Internet Engineering Task Force (IETF)
Request for Comments: 9178
Category: Informational
ISSN: 2070-1721

J. Arkko
Ericsson
A. Eriksson
Independent
A. Keränen
Ericsson

May 2022

Building Power-Efficient Constrained Application Protocol (CoAP) Devices for Cellular Networks

Abstract

This memo discusses the use of the Constrained Application Protocol (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.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9178.

Table of Contents

1. Introduction

This memo discusses the use of the Constrained Application Protocol (CoAP) [RFC7252] 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 of code for the entire application above the 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 [Tiny-CoAP].

This memo primarily targets 3GPP cellular networks in their 2G, 3G, LTE, and 5G 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 a point-to-point communications model and radio technology for the devices in the network.

Our focus is on devices that need to be optimized for power usage and 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 types of communication 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, ongoing, and generally complementary to the techniques presented in this memo. However, the list of techniques described in this document as useful for a particular application may change with the evolution of these underlying technologies.

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. Sections 5, 6, 7, and 8 give guidelines for the use of CoAP in these scenarios.

This document was originally finalized in 2016 but is published six years later due to waiting for key references to reach RFC status. Therefore, some of the latest advancements in cellular network, CoAP, and other technologies are not discussed here, and some of the references point to documents that were state of the art in 2016.

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 a 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 solar cells that are as small and inexpensive as possible.

In battery-operated devices, power usage is even more important. For instance, one of the authors employs over a hundred different sensor devices in their 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 a 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.

Many of these situations lead to a requirement for minimizing power usage and/or maximizing battery lifetimes. Using the power usage strategies described in [RFC7228], mains-powered sensor-type devices can use the Always-on strategy, whereas battery-operated or energy-harvesting devices need to adjust behavior based on the communication interval. For intervals on the order of seconds, the Low-power strategy is appropriate. For intervals ranging from minutes to hours, either the Low-power or Normally-off strategy is suitable. Finally, for intervals lasting days or longer, Normally-off is usually the best choice. Unfortunately, much of our current technology has been built with different objectives in mind -- for instance, 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 on the order of years or even a decade or longer. Some communication devices already reach multi-year lifetimes, and continuous improvements 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.

Power usage cannot be evaluated based solely 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 cooperation 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 a point-to-point communications model and radio technology. 2G, 3G, LTE, and 5G networks are examples of such networks but are 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 are 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 economic 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 worldwide coverage. This means that applications where coverage is important can be built. For instance, most transport-sector applications require national or even worldwide 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 that a device might need to contact to perform its application functions.

  Public networks often provide web proxies and other functionality that can, in some cases, make significant improvements related to delays and costs of communication over the wireless link. For instance, resolving server 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. Likewise, a CoAP proxy or Publish-Subscribe (pub/sub) Broker [CoAP-PubSub] can assist a CoAP device in communication. However, unlike HTTP web proxies, CoAP proxies and brokers are not yet widely deployed in public networks.

  Similarly, given the lack of available IPv4 addresses, 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 (on either 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 cannot be reasonably used for any local discovery purposes. While the configuration of the service provider's router for specific users is theoretically possible, this is difficult to achieve in practice, at least for any small user that cannot 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 to see 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 so that they can communicate 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 the 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 they want 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: (1) fine-grained mechanisms that deal with how often the device needs to wake up for paging messages and (2) cruder mechanisms where the device simply disconnects from the network for a period of time. There are costs and benefits associated with each method, but those are not relevant for this memo, as long as some control method exists. Furthermore, devices could use Delay-Tolerant Networking (DTN) mechanisms [RFC4838] to relax the requirements for timeliness of connectivity and message delivery.

4. Scenarios

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

* 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 previously agreed-upon 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 the freedom to choose when a 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 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 cannot discover who they need to talk to, 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. Registration also facilitates low-power operation, since a device can delegate part of the discovery signaling and reachability requirements to another node.

The registration part is easy, e.g., with a resource directory. The device should perform the necessary registration with such a resource directory -- for instance, as specified in [RFC9176]. In order to do this registration, the device needs to know its Constrained RESTful Environments (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 and then doing a POST on the path of the discovered resource.

Other mechanisms enabling device discovery and delegation of functionality to a non-sleepy node include those discussed in [CoRE-Mirror] and [CoAP-PubSub].

However, current CoAP specifications provide only limited support for discovering the resource directory or other registration services. Local multicast discovery only works in LAN-type networks; it does not work in the public cellular networks discussed in this document. We recommend the following alternate methods for discovery:

* Manual Configuration

  The DNS name of the resource directory 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 cooperate 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 that specific actions 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 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, for example, answer the question "This is a device with identity ID 2456; where is my home registration server?" However, at present, no such resolution system exists. (Note: The EPCGlobal system for Radio Frequency Identification (RFID) resolution is reminiscent of this approach.)

Besides manual configuration, these alternate mechanisms are mostly suitable for large manufacturers and deployments. Good automated mechanisms for discovery of devices that are manufactured and deployed in small quantities are still needed.

6. Data Formats

A variety of data formats exist for passing around data. These data formats include XML, JavaScript Object Notation (JSON) [RFC8259], Efficient XML Interchange (EXI) [W3C.REC-exi-20140211], Concise Binary Object Representation (CBOR) [RFC8949], and various text formats. Message lengths can have a significant effect on the amount of energy required for the communications, and as 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 that readers refer to [RFC8428] for a compact but easily processed and extendable format.

