CORE
Internet-Draft
Intended status: Standards Track
Expires: July 10, 2021

M. Boucadair
Orange
J. Shallow

January 6, 2021

Constrained Application Protocol (CoAP) Block-Wise Transfer Options for Faster Transmission

draft-ietf-core-new-block-04

Abstract

This document specifies alternative Constrained Application Protocol (CoAP) Block-Wise transfer options: Q-Block1 and Q-Block2 Options.

These options are similar to the CoAP Block1 and Block2 Options, not a replacement for them, but do enable faster transmission rates for large amounts of data with less packet interchanges as well as supporting faster recovery should any of the blocks get lost in transmission.

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 July 10, 2021.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252], although inspired by HTTP, was designed to use UDP instead of TCP. The message layer of CoAP over UDP includes support for reliable delivery, simple congestion control, and flow control. [RFC7959] introduced the CoAP Block1 and Block2 Options to handle data records that cannot fit in a single IP packet, so not having to rely on IP fragmentation and was further updated by [RFC8323] for use over TCP, TLS, and WebSockets.

The CoAP Block1 and Block2 Options work well in environments where there are no or minimal packet losses. These options operate synchronously where each individual block has to be requested and can only ask for (or send) the next block when the request for the previous block has completed. Packet, and hence block transmission rate, is controlled by Round Trip Times (RTTs).

There is a requirement for these blocks of data to be transmitted at higher rates under network conditions where there may be asymmetrical transient packet loss (i.e., responses may get dropped). An example is when a network is subject to a Distributed Denial of Service (DDoS) attack and there is a need for DDoS mitigation agents relying upon CoAP to communicate with each other (e.g., [I-D.ietf-dots-telemetry]). As a reminder, [RFC7959] recommends the use of Confirmable (CON) responses to handle potential packet loss. However, such a recommendation does not work with a flooded pipe DDoS situation.

1.1. Alternative CoAP Block-Wise Transfer Options

This document introduces the CoAP Q-Block1 and Q-Block2 Options. These options are similar in operation to the CoAP Block1 and Block2 Options, respectively. They are not a replacement for them, but have the following benefits:

- They can operate in environments where packet loss is highly asymmetrical.
- They enable faster transmissions of sets of blocks of data with less packet interchanges.
- They support faster recovery should any of the blocks get lost in transmission.
- They support sending an entire body using Non-confirmable (NON) without requiring a response from the peer.

There are the following disadvantages over using CoAP Block1 and Block2 Options:

- Loss of lock-stepping so payloads are not always received in the correct (block ascending) order.
- Additional congestion control measures need to be put in place for NON (Section 6.2).
- To reduce the transmission times for CON transmission of large bodies, NSTART needs to be increased from 1, but this affects congestion control where other parameters need to be tuned (Section 4.7 of [RFC7252]). Such tuning is out of scope of this document.

Using NON messages, the faster transmissions occur as all the blocks can be transmitted serially (as are IP fragmented packets) without having to wait for a response or next request from the remote CoAP peer. Recovery of missing blocks is faster in that multiple missing blocks can be requested in a single CoAP packet. Even if there is asymmetrical packet loss, a body can still be sent and received by the peer whether the body comprises of a single or multiple payloads assuming no recovery is required.

Note that similar performance benefits can be applied to Confirmable messages if the value of NSTART is increased from 1 (Section 4.7 of [RFC7252]). However, the use of Confirmable messages will not work if there is asymmetrical packet loss. Some examples with Confirmable messages are provided in Appendix A.

There is little, if any, benefit of using these options with CoAP running over a reliable connection [RFC8323]. In this case, there is no differentiation between Confirmable and NON as they are not used.

A CoAP endpoint can acknowledge all or a subset of the blocks. Concretely, the receiving CoAP endpoint informs the CoAP sender endpoint either successful receipt or reports on all blocks in the body that have not yet been received. The CoAP sender endpoint will then retransmit only the blocks that have been lost in transmission.

Q-Block1 and Q-Block2 Options can be used instead of Block1 and Block2 Options when the different transmission properties are required. If the new option is not supported by a peer, then transmissions can fall back to using Block1 and Block2 Options, respectively.

The deviations from Block1 and Block2 Options are specified in Section 3. Pointers to appropriate [RFC7959] sections are provided.

The specification refers to the base CoAP methods defined in Section 5.8 of [RFC7252] and the new CoAP methods, FETCH, PATCH, and iPATCH introduced in [RFC8132].

Q-Block1 and Q-Block2 Options are designed to work with Non-confirmable requests and responses, in particular.

1.2. CoAP Response Code (4.08) Usage

This document adds a media type for the 4.08 (Request Entity Incomplete) response defining an additional message format for reporting on payloads using the Q-Block1 Option that are not received by the server.

See Section 4 for more details.

1.3. Applicability Scope

The block-wise transfer specified in [RFC7959] covers the general case, but falls short in situations where packet loss is highly asymmetrical. The mechanism specified in this document provides roughly similar features to the Block1/Block2 Options. It provides additional properties that are tailored towards the intended use case of Non-Confirmable transmission. Concretely, this mechanism primarily targets applications such as DDoS Open Threat Signaling (DOTS) that can't use Confirmable (CON) responses to handle potential packet loss and that support application-specific mechanisms to assess whether the remote peer is able to handle the messages sent by a CoAP endpoint (e.g., DOTS heartbeats in Section 4.7 of [RFC8782]).

The mechanism includes guards to prevent a CoAP agent from overloading the network by adopting an aggressive sending rate. These guards MUST be followed in addition to the existing CoAP congestion control as specified in Section 4.7 of [RFC7252]. See Section 6 for more details.

This mechanism is not intended for general CoAP usage, and any use outside the intended use case should be carefully weighed against the loss of interoperability with generic CoAP applications. It is hoped that the experience gained with this mechanism can feed future extensions of the block-wise mechanism that will both be generally applicable and serve this particular use case.

It is not recommended that these options are used in a NoSec security mode (Section 9 of [RFC7252]) as the source endpoint needs to be trusted. Using OSCORE [RFC8613] does provide a security context and, hence, a trust of the source endpoint. However, using a NoSec

security mode may still be inadequate for reasons discussed in Section 11.

2. Terminology

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 BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers should be familiar with the terms and concepts defined in [RFC7252].

The terms "payload" and "body" are defined in [RFC7959]. The term "payload" is thus used for the content of a single CoAP message (i.e., a single block being transferred), while the term "body" is used for the entire resource representation that is being transferred in a block-wise fashion.

3. The Q-Block1 and Q-Block2 Options

3.1. Properties of Q-Block1 and Q-Block2 Options

The properties of Q-Block1 and Q-Block2 Options are shown in Table 1. The formatting of this table follows the one used in Table 4 of [RFC7252] (Section 5.10). The C, U, N, and R columns indicate the properties Critical, UnSafe, NoCacheKey, and Repeatable defined in Section 5.4 of [RFC7252]. Only Critical and UnSafe columns are marked for the Q-Block1 Option. Critical, UnSafe, and Repeatable columns are marked for the Q-Block2 Option. As these options are UnSafe, NoCacheKey has no meaning and so is marked with a dash.

