CORE
Internet-Draft
Intended status: Standards Track
Expires: December 19, 2020

M. Boucadair
Orange
J. Shallow

June 17, 2020

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

draft-bosh-core-new-block-04

Abstract

This document specifies new Constrained Application Protocol (CoAP) Block-Wise transfer options: Block3 and Block4 Options.

These options are similar to the CoAP Block1 and Block2 Options, but 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 December 19, 2020.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Existing CoAP Block-Wise Transfer Options 2
1.2. New CoAP Block-Wise Transfer Options 3
1.3. New CoAP Response Code 4
1.4. Applicability Scope 4
2. Terminology 4
3. The Block3 and Block4 Options 5
3.1. Properties of Block3 and Block4 Options 5
3.2. Structure of Block3 and Block4 Options 6
3.3. Using the Block3 Option 7
3.4. Using the Block 4 Option 9
3.5. Working with Observe and Block4 Options 11
3.6. Working with Size1 and Size2 Options 11
3.7. Use of Block3 and Block4 Options Together 11
4. TBA3 (Missing Payloads) Response Code 11
5. Caching Considerations 12
6. HTTP-Mapping Considerations 13
7. Examples of Selective Block Recovery 13
7.1. Block3 Option: Non-Confirmable Example 13
7.2. Block4 Option: Non-Confirmable Example 15
8. IANA Considerations 16
8.1. New CoAP Options 16
8.2. New CoAP Response Code 17
8.3. New Content Format 17
9. Security Considerations 17
10. Acknowledgements 17
11. References 18
11.1. Normative References 18
11.2. Informative References 19
Appendix A. Examples with Confirmable Messages 19
A.1. Block3 Option 20
A.2. Block4 Option 21
Authors' Addresses 23

1. Introduction

1.1. Existing CoAP Block-Wise Transfer Options

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 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 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 transient packet loss. 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 use of Confirmable (CON) responses to handle potential packet loss; which does not work with a flooded pipe DDoS situation.

1.2. New CoAP Block-Wise Transfer Options

This document introduces the CoAP Block3 and Block4 Options. These options are similar in operation to the CoAP Block1 and Block2 Options respectively, but enable faster transmissions of sets of blocks of data with less packet interchanges as well as supporting faster recovery should any of the Blocks get lost in transmission.

Using Non-confirmable (NON) messages, the faster transmissions occur as all the Blocks can be transmitted serially (as are IP fragmented packets) without having to wait for an acknowledgement 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.

Note that the same performance benefits can be applied to Confirmable messages if the value of NSTART is increased from 1 (Section 4.7 of [RFC7252]). Some sample 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 endpoint sender either successful receipt or reports on all blocks in the body

that have been not yet been received. The CoAP endpoint sender will then retransmit only the blocks that have been lost in transmission.

Block3 and Block4 Options are used instead of Block1 and Block2 Options respectively because the transmission semantics and usage have changed.

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].

1.3. New CoAP Response Code

This document defines a new CoAP Response Code (Section 5.9 of [RFC7252]), called TBA3 (Missing payloads), to report on payloads using the Block3 Option that are not received by the server.

See Section 4 for more details.

1.4. Applicability Scope

The mechanism specified in the document 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].

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]).

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 Block3 and Block4 Options

3.1. Properties of Block3 and Block4 Options

The properties of Block3 and Block4 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 C column is marked for the Block3 Option. Only C and R columns are marked for the Block4 Option.

Number	C	U	N	R	Name	Format	Length	Default
TBA1	x				Block3	uint	0-7	(none)
TBA2	x			x	Block4	uint	0-7	(none)

Table 1: CoAP Block3 and Block4 Option Properties

The Block3 and Block4 Options can be present in both the request and response messages. The Block3 Option pertains to the request payload and the Block4 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).

Block3 is useful with the payload-bearing POST, PUT, PATCH, and iPATCH requests and their responses (2.01 and 2.04). Block4 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]).

To indicate support for Block4 responses, the CoAP client MUST include the Block4 Option in a GET or similar requests so that the server knows that the client supports this Block4 functionality should it needs 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 Block3 Option is present in a request or Block4 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 Block3 (or Block4) Option is intended to be optional. However, when it is present in a CoAP message, it MUST be processed (or the message rejected). Therefore, Block3 and Block4 Options are identified as Critical options.

