Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 17, 2014

J. Arkko
A. Eriksson
A. Keranen
Ericsson

February 13, 2014

Building Power-Efficient CoAP Devices for Cellular Networks
draft-ietf-lwig-cellular-01

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

Copyright Notice

Table of Contents

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

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 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 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.

	SENSOR COMMUNICATION INTERVAL		
POWER SOURCE	Seconds	Minutes or Hours	Days and longer
Battery	Low-power	Low-power or Normally-off	Normally-off
Harvesting	Low-power	Low-power or Normally-off	Normally-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, using the power usage strategies described in [I-D.ietf-lwig-terminology], is shown in Figure 1. 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 them 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.

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

(Access Point Name). 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.ietf-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, JavaScript Object Notation (JSON) [RFC4627], Efficient XML Interchange (EXI) [W3C.REC-exi-20110310], 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 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

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

[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-core-observe]
Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-11 (work in progress), October 2013.

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

[I-D.ietf-core-resource-directory]
Shelby, Z., Bormann, C., and S. Krco, "CoRE Resource Directory", draft-ietf-core-resource-directory-01 (work in progress), December 2013.

[W3C.REC-exi-20110310]
Kamiya, T. and J. Schneider, "Efficient XML Interchange (EXI) Format 1.0", World Wide Web Consortium Recommendation REC-exi-20110310 http://www.w3.org/TR/2011/REC-exi-20110310, March 2011.

[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.

[I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-07 (work in progress), February 2014.

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., Giacomin, 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, 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