 6LoWPAN based WSNs. A comparison among SOAP-over-HTTP, SOAP-over-UDP, SOAP-over-CoAP is carried out which demonstrates the higher suitability of SOAP-over-CoAP binding for constrained scenarios - with lower introduced delays and overhead than HTTP binding and higher reliabilities than a simple UDP binding. Finally note that CoAP here is used as a transport mechanism for SOAP envelopes and does not intent to exploit all the features offered by CoAP (i.e. multicast, publish/subscribe). Moreover, note as well that the current CoAP specification, as it is, does not support SOAP binding [20] (i.e. it needs to be modified).

B. User Interaction

On another note, it is the view of the researchers in [21] that the realisation of the Internet of Things (i.e. the connection of every single device to the Internet) currently faces two challenges: achieving a scalable application layer for interoperability and a common programming model for application developers. To face these challenges, the authors propose using a RESTful architecture, with the Constrained Application Protocol (CoAP) playing a key role, where applications can be developed outside the embedded domain and run in the cloud - which keeps devices simple and low-cost. This can be achieved by simply using the interactions provided by CoAP to monitor or subscribe to resources or control devices. The same researchers in [22] develop Copper, a generic browser for networked embedded devices running CoAP. The objective of this Firefox based browser is to allow users to observe and control devices, through CoAP interactions, by adopting well-known patterns from the Web, such as browsing, bookmarking, and linking. Moreover, the same researchers in [23] show how RESTful interactions may be used to create network $mashups$ and how they may be used for BMS applications such as smart metering.

On the other hand, researchers in [24] evaluate the use IETF RPL routing protocol [25] and CoAP application protocol to allow the user configure a wireless sensor network. More specifically, they focus on the wireless transmission and installation of software at deployment time using the aforementioned protocols. With regards to CoAP performance for this scenario, since it handles end-to-end transmissions, all the performed simulations end with a successful data download and installation. This work shows that the catching feature of CoAP is beneficial for software updates as it allows storing software at intermediary nodes. As future work, the authors are building a deployment tool for IP-based sensor networks which will be used to configure and reprogram individual nodes based on their IP addresses. This tool has already been showcased in [26] but it is not available for download yet. Although the authors in this demo propose a way to identify the device with QR codes and location, no mechanism is provided to map this to the specific IP address of the device. This means that for instance a user can not download code to a specific node (IP address) located in a specific location unless some other mechanism is employed to link that information (i.e. IP address and Location). At the moment, nodes may request code after being prompted by the user whether they have some specific code available.

Work in [27] also discusses the integration of WSNs with the Web. The authors design a Web platform for WSN monitoring which integrates a REST/CoAP WSN with a REST/HTTP Web application, thus implementing a CoAP/HTTP proxy. The authors show that with this proxy implementation it is possible to visualise CoAP based WSN measurements on a HTTP based web browser.

Moreover, researchers in [28] discuss the necessity of identifying ”things” (i.e. belonging to the Internet of Things) by their characteristics such as name, capability, etc. As having only a network address (i.e. IP/MAC address) does not provide any information of an object - note that this was a limitation identified in [12]. With this purpose, the authors propose the use of Electronic Product Codes (EPC) based URIs to determine the nature of a ”thing”. The EPC number is used in the CoAP URI to identify the object, which in this case are home appliances. Using CoAP based RESTful Web services, the authors display information of the monitored appliances, that can be now identified using the EPC based URI, on a Twitter account and iPad application to give information to the user on their energy consumption, thus showing that CoAP can be elegantly combined with existing popular Web applications.