The Block3 and Block4 Options are safe to forward. That is, a CoAP proxy that does not understand the Block3 (or Block4) Option should forward the option on.

The Block4 Option is repeatable when requesting re-transmission of missing Blocks, but not otherwise. Except that case, any request carrying multiple Block3 (or Block4) Options MUST be handled following the procedure specified in Section 5.4.5 of [RFC7252].

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 body of blocks will be subjected to PROBING_RATE (Section 4.7 of [RFC7252]).

The Block3 and Block4 Options, like the Block1 and Block2 Options, are both a class E and a class U in terms of OSCORE processing (see Section 4.1 of [RFC8613]): The Block3 (or Block4) Option MAY be an Inner or Outer option. 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.

3.2. Structure of Block3 and Block4 Options

The structure of Block3 and Block4 Options follows the structure defined in Section 2.2 of [RFC7959].

There is no default value for the Block3 and Block4 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 Block3 Option

The Block3 Option is used when the client wants to send a large amount of data to the server using the POST, PUT, PATCH, or iPATCH methods where the data and headers do not fit into a single packet.

When Block3 Option is used, the client MUST include Request-Tag Option [I-D.ietf-core-echo-request-tag]. The Request-Tag value MUST be the same for all of the blocks in the body of data that is being transferred. It is also used to identify a particular block that needs to be re-transmitted. The Request-Tag is opaque in nature, but it is RECOMMENDED that the client treats it 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 server still treats it as an opaque entity. The Request-Tag value MUST be different for distinct bodies or sets of blocks of data and SHOULD be incremented whenever a new body of data is being transmitted for a CoAP session between peers. The initial Request-Tag value SHOULD be randomly generated by the client.

The client sends all the individual payloads of the body using Block3 and Request-Tag Options, only expecting a response when all the payloads have been sent. It is RECOMMENDED that after transmission of every set of MAX_PAYLOADS payloads of a single body, a delay is introduced of ACK_TIMEOUT (Section 4.8.2 of [RFC7252]) before the next set of payload transmissions to manage potential congestion issues. MAX_PAYLOADS should be configurable with a default value of 10.

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

For NON transmissions, it is permissible, but not required, to send the ultimate payload of a MAX_PAYLOADS set as a Confirmable packet. If a Confirmable packet is used, then the client MUST wait for the ACK to be returned before sending the next set of payloads, which can be in time terms less than the ACK_TIMEOUT delay.

Also, for NON transmissions, it is permissible, but not required, to send a Confirmable packet for the final payload of a body (that is, M bit unset). If a Confirmable packet is used, then the client MUST wait for the 2.01 (Created) or 2.04 (Changed) Response Codes to be returned for successful transmission, or TBA3 (Missing Payloads) Response Code to then resend the missing blocks (if any).

With NON transmission, the server acknowledges receipt of all of the payloads that make up the body or can respond at any time during the receipt of the payloads to acknowledge that some of the payloads have arrived, but others are missing. It is RECOMMENDED that, unless there are receipt issues, the server only responds when the final payload (i.e., M bit unset) is received.

For Confirmable transmission, the server MUST continue to acknowledge each packet. NSTART will also need to be increased from the default (1) to get faster transmission rates.

Tokens MUST be included. Each individual payload of the body MUST have a different Token value.

A 2.01 (Created) or 2.04 (Changed) Response Code indicates successful receipt of the entire body. The 2.31 (Continue) Response Code MUST NOT be used.

A 4.00 (Bad Request) Response Code MUST be returned if the request does not include a Request-Tag Option but does include a Block3 option.

A 4.02 (Bad Option) Response Code MUST be returned if the server does not support the Block3 Option.

Use of 4.08 (Request Entity Incomplete) Response Code is discouraged when using Block3 Option because packets may arrive out of sequence; TBA3 (Missing Payloads) Response Code (Section 4) SHOULD be used instead. However, 4.08 (Request Entity Incomplete) Response Code is still valid to reject a Content-Format mismatch.

A 4.13 (Request Entity Too Large) Response Code can be returned under similar conditions to those discussed in Section 2.9.3 of [RFC7959].

A TBA3 (Missing Payloads) Response Code indicates that some of the payloads are missing and need to be resent. The client then re-transmits the missing payloads using the Request-Tag and Block3 to specify the block number, SZX, and M bit as appropriate. The Request-Tag value to use is determined from the payload of the TBA3 (Missing Payloads) Response Code. If the client dos not recognize the Request-Tag, the client can ignore this response. As discussed above, the sending of the list of missing blocks is subject to MAX_PAYLOADS.

