Constrained Application Protocol (CoAP) Block-Wise Transfer Options for
                        Faster Transmission
                    draft-ietf-core-new-block-05

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

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:

o  They can operate in environments where packet loss is highly
   asymmetrical.

o  They enable faster transmissions of sets of blocks of data with
   less packet interchanges.

o  They support faster recovery should any of the blocks get lost in
   transmission.

o  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:

o  Loss of lock-stepping so payloads are not always received in the
   correct (block ascending) order.

o  Additional congestion control measures need to be put in place for
   NON (Section 6.2).

o  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.

Q-Block2 Option is useful with GET, POST, PUT, FETCH, PATCH, and
iPATCH requests and their payload-bearing responses (2.01, 2.02,
2.03, 2.04, and 2.05) (Section 5.5 of [RFC7252]).

However, with methods needing both Q-Block1 for the request and
Q-Block2 for the response, and there is need for recovery following
payload loss, the numbers of packets needed to initiate and complete
the recovery can get very large as a function of the severity of the
experienced loss.

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

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.

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].

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.

With CoAP over UDP, the way a request message is rejected for
critical options depends on the message type.  A Confirmable message
with an unrecognized critical option is rejected with a 4.02 (Bad
Option) response (Section 5.4.1 of [RFC7252]).  A Non-confirmable
message with an unrecognized critical option is either rejected with
a Reset message or just silently ignored (Sections 5.4.1 and 4.3 of
[RFC7252]).  To reliably get a rejection message, it is therefore
REQUIRED that clients use a Confirmable message for determining
support for Q-Block1 and Q-Block2 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 (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 3.7 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
(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.  The token used SHOULD be from the last received payload.  The client should then release all of the tokens used for this body.

2.02 (Deleted)

   This Response Code indicates successful receipt of the entire body and the resource was deleted when using POST (Section 5.8.2 [RFC7252]).  The token used SHOULD be from the last received payload.  The client should then release all of the tokens used for this body.

2.04 (Changed)

   This Response Code indicates successful receipt of the entire body and the resource was updated.  The token used SHOULD be from the last received payload.  The client should then release all of the tokens used for this body.

2.05 (Content)

   This Response Code indicates successful receipt of the entire FETCH request body (Section 2 of [RFC8132]) and the appropriate representation of the resource has been returned.  The token used in the response MUST be the one in the FETCH request that has the Q-Block1 with the M bit unset.  If the FETCH request includes the Observe Option, then the server MUST use the same token for returning any Observe triggered responses.  The client should then release all of the tokens used for this body unless a resource is being observed.

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.  The token used SHOULD be from the last received payload.

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 (Section 6.2) payloads have been received by the
server.

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.

The token used in the response SHOULD be from the last received
payload.  See Section 4 for further information.

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 has different meanings based on the NUM value:

NUM is zero:  This is a request for the entire body.

'NUM modulo MAX_PAYLOADS' is zero, while NUM is not           zero:

   This is used to confirm that the current set of MAX_PAYLOADS payloads (the latest one having block number NUM-1) has been successfully received and that, upon receipt of this request, the server can continue to send the next set of payloads (the first one having block number NUM).  This is the 'Continue' Q-Block-2.

Any other value of NUM:  This is a request for that block and for all of the remaining blocks in the current MAX_PAYLOADS set.

If the request includes multiple Q-Block2 Options and these options overlap (e.g., combination of M being set (this and 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 3.7 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.  It is permissible to set the M bit to request this and
missing blocks from this MAX_PAYLOADS set.  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-Block1 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 server reports any missing payload, the client re-sends the
   missing payload(s).  Each re-sent payload is treated as a new request
   and the Observe Option MUST be included in the request.

   If the response (or associated response) uses Q-Block2 Option, and
   one of the response payloads is missing, the request for the missing
   payload(s) is treated as a new request.  The request MUST comprise of
   all of the request payloads and the Observe Option MUST NOT be
   included in either the request or response.

3.6.  **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 in either the
   request or response.  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.

3.7.  **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.

3.8.  **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].  It comprises of one or more 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" (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 last block number received so far with the same Request-Tag value.  Note that the use of any received token with the same Request-Tag would work, but providing the one used in the last received block number will aid any troubleshooting.  The client will use the token to determine what is the previously sent request to obtain the Request-Tag value to be used.

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 other 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_MAX_RETRANSMIT, 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.  By default, 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_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.  After this occurs, the peer SHOULD consider the
   body stale and remove all references to it.  By default,
   NON_MAX_RETRANSMIT has the same value as MAX_RETRANSMIT (Section 4.8
   of [RFC7252]).