C. Autoconfiguration

Researchers in [29] deal with the necessity of automatically discovering network devices and their offered services so that requirements for network configuration are minimal. With this in mind, the authors propose new CoAP methods so that the Universal Plug and Play (UPnP) standard can be utilised to discover services offered by a constrained network by means of a CoAP/UPnP bridge residing on the constrained network gateway (i.e. the same gateway that would perform the CoAP/HTTP bridging). In the proposed scenario, UPnP over HTTP commands are translated to the extended CoAP methods at the bridge and vice versa. Moreover, the authors analyse which working conditions are optimal to minimise chances of overloading the network with discovery based traffic. In this line of research, IETF CoRE work in progress proposes in [8] a discovery service for CoAP networks with the objective of allowing devices to discover other subnets and devices (CoAP servers and gateways) associated to the subnets (note that subnets can be networks using different technologies at the link layer such as WiFi or Zigbee). Although this draft proposes a method to discover devices, it does not allow at this point discovering the resources offered by those devices (i.e. a server discovery only returns the server addresses but not the hosted resources). Since just the device address does not provide any useful information to the application, discovering resources would at this moment require an additional mechanism such as asking for hosted resources to every server that has been discovered -which would introduce more overhead.

Moreover, the authors of the draft in [30] introduce the concept of service discovery based on DNS Service Discovery (DNS-SD). Designed with scalability in mind, service discovery here is concerned with finding the IP address, port, protocol, and path of a named service hosted by a CoAP server. This differs from resource discovery which is a fine-grained enumeration of server resources as described in [7]. With the proposed service discovery approach, the resolution service/host could be performed automatically by means of a DNS-SD server that would store the service/host relationships. Some possible examples for a building management case are provided as well in [30] on how to manually create the service/host relationship that would be stored on the DNS-SD server to later allow service discovery. Note that this is related to the resource directory concept proposed in [9] where servers may register their hosted resources on a resource directory server which would allow servers to perform resource lookups.

D. Applicability: Building Management & Smart Grid

CoAP design rests on the requirements identified by 6lowApp in [31], [32] for smart grid and building automation applications. Thus, developments have been made recently focusing on the use of CoAP for such environments. In this line, the authors of the draft in [30] are proposing several mechanisms to merge popular legacy application layer protocols of the building management field such as BACnet, LON, etc. with CoAP. For such cases, the authors in [30] propose that legacy protocol resources (i.e. BACnet objects) would be mapped into CoAP resources (i.e. URI paths) and legacy protocol communication data would be transported as payload in CoAP messages. It is the view of the authors in [30] that this merger would allow progressing from the current systems to all-IP systems. Also, for the building management case, the authors of the draft in [5] propose as well the integration of group communications by means of CoAP multicasts. Group communications play a key role in building management applications as they allow a single device to communicate with several others sharing similar characteristics (i.e. every temperature sensor, every device in a room, etc.). Moreover, with regards to smart grid applications, it is worth mentioning that an initial agreement has been made by the ZigBee Smart Energy Working Group for the use of HTTP and CoAP for Smart Energy version 2 standard.

E. Limitations

Researchers in [33] identify two security problems that still need to be solved by CoAP developers. The first one is achieving DTLS/TLS translation when CoAP/HTTP mapping is used at a proxy so that this kind of mapping can provide an end-to-end secure connection. The second would be allowing secure multicast communications which currently are not supported by DTLS. Future work of [33] considers analysing how to solve these security limitations.

Moreover, the Internet draft in [12] analyses the suitability of CoAP for a monitoring application. This work finds several limitations that should be addressed for future releases of CoAP such as the need for standardising semantics for interoperability purposes or the need to allow catching requests and not just responses.

F. Summary

In summary, the following findings can be outlined:

