A Real-Time Middleware Platform for the Smart Grid

Tatjana Predojev, *Student Member, IEEE,* Adel Al-Hezmi, *Member, IEEE,* Jesus Alonso-Zarate, *Senior Member, IEEE* and Mischa Dohler, *Fellow, IEEE*

***Abstract*—The currently used communications layer in the electric power grid is heavily outdated and thus not able to cope with emerging Smart Grid application requirements. Notably, new measurement devices and advanced control applications are being deployed with the aim to greatly improve the stability and efficiency of the grid. A communication infrastructure together with a middleware that delivers updates across wide areas in a timely and reliable manner is thus a cornerstone in the emerging Smart Grid. In this paper, we analyse distributed computing technologies that meet the stringent communications requirements. We show that the solutions and practices developed for Machine-To-Machine (M2M) communications, ETSI M2M middleware in particular, can be successfully applied to future Smart Grid networks. The pitfalls of the proposed architecture are identified and an upgrade is devised to provide for the critical event-driven, real-time communication required by some applications.**

I. Introduction

THE large scale blackouts in the previous decades were the motivation for the deployment of Supervisory Control and Data Acquisition (SCADA) systems, whose purpose is to monitor the state of the power grid. These systems, based on the decades-old technology, have changed little over the years. With SCADA, a substation collects local measurements and delivers them to the application in the control centre, managed by human operators. Communication links between the individual substations and the control centre form a star topology, and the collected data is stored in a centralised database. The data rates on links are low, up to a few kilobits per second, and the latency is in the range of several seconds or more. As a consequence, SCADA communication system limits the protection and control operations that the modern power applications and devices are capable of achieving [1].

One of these modern devices is the Phasor Measurement Unit (PMU). PMUs can measure bus voltages, currents, frequency etc. at the rates of several times per power cycle. They synchronise with the Global Positioning System (GPS) clocks in order to timestamp their measurements. PMUs deployed over a wide area provide a global time-synchronised snapshot of the power grid state, which contributes to the real-time situational awareness that cannot be achieved with SCADA. With this information, the scope of control applications widens, compared to the local control performed today within a substation. This in turn allows for global protective actions to be performed even before the power grid becomes unstable. However, in order to benefit from a wide area measurement system, an enhanced communication system is needed to deliver the measurements to the networked control applications [2]. The system consists of two major parts: communication infrastructure and middleware [3]. The currently developed technical solutions for such middleware, as well as their shortcomings, are addressed in this paper.

A common, grid-wide, universal middleware for distributed networked operations facilitates real-time interactions between the devices and applications. The shared middleware also simplifies application development on a variety of platforms, operating systems, networking technologies etc. One notable example is GridStat [4], a publish-subscribe middleware implemented with the Common Object Request Broker Architecture (CORBA). A competitor to CORBA for implementing a distributed system, one that is becoming increasingly predominant, is the web service technology. RESTful web services are particularly favoured over other solutions, primarily because of their simplicity while providing the full functionality that other solutions can offer. We make a case for the web services to constitute the basis of the Smart Grid communication framework, and then we analyse ETSI M2M web service middleware [5] in the context of the Smart Grid. ETSI M2M document on the Smart Grid [6] outlines some adequate use cases and their requirements, without providing the technical solutions to realise them. Therefore, we first devise a working architecture that relies on the ETSI M2M components mapped to the Smart Grid. Then we analyse the application layer protocols used in web services while focusing on real-time performance.

A middleware proposed in this paper addresses one major disadvantage of the request-response (client-server) web model using the Hyper Text Transfer Protocol (HTTP): the lack of support for real-time communications. For instance, delivering a stream of PMU updates to the client application is cumbersome and does not cater for the strict latency requirement. Although various workarounds have been devised, e.g. [7], a client-server web model was not initially designed to support real-time events. We address this problem with two upgrades and specify when to use each of them. To the best of our knowledge, this is the first attempt of considering the entire set of Smart Grid requirements, including real-time communications, with one ubiquitous communication middleware based on web service technology.

In Section II, we briefly summarise the essential communi-

T. Predojev and J. Alonso are with the CTTC, 08860 Barcelona, Spain, A. Al-Hezmi is with the Fraunhofer FOKUS, 10589 Berlin, Germany, M. Dohler is with the King's College, WC2R 2LS London, UK, e-mail: {tpredojev, jalonso}@cttc.es, adel.al-hezmi@fokus.fraunhofer.de, mischa.dohler@kcl.ac.uk

cation requirements for the Smart Grid and define the traffic patterns of example grid applications. In Section III, we adapt the ETSI M2M architecture to the Smart Grid scenario, identifying key components, their location and interconnections. Finally, the upgrade of the middleware to address the real-time communication requirement, is presented in Section IV.

