TOC

6lowapp	Z. Shelby
Internet-Draft	Sensinode
Intended status: Informational	M. Garrison Stuber
Expires: June 26, 2010	Itron
	D. Sturek
	Pacific Gas & Electric
	B. Frank
	Tridium, Inc
	R. Kelsey
	Ember
	December 23, 2009

CoAP Feature Analysis
draft-shelby-6lowapp-coap-00

Abstract

This document considers the requirements and resulting features needed for the design of the Constrained Application Protocol (CoAP). Starting from requirements for energy and building automation applications, the basic features are identified along with an analysis of possible realizations. The goal of the document is to provide a basis for protocol design and related discussion.

Status of this Memo

Table of Contents

---

1. Introduction

TOC

The use of web services on the Internet has become ubiquitous in most applications, and depends on the basic REST architecture of the web. The proposed Constrained RESTful Environments (CoRE) working group aims at extending the REST architecture to a suitable form for the most constrained nodes (e.g. 8-bit microcontrollers with limited RAM and ROM) and networks (e.g. 6LoWPAN). One of the main goals of CoRE is to design a generic RESTful protocol for the special requirements of this constrained environment, especially considering energy and building automation applications. The result of this work should be a Constrained Application Protocol (CoAP) which easily traslates to HTTP for integration with the web while meeting specialized requirements such as multicast support, very low overhead and simplicity.
This document first analyzes the requirements for CoAP from the proposed charter and related application requirement drafts in Section 1.1 (CoAP Requirements). The key features needed for the CoAP protocol are then identified in Section 2 (CoAP Feature Analysis). Possible ways of realizing each feature are considered and recommendations made where possible. Finally, the applicability of these features to energy, building automation and general M2M applications is considered in Section 3 (Applicability).

1.1. CoAP Requirements

TOC

Paragraph 1. The following requirements for CoAP have been identified in the proposed charter of the working group, in the 6lowapp problem statement [I-D.bormann-6lowpan-6lowapp-problem] (Bormann, C., Sturek, D., and Z. Shelby, "6LowApp: Problem Statement for 6LoWPAN and LLN Application Protocols," July 2009.), or in the application specific requirement documents. The requirements relevant to CoAP are summarizes as follows:

> **REQ1:** Nodes have limited code size and limited RAM (typically on the order of 128K flash and 4K of RAM). [charter], [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman,

D., Stuber, M., and S. Ashton, “Smart Energy Requiements for 6LowApp,” October 2009.)

REQ2: A header size on the order of 10 bytes, and assumes an underlying network bandwidth of several dozen kbit/s. [charter]

REQ3: Ability to deal with sleeping nodes. Devices may be powered off at any point in time but periodically "wake up" for brief periods of time. [charter], [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, “Smart Energy Requiements for 6LowApp,” October 2009.), [I-D.gold-6lowapp-sensei] (Gold, R., Krco, S., Gluhak, A., and Z. Shelby, “SENSEI 6lowapp Requirements,” October 2009.)

REQ4: Protocol must support the caching of recent resource requests, along with caching subscriptions to sleeping nodes. [charter]

REQ5: Must support the manipulation of simple resources on constrained nodes and networks. The architecture requires push, pull and a notify approach to manipulating resources. CoAP will be able to set and query a resource on a Device. It will allow a Device to publish a value or event to another Device. [charter], [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, “Smart Energy Requiements for 6LowApp,” October 2009.), [I-D.martocci-6lowapp-building-applications] (Martocci, J. and A. Schoofs, “Commercial Building Applications Requirements,” October 2009.), [I-D.gold-6lowapp-sensei] (Gold, R., Krco, S., Gluhak, A., and Z. Shelby, “SENSEI 6lowapp Requirements,” October 2009.)

REQ6: Must define a mapping from CoAP to a HTTP REST API; this mapping will not depend on a specific application and must be as transparent as possible using standard protocol response and error codes where possible. [charter], [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, “Smart Energy Requiements for 6LowApp,” October 2009.), [I-D.gold-6lowapp-sensei] (Gold, R., Krco, S., Gluhak, A., and Z. Shelby, “SENSEI 6lowapp Requirements,” October 2009.)