If the server has not received the final payload (i.e., a block with M bit unset), but one or other payloads have been received, it SHOULD wait for up to MAX_TRANSMIT_SPAN (Section 4.8.2 of [RFC7252]) before sending the TBA3 (Missing Payloads) Response Code. However, this

timer MAY be reduced to two times ACK_TIMEOUT before sending a TBA3 (Missing Payloads) Response Code to cover the situation where MAX_PAYLOADS has been triggered by the client causing a break in transmission.

In all cases, multiple TBA3 (Missing Payloads) Response Codes are traffic limited by PROBING_RATE.

If the client transmits a new body of data with a new Request-Tag to the same server, the server MUST remove any partially received body held for a previous Request-Tag for that resource.

If the server receives a duplicate block with the same Request-Tag, it SHOULD silently ignore the packet.

A server SHOULD only maintain a partial body (missing payloads) for up to EXCHANGE_LIFETIME (Section 4.8.2 of [RFC7252]).

3.4. Using the Block 4 Option

Support for the receipt of Block4 Option by the client is indicated by using the Block4 Option in the GET, POST, PUT, FETCH, PATCH or iPATCH request. If the Block4 Option is not included in the request, the server MUST NOT send data using the Block4 Option, but can use Block2 if supported instead.

In a request, the Block4 MUST always have the M bit set to 0.

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 sending of the payloads is subject to MAX_PAYLOADS. If MAX_PAYLOADS is exceeded, the server MUST introduce an ACK_TIMEOUT delay before transmitting the next set of payloads.

The ETag is opaque in nature, but it is RECOMMENDED that the server treats it 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 client still treats it as an opaque entity. The ETag value MUST be different for distinct bodies or sets of blocks of data and SHOULD be incremented whenever a new body of data is being transmitted for a CoAP session between peers. The initial ETag value SHOULD be randomly generated by the server.

For NON transmission, it is permissible, but not required, to send the ultimate payload of a MAX_PAYLOADS set as a Confirmable packet.

If a Confirmable packet is used, then the server MUST wait for the ACK to be received before sending the next set of payloads, which can be in time terms less than the ACK_TIMEOUT delay.

Also, for NON transmission, it is permissible, but not required, to send a Confirmable packet for the final payload of a body (i.e., M bit unset). If a Confirmable packet is used, the server MUST wait for the ACK to be returned for successful transmission.

If the client detects that some of the payloads are missing, the missing payloads are requested by issuing a new GET, POST, PUT, FETCH, PATCH, or iPATCH request that contains one or more Block4 Options that define the missing blocks. A new Token value MUST be used for this request. The rate of requests for missing blocks is subject to PROBING_RATE. The ETag Option MUST NOT be used in the request as the server could respond with a 2.03 (Valid Response) with no payload. If the server responds with a different ETag Option value (as the resource representation has changed), then the client SHOULD drop all the payloads for the current body that are no longer valid.

The client may elect to request the missing blocks or just ignore the partial body.

All the payload responses to a specific GET, POST, PUT, FETCH, PATCH, or iPATCH request MUST have the same Token value as in the request.

With NON transmission, the client only needs to indicate that some of the payloads are missing by issuing a GET, POST, PUT, FETCH, PATCH, or iPATCH request for the missing blocks.

For Confirmable transmission, the client SHOULD continue to acknowledge each packet as well as issuing a separate GET, POST, PUT, FETCH, PATCH, or iPATCH for the missing blocks. NSTART will also need to be increased from the default (1) to get faster transmission rates.

If the server transmits a new body of data (e.g., a triggered Observe) with a new ETag to the same client with the same Token value, the client MUST remove any partially received body held for a previous ETag for that Token.

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 EXCHANGE_LIFETIME (Section 4.8.2 of [RFC7252]) or as defined by the Max-Age Option whichever is the less.

3.5. Working with Observe and Block4 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.

Likewise, the Tokens MUST all have the same value for all the blocks of the same body. This is so that if any of the blocks get lost during transmission (including the first one), the receiving CoAP endpoint can take the appropriate decisions as to how to continue (implementation specific).

If the client requests missing blocks, the client MUST use a different Token; all repeated missing blocks for that new request MUST use the new Token.

