CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 January 2026

C. Gomez
A. Calveras
UPC

July 2025

Constrained Application Protocol (CoAP) over Bundle Protocol (BP)
draft-gomez-core-coap-bp-04

Abstract

The Bundle Protocol (BP) was designed to enable end-to-end communication in challenged networks. The Constrained Application Protocol (CoAP), which was designed for constrained-node networks, may be a suitable application-layer protocol for the scenarios where BP is used. This document specifies how CoAP is carried over BP.

Status of This Memo

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

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

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

This Internet-Draft will expire on 2 January 2026.

Table of Contents

1. Introduction 3
2. Terminology 3
2.1. Requirements language 3
2.2. Background on previous specifications 3
2.3. New terms 4
3. Architecture 4
4. Messages 4
4.1. Messaging model 5
4.2. Single message format 6
4.3. Payload-length option 7
4.3.1. Payload-length option and OSCORE 7
5. Encapsulating bundle 8
6. CoAP parameter settings and related times 9
7. Observe 11
8. Block-wise transfers 12
8.1. Main CoAP block-wise transfer parameters 12
9. Proxying 13
9.1. Proxying scenarios 14
9.2. Proxying over BP 15
9.3. Proxy operation and message aggregation 15
10. URI Scheme 16
11. Securing CoAP over BP 16
12. IANA Considerations 18
12.1. Creation of two new reserved domains in the .arpa name space 18
12.1.1. Domain Name Reservation Considerations 18
12.2. ipn URI Scheme Well-known Service Number for CoAP . . . 19
12.3. CoAP Option Numbers Registry 19
13. Implementation Status 19
13.1. Space CoAP 20
13.2. Other CoAP over BP implementations 22
14. Security Considerations 22
15. Acknowledgments 22
16. References 23
16.1. Normative References 23
16.2. Informative References 25
Appendix A. Reference CoAP parameter values for interplanetary communication 26
Appendix B. Message ID size, EXCHANGE_LIFETIME, and maximum CoAP message rate 30
Authors' Addresses 32

1. Introduction

The Delay-Tolerant Networking (DTN) architecture has been designed to enable communication in challenged networks, which are characterized by long delays, intermittent connectivity, and high error rates, among other constraints [RFC4838][RFC7228]. DTN was mainly intended for deep space communication (e.g., to enable an Interplanetary Internet). However, it is also applicable to enable communication on Earth in environments exhibiting relatively similar features, such as sensor networks or temporarily disconnected areas.

The Bundle Protocol (BP) is the fundamental component of DTN. BP is a message-oriented protocol that operates as a store-carry-forward overlay atop the transport-layer protocols of a number of constituent networks [RFC9171]. The protocol data unit of BP is called a bundle. Application-layer functionality runs atop BP.

The Constrained Application Protocol (CoAP) is an application-layer protocol that was specifically designed for constrained-node networks [RFC7252][RFC7228], which are typical in Internet of Things (IoT) scenarios. Such environments are often characterized by significantly constrained node and network features, including low computational capacity, limited energy availability (which often leads to the use of duty-cycled links), low bandwidth, high latency, and high loss rates. Accordingly, CoAP offers several features, which are also suitable for DTN, including lightweight operation, asynchronous message exchanges, and a significant degree of flexibility, based on RESTful principles.

The present document specifies how CoAP is carried over 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.

2.2. Background on previous specifications

The reader is expected to be familiar with the terms and concepts defined by the DTN main specifications (e.g., [RFC4838], [RFC9171], and [RFC9172]), and the CoAP main specifications (e.g., [RFC7252], [RFC7641], [RFC7959], [RFC8323], and [RFC9177]).

2.3. New terms

Single message: a CoAP message, as defined in RFC 7252. In CoAP over BP, a Single message is carried as the block-type-specific data field of the Bundle Payload Block of the encapsulating bundle.

Aggregate message: a concatenation of Single messages that carry the Payload-length option (see Section 4.3). In CoAP over BP, an Aggregate message is carried as the block-type-specific data field of the Bundle Payload Block of the encapsulating bundle.

3. Architecture

Figure 1 illustrates the protocol stack model for CoAP over BP. (Note: this figure is the same as Figure 1 of RFC 9171, except for the indication of CoAP's location in the protocol stack model.) In this model, CoAP entities exchange application-layer messages carried by BP over an end-to-end path composed of a number of constituent networks.