   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_MAX_RETRANSMIT  |             4 |
| NON_TIMEOUT         |           2 s |
| NON_RECEIVE_TIMEOUT |           4 s |
| NON_PROBING_WAIT    |         247 s |
| NON_PARTIAL_TIMEOUT |         247 s |
+---------------------+---------------+
```

Table 3: Congestion Control 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].  Note that MAX_PAYLOADS,
   NON_MAX_RETRANSMIT, and NON_TIMEOUT can be negotiated between DOTS
   peers as per [I-D.bosh-dots-quick-blocks].

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.  The
newly derived value for MAX_PAYLOADS should be used for both ends of
this particular CoAP peer link.  Note that the CoAP peer will not

know about the MAX_PAYLOADS change until it is reconfigured.  As a
consequence of not being reconfigured, 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) Response Code on receipt of all of the MAX_PAYLOADS
payloads to prevent the client unnecessarily delaying.  Otherwise the
server SHOULD delay for NON_RECEIVE_TIMEOUT, before sending the 4.08
(Request Entity Incomplete) Response Code.  The NON_RECEIVE_TIMEOUT
delay may be reduced for the first time that this 4.08 (Request
Entity Incomplete) is sent if either the last of the MAX_PAYLOADS
payloads or the final payload with the M bit unset arrives, but
SHOULD NOT be reduced to zero as packets may arrive in the wrong
order.

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
received 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 on receipt of all of the
MAX_PAYLOADS payloads to prevent the server unnecessarily delaying.
Otherwise the client SHOULD delay for NON_RECEIVE_TIMEOUT, before
sending the request for the missing payloads.  The
NON_RECEIVE_TIMEOUT delay may be reduced for the first time that this
request is sent if either the last of the MAX_PAYLOADS payloads or
the final payload with the M bit unset arrives, but SHOULD NOT be
reduced to zero as packets may arrive in the wrong order.

The request that the client sends to acknowledge the receipt of all
the current set of MAX_PAYLOADS payloads is the 'Continue' Q-Block2
Option ('NUM modulo MAX_PAYLOADS' is zero, NUM is not zero, and M bit
is set (Section 3.4)).  The server SHOULD recognize this 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 the remaining missing blocks.

It is possible that the server may start transmitting the next set of
MAX_PAYLOADS payloads before the client times out on waiting for the
last of the previous MAX_PAYLOADS payloads.  Upon receipt of the
first payload from the new set of MAX_PAYLOADS payloads, the client
SHOULD send a request indicating any missing payloads from any
previous set of MAX_PAYLOADS payloads.  Upon receipt of such request,
the server 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.

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 with Non-confirmable Messages

This section provides some sample flows to illustrate the use of Q-Block1 and Q-Block2 Options with NON.  Examples with CON are provided in Appendix A.

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

### 9.1.1.  A Simple 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:0x81 T:0xc0 RT=9 QB1:0/1/1024
         +--------->| NON PUT /path M:0x82 T:0xc1 RT=9 QB1:1/1/1024
         +--------->| NON PUT /path M:0x83 T:0xc2 RT=9 QB1:2/1/1024
         +--------->| NON PUT /path M:0x84 T:0xc3 RT=9 QB1:3/0/1024
         |<---------+ NON 2.04 M:0xf1 T:0xc3
         |    ...    |
```

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

### 9.1.2.  Handling MAX_PAYLOADS Limits

Figure 4 depicts an example of a NON PUT request conveying Q-Block1
Option.  The number of payloads exceeds MAX_PAYLOADS.  All the blocks
are received by the server.

```
          CoAP         CoAP
        Client       Server
          |            |
          +--------->| NON PUT /path M:0x01 T:0xf1 RT=10 QB1:0/1/1024
          +--------->| NON PUT /path M:0x02 T:0xf2 RT=10 QB1:1/1/1024
          +--------->| [[Payloads 3 - 9 not detailed]]
          +--------->| NON PUT /path M:0x0a T:0xfa RT=10 QB1:9/1/1024
        [[MAX_PAYLOADS has been reached]]
          |      [[MAX_PAYLOADS blocks receipt acknowledged by server]]
          |<---------+ NON 2.31 M:0x81 T:0xfa
          +--------->| NON PUT /path M:0x0b T:0xfb RT=10 QB1:10/0/1024
          |<---------+ NON 2.04 M:0x82 T:0xfb
          |    ...    |
```

Figure 4: Example of MAX_PAYLOADS NON Request with Q-Block1 Option
(Without Loss)