3.6. Working with 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.

It is RECOMMENDED that the Size1 Option is used with the Block3 Option. It is also RECOMMENDED that the Size2 Option is used with the Block4 Option.

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

3.7. Use of Block3 and Block4 Options Together

The behavior is similar to the one defined in Section 3.3 of [RFC7959] with Block3 substituted for Block1 and Block4 for Block2.

4. TBA3 (Missing Payloads) Response Code

TBA3 (Missing Payloads) Response Code is a new client error status code (Section 5.9.2 of [RFC7252]) 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 TBA3 (Missing Payloads) Response Code is encoded as a CBOR Sequence [RFC8742]. First is CBOR encoded Request-Tag followed by 1 or more missing CBOR encoded missing block numbers.

The missing block numbers MUST be unique per TBA3 (Missing Payloads) when created by the server; the client SHOULD drop any duplicates in the same TBA3 (Missing Payloads) message.

The Content-Format Option (Section 5.10.3 of [RFC7252]) MUST be used in the TBA3 (Missing Payloads) Response Code. It MUST be set to "application/missing-blocks+cbor-seq" (see Section 8.3).

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

```
TBA3-payload = (request-tag, missing-block-list)
; A copy of the opaque Request-Tag value
request-tag = bstr
missing-block-list = [1 * missing-block-number]
; A unique block number not received
missing-block-number = uint
```

Figure 1: Structure of the Missing Blocks Payload

If the size of the TBA3 (Missing Payloads) 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 necessary, multiple TBA3 (Missing Payloads) Response Codes can be sent back; each covering a Request-Tag and a unique set of missing blocks. The same Token can be used for the multiple TBA3 (missing Payloads) if this is the case.

5. Caching Considerations

The Block3 and Block4 Options are part of the cache key. As such, a CoAP proxy that does not understand the Block3 and Block4 Options must follow the recommendations in Section 5.7.1 of [RFC7252] for caching.

This specification does not require a proxy to obtain the complete representation before it serves parts of it to the client. Otherwise, the considerations discussed in Section 2.10 of [RFC7959] apply for the Block3 and Block4 Options (with Block3 substituted for Block1 and Block4 substituted for Block2) for proxies that support Block3 and Block4 Options.

A proxy that supports Block4 Option MUST be prepared to receive a GET or similar message indicating one or more missing blocks. The proxy can serve from its cache missing blocks that are available in its cache in a set as a server would send all the Block4s. If one or more requested blocks are not available in the cache, the proxy SHOULD update the GET request by removing the blocks that it can

serve from the cache, and then forward on the request to the next hop.

Alternatively, the original request unmodified (from the missing block perspective) MAY be forwarded on to the server. All the responses are then passed back to the client with the cache getting updated.

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).

6. 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.