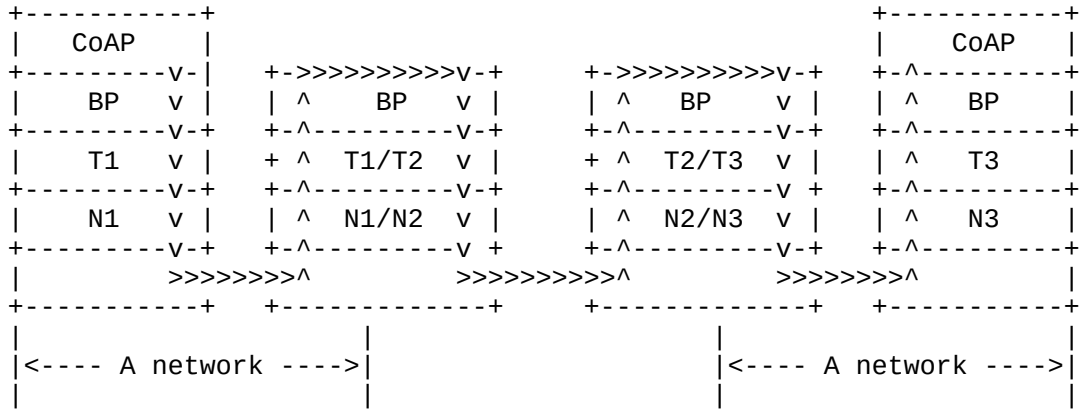

Figure 1: BP and CoAP in the protocol stack model

4. Messages

4.1. Messaging model

The CoAP base specification was produced assuming UDP as the underlying transport-layer protocol [RFC 7252]. Like UDP, BP is a message-oriented protocol. Furthermore, BP does not provide bundle retransmission. Therefore, when CoAP is used over BP, the same messaging model defined for CoAP in RFC 7252 is used, and the CoAP signaling messages defined in RFC 8323 (which are intended for use over reliable transports) MUST NOT be used.

Figure 2 shows the two-sublayer structure of CoAP, when used over BP.

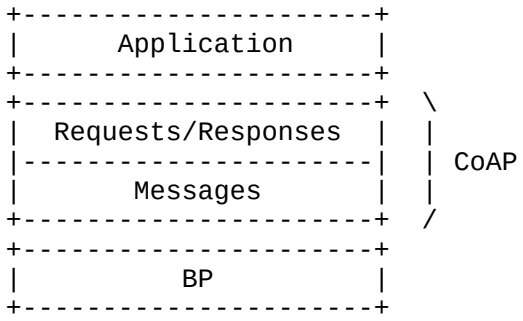

Figure 2: Abstract Layering of CoAP over BP

CoAP follows a client/server model, whereby a client may request an action on a resource on a server. Upon receipt of a request, the server sends a response, including a response code, which may also include a resource representation. (Note that, if a request includes the "No-Response" option [RFC7967], the server may suppress the response.) Requests and responses are encapsulated in messages.

CoAP defines four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgment (ACK), and Reset (RST). CON messages elicit ACKs, whereas NON messages do not. For CON messages, CoAP uses stop-and-wait retransmission with exponential back-off. A RST message is sent by a CoAP endpoint that has received a message but is unable to process it.

When CoAP is used over BP, a source bundle node MAY set the "request reporting of bundle delivery" flag in the bundle’s status report request field of a bundle that encapsulates a CoAP CON message. Upon receipt of a bundle that carries a CoAP CON message with the "request reporting of bundle delivery" flag set, the receiver MAY opt to only send the corresponding bundle delivery status report and omit sending a bundle encapsulating a CoAP ACK message, if and only if the CoAP

ACK message does not carry a payload. In that case, if the CoAP CON message sender receives the status report sent in response to its bundle-encapsulated CON message, it MUST assume that the status report serves as CoAP ACK for the CON message.

(Note: the assumption is that the status report size is shorter than the size of a bundle encapsulating a CoAP ACK message that does not carry a payload. To be further confirmed.)

4.2. Single message format

In CoAP over BP, the format of a Single message (Figure 4) is the same as the CoAP message format defined in RFC 7252 (Figure 3), except for the Message ID size, which is increased to 24 bits for CoAP over BP. The reason for this change is avoiding a severe limitation on the number of messages a sender can send per time unit, considering the latency values in the environments where CoAP over BP may be used, and that, as stated in RFC 7252, "the same Message ID MUST NOT be reused (in communicating with the same endpoint) within the EXCHANGE_LIFETIME". See Appendix B for further details.

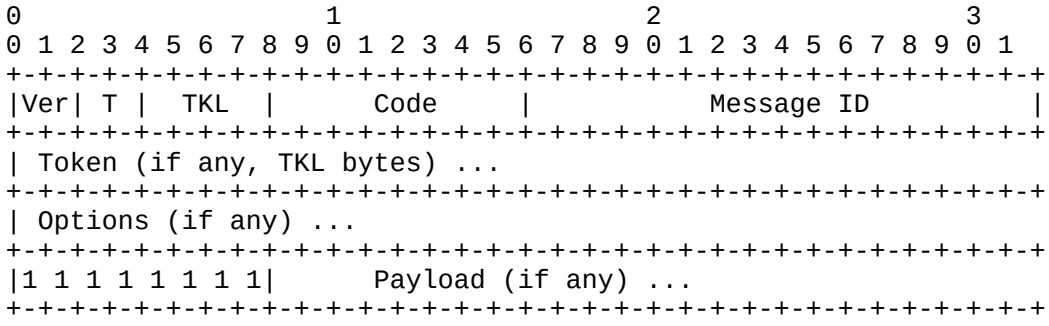

Figure 3: CoAP Message Format as defined in RFC 7252

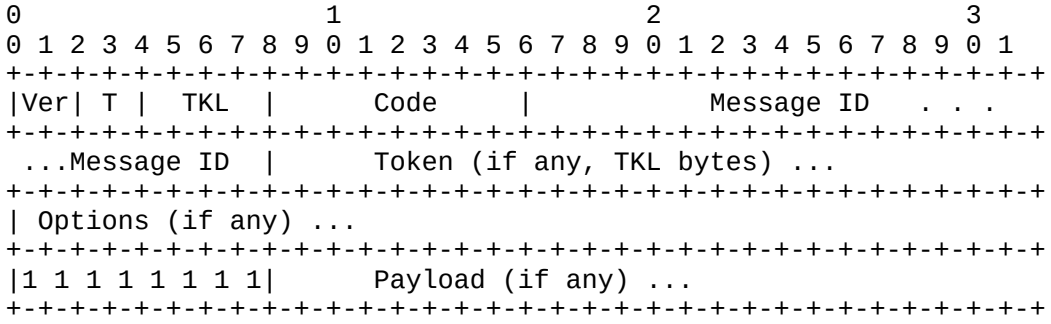

Figure 4: Single Message Format over BP

4.3. Payload-length option

CoAP messages destined to the same endpoint MAY be aggregated and carried as the payload of the underlying protocol data unit.