REQ7: Each interface profile will define the Resources on the Device that applications can manipulate and what manipulations are possible. CoAP must be able to discover Devices on the CNN and to interrogate them to find out which interface profiles they support. [charter]

REQ8: CoAP will support a non-reliable multicast message to be sent to a group of Devices to manipulate a resource on all the Devices simultaneously (roughly within 50 ms of each other) [charter]. The use of multicast for discovery and advertisement must be supported, along with the support of unicast responses [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, "Smart Energy Requiements for 6LowApp," October 2009.).

REQ9: The protocol will operate by default over UDP, it may optionally be bound to TCP or other reliable transports. [charter], [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, "Smart Energy Requiements for 6LowApp," October 2009.), [I-D.martocci-6lowapp-building-applications] (Martocci, J. and A. Schoofs, "Commercial Building Applications Requirements," October 2009.)

REQ10: Reliability must be provided for application layer messages [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, "Smart Energy Requiements for 6LowApp," October 2009.). Must achieve < 0.01% Application layer errors on all messages assuming a .1% Network layer error rate.

REQ11: Latency times should be mimimized of the Home Area Network (HAN), and ideally a typical exchange should consist of just a single request and a single response message. [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, "Smart Energy Requiements for 6LowApp," October 2009.)

REQ12: Internet media type and transfer encoding type support. [I-D.sturek-6lowapp-smartenergy] (Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, "Smart Energy Requiements for 6LowApp," October 2009.), [I-D.gold-6lowapp-sensei] (Gold, R., Krco, S., Gluhak, A., and Z. Shelby, "SENSEI 6lowapp Requirements," October 2009.)

---

2. CoAP Feature Analysis

TOC

This section introduces the minimum set of features needed to realize CoAP, and looks at the possible options for realizing them. These features are considered in light on the requirements listed in

Section 1.1 (CoAP Requirements). The goal is to consider the cross-dependencies, benefits and drawbacks of alternatives for realizing CoAP and to narrow down the options where obvious.

2.1. Compact Header

TOC

There is a requirement for a header overhead on the order of 10 bytes with limited complexity due to node limitations. It should be noted that in wireless networks bits transmitted are much more expensive than processing cycles. The following header design options are considered:

Fixed approach: The simplest approach is to assume as fixed set of byte-aligned fields. The use of variable length fields should be avoided if possible, one obvious exception being a string URL (see Section 2.5 (URLs)). This results in a simple header that can be represented as a struct and easily parsed/created. The disadvantages are difficult evolvability and the tendency to design missing tranport features on-top of CoAP.

Extensible approach: The approach of [I-D.frank-6lowapp-chopan] (Frank, B., "Chopan - Compressed HTTP Over PANs," September 2009.) is to encode HTTP headers as binary tuples, assuming that a large number of optional headers will be needed. A similar approach could be takenin CoAP, giving total header flexibility. The disadvantage is header parsing complexity.

Hybrid approach: It is unclear how much extensibility is really required from the headers of CoAP. Some of the fields in the protocol will obviously require a sufficient value space for future extensions, such as for indicating content type. Other headers are clearly optional, such as those related to cache control (see Section 2.6 (Caching)) or even the URL (see Section 2.5 (URLs)). A hybrid approach would be to design a small fixed header, with the ability to include extension headers, such as in ICMP [RFC0792] (Postel, J., "Internet Control Message Protocol," September 1981.).

Considering the features foreseen by this document, some kind of extensible hybrid approach is recommended. Most features are fixed for messages, whereas only a some are expected to be optional.

TOC

2.2. Basic Messages

It is assumed that basic Request and Response messages will be required by the CoAP protocol. This also provides a natural mapping to HTTP (See Section 2.10 (HTTP Mapping)) and the response may be useful as an acknowledgement in UDP reliability (See Section 2.8.1 (UDP)). It can be considered that CoAP methods are different kinds of Request messages.