7. Examples of Selective Block Recovery

This section provides some sample flows to illustrate the use of Block3 and Block4 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
  B3: Block3 Option values NUM/More/SZX
  B4: Block3 Option values NUM/More/SZX
   \: Trimming long lines
[[]]: Comments
-->X: Message loss
X<--: Message loss
```

Figure 2: Notations Used in the Figures

7.1. Block3 Option: Non-Confirmable Example

Figure 3 depicts an example of a NON PUT request conveying Block3 Option. All the blocks are received by the server.

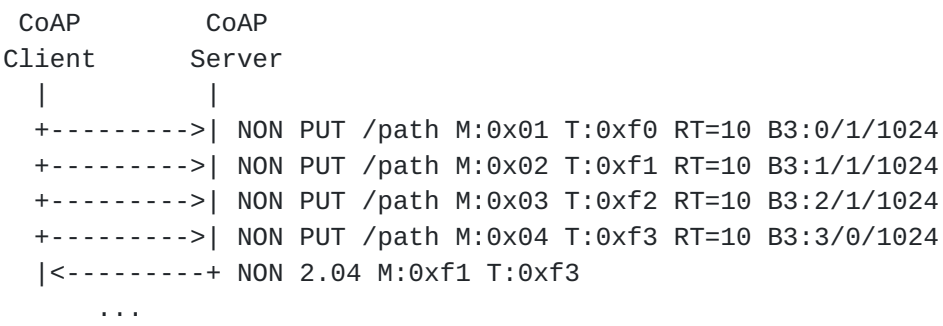

Figure 3: Example of NON Request with Block3 Option (Without Loss)

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.

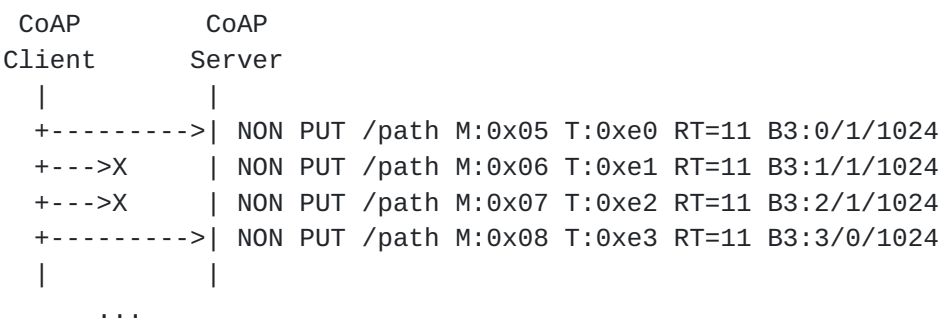

Figure 4: Example of NON Request with Block3 Option (With Loss)

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.

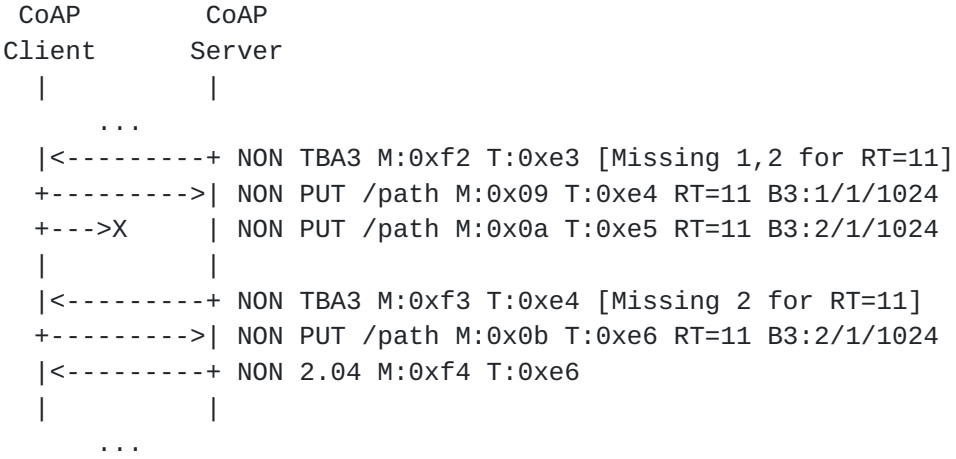

Figure 5: Example of NON Request with Block3 Option (Blocks Recovery)

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

7.2. Block4 Option: Non-Confirmable Example

Figure 6 illustrates the example of Block4 Option. The client sends a NON GET carrying an Observe and a Block4 Options. The Block4 Option indicates a size hint (1024 bytes). This request is replied by the