

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 22, 2013

J. Arkko
A. Eriksson
A. Keranen
Ericsson
February 18, 2013

Building Power-Efficient CoAP Devices for Cellular Networks
draft-arkko-lwig-cellular-00

Abstract

This memo discusses the use of the Constrained Application Protocol (CoAP) protocol in building sensors and other devices that employ cellular networks as a communications medium. Building communicating devices that employ these networks is obviously well known, but this memo focuses specifically on techniques necessary to minimize power consumption.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Goals for Low-Power Operation	4
3. Link-Layer Assumptions	7
4. Scenarios	9
5. Discovery and Registration	9
6. Data Formats	11
7. Real-Time Reachable Devices	11
8. Sleepy Devices	12
8.1. Implementation Considerations	13
9. Security Considerations	14
10. IANA Considerations	14
11. References	15
11.1. Normative References	15
11.2. Informative References	15
Appendix A. Acknowledgments	16
Authors' Addresses	16

1. Introduction

This memo discusses the use of the Constrained Application Protocol (CoAP) protocol [I-D.ietf-core-coap] in building sensors and other devices that employ cellular networks as a communications medium. Building communicating devices that employ these networks is obviously well known, but this memo focuses specifically on techniques necessary to minimize power consumption. CoAP has many advantages, including being simple to implement; a thousand lines for the entire software above IP layer is plenty for a CoAP-based sensor, for instance. However, while many of these advantages are obvious and easily obtained, optimizing power consumption remains challenging and requires careful design [I-D.arkko-core-sleepy-sensors].

The memo targets primarily 3GPP cellular networks in their 2G, 3G, and LTE variants and their future enhancements, including possible power efficiency improvements at the radio and link layers. The exact standards or details of the link layer or radios are not relevant for our purposes, however. To be more precise, the material in this memo is suitable for any large-scale, public network that employs point-to-point communications model and radio technology.

Our focus is devices that need to be optimized for power usage, and on devices that employ CoAP. As a general technology, CoAP is similar to HTTP. It can be used in various ways and network entities may take on different roles. This freedom allows the technology to be used in efficient and less efficient ways. Some guidance is needed to understand what communication models over CoAP are recommended when low power usage is a critical goal.

The recommendations in this memo should be taken as complementary to device hardware optimization, microelectronics improvements, and further evolution of the underlying link and radio layers. Further gains in power efficiency can certainly be gained on several fronts; the approach that we take in this memo is to do what can be done at the IP, transport, and application layers to provide the best possible power efficiency. Application implementors generally have to use the current generation microelectronics, currently available radio networks and standards, and so on. This focus in our memo should by no means be taken as an indication that further evolution in these other areas is unnecessary. Such evolution is useful, is ongoing, and is generally complementary to the techniques presented in this memo. The evolution of underlying technologies may change what techniques described here are useful for a particular application, however.

The rest of this memo is structured as follows. Section 2 discusses the need and goals for low-power devices. Section 3 outlines our

expectations for the low layer communications model. Section 4 describes the two scenarios that we address, and Section 5, Section 6, Section 7 and Section 8 give guidelines for use of CoAP in these scenarios.

2. Goals for Low-Power Operation

There are many situations where power usage optimization is unnecessary. Optimization may not be necessary on devices that can run on power feed over wired communications media, such as in Power-over-Ethernet (PoE) solutions. These devices may require a rudimentary level of power optimization techniques just to avoid keep overall energy costs and aggregate power feed sizes at a reasonable level, but more extreme techniques necessary for battery powered devices are not required. The situation is similar with devices that can be easily be connected to mains power. Other types of devices may get an occasional charge of power from energy harvesting techniques. For instance, some environmental sensors can run on solar cells. Typically, these devices still have to regulate their power usage in a strict manner, for instance to be able to use as small and inexpensive solar cells as possible.

In battery operated devices the power usage is even more important. For instance, one of the authors employs over a hundred different sensor devices in his home network. A majority of these devices are wired and run on PoE, but in most environments this would be impractical because the necessary wires do not exist. The future is in wireless solutions that can cover buildings and other environments without assuming a pre-existing wired infrastructure. In addition, in many cases it is impractical to provide a mains power source. Often there are no power sockets easily available in the locations that the devices need to be in, and even if there were, setting up the wires and power adapters would be more complicated than installing a standalone device without any wires.

Yet, with a large number of devices the battery lifetimes become critical. Cost and practical limits dictate that devices can be largely just bought and left on their own. For instance, with hundred devices, even a ten-year battery lifetime results in a monthly battery change for one device within the network. This may be impractical in many environments. In addition, some devices may be physically difficult to reach for a battery change. Or, a large group of devices -- such as utility meters or environmental sensors -- cannot be economically serviced too often, even if in theory the batteries could be changed.

POWER SOURCE	SENSOR COMMUNICATION INTERVAL		
	Seconds	Minutes or Hours	Days and longer
Battery	Low-power	Low-power or Always-off	Always-off
Harvesting	Low-power	Low-power or Always-off	Always-off
Mains	Always-on	Always-on	Always-on

Figure 1: Power usage strategies for different classes of applications

Many of these situations lead to a requirement for minimizing power usage and/or maximizing battery lifetimes. A summary of the different situations for sensor-type devices is shown in Figure 1 above. Unfortunately, much of our current technology has been built with different objectives in mind. Networked devices that are "always on", gadgets that require humans to recharge it every couple of days, and protocols that have been optimized to maximize throughput rather than conserve resources.

Long battery lifetimes are required for many applications, however. In some cases these lifetimes should be in the order of years or even a decade or longer. Some communication devices already reach multi-year lifetimes, and continuous improvement in low-power electronics and advances in radio technology keep pushing these lifetimes longer. However, it is perhaps fair to say that battery lifetimes are generally too short at present time.

The general strategies for power usage from Figure 1 can be described as follows:

Always-on

Under this strategy, there is no reason for extreme measures for power saving. The device can stay on in the usual manner all the time. It may be useful to employ a power-friendly hardware or limit the number of wireless transmissions, CPU speeds, and other aspects for general power saving and cooling needs, but the device can be in the network all the time.

Always-off

Under this strategy, the device sleeps such long periods at a time that once it wakes up, it makes sense for it to not pretend that it is connected to the network during those times. These devices re-attach to the network as they are woken up. The main optimization goal is to minimize the effort during such re-attachment process and any resulting application communications.

If the device sleeps for long periods of time, the relative increase in energy expenditure during reattachment for infrequent communication may be acceptable.

Low-power

These devices need to operate on very small amount of power, but still be able to communicate in a relatively frequent basis. This implies that extremely low power solutions needs to be used for the hardware, chosen link layer mechanisms, and so on. Typically, given the small amount of time between transmissions, despite their sleep state these devices retain some form of network attachment to the network. Techniques used for minimizing power usage for the network communications include minimizing any work from re-establishing communications after waking up, tuning the communications frequency, and paging frequency and other parameters appropriately.

Power usage can not be evaluated solely based on lower layer communications. The entire system, including upper layer protocols and applications is responsible for the power consumption as a whole. The lower communication layers have already adopted many techniques that can be used to reduce power usage, such as scheduling device wake-up times. Further reductions will likely need some co-operation from the upper layers so that unnecessary communications, denial-of-service attacks on power consumption, and other power drains are eliminated.

Of course, application requirements ultimately determine what kinds of communications are necessary. For instance, some applications require more data to be sent than others. The purpose of the guidelines in this memo is not to prefer one or the other application, but to provide guidance on how to minimize the amount of communications overhead that is not directly required by the application. While such optimization is generally useful, it is relatively speaking most noticeable in applications that transfer only a small amount of data, or operate only infrequently.

3. Link-Layer Assumptions

We assume that the underlying communications network can be any large-scale, public network that employs point-to-point communications model and radio technology. 2G, 3G, and LTE networks are examples of such networks, but not the only possible networks with these characteristics.

In the following we look at some of these characteristics and their implications. Note that in most cases these characteristics are not properties of the specific networks but rather inherent in the concept of public networks.

Public networks

Using a public network service implies that applications can be deployed without having to build a network to go with them. For economical reasons, only the largest users (such as utility companies) could afford to build their own network, and even they would not be able to provide a world-wide coverage. This means that applications where coverage is important can be built. For instance, most transport sector applications require national or even world-wide coverage to work.

But there are other implications, as well. By definition, the network is not tailored for this application and with some exceptions, the traffic passes through the Internet. One implication of this is that there are generally no application-specific network configurations or discovery support. For instance, the public network helps devices to get on the Internet, set up default routers, configure DNS servers, and so on, but does nothing for configuring possible higher-layer functions, such as servers the device might need to contact to perform its application functions.

Public networks often provide web proxies, and these can in some cases make a significant improvement for delays and cost of communication over the wireless link. For instance, collecting content from a large number of servers used to render a web page and resolving their DNS names in a proxy instead of the user's device may cut down on the general chattiness of the communications, therefore reducing overall delay in completing the entire transaction. However, as of today such proxies are provided only for HTTP communications, not for CoAP.

Similarly, given the lack of available IPv4 addresses, the chances are that many devices are behind a network address translation (NAT) device. This means that they are not easily reachable as

servers. Alternatively, the devices may be directly on the global Internet (either on IPv4 or IPv6) and easily reachable as servers. Unfortunately, this may mean that they also receive unwanted traffic, which may have implications for both power consumption and service costs.

Point-to-point link model

This is a common link model in cellular networks. One implication of this model is that there will be no other nodes on the same link, except maybe for the service provider's router. As a result, multicast discovery can not be reasonably used for any local discovery purposes. While the configuration of the service provider's router for specific users is theoretically possible, in practice this is difficult to achieve, at least for any small user that can not afford a network-wide contract for a private APN. The public network access service has little per-user tailoring.

Radio technology

The use of radio technology means that power is needed to operate the radios. Transmission generally requires more power than reception. However, radio protocols have generally been designed so that a device checks periodically whether it has messages. In a situation where messages arrive seldom or not at all, this checking consumes energy. Research has shown that these periodic checks (such as LTE paging message reception) are often a far bigger contributor to energy consumption than message transmission.

Note that for situations where there are several applications on the same device wishing to communicate with the Internet in some manner, bundling those applications together at the same time can be very useful. Some guidance for these techniques in the smartphone context can be found in [Android-Bundle].

Naturally, each device has a freedom to decide when it sends messages. In addition, we assume that there is some way for the devices to control when or how often it wants to receive messages. Specific methods for doing this depend on the specific network being used and also tend to change as improvements in the design of these networks are incorporated. The reception control methods generally come in two variants, fine grained mechanisms that deal with how often the device needs to wake-up for paging messages, and more crude mechanisms where the device simply disconnects from the network for a period of time. There are associated costs and benefits to each method, but those are not relevant for this memo, as long as some control method exists.

4. Scenarios

Not all applications or situations are equal. They may require different solutions or communication models. This memo focuses on two common scenarios:

Real-Time Reachable Devices

This scenario involves all communication that requires real-time or near real-time communications with a device. That is, a network entity must be able to reach the device with a small time lag at any time, and no pre-agreed wake-up schedule can be arranged. By "real-time" we mean any reasonable end-to-end communications latency, be it measured in milliseconds or seconds. However, unpredictable sleep states are not expected.

Examples of devices in this category include sensors that must be measurable from a remote source at any instant in time, such as process automation sensors and actuators that require immediate action, such as light bulbs or door locks.

Sleepy Devices

This scenario involves freedom to choose when device communicates. The device is often expected to be able to be in a sleep state for much of its time. The device itself can choose when it communicates, or it lets the network assist in this task.

Examples of devices in this category include sensors that track slowly changing values, such as temperature sensors and actuators that control a relatively slow process, such as heating systems.

Note that there may be hard real-time requirements, but they are expressed in terms of how fast the device can communicate, not in terms of how fast it can respond to a network stimuli. For instance, a fire detector can be classified as a sleepy device as long as it can internally quickly wake up on detecting fire and initiate the necessary communications without delay.

5. Discovery and Registration

In both scenarios the device will be attached to a public network. Without special arrangements, the device will also get a dynamically assigned IP address or an IPv6 prefix. At least one but typically several router hops separate the device from its communicating peers such as application servers. As a result, the address or even the existence of the device is typically not immediately obvious to the

other nodes participating in the application. As discussed earlier, multicast discovery has limited value in public networks; network nodes cannot practically discover individual devices in a large public network. And the devices can not discover who they need to talk, as the public network offers just basic Internet connectivity.

Our recommendation is to initiate a discovery and registration process. This allows each device to inform its peers that it has connected to the network and that it is reachable at a given IP address.

The registration part is easy; a resource directory or mirror proxy can be used. The device should perform the necessary registration with these devices, for instance, as specified in [I-D.shelby-core-resource-directory] and [I-D.vial-core-mirror-proxy]. In order to do this registration, the device needs to know its CORE Link Format description, as specified in [RFC6690]. In essence, the registration process involves performing a GET on `.well-known/core/?rt=core-rd` at the address of the resource directory (or `rt=core-mp` for mirror proxies), and then doing a POST on the path of the discovered resource.

However, current CoAP specifications provide limited support for discovering the resource directory or mirror proxy. Local multicast discovery only works in LAN-type networks, but not in these public cellular networks. Our recommended alternate methods for discovery are the following:

Manual Configuration

The DNS name of the resource directory or mirror proxy is manually configured. This approach is suitable in situations where the owner of the devices has the resources and capabilities to do the configuration. For instance, a utility company can typically program its metering devices to point to the company servers.

Manufacturer Server

The DNS name of the directory or proxy is hardwired to the software by the manufacturer, and the directory or proxy is actually run by the manufacturer. This approach is suitable in many consumer usage scenarios, where it would be unreasonable to assume that the consumer runs any specific network services. The manufacturer's web interface and the directory/proxy servers can co-operate to provide the desired functionality to the end user. For instance, the end user can register a device identity in the manufacturer's web interface and ask specific actions to be taken when the device does something.

Delegating Manufacturer Server

The DNS name of the directory or proxy is hardwired to the software by the manufacturer, but this directory or proxy merely redirects the request to a directory or proxy run by the whoever bought the device. This approach is suitable in many enterprise environments, as it allows the enterprise to be in charge of actual data collection and device registries; only the initial bootstrap goes through the manufacturer. In many cases there are even legal requirements (such as EU privacy laws) that prevent providing unnecessary information to third parties.

Common Global Resolution Infrastructure

The delegating manufacturer server model could be generalized into a reverse-DNS -like discovery infrastructure that could answer the question "this is device with identity ID, where is my home registration server?". However, at present no such resolution system exists. (Note: The EPCGlobal system for RFID resolution is reminiscent of this approach.)

6. Data Formats

A variety of data formats exist for passing around data. These data formats include XML, JSON, EXI, and text formats. Message lengths can have a significant effect on the amount of energy required for the communications, and such it is highly desirable to keep message lengths minimal. At the same time, extreme optimization can affect flexibility and ease of programming. The authors recommend [I-D.jennings-senml] as a compact, yet easily processed and extendable textual format.