An Aggregate message is a concatenation of Single messages that carry the Payload-length option. The Payload-length option defined in this subsection (see Figure 5, which extends "Table 4: Options" of [RFC7252]) indicates the size of the payload of a CoAP message. The option value is an integer number of bytes. The Payload-length option is critical, Unsafe-to-forward, and not repeatable.

No.	C	U	N	R	Name	Format	Length	Default
TBD	x	x	-		Payload-length	uint	0 or more	(none)

C = Critical, U = Unsafe, N = NoCacheKey, R = Repeatable
(*) See below.

Figure 5: The Payload-length option

4.3.1. Payload-length option and OSCORE

The Payload-length option is a "Class U" (i.e., "unprotected") option for Object Security for Constrained RESTful Environments (OSCORE) [RFC 8613].

If a CoAP message is intended to carry the Payload-length option, while benefitting from OSCORE protection, first an OSCORE message is built using the CoAP message (without the Payload-length option) as an input, as described in RFC 8613. After that, the Payload-length option is inserted in the OSCORE message, where the payload size indicated by the Payload-length option corresponds to the OSCORE message payload size.

OSCORE messages carrying the Payload-length option and being destined to the same endpoint MAY be aggregated and carried as the payload of the underlying protocol data unit.

5. Encapsulating bundle

In order to transmit a CoAP message (either a Single message or an Aggregate message) over BP, the CoAP message MUST be carried as the block-type-specific data field of the Bundle Payload Block (block type 1) of an encapsulating bundle.

The lifetime field of the bundle encapsulating a CON Single message MUST be set to EXCHANGE_LIFETIME (see Section 6). The lifetime field of the bundle encapsulating a NON Single message MUST be set to NON_LIFETIME (see Section 6).

For Aggregate messages:

- If an Aggregate message only comprises CON messages, the lifetime field of the encapsulating bundle is set to EXCHANGE_LIFETIME + MAX_AGGR_DELAY. (Note: MAX_AGGR_DELAY indicates the maximum time since the first Single message belonging to an Aggregate message is generated until the Aggregate message is passed to the BP layer.)

- If an Aggregate message comprises only NON messages, the lifetime field of the encapsulating bundle is set to NON_LIFETIME + MAX_AGGR_DELAY.

- If an Aggregate message comprises at least one CON message and one NON message, the lifetime field of the encapsulating bundle is set to the max(imum of EXCHANGE_LIFETIME, and NON_LIFETIME)+ MAX_AGGR_DELAY.

In some cases, upon receipt of a CoAP message, the receiving endpoint needs to transmit a CoAP message in response to the sender. The Destination EID and Source Node ID fields of the primary bundle block of the bundle encapsulating such a CoAP message sent as a response SHALL be set as follows:

Destination EID: The Destination EID SHALL be identical to the Source Node ID of the bundle encapsulating the received CoAP message that produces the response.

Source Node ID: The Source Node ID SHALL be identical to the Destination EID of the bundle encapsulating the received CoAP message that produces the response.

6. CoAP parameter settings and related times

This section discusses the main CoAP parameters and times that are relevant in the environments where BP may be used. (Note that the complete set of parameters, assumptions, default values, and related times in CoAP can be found in Section 4.8 of RFC 7252.)

Most of these CoAP parameters and times are relevant for CON messages. Note that, in some scenarios, the protocols operating below BP may support reliability and congestion control. In that case, using NON messages might suffice to achieve a reasonable degree of reliability. The congestion control considerations for NON message transmission would still apply, though (see Sections 4.7 and 4.8 of RFC 7252).

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 [RFC 7252].

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.

For CoAP over BP, ACK_TIMEOUT should be set to a value of at least the expected RTT, which may be of an order of magnitude several times greater than the default one (see Appendix A).

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 environments where BP is used, it may be suitable to set this parameter to a value lower than the default one (see Appendix A).

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 in non-challenged environments. Therefore, in environments where BP is used, MAX_LATENCY may need to be increased by at least 2-3 orders of magnitude.

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 environments where BP is used.

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 challenged environments (e.g., deep space), and considering the increased values for protocol parameters and network characteristics described above, EXCHANGE_LIFETIME will be at least 2 (and perhaps a greater number of) orders of magnitude greater than the default one (see Appendix A).

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 challenged environments (e.g, deep space) it may need to be increased by 2-3 orders of magnitude.

Note that CoAP 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.