Number	C	U	N	R	Name	Format	Length	Default
TBA1	x	x	-		Q-Block1	uint	0-3	(none)
TBA2	x	x	-	x	Q-Block2	uint	0-3	(none)

Table 1: CoAP Q-Block1 and Q-Block2 Option Properties

The Q-Block1 and Q-Block2 Options can be present in both the request and response messages. The Q-Block1 Option pertains to the request payload and the Q-Block2 Option pertains to the response payload. The Content-Format Option applies to the body, not to the payload (i.e., it must be the same for all payloads of the same body).

Q-Block1 Option is useful with the payload-bearing POST, PUT, FETCH, PATCH, and iPATCH requests and their responses (2.01 and 2.04).

Q-Block2 Option is useful with GET, POST, PUT, FETCH, PATCH, and iPATCH requests and their payload-bearing responses (2.01, 2.03, 2.04, and 2.05) (Section 5.5 of [RFC7252]).

A CoAP endpoint (or proxy) MUST support either both or neither of the Q-Block1 and Q-Block2 Options.

To indicate support for Q-Block2 responses, the CoAP client MUST include the Q-Block2 Option in a GET or similar request, the Q-Block2 Option in a PUT or similar request, or the Q-Block1 Option in a PUT or similar so that the server knows that the client supports this Q-Block2 functionality should it need to send back a body that spans multiple payloads. Otherwise, the server would use the Block2 Option (if supported) to send back a message body that is too large to fit into a single IP packet [RFC7959].

If Q-Block1 Option is present in a request or Q-Block2 Option in a response (i.e., in that message to the payload of which it pertains), it indicates a block-wise transfer and describes how this specific block-wise payload forms part of the entire body being transferred. If it is present in the opposite direction, it provides additional control on how that payload will be formed or was processed.

Implementation of the Q-Block1 and Q-Block2 Options is intended to be optional. However, when it is present in a CoAP message, it MUST be processed (or the message rejected). Therefore, Q-Block1 and Q-Block2 Options are identified as Critical options.

The Q-Block1 and Q-Block2 Options are unsafe to forward. That is, a CoAP proxy that does not understand the Q-Block1 (or Q-Block2) Option MUST reject the request or response that uses either option.

The Q-Block2 Option is repeatable when requesting retransmission of missing blocks, but not otherwise. Except that case, any request carrying multiple Q-Block1 (or Q-Block2) Options MUST be handled following the procedure specified in Section 5.4.5 of [RFC7252].

The Q-Block1 and Q-Block2 Options, like the Block1 and Block2 Options, are both a class E and a class U in terms of OSCORE processing (Table 2). The Q-Block1 (or Q-Block2) Option MAY be an Inner or Outer option (see Section 4.1 of [RFC8613]). The Inner and Outer values are therefore independent of each other. The Inner option is encrypted and integrity protected between clients and servers, and provides message body identification in case of end-to-end fragmentation of requests. The Outer option is visible to

proxies and labels message bodies in case of hop-by-hop fragmentation of requests.

Number	Name	E	U
TBA1	Q-Block1	x	x
TBA2	Q-Block2	x	x

Table 2: OSCORE Protection of Q-Block1 and Q-Block2 Options

3.2. Structure of Q-Block1 and Q-Block2 Options

The structure of Q-Block1 and Q-Block2 Options follows the structure defined in Section 2.2 of [RFC7959].

There is no default value for the Q-Block1 and Q-Block2 Options. Absence of one of these options is equivalent to an option value of 0 with respect to the value of block number (NUM) and more bit (M) that could be given in the option, i.e., it indicates that the current block is the first and only block of the transfer (block number is set to 0, M is unset). However, in contrast to the explicit value 0, which would indicate a size of the block (SZX) of 0, and thus a size value of 16 bytes, there is no specific explicit size implied by the absence of the option -- the size is left unspecified. (As for any uint, the explicit value 0 is efficiently indicated by a zero-length option; this, therefore, is different in semantics from the absence of the option).

3.3. Using the Q-Block1 Option

The Q-Block1 Option is used when the client wants to send a large amount of data to the server using the POST, PUT, FETCH, PATCH, or iPATCH methods where the data and headers do not fit into a single packet.

When Q-Block1 Option is used, the client MUST include a Request-Tag Option [I-D.ietf-core-echo-request-tag]. The Request-Tag value MUST be the same for all of the requests for the body of data that is being transferred. It is also used to identify a particular payload of a body that needs to be retransmitted. The Request-Tag is opaque, the server still treats it as opaque but the client SHOULD ensure that it is unique for every different body of transmitted data.

> Implementation Note: It is suggested that the client treats the Request-Tag as an unsigned integer of 8 bytes in length. An implementation may want to consider limiting this to 4 bytes to reduce packet overhead size. The initial Request-Tag value should

be randomly generated and then subsequently incremented by the client whenever a new body of data is being transmitted between peers.

Section 3.6 discusses the use of Size1 Option.

For Confirmable transmission, the server continues to acknowledge each packet, but a response is not required (whether separate or piggybacked) until successful receipt of the body or, if some of the payloads are sent as Non-confirmable and have not arrived, a retransmit missing payloads response is needed.

Each individual payload of the body is treated as a new request (see Section 5).

The client MUST send the payloads with the block numbers increasing, starting from zero, until the body is complete (subject to any congestion control (Section 6)). Any missing payloads requested by the server must in addition be separately transmitted with increasing block numbers.

The following Response Codes are used:

2.01 (Created)

This Response Code indicates successful receipt of the entire body and the resource was created.

2.04 (Changed)

This Response Code indicates successful receipt of the entire body and the resource was updated.

2.31 (Continue)

This Response Code can be used to indicate that all of the blocks up to and including the Q-Block1 Option block NUM (all having the M bit set) in the response have been successfully received.

A response using this Response Code SHOULD NOT be generated for every received Q-Block1 Option request. It SHOULD only be generated when all the payload requests are Non-confirmable and MAX_PAYLOADS payloads have been received by the server (Section 6.2).

It SHOULD NOT be generated for CON.

4.00 (Bad Request)

This Response Code MUST be returned if the request does not include both a Request-Tag Option and a Size1 Option but does include a Q-Block1 option.

4.02 (Bad Option)

Either this Response Code or a reset message MUST be returned if the server does not support the Q-Block1 Option.

4.08 (Request Entity Incomplete)

This Response Code returned without Content-Type "application/missing-blocks+cbor-seq" (Section 10.3) is handled as in Section 2.9.2 [RFC7959].

This Response Code returned with Content-Type "application/missing-blocks+cbor-seq" indicates that some of the payloads are missing and need to be resent. The client then retransmits the missing payloads using the same Request-Tag, Size1 and Q-Block1 to specify the block number, SZX, and M bit as appropriate.

The Request-Tag value to use is determined from the token in the 4.08 (Request Entity Incomplete) response and then finding the matching client request which contains the Request-Tag that is being used for this Q-Block1 body.

4.13 (Request Entity Too Large)

This Response Code can be returned under similar conditions to those discussed in Section 2.9.3 of [RFC7959].

This Response Code can be returned if there is insufficient space to create a response PDU with a block size of 16 bytes (SZX = 0) to send back all the response options as appropriate. In this case, the Size1 Option is not included.