7. Real-Time Reachable Devices

These devices are often best modeled as CoAP servers. The device will have limited control on when it receives messages, and it will have to listen actively for messages, up to the limits of the underlying link layer. If the device acts also in client role in some phase of its operation, it can control how many transmissions it makes on its own behalf.

The packet reception checks should be tailored according to the requirements of the application. If sub-second response time is not needed, a slightly more infrequent checking process may save some power.

For sensor-type devices, the CoAP OBSERVE extension [I-D.ietf-core-observe] may be supported. This allows the sensor to track changes to the sensed value, and make an immediate observation response upon a change. This may reduce the amount of polling needed to be done by the client. Unfortunately, it does not reduce the time that the device needs to be listening for requests. Subscription requests from other clients than the currently registered one may come at any time, the current client may change its request, and the device still needs to respond to normal queries as a server. As a result, the sensor can not rely having to communicate only on its own choice of observation interval.

In order to act as a server, the device needs to be placed in a public IPv4 address, be reachable over IPv6, or hosted in a private network. If the the device is hosted on a private network, then all other nodes need to access this device also need to reside in the same private network. There are multiple ways to provide private networks over public cellular networks. One approach is to dedicate a special Access Point Name or APN for the private network. Corporate access via cellular networks has often been arranged in this manner, for instance. Another approach is to use Virtual Private Networking (VPN) technology, for instance IPsec-based VPNs.

Power consumption from unwanted traffic is problematic in these devices, unless placed in a private network or protected by a operator-provided firewall service. Devices on an IPv6 network will have some protection through the nature of the 2^{64} address allocation for a single terminal in a 3GPP cellular network; the attackers will be unable to guess the full IP address of the device. However, this protects only the device from processing a packet, but since the network will still deliver the packet to any of the addresses within the assigned 64-bit prefix, packet reception costs are still incurred.

Note that the the VPN approach can not prevent unwanted traffic received at the tunnel endpoint address, and may require keep-alive traffic. Special APNs can solve this issue, but require explicit arrangement with the service provider.

8. Sleepy Devices

These devices are best modeled as devices that can delegate queries to some other node. For instance, as mirror proxy clients [I-D.vial-core-mirror-proxy]. When the device initializes itself, it makes a registration of itself in a mirror proxy as described above in Section 5 and then continues to send periodic updates of sensor values.

As a result, the device acts only as a client, not a server, and can shut down all communication channels while it is during its sleeping period. The length of the sleeping period depends on power and application requirements. Some environmental sensors might use a day or a week as the period, while other devices may use a smaller values ranging from minutes to hours.

Other approaches for delegation include CoAP-options described in [I-D.castellani-core-alive] [I-D.fossati-core-publish-monitor-options]. In this memo we use mirror proxies as an example, because of their ability to work with both HTTP and CoAP implementations; but the concepts are similar and the IETF work is still in progress so the final protocol details are yet to be decided.

The ability to shut down communications and act as only a client has four impacts:

- o Radio transmission and reception can be turned off during the sleeping period, reducing power consumption significantly.
- o However, some power and time is consumed by having to re-attach to the network after the end of a sleep period.
- o The window of opportunity for unwanted traffic to arrive is much smaller, as the device is listening for traffic only part of the time. Note that networks may cache packets for some time though. On the other hand, stateful firewalls can effectively remove much of unwanted traffic for client type devices.
- o The device may exist behind a NAT or a firewall without being impacted. Note that "Simple Security" basic IPv6 firewall capability [RFC6092] blocks inbound UDP traffic by default, so just moving to IPv6 is not direct solution to this problem.

For sleepy devices that represent actuators, it is also possible to use the mirror proxy model. The device can make periodic polls to the proxy to determine if a variable has changed.

8.1. Implementation Considerations

There are several challenges in implementing sleepy devices. They need hardware that can be put to an appropriate sleep mode but yet awakened when it is time to do something again. This is not always easy in all hardware platforms. It is important to be able to shut down as much of the hardware as possible, preferably down to everything else except a clock circuit. The platform also needs to support re-awakening at suitable time scales, as otherwise the device

needs to be powered up too frequently.

Most commercial cellular modem platforms do not allow applications to suspend the state of the communications stack. Hence, after a power-off period they need to re-establish communications, which takes some amount of time and extra energy.

Implementations should have a coordinated understanding of the state and sleeping schedule. For instance, it makes no sense to keep a CPU powered up, waiting for a message when the lower layer has been told that the next possible paging opportunity is some time away.

The cellular networks have a number of adjustable configuration parameters, such as the maximum used paging interval. Proper setting of these values has an impact on the power consumption of the device, but with the current business practices, such settings are rarely negotiated when the user's subscription is provisioned.

9. Security Considerations

There are no particular security aspects with what has been discussed in this memo, except for the ability to delegate queries for a resource to another node. Depending on how this is done, there are obvious security issues which have largely NOT yet been addressed in the relevant Internet Drafts [I-D.vial-core-mirror-proxy] [I-D.castellani-core-alive] [I-D.fossati-core-publish-monitor-options]. However, we point out that in general, security issues in delegation can be solved either through reliance on your local network support nodes (which may be quite reasonable in many environments) or explicit end-to-end security. Explicit end-to-end security through nodes that are awake at different times means in practice end-to-end data object security. We have implemented one such mechanism for sleepy nodes as described in [I-D.aks-crypto-sensors].

The security considerations relating to CoAP [I-D.ietf-core-coap] and the relevant link layers should apply. Note that cellular networks universally employ per-device authentication, integrity protection, and for most of the world, encryption of all their communications. Additional protection of transport sessions is possible through mechanisms described in [I-D.ietf-core-coap] or data objects.

10. IANA Considerations

There are no IANA impacts in this memo.

11. References

11.1. Normative References

- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.
- [I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank,
"Constrained Application Protocol (CoAP)",
draft-ietf-core-coap-13 (work in progress), December 2012.
- [I-D.ietf-core-observe]
Hartke, K., "Observing Resources in CoAP",
draft-ietf-core-observe-07 (work in progress),
October 2012.
- [I-D.vial-core-mirror-proxy]
Vial, M., "CoRE Mirror Server",
draft-vial-core-mirror-proxy-01 (work in progress),
July 2012.
- [I-D.shelby-core-resource-directory]
Shelby, Z., Krco, S., and C. Bormann, "CoRE Resource
Directory", draft-shelby-core-resource-directory-04 (work
in progress), July 2012.
- [I-D.jennings-senml]
Jennings, C., Shelby, Z., and J. Arkko, "Media Types for
Sensor Markup Language (SENML)", draft-jennings-senml-10
(work in progress), October 2012.

11.2. Informative References

- [RFC6092] Woodyatt, J., "Recommended Simple Security Capabilities in
Customer Premises Equipment (CPE) for Providing
Residential IPv6 Internet Service", RFC 6092,
January 2011.
- [I-D.arkko-core-sleepy-sensors]
Arkko, J., Rissanen, H., Loreto, S., Turanyi, Z., and O.
Novo, "Implementing Tiny COAP Sensors",
draft-arkko-core-sleepy-sensors-01 (work in progress),
July 2011.
- [I-D.aks-crypto-sensors]
Sethi, M., Arkko, J., Keranen, A., and H. Rissanen,
"Practical Considerations and Implementation Experiences

in Securing Smart Object Networks",
draft-aks-crypto-sensors-02 (work in progress),
March 2012.

[I-D.castellani-core-alive]

Castellani, A. and S. Loreto, "CoAP Alive Message",
draft-castellani-core-alive-00 (work in progress),
March 2012.

[I-D.fossati-core-publish-monitor-options]

Fossati, T., Giacomini, P., and S. Loreto, "Publish and
Monitor Options for CoAP",
draft-fossati-core-publish-monitor-options-01 (work in
progress), March 2012.

[Android-Bundle]

"Optimizing Downloads for Efficient Network Access",
Android developer note [http://developer.android.com/
training/efficient-downloads/
efficient-network-access.html](http://developer.android.com/training/efficient-downloads/efficient-network-access.html), February 2013.

Appendix A. Acknowledgments

The authors would like to thank Zach Shelby, Jan Holler, Salvatore Loreto, Matthew Vial, Thomas Fossati, Mohit Sethi, Jan Melen, Joachim Sachs, Heidi-Maria Rissanen, Sebastien Pierrel, Kumar Balachandran, Muhammad Waqas Mir, Cullen Jennings, Markus Isomaki, Hannes Tschofenig, and Anna Larmo for interesting discussions in this problem space.

Authors' Addresses

Jari Arkko
Ericsson
Jorvas 02420
Finland

Email: jari.arkko@piuha.net

Anders Eriksson
Ericsson
Stockholm 164 83
Sweden

Email: anders.e.eriksson@ericsson.com

Ari Keranen
Ericsson
Jorvas 02420
Finland

Email: ari.keranen@ericsson.com

LWIG Working Group
Internet-Draft
Intended status: Informational
Expires: September 27, 2012

A. Castellani
University of Padova
March 26, 2012

Learning CoAP separate responses by examples
draft-castellani-lwig-coap-separate-responses-00

Abstract

This draft aims at providing interesting examples of CoAP separate responses that are useful to aid CoAP implementers on understanding possible rare situation incurring.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 27, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Taxonomy of cases	3
2.1. Request lost	3
2.2. Request ACK lost	3
2.3. Response lost	4
2.4. Response ACK lost	4
3. Client perspective	5
3.1. No request ACK received	5
3.2. Response ACK lost	6
4. Server perspective	7
4.1. Request ACK lost	7
4.2. Response lost	7
5. Acknowledgements	9
6. Normative References	9
Author's Address	9

1. Introduction

To be done if interest is shown (TBDIIIS).

2. Taxonomy of cases

In the following sections all the possible situations are briefly described.

2.1. Request lost

```

C           S
| CON MID=0x1234 |
| PUT /increment |
|----->X      |

```

Figure 1: Example of request lost

As shown in Figure 1, this case includes all the situations where the request, including all its retransmissions, has never got through the network up to the server.

2.2. Request ACK lost

```

C           S
| CON MID=0x1234 |
| PUT /increment |
|----->        |
| ACK MID=0x1234 |
|   X<-----    |

```

Figure 2: Example of request ACK lost

As shown in Figure 2, this case includes all the situations where the request has got to the server, but the corresponding ACK has never got through the network up to the client.

2.3. Response lost

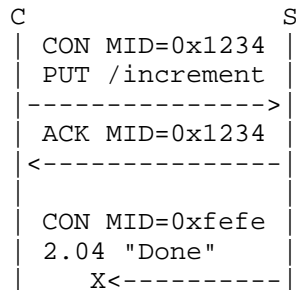


Figure 3: Example of response lost

As shown in Figure 3, this case includes all the situations where the response and its ACK have not been lost, but the corresponding separate response, including all its retransmissions, has never got through the network up to the client.

2.4. Response ACK lost

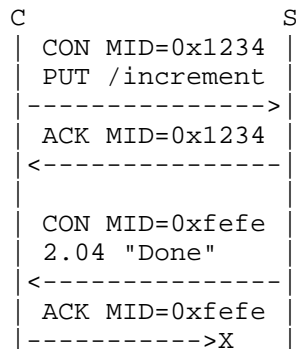


Figure 4: Example of response ACK lost

As shown in Figure 3, this case includes all the situations where the response, its ACK, and the corresponding separate response have not been lost by the network, but the last ACK sent by the client has been lost.

3. Client perspective

In this section interesting situations incurring to a client implementation are discussed.

3.1. No request ACK received

The client cannot distinguish between the first two cases request lost (see Section 2.1) and request ACK lost (see Section 2.1). We call this state C_NO_ACK.

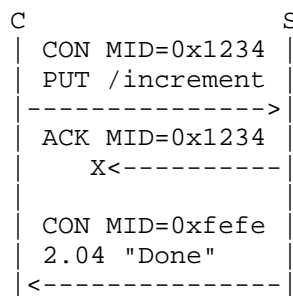


Figure 5: Response without ACK

Figure 5 shows an example where the client is in the C_NO_ACK state and it receives a CON response in the same session (i.e. matching [loc-host, loc-port, rem-host, rem-port, token]). A realistic but optimistic implementation might think that this response is related to the previous not acked request; a pessimistic but paranoid implementation could decide that the response is NOT related to the previous response.

Client implementations supporting only the empty Token (no Token support) are encouraged to randomly select local UDP source port at each new request; this implementation shrewdness smoothly resolves confusion.

Always having the Token Option set to a random value realistically resolves any possible confusion in this case, at the obvious cost of its added complexity in the client implementation and network overhead.

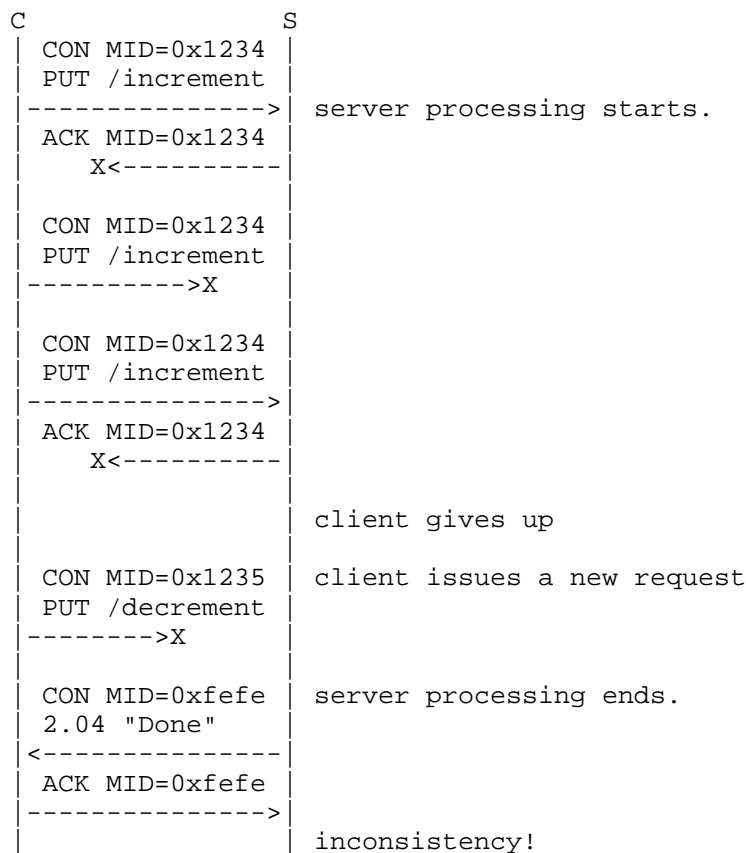


Figure 6: Naive client

Figure 6 shows an incident occurring to a naive client implementation using the empty Token and a static local UDP port. This leads to the indication that a client should in general avoid reusing the same session, i.e., [loc-host, loc-port, rem-host, rem-port, token], even if it has failed.