server using four (4) blocks that are transmitted to the client without any loss. Each of these blocks carries a Block4 Option. The same process is repeated when an Observe is triggered, but no loss is experienced by any of the notification blocks.

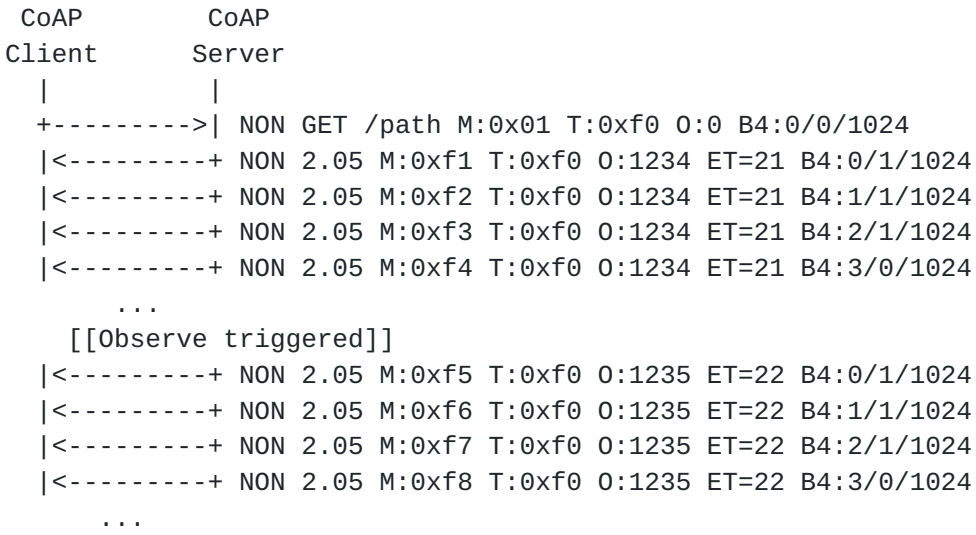

Figure 6: Example of NON Notifications with Block4 Option (Without Loss)

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 request their retransmission. It does so by indicating the blocks that were successfully received.

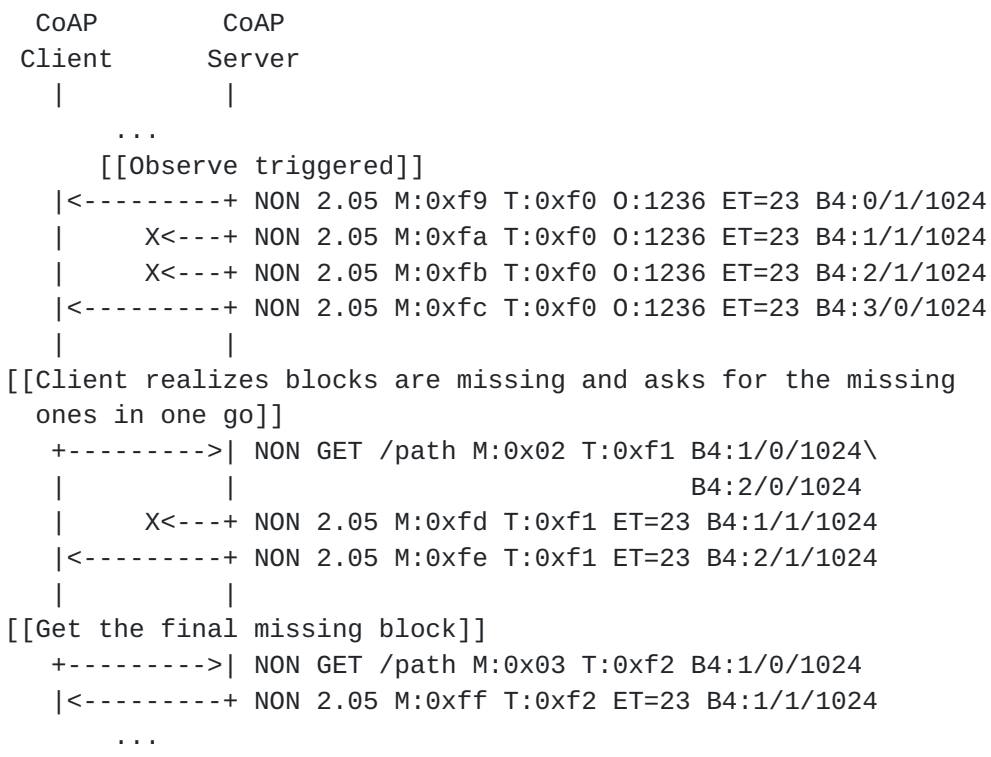

Figure 7: Example of NON Notifications with Block4 Option (Blocks Recovery)

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

8. IANA Considerations

8.1. New CoAP Options

IANA is requested to add the following entries to the "CoAP Option Numbers" sub-registry available at https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#option-numbers:

Number	Name	Reference
TBA1	Block3	[RFCXXXX]
TBA2	Block4	[RFCXXXX]

Table 2: CoAP Block3 and Block4 Option Numbers

This document suggests 21 (TBA1) and 25 (TBA2) as a values to be assigned for the new option numbers.

8.2. New CoAP Response Code

IANA is requested to add the following entry to the "CoAP Response Codes" sub-registry available at https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#response-codes:

