CoRE Working Group
Internet-Draft
Intended status: Informational
Expires: 2 January 2026

C. Gomez
UPC
S. Aguilar
Sateliot

July 2025

CoAP in Space
draft-gomez-core-coap-space-03

Abstract

This document provides guidance on using the Constrained Application Protocol (CoAP) in spatial environments characterized by long delays and intermittent communication opportunities. Such environments include some Low Earth Orbit (LEO) satellite-based scenarios, as well as deep space scenarios. The document focuses on the approach whereby an IP protocol stack is used for end-to-end communication.

Table of Contents

1. Introduction

Deep space communication occurs between devices on or orbiting different celestial bodies (e.g., different planets of the Solar System). Such environments are characterized by long delays (e.g., in the order of minutes or hours), intermittent communication opportunities, limited energy resources, and relatively low bandwidth in some cases.

Similar characteristics can be found in Non-Terrestrial Networks (NTN) based on sparse Low Earth Orbit (LEO) satellite constellations that provide direct connectivity to Internet of Things (IoT) devices

on Earth, albeit with discontinuous coverage. In such cases, an IoT device may need to wait until it is visited by a satellite to be able to transmit its data. In addition, if the satellite does not have an immediately available link with a ground station or with a second satellite, the first satellite needs to perform store-and-forward operation. This paradigm supports delay-tolerant, non-real-time communication services. Note that extensions to enable store-and-forward operation are being standardized by 3GPP in Release 19 [_GPP].

The Internet Protocol (IP) stack was considered unsuitable for delay-tolerant environments more than two decades ago, leading to the design of the Delay-Tolerant Networking (DTN) architecture [RFC4838] and the Bundle Protocol (BP) [RFC5050] [RFC9171]. However, recent work has revisited such assessment, and it has discussed solutions to use the IP protocol stack in deep space communication [I-D.many-deepspace-ip-assessment][I-D.huitema-quic-in-space].

From the application layer point of view, the analysis in [I-D.many-deepspace-ip-assessment] focuses on the use of HTTP (over QUIC [RFC9000]) in deep space scenarios. However, it also explicitly mentions that the Constrained Application Protocol (CoAP) [RFC7252] "is worth considering for application transport in deep space".

CoAP is an application-layer protocol based on Representational State Transfer (REST). In CoAP, endpoints called clients make requests with the aim to manipulate resources handled by other endpoints called servers. The latter provide responses back to the clients.

CoAP offers several features suitable for its use in delay-tolerant space environments, including lightweight operation, asynchronous message exchanges, and a significant degree of flexibility. This document provides guidance on the use of CoAP for delay-tolerant communication in space environments. Use of CoAP over BP [RFC9171] is outside the scope of this document. Note that there is work in progress intended to specify how CoAP can be carried over BP [I-D.gomez-core-coap-bp].

2. Terminology

2.1. Requirements language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119], [RFC8174], when, and only when, they appear in all capitals, as shown here.

3. CoAP transport

3.1. Overview and underlying transport

CoAP was originally designed to use UDP as its underlying transport protocol [RFC7252]. The message layer of CoAP over UDP supports optional message reliability, simple congestion control, and flow control. A CoAP message that requires reliable delivery is marked as a Confirmable (CON) message. The recipient needs to send an Acknowledgment (ACK) message to confirm successful reception of a CON message. A sender uses a retransmission mechanism with a default timeout and an exponential back-off between retransmissions. A CoAP message that does not require reliability is marked as a Non-confirmable (NON) message. NON messages are not acknowledged.

Subsequently, CoAP was adapted to be carried also over other transports, such as TCP, Transport Layer Security (TLS), and WebSockets [RFC8323]. However, due to the long delays in delay-tolerant environments, initial handshake exchanges (e.g., the three-way handshake of TCP) penalize communication performance significantly. In addition, when TCP is used as the underlying transport-layer protocol, the ability of optionally requesting reliable delivery for a given message (as offered by CoAP over UDP) is lost. Two further advantages of UDP-based CoAP transport are a shorter header size and support for multicast. Therefore, this document will focus on CoAP as used over UDP as the underlying transport [RFC7252].