3.2. Response ACK lost

After sending the ACK to the response, a provident client implementation keeps the request session up for some time. This avoids issuing new requests in the same session (i.e. [loc-host, loc-port, rem-host, rem-port, token]), and allows smoothly responding with an empty ACK to response retransmissions received by the server.

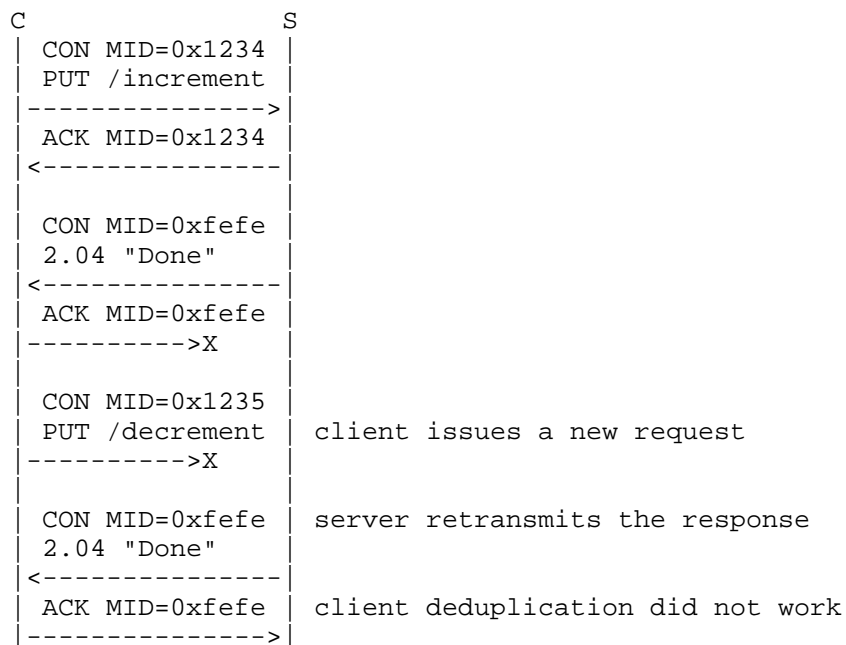


Figure 7: Inexperienced client

Figure 7 shows an incident occurring to an inexperienced client not having robust deduplication in place and reusing the same session.

4. Server perspective

TBDIIIS

4.1. Request ACK lost

TBDIIIS

4.2. Response lost

The server is said to be in S_NO_ACK when no empty ack to the response is received by the server.

```

C      S
|      |
| CON MID=0x1234 |
| PUT /increment  |
|----->|
| ACK MID=0x1234  |
|<-----|
|
| CON MID=0xfefe  |
| 2.04 "Done"     |
|<-----|
| ACK MID=0xfefe  |
|----->X
|
| CON MID=0xfefe  |
| 2.04 "Done"     |
|<-----|
| RST MID=0xfefe  |
|----->|

```

Figure 8: Forgetful client

Figure 8 shows the realistic case of a server in state S_NO_ACK, that deals with a client that do not maintains a successful session open after receiving the response. An optimistic server implementation might think that the client has received the response even if it has replied with a RST.

```

C      S
|      |
| CON MID=0x1234 |
| PUT /increment  |
|----->|
| ACK MID=0x1234  |
|<-----|
|
| CON MID=0xfefe  |
| 2.04 "Done"     |
|   X<-----|
|
| CON MID=0xfefe  |
| 2.04 "Done"     |
|<-----|
| RST MID=0xfefe  |
|----->|

```

client goes down for reboot

client is up again

Figure 9: Rebooting client

However as Figure 9 shows that the very same result might be happen if the server is interacting with a rebooting client.

Open issue: Should the server change its behaviour depending on the fact that it received a RST instead of an ACK? (e.g., Should it apply the actual change to the resource only after an ACK to the response has been received?)

5. Acknowledgements

Special thanks to Carsten Bormann for substantial contributions on this topic that significantly aided me in compiling this document, to Mattia Gheda for the long discussion had on this topic, and to Jeroen Hoebeke that publicly started discussion on this topic in the CoRE mailing list.

Thanks to Zach Shelby, Salvatore Loreto and Guido Moritz for helpful comments and discussions that have shaped the document.

6. Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank,
"Constrained Application Protocol (CoAP)",
draft-ietf-core-coap-09 (work in progress), March 2012.

Author's Address

Angelo P. Castellani
University of Padova
Via Gradenigo 6/B
Padova 35131
Italy

Email: angelo@castellani.net

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 22, 2013

Z. Cao
China Mobile
X. He
Hitachi (China) Research and
Development Corporation
M. Kovatsch
ETH Zurich
February 18, 2013

Energy Efficient Implementation of IETF Protocols on Constrained Devices
draft-hex-lwig-energy-efficient-00

Abstract

This document summarizes the problems and current practices of energy efficient protocol implementation on constrained devices, mostly about how to make the protocols within IETF scope behave energy friendly.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 22, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions used in this document	3
1.2. Terminology	3
2. Overview	4
3. MAC and Radio Duty Cycling	5
4. IP Adaption and Transport Layer	6
5. Routing Protocols	6
6. Application Layer	7
7. Cross Layer Optimization	7
8. Summary	7
9. IANA Considerations	8
10. Security Considerations	8
11. References	8
11.1. Normative References	8
11.2. Informative References	9
Authors' Addresses	9

1. Introduction

In many scenarios of embedded systems, the networked system is composed of many battery-powered devices. For example, in an environmental monitoring system or a temperature and humidity monitoring system in the data center, there are no always-on and handy sustained power supplies for the large number of small devices. In such deployment environments, it is necessary to optimize the energy consumption of the entire system, including computing, application layer behavior, and lower layer communication.

Various research efforts have been spent on this "energy efficiency" problem. Most of this research has focused on how to optimize the system's power consumption regarding a certain deployment scenario or how could an existing network function such as routing or security be more energy-efficient. Only few efforts were spent on energy-efficient designs for IETF protocols and standardized network stacks for such constrained devices [I-D.kovatsch-lwig-class1-coap].

The IETF has developed a suite of Internet protocols suitable for such small devices, including 6LoWPAN [RFC6282], 6LoWPAN-ND [RFC6775], RPL[RFC6550], and CoAP[I-D.ietf-core-coap]. This document tries to summarize the design considerations of making the IETF protocol suite as energy-efficient as possible. While this document does not provide detailed and systematic solutions to the energy efficiency problem, it summarizes the design efforts and analyzes the design space of this problem.

After reviewing the energy-efficient design of each layer, an overall conclusion is summarized. Though the lower layer communication optimization is the key part of energy efficient design, the protocol design at the network and application layers is also important to make the device battery-friendly.

1.1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]

1.2. Terminology

The terminologies used in this document can be referred to [I-D.bormann-lwig-terms].

2. Overview

The IETF has developed multiple protocols to enable end-to-end IP communication between constrained nodes and fully capable nodes. This work has witnessed the evolution of the traditional Internet protocol stack to a light-weight Internet protocol stack. As show in Figure 1 below, the IETF has developed CoAP as the application layer and 6LoWPAN as the adaption layer to run IPv6 over IEEE 802.15.4 and Bluetooth Low-Energy, with the support of routing by RPL and efficient neighbor discovery by 6LoWPAN-ND.

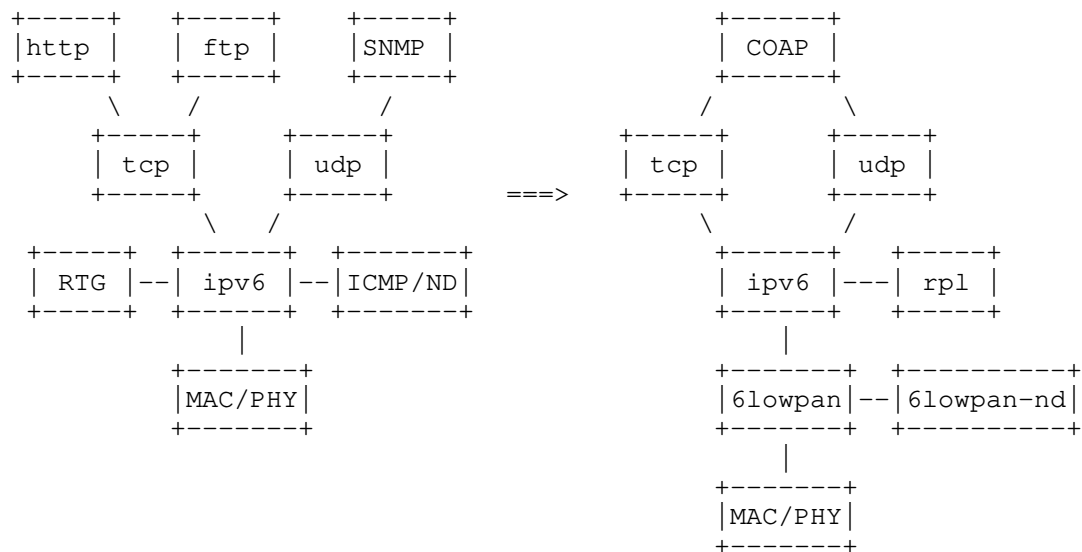


Figure 1: Traditional and Lightweight Internet Protocol Stack

There are comprehensive measurements of wireless communication [Powertrace]. Below we list the energy consumption profile of the most common atom operations on a prevalent sensor node platform. The measurement was based on the Tmote Sky with ContikiMAC as the radio duty cycling algorithm. From the measurement, we can see that optimized transmissions and reception consume almost the same amount of energy. For IEEE 802.15.4 and UWB radios, transmitting is actually even cheaper than receiving. Only for broadcast and non-synchronized communication transmissions become costly in terms of energy because they need to flood the medium for a long time.

Activity	Energy (uJ)
Broadcast reception	178
Unicast reception	222
Broadcast transmission	1790
Non-synchronized unicast transmission	1090
Synchronized unicast transmission	120
Unicast TX to awake receiver	96

Figure 2: Power consumption of atom operations on the Tmote Sky with ContikiMAC

3. MAC and Radio Duty Cycling

In low-power wireless networks, communication and power consumption are intertwined. The communication device is typically the most power-consuming component, but merely refraining from transmissions is not enough to attain a low power consumption: the radio consumes as much power in listen mode as when actively transmitting, as show in Figure 2 . To reduce power consumption, the radio must be switched completely off -- duty-cycled -- as much as possible. ContikiMAC is a very typical Radio Duty Cycling protocol [ContikiMAC].

From the perspective of MAC&RDC, all upper layer protocols, such as routing, RESTful communication, adaption, and management flows, are all applications. Since the duty cycling algorithm is the key to energy-efficiency of the wireless medium, it synchronizes the TX/RX request from the higher layer.

The MAC&RDC are not in the scope of the IETF, yet lower layer designers and chipset manufactures take great care of the problem. For the IETF protocol designers, however, it is good to know the behaviors of lower layers so that the designed protocols can work perfectly with them.

Once again, the IETF protocols we are going to talk about in the following sections are the customers of the lower layer. If they want to get better service in a cooperative way, they should be considerative and understand each other.

4. IP Adaption and Transport Layer

6LoWPAN is the adaption layer to run IPv6 over IEEE 802.15.4 MAC&PHY. It was born to fill the gap that the IPv6 layer does not support fragmentation and assembly of <1280-byte packets while IEEE 802.15.4 only supports a MTU of 127 bytes.

IPv6 is the basis for the higher layer protocols, including both TCP/UDP transport and applications. So they are quite ignorant of the transmission and reception behaviors, and are almost neutral to the energy-efficiency problem.

What the network stack can optimize is to save the computing power. For example the Contiki implementation has multiple cross layer optimizations for buffers and energy management, e.g., the computing and validation of UDP/TCP checksums without the need of reading IP headers from a different layer. These optimizations are software implementation techniques, and out of the scope of IETF and the LWIG working group.

The 6LoWPAN contributes to the energy-efficiency problem in two ways. First of all, it swaps computing with communication. 6LoWPAN applies compression of the IPv6 header. This means less amount of data will be handled by the lower layer, but both the sender and receiver should spend more computing power on the compression and decompression of the packets over the air. Secondly, the 6LoWPAN working group developed the energy-efficient Neighbor Discovery called 6LoWPAN-ND, which is an energy efficient replacement of the IPv6 ND in constrained environments. IPv6 Neighbor Discovery was not designed for non-transitive wireless links, as its heavy use of multicast makes it inefficient and sometimes impractical in a low-power and lossy network. 6LoWPAN-ND describes simple optimizations to IPv6 Neighbor Discovery, its addressing mechanisms, and duplicate address detection for Low-power Wireless Personal Area Networks and similar networks.

5. Routing Protocols

The routing protocol designed by the IETF for constrained environments is called RPL [RFC6550]. As a routing protocol, RPL has to exchange messages periodically and keep routing states for each destination. RPL is optimized for the many-to-one communication pattern, where network nodes primarily send data towards the border router, but has provisions for any-to-any routing as well.

The authors of the Powertrace tool studied the power profile of RPL. It divides the routing protocol into control and data traffic. The

control channel uses ICMP messages to establish and maintain the routing states. The data channel is any application that uses RPL for routing packets. The study has shown that the power consumption of the control traffic goes down over time and data traffic stays relatively constant. The study also reflects that the routing protocol should keep the control traffic as low as possible to make it energy-friendly.

6. Application Layer

CoAP [I-D.ietf-core-coap] was designed as a RESTful application protocol, connecting the services of smart devices to the World Wide Web. CoAP is not a chatty protocol, it provides basic communication services such as service discovery and GET/POST/PUT/DELETE methods with a binary header.

The energy-efficient design is implicitly included in the CoAP protocol design. To reduce regular and frequent queries of the resources, CoAP provides an observe mode, in which the requester registers its interest of a certain resource and the responder will report the value whenever it was updated. This reduces the request response roundtrip while keeping information exchange a ubiquitous service.

7. Cross Layer Optimization

The cross layer optimization is a technique used in many scenarios. There are some technologies for power efficient optimization via PHY to Routing cross layer design [Cross-layer-Optimization]. In this research, cross-layer optimization frameworks have been developed to minimize the total power consumption or to maximize the utility-power tradeoff using cooperative diversity.

Also a cross-layer design in multihop wireless networks is proposed for congestion control, routing and scheduling--in transport, network and link layers into a coherent framework [Cross-layer-design]. This method and thinking could be applied to the implementation of energy effective cross layer design.

8. Summary

We find a summary section necessary although most IETF documents do not contain it. The points we would like to summarize are as follows.