II. Smart Grid Communication Requirements

Many recent studies, e.g. [4], [8], [9] identify three principal Smart Grid communication requirements:

- Quality of Service (QoS)
- Flexibility
- Security

A. Quality of Service

Regarding QoS, a major constraint is put on data *latency*. Four categories with different latency requirements are identified depending on the network application: protection, control, monitoring or reporting. Some concrete values for the maximum response times are [6]:

- protection $1 - 10\ ms$ (real-time)
- control $100\ ms$ (real-time)
- monitoring $1\ s$ (near real-time)
- reporting, billing, post-incidental analysis $>\ 1\ h$ (slow)

In addition to the latency constraints, a certain data availability that is guaranteed by the network is expected, e.g. $> 99.99\%$ for critical data updates [4].

B. Flexibility

A communication system for Smart Grids needs to support a variety of data update rates, latencies, heterogeneous underlying networks, operating systems etc. In addition, [4] stresses the need for an *open* interface, one that is easily extendable and allows for application interoperability across multiple vendors. In this respect, web services are more flexible than the CORBA middleware.

Flexibility also applies to the middleware design: data updates should be easily available to any legitimate participant at any location, at a predefined rate, which cannot be easily achieved with SCADA. Not all updates that a device is generating, but only the requested ones, need to go to the right application at the right rate and latency. Therefore, a certain *filtering* functionality is needed.

C. Security

A critical infrastructure such as the power grid has to be highly secure. Security covers the authentication schemes in order to deny access to unauthorised parties, and the proven techniques for data confidentiality and data integrity. In addition, the privacy concern should be addressed to prevent any possible misuses.

Table I
Communication requirements for three example Smart Grid applications

Application (subscriber)	Oscillation Monitoring	State Estimation	Demand Response
purpose	control	monitoring	peak shaving
latency	$< 100\ ms$	$< 1\ s$	seconds to minutes
category	real-time	near real-time	slow
device (publisher)	PMU directly	substation or PMU	gateway for smart appliances
rate	30 updates/s	5 updates/s	4 updates/min
scope	selected PMUs	all PMUs	all gateways
# publishers	< 100	hundreds	millions
location	transmission	distribution	consumer

D. Power Grid Data Traffic Patterns

We have analysed data requirements of three example Smart Grid applications in order to deduce a traffic pattern of the power grid measurement updates. Devices, e.g. PMUs or smart meters and appliances, take measurements and publish data for the interested applications to subscribe to. The special (and very different) requirements of each application are given in Table I. We assume that the Oscillation Monitoring application will be used for control, which explains the strict latency constraint. This application monitors oscillations, primarily within the transmission grid, that can bring the power grid to an unstable state. State Estimation is increasingly calculated for the distribution grid to supplement the one in the transmission grid, so that the complete overview of the power grid state is available. In this case, requirements are somewhat loosened compared to the Oscillation Monitoring, but the number of devices is higher. Finally, we consider Demand Response application that processes consumption data coming from smart appliances. In this case, neither the latency constraint, nor the update rate is critical compared to the other two applications.

III. M2M Communications in Smart Grid

A wide area real-time situational awareness is impossible to achieve and coordinate if it depends on a number of human operators. One of the visions of the Smart Grid concept is to minimise the human involvement, or even to exclude the human from the control loop entirely. Towards this end, the Smart Grid can capitalise on the technologies for M2M communication systems that are being developed and deployed [8]. To enable distributed computing between remote applications and devices, a middleware platform is needed. The prevailing M2M middleware will likely be implemented with the web service technology to provide the required interoperability and scalability.