3.2. Main CoAP parameters and times relevant to delay-tolerant space environments

This section discusses the main parameters and times that are relevant in the context of delay-tolerant space environments. (Note that the complete set of parameters, assumptions, default values, and related times in CoAP can be found in Section 4.8 of RFC7252.)

As a congestion control measure, the maximum number of outstanding interactions between a client and a given server is limited to NSTART, which is set to a default value of 1. A greater value for NSTART can be used only when mechanisms that ensure congestion control safety are used.

The main parameters related with CON messages are indicated next.

ACK_TIMEOUT and ACK_RANDOM_FACTOR. These two parameters determine the duration of the initial retransmission timeout, which is set to a randomly chosen value between ACK_TIMEOUT and ACK_TIMEOUT * ACK_RANDOM_FACTOR. The default values for ACK_TIMEOUT and ACK_RANDOM_FACTOR are 2 s and 1.5, respectively. Therefore, the default initial retransmission timeout in CoAP is between 2 and 3 s.

ACK_TIMEOUT should be set to a value of at least the expected RTT, which in delay-tolerant environments such as deep space may be several orders of magnitude greater than the default one (see Appendix A of [I-D.gomez-core-coap-bp]).

ACK_RANDOM_FACTOR needs to be at least equal to or greater than 1.0. The default value of 1.5 is intended to avoid synchronization effects among different senders when RTTs are in the order of seconds. However, the greater latency in delay-tolerant environments may reduce the risk of synchronization effects therein. In such case, a lower ACK_RANDOM_FACTOR may help reduce total message delivery latency when retries are performed.

MAX_RETRANSMIT. This parameter defines the maximum number of retries for a given CON message. The default value for this parameter is 4. Since there is an exponential back-off between retransmissions, and considering the delay values in delay-tolerant environments, it may be suitable to set this parameter to a value lower than the default one.

The following assumptions on the characteristics of the network and the nodes need to be considered:

MAX_LATENCY is the maximum time a datagram is expected to take from the start of its transmission to the completion of its reception. In RFC 7252, this value is arbitrarily set to 100 s, which is close to the historic Maximum Segment Lifetime (MSL) of 120 s defined in the TCP specification [RFC9293]. However, such value assumes communication between devices on Earth. In delay-tolerant environments, MAX_LATENCY may need to be increased by several orders of magnitude (e.g., at least 1-2 orders of magnitude in deep space, See Appendix A of [I-D.gomez-core-coap-bp]).

PROCESSING_DELAY is the time since a node receives a CON message until it transmits an ACK in response. In RFC 7252, this value is assumed to be of at most the default ACK_TIMEOUT value of 2 s. For the sake of limiting latency, it is assumed that the same value can be used also in delay-tolerant environments.

A relevant CON message derived time is EXCHANGE_LIFETIME. This time indicates the maximum possible time since a CON message is sent for the first time, until ACK reception (which may potentially occur after several retries). EXCHANGE_LIFETIME includes the following components: the total time since the first transmission attempt of a CON message until the last one (called MAX_TRANSMIT_SPAN in RFC 7252), a MAX_LATENCY for the CON, PROCESSING_DELAY, and a MAX_LATENCY for the ACK. The default value for EXCHANGE_LIFETIME is 247 s. However, in delay-tolerant environments, and considering the modified values for protocol parameters and the network characteristics described above, EXCHANGE_LIFETIME may have to be even several orders of magnitude greater than the default one (e.g., at least 2-3 orders of magnitude in deep space, See Appendix A of [I-D.gomez-core-coap-bp]).

The main time related with NON messages is NON_LIFETIME. This is the time since a NON message is transmitted until its Message ID can be safely reused. This time is actually equal to MAX_LATENCY, therefore its default value is 100 s. However, as described earlier, in delay-tolerant environments it may need to be increased by several orders of magnitude (e.g., at least 1-2 orders of magnitude in deep space, See Appendix A of [I-D.gomez-core-coap-bp]).

Note that implementations may also need to be adapted if they have been designed to use 8-bit timers to handle CON or NON message lifetimes (e.g., to retire Message IDs) in seconds.