- a. All Internet protocols, which are in the scope of the IETF, are customers of the lower layers (PHY, MAC, and Duty-cycling). In order to get a better service, the designers of higher layers should know them better.
- b. The IETF has developed multiple protocols for constrained networked devices. A lot of implicitly included design principles have been used in these protocols.
- c. The power trace analysis of different protocol operations showed that for radio-duty-cycled networks broadcasts should be avoided. Saving unnecessary states maintenance is also an effective method to be energy-friendly.

9. IANA Considerations

This document has no IANA requests.

10. Security Considerations

This document discusses the energy efficient protocol design, and does not incur any changes or challenges on security issues besides what the protocol specifications have analyzed.

11. References

11.1. Normative References

[Announcementlayer]

Dunkels, A., "The Announcement Layer: Beacon Coordination for the Sensornet Stack. In Proceedings of EWSN 2011".

[ContikiMAC]

Dunkels, A., "The ContikiMAC Radio Duty Cycling Protocol, SICS Technical Report T2011:13", December 2011.

[Cross-layer-Optimization]

Le and Hossain, "Cross-Layer Optimization Frameworks for Multihop Wireless Networks Using Cooperative Diversity", July 2008.

[Cross-layer-design]

Chen, Low, and Doyle, "Cross-layer design in multihop wireless networks", 2011.

[I-D.bormann-lwig-terms]

Bormann, C. and M. Ersue, "Terminology for Constrained Node Networks", draft-bormann-lwig-terms-00 (work in progress), November 2012.

[I-D.ietf-core-coap]

Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-13 (work in progress), December 2012.

[I-D.kovatsch-lwig-class1-coap]

Kovatsch, M., "Implementing CoAP for Class 1 Devices", draft-kovatsch-lwig-class1-coap-00 (work in progress), October 2012.

[Powertrace]

Dunkels, Eriksson, Finne, and Tsiftes, "Powertrace: Network-level Power Profiling for Low-power Wireless Networks", March 2011.

11.2. Informative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.
- [RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

Authors' Addresses

Zhen Cao (Ed.)
China Mobile
Xuanwumenxi Ave. No.32
Beijing, 100871
P.R.China

Phone:
Email: zehn.cao@gmail.com, caozhen@chinamobile.com

Xuan He
Hitachi (China) Research and Development Corporation
301, Tower C North, Raycom, 2 Kexuyuan Nanlu, Haidian District
Beijing, 100190
P.R.China

Phone:
Fax:
Email: xhe@hitachi.cn
URI:

Matthias Kovatsch
ETH Zurich
Universitaetstrasse 6
Zurich, CH-8092
Switzerland

Phone:
Fax:
Email: kovatsch@inf.ethz.ch
URI:

LWIG Working Group
Internet-Draft
Intended status: Informational
Expires: August 14, 2014

C. Bormann
Universitaet Bremen TZI
M. Ersue
Nokia Siemens Networks
A. Keranen
Ericsson
February 10, 2014

Terminology for Constrained Node Networks
draft-ietf-lwig-terminology-07

Abstract

The Internet Protocol Suite is increasingly used on small devices with severe constraints on power, memory and processing resources, creating constrained node networks. This document provides a number of basic terms that have turned out to be useful in the standardization work for constrained node networks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 14, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Core Terminology	3
2.1. Constrained Nodes	4
2.2. Constrained Networks	5
2.2.1. Challenged Networks	6
2.3. Constrained Node Networks	6
2.3.1. LLN ("low-power lossy network")	7
2.3.2. LoWPAN, 6LoWPAN	7
3. Classes of Constrained Devices	8
4. Power Terminology	10
4.1. Scaling Properties	10
4.2. Classes of Energy Limitation	10
4.3. Strategies of Using Power for Communication	11
5. Security Considerations	13
6. IANA Considerations	13
7. Acknowledgements	13
8. Informative References	14
Authors' Addresses	16

1. Introduction

Small devices with limited CPU, memory, and power resources, so called constrained devices (often used as a sensor/actuator, a smart object, or a smart device) can form a network, becoming "constrained nodes" in that network. Such a network may itself exhibit constraints, e.g. with unreliable or lossy channels, limited and unpredictable bandwidth, and a highly dynamic topology.

Constrained devices might be in charge of gathering information in diverse settings including natural ecosystems, buildings, and factories and sending the information to one or more server stations. They also act on information, by performing some physical action, including displaying it. Constrained devices may work under severe resource constraints such as limited battery and computing power, little memory, as well as insufficient wireless bandwidth and ability to communicate; these constraints often exacerbate each other. Other entities on the network, e.g., a base station or controlling server, might have more computational and communication resources and could support the interaction between the constrained devices and applications in more traditional networks.

Today diverse sizes of constrained devices with different resources and capabilities are becoming connected. Mobile personal gadgets, building-automation devices, cellular phones, Machine-to-machine (M2M) devices, etc. benefit from interacting with other "things" nearby or somewhere in the Internet. With this, the Internet of Things (IoT) becomes a reality, built up out of uniquely identifiable and addressable objects (things). And over the next decade, this could grow to large numbers [fifty-billion] of Internet-connected constrained devices, greatly increasing the Internet's size and scope.

The present document provides a number of basic terms that have turned out to be useful in the standardization work for constrained environments. The intention is not to exhaustively cover the field, but to make sure a few core terms are used consistently between different groups cooperating in this space.

In this document, the term "byte" is used in its now customary sense as a synonym for "octet". Where sizes of semiconductor memory are given, the prefix "kibi" (1024) is combined with "byte" to "kibibyte", abbreviated "KiB", for 1024 bytes [ISQ-13].

In computing, the term "power" is often used for the concept of "computing power" or "processing power", as in CPU performance. Unless explicitly stated otherwise, in this document the term stands for electrical power. "Mains-powered" is used as a short-hand for being permanently connected to a stable electrical power grid.

2. Core Terminology

There are two important aspects to `_scaling_` within the Internet of Things:

- o Scaling up Internet technologies to a large number [fifty-billion] of inexpensive nodes, while
- o scaling down the characteristics of each of these nodes and of the networks being built out of them, to make this scaling up economically and physically viable.

The need for scaling down the characteristics of nodes leads to `_constrained nodes_`.

2.1. Constrained Nodes

The term "constrained node" is best defined by contrasting the characteristics of a constrained node with certain widely held expectations on more familiar Internet nodes:

Constrained Node: A node where some of the characteristics that are otherwise pretty much taken for granted for Internet nodes at the time of writing are not attainable, often due to cost constraints and/or physical constraints on characteristics such as size, weight, and available power and energy. The tight limits on power, memory and processing resources lead to hard upper bounds on state, code space and processing cycles, making optimization of energy and network bandwidth usage a dominating consideration in all design requirements. Also, some layer 2 services such as full connectivity and broadcast/multicast may be lacking.

While this is not a rigorous definition, it is grounded in the state of the art and clearly sets apart constrained nodes from server systems, desktop or laptop computers, powerful mobile devices such as smartphones etc. There may be many design considerations that lead to these constraints, including cost, size, weight, and other scaling factors.

(An alternative name, when the properties as a network node are not in focus, is "constrained device".)

There are multiple facets to the constraints on nodes, often applying in combination, e.g.:

- o constraints on the maximum code complexity (ROM/Flash);
- o constraints on the size of state and buffers (RAM);
- o constraints on the amount of computation feasible in a period of time ("processing power");
- o constraints on the available (electrical) power;
- o constraints on user interface and accessibility in deployment (ability to set keys, update software, etc.).

Section 3 defines a small number of interesting classes ("class-N" for N=0,1,2) of constrained nodes focusing on relevant combinations of the first two constraints. With respect to available (electrical) power, [RFC6606] distinguishes "power-affluent" nodes (mains-powered or regularly recharged) from "power-constrained nodes" that draw their power from primary batteries or by using energy harvesting; more detailed power terminology is given in Section 4.

The use of constrained nodes in networks often also leads to constraints on the networks themselves. However, there may also be constraints on networks that are largely independent from those of the nodes. We therefore distinguish `_constrained networks_` and `_constrained node networks_`.

2.2. Constrained Networks

We define "constrained network" in a similar way:

Constrained Network: A network where some of the characteristics pretty much taken for granted with link layers in common use in the Internet at the time of writing, are not attainable.

Constraints may include:

- o low achievable bit rate/throughput (including limits on duty cycle),
- o high packet loss, high packet loss (delivery rate) variability,
- o highly asymmetric link characteristics,
- o severe penalties for using larger packets (e.g., high packet loss due to link layer fragmentation),
- o limits on reachability over time (a substantial number of devices may power off at any point in time but periodically "wake up" and can communicate for brief periods of time)
- o lack of (or severe constraints on) advanced services such as IP multicast.

More generally, we speak of constrained networks whenever at least some of the nodes involved in the network exhibit these characteristics.

Again, there may be several reasons for this:

- o cost constraints on the network,

- o constraints of the nodes (for constrained node networks),
- o physical constraints (e.g., power constraints, environmental constraints, media constraints such as underwater operation, limited spectrum for very high density, electromagnetic compatibility),
- o regulatory constraints, such as very limited spectrum availability (including limits on effective radiated power and duty cycle), or explosion safety,
- o technology constraints, such as older and lower speed technologies that are still operational and may need to stay in use for some more time.

2.2.1. Challenged Networks

A constrained network is not necessarily a `_challenged_` network [FALL]:

Challenged Network: A network that has serious trouble maintaining what an application would today expect of the end-to-end IP model, e.g., by:

- o not being able to offer end-to-end IP connectivity at all;
- o exhibiting serious interruptions in end-to-end IP connectivity;
- o exhibiting delay well beyond the Maximum Segment Lifetime (MSL) defined by TCP [RFC0793].

All challenged networks are constrained networks in some sense, but not all constrained networks are challenged networks. There is no well-defined boundary between the two, though. Delay-Tolerant Networking (DTN) has been designed to cope with challenged networks [RFC4838].

2.3. Constrained Node Networks

Constrained Node Network: A network whose characteristics are influenced by being composed of a significant portion of constrained nodes.

A constrained node network always is a constrained network because of the network constraints stemming from the node constraints, but may also have other constraints that already make it a constrained network.

The rest of this subsection introduces two additional terms that are in active use in the area of constrained node networks, without an intent to define them: LLN and (6)LoWPAN.

2.3.1. LLN ("low-power lossy network")

A related term that has been used to describe the focus of the IETF working group on Routing Over Low power and Lossy networks (ROLL) is "low-power lossy network" (LLN). The ROLL terminology document [RFC7102] defines LLNs as follows:

LLN: Low power and Lossy networks (LLNs) are typically composed of many embedded devices with limited power, memory, and processing resources interconnected by a variety of links, such as IEEE 802.15.4 or Low Power WiFi. There is a wide scope of application areas for LLNs, including industrial monitoring, building automation (HVAC, lighting, access control, fire), connected home, healthcare, environmental monitoring, urban sensor networks, energy management, assets tracking and refrigeration.. [sic]

Beyond that, LLNs often exhibit considerable loss at the physical layer, with significant variability of the delivery rate, and some short-term unreliability, coupled with some medium term stability that makes it worthwhile to construct medium-term stable directed acyclic graphs for routing and do measurements on the edges such as ETX [RFC6551]. Not all LLNs comprise low power nodes [I-D.hui-vasseur-roll-rpl-deployment].

LLNs typically are composed of constrained nodes; this leads to the design of operation modes such as the "non-storing mode" defined by RPL (the IPv6 Routing Protocol for Low-Power and Lossy Networks [RFC6650]). So, in the terminology of the present document, an LLN is a constrained node network with certain network characteristics, which include constraints on the network as well.

2.3.2. LoWPAN, 6LoWPAN

One interesting class of a constrained network often used as a constrained node network is the "LoWPAN" [RFC4919], a term inspired from the name of the IEEE 802.15.4 working group (low-rate wireless personal area networks (LR-WPANs)). The expansion of that acronym, "Low-Power Wireless Personal Area Network" contains a hard to justify "Personal" that is due to the history of task group naming in IEEE 802 more than due to an orientation of LoWPANs around a single person. Actually, LoWPANs have been suggested for urban monitoring, control of large buildings, and industrial control applications, so the "Personal" can only be considered a vestige. Occasionally the term is read as "Low-Power Wireless Area Networks" (LoWPANs) [WEI].

Originally focused on IEEE 802.15.4, "LoWPAN" (or when used for IPv6, "6LoWPAN") also refers to networks built from similarly constrained link layer technologies [I-D.ietf-6lowpan-btle] [I-D.mariager-6lowpan-v6over-dect-ule] [I-D.brandt-6man-lowpanz].

3. Classes of Constrained Devices

Despite the overwhelming variety of Internet-connected devices that can be envisioned, it may be worthwhile to have some succinct terminology for different classes of constrained devices. In this document, the class designations in Table 1 may be used as rough indications of device capabilities:

Name	data size (e.g., RAM)	code size (e.g., Flash)
Class 0, C0	<< 10 KiB	<< 100 KiB
Class 1, C1	~ 10 KiB	~ 100 KiB
Class 2, C2	~ 50 KiB	~ 250 KiB

Table 1: Classes of Constrained Devices (KiB = 1024 bytes)

As of the writing of this document, these characteristics correspond to distinguishable clusters of commercially available chips and design cores for constrained devices. While it is expected that the boundaries of these classes will move over time, Moore's law tends to be less effective in the embedded space than in personal computing devices: Gains made available by increases in transistor count and density are more likely to be invested in reductions of cost and power requirements than into continual increases in computing power.

Class 0 devices are very constrained sensor-like motes. They are so severely constrained in memory and processing capabilities that most likely they will not have the resources required to communicate directly with the Internet in a secure manner (rare heroic, narrowly targeted implementation efforts notwithstanding). Class 0 devices will participate in Internet communications with the help of larger devices acting as proxies, gateways or servers. Class 0 devices generally cannot be secured or managed comprehensively in the traditional sense. They will most likely be preconfigured (and will be reconfigured rarely, if at all), with a very small data set. For management purposes, they could answer keepalive signals and send on/off or basic health indications.

Class 1 devices are quite constrained in code space and processing capabilities, such that they cannot easily talk to other Internet nodes employing a full protocol stack such as using HTTP, TLS and related security protocols and XML-based data representations. However, they have enough power to use a protocol stack specifically designed for constrained nodes (such as CoAP over UDP [I-D.ietf-core-coap]) and participate in meaningful conversations without the help of a gateway node. In particular, they can provide support for the security functions required on a large network. Therefore, they can be integrated as fully developed peers into an IP network, but they need to be parsimonious with state memory, code space, and often power expenditure for protocol and application usage.

Class 2 devices are less constrained and fundamentally capable of supporting most of the same protocol stacks as used on notebooks or servers. However, even these devices can benefit from lightweight and energy-efficient protocols and from consuming less bandwidth. Furthermore, using fewer resources for networking leaves more resources available to applications. Thus, using the protocol stacks defined for more constrained devices also on Class 2 devices might reduce development costs and increase the interoperability.