2.3. REST Methods

TOC

The core methods of REST must be supported within CoAP. To minimize confusion with HTTP methods, having their own protocol semantics, in CoAP we call the basic REST methods READ, WRITE, CREATE, DESTROY. Additionally, CoAP must support a light-weight Subscribe/Notify mechanism (see Section 2.7 (Subscribe/Notify)). This may require a new NOTIFY method. The discovery mechanism of CoAP may also require a new method called DISCOVER which has different semantics than a READ (see Section 2.9 (Resource Discovery)). In order to maintain compatibility with HTTP, these new messages must be mapped to a standard HTTP method. See Section 2.10 (HTTP Mapping) for more about HTTP mapping.

2.4. Content-type encoding

TOC

In order to support hetergenous uses, it is important that CoAP is transparent to the use of different application payloads. In order for the application process receiving a packet to properly parse a payload, its content-type and encoding should be explicitly known from the header (as e.g. with HTTP). The use of typical binary encodings for XML is discussed in [I-D.shelby-6lowapp-encoding] (Shelby, Z., Luimula, M., and D. Peintner, “Efficient XML Encoding and 6LowApp,” October 2009.), which includes recommendations for header indication. The draft recommends the indication of at least 10 Internet media types (MIME) [RFC2046] (Freed, N. and N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types,” November 1996.) and 2 content transfer encodings.
It is obvious that string names of Internet media types [RFC2046] (Freed, N. and N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types,” November 1996.) are not appropriate for use in the CoAP header. But then how to make this small yet extensible? One possible solution is to simply assign codes to a small subset of common MIME and content transfer encoding types and have IANA maintain that. A field of 16-bits should be sufficient for encoding both media and content transfer encoding types.

2.5. URLs

TOC

The Universal Resource Locator (URL) [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.) is an important feature of the REST architecture, the relative part of the URL indicates which resource on the server is being manipulated. It is surely useful for CoAP to support string URLs, which requires a variable length-value field. Although URLs can be designed for compactness, this still often results in 10s of bytes of overhead. The encoding of of a URL string needs to be considered, as this is becoming increasingly complex. It is recommended that only US-ASCII is supported in URL strings for CoAP as defined in [RFC3986] (Berners-Lee, T., Fielding, R., and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” January 2005.), or even a stricter subset as URL parsing is complex and may result in security problems.
Constrained devices are not general purpose web servers, and thus often won't host but a small set of resources with fixed URLs. Thus in addition to string URLs a feature for compressing fixed URLs would be useful.
One way of achieving this would be to assign an integer identifier (7-8 bits should be sufficient) to each fixed URL in an off-line interface description (e.g. Web Application Description Langauge (WADL)) or in its profile. This identifier could be encoded in the URL length field instead of the string length.

2.6. Caching

TOC

The cachability of CoAP messages will be important, especially with the sleeping node configurations and power limitations typically found in constrained networks and nodes. What features of cachability are really required and how much energy are we willing to spend on it? Roughly 50% of the HTTP specifications are dedicated to sohpisticated caching. With CoAP we should look at the bare minimum caching feature possible.
Before talking about caching solutiongs, we should consider in what scenarios caching will actually be required. The following two scenarios have been identified:

* An intermediate CoAP proxy may cache resources and answer READ requests using a cached version. The resource may be cached from previous responses or notifications. This requires at least Max-Age cache control information about each resource.

*An intermediate CoAP proxy may cache subscriptions to a sleeping node. This requires at least Max-Age information about the subscription.

Three possible approaches have been identified for caching support.

In-band approach: One approach is suggested in [I-D.frank-6lowapp-chopan] (Frank, B., “Chopan - Compressed HTTP Over PANs,” September 2009.), which analyses the subset of features from HTTP that could be used for simple sensor data purposes. The proposal is that simply using the using the HTTP Age header (for resource age) and Cache-Control header (for max-age). Max-age may also be applied in requests. Both headers make use of a 2-byte value in seconds. The advantage of this approach is that cache control information is easily available from the header. The disadvantage is the header overhead.

Out-of-band approach: Here the CoAP protocol would be agnostic to the cachability of the resources it is carrying, instead leaving the definition of cache control parameters to the body of the resources in an application specific way. The disadvantage is that this makes proxies dependent on the application.

Discovery approach: In this approach the cache control information for resources is defined off-line in the list of a server's resources. This approach is used e.g. in the SENSEI system [I-D.gold-6lowapp-sensei] (Gold, R., Krco, S., Gluhak, A., and Z. Shelby, “SENSEI 6lowapp Requirements,” October 2009.). The disadvantage id that the caching is dependent on the profile, which may not be a problem if the cache information is in a universal format (see Section 2.9 (Resource Discovery)).