4. Caching

RFC 7252 states that "CoAP endpoints MAY cache responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests". Note that caching may also offer energy consumption savings.

In delay-tolerant space scenarios, the efficiency provided by the caching feature is particularly suitable. Nevertheless, it needs to be adapted to the characteristics of the scenario, especially in terms of latency.

A cached response can be reused as long as it is considered "fresh". In order to determine the freshness of a response, the origin server uses the Max-Age option to indicate that the response is to be considered not fresh after its age is greater than the specified number of seconds.

The default Max-Age value is 60 seconds. When a response does not carry a Max-Age option, it is considered to have an associated Max-Age value equal to the default one. Also, the Max-Age value is

intended to be current at the time of transmission. Therefore, considering the latencies of delay-tolerant environments, if a response is intended to be cacheable, the origin server needs to include a Max-Age option of an appropriate value with the response (the maximum possible option value being 2**32-1 seconds (i.e., ~136.1 years)). Of course, it will only make sense to consider that a response is cacheable if it can be fresh for a time greater than the expected latency between the origin server and the caching CoAP endpoint. If a CoAP endpoint receives a response known to be not fresh (e.g., if communication latency is greater than its associated Max-Age), the CoAP endpoint will not store the response.

5. Proxying

RFC 7252 defines a "proxy" as "An intermediary that mainly is concerned with forwarding requests and relaying back responses, possibly performing caching, namespace translation, or protocol translation in the process". The same specification also states that "A proxy is a CoAP endpoint that can be tasked by CoAP clients to perform requests on their behalf". Among others, this can be useful "to service the response from a cache in order to reduce response time and network bandwidth or energy consumption". The latter are advantages that may be desirable as well in delay-tolerant environments.

Depending on the protocol(s) supported at each side of the proxy, a proxy can be a "CoAP-to-CoAP proxy", which "maps from a CoAP request to a CoAP request", or a "cross-proxy", which "translates between different protocols, such as a CoAP-to-HTTP proxy or an HTTP-to-CoAP proxy" [RFC 7252]. Figure 2 and Figure 3 illustrate the upper-layer protocol stacks for a CoAP-to-CoAP proxy and an HTTP-to-CoAP cross-proxy. Figure 1 illustrates a proxyless scenario, where CoAP is used end-to-end, between a CoAP origin server and a CoAP client. (Note: a practical scenario used in actual Earth-to-Mars communications comprises three entities: Earth, Mars orbiter and Mars rover [Blanchet].)

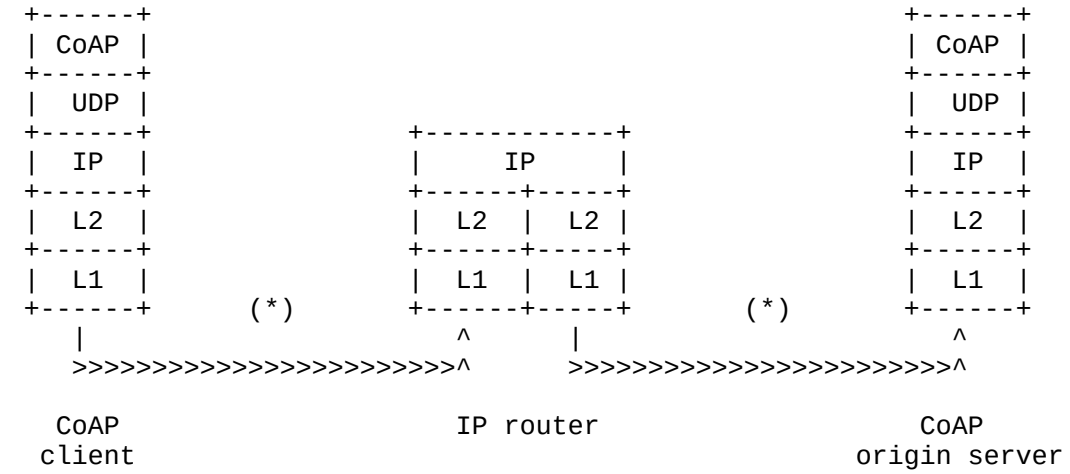

Figure 1: Direct CoAP communication in a scenario without a proxy. (*) There may be zero or more IP routers between the CoAP client and the IP router shown in the figure, and zero or more IP routers between the latter and the CoAP origin server.