A. RESTful Web Services

Representational State Transfer (REST) offers software architecture guidelines for implementing a web service [10]. A web service is hosted on a server to be accessed by

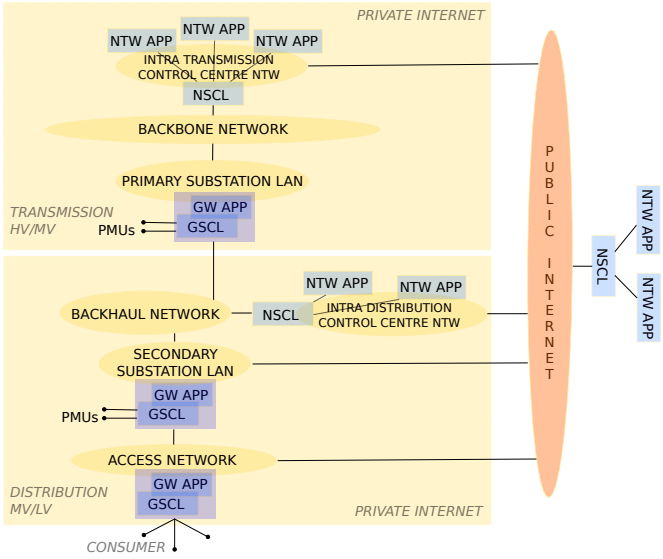

Figure 1. A mapping of ETSI M2M components to the Smart Grid

clients. Instead of exposing the methods, a RESTful web service exposes internal *data*, organised into data units called resources. Resources define the functionality of the service exclusively through the specific data that they expose. Resources are accessed and modified using the methods of the web application layer protocol, in most cases this is HTTP (or Constrained Application Protocol (CoAP) for resource-constrained devices). HTTP is an application layer protocol that operates over a TCP/IP connection. A client accesses the service by sending a HTTP request to the specific resource on the server and the result is delivered in a HTTP response. The communication is always initiated by the client and realised by the exchange of HTTP messages that contain a representation of the addressed resource, which is a serializable document that holds info on the resource's state. Every HTTP message begins with a HTTP method (in a request) or a response code (in a response), followed by a header and (optionally) a message body.

There are four essential methods for data manipulation in REST: Create, Retrieve, Update and Delete (CRUD methods for short), and the web application layer protocol has to support them (as HTTP and CoAP do). In this way the web application layer becomes part of the service, and not just a wrapper around it. A resource is referenced with a globally unique Uniform Resource Identifier (URI link) and any RESTful resource is manipulated with the same set of CRUD methods.

B. Mapping of ETSI M2M to the Smart Grid

ETSI M2M architecture relies on RESTful web services. Essentially, ETSI standardised the resource structure that is hosted on the servers. In addition, between the network interfaces and the application, ETSI M2M introduces a middleware layer denoted *Service Capability Layer* (xSCL) to support the service logic, where x can stand for Network, Gateway or Device (therefore, we have NSCL, GSCL and DSCL modules). The purpose of xSCL is to hide the specific underlying network from the application and allow for the common interface to be used across all application types. For example, xSCL is responsible for establishing a transport session for the data transfer, taking care of all security aspects, checking if a device is online, executing remote software updates, etc. ETSI M2M is implemented in the Fraunhofer FOKUS testbed [11] and in InterDigital Communications prototype [12].

The ETSI M2M architecture is distributed rather than centralised. This is achieved with GSCL at the gateway, which is a lightweight web server meant to serve as a network proxy. In effect, GSCL offloads the processing load from the NSCL, which is a fully fledged web server. Figure 1 shows our mapping of ETSI M2M to the Smart Grid that is explained in the continuation.

The basic energy layer infrastructure of the power grid is organised into the transmission and the distribution grid. The transmission grid consists of high voltage lines which transmit electric power from the place where it is generated (hydro or coal plants) closer to the place where it is consumed (either by industrial or residential consumers). Before entering the distribution grid, power is stepped down to medium voltage in the primary substations. The primary substations mark the end of the transmission and the beginning of distribution grid. The distribution grid consists of medium voltage lines leading to the secondary substations, where the power is stepped down again to low voltage that is serviced to a consumer [13].

As Figure 1 shows, NSCLs are located in the control centre, in the transmission and/or distribution grid. PMUs and the consumer devices such as smart meters connect to the corresponding gateway (GSCL) either at the substation or at the consumer premises, respectively. Network applications are clients who subscribe to the resources at NSCL, published by the PMUs and other devices. According to Table I, Oscillation Monitor is a network application in the transmission control centre, State Estimation is located in the distribution control centre and Demand Response communications can go over the secure public Internet, given the relaxed QoS requirements. All critical measurement updates are exchanged on the private Internet.