Based on the current analysis either the In-band or Profile approach would be reasonable for CoAP considering the requirements.

2.7. Subscribe/Notify

TOC

CoAP is required to integrate a push model for interaction in addition to traditional request/response. Meaning that interested clients could subscribe to a resource (a URL), and receive notifications to a call-back URL of their choice. In its most basic form a notification would be sent each time the resource changes. There are many issues to consider including managing subscription leasing and timeouts, how to batch multiple changes and how to tune notify times. Before considering the details, there are a few general general models possible for realizing the Subscribe/Notify mechanism:

Resource:
Subscribe is realized using CREATE on a well known resource (e.g. /subsribe) with the URL of the resource of interest and a URL call-back in the body). Notifications would be made using a NOTIFY message to the call-back URL. Likewise, de-subscribe is realized using DESTROY on the same well known resource with the URL in the body. Notifications would cease after the DESTROY.

Watch: This method would require a CREATE to a new URI to "create" a new watch resource. WRITE is then used to add/remove a set of URIs being "watched" along with call-backs.

2.8. Transport Binding

TOC

The CoAP protocol will operate by default over UDP and it may optionally be bound to TCP or other reliable transports. In this section we look at the issues regarding these bindings.

2.8.1. UDP

TOC

The goal of binding CoAP to UDP is to provide the bare minimum features for the protocol to operate over UDP, going nowhere near trying to re-create the full feature set of TCP. The bare minimum features required would be:

* Stop-and-wait would be sufficient for reliability. A simple response message itself would suffice as an acknowledgement with retransmission support. Not all requests require reliability, thus this should be optional. Performance is not the key here and for more sophisticated reliability and flow control TCP can be used.

* A sequence number (transaction ID) is needed to match responses to open requests and would be generated by the client. A 12-16 bit unsigned interger would be sufficient. [I-D.frank-6lowapp-chopan] (Frank, B., “Chopan - Compressed HTTP Over PANs,” September 2009.) also considered this solution.

* Multicast support. Providing reliability with a multicast destination address would be very complex. Therefore the goal is to provide non-reliable multicast service. In many cases there may not be a response to a multi-cast message. A multicast

command might result in an action being taken at a device, but no response being sent. Therefore a multicast request may be answered with a unicast response, however without reliability (retransmission e.g.).

2.8.2. TCP

TOC

The CoAP protocol also has the goal of defining a TCP binding. As TCP provides a reliable stream this binding does not require anything special from the CoAP protcol design. The same basic messages could be applied over TCP without stop-and-wait. A transaction ID should still be used over TCP.

2.9. Resource Discovery

TOC

CoAP is required to support the discovery of resources offered by CoAP servers. In order to achieve this, the protocol would need to suport multicast with optional responses for discovery (over UDP), along with unicast requests and posting of profiles (over UDP or TCP). A well-known resource (/profile) could be used to enable profile discovery through a new DISCOVER method along with profile sending on an optional broker with a CREATE or modification with a WRITE to /profile. The response to a DISCOVER message would include a list of resource URLs available, or those matched if the DISCOVER has a body with one or more URLs. Such a resource list could include optional information such as the URL to a description of the interface.
Section 2.6 (Caching) discusses different caching models. If caching would be realized using the Profile model, then the resource list would need to indicate at least the Max-Age of each resource.

2.10. HTTP Mapping

TOC

It shall be possible to map from CoAP directly to HTTP, CoAP however only offers a subset of the HTTP protocol features. As a result, programs implementing translation between HTTP and CoAP must either implement other HTTP 1.1 commands on behalf of the CoAP nodes (e.g. LINK, TRACE, OPTIONS), or must reject such request. The primary responsibility of a program translating between HTTP and CoAP is to rewrite the headers, translating between the highly optimized CoAP headers and plain text HTTP headers. It must also manage/maintain TCP sessions necessary for HTTP. Depending on how some of the features of CoAP are realized, the mapping may also need to make further translations for subscription or caching.