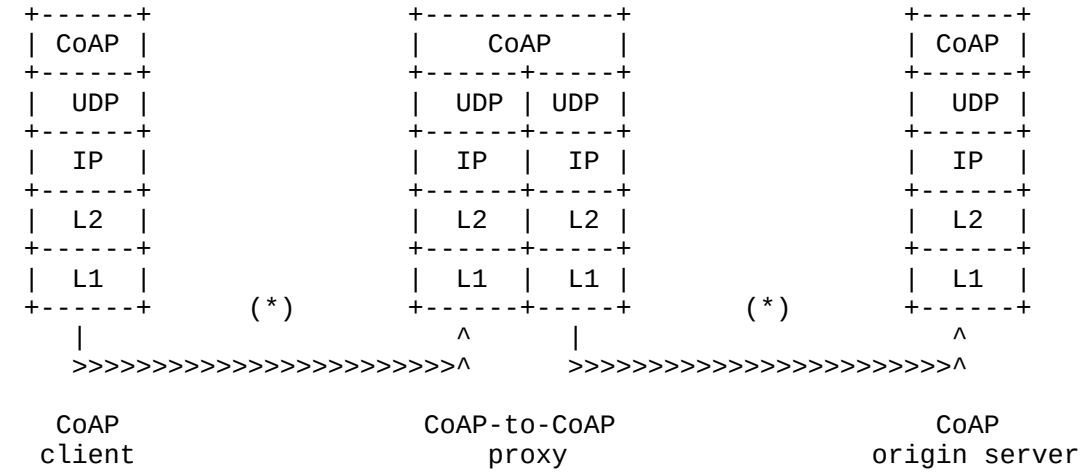

Figure 2: CoAP-to-CoAP proxy scenario. (*) There may be zero or more IP routers between the CoAP client and the CoAP-to-CoAP proxy, and zero or more IP routers between the CoAP-to-CoAP proxy and the CoAP origin server.

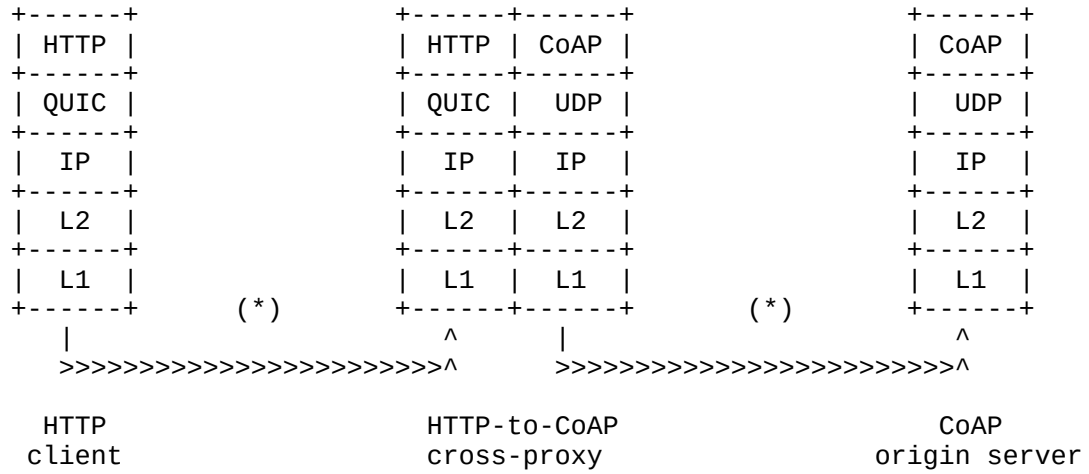

Figure 3: HTTP-to-CoAP proxy scenario. (*) There may be zero or more IP routers between the HTTP client and the HTTP-to-CoAP cross-proxy, and zero or more IP routers between the cross-proxy and the CoAP origin server.