ETSI M2M middleware supports the communication requirements given in Table I. Adjustable update rates are supported by the filtering functionality of NSCL and GSCL. Any publishing device can create its data resource that is announced to the applications through the middleware. In addition, because of the distributed architecture (GSCL in the substation or at the consumer, NSCL in the control centres), the solution is scalable. Flexibility is enabled with the standardised, generic resource structure that can be easily extended and modified to suit the specific application needs. Interoperability is provided with the web service technology, available to any participant using the web protocols (HTTP or CoAP). Security is inherently supported in xSCLs.

Regarding vertical requirements of each application in Table I, Demand Response is fully supported by the original ETSI M2M specification in [5]. Conversely, real-time communication requirements of Oscillation Monitoring and State Estimation pose a challenge for the current architecture. Detailed analysis of how the real-time communication requirements are

addressed with the upgrades proposed in this paper for ETSI M2M, is given in the next Section.

IV. Real-time Communication with REST

Although the subscription mechanism is supported by ETSI M2M, the solution to enable it presented in [5] only specifies HTTP long polling techniques. With long polling, instead of an immediate response, a client's HTTP request (poll) is held until a response is available at the server. Like this, the server can deliver an update as soon as it arrives because it has an open connection to the client provided by the long poll request. In the case of frequent updates, each response is immediately followed by a new poll, which (nearly) achieves real-time performance.

However, the long polling technique and the others alike are a workaround for the fact that HTTP clients are not designed to accept real-time updates. Given that there are applications whose main functionality critically depends on the real-time update delivery, we have considered two alternative solutions:

- CoAP using UDP/IP
- Websocket using TCP/IP

CoAP relies on UDP, which is connectionless, best-effort transport protocol. As such, UDP does not guarantee reliable, ordered delivery; there are no retransmissions of lost packets and thus no additional, unpredictable latency that is characteristic of TCP. Because of low latency, UDP is preferred to TCP for the delivery of PMU updates [3], even at the cost of some dropped packets. In particular, non-confirmable CoAP messages are suitable for PMU updates. CoAP supports the full set of CRUD methods and thus provides the required REST functionality.

With CoAP, the classical CRUD methods in the proposed middleware can be extended with Subscribe and Unsubscribe. Subscribe is a special case of GET request: a client subscribes to a resource thus indicating that it wants to be notified about any update of the resource, as long as the subscription is active. Therefore, Subscribe method extends the GET method by requesting the continuity of data retrievals. This is implemented by enabling the Observe option in the CoAP request header, as specified in [14]. The server sends a CoAP GET response (but without the header overhead) each time an update is available, without having to receive a CoAP GET request first.

On the other hand, some Smart Grid messages (e.g. control commands or alarms) require reliable, real-time delivery. In our solution, reliable message exchange is provided with Websockets. Websocket is an application layer protocol that enables a bidirectional, full-duplex socket connection over TCP/IP [15]. A Websocket connection is initiated with the special HTTP upgrade request that, if accepted and approved by the server, creates a persistent socket connection that is not released until explicitly requested (by the client or the server). After the HTTP upgrade request and response, the messages that follow do not conform to the HTTP format, but only add minimal headers to the application data. The full HTTP header is redundant because data is exchanged within an ongoing session and not as independent requests/responses. To comply with an event-driven communication scheme, a client application that supports Websockets needs to implement event listeners (onopen, onmessage, onclose). The message flow of long polling, CoAP and Websocket is compared in Figure 2.

Table II
HTTP long polling vs. CoAP and Websocket

	Header overhead	Network latency	TCP Acknowledgements
HTTP	$\simeq$ 100 bytes	est. 5 ms (RTT)	2 per each update
CoAP	6 bytes	est. 2.5 ms	N/A
WS	6 bytes	est. 2.5 ms	1 per each update
savings	94%	50%	100% (CoAP) 50% (WS)

Upgrading the ETSI M2M subscription mechanism with CoAP and Websockets can bring significant improvements to the real-time performance of the system. CoAP is intended for a stream of measurements generated at a predefined rate, whose timely delivery takes precedence over delivering each and every measurement. In other words, it is better to send new available measurement to the subscribed application than retransmit the old one, and therefore meet the strict latency requirement. Applications that use CoAP to get measurements tolerate certain message loss. Conversely, Websockets are targeted only at protection and control applications that exchange frequent messages exceeding e.g. one update per second, while complying with critical latency requirements and reliable message delivery.