### 9.1.3.  Handling MAX_PAYLOADS with Recovery

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 5.

```
          CoAP         CoAP
        Client       Server
          |            |
          +--------->| NON PUT /path M:0x11 T:0xe1 RT=11 QB1:0/1/1024
          +--->X     | NON PUT /path M:0x12 T:0xe2 RT=11 QB1:1/1/1024
          +--------->| [[Payloads 3 - 8 not detailed]]
          +--------->| NON PUT /path M:0x19 T:0xe9 RT=11 QB1:8/1/1024
          +--->X     | NON PUT /path M:0x1a T:0xea RT=11 QB1:9/1/1024
        [[MAX_PAYLOADS has been reached]]
          |    ...    |
        [[NON_TIMEOUT (client) delay expires]]
          |      [[Client starts sending next set of payloads]]
          +--->X     | NON PUT /path M:0x1b T:0xeb RT=11 QB1:10/1/1024
          +--------->| NON PUT /path M:0x1c T:0xec RT=11 QB1:11/1/1024
          |            |
```

Figure 5: Example of MAX_PAYLOADS NON Request with Q-Block1 Option
(With Loss)

On seeing a payload from the next set of payloads, the server
realizes that some blocks are missing from the previous MAX_PAYLOADS
payloads and asks for the missing blocks in one go (Figure 6).  It
does so by indicating which blocks have not been received in the data
portion of the response.  The token used in the response should be
the token that was used in the last block number received payload.

The client can then derive the Request-Tag by matching the token with
the sent request.

```
       CoAP         CoAP
      Client       Server
        |            |
        |<---------+ NON 4.08 M:0x91 T:0xec [Missing 1,9 [for RT=11]]
        |      [[Client responds with missing payloads]]
       +--------->| NON PUT /path M:0x1d T:0xed RT=11 QB1:1/1/1024
       +--------->| NON PUT /path M:0x1e T:0xee RT=11 QB1:9/1/1024
        |      [[Client continues sending next set of payloads]]
       +--------->| NON PUT /path M:0x1f T:0xef RT=11 QB1:12/0/1024
        |    ...     |
     [[NON_RECEIVE_TIMEOUT (server) delay expires]]
        |       [[The server realizes a block is still missing and asks
        |          for the missing one]]
        |<---------+ NON 4.08 M:0x92 T:0xef [Missing 10 [for RT=11]]
       +--------->| NON PUT /path M:0x20 T:0xf0 RT=11 QB1:10/1/1024
        |<---------+ NON 2.04 M:0x93 T:0xf0
        |    ...     |
```

            Figure 6: 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. Similarly, the server can elect to not to
   continue asking for missing blocks.  Both these decisions are
   implementation specific.

## 9.2.  Q-Block2 Option

### 9.2.1.  A Simple Example

   Figure 7 illustrates the example of Q-Block2 Option.  The client
   sends a NON GET carrying Observe and 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.

```
         CoAP         CoAP
        Client       Server
          |            |
         +--------->| NON GET /path M:0x01 T:0xc0 O:0 QB2:0/1/1024
          |<---------+ NON 2.05 M:0xf1 T:0xc0 O:1220 ET=19 QB2:0/1/1024
          |<---------+ NON 2.05 M:0xf2 T:0xc0 O:1220 ET=19 QB2:1/1/1024
          |<---------+ NON 2.05 M:0xf3 T:0xc0 O:1220 ET=19 QB2:2/1/1024
          |<---------+ NON 2.05 M:0xf4 T:0xc0 O:1220 ET=19 QB2:3/0/1024
          |    ...     |
          |      [[Observe triggered]]
          |<---------+ NON 2.05 M:0xf5 T:0xc0 O:1221 ET=20 QB2:0/1/1024
          |<---------+ NON 2.05 M:0xf6 T:0xc0 O:1221 ET=20 QB2:1/1/1024
          |<---------+ NON 2.05 M:0xf7 T:0xc0 O:1221 ET=20 QB2:2/1/1024
          |<---------+ NON 2.05 M:0xf8 T:0xc0 O:1221 ET=20 QB2:3/0/1024
          |    ...     |
```

Figure 7: Example of NON Notifications with Q-Block2 Option (Without
Loss)