7. Observe

The CoAP 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) [draft-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 environments 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 over BP, determining whether a notification was sent by the server later than another notification MUST be performed based on the creation timestamps of the corresponding bundles encapsulating the two notifications. The duration of 128 seconds may be insufficient in many scenarios. In such cases, the duration needs to be chosen as a value greater than the MAX_LATENCY of the scenario (see Appendix A).

8. Block-wise transfers

CoAP supports functionality that allows carrying large payloads by means of block-wise transfers [RFC7959], [RFC9177]. BP also supports fragmentation and reassembly functionality. RFC 7959 states, in the context of fragmentation and reassembly functionality being available at several protocol stack layers, that "the fragmentation/reassembly process burdens the lower layers with conversation state that is better managed in the application layer". However, an implicit assumption in RFC 7959 is that details on the data unit sizes that can be carried over the different links of an end-to-end path are known in advance by the sender.

When CoAP is used over BP, CoAP block-wise transfers MAY be used if the source knows in advance the duration and type of expected contacts (e.g., scheduled or predicted) between the BP nodes that will forward the bundles from the source bundle node to the destination bundle node. This does not preclude the use of BP fragmentation and reassembly when deemed necessary.

There exist two CoAP specifications that allow to perform block-wise transfers: [RFC7959] and [RFC9177].

As per RFC 7959, 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, which is not suitable for environments with long delays.

RFC 9177 is particularly suitable for DTN 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.

8.1. Main CoAP block-wise transfer 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 for deep space?

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 previously, in challenged networks, ACK_TIMEOUT should be set to a value greater than default. When CoAP is used in deep space, 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, in challenged networks, 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 deep space, the same considerations regarding MAX_RETRANSMIT in Section 5 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.

9. 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 the environments where BP

is used.

9.1. Proxying scenarios

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 6 and Figure 7 illustrate the upper-layer protocol stacks for a CoAP-to-CoAP proxy and an HTTP-to-CoAP cross-proxy, when CoAP or HTTP [draft-blanchet-dtn-http-over-bp] run over BP, respectively.

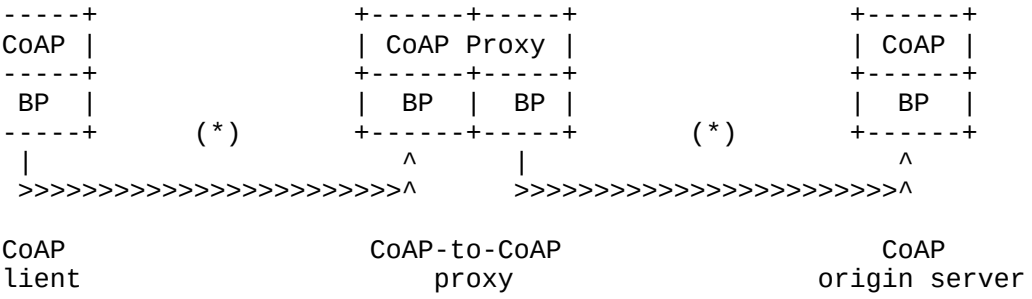

Figure 6: CoAP-to-CoAP proxy scenario. (*) There may be zero or more bundle nodes between the CoAP client and the CoAP-to-CoAP proxy, and zero or more bundle nodes between the CoAP-to-CoAP proxy and the CoAP origin server.

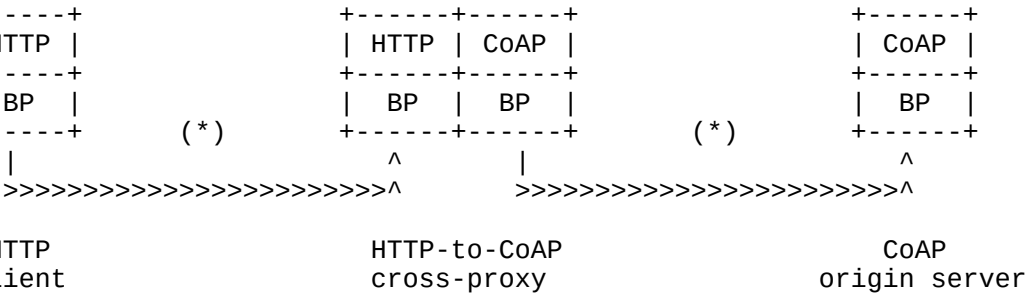

Figure 7: HTTP-to-CoAP proxy scenario. (*) There may be zero or more bundle nodes between the HTTP client and the HTTP-to-CoAP cross-proxy, and zero or more bundle nodes between the cross-proxy and the CoAP origin server.

9.2. Proxying over BP

RFC 7252 states that "When using a proxy, the URI of the resource to request is included in the request, while the destination IP address is set to the address of the proxy". However, that statement assumes the original design of CoAP, where IP is used at the network layer. In CoAP over BP, the destination EID of the encapsulating bundle is set to the EID of the bundle node that implements the CoAP proxy.

Also, RFC 7252 states, when describing forward-proxy operation: "For a CoAP-to-CoAP proxy, the origin server's IP address and port are determined by the authority component of the request URI". In CoAP over BP, the authority component of the request URI provides the origin server's EID.

9.3. Proxy operation and message aggregation

When a proxy that supports the Payload-length option receives an Aggregate message, it disaggregates the Single messages the Aggregate message is composed of. Then, the proxy operates normally (i.e., as described in [RFC 7252]) upon each Single message.