---

3. Applicability

TOC

This sections looks at the applicability of the CoAP features for energy, building automation and other macine-to-machine (M2M) applications.

3.1. Energy Applications

TOC

Rising energy prices, concerns about global warming and energy resource depletion, and societal interest in more ecologically friendly living have resulted in government mandates for Smart Energy solutions. In a Smart Energy environment consumers of energy have direct, immediate access to information about their consumption, and are able to take action based on that information. Smart Energy systems also allow device to device communication to optimize the transport, reliability, and safety of energy delivery systems. While often Smart Energy solutions are electricity-centric, i.e. Smart Grid, gas and water are also subject to the same pressures, and can benefit from the same technology.
Smart Energy Transactions typically include the exchange of current consumption information, text messages from providers to consumers, and control signals requesting a reduction in consumption. Advanced features such as billing information, energy prepayment transactions, management of distributed energy resources (e.g. generators and photo-voltaics), and management of electric vehicles are also being developed.
Smart Energy benefits from Metcalfe's Law. The more devices that are part of a smart energy network within the home or on the grid, the more valuable it becomes. Showing a consumer how much energy they are using is useful. Combining that with specific information about their major appliances, and enabling them to adjust their consumption based on current pricing and system demand is much much more powerful. To do this however requires a system that is resillient, low cost, and easy to install. In many areas this is being done with systems built around IEEE 802.15.4 radios. In the United States, there are over 30 million electric meters that will be deployed with these radios. These radios will be combined to form a mesh network, enabling Smart Energy communication within the home. The maximum packet size for IEEE 802.15.4 is only 127 bytes. Additionally, there is the well known issue of how TCP manages congestion working sub-optimally over wireless

networks. IEEE 802.15.4 is ideal for these applications because of its low cost and its support for battery powered devices; however, it is not as well suited for heavier protocols like HTTP. These technical issues with IEEE 802.15.4 networks combined with a desire to facilitate broader compatibility, makes a protocol like CoAP desireable. Its REST architecture will allow seamless compatibility with the rest of the Internet, allowing it to be easily integrated with web browsers and web-based service providers, while at the same time being appropriately sized for the low-cost networks necessary for its success.

3.2. Building Automation

TOC

Building automation applications were analyzed in detail including use cases in [I-D.martocci-6lowapp-building-applications] (Martocci, J. and A. Schoofs, “Commercial Building Applications Requirements,” October 2009.). Although many of the embedded control solutions for building automation make use of industry-specific application protocols like BACnet over IP, there is a growing use of web services in building monitoring, remote control and IT integration. The OASIS oBIX standard [ref] is one example of the use of web services for the monitoring and interconnection of heterogeneous building systems. Several of the CoAP requirements have been taken from [I-D.martocci-6lowapp-building-applications] (Martocci, J. and A. Schoofs, “Commercial Building Applications Requirements,” October 2009.). The resulting features should allow for peer-to-peer interactions as well as node-server request/response and push interfactions for monitoring and some control purposes. For building automation control with very strict timing requirements using e.g. multicast, further features may be required on top of CoAP.

3.3. General M2M Applications

TOC

CoAP provides a natural extension of the REST architecture into the domain of constrained nodes and networks, aiming at requirements from automation applications in energy and building automation. A very wide range of machine-to-machine (M2M) applications have similar requirements to those considered in this document, and thus it is foreseen that CoAP may be widely applied in the industry. One standardization group considering a general M2M architecture and API is the ETSI M2M TC [ref], which considers a wide range of applications including energy. Another group developing solutions for general embedded device control is the OASIS Device Proile Web Services (DPWS) group. The consideration of DPWS over 6LoWPAN is available in

[I-D.moritz-6lowapp-dpws-enhancements] (Moritz, G., “DPWS for 6LoWPAN,” December 2009.).

4. Conclusions

TOC

This document analyzed the requirements associated with the design of the foreseen Constrained Application Protocol (CoAP). Based on these requirements a list of minumum features was analyzed along with different options for realizing them. If possible a recommendation was also made where obvious. Finally, the identified features of CoAP are considered for energy, building automation and M2M applications. This document is meant to serve as a basis for the design of the CoAP protocol and relevant discussion.