### 9.2.2.  Handling MAX_PAYLOADS Limits

Figure 8 illustrates the same as Figure 7 but this time has eleven
(11) payloads which exceeds MAX_PAYLOADS.  There is no loss
experienced.

```
        CoAP        CoAP
       Client      Server
         |           |
        +--------->| NON GET /path M:0x01 T:0xf0 O:0 QB2:0/1/1024
        |<---------+ NON 2.05 M:0x81 T:0xf0 O:1234 ET=21 QB2:0/1/1024
        |<---------+ NON 2.05 M:0x82 T:0xf0 O:1234 ET=21 QB2:1/1/1024
        |<---------+ [[Payloads 3 - 9 not detailed]]
        |<---------+ NON 2.05 M:0x8a T:0xf0 O:1234 ET=21 QB2:9/1/1024
     [[MAX_PAYLOADS has been reached]]
        |      [[MAX_PAYLOADS blocks acknowledged by client using
        |         'Continue' Q-Block2]]
        +--------->| NON GET /path M:0x02 T:0xf1 QB2:10/1/1024
        |<---------+ NON 2.05 M:0x8b T:0xf0 O:1234 ET=21 QB2:10/0/1024
        |    ...    |
        |      [[Observe triggered]]
        |<---------+ NON 2.05 M:0x91 T:0xf0 O:1235 ET=22 QB2:0/1/1024
        |<---------+ NON 2.05 M:0x92 T:0xf0 O:1235 ET=22 QB2:1/1/1024
        |<---------+ [[Payloads 3 - 9 not detailed]]
        |<---------+ NON 2.05 M:0x9a T:0xf0 O:1235 ET=22 QB2:9/1/1024
     [[MAX_PAYLOADS has been reached]]
        |      [[MAX_PAYLOADS blocks acknowledged by client using
        |         'Continue' Q-Block2]]
        +--------->| NON GET /path M:0x03 T:0xf2 QB2:10/1/1024
        |<---------+ NON 2.05 M:0x9b T:0xf0 O:1235 ET=22 QB2:10/0/1024
     [[Body has been received]]
        |    ...    |
```

      Figure 8: Example of NON Notifications with Q-Block2 Option (Without
                                   Loss)

## 9.2.3.  Handling MAX_PAYLOADS with Recovery

   Figure 9 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.

```
          CoAP          CoAP
        Client        Server
           |    ...    |
           |       [[Observe triggered]]
           |<---------+ NON 2.05 M:0xa1 T:0xf0 O:1236 ET=23 QB2:0/1/1024
           |      X<---+ NON 2.05 M:0xa2 T:0xf0 O:1236 ET=23 QB2:1/1/1024
           |<---------+ [[Payloads 3 - 8 not detailed]]
           |      X<---+ NON 2.05 M:0xaa T:0xf0 O:1236 ET=23 QB2:9/1/1024
        [[MAX_PAYLOADS has been reached]]
           |    ...    |
        [[NON_TIMEOUT (server) delay expires]]
           |       [[Server sends next set of payloads]]
           |<---------+ NON 2.05 M:0xab T:0xf0 O:1236 ET=23 QB2:10/0/1024
           |    ...    |
        [[NON_RECEIVE_TIMEOUT (client) delay expires]]
           |       [[Client realizes blocks are missing and asks for the
           |          missing ones in one go]]
           +--------->| NON GET /path M:0x04 T:0xf3 QB2:1/0/1024\
           |          |                          QB2:9/0/1024
           |      X<---+ NON 2.05 M:0xac T:0xf3 ET=23 QB2:1/1/1024
           |<---------+ NON 2.05 M:0xad T:0xf3 ET=23 QB2:9/1/1024
           |    ...    |
        [[NON_RECEIVE_TIMEOUT (client) delay expires]]
           |       [[Client realizes block is still missing and asks for
           |          missing block]]
           +--------->| NON GET /path M:0x05 T:0xf4 QB2:1/0/1024
           |<---------+ NON 2.05 M:0xae T:0xf4 ET=23 QB2:1/1/1024
        [[Body has been received]]
           |    ...    |
```
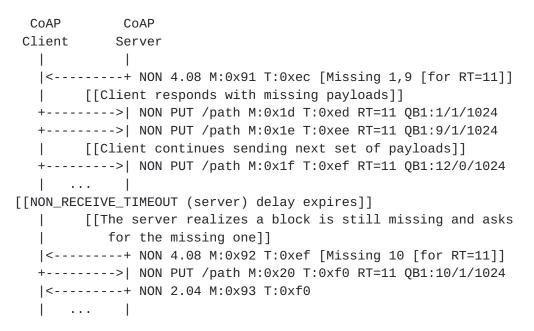
Figure 9: Example of NON Notifications with Q-Block2 Option (Blocks
Recovery)