6. Observe

The Observe Option allows a server to send notifications carrying a representation of the current state of a resource to interested clients called observers [RFC7641]. The latter need to initially register at a specific server that they are interested in being notified whenever the resource state changes. There is also work in progress intended to allow a CoAP client to limit notifications to those where the state representation of a resource fulfills certain constraints (e.g., a minimum/maximum value) [I-D.ietf-core-conditional-attributes].

Observe generally provides significant performance benefits, since, after the registration, the client does not have to send a request to receive a notification. This feature is particularly beneficial in delay tolerant environments (e.g., deep space), where end-to-end latency is high, and energy and bandwidth resources may be constrained.

As per the Observe specification, when the time between the two last notifications received by a CoAP client is greater than 128 seconds, it can be concluded that the last one received is also the latest sent by the server. The duration of 128 seconds was chosen as a number greater than the default MAX_LATENCY value of the base CoAP specification. When CoAP is used in delay-tolerant environments

(e.g., deep space), the duration of 128 seconds may be insufficient. In such case, the duration needs to be chosen as a value greater than the MAX_LATENCY of the scenario (See Appendix A of [I-D.gomez-core-coap-bp]).

7. Block-wise transfers

7.1. Overview

There exist two CoAP specifications that define functionality that allows to carry large CoAP payloads (i.e., payloads that do not fit a single packet) by means of block-wise transfers: [RFC7959] and [RFC9177].

RFC 7959 defines the Block1 and Block2 options, whereby, in a block-wise transfer, a CoAP endpoint can only ask for (or send) the next block after the previous block has been transferred. Furthermore, RFC 7959 recommends the use of CON messages. Therefore, communication follows a stop-and-wait pattern.

RFC 9177, which defines the Q-Block1 and Q-Block2 options, is particularly suitable for delay-tolerant environments, as it enables block-wise transfers using NON messages. Thus, blocks can be transmitted serially without having to wait for a response or next request from the remote CoAP peer. Recovery of multiple missing blocks (which can be reported at once in a single CoAP message) is also supported.

The Q-Block1 option is defined for payload-bearing (e.g., POST, PUT, FETCH, PATCH, and iPATCH) requests and their responses. The Q-Block2 option is useful for requests (e.g., GET, POST, PUT, FETCH, PATCH, and iPATCH) and their payload-bearing responses.

7.2. Main related parameters

The following new parameters are defined by RFC 9177, for use with NON messages and the Q-Block1 and Q-Block2 options: MAX_PAYLOADS, NON_TIMEOUT, NON_TIMEOUT_RANDOM, NON_RECEIVE_TIMEOUT, NON_MAX_RETRANSMIT, NON_PROBING_WAIT, and NON_PARTIAL_TIMEOUT.

MAX_PAYLOADS indicates the number of consecutive blocks an endpoint can transmit without eliciting a message from the other endpoint. The default value defined for this parameter is 10, which is in line with the initial window size currently defined for TCP [RFC6928].

TO-DO: MAX_PAYLOADS recommended setting?

NON_TIMEOUT is the minimum time between sending two consecutive sets of MAX_PAYLOADS blocks that correspond to the same body. The actual time between sending two consecutive sets of MAX_PAYLOADS blocks is called NON_TIMEOUT_RANDOM, which is calculated as NON_TIMEOUT * ACK_RANDOM_FACTOR. In RFC 9177, NON_TIMEOUT is defined as having the same value as ACK_TIMEOUT. ACK_RANDOM_FACTOR is set to 1.5, following RFC 7252. As a result, by default, NON_TIMEOUT_RANDOM is equal to a randomly chosen value between 2 and 3 s.

The NON_TIMEOUT_RANDOM inactivity interval described above is introduced to avoid causing congestion due to the transmission of MAX_PAYLOADS itself. As discussed in Section 3.2, in delay-tolerant environments, ACK_TIMEOUT should be set to a value greater than default. However, when CoAP is used in such environments, NON_TIMEOUT, and thus NON_TIMEOUT_RANDOM, need to be adjusted considering the characteristics of the end-to-end path, independent of ACK_TIMEOUT.