Constrained devices with capabilities significantly beyond Class 2 devices exist. They are less demanding from a standards development point of view as they can largely use existing protocols unchanged. The present document therefore does not make any attempt to define classes beyond Class 2. These devices can still be constrained by a limited energy supply.

With respect to examining the capabilities of constrained nodes, particularly for Class 1 devices, it is important to understand what type of applications they are able to run and which protocol mechanisms would be most suitable. Because of memory and other limitations, each specific Class 1 device might be able to support only a few selected functions needed for its intended operation. In other words, the set of functions that can actually be supported is not static per device type: devices with similar constraints might choose to support different functions. Even though Class 2 devices have some more functionality available and may be able to provide a more complete set of functions, they still need to be assessed for the type of applications they will be running and the protocol functions they would need. To be able to derive any requirements, the use cases and the involvement of the devices in the application and the operational scenario need to be analyzed. Use cases may combine constrained devices of multiple classes as well as more traditional Internet nodes.

4. Power Terminology

Devices not only differ in their computing capabilities, but also in available electrical power and/or energy. While it is harder to find recognizable clusters in this space, it is still useful to introduce some common terminology.

4.1. Scaling Properties

The power and/or energy available to a device may vastly differ, from kilowatts to microwatts, from essentially unlimited to hundreds of microjoules.

Instead of defining classes or clusters, we simply state, in SI units, an approximate value for one or both of the quantities listed in Table 2:

Name	Definition	SI Unit
Ps	Sustainable average power available for the device over the time it is functioning	W (Watt)
Et	Total electrical energy available before the energy source is exhausted	J (Joule)

Table 2: Quantities Relevant to Power and Energy

The value of Et may need to be interpreted in conjunction with an indication over which period of time the value is given; see the next subsection.

Some devices enter a "low-power" mode before the energy available in a period is exhausted, or even have multiple such steps on the way to exhaustion. For these devices, Ps would need to be given for each of the modes/steps.

4.2. Classes of Energy Limitation

As discussed above, some devices are limited in available energy as opposed to (or in addition to) being limited in available power. Where no relevant limitations exist with respect to energy, the device is classified as E9. The energy limitation may be in total energy available in the usable lifetime of the device (e.g. a device with a non-replaceable primary battery, which is discarded when this battery is exhausted), classified as E2. Where the relevant limitation is for a specific period, this is classified as E1, e.g. a

limited amount of energy available for the night with a solar-powered device, or for the period between recharges with a device that is manually connected to a charger, or by a periodic (primary) battery replacement interval. Finally, there may be a limited amount of energy available for a specific event, e.g. for a button press in an energy harvesting light switch; this is classified as E0. Note that many E1 devices in a sense also are E2, as the rechargeable battery has a limited number of useful recharging cycles.

In summary, we distinguish (Table 3):

Name	Type of energy limitation	Example Power Source
E0	Event energy-limited	Event-based harvesting
E1	Period energy-limited	Battery that is periodically recharged or replaced
E2	Lifetime energy-limited	Non-replaceable primary battery
E9	No direct quantitative limitations to available energy	Mains powered

Table 3: Classes of Energy Limitation

4.3. Strategies of Using Power for Communication

Especially when wireless transmission is used, the radio often consumes a big portion of the total energy consumed by the device. Design parameters such as the available spectrum, the desired range, and the bitrate aimed for, influence the power consumed during transmission and reception; the duration of transmission and reception (including potential reception) influence the total energy consumption.

Based on the type of the energy source (e.g., battery or mains power) and how often device needs to communicate, it may use different kinds of strategies for power usage and network attachment.

The general strategies for power usage can be described as follows:

Always-on: This strategy is most applicable if there is no reason for extreme measures for power saving. The device can stay on in

the usual manner all the time. It may be useful to employ power-friendly hardware or limit the number of wireless transmissions, CPU speeds, and other aspects for general power saving and cooling needs, but the device can be connected to the network all the time.

Normally-off: Under this strategy, the device sleeps such long periods at a time that once it wakes up, it makes sense for it to not pretend that it has been connected to the network during sleep: The device re-attaches to the network as it is woken up. The main optimization goal is to minimize the effort during such re-attachment process and any resulting application communications.

If the device sleeps for long periods of time, and needs to communicate infrequently, the relative increase in energy expenditure during reattachment may be acceptable.

Low-power: This strategy is most applicable to devices that need to operate on a very small amount of power, but still need to be able to communicate on a relatively frequent basis. This implies that extremely low power solutions needs to be used for the hardware, chosen link layer mechanisms, and so on. Typically, given the small amount of time between transmissions, despite their sleep state these devices retain some form of network attachment to the network. Techniques used for minimizing power usage for the network communications include minimizing any work from re-establishing communications after waking up, tuning the frequency of communications (including "duty cycling", where components are switched on and off in a regular cycle), and other parameters appropriately.

In summary, we distinguish (Table 4):

Name	Strategy	Ability to communicate
P0	Normally-off	Re-attach when required
P1	Low-power	Appears connected, perhaps with high latency
P9	Always-on	Always connected

Table 4: Strategies of Using Power for Communication

Note that the discussion above is at the device level; similar considerations can apply at the communications interface level. This document does not define terminology for the latter.

A term often used to describe power-saving approaches is "duty-cycling". This describes all forms of periodically switching off some function, leaving it on only for a certain percentage of time (the "duty cycle").

[RFC7102] only distinguishes two levels, defining a Non-sleepy Node as a node that always remains in a fully powered on state (always awake) where it has the capability to perform communication (P9), and a Sleepy Node as a node that may sometimes go into a sleep mode (a low power state to conserve power) and temporarily suspend protocol communication (P0); there is no explicit mention of P1.

5. Security Considerations

This document introduces common terminology that does not raise any new security issue. Security considerations arising from the constraints discussed in this document need to be discussed in the context of specific protocols. For instance, [I-D.ietf-core-coap] section 11.6, "Constrained node considerations", discusses implications of specific constraints on the security mechanisms employed. [I-D.ietf-roll-security-threats] provides a security threat analysis for the RPL routing protocol. Implementation considerations for security protocols on constrained nodes are discussed in [I-D.ietf-lwig-ikev2-minimal] and [I-D.ietf-lwig-tls-minimal]. A wider view at security in constrained node networks is provided in [I-D.garcia-core-security].

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

Dominique Barthel and Peter van der Stok provided useful comments; Charles Palmer provided a full editorial review.

Peter van der Stok insisted that we should have power terminology, hence Section 4. The text for Section 4.3 is mostly lifted from a previous version of [I-D.ietf-lwig-cellular] and has been adapted for this document.

8. Informative References

- [FALL] Fall, K., "A Delay-Tolerant Network Architecture for Challenged Internets", SIGCOMM 2003, 2003.
- [I-D.brandt-6man-lowpanz]
Brandt, A. and J. Buron, "Transmission of IPv6 packets over ITU-T G.9959 Networks", draft-brandt-6man-lowpanz-02 (work in progress), June 2013.
- [I-D.garcia-core-security]
Garcia-Morchon, O., Kumar, S., Keoh, S., Hummen, R., and R. Struik, "Security Considerations in the IP-based Internet of Things", draft-garcia-core-security-06 (work in progress), September 2013.
- [I-D.hui-vasseur-roll-rpl-deployment]
Vasseur, J., Hui, J., Dasgupta, S., and G. Yoon, "RPL deployment experience in large scale networks", draft-hui-vasseur-roll-rpl-deployment-01 (work in progress), July 2012.
- [I-D.ietf-6lowpan-btle]
Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "Transmission of IPv6 Packets over BLUETOOTH Low Energy", draft-ietf-6lowpan-btle-12 (work in progress), February 2013.
- [I-D.ietf-core-coap]
Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.
- [I-D.ietf-lwig-cellular]
Arkko, J., Eriksson, A., and A. Keranen, "Building Power-Efficient CoAP Devices for Cellular Networks", draft-ietf-lwig-cellular-00 (work in progress), August 2013.
- [I-D.ietf-lwig-ikev2-minimal]
Kivinen, T., "Minimal IKEv2", draft-ietf-lwig-ikev2-minimal-01 (work in progress), October 2013.
- [I-D.ietf-lwig-tls-minimal]
Kumar, S., Keoh, S., and H. Tschofenig, "A Hitchhiker's Guide to the (Datagram) Transport Layer Security Protocol for Smart Objects and Constrained Node Networks", draft-ietf-lwig-tls-minimal-00 (work in progress), September 2013.

- [I-D.ietf-roll-security-threats]
Tsao, T., Alexander, R., Dohler, M., Daza, V., Lozano, A.,
and M. Richardson, "A Security Threat Analysis for Routing
Protocol for Low-power and lossy networks (RPL)", draft-
ietf-roll-security-threats-06 (work in progress), December
2013.
- [I-D.mariager-6lowpan-v6over-dect-ule]
Mariager, P., Petersen, J., and Z. Shelby, "Transmission
of IPv6 Packets over DECT Ultra Low Energy", draft-
mariager-6lowpan-v6over-dect-ule-03 (work in progress),
July 2013.
- [ISQ-13] International Electrotechnical Commission, "International
Standard -- Quantities and units -- Part 13: Information
science and technology", IEC 80000-13, March 2008.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC
793, September 1981.
- [RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst,
R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant
Networking Architecture", RFC 4838, April 2007.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6
over Low-Power Wireless Personal Area Networks (6LoWPANs):
Overview, Assumptions, Problem Statement, and Goals", RFC
4919, August 2007.
- [RFC6551] Vasseur, JP., Kim, M., Pister, K., Dejean, N., and D.
Barthel, "Routing Metrics Used for Path Calculation in
Low-Power and Lossy Networks", RFC 6551, March 2012.
- [RFC6606] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem
Statement and Requirements for IPv6 over Low-Power
Wireless Personal Area Network (6LoWPAN) Routing", RFC
6606, May 2012.
- [RFC6650] Falk, J. and M. Kucherawy, "Creation and Use of Email
Feedback Reports: An Applicability Statement for the Abuse
Reporting Format (ARF)", RFC 6650, June 2012.
- [RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and
Lossy Networks", RFC 7102, January 2014.
- [WEI] Shelby, Z. and C. Bormann, "6LoWPAN: the Wireless Embedded
Internet", ISBN 9780470747995, 2009.

[fifty-billion]

Ericsson, "More Than 50 Billion Connected Devices",
Ericsson White Paper 284 23-3149 Uen, February 2011,
<[http://www.ericsson.com/res/docs/whitepapers/
wp-50-billions.pdf](http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf)>.

Authors' Addresses

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Mehmet Ersue
Nokia Siemens Networks
St.-Martinstrasse 76
81541 Munich
Germany

Phone: +49 172 8432301
Email: mehmet.ersue@nsn.com

Ari Keranen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: ari.keranen@ericsson.com

LWIG Working Group
Internet-Draft
Intended Status: Informational
Expires: August 29, 2013

S. Keoh
S. Kumar
O. Garcia-Morchon
Philips Research
February 25, 2013

Securing the IP-based Internet of Things with DTLS
draft-keoh-lwig-dtls-iot-01

Abstract

The IP-based Internet of Things (IoT) refers to the pervasive interaction of smart devices and people enabling new applications by means of IP protocols. Traditional IP protocols will be further complemented by 6LoWPAN and CoAP to make the IoT feasible on small devices. Security and privacy are a must for such an environment. Due to mobility, limited bandwidth, resource constraints, and new communication topologies, existing security solutions need to be adapted. We propose a security architecture for the IoT in order to provide network access control to smart devices, the management of keys and securing unicast/multicast communication. Devices are authenticated and granted network access by means of a pre-shared key (PSK) based security handshake protocol. The solution is based on Datagram Transport Layer Security (DTLS). Through the established secure channels, keying materials, operational and security parameters are distributed, enabling devices to derive session keys and group keys. The solution relies on the DTLS Record Layer for the protection of unicast and multicast data flows. We have prototyped and evaluated the security architecture. The DTLS architecture allows for easier interaction and interoperability with the Internet due to the extensive use of TLS. However, it exhibits performance issues constraining its deployment in some network topologies and hence would require further optimizations.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	4
1.1	Terminology	5
2.	Related Work and Background	6
3	Use Cases & Problem Statement	7
3.1	Problem Statement and Requirements	8
3.2	Threat model, Security Goals & Assumptions	8
4	Design	10
4.1	Overview	10
4.2	Secure Network Access	11
4.3	Key Management	12
4.3.1	Management of unicast keys	12
4.3.2	Management of multicast keys	13
4.4	Secure Uni- and Multicast Communication	14
4.4.1	Unicast Communication	14
4.4.2	Multicast Communication	14
5	Implementation and Evaluation	14
5.1	Prototype Implementation	14
5.2	Memory Consumption	15
5.3	Communication Overhead	16

5.4 Message Delay, Success Rate and Bandwidth	16
6. Conclusions and future work	18
7 Security Considerations	18
8 IANA Considerations	18
9. Acknowledgements	18
10 References	18
10.1 Normative References	18
9.2 Informative References	19
Authors' Addresses	20

1 Introduction

The IP-based Internet of Things (IoT) will enable smart and mobile devices equipped with sensing, acting, and wireless communication capabilities to interact and cooperate with each other in a pervasive way by means of IP connectivity. IP protocols play a key role in this vision since they allow for end-to-end connectivity using standard protocols ensuring that different smart devices can easily communicate with each other in an inexpensive way. Protocols such as IPv6, TCP and HTTP that are commonly used in traditional networks will be complemented by IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) and Constrained Application Protocol (CoAP) currently in development in IETF.

This allows smart and mobile devices used for various applications like healthcare monitoring, industrial automation and smart cities to be seamlessly connected to the Internet, thus creating a plethora of IoT applications. An example application is smart-metering, in which a smart-meter can communicate with consumer electronics and other devices in a building/household to retrieve and manage energy consumption. Additionally, a set of lighting devices could be controlled and managed efficiently by the smart-meter, e.g., dimming them down during peak energy periods by means of a multicast message.

Security and privacy are mandatory requirements for the IP-based IoT in order to ensure its acceptance. The interaction between devices must be regulated in the sense that authorized devices joining a specific IoT network in a given location will be granted access to only certain resources provided by the IoT.

To enable this, the IoT network has to:

1. Authorize the joining of the smart device, such that it is provisioned and configured with the corresponding operational parameters, thus providing "network access".
2. Establish and derive pairwise keys, application session keys and multicast keys to enable devices to secure its communication links with each other, and for that, "key management" is needed.
3. Devices should be able to communicate within the network, either securely pairwise or in a "secure multicast" group.

In order to achieve these three security functionalities, there are several challenges: (i) no standard solution exists yet; (ii) mobility of smart devices should be accounted for; (iii) the solution needs to be applicable to large scale deployments; (iv) new communication patterns introduced in IoT such as multicast (beyond just end-to-end communication links), and thus, IP security protocols need to be adapted; and (v) the available resources (bandwidth,

memory, and CPU) are tightly constrained.