A proxy MAY aggregate messages that are destined to the same endpoint, even if they are originally triggered by different endpoints. For example, in Figure 8, if Client 2 sends a request targetting a resource at Origin server X and it also sends another request targetting another resource at Origin server Y, the subsequent responses to Client 2 may be aggregated by the proxy. In another example, if Client 2 and Client 3 send requests targetting resources at Origin server Z, and the proxy cannot service those responses from a cache, the proxy may aggregate the requests that will be sent to Origin server Z.

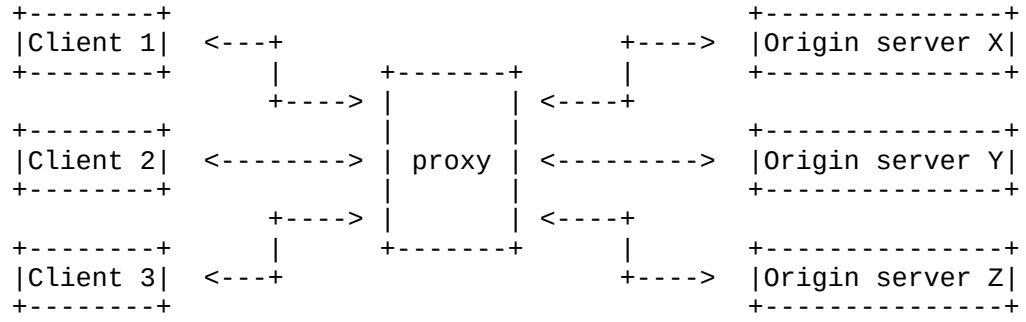

Figure 8: Proxy-based scenario including several clients and origin servers.

10. URI Scheme

The URI scheme for CoAP over BP is "coap" as per the recommendation of Section 6 of [draft-ietf-core-transport-indication]. The "coap" scheme is defined in Section 6 of [RFC7252].

When the endpoint ID of the target resource is based on the "dtn" scheme, the authority component of the URI is formed as the reg-name of the endpoint ID, followed by .dtn.arpa.

When the endpoint ID of the target resource is based on the "ipn" scheme, the authority component of the URI is formed as the service-nbr, followed by the nbr-delim (".") and the node-nbr of the endpoint ID, followed by .ipn.arpa.

User information and port are always absent with the URI scheme used in CoAP over BP.

For example, under the rules of Section 6 of [RFC7252], the URI of a request for the discovery resource of a CoAP over BP entity with endpoint ID dtn://JupiterSensor would be:

coap://JupiterSensor.dtn.arpa/.well-known/core

Similarly, the URI of a request for the discovery resource of a CoAP over BP entity with endpoint ID ipn:81.2 would be:

coap://2.81.ipn.arpa/.well-known/core

TO-DO: request a Well-known Service Number for CoAP in the ipn URI Scheme Well-known Service Numbers for BPv7 registry [draft-ietf-dtn-ipn-update].

11. Securing CoAP over BP

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 for CoAP 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 [draft-ietf-core-oscore-groupcomm] is used to secure CoAP group communication [draft-ietf-core-groupcomm-bis].

In OSCORE, the communicating endpoints require a shared security context. An interesting aspect of OSCORE for the environments where BP is used 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 contrast, DTLS does require an initial handshake. For this reason, the use of DTLS to secure CoAP over BP is generally NOT RECOMMENDED, possible exceptions being environments where the latency penalty is considered acceptable.

On the other hand, Bundle Protocol Security (BPSec) [RFC 9172] provides security services for BP bundles, allowing to protect (with integrity and/or confidentiality) one or more blocks of a bundle. BPSec may be used to provide end-to-end protection between the bundle source and the bundle destination.

When CoAP is carried over BP, the CoAP message will be carried as the block-type-specific data field of the Bundle Payload Block (block type 1) of an encapsulating bundle. If OSCORE is used to protect CoAP, only the CoAP message payload, one CoAP message header field, and some of the CoAP options are protected. Currently, all CoAP message fields that are protected by OSCORE are provided with confidentiality and integrity protection. BPSec allows to protect all fields of the carried CoAP message. However, in the context of CoAP over BP, the scope of BPSec protection is delimited by a bundle node implementing a CoAP endpoint at the application layer, including a CoAP proxy. Therefore, when one or more CoAP proxies are present between a CoAP client and a CoAP origin server, BPSec cannot ensure the protection of application-layer data between those two CoAP endpoints. In that case, OSCORE SHOULD be used to protect application-layer data between the two CoAP endpoints. Note that, in some scenarios, a CoAP client might not be aware that it is communicating with a reverse-proxy (instead of the origin CoAP server).

In scenarios without CoAP proxies, both OSCORE or BPSec MAY be used to provide end-to-end application-layer data protection. As discussed above, BPSec allows to protect all fields of the carried CoAP message.

TO-DO: any reason why both OSCORE *and* BPSec should be simultaneously used in a scenario known to be proxy-less?

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. Note that BP provides additional protection against replay attacks, since a bundle includes a creation timestamp and a lifetime field. If the bundle is replayed outside of its lifetime, the bundle will be discarded and the replay attack will fail (see Section 8.2.4 of RFC 9172).

TO-DO: security requirements of CoAP requests and responses over BP.