If the server has not received all the payloads of a body, but one or more NON payloads have been received, it SHOULD wait for up to NON_RECEIVE_TIMEOUT (Section 6.2) before sending a 4.08 (Request Entity Incomplete) response. Further considerations related to the transmission timings of 4.08 (Request Entity Incomplete) and 2.31 (Continue) Response Codes are discussed in Section 6.2.

If a server receives payloads with different Request-Tags for the same resource, it should continue to process all the bodies as it has no way of determining which is the latest version, or which body, if any, the client is terminating the transmission for.

If the client elects to stop the transmission of a complete body, it SHOULD "forget" all tracked tokens associated with the body's Request-Tag so that a reset message is generated for the invalid token in the 4.08 (Request Entity Incomplete) response. The server on receipt of the reset message SHOULD delete the partial body.

If the server receives a duplicate block with the same Request-Tag, it SHOULD ignore the payload of the packet, but MUST still respond as if the block was received for the first time.

A server SHOULD only maintain a partial body (missing payloads) for up to NON_PARTIAL_TIMEOUT (Section 6.2).

3.4. Using the Q-Block2 Option

In a request for any block number, the M bit unset indicates the request is just for that block. If the M bit is set, this indicates that this is a request for that block and for all of the remaining blocks within the body. If the request includes multiple Q-Block2 Options and these options overlap (e.g., combination of M being set (this and all the later blocks) and being unset (this individual block)) resulting in an individual block being requested multiple times, the server MUST only send back one instance of that block. This behavior is meant to prevent amplification attacks.

The payloads sent back from the server as a response MUST all have the same ETag (Section 5.10.6 of [RFC7252]) for the same body. The server MUST NOT use the same ETag value for different representations of a resource.

The ETag is opaque, the client still treats it as opaque but the server SHOULD ensure that it is unique for every different body of transmitted data.

> Implementation Note: It is suggested that the server treats the ETag as an unsigned integer of 8 bytes in length. An implementation may want to consider limiting this to 4 bytes to reduce packet overhead size. The initial ETag value should be randomly generated and then subsequently incremented by the server whenever a new body of data is being transmitted between peers.

Section 3.6 discusses the use of Size2 Option.

The client may elect to request any detected missing blocks or just ignore the partial body. This decision is implementation specific.

The client SHOULD wait for up to NON_RECEIVE_TIMEOUT (Section 6.2) after the last received payload for NON payloads before issuing a

GET, POST, PUT, FETCH, PATCH, or iPATCH request that contains one or more Q-Block2 Options that define the missing blocks with the M bit unset. Further considerations related to the transmission timing for missing requests are discussed in Section 6.2.

The requested missing block numbers MUST have an increasing block number in each additional Q-Block2 Option with no duplicates. The server SHOULD respond with a 4.00 (Bad Request) to requests not adhering to this behavior.

For Confirmable responses, the client continues to acknowledge each packet. The server will detect failure to send a packet, but the client can issue, after a MAX_TRANSMIT_SPAN delay, a separate GET, POST, PUT, FETCH, PATCH, or iPATCH for any missing blocks as needed.

If the client receives a duplicate block with the same ETag, it SHOULD silently ignore the packet.

A client SHOULD only maintain a partial body (missing payloads) for up to NON_PARTIAL_TIMEOUT (Section 6.2) or as defined by the Max-Age Option (or its default), whichever is the less.

The ETag Option should not be used in the request for missing blocks as the server could respond with a 2.03 (Valid Response) with no payload. It can be used in the request if the client wants to check the freshness of the currently cached body response.

If the server detects part way through a body transfer that the resource data has changed and the server is not maintaining a cached copy of the old data, then the body response SHOULD be restarted with a different ETag Option value. Any subsequent missing block requests MUST be responded to using the latest ETag Option value.

If the server responds during a body update with a different ETag Option value (as the resource representation has changed), then the client should treat the partial body with the old ETag as no longer being fresh.

If the server transmits a new body of data (e.g., a triggered Observe) with a new ETag to the same client as an additional response, the client should remove any partially received body held for a previous ETag for that resource as it is unlikely the missing blocks can be retrieved.

If there is insufficient space to create a response PDU with a block size of 16 bytes (SZX = 0) to send back all the response options as appropriate, a 4.13 (Request Entity Too Large) is returned without the Size1 Option.

3.5. Using Observe and Q-Block2 Options

As the blocks of the body are sent without waiting for acknowledgement of the individual blocks, the Observe value [RFC7641] MUST be the same for all the blocks of the same body.

If the client requests missing blocks, this is treated as a new Request and the Observe Option MUST NOT be included. If the ETag value in the response changes, then the previously received partial body should be considered as not fresh and the whole body re-requested.

3.6. Using Size1 and Size2 Options

Section 4 of [RFC7959] defines two CoAP options: Size1 for indicating the size of the representation transferred in requests and Size2 for indicating the size of the representation transferred in responses.

The Size1 or Size2 option values MUST exactly represent the size of the data on the body so that any missing data can easily be determined.

The Size1 Option MUST be used with the Q-Block1 Option when used in a request. The Size2 Option MUST be used with the Q-Block2 Option when used in a response.

If Size1 or Size2 Options are used, they MUST be used in all payloads of the body and MUST preserve the same value in each of those payloads.

3.7. Using Q-Block1 and Q-Block2 Options Together

The behavior is similar to the one defined in Section 3.3 of [RFC7959] with Q-Block1 substituted for Block1 and Q-Block2 for Block2.

4. The Use of 4.08 (Request Entity Incomplete) Response Code

4.08 (Request Entity Incomplete) Response Code has a new Content-Type "application/missing-blocks+cbor-seq" used to indicate that the server has not received all of the blocks of the request body that it needs to proceed.

Likely causes are the client has not sent all blocks, some blocks were dropped during transmission, or the client has sent them sufficiently long ago that the server has already discarded them.

The data payload of the 4.08 (Request Entity Incomplete) response is encoded as a CBOR Sequence [RFC8742]. There are one or more missing CBOR encoded missing block numbers. The missing block numbers MUST be unique in each 4.08 (Request Entity Incomplete) response when created by the server; the client SHOULD drop any duplicates in the same 4.08 (Request Entity Incomplete) response.

The Content-Format Option (Section 5.10.3 of [RFC7252]) MUST be used in the 4.08 (Request Entity Incomplete) response. It MUST be set to "application/missing-blocks+cbor-seq" (see Section 10.3).

The Concise Data Definition Language [RFC8610] for the data describing these missing blocks is as follows:

```
; This defines an array, the elements of which are to be used
; in a CBOR Sequence:
payload = [+ missing-block-number]
; A unique block number not received:
missing-block-number = uint
```

Figure 1: Structure of the Missing Blocks Payload

The token to use for the response SHOULD be the token that was used in the highest block number received payload. The Q-Block1 Option from the same packet SHOULD be included.

If the size of the 4.08 (Request Entity Incomplete) response packet is larger than that defined by Section 4.6 [RFC7252], then the number of missing blocks MUST be limited so that the response can fit into a single packet. If this is the case, then the server can send subsequent 4.08 (Request Entity Incomplete) responses containing the missing blocks on receipt of a new request providing a missing payload with the same Request-Tag.