Of course, the three research topics are not new, and indeed, some of them (e.g., key management or secure broadcast) have been extensively analyzed in the wireless sensor network literature during the last decade. However, the last step of applying those results to actual standards to get a working solution is still a missing and crucial step towards the success of the IP-based IoT. This work analyzes how this can be achieved.

We present a security architecture for the IP-based IoT in order to explore how these functionalities can be achieved by adapting and extending IP security protocols. With this, our goal is to analyze the trade-offs regarding performance, security, and interoperability so that we obtain a solution that performs reliably, offers high security, and is as interoperable as possible with the standard Internet.

The solution is based on Datagram Transport Layer Security (DTLS). We use the DTLS handshake for network access. For key management, we integrate with the Adapted Multimedia Internet KEYing (AMIKY) protocol for efficient key management and generation of pairwise keys within an IoT network. Secure multicast operation is enabled through the direct use of DTLS record layer with the multicast keys to protect CoAP messages on top of IP multicast.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This Internet Draft defines the following terminology:

Internet of Things (IoT): A paradigm in which a diverse set of devices with different resources and capabilities (including sensors, actuators, smart phones, etc) are connected to the Internet, each is equipped with a uniquely identifiable IP address that can be contacted from anywhere and at anytime.

IoT domain: An IoT network that is connected to the public Internet through a number of 6LoWPAN border routers where the devices and services in the network are managed by a domain manager that could be located within the IoT network itself or in the public Internet.

Network access: A joining device is authenticated and then checked whether it is authorized to join a network. An IP address and a link-layer (L2) key are allocated to the joining device upon successful

authentication and authorization of the joining device, hence enabling the device to communicate in the secure network. This process is called network access.

Key management: A process of distributing, updating and renewing cryptographic materials including keying materials for deriving unicast and multicast keys, random numbers, session keys for unicast communication, and multicast group keys.

Security handshake protocol: A security protocol to authenticate two communicating devices and subsequently establishes a shared secret-key between them to secure their communication. The Datagram Transport Layer Security (DTLS) is referred as the security handshake protocol in this specification.

Link local address: A stateless IPv6 address that is intended for a point-to-point communication between two devices that are within the communication of each other. The packets with a link local address will not be routed or forwarded further by routers.

Pairwise key: A secret symmetric key that is shared between two communicating devices in the network, enabling them to encrypt and authenticate data packets exchanged between them.

Multicast key: A secret symmetric key that is shared by a group of devices in the network. It is used to protect the multicast group communication.

2. Related Work and Background

The "Datagram Transport Layer Security (DTLS)" protocol [RFC4347] is a datagram-compatible adaptation of TLS that runs on top of UDP. DTLS uses similar messages as defined in TLS including the DTLS handshake to establish a secure unicast link and the DTLS record layer to protect this link. The "DTLS handshake" supports different types of authentication mechanisms, e.g., using a pre-shared key, public-key certificates, and raw public-keys. DTLS is the mandatory standard for protection of CoAP [I-D.ietf-core-coap] messages. DTLS differs from TLS mainly in three aspects:

- (1) DTLS provides DoS protection through a stateless cookie exchange;
- (2) DTLS adds functionality to ensure a reliable link during its handshake to solve UDP's inherent packet losses and reordering;
- (3) The record layer includes an explicit sequence number (again, due to the reordering issues in UDP) so that payload integrity and reply protection can be ensured.

The Adapted Multimedia KEYing (AMIKEY) [I-D.alexander-roll-mikey] is

used to provide keying material for securing uni- and multicast communications within constrained networks and devices. It is based on MIKEY, a Key Management Protocol intended for use in real-time applications [RFC3830]. For this purpose, AMIKEY provides different message exchanges that may be transported directly over UDP and TCP. Essentially, they can be integrated within other protocol like DTLS. Our solutions make use of AMIKEY's key derivation mechanism as we consider it to be efficient for constrained networks.

3 Use Cases & Problem Statement

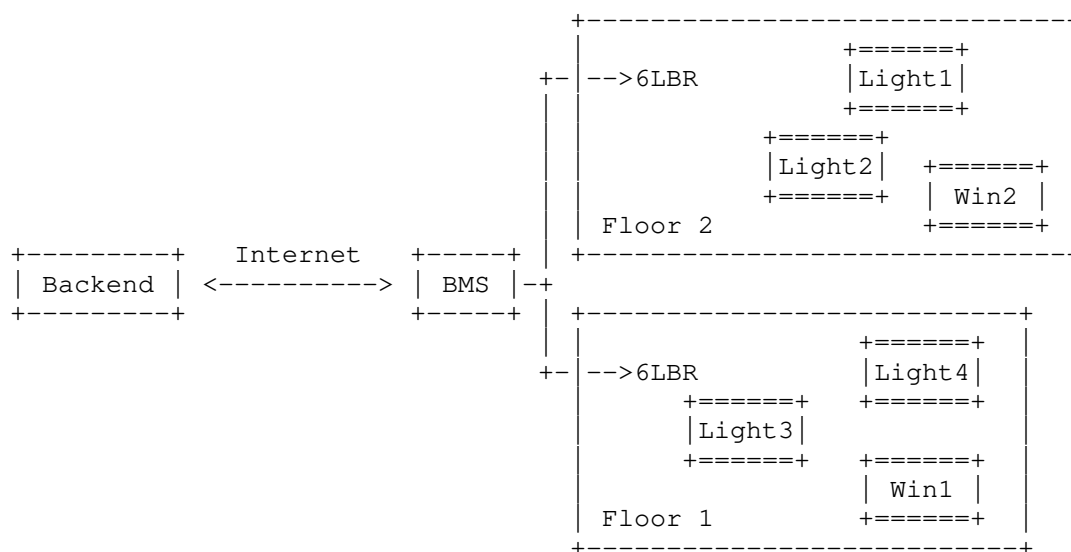


Figure 1: Building Management Systems (BMS) Scenario

Our work targets an "IoT network" running 6LoWPAN/CoAP over multiple hops with both uni- and multicast links, i.e., typical edge networks in the IoT. Devices in the "IoT network" are mobile or stationary and exhibit tight processing, memory and bandwidth constraints. The "IoT network" is connected to the public Internet through a number of 6LoWPAN border routers (6LBR). Further, we consider a centrally managed scenario in which the devices and services in each "IoT network" are managed by a "domain manager". The "domain manager" could be located within the "IoT network" or in the public Internet. The "domain manager" along with the "IoT networks" it manages is denoted as the "IoT domain". Figure 1 illustrates an example of Building Management Systems (BMS) scenario where smart devices within the building (e.g., lighting devices, window blinds) form several multi-hop IoT networks connected to a remote building management

system via some border routers.

3.1 Problem Statement and Requirements

We consider an IoT domain with many devices that dynamically join the network, then provide or request a certain service and finally leave the network. These services are provided or requested using either unicast (e.g., switching on the heater) or multicast group communication (e.g., switching on all lights in a room). We identify three main problems that currently lack a standardized solution for IoT networks:

- o Network access -- A new joining device must only be able to communicate in a secure IoT network after securely joining the IoT network and receiving all necessary access parameters, e.g., commissioning a new lighting bulb into the building network.

The multi-hop nature of IoT networks leads to a key challenge here since a joining device and the domain manager cannot reach each other by means of regular IP routing so that specific approaches are needed.

Similarly, devices that leave the IoT network should not be able to access the network with previous access parameters.

- o Key management -- A lightweight mechanism to derive and manage different keys to secure interactions in the IoT domain is required, e.g., different pairwise, group, and network keys.
- o Secure uni- and multicast communication -- A secure transport protocol is needed to protect the communication in the IoT domain.

This includes both unicast and multicast links, protected using the derived keys, e.g., preserving the integrity and confidentiality of the exchanged commands.

Additional requirements are that the solutions need to be scalable for an IoT domain with several hundreds or thousands of resource constrained devices and be based on standard IP protocols for easier interaction and interoperability with the Internet. In this work, we provide a solution to these three problems based on a smart combination of DTLS and AMIKEY with only minimal modifications.

3.2 Threat model, Security Goals & Assumptions

We assume the Internet Threat Model [RFC3552] in which a malicious

adversary can read and forge network traffic between devices at any point during transmission, but assume that devices themselves are secure. In many IoT application areas the network is indeed untrusted (e.g., wireless communications in public places, large factories, office buildings). Security of the end devices is important to create a secure IoT scenario, however device security is not within the scope of this draft. The Internet Threat Model is thus a reasonable choice in our context.

We further identify the following threats in an IoT domain and the corresponding security goals:

- o Secure Network Access -- Attackers can perform network attacks, e.g., flooding the network and using the network for other communication purposes. To this end, only devices that have been authenticated and authorized through a secure network access process should be allowed to communicate within the network.
- o Key Management -- Attackers can attempt to compromise the keying materials, pairwise keys, or multicast group keys by exploiting the vulnerability on the devices. Therefore, secure key derivation and key update mechanisms are required to manage all cryptographic keys. Similarly, compromising the derived keys does not enable the attackers to obtain information about the keying materials.
- o Secure Uni- and Multicast -- The adversary can maliciously modify either unicast or multicast traffic in the IoT network. Additionally, the adversary can eavesdrop on the data exchanged within an IoT domain. For unicast, two communicating parties must establish a pairwise key to secure the confidentiality, integrity and authenticity of the information exchanged. Multicast communication is protected using a group key, thus only allowing authenticated and authorized group members to send messages to a multicast group. We assume that the multicast group members can trust each other since they are authenticated and authorized by the domain manager. Therefore, we do not additionally require source authentication in the messages as will be detailed further later on. If a device leaves a multicast group, it must not be able to rejoin and send messages to the group later on.

Due to the resource constrained devices in the IoT network, our proposed security architecture is based on the assumption that a device has been configured with a PSK that is known "a priori" to the domain manager of the IoT domain it wants to join. This assumption is reasonable since a PSK could be embedded and registered during the manufacturing process of a device and the domain manager can retrieve

it from a central server. If more powerful devices are available, our secure network access can be easily updated to work with public-key cryptography.

4 Design

In this section, we detail the design of our solution to the three problems (i) secure network access, (ii) key management, and (iii) uni- and multicast communication as identified in Section 3.1. The solution is mainly built on DTLS and AMIKEY.

4.1 Overview

The first phase accounts for "secure network access". In our architecture, the network is protected at link layer by means of a symmetric-key (L2 key), which is unknown to the joining device "a priori". Using its link local address, the joining device authenticates itself to the domain manager of the IoT domain by means of an initial handshake (DTLS) that is based on a PSK. The PSK is assumed to have been pre-configured in the device (cf. Section 3.2). On success, the domain manager issues access parameters (L2 key) that would allow the joining device to access the secured IoT network and to receive a routable IPv6 address. As the link local address only enables one hop communication, this poses a key challenge in multi-hop networks in that the domain manager cannot be reached directly. In the proposed solution, this issue is dealt with by means of a relay device, e.g., as was done in the PANA protocol by means of the PANA relay element [RFC6345]. Figure 2 shows the generic logic of the relay element.

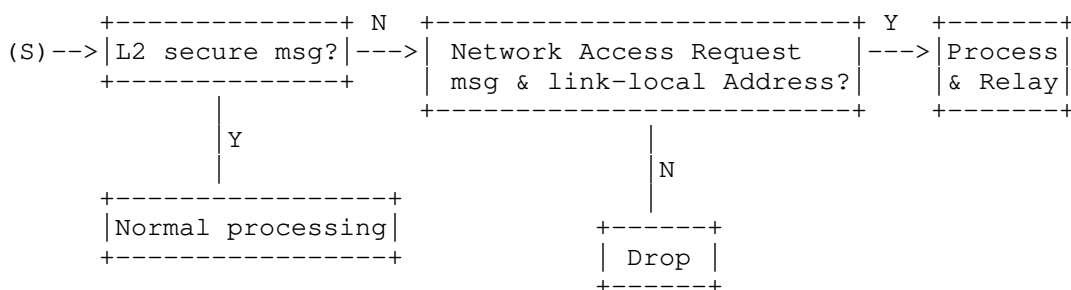


Figure 2: Relay logic for secure network access

The second phase deals with "key management". The joining device is provided with keying material to interact with other devices either in pairs or groups. For pairwise key generation, two communicating devices that wish to interact with each other can derive a pairwise

key based on their identities, e.g., using a polynomial scheme [Blundo-polynomial]. For group key generation, we assume that the domain manager controls the multicast groups. A joining device that wishes to participate in a multicast group indicates this to the domain manager during the initial handshake, and if authorized, receives the required multicast group keys from the domain manager.

Neither pairwise nor group keys are used directly, instead they serve as root keying material in the MIKEY key derivation mechanism in order to derive fresh purpose-specific session keys for any pair or group of devices in the IoT network. The protocol framework for requesting and managing these purpose-specific keys is based on the lightweight MIKEY-extension called AMIKEY [I-D.alexander-roll-mikey].

The final phase, "secure uni- and multicast", is achieved by using CoAP carried over the DTLS record layer. The pairwise or group keys derived in the key management phase are used to protect the communication links.

4.2 Secure Network Access

We describe the details of the initial DTLS based handshakes as well as how multi-hop environments are handled.

IETF CoRE working group defines DTLS for securing the transport of messages in an IoT domain. In this approach, the DTLS handshake protocol is used during the secure network access phase. The DTLS might be based on public-key certificates or raw public-keys as specified in CoAP. Our design uses DTLS-PSK [RFC4279], because it incurs less overhead and reduces the number of exchanged messages. We now describe how DTLS-PSK is used in a single- and multi-hop scenario.

Single-hop: In this case, the joining device can be authenticated with the domain manager by performing the DTLS handshake based on the PSK and relying on the link-local address. Once the device has been authenticated and authorized for the network, the established DTLS secure channel between the domain manager and the joining device is used to issue the L2 key to the device. DTLS-PSK provides resiliency against Denial-of-Service attacks through a cookie mechanism.

Multi-hop: DTLS runs on UDP but communication is limited to a single hop due to the link local address. To deal with this, a relay device is responsible for forwarding the messages by using a mapping between the link local address of the joining device and the relay's IP address. The relaying logic consists of changing the link local address of the joining device with

the relay device's IP address, in a similar way as done in PANA relay element [6345]. This indeed allows for the provisioning of L2 key to the joining device so that it can later receive its IP address through the neighbor discovery protocol [6775]. However, note that the DTLS channel established during the handshake is bound to the relay's IP address and not the new IP address of the joining device. Hence, if the device were to communicate with the domain manager again, it would have to redo the DTLS handshake using its new IP address. This means that the DTLS channel established during network access is transient and it is closed by the relay device once the handshake is finished.