- Low power operation: CoAP, as it introduces lower overhead, has been shown to be more energy efficient than HTTP and thus more suited for embedded wireless networks. Moreover, it has also been shown that low power operation can be achieved at the lower layers independently of CoAP (i.e. duty cycling layer, etc.). Finally, EXI compression combined with CoAP has been shown to be a much more efficient approach in terms of energy than the XML/CoAP combination.
- Delay: CoAP can satisfy tighter delay requirements than HTTP due to the fact that it rests on top of UDP.
- Reliability: reliable communications can be achieved thanks to the additional end-to-end reliability mechanisms implemented in CoAP to complement UDP communications. The successful handling of requests at the servers will inevitably depend on the number of users and requests.
- Bindings: SoAP over CoAP binding has been proposed as another way to offer Web services in low power wireless embedded networks. It is yet to be seen whether this binding will be required. Note that the current CoAP specification needs to be modified to support it.
- Ease of use: the possibilities of integrating CoAP into Web applications such as Twitter or Web browsers such as Firefox makes it very attractive for users and developers.
- Reduced configuration efforts: IPv6 already offers autoconfiguration capabilities. Moreover, CoAP developers are proposing service, resource and subnet discovery mechanisms. Adaptation with UPnP has also been proposed by researchers.
- Limitations: since CoAP is work in progress, there still exist several limitations that must be addressed, some of which have been investigated already by the work reviewed here.

IV. Implementations

There are several CoAP implementations and applications that may be used for testing purposes. Those are detailed next:

- Libcoap: Libcoap is a C# implementation of CoAP. The latest Libcoap release implements the draft-ietf-core-coap-03 version and includes support for resource observation and block-wise transfers. It is publicly available in [34] and has been ported to TinyOS [35]. According to [35], the TinyOS version does not implement POST and DELETE methods, well-known/core, block-wise transfers or observer model at the moment.
- Contiki OS CoAP: Contiki OS also provides its public implementation of CoAP [36]. This implementation supports all CoAP methods (PUT, POST, DELETE, GET) and includes features such as well-known/core, block-wise transfers or observer model. The latest non-official CoAP developments for Contiki can also be obtained in [37]
- jCoap: jCoAP is a CoAP Java implementation compatible to Java SE and Android. jCoAP is an early-stage project from the University of Rostock and can be downloaded freely in [38].
- CoAPy: CoAPy is an early-stage Python implementation of CoAP, intended to allow Python clients and servers. It can be obtained in [39].
- Copper: Copper is a Firefox based web browser that allows performing CoAP based interactions with constrained devices. It is publicly available in [40].
- Californium: Californium (Cf) is a very modular CoAP framework written in Java using a flexible, layered architecture. It allows easy creation of cloud-based client and server applications as well as CoAP proxies. It can be obtained in [41].
- Nodes: Nodes is an Android based HTTP-CoAP bridge that allows browsing and communicating with local CoAP devices using IPv6 addressing. It can be obtained free of charge in the Android Market.
- Erika OS CoAP: Erika OS developers are working at the moment on implementing CoAP [42].

Note that since CoAP work is in progress, necessarily these implementations are in progress as well.

V. Analysis

Most of the work up to now has focused on the implementation of CoAP over several operating systems as well as comparing it to HTTP in terms of energy and latencies.

Little has been done however on evaluating CoAP for different machine-to-machine applications. Nevertheless, analysing the suitability of CoAP for different applications may expose new requirements. For instance, let us consider an alarm application that performs a POST whenever out-of-bounds conditions are detected. This kind of application may require some degree of quality of service (i.e. maximum allowed latency). However, no mechanisms are available at this point to assure that the application data will be served with the required Quality of Service. Note for instance that in SOAP based Web services the quality of service requirements to transmit a SOAP envelope (i.e. use DiffServ, RSVP, etc.) can be embedded in the SOAP header [43]. Thus, it is the authors' view that CoAP should not only be evaluated in terms of energy, overhead, etc. but it should also be evaluated in terms of suitability for the proposed scenarios or applications. This may uncover possible limitations or additional requirements.