The missing blocks MUST be reported in ascending order without any duplicates. The client SHOULD silently drop 4.08 (Request Entity Incomplete) responses not adhering with this behavior.

Implementation Note: Updating the payload without overflowing the overall packet size as each block number can be of varying length needs consideration. It is possible to use Indefinite-Length Arrays (Section 3.2.2 of [RFC8949]), or alternatively limit the array count to 23 so that the initial byte with the array major type and data length in the additional information can be updated with the overall count once the payload count is confirmed. Further restricting the count to MAX_PAYLOADS means that congestion control is less likely to be invoked on the server.

The 4.08 (Request Entity Incomplete) with Content-Type "application/missing-blocks+cbor-seq" SHOULD NOT be used when using Confirmable requests or a reliable connection [RFC8323] as the client will be able to determine that there is a transmission failure of a particular payload and hence that the server is missing that payload.

5. The Use of Tokens

Each new request MUST use a unique Token (Section 4 of [I-D.ietf-core-echo-request-tag]). Additional responses may use the same Token.

Implementation Note: To minimize on the number of tokens that have to be tracked by clients, it is recommended that the bottom 32 bits is kept the same for the same body and the upper 32 bits contains the individual payload number.

Servers continue to treat the token as a unique opaque entity. If an individual payload has to be resent (e.g., requested upon packet loss), then the retransmitted packet is treated as a new request (i.e., the bottom 32 bits must change).

6. Congestion Control

The transmission of the payloads of a body either SHOULD all be Confirmable or all be Non-confirmable. This is meant to simplify the congestion control procedure.

6.1. Confirmable (CON)

Congestion control for CON requests and responses is specified in Section 4.7 of [RFC7252]. For faster transmission rates, NSTART will need to be increased from 1. However, the other CON congestion control parameters will need to be tuned to cover this change. This tuning is out of scope of this document as it is expected that all requests and responses using Q-Block1 and Q-Block2 will be Non-confirmable.

It is implementation specific as to whether there should be any further requests for missing data as there will have been significant transmission failure as individual payloads will have failed after MAX_TRANSMIT_SPAN.

6.2. Non-confirmable (NON)

This document introduces new parameters MAX_PAYLOADS, NON_TIMEOUT, NON_RECEIVE_TIMEOUT, NON_PROBING_WAIT, and NON_PARTIAL_TIMEOUT primarily for use with NON.

MAX_PAYLOADS should be configurable with a default value of 10. Both CoAP endpoints SHOULD have the same value (otherwise there will be transmission delays in one direction) and the value MAY be negotiated between the endpoints to a common value by using a higher level protocol (out of scope of this document). This is the maximum number of payloads that can be transmitted at any one time.

> Note: The default value of 10 is chosen for reasons similar to those discussed in Section 5 of [RFC6928].

NON_TIMEOUT is the maximum period of delay between sending sets of MAX_PAYLOADS payloads for the same body. NON_TIMEOUT has the same value as ACK_TIMEOUT (Section 4.8 of [RFC7252]).

NON_RECEIVE_TIMEOUT is the maximum time to wait for a missing payload before requesting retransmission. NON_RECEIVE_TIMEOUT has a value of twice NON_TIMEOUT.

NON_PROBING_WAIT is used to limit the potential wait needed calculated when using PROBING_WAIT. NON_PROBING_WAIT has the same value as computed for EXCHANGE_LIFETIME (Section 4.8.2 of [RFC7252]).

NON_PARTIAL_TIMEOUT is used for expiring partially received bodies. NON_PARTIAL_TIMEOUT has the same value as computed for EXCHANGE_LIFETIME (Section 4.8.2 of [RFC7252]).

Parameter Name	Default Value
MAX_PAYLOADS	10
NON_TIMEOUT	2 s
NON_RECEIVE_TIMEOUT	4 s
NON_PROBING_WAIT	247 s
NON_PARTIAL_TIMEOUT	247 s

Table 3: Derived Protocol Parameters

PROBING_RATE parameter in CoAP indicates the average data rate that must not be exceeded by a CoAP endpoint in sending to a peer endpoint that does not respond. The single body of blocks will be subjected to PROBING_RATE (Section 4.7 of [RFC7252]), not the individual packets. If the wait time between sending bodies that are not being responded to based on PROBING_RATE exceeds NON_PROBING_WAIT, then the gap time is limited to NON_PROBING_WAIT.

> Note: For the particular DOTS application, PROBING_RATE and other transmission parameters are negotiated between peers. Even when

not negotiated, the DOTS application uses customized defaults as discussed in Section 4.5.2 of [RFC8782].

Each NON 4.08 (Request Entity Incomplete) response is subject to PROBING_RATE.

Each NON GET or FETCH request using Q-Block2 Option is subject to PROBING_RATE.

As the sending of many payloads of a single body may itself cause congestion, it is RECOMMENDED that after transmission of every set of MAX_PAYLOADS payloads of a single body, a delay is introduced of NON_TIMEOUT before sending the next set of payloads to manage potential congestion issues.

If the CoAP peer reports at least one payload has not arrived for each body for at least a 24 hour period and it is known that there are no other network issues over that period, then the value of MAX_PAYLOADS can be reduced by 1 at a time (to a minimum of 1) and the situation re-evaluated for another 24 hour period until there is no report of missing payloads under normal operating conditions. Note that the CoAP peer will not know about the MAX_PAYLOADS change until it is reconfigured. As a consequence, the peer may indicate that there are some missing payloads prior to the actual payload being transmitted as all of its MAX_PAYLOADS payloads have not arrived.

The sending of a set of missing payloads of a body is subject to MAX_PAYLOADS set of payloads.

For Q-Block1 Option, if the server responds with a 2.31 (Continue) Response Code for the latest payload sent, then the client can continue to send the next set of payloads without any delay. If the server responds with a 4.08 (Request Entity Incomplete) Response Code, then the missing payloads SHOULD be retransmitted before going into another NON_TIMEOUT delay prior to sending the next set of payloads.

For the server receiving NON Q-Block1 requests, it SHOULD send back a 2.31 (Continue) or 4.08 (Request Entity Incomplete) Response Code on receipt of the last of the MAX_PAYLOADS payloads to prevent the client unnecessarily delaying. If the last of the MAX_PAYLOADS payloads does not arrive (or the final payload where the M bit is not set does not arrive), then the server SHOULD delay for NON_RECEIVE_TIMEOUT before sending the 4.08 (Request Entity Incomplete) Response Code.

It is possible that the client may start transmitting the next set of MAX_PAYLOADS payloads before the server times out on waiting for the last of the previous MAX_PAYLOADS payloads. On receipt of the first payload from the new set of MAX_PAYLOADS payloads, the server SHOULD send a 4.08 (Request Entity Incomplete) Response Code indicating any missing payloads from any previous MAX_PAYLOADS payloads. Upon receipt of the 4.08 (Request Entity Incomplete) Response Code, the client SHOULD send the missing payloads before continuing to send the remainder of the MAX_PAYLOADS payloads and then go into another NON_TIMEOUT delay prior to sending the next set of payloads.