Subscriptions with CoAP or Websocket (WS) complement the standard operations defined in ETSI M2M architecture. The improvements that directly influence the real-time performance are summarised in Table II. The estimates in the table highlight the most important improvements obtained with upgrades presented in this paper, in comparison to the original long polling solution. Table II focuses on the impact of the application layer protocol, rather than the influence of underlying network disturbances. To estimate the value of Round Trip Time (RTT), we assumed five hops over 100 km optical fibre each, and no queueing at routers (recall that network imperfections are not considered in the estimate). In detail, the improvements are the following:

- *Header overhead is reduced.* The values in Table II assume typical minimal HTTP header per request/response pair (no cookies). CoAP minimal header (4 bytes) is extended with the Observe option (2 bytes) to allow subscriptions [14]. In the Websocket case, only WS frame delimiters account for the header overhead. Since data is exchanged within an ongoing Websocket session, session parameters are exchanged only at the beginning, so that the communication can continue without the full HTTP header overhead attached to each data update.
- *Network latency is halved.* An update is sent as soon as it is available and it takes *one* network trip to deliver it, after which another update can be sent. HTTP request and response pairs take the network *round* trip per each update (RTT in Table II). HTTP application layer thus effectively doubles the latency component originating

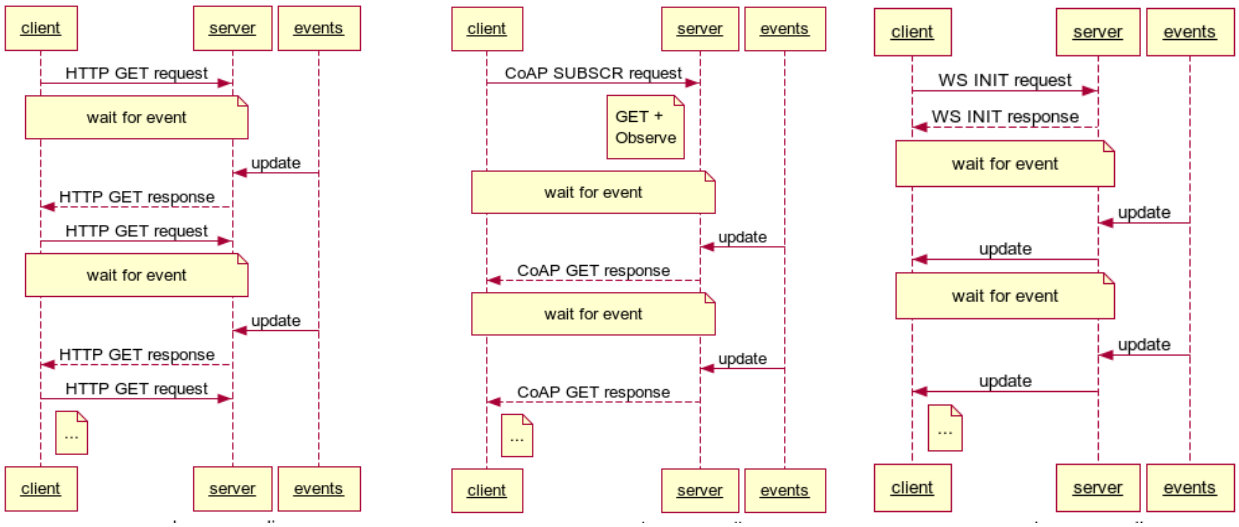

Figure 2. Sequence diagram for long polling (left), CoAP (middle) and Websocket messaging (right)

from the network, as can be seen in Figure 2.

- *Acknowledgements are either eliminated, or reduced.* Without Acknowledgements, there is less traffic and thus the network is less prone to congestion problems. HTTP requires that both the request and response are acknowledged, while with Websocket only the message containing updates on the data requested at the beginning of the session is acknowledged. Persistent TCP/IP connections which remain open for multiple messages were assumed in both cases.

Finally, there is a downside to integrating CoAP and Websockets into the architecture. The downside is the added complexity, since three different modes of operation are supported instead of just one: stateless and reliable HTTP, stateless and unreliable but low-latency CoAP and session-oriented, reliable and low-latency Websocket. However, this architecture reflects the diversity of the application requirements, as well as the heterogeneity of the complex communication infrastructure.