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

## 9.2.4.  Handling Recovery using M-bit Set

Figure 10 shows the example of an Observe that is triggered but only
the first two notification blocks reach 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.

```
         CoAP        CoAP
        Client      Server
          |    ...     |
          |       [[Observe triggered]]
          |<---------+ NON 2.05 M:0xb1 T:0xf0 O:1237 ET=24 QB2:0/1/1024
          |<---------+ NON 2.05 M:0xb2 T:0xf0 O:1237 ET=24 QB2:1/1/1024
          |      X<---+ NON 2.05 M:0xb3 T:0xf0 O:1237 ET=24 QB2:2/1/1024
          |      X<---+ [[Payloads 4 - 9 not detailed]]
          |      X<---+ NON 2.05 M:0xb9 T:0xf0 O:1237 ET=24 QB2:9/1/1024
      [[MAX_PAYLOADS has been reached]]
          |    ...     |
      [[NON_TIMEOUT (server) delay expires]]
          |       [[Server sends next set of payloads]]
          |      X<---+ NON 2.05 M:0xba T:0xf0 O:1237 ET=24 QB2:10/0/1024
          |    ...     |
      [[NON_RECEIVE_TIMEOUT (client) delay expires]]
          |       [[Client realizes blocks are missing and asks for the
          |          missing ones in one go by setting the M bit]]
         +--------->| NON GET /path M:0x06 T:0xf5 QB2:2/1/1024
          |<---------+ NON 2.05 M:0xbb T:0xf5 ET=24 QB2:2/1/1024
          |<---------+ [[Payloads 3 - 9 not detailed]]
          |<---------+ NON 2.05 M:0xc2 T:0xf5 ET=24 QB2:9/1/1024
      [[MAX_PAYLOADS has been reached]]
          |       [[MAX_PAYLOADS acknowledged by client using 'Continue'
          |          Q-Block2]]
         +--------->| NON GET /path M:0x87 T:0xf6 QB2:10/1/1024
          |<---------+ NON 2.05 M:0xc3 T:0xf0 O:1237 ET=24 QB2:10/0/1024
      [[Body has been received]]
          |    ...     |
```

Figure 10: Example of NON Notifications with Q-Block2 Option (Blocks
                    Recovery with M bit Set)

## 9.3.  Q-Block1 and Q-Block2 Options

### 9.3.1.  A Simple Example

Figure 11 illustrates the example of a FETCH using both Q-Block1 and
Q-Block2 Options along with an Observe Option.  No loss is
experienced.

```
     CoAP        CoAP
    Client      Server
      |           |
      +--------->| NON FETCH /path M:0x10 T:0x90 O:0 RT=30 QB1:0/1/1024
      +--------->| NON FETCH /path M:0x11 T:0x91 O:0 RT=30 QB1:1/1/1024
      +--------->| NON FETCH /path M:0x12 T:0x93 O:0 RT=30 QB1:2/0/1024
      |<---------+ NON 2.05 M:0x60 T:0x93 O:1320 ET=90 QB2:0/1/1024
      |<---------+ NON 2.05 M:0x61 T:0x93 O:1320 ET=90 QB2:1/1/1024
      |<---------+ NON 2.05 M:0x62 T:0x93 O:1320 ET=90 QB2:2/1/1024
      |<---------+ NON 2.05 M:0x63 T:0x93 O:1320 ET=90 QB2:3/0/1024
      |    ...    |
      |      [[Observe triggered]]
      |<---------+ NON 2.05 M:0x64 T:0x93 O:1321 ET=91 QB2:0/1/1024
      |<---------+ NON 2.05 M:0x65 T:0x93 O:1321 ET=91 QB2:1/1/1024
      |<---------+ NON 2.05 M:0x66 T:0x93 O:1321 ET=91 QB2:2/1/1024
      |<---------+ NON 2.05 M:0x67 T:0x93 O:1321 ET=91 QB2:3/0/1024
      |    ...    |
```

Figure 11: Example of NON FETCH with Q-Block1 and Q-Block2 Options
(Without Loss)