For the client receiving NON Q-Block2 responses, it SHOULD send a request for the next set of payloads or a request for the missing payloads upon receipt of the last of the MAX_PAYLOADS payloads to prevent the server unnecessarily delaying the transmission of the body. If the last of the MAX_PAYLOADS payloads does not arrive (or the final payload where the M bit is not set does not arrive), then the client SHOULD delay for NON_RECEIVE_TIMEOUT before sending the request for the missing payloads.

The request that the client sends to acknowledge the receipt of all the current set of MAX_PAYLOADS payloads SHOULD contain a special case Q-Block2 Option with NUM set to the first block of the next set of MAX_PAYLOADS payloads and the M bit set to 1. The server SHOULD recognize this special case as a continue request and just continue the transmission of the body (including Observe Option, if appropriate for an unsolicited response) rather than as a request for missing blocks.

The client does not need to acknowledge the receipt of the entire body.

> Note: If there is asymmetric traffic loss causing responses to never get received, a delay of NON_TIMEOUT after every transmission of MAX_PAYLOADS blocks will be observed. The endpoint receiving the body is still likely to receive the entire body.

7. Caching Considerations

Caching block based information is not straight forward in a proxy. For Q-Block1 and Q-Block2 Options, for simplicity it is expected that the proxy will reassemble the body (using any appropriate recovery options for packet loss) before passing on the body to the appropriate CoAP endpoint. This does not preclude an implementation doing a more complex per payload caching, but how to do this is out of the scope of this document. The onward transmission of the body does not require the use of the Q-Block1 or Q-Block2 Options as these

options may not be supported in that link. This means that the proxy must fully support the Q-Block1 and Q-Block2 Options.

How the body is cached in the CoAP client (for Q-Block1 transmissions) or the CoAP server (for Q-Block2 transmissions) is implementation specific.

As the entire body is being cached in the proxy, the Q-Block1 and Q-Block2 Options are removed as part of the block assembly and thus do not reach the cache.

For Q-Block2 responses, the ETag Option value is associated with the data (and onward transmitted to the CoAP client), but is not part of the cache key.

For requests with Q-Block1 Option, the Request-Tag Option is associated with the build up of the body from successive payloads, but is not part of the cache key. For the onward transmission of the body using CoAP, a new Request-Tag SHOULD be generated and used. Ideally this new Request-Tag should replace the client's request Request-Tag.

It is possible that two or more CoAP clients are concurrently updating the same resource through a common proxy to the same CoAP server using Q-Block1 (or Block1) Option. If this is the case, the first client to complete building the body causes that body to start transmitting to the CoAP server with an appropriate Request-Tag value. When the next client completes building the body, any existing partial body transmission to the CoAP server is terminated and the new body representation transmission starts with a new Request-Tag value. Note that it cannot be assumed that the proxy will always receive a complete body from a client.

A proxy that supports Q-Block2 Option MUST be prepared to receive a GET or similar request indicating one or more missing blocks. The proxy will serve from its cache the missing blocks that are available in its cache in the same way a server would send all the appropriate Q-Block2s. If the cache key matching body is not available in the cache, the proxy MUST request the entire body from the CoAP server using the information in the cache key.

How long a CoAP endpoint (or proxy) keeps the body in its cache is implementation specific (e.g., it may be based on Max-Age).

8. HTTP-Mapping Considerations

As a reminder, the basic normative requirements on HTTP/CoAP mappings are defined in Section 10 of [RFC7252]. The implementation guidelines for HTTP/CoAP mappings are elaborated in [RFC8075].

The rules defined in Section 5 of [RFC7959] are to be followed.

9. Examples of Selective Block Recovery

This section provides some sample flows to illustrate the use of Q-Block1 and Q-Block2 Options. Figure 2 lists the conventions that are used in the following subsections.

```
   T: Token value
   O: Observe Option value
   M: Message ID
  RT: Request-Tag
  ET: ETag
 QB1: Q-Block1 Option values NUM/More/SZX
 QB2: Q-Block2 Option values NUM/More/SZX
   \: Trimming long lines
[[]]: Comments
-->X: Message loss (request)
X<--: Message loss (response)
 ...: Passage of time
```

Figure 2: Notations Used in the Figures

9.1. Q-Block1 Option: Non-Confirmable Example

Figure 3 depicts an example of a NON PUT request conveying Q-Block1 Option. All the blocks are received by the server.

```
        CoAP        CoAP
       Client      Server
         |          |
         +--------->| NON PUT /path M:0x01 T:0xf0 RT=10 QB1:0/1/1024
         +--------->| NON PUT /path M:0x02 T:0xf1 RT=10 QB1:1/1/1024
         +--------->| NON PUT /path M:0x03 T:0xf2 RT=10 QB1:2/1/1024
         +--------->| NON PUT /path M:0x04 T:0xf3 RT=10 QB1:3/0/1024
         |<---------+ NON 2.04 M:0xf1 T:0xf3
         |   ...    |
```

Figure 3: Example of NON Request with Q-Block1 Option (Without Loss)

Consider now a scenario where a new body of data is to be sent by the client, but some blocks are dropped in transmission as illustrated in Figure 4.

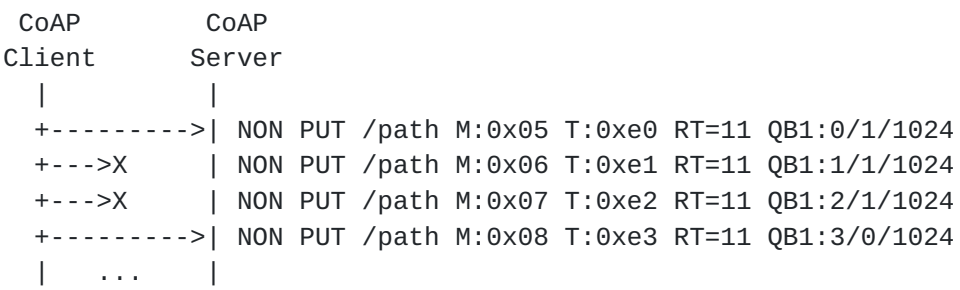

Figure 4: Example of NON Request with Q-Block1 Option (With Loss)

The server realizes that some blocks are missing and asks for the missing ones in one go (Figure 5). It does so by indicating which blocks have been received in the data portion of the response. The Token just needs to be one of those that have been received for this Request-Tag, so the client can derive the Request-Tag.

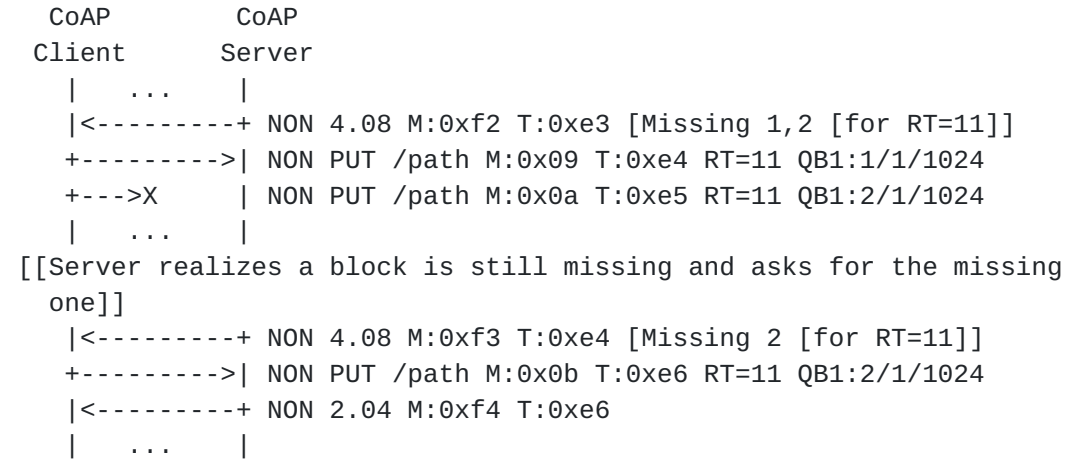

Figure 5: Example of NON Request with Q-Block1 Option (Blocks Recovery)

Under high levels of traffic loss, the client can elect not to retry sending missing blocks of data by "forgetting" all the tracked tokens for this Request-Tag. This decision is implementation specific.