4.3 Key Management

This section describes the details of key management for unicast and group communication. The goal of this phase is to set up an AMIKEY crypto-session bundle (CSB) for unicast as well as group communication. A CSB is built from some root keying material (the TEK Generation Key (TGM)) and some random bits ("RAND"). AMIKEY then defines a lightweight mechanism for the derivation and management of fresh purpose-specific keys, called Traffic Encryption Keys (TEKs), that are used to secure the communication links. We now explain, for both the unicast and multicast case, how the root keying material, i.e., the TGM, is obtained, how the CSB is negotiated and set up, and finally how TEKs are requested and generated.

4.3.1 Management of unicast keys

DTLS runs on the transport layer, and thus we can use it directly to protect the applications. Two communicating devices can establish a pairwise keys using a polynomial scheme, in which each device's polynomial share is used to facilitate fast pairwise key agreement between them. This pairwise key serves as the PSK in DTLS-PSK [RFC4279] enabling any two applications running on the devices to derive a session key. In detail:

1. Two applications running on the devices "D1" and "D2" start a DTLS-PSK handshake. They exchange their identities ID_D1 and ID_D2 as extensions to the first two handshake messages, the "ClientHello" and "ServerHello".
2. Both devices then generate a pairwise key, e.g., if a polynomial scheme is used, they use their polynomial shares and their respective identities to arrive on a pairwise key:
$$K(D1,D2) = F(ID_D1, ID_D2) = F(ID_D2, ID_D1).$$
3. The derived key $K(D1,D2)$ is used as the PSK to complete the DTLS-PSK handshake. This PSK can be regarded as the TGM.
4. The result of the DTLS-PSK handshake is a session key used to protect the communication link between the two applications on

both devices.

4.3.2 Management of multicast keys

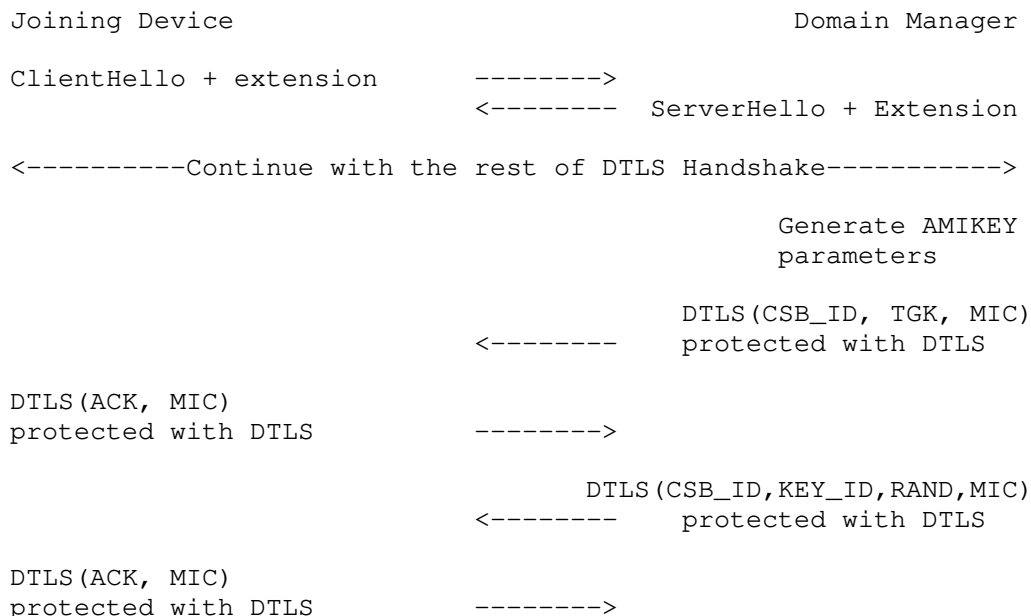


Figure 3: DTLS-based Multicast Key Management

The joining device first indicates during the network access phase that it wishes to join a certain multicast group by adding a request with the group id in the extension part of the "ClientHello". The domain manager then issues the multicast group keys to the device, if it has been authorized to join. It is carried as payload over DTLS record layer after the initial handshake has finished as shown in Figure 3. Two new "Content Types" for the DTLS header have been defined for this purpose to distinguish between the DTLS protected application data and key management data. They are the necessary parameters to set up a CSB, i.e. (1) "Master Key Data" (e.g., the TGK) and (2) "Security Parameters Data" (e.g., the "RAND" values). The fresh TEKs can then be derived from the CSB by every group member for secure group communication.

When a device leaves a group, the domain manager deletes it from the list of authenticated nodes, increases the TEK_ID and starts a new TEK derivation process. The parameters required for generation of the fresh TEK are encrypted so that the leaving device cannot derive the new TEK and cannot rejoin without re-authorization.

4.4 Secure Uni- and Multicast Communication

Once the pairwise session keys or multicast group session key has been derived, a secure channel can be created to transport data (CoAP messages) between the devices in the IoT network. For this, we rely on the DTLS record-layer to create a secure transport layer for CoAP. In this case, the standard DTLS is used to secure the unicast communication.

For multicast communication, the combination of our proposed handshake protocols with the usage of DTLS record-layer for multicast security is based on [I-D.keoh-multicast-security].

4.4.1 Unicast Communication

Any pair of devices in the IoT domain that wish to communicate with each other, establish a CSB and derive a fresh unicast TEK through DTLS as described in Section 4.3.1. The TEK is used in the DTLS record layer (based on AES-CCM) to protect the message exchange between two applications. AES-CCM [RFC3610] is an AES mode of operation that defines the use of AES-CBC for MAC generation with AES-CTR for encryption. The CCM counter (corresponding to the DTLS epoch and sequence number fields) are initialized to 0 upon TEK establishment and used in the nonce construction in a standard way.

4.4.2 Multicast Communication

The multicast solution relies on IP multicast (i.e., an IP multicast address) for routing purposes and adds a security layer on top. To protect the communication, a group of devices establishes a multicast CSB and fresh TEKs using DTLS as described in Section 4.3.2. The TEK is then used to protect CoAP messages transported over the DTLS record layer in AES-CCM mode and routed via IP-multicast. Each device in the group uses a CCM nonce composed of a fixed common part (the content type from the DTLS record layer and the group identifier) and a variable part (the epoch and sequence number fields in the DTLS record layer). This ensures a unique nonce for each message [RFC3610] in the context of a same key.

5 Implementation and Evaluation

This section describes the prototype of our security solution and evaluates the memory and communication overheads.

5.1 Prototype Implementation

The prototype is written in C and run as an application on Contiki OS 2.5 [Dunkels-contiki], an event-driven open source operating system

for constrained devices. They were tested in the Cooja simulator and then ported to run on Redbee Econotag hardware, which features a 32-bit CPU, 128 KB of ROM, 128 KB of RAM, and an IEEE 802.15.4 enabled-radio with an AES hardware coprocessor. The prototype comprises all necessary functionalities to adapt to the roles as a domain manager or a joining device.

The prototype is based on the "TinyDTLS" [Bergmann-Tinydtls] library and includes most of the extensions defined in Section 4 and the adaptation as follows:

- (1) We disabled the cookie mechanism in order to fit messages to the available packet sizes and hence reducing the total number of messages when performing the DTLS handshake.
- (2) We used separate delivery instead of flight grouping of messages and redesigned the retransmission mechanism accordingly.
- (3) We modified the "TinyDTLS" AES-CCM module to use the AES hardware coprocessor.
- (4) The Relay Element functionality for multi-hop scenario has not been implemented.
- (5) We expanded the DTLS state machine with the necessary additions for our key management solution.

The following subsections further analyze the memory and communication overhead of the solution in a single-hop scenario.

5.2 Memory Consumption

Table 1 presents the codesize and memory consumption of the prototype differentiating (i) the state machine for the handshake, (ii) the cryptographic primitives, (iii) the key management functionality and (iv) the DTLS record layer mechanism for secure multicast communications.

The use of DTLS appears to incur large memory footprint both in ROM and RAM for two reasons. First, DTLS handshake defines many message types and this adds more complexity to its corresponding state machine. The logic for message re-ordering and retransmission also contribute to the complexity of the DTLS state machine. Second, DTLS uses SHA2-based crypto suites which is not available from the hardware crypto co-processor.

	DTLS	
	ROM	RAM
State Machine	8.15	1.9
Cryptography	3.3	1.5
Key Management	1.0	0.0
DTLS Record Layer	3.7	0.5
TOTAL	16.15	3.9

Table 1: Memory Requirements in KB

5.3 Communication Overhead

We evaluated the communication overhead in the context of "network access" and multicast "key management". In particular, we examine the message overhead and the number of exchanged bytes under ideal condition without any packet loss in a single-hop scenario, i.e., domain manager and a joining device are in communication range.

	DTLS
No. of Message	12
No. of round trips	4
802.15.4 headers	168B
6LowPAN headers	480B
UDP headers	96B
Application	487B
TOTAL	1231B

Table 2: Communication overhead for Network Access and Multicast Key Management

Table 2 summarizes the required number of round trips, number of messages and the total exchanged bytes for the DTLS-based handshake carried out in ideal conditions, i.e., in a network without packet losses. DTLS handshake is considered complex as it involves the exchange of 12 messages to complete the handshake. Further, DTLS runs on top the transport layer, i.e., UDP, and hence this directly increases the overhead due to lower layer per-packet protocol headers.

5.4 Message Delay, Success Rate and Bandwidth

Section 5.3 provided an evaluation of the protocol in an ideal condition, thus establishing the the baseline protocol overhead. We further examined and simulated the protocol behavior by tuning the packet loss ratio. In particular, we examined the impact of packet loss on message delay, success rate and number of messages exchanged in the handshake.

We consider a complete handshake to include the protocols to perform "network access" and "multicast key management". Figure 4 shows the percentage of successful handshakes as a function of timeouts and packet loss ratios. As expected, a higher packet loss ratio and smaller timeout (15s timeout) result in a failure probability of completing the DTLS handshake. When the packet loss ratio reaches 0.5, practically no DTLS handshake would be successful.

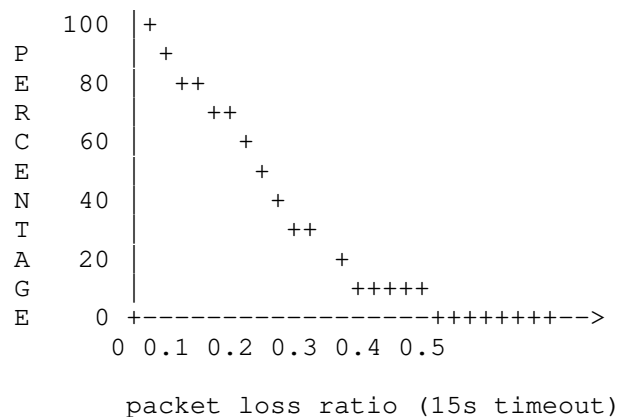


Figure 4: Average % of successful handshakes

Delays in network access and communication are intolerable since they lead to higher resource consumption. As the solution relies on PSK, the handshake protocol only incurs a short delay of a few milliseconds when there is no packet loss. However, as the packet loss ratio increases, the delay in completing the handshake becomes significant because loss packets must be retransmitted. Our implementation shows that with a packet loss ratio of 0.5, the the times to perform network access and multicast key management could take up to 24s.

Finally, another important criterion is the number of messages exchanged in the presence of packet loss. A successful handshake could incur up to 35 or more messages to be transmitted when the packet loss ratio reaches 0.5. This is mainly because the DTLS retransmission is complex and often requires re-sending multiple

messages even when only a single message has been lost.

6. Conclusions and future work

This Internet Draft presented an approach to secure the IP-based Internet of Things using DTLS with the focus on (i) secure network access, (ii) key management, and (iii) secure uni- and multicast communication. Apart from secure unicast communication with DTLS, there are no standard IP solutions in IETF for these unavoidable problems when deploying an IoT network. We have shown that the existing IP-based security protocol, i.e., DTLS can be used and adapted to cater for low resource devices (bandwidth, memory and CPU) and new communication patterns such as multicast and multi-hop network access.

As a proof of concept, we implemented the an architecture based on DTLS over Contiki OS running on a Redbee Econotag. Our work has shown that the re-use of the existing standardized DTLS protocol to solve these problems with a compromise in efficiency is feasible. We hope that our work provides valuable protocol designs and evaluation results (with their pros and cons) which can provide the much needed direction for the standardization effort in IETF to ensure best solutions are adopted.

7 Security Considerations

This document discusses various design aspects for of DTLS implementations. As such this document, in entirety, concerns security.

8 IANA Considerations

tbd

9. Acknowledgements

The authors greatly acknowledge discussion, comments and feedback from Dee Denteneer and Jan Henrik Ziegeldorf. We also appreciate the prototyping and implementation efforts by Pedro Moreno-Sanchez and Francisco Vidal-Meca who worked as interns at Philips Research.

10 References

10.1 Normative References

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer

Security", RFC 4347, April 2006.

- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC4279] Eronen, P., Ed., and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.
- [RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.

9.2 Informative References

- [I-D.ietf-core-coap] Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-12 (work in progress), October 2012.
- [I-D.alexander-roll-mikey] Alexander, R., and Tsao, T. "Adapted Multimedia Internet KEYing (AMIKEY): An extension of Multimedia Internet KEYing (MIKEY) Methods for Generic LLN Environments", draft-alexander-roll-mikey-lln-key-mgmt-04 (work-in-progress), September 2012.
- [Blundo-polynomial] Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., and Yung, M. "Perfectly-Secure Key Distribution for Dynamic Conferences", Advances in Cryptology (CRYPTO'92), 1993.
- [RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., and Yegin, A. "Protocol for Carrying Authentication for Network Access (PANA) Relay Element", RFC 6345, August 2011.
- [RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and Bormann, C. "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[I-D.keoh-multicast-security]

Keoh, S., Garcia-Morchon, O., and Kumar, S. "DTLS-based Multicast Security for Low-Power and Lossy Networks (LLNs)" (work-in-progress), October 2012.

[Dunkels-Contiki]

Dunkels, A., Gronvall, B., and Voigt, T. "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors", In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, IEEE, 2004.

[Bergmann-Tinydtls] Bergmann, O. "TinyDTLS - A Basic DTLS Server Template", <http://tinydtls.sourceforge.net>, 2012.