### 9.3.2.  Handling MAX_PAYLOADS Limits

Figure 12 illustrates the same as Figure 11 but this time has eleven
(11) payloads in both directions which exceeds MAX_PAYLOADS.  There
is no loss experienced.

```
     CoAP        CoAP
    Client      Server
      |          |
      +--------->| NON FETCH /path M:0x30 T:0xa0 O:0 RT=10 QB1:0/1/1024
      +--------->| NON FETCH /path M:0x31 T:0xa1 O:0 RT=10 QB1:1/1/1024
      +--------->| [[Payloads 3 - 9 not detailed]]
      +--------->| NON FETCH /path M:0x39 T:0xa9 O:0 RT=10 QB1:9/1/1024
   [[MAX_PAYLOADS has been reached]]
      |     [[MAX_PAYLOADS blocks receipt acknowledged by server]]
      |<---------+ NON 2.31 M:0x80 T:0xa9
      +--------->| NON FETCH /path M:0x3a T:0xaa O:0 RT=10 QB1:10/0/1024
      |<---------+ NON 2.05 M:0x81 T:0xaa O:1334 ET=21 QB2:0/1/1024
      |<---------+ NON 2.05 M:0x82 T:0xaa O:1334 ET=21 QB2:1/1/1024
      |<---------+ [[Payloads 3 - 9 not detailed]]
      |<---------+ NON 2.05 M:0x8a T:0xaa O:1334 ET=21 QB2:9/1/1024
   [[MAX_PAYLOADS has been reached]]
      |     [[MAX_PAYLOADS blocks acknowledged by client using
      |        'Continue' Q-Block2]]
      +--------->| NON FETCH /path M:0x3b T:0xab QB2:10/1/1024
      |<---------+ NON 2.05 M:0x8b T:0xaa O:1334 ET=21 QB2:10/0/1024
      |   ...    |
      |     [[Observe triggered]]
      |<---------+ NON 2.05 M:0x8c T:0xaa O:1335 ET=22 QB2:0/1/1024
      |<---------+ NON 2.05 M:0x8d T:0xaa O:1335 ET=22 QB2:1/1/1024
      |<---------+ [[Payloads 3 - 9 not detailed]]
      |<---------+ NON 2.05 M:0x95 T:0xaa O:1335 ET=22 QB2:9/1/1024
   [[MAX_PAYLOADS has been reached]]
      |     [[MAX_PAYLOADS blocks acknowledged by client using
      |        'Continue' Q-Block2]]
      +--------->| NON FETCH /path M:0x3c T:0xac QB2:10/1/1024
      |<---------+ NON 2.05 M:0x96 T:0xaa O:1335 ET=22 QB2:10/0/1024
   [[Body has been received]]
      |   ...    |
```

     Figure 12: Example of NON FETCH with Q-Block1 and Q-Block2 Options
                            (Without Loss)

### 9.3.3.  Handling Recovery

   Consider now a scenario where there are some blocks are lost in
   transmission as illustrated in Figure 13.

```
     CoAP         CoAP
   Client        Server
      |            |
      +--------->| NON FETCH /path M:0x50 T:0xc0 O:0 RT=31 QB1:0/1/1024
      +--->X      | NON FETCH /path M:0x51 T:0xc1 O:0 RT=31 QB1:1/1/1024
      +--->X      | NON FETCH /path M:0x52 T:0xc2 O:0 RT=31 QB1:2/1/1024
      +--------->| NON FETCH /path M:0x53 T:0xc3 O:0 RT=31 QB1:3/0/1024
      |   ...     |
   [[NON_RECEIVE_TIMEOUT (server) delay expires]]
```

Figure 13: Example of NON FETCH with Q-Block1 and Q-Block2 Options
(With Loss)

The server realizes that some blocks are missing and asks for the
missing blocks in one go (Figure 14).  It does so by indicating which
blocks have not been received in the data portion of the response.
The token used in the response is be the token that was used in the
last block number received payload.  The client can then derive the
Request-Tag by matching the token with the sent request.

```
     CoAP         CoAP
   Client        Server
      |            |
      |<---------+ NON 4.08 M:0xa0 T:0xc3 [Missing 1,2 [for RT=31]]
      |      [[Client responds with missing payloads]]
      +--------->| NON FETCH /path M:0x54 T:0xc4 O:0 RT=31 QB1:1/1/1024
      +--------->| NON FETCH /path M:0x55 T:0xc5 O:0 RT=31 QB1:2/1/1024
      |      [[Server received FETCH body,
      |         starts transmitting response body]]
      |<---------+ NON 2.05 M:0xa1 T:0xc3 O:1236 ET=23 QB2:0/1/1024
      |      X<---+ NON 2.05 M:0xa2 T:0xc3 O:1236 ET=23 QB2:1/1/1024
      |<---------+ NON 2.05 M:0xa3 T:0xc3 O:1236 ET=23 QB2:2/1/1024
      |      X<---+ NON 2.05 M:0xa4 T:0xc3 O:1236 ET=23 QB2:3/0/1024
      |   ...     |
   [[NON_RECEIVE_TIMEOUT (client) delay expires]]
      |            |
```