9.2. Q-Block2 Option: Non-Confirmable Example

Figure 6 illustrates the example of Q-Block2 Option. The client sends a NON GET carrying an Observe and a Q-Block2 Options. The Q-Block2 Option indicates a block size hint (1024 bytes). This request is replied to by the server using four (4) blocks that are

transmitted to the client without any loss. Each of these blocks carries a Q-Block2 Option. The same process is repeated when an Observe is triggered, but no loss is experienced by any of the notification blocks.

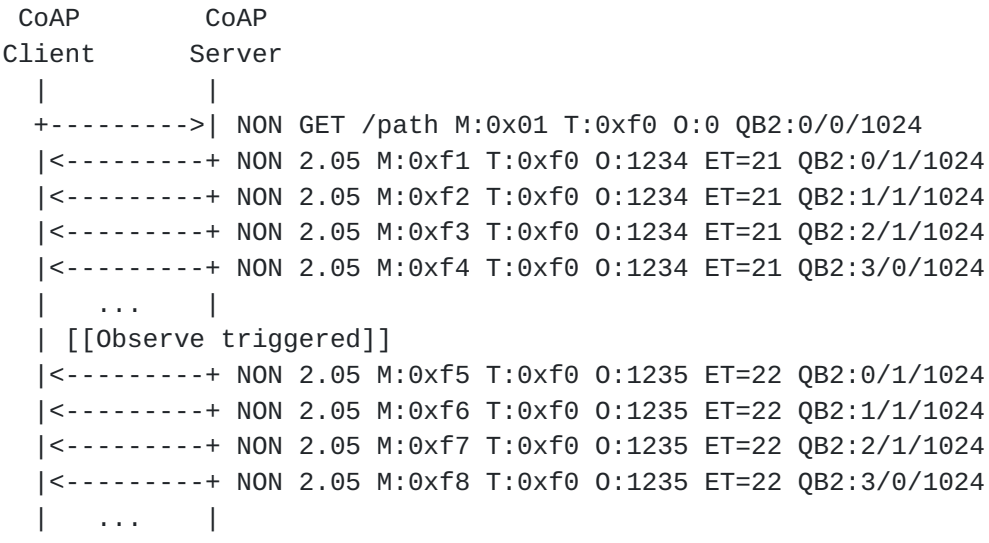

Figure 6: Example of NON Notifications with Q-Block2 Option (Without Loss)

Figure 7 shows the example of an Observe that is triggered but for which some notification blocks are lost. The client detects the missing blocks and requests their retransmission. It does so by indicating the blocks that were successfully received.

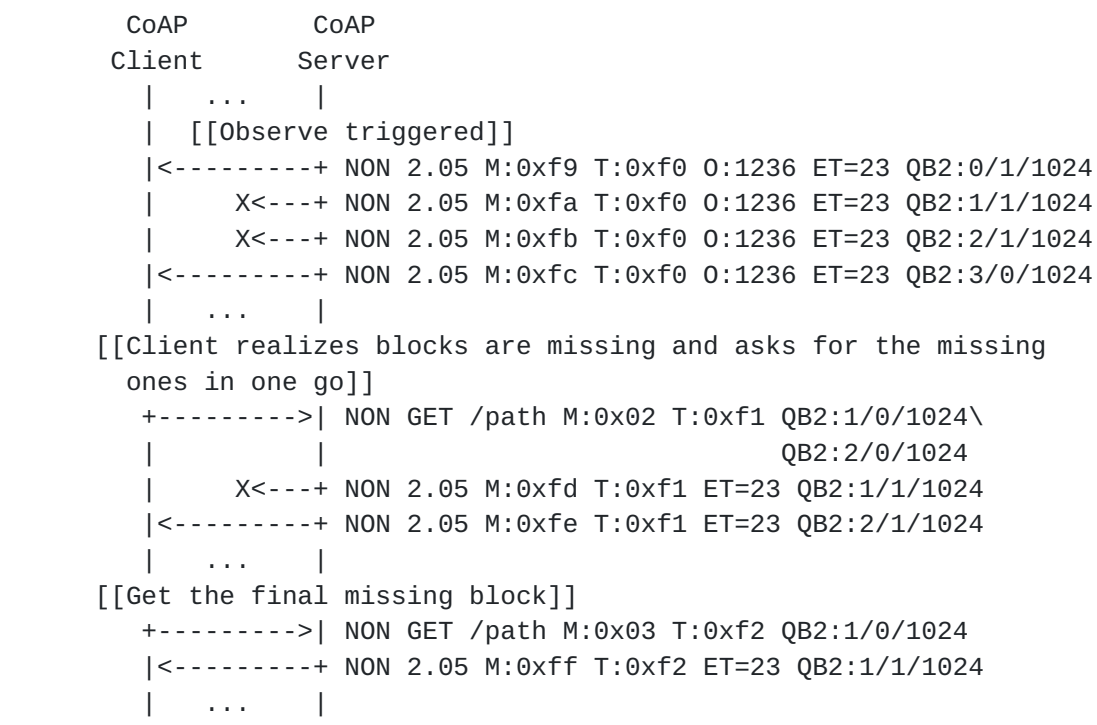

Figure 7: Example of NON Notifications with Q-Block2 Option (Blocks Recovery)

Under high levels of traffic loss, the client can elect not to retry getting missing blocks of data. This decision is implementation specific.

Figure 8 shows the example of an Observe that is triggered but only the first two notification blocks reaches the client. In order to retrieve the missing blocks, the client sends a request with a single Q-Block2 Option with the M bit set.