Authors' Addresses

Sye Loong Keoh
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: sye.loong.keoh@philips.com

Sandeep S. Kumar
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: sandeep.kumar@philips.com

Oscar Garcia-Morchon
Philips Research
High Tech Campus 34
Eindhoven 5656 AE
NL

Email: oscar.garcia@philips.com

Light-Weight Implementation Guidance
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

M. Kovatsch
ETH Zurich
October 15, 2012

Implementing CoAP for Class 1 Devices
draft-kovatsch-lwig-class1-coap-00

Abstract

The Constrained Application Protocol (CoAP) is designed for resource-constrained nodes and networks, e.g., sensor nodes in low-power lossy networks (LLNs). Still, to implement this Internet protocol on Class 1 devices, i.e., ~10KiB of RAM and ~100KiB of ROM, light-weight implementation techniques are necessary. This document provides the lessons learned from implementing CoAP for Contiki, an operating system for tiny, battery-operated networked embedded systems. The information may become part of the Light-Weight Implementation Guidance document planned by the IETF working group LWIG.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Implementing CoAP	3
2.1. Memory Management	4
2.2. Message Buffers	4
2.3. Retransmissions	5
2.4. Separate Responses	5
2.5. Deduplication	5
2.6. Observing	6
2.7. Blockwise Transfers	6
2.8. Developer API	7
3. Low-power Wireless	7
3.1. Radio Duty Cycling	8
3.2. Sleepy Nodes	8
4. Security Considerations	9
5. Informative References	9
Author's Address	10

1. Introduction

The Internet protocol suite is a suitable solution to realize an Internet of Things (IoT), a network of tiny networked embedded devices that create a link to the physical world. The narrow waist of IP can be used to directly access sensor readings throughout a sustainable city, acquire the necessary information for the smart grid, or control smart homes, buildings, and factories---seamlessly from the existing IT infrastructure. The layered architecture helps to manage the complexity, as multiple aspects such as routing over lossy links, link layer adaption, and low-power communication have to be addressed. Nonetheless, attention has to be given to achieve light-weight implementations that can run on resource-constrained devices such as sensor nodes with only microcontroller units (MCUs), ~10KiB of RAM, and ~100KiB of ROM [I-D.ietf-lwig-guidance]. Figure 1 depicts a typical stack configuration for such Class 1 devices. This document discusses a light-weight implementation of CoAP at the application layer in Section 2 and techniques for energy-efficiency such as radio duty cycling in Section 3.

Layer	Protocol
Application	CoAP
Transport	UDP
Network	IPv6 / RPL
Adaptation	6LoWPAN
MAC	CSMA / link-layer bursts
Radio Duty Cycling	ContikiMAC
Physical	IEEE 802.15.4

A typical stack configuration for Class 1 devices.

Figure 1

2. Implementing CoAP

The following experience stems from implementing CoAP for the Contiki operating system [ERBIUM], but is generalized for any embedded OS. The information is not meant to be a final solution, but a first step towards a Light-Weight Implementation Guidance contribution. Alternatives will be incorporated throughout the merging process. The document assumes detailed knowledge of CoAP, its message format and interaction model. For more information, please refer to [I-D.ietf-core-coap], [I-D.ietf-core-block], and [I-D.ietf-core-observe].

2.1. Memory Management

For embedded systems, it is common practice to allocate memory statically to ensure stable behavior, as no memory management unit (MMU) or other abstractions are available. For a CoAP node, the two key parameters are the number of (re)transmission buffers and the maximum message size that must be supported by each buffer. It is common practice to set the maximum message size far below the 1280-byte MTU of 6LoWPAN to allow more than one open confirmable transmissions at a time (in particular for observe notifications). Note that implementations on constrained platforms often not even support the full MTU. Larger messages must then use blockwise transfers [I-D.ietf-core-block], while a good trade-off between 6LoWPAN fragmentation and CoAP header overhead must be found. Usually the amount of available free RAM dominates this decision, on current platforms ending up at a maximum message size of 128 or 256 bytes plus maximum estimated header size.

2.2. Message Buffers

Class 1 devices usually run an OS or event loop system with cooperative multi-threading. This allows to optimize memory usage through in-place processing and reuse of buffers. Incoming payload and byte strings of the header can be directly accessed in the IP buffer, which is provided by the OS, using pointers. For numeric options, there are two alternatives: Either process the header on the fly when an option is accessed or initially parse/allocate all values into a local data structure. Although the latter choice requires an additional amount of memory, it is preferable. First, local processing anyway requires integers in host byte order and stored in a variable of corresponding type. Second, on-the-fly processing might force developers to set options for outgoing messages in a specific order or cause extensive memmove operations due to CoAP's delta encoding.

CoAP servers can significantly benefit from in-place processing, as they can create responses directly in the incoming IP buffer. When a CoAP server only sends piggy-backed or non-confirmable responses, no additional buffer is required in the application layer. This, however, requires an elaborated timing so that no incoming data is overwritten before it was processed. Note that an embedded OS usually reuses a single buffer for incoming and outgoing IP packets. So, either care or a buffer to save the incoming data has to be spent in any case.

For clients, this is only an option for non-reliable requests that do not need to be kept for retransmission. Using the IP also for retransmissions would require to forbid any packet reception during

an open request, but could be applied in some cases.

Empty ACKs and RST messages can promptly be assembled and sent using the IP buffer. The first few bytes are usually parsed into the local data structure and can be overwritten without harm.

2.3. Retransmissions

CoAP's reliable transmissions require the before-mentioned retransmission buffers. For clients, obviously the request has to be stored, preferably already serialized. For servers, retransmissions apply for confirmable separate responses and confirmable notifications [I-D.ietf-core-observe]. As separate responses stem from long-lasting resource handlers, the response should be stored for retransmission instead of re-dispatching a stored request (which would allow for updating the representation). For confirmable notifications, please see Section 2.6, as simply storing the response can break the concept of eventual consistency.

String payloads such as JSON require a buffer to print to. By splitting the retransmission buffer into header and payload part, it can be reused. First to generate the payload and then storing the CoAP message by serializing into the same memory. Thus, providing a retransmission for any message type can save the need for a separate application buffer. This, however, requires an estimation about the maximum expected header size to split the buffer and a memmove to concatenate the two parts.

2.4. Separate Responses

Separate responses are required for long-lasting resource handlers that are too expensive to continuously update in the background to instantly answer from a fresh cache. If possible, those handlers should be realized with split phase execution (e.g., enable a slow sensor, return, and wait for a callback) to not fully block the server during that time. A convenient mechanism to store required data such as the client address and to automatically send the empty ACK could be provided by the implementation. This avoids code duplication when the server has multiple separate resource handlers.

2.5. Deduplication

Deduplication is heavy for Class 1 devices, as the number of peer addresses can be vast. Servers should be kept stateless, i.e., the REST API should be designed idempotent whenever possible. When this is not the case, the resource handler could perform an optimized deduplication by exploiting knowledge about the application. Another, server-wide strategy is to only keep track of non-idempotent

requests.

2.6. Observing

At the server, the list of observers should be stored per resource to only have a handle per observable resource in a superordinate list instead of one resource handle per observer entry. Then for each observer, at least address, port, token, and the last outgoing message ID has to be stored. The latter is needed to match incoming RST messages and cancel the observe relationship.

Besides the list of observers, it is best to have one retransmission buffer per observable resource. Each notification is serialized once into this buffer and only address, port, and token are changed when iterating over the observer list (note that different token lengths might require realignment). The advantage becomes clear for confirmable notifications: Instead of one retransmission buffer per observer, only one buffer and only individual retransmission counters and timers in the list entry need to be stored. When the notifications can be sent fast enough, even a single timer would suffice. Furthermore, per-resource buffers simplify the update with a new resource state during open deliveries.

2.7. Blockwise Transfers

Blockwise transfers have the main purpose of providing fragmentation at the application layer, where partial information can be processed. This is not possible at lower layers such as 6LoWPAN, as only assembled packets can be passed up the stack. While [I-D.ietf-core-block] also anticipates atomic handling of blocks, i.e., only fully received CoAP messages, this is not possible on Class 1 devices.

When receiving a blockwise transfer, each blocks is usually passed to a handler function that for instance performs stream processing or writes the blocks to external memory such as flash. Although there are no restrictions in [I-D.ietf-core-block], it is beneficial for Class 1 devices to only allow ordered transmission of blocks. Otherwise on-the-fly processing would not be possible.

When sending a blockwise transfer, Class 1 devices usually do not have sufficient memory to print the full message into a buffer, and slice and send it in a second step. When transferring the CoRE Link Format from /.well-known/core for instance, a generator function is required that generates slices of a large string with a specific offset length (a 'sonprintf()'). This functionality is required recurrently and should be included in a library.

2.8. Developer API

Bringing a Web transfer protocol to constrained environments does not only change the networking of the corresponding systems, but also the way they should be programmed. A CoAP implementation should provide a developer API similar to REST frameworks in traditional computing. A server should not be created around an event loop with several function calls, but rather by implementing handlers following the resource abstraction.

So far, the following types of RESTful resources were identified:

NORMAL A normal resource defined by a static Uri-Path that is associated with a resource handler function. Allowed methods could already be filtered by the implementation based on flags. This is the basis for all other resource types.

PARENT A parent resource manages several sub-resources by programmatically evaluating the Uri-Path, which may be longer than that of the parent resource. Defining a URI templates (see [RFC6570]) would be a convenient way to pre-parse arguments given in the Uri-Path.

PERIODIC A resource that has an additional handler function that is triggered periodically by the CoAP implementation with a resource-defined interval. It can be used to sample a sensor or perform similar periodic updates. Usually, a periodic resource is observable and sends the notifications in the periodic handler function. These periodic tasks are quite common for sensor nodes, thus it makes sense to provide this functionality in the CoAP implementation and avoid redundant code in every resource.

EVENT An event resource is similar to an periodic resource, only that the second handler is called by an irregular event such as a button.

3. Low-power Wireless

The Internet of wireless things needs power-efficient protocols, but existing protocols have typically been designed without explicit power-efficiency. CoAP is optimized to run over low-power link layers such IEEE 802.15.4, but in low-power wireless systems, ultimate power-efficiency translates into the ability to keep the radio off as much as possible, as the radio transceiver is typically the most power-consuming component. This can be achieved in two ways: So called radio duty cycling (RDC) aims to keep the radio off as much as possible, but performs periodic channel checks to realize

a virtual always-on link. Sleepy nodes instead put the radio into hibernation for a long period during which the node is fully disconnected from the network.

3.1. Radio Duty Cycling

RDC can be achieved through a separate, independent layer between PHY and MAC as depicted in Figure 1. The upper layers can remain more or less untouched and only experience a higher latency, which might require tweaking the timeout parameters. State-of-the-art RDC layers can achieve an idle duty cycling way below 1% while checking the channel several times per second. ContikiMAC for instance achieves a 0.3% cycle with a channel check rate of 4 Hz, which results in a worst-case delay of 250ms per hop. While saving energy, ContikiMAC also makes link-layer transmissions more robust due to its retransmission policy. Please refer to [CONMAC] for details.

In general, RDC can be divided into two approaches: sender initiated (e.g., ContikiMAC) and receiver initiated (e.g., A-MAC [AMAC]). In the first approach, the sender enables the radio first and continuously transmits its message in a strobe until a link-layer ACK is received (note that for IEEE 802.15.4 transceivers, transmitting consumes less energy than receiving). Receivers turn on their radio only periodically to check for these announcements. If they sense a busy channel, the radio is kept on to receive a potential message and finally acknowledge it. In the other approach, the receiver periodically announces that it will keep the radio on for receiving for a while. The senders turns on its radio and listens for an announcement of the recipient. When that is received, it transmits the message (following the scheme of the above MAC layer of course, while back-offs must match the awake time after announcements). Which approach is optimal mainly depends on the communication pattern of the application. Sender initiated RDCs are more efficient for IEEE 802.15.4, but the strobes can congest a busy channel.

3.2. Sleepy Nodes

Going to sleep for a longer time is not transparent for the application layer, as nodes need to re-synchronize and maybe re-associate with the network. Several drafts in the IETF CoRE working group cover this strategy for low-power wireless networking (cf. [I-D.vial-core-mirror-proxy], [I-D.fossati-core-publish-option], [I-D.fossati-core-monitor-option], and [I-D.rahman-core-sleepy]). Such features will have to be integrated into the nodes CoAP implementation as well as the back-end systems. In addition, alternatives to standard diagnosis tools such as ICMP ping will have to be provided, e.g., heartbeats by the application.

This strategy is particular useful for communications other than IEEE 802.15.4. Low-power Wi-Fi for instance is mainly based on long sleeping periods with short wake-up cycles. Although the data rate would be high enough for HTTP over TCP, low-power Wi-Fi can greatly benefit from CoAP and its shorter round trip times. For further information about sleepy nodes based on low-power Wi-Fi see [LPWIFI].

4. Security Considerations

T.B.D.

5. Informative References

- [AMAC] Dutta, P., Dawson-Haggerty, S., Y., A., Liang, C., and A. Terzis, "Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless", In Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys 2010). Zurich, Switzerland, November 2010.
- [CONMAC] Dunkels, A., "The ContikiMAC Radio Duty Cycling Protocol", SICS Technical Report T2011:13, ISSN 1100-3154, December 2011.
- [ERBIUM] Kovatsch, M., Duquennoy, S., and A. Dunkels, "A Low-Power CoAP for Contiki", In Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011). Valencia, Spain, October 2011.
- [I-D.fossati-core-monitor-option] Fossati, T., Giacomini, P., and S. Loreto, "Monitor Option for CoAP", draft-fossati-core-monitor-option-00 (work in progress), July 2012.
- [I-D.fossati-core-publish-option] Fossati, T., Giacomini, P., and S. Loreto, "Publish Option for CoAP", draft-fossati-core-publish-option-00 (work in progress), July 2012.
- [I-D.ietf-core-block] Bormann, C. and Z. Shelby, "Blockwise transfers in CoAP", draft-ietf-core-block-09 (work in progress), August 2012.
- [I-D.ietf-core-coap] Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)",

draft-ietf-core-coap-12 (work in progress), October 2012.

[I-D.ietf-core-observe]

Hartke, K., "Observing Resources in CoAP",
draft-ietf-core-observe-06 (work in progress),
September 2012.

[I-D.ietf-lwig-guidance]

Bormann, C., "Guidance for Light-Weight Implementations of
the Internet Protocol Suite", draft-ietf-lwig-guidance-02
(work in progress), August 2012.

[I-D.rahman-core-sleepy]

Rahman, A., "Enhanced Sleepy Node Support for CoAP",
draft-rahman-core-sleepy-00 (work in progress), July 2012.

[I-D.vial-core-mirror-proxy]

Vial, M., "CoRE Mirror Server",
draft-vial-core-mirror-proxy-01 (work in progress),
July 2012.

[LPWIFI] Ostermaier, B., Kovatsch, M., and S. Santini, "Connecting
Things to the Web using Programmable Low-power WiFi
Modules", In Proceedings of the 2nd International Workshop
on the Web of Things (WoT 2011). San Francisco, CA, USA,
June 2011.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R.,
Levis, P., Pister, K., Struik, R., Vasseur, JP., and R.
Alexander, "RPL: IPv6 Routing Protocol for Low-Power and
Lossy Networks", RFC 6550, March 2012.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M.,
and D. Orchard, "URI Template", RFC 6570, March 2012.

Author's Address

Matthias Kovatsch
ETH Zurich
Universitaetstrasse 6
Zurich, CH-8092
Switzerland

Phone: +41 44 632 06 87
Email: kovatsch@inf.ethz.ch