NON_RECEIVE_TIMEOUT is the initial time that a receiver will wait for a missing block within MAX_PAYLOADS before requesting retransmission for the first time. Every time the missing payload is re-requested, the time to wait value doubles. NON_RECEIVE_TIMEOUT has a default value of 2*NON_TIMEOUT. As described earlier, when CoAP is used in delay-tolerant environments, NON_TIMEOUT needs to be adjusted considering the characteristics of the end-to-end path.

NON_MAX_RETRANSMIT is the maximum number of times a request for the retransmission of missing payloads can occur without a response from the remote peer. By default, NON_MAX_RETRANSMIT has the same value as MAX_RETRANSMIT (Section 4.8 of [RFC7252]). Accordingly, when CoAP is used in delay-tolerant environments (e.g., deep space), the same considerations regarding MAX_RETRANSMIT in Section 3.2 apply to NON_MAX_RETRANSMIT as well. That is, when CoAP is used in space, while the default value for this parameter is 4, it may be suitable to set this parameter to a value lower than the default one.

8. Message aggregation

The CoAP over BP specification introduces the CoAP Payload-length option, which allows CoAP messages destined to the same endpoint to be aggregated and carried as the payload of a single encapsulating lower-layer data unit [I-D.gomez-core-coap-bp]. When CoAP is used over UDP (as is the focus of the present document), the encapsulating lower-layer data unit is a UDP datagram.

A concatenation of messages that carry the Payload-length option is called an Aggregate message [I-D.gomez-core-coap-bp]. In some scenarios, message aggregation may be compatible with application requirements, while allowing to reduce protocol overhead and increase node and network performance.

9. CoAP group communication

In CoAP group communication, a client sends multicast CoAP request messages over UDP/IP multicast as default transport. Each server in the target destination group sends a response message back to the client over UDP/IP unicast, although a server can suppress its response for several reasons (see Section 3.1.2 of [I-D.ietf-core-groupcomm-bis]).

[I-D.ietf-core-groupcomm-bis] defines the minimum time between reuse of Token values for different group requests, MIN_TOKEN_REUSE_TIME, to be greater than:

MIN_TOKEN_REUSE_TIME = (NON_LIFETIME + MAX_LATENCY + MAX_SERVER_RESPONSE_DELAY)

where MAX_SERVER_RESPONSE_DELAY is the expected maximum response delay over all servers that the client can send a CoAP group request to. [I-D.ietf-core-groupcomm-bis] states that, "using the default CoAP parameters, the Token reuse time MUST be greater than 250 seconds plus MAX_SERVER_RESPONSE_DELAY".

[I-D.ietf-core-groupcomm-bis] also adds that, while a possible approach is to generate a new unique Token for every new group request, if a client has to reuse Token values for some reason, MAX_SERVER_RESPONSE_DELAY = 250 seconds is a suitable value, therefore leading to a time between Token reuses greater than MIN_TOKEN_REUSE_TIME = 500 seconds. However, in a delay-tolerant scenario, MIN_TOKEN_REUSE_TIME needs to be determined considering the latency of that scenario.

10. Security

The base CoAP specification defines a binding to Datagram Transport Layer Security (DTLS) [RFC7252][RFC9147]. There are four possible DTLS security modes: NoSec, PreSharedKey, RawPublicKey, and Certificate. The NoSec and RawPublicKey modes are mandatory to implement.

Subsequently, Object Security for Constrained RESTful Environments (OSCORE) was specified [RFC8613]. OSCORE is a security protocol that allows to protect an application-layer data payload end-to-end, even

in the presence of untrusted proxies in the path between two endpoints. The Group OSCORE protocol is also being used to secure CoAP group communication [I-D.ietf-core-oscore-groupcomm], in contrast with the initial CoAP group communication specification [RFC 7390], which assumed that CoAP over IP multicast was not encrypted, nor authenticated, nor access controlled.

In OSCORE, the communicating endpoints require a shared security context. An interesting aspect of OSCORE for delay-tolerant environments (e.g., deep space) is that, if the materials used to establish such context are pre-shared, there is no initial handshake prior to actual communication, thus avoiding a significant latency penalty.