CoAP is proposed as a transport agnostic extension of REST for deployment in confined computing environments. The intent is to align CoAP with HTTP wherever possible to leverage the web services computing environment already in place.

Whereas REST envisions just 4 primitives (READ, SET, CREATE and DESTROY), CoAP proposes to extend this paradigm with a NOTIFY primitive to enable publish/subscribe along with a DISCOVER primitive to support multicast discovery of services denoted by URL. The main architectural difference between READ and the new discovery primitive is the requirement to not respond if the URL is not present on a local device.

Finally, CoAP seeks to preserve the caching facilities of HTTP and extend that capability for power saving devices that are not always active on the network.

5. Security Considerations

TOC

Some of the features considered in this document will need further security considerations during a protocol design. For example the use of string URLs may have entail security risks due to complex processing on limited microcontroller implementations.

The CoAP protocol will be designed for use with (D)TLS or object security. A protocol design should consider how integration with these security methods will be done, how to secure the CoAP header and other implications.

6. IANA Considerations

TOC

This draft requires no IANA consideration.

7. Acknowledgments

TOC

Thanks to Cullen Jennings for helpful comments and discussions.

8. References

TOC

8.1. Normative References

TOC

[I-D.frank-6lowapp-chopan]	Frank, B., "Chopan - Compressed HTTP Over PANs," draft-frank-6lowapp-chopan-00 (work in progress), September 2009 (TXT).
[I-D.gold-6lowapp-sensei]	Gold, R., Krco, S., Gluhak, A., and Z. Shelby, "SENSEI 6lowapp Requirements," draft-gold-6lowapp-sensei-00 (work in progress), October 2009 (TXT).
[I-D.martocci-6lowapp-building-applications]	Martocci, J. and A. Schoofs, "Commercial Building Applications Requirements," draft-martocci-6lowapp-building-applications-00 (work in progress), October 2009 (TXT).
[I-D.shelby-6lowapp-encoding]	Shelby, Z., Luimula, M., and D. Peintner, "Efficient XML Encoding and 6LowApp," draft-shelby-6lowapp-encoding-00 (work in progress), October 2009 (TXT).
[I-D.sturek-6lowapp-smartenergy]	Sturek, D., Shelby, Z., Lohman, D., Stuber, M., and S. Ashton, "Smart Energy Requiements for 6LowApp," draft-sturek-6lowapp-smartenergy-00 (work in progress), October 2009 (TXT).
[RFC2046]	Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," RFC 2046, November 1996 (TXT).
[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," STD 66, RFC 3986, January 2005 (TXT, HTML, XML).

8.2. Informative References

TOC

[I-D.bormann-6lowpan-6lowapp-problem]	Bormann, C., Sturek, D., and Z. Shelby, "6LowApp: Problem Statement for 6LoWPAN and LLN Application Protocols," draft-bormann-6lowpan-6lowapp-problem-01 (work in progress), July 2009 (TXT).
[I-D.moritz-6lowapp-dpws-enhancements]	Moritz, G., "DPWS for 6LoWPAN," draft-moritz-6lowapp-dpws-enhancements-00 (work in progress), December 2009 (TXT).
[RFC0792]	Postel, J., "Internet Control Message Protocol," STD 5, RFC 792, September 1981 (TXT).

Authors' Addresses

TOC

	Zach Shelby
	Sensinode
	Kidekuja 2
	Vuokatti 88600
	FINLAND
Phone:	+358407796297
Email:	zach@sensinode.com
	Michael Garrison Stuber
	Itron
	2111 N. Molter Road
	Liberty Lake, WA 99025
	U.S.A.
Phone:	+1.509.891.3441
Email:	Michael.Stuber@itron.com
	Don Sturek
	Pacific Gas & Electric
	77 Beale Street
	San Francisco, CA
	USA
Phone:	+1-619-504-3615
Email:	d.sturek@att.net
	Brian Frank
	Tridium, Inc
	Richmond, VA
	USA

Phone:	
Email:	brian.tridium@gmail.com
	Richard Kelsey
	Ember
	47 Farnsworth Street
	Boston, MA 02210
	U.S.A.
Phone:	+1.617.951.1201
Email:	richard.kelsey@ember.com