12. IANA Considerations

12.1. Creation of two new reserved domains in the .arpa name space

IANA is asked to create two new reserved domain names in the .arpa name space as described in [RFC6761]: the suffixes .dtn.arpa and .ipn.arpa.

The expectation for application software is that no DNS resolution is attempted; instead, the prefix is processed into an endpoint ID, and any operation on that endpoint ID is pointed to the BP node(s) registered in that endpoint ID.

12.1.1. Domain Name Reservation Considerations

The Domain Reservation Considerations from Section 5 of [RFC6761] for both domain names (.dtn.arpa and .ipn.arpa) are:

* Users: users are not expected to recognize those names as special.

* Application Software: application software is expected to pass those names on to their CoAP over BP implementation. CoAP over BP implementations are expected to recognize those names as BP endpoint IDs and MUST NOT pass them on to DNS-based resolvers (unless the name resolution API happens to explicitly support resolution into endpoint ID, see below).

* Name resolution APIs and libraries: name resolution APIs and libraries MAY indicate that .dtn.arpa and .ipn.arpa names resolve to the endpoint ID encoded inside them (but no details for this are specified in known resolution APIs or libraries). Otherwise, they SHOULD report them as NXDOMAIN.

* Caching DNS Servers: caching DNS servers MAY recognize the special domains and report them as NXDOMAIN. Otherwise, they will cache the .arpa DNS servers’ responses.

* Authoritative DNS Servers: authoritative DNS servers MAY recognize the special domains and report them as NXDOMAIN.

* DNS Server Operators: No impact on DNS server operators is expected.

* DNS Registries/Registrars: Any changes to .dtn.arpa or .ipn.arpa require updates to this document and the corresponding process through IANA.

12.2. ipn URI Scheme Well-known Service Number for CoAP

IANA is requested to assign a Well-known Service Number for CoAP in the ipn URI Scheme Well-known Service Numbers for BPv7 registry [draft-ietf-dtn-ipn-update].

12.3. CoAP Option Numbers Registry

IANA is kindly requested to add the Payload-length option to the CoAP Option Numbers registry:

Number	Name	Reference
TBD1	Payload-length	[[this document]]

Figure 9: CoAP option number assignment for the Payload-length option.

13. Implementation Status

(Note to the RFC Editor: please remove this entire section before publication [RFC 7942].)

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

13.1. Space CoAP

- Organization responsible for the implementation: Universitat Politecnica de Catalunya (UPC).

- Implementation name: Space CoAP.

- Links: https://github.com/ENTEL-WNG/CoAPoverBP-no-proxy-version https://github.com/ENTEL-WNG/CoAPoverBP-proxy-version https://github.com/ENTEL-WNG/CoAPoverBP-proxy-version-BP

- Overview: three CoAP over BP scenarios are supported, based on modifications made on top of the aiocoap library (CoAP implementation) and the uD3TN library (BP implementation):

  o CoAPoverBP-no-proxy-version: direct integration of CoAP over BP, where both the CoAP client and server run directly on bundle nodes. CoAP messages are buffered, optionally aggregated and then encapsulated in BP bundles. Persistent storage and scheduled contact plans manage delayed delivery. No CoAP proxying is used. Message creation, aggregation and parsing happen entirely within the bundle nodes.

  o CoAPoverBP-proxy-version: CoAP-to-BP proxy nodes that bridge conventional CoAP-over-UDP clients and servers with a BP network. CoAP participants communicate via UDP with local proxy applications running on bundle nodes. These proxies are responsible for encapsulating CoAP messages into BP bundles for transmission across the DTN and vice versa. Each proxy preserves the CoAP Message ID and Token hop-based while adding a custom Payload-length option to enable message aggregation and correct reassembly. Aggregate messages are stored, carried and forwarded through a uD3TN-based DTN topology with scheduled contact plans and persistent storage.

  o CoAPoverBP-proxy-version-BP: CoAP-over-BP proxy architecture where both CoAP client and server operate directly on bundle nodes and a dedicated proxy mediates CoAP over BP. The proxy performs forwarding of CoAP messages with hop-by-hop based Message ID and Token matching. CoAP messages are buffered and optionally aggregated, with each message tagged using a custom Payload-length

option to allow for message disaggregation on the receiving side. All forwarding, storage and delay simulation is managed using uD3TN with persistent storage and scheduled contacts between nodes.

- Implementation level of maturity: research

- Coverage:

  - Two proxy and one non-proxy scenario

  - CoAP Client and Server with multiple Bundle Nodes

  - Contact plan scheduling for delay simulation

  - Persistent storage on bundle node with bundle decision maker

  - PUT, POST, GET, DELETE methods

  - NON and CON message handling

  - Hop-based token and MID matching. Round Trip is Piggybacked for simplicity

  - Extended Message ID to 24 bits and timers according to draft-gomez-core-coap-bp-03

  - Message aggregation and payload_length option field

  - Space CoAP Lua dissector

  - CoAP over BP URI Scheme

- Licensing: This project incorporates code from several open source libraries and includes original code and modifications.