Code	Description	Reference
TBA3	Missing Payloads	[RFCXXXX]

Table 3: New CoAP Response Code

This document suggests 4.19 (TBA3) as a value to be assigned for the new Response Code.

8.3. 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 available at https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats:

- Media Type: application/missing-blocks+cbor-seq
- Encoding: -
- Id: TBD4
- Reference: [RFCXXXX]

9. Security Considerations

Security considerations discussed in Section 9 of [RFC7959] should be taken into account.

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

10. Acknowledgements

Thanks to Achim Kraus and Jim Schaad for the comments on the mailing list.

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.

11. References

11.1. 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-09 (work in progress), March 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>.

11.2. Informative References

[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-08 (work in progress), May 2020.

[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. Block3 Option

Let's now consider the use Block3 Option with a CON request as shown in Figure 8. All the blocks are acknowledged (ACK).

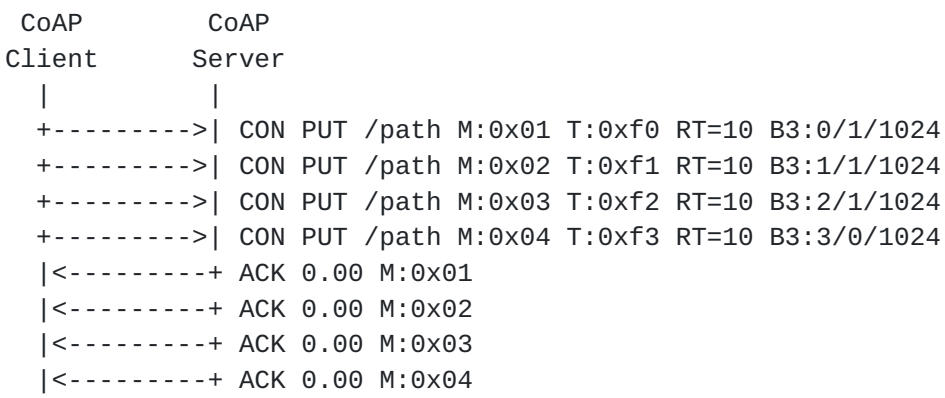