Even though CoAP work is still in progress, there are already signs of an architectural discussion in the research community: should CoAP be used as transport mechanism for SOAP based Web services or should it be used to allow machine-to-machine or human-to-machine RESTful interactions? Although CoAP can also be used to support SOAP-based architectures, the initial intention of CoAP development (and its offered features such as multicasting or publish/subscribe methods) has been the realisation of a RESTful architecture in constrained networks. However, since compression mechanisms such as EXI make it possible to have small payloads suited to constrained networks, it is probably too early to predict whether SOAP will be finally discarded for these wireless embedded networks or if both will coexist depending on the scenario or application. For instance, when requiring simplicity and ease of adoption a RESTful design may be more appropriate [44]. On the other hand, a SOAP-based architecture may be better suited to offer more advanced requirements such as quality of service [44]. Nevertheless, at this point in time and as seen in the previous state-of-the-art review, a RESTful-based architecture seems to have gained more support among constrained wireless network researchers due to its lightweight character, simplicity and the features offered by CoAP to interact with resources (note that a SoAP-based architecture only makes use of the POST method to transport the envelopes). Perhaps the suitability of each scheme will become clearer when both architectures are more developed and evaluated for different possible applications.

Finally, since CoAP is still work in progress there are some limitations of the current release that have to be addressed such as the security problems outlined in [33], the modifications proposed in [20] to allow the SOAP-over-CoAP transport or the drawbacks identified by the draft in [12] with regards to allowing catching of requests, including support for more data formats or defining common semantics for interoperability.

VI. Conclusion

This work has surveyed and analysed research on the Constrained Application Protocol (CoAP). CoAP is being developed by the IETF CoRE working group to allow RESTful interactions in low power embedded networks. Current research shows that CoAP is better suited for low power networks than other RESTful protocols such as HTTP as it introduces lower overhead and delays while maintaining reliability. Moreover, existing research has also shown that CoAP can be easily integrated with existing Web technologies. Nevertheless, the current specification still faces many challenges ranging from lack of universal semantics to some security limitations. Moreover, little research has been done on putting CoAP under test for different machine-to-machine applications. CoAP is currently attracting the attention from the embedded research community and even some companies are releasing some CoAP based products [3], [4], indicating that the RESTful trend for low power embedded networks will keep on gaining interest.

References

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

[2] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained application protocol (coap)," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft.*, [Expires May 4, 2012].

[3] "Watteco restful devices http://www.watteco.com/," [last accessed January 2012].

[4] "Sensinode nanoservice http://www.sensinode.com," [last accessed January 2012].

[5] A. Rahman and E. Dijk, "Group communication for coap," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft*, [Expires July 13, 2012].

[6] K. Hartke and Z. Shelby, "Observing resources in coap," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft.*, [Expires May 3, 2012].

[7] Z. Shelby, "Core link format," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft*, [Expires January 26, 2012].

[8] A. Brandt, "Discovery of coap servers across subnets," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft.*, [Expires September, 2011].

[9] Z. Shelby and S. Krco, "Core resource directory," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft*, [Expires May 4, 2012].

[10] C. Bormann and Z. Shelby, "Blockwise transfers in coap," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft*, [Expires January 12, 2012].

[11] 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*, oct. 2011, pp. 855 –860.

[12] J. Arkko, H. Rissanen, S. L. Z. Turanyi, and O. Novo, "Implementing tiny coap sensors," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft*, [Expires Jan 6, 2012].

[13] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "Evaluation of constrained application protocol for wireless sensor networks," in *Local Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*, oct. 2011, pp. 1 –6.

[14] K. Kuladinithi, O. Bergmann, T. Ptsch, M. Becker, and C. Gorg, "Implementation of coap and its application in transport logistics," *In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*, 2011.

[15] W. Colitti, K. Steenhaut, and N. D. Caro, "Integrating wireless sensor networks with web applications," *In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*, 2011.

[16] 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.

[17] "Efficient xml interchange (exi) format 1.0 http://www.w3.org/tr/exi/," [last accessed January 2012].

[18] G. Moritz, F. Golatowski, and D. Timmermann, "A coap based soap transport binding," in *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, sept. 2011, pp. 1 –4.

[19] ——, "A lightweight soap over coap transport binding for resource constraint networks," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, oct. 2011, pp. 861 –866.

[20] G. Moritz, "Soap-over-coap binding," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft*, [Expires December 19, 2011].