7. Real-Time Reachable Devices

These devices are often best modeled as CoAP servers. The device will have limited control over when it receives messages, and it will have to listen actively for messages, up to the limits of the underlying link layer. If in some phase of its operation the device also acts in the role of a client, 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 more infrequent checking process may save some power.

For sensor-type devices, the CoAP Observe extension (Observe option) [RFC7641] 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 clients other than the currently registered client may come in 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 cannot rely on 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 be hosted in a private network. If the device is hosted on a private network, then all other nodes that 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 Network (VPN) technology -- for instance, IPsec-based VPNs.

Power consumption from unwanted traffic is problematic in these devices, unless they are placed in a private network or protected by an operator-provided firewall service. Devices on an IPv6 network will be afforded some protection due to 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 VPN approach cannot prevent unwanted traffic received at the tunnel endpoint address and may require keep-alive traffic. Special APNs can solve this issue but require an 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 servers [CoRE-Mirror] or CoAP pub/sub Clients [CoAP-PubSub]. When the device initializes itself, it makes a registration of itself in a server or broker 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 and not as a server, and can shut down all communication channels 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 smaller values ranging from minutes to hours.

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

* Radio transmission and reception can be turned off during the sleeping period, reducing power consumption significantly.

* However, some power and time are consumed by having to reattach to the network after the end of a sleep period.

* 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, however, that networks may cache packets for some time. On the other hand, stateful firewalls can effectively remove much of the unwanted traffic for client-type devices.

* The device may exist behind a NAT or a firewall without being impacted. Note that the "simple security" basic IPv6 firewall capability [RFC6092] blocks inbound UDP traffic by default, so just moving to IPv6 is not a direct solution to this problem.

For sleepy devices that represent actuators, it is also possible to use the mirror server or pub/sub broker model. A device can receive information from the server or broker about variable changes via either polling or notifications.

8.1. Implementation Considerations

There are several challenges related to implementing sleepy devices. They need hardware that can be placed in an appropriate sleep mode but 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 reawakening at suitable timescales, 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 settings of these values have an impact on the power consumption of the device, but with current business practices, such settings are rarely negotiated when the user's subscription is provisioned.

9. Security Considerations

There are no particular security aspects related to 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 that have largely NOT yet been addressed in the relevant Internet-Drafts [CoRE-Mirror] [CoAP-Alive] [CoAP-Publ-Monitor]. However, we point out that, in general, security issues in delegation can be solved through either 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 [RFC8387].

The security considerations relating to CoAP [RFC7252] 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 [RFC7252] or data objects.

10. IANA Considerations

This document has no IANA actions.

11. References

11.1. Normative References

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,

<https://www.rfc-editor.org/info/rfc6690>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC9176] Amsüss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/info/rfc9176>.

[W3C.REC-exi-20140211] Schneider, J., Kamiya, T., Peintner, D., and R. Kyusakov, "Efficient XML Interchange (EXI) Format 1.0 (Second Edition)", World Wide Web Consortium Recommendation REC-exi-20140211, February 2014, <https://www.w3.org/TR/exi/>.

[RFC8428] Jennings, C., Shelby, Z., Arkko, J., Keranen, A., and C. Bormann, "Sensor Measurement Lists (SenML)", RFC 8428, DOI 10.17487/RFC8428, August 2018, <https://www.rfc-editor.org/info/rfc8428>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

11.2. Informative References

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/info/rfc4838>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <https://www.rfc-editor.org/info/rfc6092>.

[Tiny-CoAP] Arkko, J., Rissanen, H., Loreto, S., Turanyi, Z., and O. Novo, "Implementing Tiny COAP Sensors", Work in Progress, Internet-Draft, draft-arkko-core-sleepy-sensors-01, 5 July 2011, <https://datatracker.ietf.org/doc/html/draft-arkko-

core-sleepy-sensors-01>.

[RFC8387] Sethi, M., Arkko, J., Keranen, A., and H. Back, "Practical Considerations and Implementation Experiences in Securing Smart Object Networks", RFC 8387, DOI 10.17487/RFC8387, May 2018, <https://www.rfc-editor.org/info/rfc8387>.

[CoAP-Alive]
Castellani, A. and S. Loreto, "CoAP Alive Message", Work in Progress, Internet-Draft, draft-castellani-core-alive-00, 29 March 2012, <https://datatracker.ietf.org/doc/html/draft-castellani-core-alive-00>.

[CoAP-Publ-Monitor]
Fossati, T., Giacomin, P., and S. Loreto, "Publish and Monitor Options for CoAP", Work in Progress, Internet-Draft, draft-fossati-core-publish-monitor-options-01, 10 March 2012, <https://datatracker.ietf.org/doc/html/draft-fossati-core-publish-monitor-options-01>.

[CoRE-Mirror]
Vial, M., "CoRE Mirror Server", Work in Progress, Internet-Draft, draft-vial-core-mirror-proxy-01, 13 July 2012, <https://datatracker.ietf.org/doc/html/draft-vial-core-mirror-proxy-01>.

[CoAP-PubSub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-10, 4 May 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-pubsub-10>.

[Android-Bundle]
"Optimize network access", Android developer note, May 2022, <https://developer.android.com/training/efficient-downloads/efficient-network-access.html>.

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
FI-02420 Jorvas
Finland
Email: jari.arkko@piuha.net

Anders Eriksson
Independent
SE-164 83 Stockholm
Sweden
Email: anders.e.eriksson@posthem.se

Ari Keränen
Ericsson
FI-02420 Jorvas
Finland
Email: ari.keranen@ericsson.com