In order to offer protection against replay attacks, OSCORE uses by default an anti-replay sliding window, with a window size of 32 [RFC 8613]. If a greater window size is deemed necessary (e.g., due to high latency in an intended scenario), that window size needs to be known by both sender and receiver at the moment of security context establishment.

11. Forward Error Correction

As of the writing, no proposal has been made to add support of Forward Error Correction (FEC) to CoAP. However, considering the significant latency penalty of delay-tolerant space environments (e.g., deep space), FEC might allow to reduce the probability of incurring additional latency (due to retries) in order to sucessfully deliver a message to its intended destination.

12. IANA Considerations

This document has no IANA considerations

13. Security Considerations

TO-DO

14. Acknowledgments

Marisa Catalan and Julia Igual from i2cat contributed to this document.

Carles Gomez has been funded in part by the Spanish Government through project PID2019-106808RA-I00, through project MCIU/AEI/10.13039/501100011033/FEDER/UE PID2023- 146378NB-I00, and by Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya 2017 through grant SGR 376 and 2021 throught grant SGR 00330.

The authors would like to thank (in alphabetical order) Christian Amsuess, Marc Blanchet, Carsten Bormann, Jaime Jimenez, Achim Kraus, and Marco Tiloca for useful considerations, reviews and comments.

15. References

15.1. Normative References

[I-D.gomez-core-coap-bp]
Gomez, C. and A. Calveras, "Constrained Application Protocol (CoAP) over Bundle Protocol (BP)", Work in Progress, Internet-Draft, draft-gomez-core-coap-bp-03, 19 January 2025, <https://datatracker.ietf.org/doc/html/draft-gomez-core-coap-bp-03>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E. and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-13, 24 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-13>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <https://www.rfc-editor.org/info/rfc6928>.

[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>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/info/rfc8323>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[RFC9177] Boucadair, M. and J. Shallow, "Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission", RFC 9177, DOI 10.17487/RFC9177, March 2022, <https://www.rfc-editor.org/info/rfc9177>.

15.2. Informative References

[Blanchet] M. Blanchet, "Earth-Mars Communication Windows Usage Study", 2024.

[I-D.huitema-quic-in-space] Huitema, C. and M. Blanchet, "QUIC in Space", Work in Progress, Internet-Draft, draft-huitema-quic-in-space-00, 24 September 2023, <https://datatracker.ietf.org/doc/html/draft-huitema-quic-in-space-00>.

[I-D.ietf-core-conditional-attributes]
Silverajan, B., Koster, M., and A. Soloway, "Conditional Query Parameters for CoAP Observe", Work in Progress, Internet-Draft, draft-ietf-core-conditional-attributes-11, 16 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-conditional-attributes-11>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and R. Hglund, "Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-25, 16 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-25>.

[I-D.many-deepspace-ip-assessment]
Blanchet, M., Huitema, C., and D. Bogdanovi, "Revisiting the Use of the IP Protocol Stack in Deep Space: Assessment and Possible Solutions", Work in Progress, Internet-Draft, draft-many-deepspace-ip-assessment-02, 10 September 2024, <https://datatracker.ietf.org/doc/html/draft-many-deepspace-ip-assessment-02>.

[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>.

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, DOI 10.17487/RFC5050, November 2007, <https://www.rfc-editor.org/info/rfc5050>.

[RFC7390] Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for the Constrained Application Protocol (CoAP)", RFC 7390, DOI 10.17487/RFC7390, October 2014, <https://www.rfc-editor.org/info/rfc7390>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9171] Burleigh, S., Fall, K., and E. Birrane, III, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <https://www.rfc-editor.org/info/rfc9171>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[_GPP] 3GPP TR23.700-29, "Technical Specification Group Services and System Aspects; Study on integration of satellite components in the 5G architecture; Phase 3 (Rel-19)", 2024.

Authors' Addresses

Carles Gomez
UPC
C/Esteve Terradas, 7
08860 Castelldefels
Spain
Email: carles.gomez@upc.edu

Sergio Aguilar
Sateliot
C/Berlin 61, Esc A Entresuelo
08029 Barcelona
Spain
Email: sergio.aguilar@sateliot.com