[21] M. Kovatsch, "Firm firmware and apps for the internet of things," in *Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications*, ser. SESENA '11. New York, NY, USA: ACM, 2011, pp. 61–62. [Online]. Available: http://doi.acm.org/10.1145/1988051.1988064

[22] ——, "Demo abstract: Human-coap interaction with copper," in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, june 2011, pp. 1 –2.

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

[24] S. Duquennoy, N. Wirstom, N. Tsiftes, and A. Dunkels., "Leveraging ip for sensor network deployment," *In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*,, April 2011.

[25] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "Rpl: Ipv6 routing protocol for low power and lossy networks," *Internet Engineering Task Force, Routing over Low power Lossy networks Group, Internet Draft*, [Expires September 14, 2011].

[26] S. Duquennoy, N. Wirstrm, and A. Dunkels, "Demo: Snap rapid sensornet deployment with a sensornet appstore," *International Conference on Embedded Networked Sensor Systems (ACM SenSys 2011)*, 2011.

[27] W. Colitti, K. Steenhaut, N. De Caro, B. Buta, and V. Dobrota, "Rest enabled wireless sensor networks for seamless integration with web applications," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, oct. 2011, pp. 867 –872.

[28] H. Hada and J. Mitsugi, "Epc based internet of things architecture," in *RFID-Technologies and Applications (RFID-TA), 2011 IEEE International Conference on*, sept. 2011, pp. 527 –532.

[29] J. Mitsugi, S. Yonemura, H. Hada, and T. Inaba, "Bridging upnp and zigbee with coap: protocol and its performance evaluation," in *Proceedings of the workshop on Internet of Things and Service Platforms*, ser. IoTSP '11. New York, NY, USA: ACM, 2011, pp. 1:1–1:8. [Online]. Available: http://doi.acm.org/10.1145/2079353.2079354

[30] P. van der Stok and K. Lynn, "Coap utilization for building control," *Internet Engineering Task Force, Constrained RESTful Environments Group, Internet Draft*, [Expires May 2, 2012].

[31] D. Sturek, Z. Shelby, D. Lohman, M. G. Stuber, and S. Ashton, "Smart energy requiements for 6lowapp," *Internet Engineering Task Force, 6lowapp, Internet Draft*, [Expires April 22, 2010].

[32] J. Martocci, A. Schoofs, and P. van der Stok, "Commercial building applications requirements," *Internet Engineering Task Force, Network Working Group, Internet Draft*, [Expires January 8, 2011].

[33] M. Brachmann, O. Garcia-Morchon, and M. Kirsche, "Security for practical coap applications: Issues and solution approaches," *GI/ITG KuVS Fachgesprch Sensornetze (FGSN)*, 2011.

[34] "Libcoap implementation http://sourceforge.net/projects/libcoap/," [last accessed Jan 2012].

[35] "Tinyos coap implementation http://docs.tinyos.net/tinywiki/index.php /coap," [last accessed Jan 2012].

[36] "Contiki coap implementation https://github.com/mkovatsc /smartapp-contiki," [last accessed Jan 2012].

[37] "Smartappcontiki https://github.com/mkovatsc/smartappcontiki," [last accessed January 2012].

[38] "jcoap implementation http://code.google.com/p/jcoap/," [last accessed Jan 2012].

[39] "Coapy implementation http://coapy.sourceforge.net/," [last accessed Jan 2012].

[40] "Copper coap browser https://addons.mozilla.org/en-us/firefox/addon/copper-270430/," [last accessed Jan 2012].

[41] "Californium coap framework http://people.inf.ethz.ch/mkovatsc/ californium.php," [last accessed Jan 2012].

[42] "Erika os coap implementation http://rtn.sssup.it/index.php/software/11-research-activities/35," [last accessed Jan 2012].

[43] "Soap version 1.2 usage scenarios http://www.w3.org/tr/xmlp-scenarios/," [last accessed Jan 2012].

[44] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*, ser. WWW '08. New York, NY, USA: ACM, 2008, pp. 805–814. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367606