The project code is licensed under the GNU Affero General Public License Version 3 (AGPLv3). See the LICENSE file in the root of this repository for the full text. The project includes modified files from the aiocoap library (https://github.com/chrysn/aiocoap), originally developed by Christian Amsuess and contributors. Modifications were made to implement draft-gomez-core-coap-bp-03. All changes are clearly marked in the source files with inline comments "experimental for draft-gomez-core-coap-bp-03".

aiocoap Library: This project includes code from the aiocoap library, which is licensed under the BSD 3-Clause License. The full text of this license can be found in the LICENSE folder in the aiocoap library.

uD3TN Library: This project includes code from the uD3TN library, which is licensed under the AGPLv3 License. The full text of this license can be found in the LICENSE file in the uD3TN library.

- Contact information:

  o Michael Karpov: michael.karpov@estudiantat.upc.edu - Main developer

  o Anna Calveras: anna.calveras@upc.edu - Project supervisor

13.2. Other CoAP over BP implementations

Other implementations of CoAP over BP have been reported over time [taracoap][boap]. However, please note that such implementations preceded the creation of the initial version of the present document, therefore they were not intended as implementations of this document.

14. Security Considerations

The Payload-length option is "Class U" for OSCORE, therefore this option, which conveys the payload size of a message, cannot be protected by means of OSCORE. A possible risk is that, even if an OSCORE message payload is protected, an attacker might try to infer some features of the communication between the involved endpoints based on the payload size of each message.

One possible solution might be based on not using the Payload-length option, at the expense of the inability of benefitting from message aggregation. On the other hand, lower-layer security (e.g., BPSec) may help mitigate the risk.

15. Acknowledgments

The authors would like to thank (in alphabetical order) Christian Amsuess, Edward J. Birrane, Marc Blanchet, Carsten Bormann, Scott Burleigh, Joshua Deaton, Jaime Jimenez, Achim Kraus, Bilhanan Silverajan, Brian Sipos, Rick Taylor, Marco Tiloca, Laurent Toutain, Rodney Van Meter, and Magnus Westerlund for useful design considerations, reviews and comments. The authors would also like to thank Michael Karpov as the main developer of the "Space CoAP" research implementation of CoAP over BP.

Carles Gomez and Anna Calveras have been funded in part by MCIU/AEI/10.13039/501100011033/FEDER/UE through project PID2023-146378NB-I00, and by Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya with the grant number 2021 SGR 00330.

16. References

16.1. Normative References

[I-D.ietf-core-transport-indication]
Ams端ss, C. and M. S. Lenders, "CoAP Transport Indication", Work in Progress, Internet-Draft, draft-ietf-core-transport-indication-08, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-transport-indication-08>.

[I-D.ietf-dtn-ipn-update]
Taylor, R. and E. J. Birrane, "Update to the ipn URI scheme", Work in Progress, Internet-Draft, draft-ietf-dtn-ipn-update-14, 27 September 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-ipn-update-14>.

[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>.

[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>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[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>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/info/rfc7595>.

[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>.

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/info/rfc7967>.

[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>.

[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>.

[RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPSec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/info/rfc9172>.

[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>.

16.2. Informative References

[boap] M. Auzias, M. Yves, F. Raimbault, "Coap over bp for a delay-tolerant internet of things", August 2015.

[Conf] S.M. Davidovich, J. Whittington, "Concept for continuous inter-planetary communications", May 1999.

[I-D.blanchet-dtn-http-over-bp] Blanchet, M., "Encapsulation of HTTP over Delay-Tolerant Networks(DTN) using the Bundle Protocol", Work in Progress, Internet-Draft, draft-blanchet-dtn-http-over-bp-03, 31 March 2025, <https://datatracker.ietf.org/doc/html/draft-blanchet-dtn-http-over-bp-03>.

[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-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>.

[I-D.ietf-core-oscore-groupcomm] Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and R. H旦glund, "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>.

[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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[taracoap] M. Adalier, A. Riffel, M. Galvan, B. Johnson, S. Burleigh, "Efficient and secure autonomous communications for deep space missions", March 2020.

Appendix A. Reference CoAP parameter values for interplanetary communication

Figure 7 shows the Round-Trip Time (RTT) between two endpoints on (or close to) different celestial bodies of the Solar System, for the maximum distances between such endpoints [Conf], and in an idealized scenario where communication latency only comprises a propagation delay component. (Note that message storing until the next connectivity opportunity may significantly increase total communication latency.) The RTT also provides a lower bound on (and an approximation of) the ACK_TIMEOUT values required to avoid spurious retransmission timer expiration.

Figure 9 provides approximate EXCHANGE_LIFETIME values that would stem from the use of ACK_TIMEOUT values such as those shown in Figure 5, for MAX_RETRANSMIT=1. (Note that the values provided in Figure 5 are also approximately equal to EXCHANGE_LIFETIME, for MAX_RETRANSMIT=0, under the conditions considered.)

For the sake of comparison, Figure 10 also provides the hypothetical, approximate EXCHANGE_LIFETIME values that would correspond to MAX_RETRANSMIT= 1, but with a retransmission scheme using a constant RTO value for message retries.

Finally, Figure 11 provides the one-way delay for communication between endpoints on (or close to) different celestial bodies of the Solar System, for the maximum distances between such endpoints, and assuming an idealized scenario where communication latency only comprises a propagation delay component. The values in this figure correspond approximately to MAX_LATENCY in the described scenarios.