With these upgrades, the application requirements presented in Table I can be adequately met. Oscillation Monitoring and State Estimation subscribe to the PMU updates announced at the corresponding NSCL using CoAP. Control commands resulting from the real-time updates are sent back to the affected power grid components over Websocket sessions. On the other hand, Demand Response non-time-critical message exchange between GSCL at the consumer and the related application registered at NSCL uses the classical HTTP without long polling that fully supports all of its requirements.

V. Conclusion

The devices and applications necessary for the transition from the electric power grid to the Smart Grid already exist. A missing link is the communication system (including the middleware) that provides the connection between the two ends while complying with the set of stringent QoS requirements. We provide arguments for using the distributed ETSI M2M architecture as the basis of this middleware. The integration of the middleware into the Smart Grid network has been devised and the appropriate placement of each component is proposed in the paper. Notably, the middleware is universally deployed at the transmission, distribution and consumer side and covers a wide set of application requirements.

One particularly challenging communication requirement is the delivery of frequent updates in real-time over wide distances. ETSI M2M RESTful web services can hardly cope with real-time communications without the introduction of system upgrades. We therefore introduce CoAP for time-critical measurements stream and Websockets for time-critical control commands and alarms. These upgrades complement the standard functionality in order to respond to the application requirements that cannot be met otherwise. The thorough analysis of how the resulting heterogeneous solution is able to meet the communication requirements of the diverse Smart Grid applications is presented in the paper.

Acknowledgements

This work has been funded by the European Commission research projects ADVANTAGE (FP7-607774) and NEWCOM# (FP7-318306) as well as Generalitat de Catalunya under grant 2014 SGR 1551, the funding of which is gratefully acknowledged.

References

[1] C. Hauser, D. Bakken, and A. Bose, "A failure to communicate: next generation communication requirements, technologies, and architecture for the electric power grid," *IEEE Power and Energy Magazine*, vol. 3, no. 2, pp. 47–55, 2005.

[2] A. Scaglione, Z. Wang, and M. Alizadeh, "New models for networked control in smart grid," in *Smart Grid Communications and Networking*, E. Hossain, Z. Han, and V. Poor, Eds. Cambridge Univ. Press, 2012.

[3] E. Ancillotti, R. Bruno, and M. Conti, "The role of communication systems in smart grids: Architectures, technical solutions and research challenges," *Computer Communications, Elsevier*, vol. 36, no. 1718, pp. 1665 – 1697, 2013.

[4] H. Gjermundrod, D. Bakken, C. Hauser, and A. Bose, "GridStat: A Flexible QoS-Managed Data Dissemination Framework for the Power Grid," *IEEE Transactions on Power Delivery*, vol. 24, no. 1, pp. 136–143, 2009.

[5] ETSI M2M TR 102 690, "Machine-to-Machine communications (M2M); Functional architecture," ETSI, Tech. Rep., 2012.

[6] ETSI M2M TR 102 935, "Applicability of M2M architecture to Smart Grid Networks," ETSI, Tech. Rep., 2012.
[7] Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. [Online]. Available: http://tools.ietf.org/html/rfc6202
[8] J. Alonso-Zarate, J. Matamoros, D. Gregoratti, and M. Dohler, "Machine-to-machine communications in Smart Grid," in *Smart Grid Communications and Networking*, E. Hossain, Z. Han, and V. Poor, Eds. Cambridge University Press, 2012.
[9] M. Albano, L. Ferreira, L. Pinho, and A. Alkhawaja, "Message-oriented middleware for smart grids," *Computer Standards & Interfaces, Elsevier*, 2014.
[10] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine, USA, 2000.
[11] M. Corici, H. Coskun, A. Elmangoush, A. Kurniawan, T. Mao, T. Magedanz, and S. Wahle, "OpenMTC: Prototyping Machine Type communication in carrier grade operator networks," in *IEEE Globecom Workshops*, 2012, pp. 1735–1740.
[12] G. Lu, D. Seed, M. Starsinic, C. Wang, and P. Russell, "Enabling Smart Grid with ETSI M2M standards," in *IEEE Wireless Communications and Networking Conference Workshops*, 2012, pp. 148–153.
[13] U. Feuchtinger, R. Frank, J. Riedl, and K. Eger, "Smart Communications for Smart Grids," Siemens, Tech. Rep., 2012.
[14] Observing Resources in CoAP. [Online]. Available: http://tools.ietf.org/html/draft-ietf-core-observe-13
[15] The WebSocket Protocol. [Online]. Available: http://tools.ietf.org/html/rfc6455