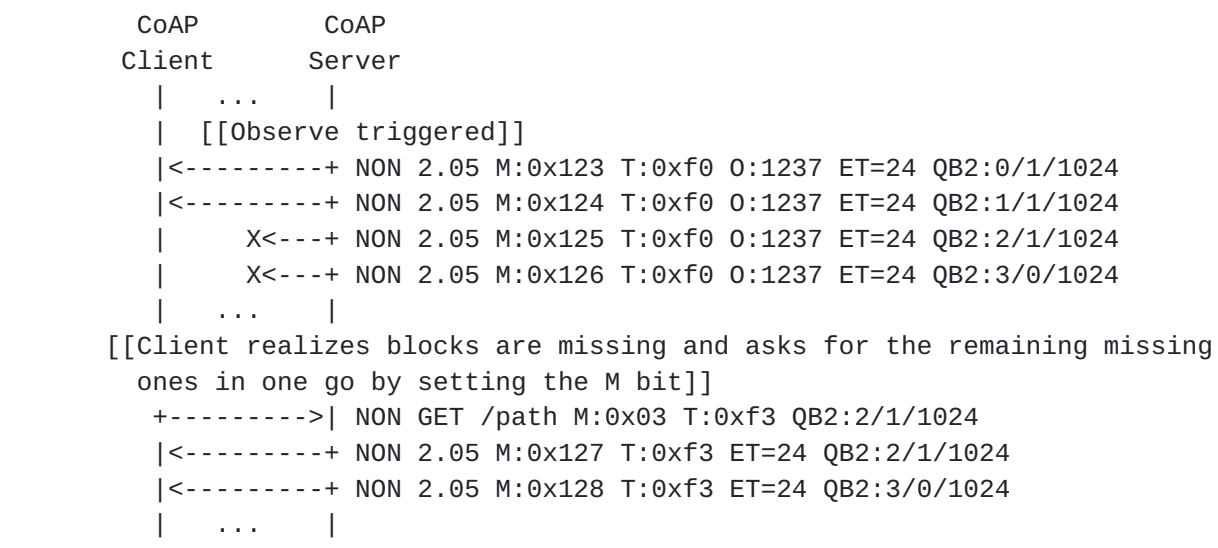

Figure 8: Example of NON Notifications with Q-Block2 Option (Blocks Recovery with M bit Set)

10. IANA Considerations

10.1. New CoAP Options

IANA is requested to add the following entries to the "CoAP Option Numbers" sub-registry [Options]:

Number	Name	Reference
TBA1	Q-Block1	[RFCXXXX]
TBA2	Q-Block2	[RFCXXXX]

Table 4: CoAP Q-Block1 and Q-Block2 Option Numbers

This document suggests 19 (TBA1) and 51 (TBA2) as a values to be assigned for the new option numbers.

10.2. New Media Type

This document requests IANA to register the "application/missing-blocks+cbor-seq" media type in the "Media Types" registry [IANA-MediaTypes]:

Type name: application

Subtype name: missing-blocks+cbor-seq

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary

Security considerations: See the Security Considerations Section of [This_Document].

Interoperability considerations: N/A

Published specification: [This_Document]

Applications that use this media type: Data serialization and deserialization.

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A
Magic number(s): N/A
File extension(s): N/A
Macintosh file type code(s): N/A

Person & email address to contact for further information: IETF, iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: none

Author: See Authors' Addresses section.

Change controller: IESG

Provisional registration? No

10.3. New Content Format

This document requests IANA to register the CoAP Content-Format ID for the "application/missing-blocks+cbor-seq" media type in the "CoAP Content-Formats" registry [Format]:

- Media Type: application/missing-blocks+cbor-seq
- Encoding: -
- Id: TBD3
- Reference: [RFCXXXX]

11. Security Considerations

Security considerations discussed in Section 7 of [RFC7959] should be taken into account.

Security considerations discussed in Sections 11.3 and 11.4 of [RFC7252] should be taken into account.

OSCORE provides end-to-end protection of all information that is not required for proxy operations and requires that a security context is set up (Section 3.1 of [RFC8613]). It can be trusted that the source endpoint is legitimate even if NoSec security mode is used. However, an intermediary node can modify the unprotected outer Q-Block1 and/or Q-Block2 Options to cause a Q-Block transfer to fail or keep requesting all the blocks by setting the M bit and, thus, causing attack amplification. As discussed in Section 12.1 of [RFC8613], applications need to consider that certain message fields and messages types are not protected end-to-end and may be spoofed or manipulated. It is NOT RECOMMENDED that the NoSec security mode is used if the Q-Block1 and Q-Block2 Options are to be used.

Security considerations related to the use of Request-Tag are discussed in Section 5 of [I-D.ietf-core-echo-request-tag].

12. Acknowledgements

Thanks to Achim Kraus, Jim Schaad, Michael Richardson, and Marco Tiloca for the comments.

Special thanks to Christian Amsuess and Carsten Bormann for their suggestions and several reviews, which improved this specification significantly.

Some text from [RFC7959] is reused for readers convenience.

13. References

13.1. Normative References

[I-D.ietf-core-echo-request-tag]
Amsuess, C., Mattsson, J., and G. Selander, "CoAP: Echo, Request-Tag, and Token Processing", draft-ietf-core-echo-request-tag-11 (work in progress), November 2020.

[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>.

[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>.

[RFC8075] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, DOI 10.17487/RFC8075, February 2017, <https://www.rfc-editor.org/info/rfc8075>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/info/rfc8132>.

[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>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/info/rfc8742>.

[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>.

13.2. Informative References

[Format] , <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>.

[I-D.ietf-dots-telemetry] Boucadair, M., Reddy.K, T., Doron, E., chenmeiling, c., and J. Shallow, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry", draft-ietf-dots-telemetry-15 (work in progress), December 2020.

[IANA-MediaTypes] IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[Options] , <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#option-numbers>.

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8782] Reddy.K, T., Ed., Boucadair, M., Ed., Patil, P., Mortensen, A., and N. Teague, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 8782, DOI 10.17487/RFC8782, May 2020, <https://www.rfc-editor.org/info/rfc8782>.

Appendix A. Examples with Confirmable Messages

These examples assume NSTART has been increased to at least 4.

The notations provided in Figure 2 are used in the following subsections.

A.1. Q-Block1 Option

Let's now consider the use Q-Block1 Option with a CON request as shown in Figure 9. All the blocks are acknowledged (ACK).

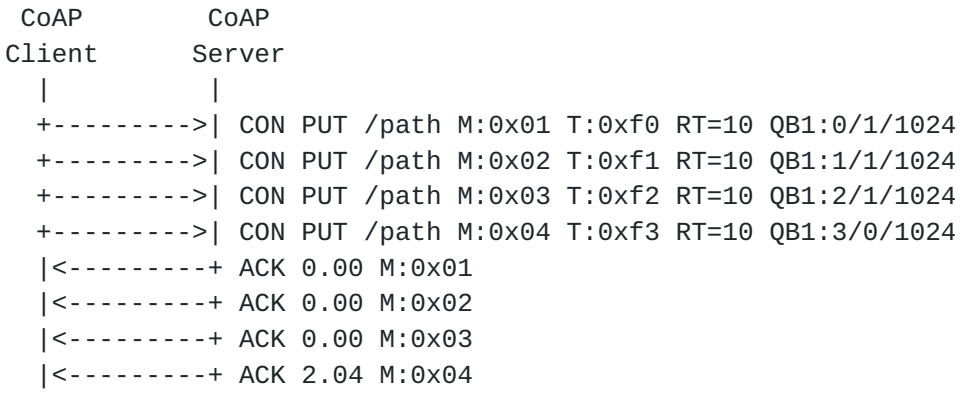

Figure 9: Example of CON Request with Q-Block1 Option (Without Loss)

Now, suppose that a new body of data is to sent but with some blocks dropped in transmission as illustrated in Figure 10. The client will retry sending blocks for which no ACK was received.