RTT, ACK_TIMEOUT (or EXCHANGE_LIFETIME, for MAX_RETRANSMIT=0)									
	Sun	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Sun	-	466	727	1,014	1,661	5,444	10,007	20,214	30,288
Mercury	-	-	1,181	1,448	1,968	5,751	10,340	20,548	30,554
Venus	-	-	-	1,735	2,382	6,158	10,741	20,948	30,955
Earth	-	-	-	-	2,642	6,424	11,008	21,215	31,222
Mars	-	-	-	-	-	6,805	11,408	21,615	31,622
Jupiter	-	-	-	-	-	-	14,944	25,151	35,425
Saturn	-	-	-	-	-	-	-	29,220	39,961
Uranus	-	-	-	-	-	-	-	-	50,168
Neptune	-	-	-	-	-	-	-	-	-

Figure 10: ACK_TIMEOUT or EXCHANGE_LIFETIME (for MAX_RETRANSMIT=0), expressed in seconds.

EXCHANGE_LIFETIME (for MAX_RETRANSMIT=1)								
	Sun	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Sun	-	1,397	2,182	3,042	4,983	16,331	30,021	60,642	90,863
Mercury	-	-	3,542	4,343	5,904	17,252	31,021	61,643	91,663
Venus	-	-	-	5,204	7,145	18,473	32,222	62,843	92,864
Earth	-	-	-	-	7,925	19,273	33,023	63,644	93,665
Mars	-	-	-	-	-	20,414	34,224	64,845	94,866
Jupiter	-	-	-	-	-	-	44,831	75,452	106,274
Saturn	-	-	-	-	-	-	-	87,661	119,883
Uranus	-	-	-	-	-	-	-	-	150,504
Neptune	-	-	-	-	-	-	-	-	-

Figure 11: EXCHANGE_LIFETIME (for MAX_RETRANSMIT=1), expressed in seconds.

EXCHANGE_LIFETIME (for MAX_RETRANSMIT=1 and no exponential backoff)									
	Sun	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Sun	-	931	1,454	2,028	3,322	10,888	20,014	40,428	60,575
Mercury	-	-	2,362	2,895	3,936	11,501	20,681	41,095	61,109
Venus	-	-	-	3,469	4,763	12,315	21,482	41,896	61,909
Earth	-	-	-	-	5,284	12,849	22,015	42,429	62,443
Mars	-	-	-	-	-	13,609	22,816	43,230	63,244
Jupiter	-	-	-	-	-	-	29,887	50,301	70,849
Saturn	-	-	-	-	-	-	-	58,440	79,922
Uranus	-	-	-	-	-	-	-	-	100,336
Neptune	-	-	-	-	-	-	-	-	-

Figure 12: Hypothetical EXCHANGE_LIFETIME (for MAX_RETRANSMIT=1), assuming CoAP message retransmission without exponential backoff, expressed in seconds.

MAX_LATENCY									
	Sun	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Sun	-	233	364	507	831	2,722	5,003	10,107	15,144
Mercury	-	-	590	724	984	2,875	5,170	10,274	15,277
Venus	-	-	-	867	1,191	3,079	5,370	10,474	15,477
Earth	-	-	-	-	1,321	3,212	5,504	10,607	15,611
Mars	-	-	-	-	-	3,402	5,704	10,807	15,811
Jupiter	-	-	-	-	-	-	7,472	12,575	17,712
Saturn	-	-	-	-	-	-	-	14,610	19,980
Uranus	-	-	-	-	-	-	-	-	25,084
Neptune	-	-	-	-	-	-	-	-	-

Figure 13: Approximate MAX_LATENCY, expressed in seconds.

Appendix B. Message ID size, EXCHANGE_LIFETIME, and maximum CoAP message rate

With default settings [RFC 7252], and a 16-bit message ID size, CoAP supports the transmission of up to 265 messages/s between a sender and its destination endpoint. If CoAP is used in scenarios involving much greater latencies (e.g., deep space), the greater EXCHANGE_LIFETIME would significantly limit the CoAP message rate. Figure 9 provides the maximum possible message rates for message ID sizes of 16 and 24 bits, and a range of EXCHANGE_LIFETIME values.

Message ID	16 bits	24 bits
#Messages per EXCHANGE_LIFETIME	65,536	16,777,216

Message rate (messages/second)		
EXCHANGE_LIFETIME (s)	Message ID_16 bits	Message_ID 24 bits
247 (default)	265.3 (default)	67,924
500	131.1	33,554
1,000	65.5	16,777
1,500	43.7	11,184
2,000	32.8	8,388
2,500	26.2	6,710
3,000	21.8	5,592
3,500	18.7	4,793
4,000	16.4	4,194
4,500	14.6	3,728
5,000	13.1	3,355
5,500	11.9	3,050
6,000	10.9	2,796
6,500	10.1	2,581
7,000	9.4	2,396
7,500	8.7	2,237
10,000	6.6	1,677
20,000	3.3	838
30,000	2.2	559
40,000	1.6	419
50,000	1.3	335
60,000	1.1	279
70,000	0.9	239
80,000	0.8	209
90,000	0.7	186
100,000	0.7	167
110,000	0.6	152
120,000	0.5	139
130,000	0.5	129
140,000	0.5	119
150,000	0.4	111

Figure 14: Maximum CoAP message rate imposed by the Message ID size and EXCHANGE_LIFETIME, expressed in messages/s.

Authors' Addresses

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

Anna Calveras
UPC
C/Jordi Girona, 1-3
08034 Barcelona
Spain
Email: anna.calveras@upc.edu