Figure 14: Example of NON Request with Q-Block1 Option (Server
Recovery)

The client realizes that not all the payloads of the response have
been returned.  The client then asks for the missing blocks in one go
(Figure 15).  However, because the original request is spread over
multiple payloads using Q-Block1, the entire FETCH body has to be
used to make the request for the missing payloads.

```
       CoAP         CoAP
      Client       Server
        |            |
       +--------->| NON FETCH /path M:0x56 T:0xc6 RT=31 QB1:0/1/1024\
        |            |                                    QB2:1/0/1024\
        |            |                                    QB2:3/0/1024
       +--------->| NON FETCH /path M:0x57 T:0xc7 RT=31 QB1:1/1/1024\
        |            |                                    QB2:1/0/1024\
        |            |                                    QB2:3/0/1024
       +--------->| NON FETCH /path M:0x58 T:0xc8 RT=31 QB1:2/1/1024\
        |            |                                    QB2:1/0/1024\
        |            |                                    QB2:3/0/1024
       +--------->| NON FETCH /path M:0x59 T:0xc9 RT=31 QB1:3/0/1024\
        |            |                                    QB2:1/0/1024\
        |            |                                    QB2:3/0/1024
        |       [[Server received FETCH request,
        |        starts transmitting missing blocks]]
        |      X<---+ NON 2.05 M:0xa5 T:0xc9 ET=23 QB2:1/1/1024
        |<---------+ NON 2.05 M:0xa6 T:0xc9 ET=23 QB2:3/0/1024
        |    ...      |
     [[NON_RECEIVE_TIMEOUT (client) delay expires]]
        |       [[Client realizes block is still missing and asks for
        |         missing block]]
       +--------->| NON FETCH /path M:0x5a T:0xca RT=31 QB1:0/1/1024\
        |            |                                    QB2:1/0/1024
       +--------->| NON FETCH /path M:0x5b T:0xcb RT=31 QB1:1/1/1024\
        |            |                                    QB2:1/0/1024
       +--------->| NON FETCH /path M:0x5c T:0xcc RT=31 QB1:2/1/1024\
        |            |                                    QB2:1/0/1024
       +--------->| NON FETCH /path M:0x5d T:0xcd RT=31 QB1:3/0/1024\
        |            |                                    QB2:1/0/1024
        |       [[Server received FETCH request,
        |        starts transmitting missing block]]
        |<---------+ NON 2.05 M:0xa7 T:0xcd ET=23 QB2:1/1/1024
     [[Body has been received]]
        |    ...      |
```

Figure 15: Example of NON Request with Q-Block1 Option (Client
                               Recovery)

## 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]:

   o  Media Type: application/missing-blocks+cbor-seq
   o  Encoding: -
   o  Id: TBD3
   o  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, and Michael Richardson for their
   comments.

   Special thanks to Christian Amsuess, Carsten Bormann, and Marco
   Tiloca 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.bosh-dots-quick-blocks]
              Boucadair, M. and J. Shallow, "Distributed Denial-of-
              Service Open Threat Signaling (DOTS) Signal Channel
              Configuration Attributes for Faster Block Transmission",
              draft-bosh-dots-quick-blocks-00 (work in progress),
              January 2021.

   [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 16.  All the blocks are acknowledged (ACK).

```
        CoAP        CoAP
       Client      Server
         |           |
         +--------->| CON PUT /path M:0x01 T:0xf0 RT=10 QB1:0/1/1024
         +--------->| CON PUT /path M:0x02 T:0xf1 RT=10 QB1:1/1/1024
         +--------->| CON PUT /path M:0x03 T:0xf2 RT=10 QB1:2/1/1024
         +--------->| CON PUT /path M:0x04 T:0xf3 RT=10 QB1:3/0/1024
         |<---------+ ACK 0.00 M:0x01
         |<---------+ ACK 0.00 M:0x02
         |<---------+ ACK 0.00 M:0x03
         |<---------+ ACK 2.04 M:0x04
         |           |
```
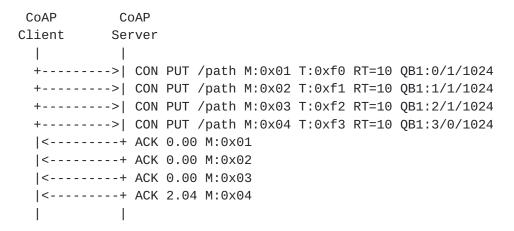
   Figure 16: Example of CON Request with Q-Block1 Option (Without Loss)