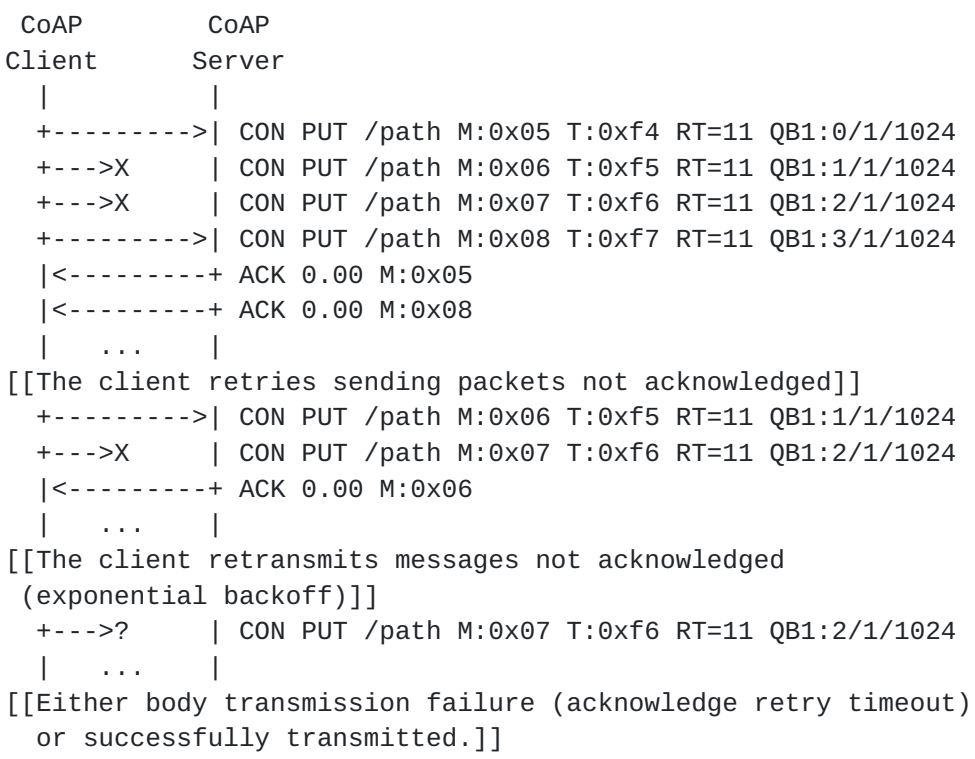

Figure 10: Example of CON Request with Q-Block1 Option (Blocks Recovery)

It is up to the implementation as to whether the application process stops trying to send this particular body of data on reaching MAX_RETRANSMIT for any payload, or separately tries to initiate the new transmission of the payloads that have not been acknowledged under these adverse traffic conditions.

If there is likely to be the possibility of network transient losses, then the use of NON should be considered.

A.2. Q-Block2 Option

An example of the use of Q-Block2 Option with Confirmable messages is shown in Figure 11.

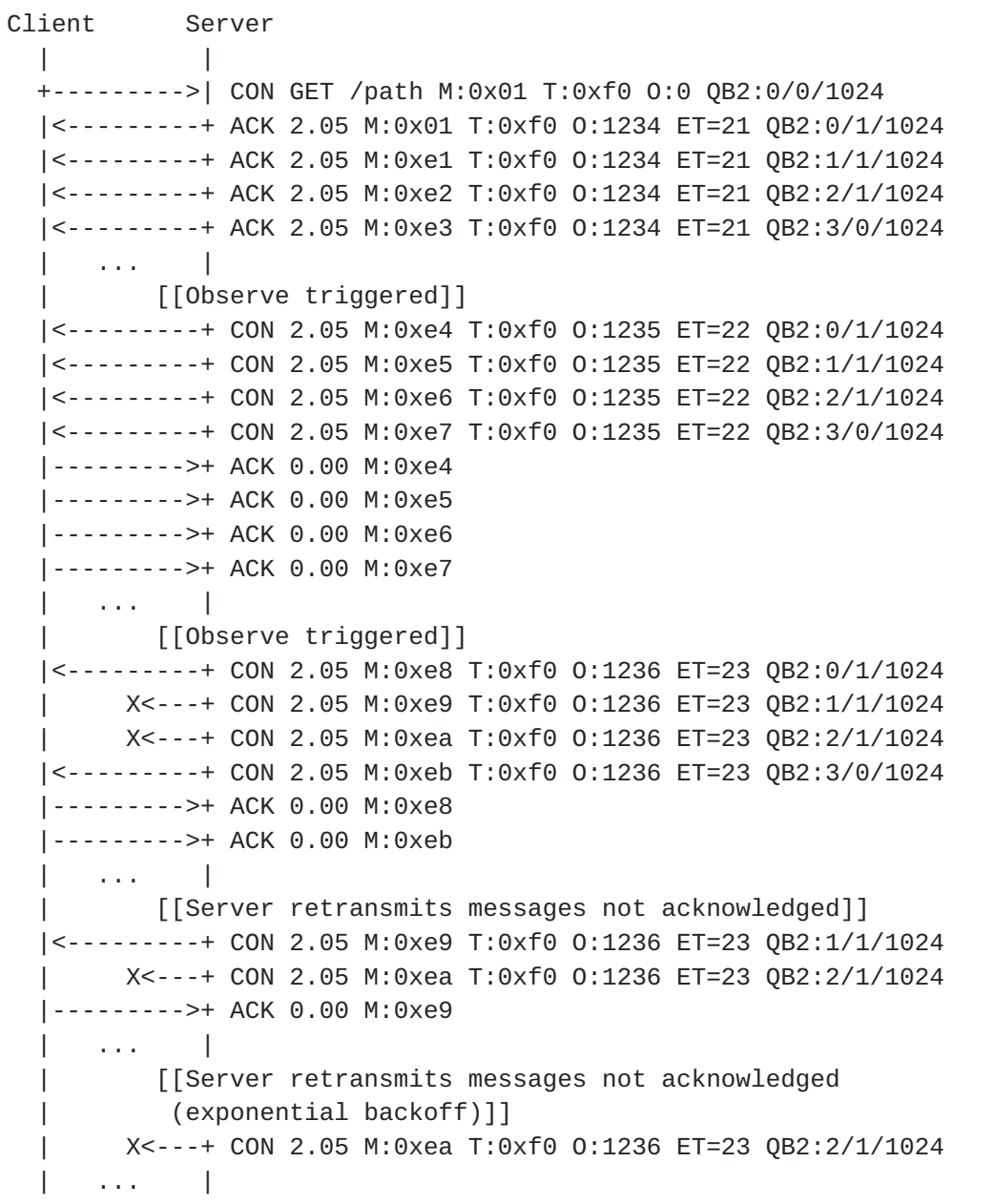

Figure 11: Example of CON Notifications with Q-Block2 Option

It is up to the implementation as to whether the application process stops trying to send this particular body of data on reaching MAX_RETRANSMIT for any payload, or separately tries to initiate the new transmission of the payloads that have not been acknowledged under these adverse traffic conditions.

If there is likely to be the possibility of network transient losses, then the use of NON should be considered.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Jon Shallow
United Kingdom

Email: supjps-ietf@jpshallow.com