Figure 8: Example of CON Request with Block3 Option (Without Loss)

Now, suppose that a new body of data is to sent but with some blocks dropped in transmission as illustrated in Figure 9. The client will retry sending blocks for which no ACK was received.

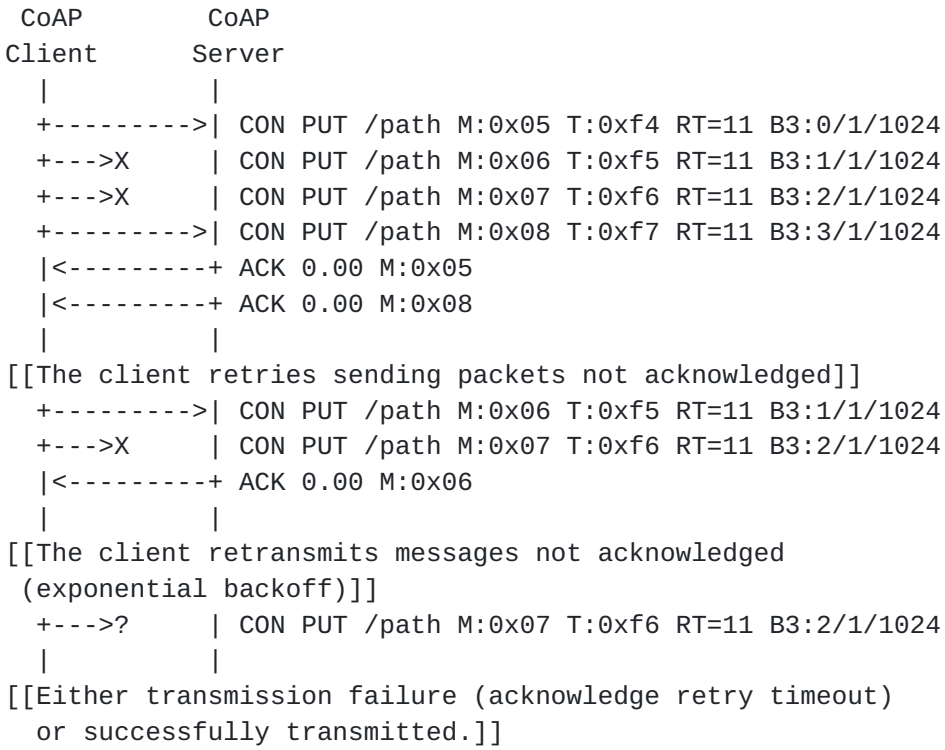

Figure 9: Example of CON Request with Block3 Option (Blocks Recovery)

It is implementation dependent as to whether a CoAP session is terminated following acknowledge retry timeout, or whether the CoAP session continues to be used under such adverse traffic conditions.

If there is likely to be the possibility of network transient losses, then the use of Non-confirmable traffic should be considered.

A.2. Block4 Option

An example of the use of Block4 Option with Confirmable messages is shown in Figure 10.

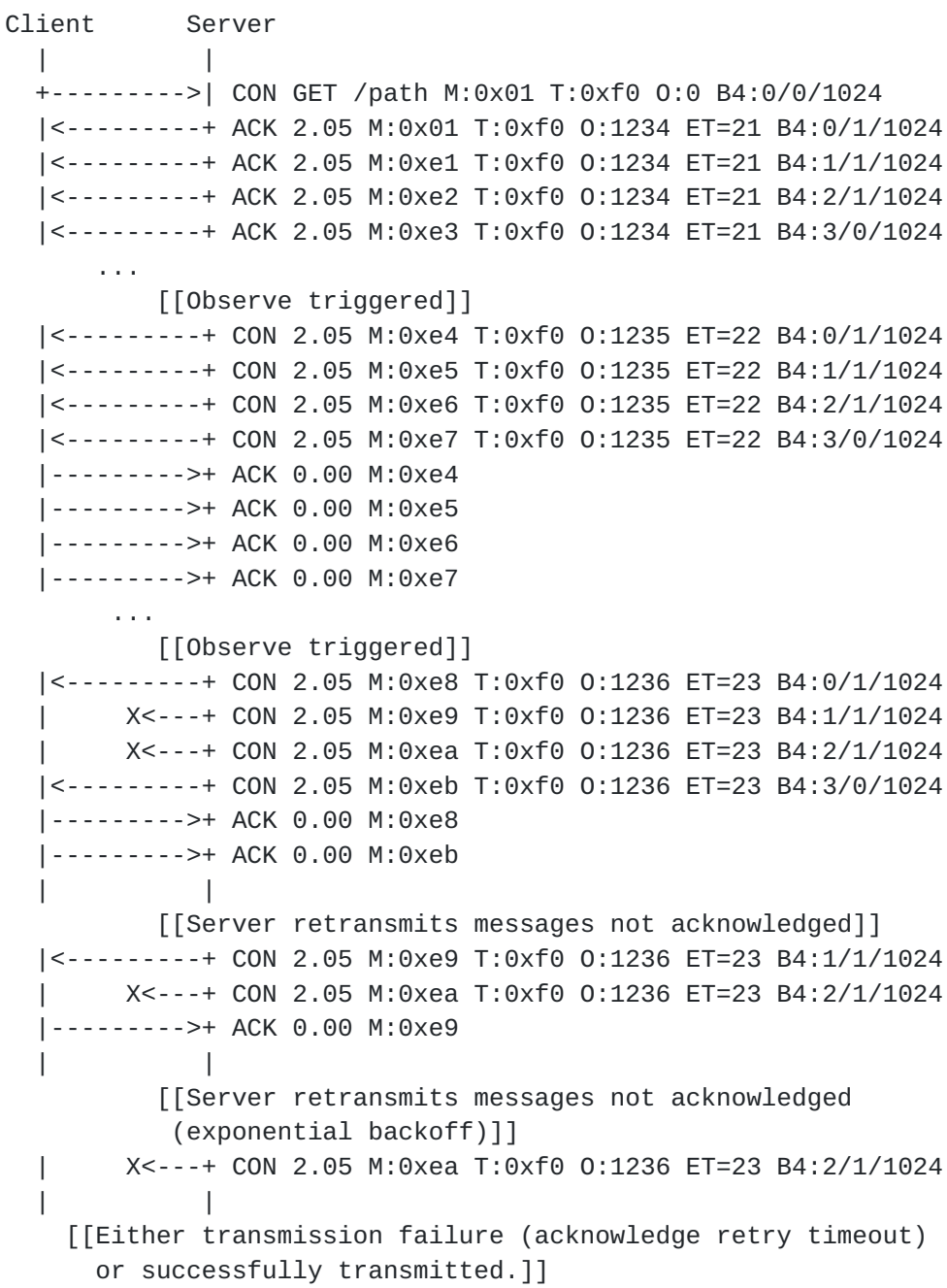

Figure 10: Example of CON Notifications with Block4 Option

It is implementation-dependent as to whether a CoAP session is terminated following acknowledge retry timeout, or whether the CoAP session continues to be used under such adverse traffic conditions.

If there is likely to be the possibility of network transient losses, then the use of Non-confirmable traffic 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