   Now, suppose that a new body of data is to be sent but with some
   blocks dropped in transmission as illustrated in Figure 17.  The
   client will retry sending blocks for which no ACK was received.

```
        CoAP        CoAP
       Client      Server
         |           |
        +--------->| CON PUT /path M:0x05 T:0xf4 RT=11 QB1:0/1/1024
        +--->X     | CON PUT /path M:0x06 T:0xf5 RT=11 QB1:1/1/1024
        +--->X     | CON PUT /path M:0x07 T:0xf6 RT=11 QB1:2/1/1024
        +--------->| CON PUT /path M:0x08 T:0xf7 RT=11 QB1:3/1/1024
        |<---------+ ACK 0.00 M:0x05
        |<---------+ ACK 0.00 M:0x08
        |   ...    |
      [[ACK_TIMEOUT (client) delay expires]]
        |     [[Client retransmits associated packet]]
        +--------->| CON PUT /path M:0x06 T:0xf5 RT=11 QB1:1/1/1024
        +--->X     | CON PUT /path M:0x07 T:0xf6 RT=11 QB1:2/1/1024
        |<---------+ ACK 0.00 M:0x06
        |   ...    |
      [[ACK_TIMEOUT exponential backoff (client) delay expires]]
        |     [[Client retransmits associated packet]]
        +--->?     | CON PUT /path M:0x07 T:0xf6 RT=11 QB1:2/1/1024
        |   ...    |
      [[Either body transmission failure (acknowledge retry timeout)
        or successfully transmitted.]]
```

Figure 17: 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 18.

```
        Client      Server
          |          |
          +--------->| CON GET /path M:0x01 T:0xf0 O:0 QB2:0/0/1024
          |<---------+ ACK 2.05 M:0x01 T:0xf0 O:1234 ET=21 QB2:0/1/1024
          |<---------+ ACK 2.05 M:0xe1 T:0xf0 O:1234 ET=21 QB2:1/1/1024
          |<---------+ ACK 2.05 M:0xe2 T:0xf0 O:1234 ET=21 QB2:2/1/1024
          |<---------+ ACK 2.05 M:0xe3 T:0xf0 O:1234 ET=21 QB2:3/0/1024
          |    ...    |
          |       [[Observe triggered]]
          |<---------+ CON 2.05 M:0xe4 T:0xf0 O:1235 ET=22 QB2:0/1/1024
          |<---------+ CON 2.05 M:0xe5 T:0xf0 O:1235 ET=22 QB2:1/1/1024
          |<---------+ CON 2.05 M:0xe6 T:0xf0 O:1235 ET=22 QB2:2/1/1024
          |<---------+ CON 2.05 M:0xe7 T:0xf0 O:1235 ET=22 QB2:3/0/1024
          |--------->+ ACK 0.00 M:0xe4
          |--------->+ ACK 0.00 M:0xe5
          |--------->+ ACK 0.00 M:0xe6
          |--------->+ ACK 0.00 M:0xe7
          |    ...    |
          |       [[Observe triggered]]
          |<---------+ CON 2.05 M:0xe8 T:0xf0 O:1236 ET=23 QB2:0/1/1024
          |      X<---+ CON 2.05 M:0xe9 T:0xf0 O:1236 ET=23 QB2:1/1/1024
          |      X<---+ CON 2.05 M:0xea T:0xf0 O:1236 ET=23 QB2:2/1/1024
          |<---------+ CON 2.05 M:0xeb T:0xf0 O:1236 ET=23 QB2:3/0/1024
          |--------->+ ACK 0.00 M:0xe8
          |--------->+ ACK 0.00 M:0xeb
          |    ...    |
       [[ACK_TIMEOUT (server) delay expires (twice in this example)]]
          |      [[Server retransmits associated packet]]
          |<---------+ CON 2.05 M:0xe9 T:0xf0 O:1236 ET=23 QB2:1/1/1024
          |      X<---+ CON 2.05 M:0xea T:0xf0 O:1236 ET=23 QB2:2/1/1024
          |--------->+ ACK 0.00 M:0xe9
          |    ...    |
       [[ACK_TIMEOUT exponential backoff (server) delay expires]]
          |      [[Server retransmits associated packet]]
          |      X<---+ CON 2.05 M:0xea T:0xf0 O:1236 ET=23 QB2:2/1/1024
          |    ...    |
       [[Either body transmission failure (acknowledge retry timeout)
         or successfully transmitted.]]
```

Figure 18: 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