CoRE   A. Bhattacharyya
Internet Draft   S. Bandyopadhyay
Intended status: Standards track   A. Pal
Expires: April 2014   Tata Consultancy Services Ltd.
                                             October 9, 2013

                    CoAP option for no server-response
                    draft-tcs-coap-no-response-option-04

   Abstract

   There can be typical M2M scenarios where responses from the data
   sink to the data source against request/ notification from the
   source might be considered redundant. This kind of open-loop
   exchange (with no reverse path from the sink to the source) may be
   desired while updating resources in constrained systems looking for
   maximized throughput with minimized resource consumption. CoAP
   already provides a non-confirmable (NON) mode of exchange where The
   receiving end-point does not respond with ACK. However, the
   receiving end-point responds the sender with a status code
   indicating "the result of the attempt to understand and satisfy the
   request".

   This draft introduces a header option: 'No-Response' to suppress
   responses from the receiver and discusses exemplary use cases which
   motivated this proposition based on real experience. This option
   also provides granularity by allowing suppression of a typical class
   or a combination of classes of responses. This option may be
   effective for both unicast and multicast scenarios.

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), its areas, and its working groups. Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   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."

Table of Contents

1. Introduction

This draft proposes a new header option 'No-Response' for
Constrained Application Protocol (CoAP). This option enables the
sender end-point to explicitly express its disinterest in getting
responses back from the receiving end-point. By default this option
expresses disinterest in any kind of response. This option should be
applicable along with non-confirmable (NON) updates. At present this
option will have no effect if used with confirmable (CON) mode.

Along with the technical details this draft presents some practical
application scenarios which should bring out the utility of this
option.

1.1. Granular suppression of responses

This option enables granularity by allowing the sender to choose the
typical class or combination of classes of responses which it is
disinterested in. For example, a sender may explicitly tell the
receiver that no response is required unless something 'bad' happens
and a response of class 4.xx or 5.xx is to be fed back to the
sender. No response is required in case of 2.xx classes. A similar
scheme is described in Section 3.7 of [I-D.ietf-core-groupcomm] on
the server side. Here the server may perform granular suppression
for group communication. But in this case the server itself decides
whether to suppress responses or not. This option enables the
clients to explicitly inform the server about the disinterest in
responses.

1.2. **Terminology**

The terms used in this draft are in conformance with those defined
in [I-D.ietf-core-coap].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC-2119.

2. **Potential benefits**

If this option is opportunistically used with fitting M2M
applications then the concerned systems may benefit in the following
aspects:

* Reduction in network clogging

* Reduction in server-side loading

* Reduction in battery consumption at the constrained end-point

* Reduction in communication cost at the constrained end-point

* May help to satisfy hard real-time requirements (since,
   waiting due to closed loop latency is completely avoided)

3. **Exemplary application scenarios**

The described scenarios are confined within a communication pattern
where there is a direct communication channel between a constrained
device (the device may well be a constrained gateway) and an
unconstrained backend. Also, we consider only the scenario of data
updates which happen through a push to the server by the client
using PUT or POST.

The application scenarios are classified into 2 categories as below:

Category 1) Data-source=constrained device; Data-sink=backend.

Category 2) Data-source=backend; Data-sink=constrained device.

Next sub-section describes the user stories and the potential
benefits in each of the cases through the use of No-Response option.
An Intelligent Traffic System (ITS) is considered as the base
application. The application scenarios are formed out of the
different aspects of ITS.

3.1. Frequent update of geo-location from vehicles to backend (Category 1)

Each vehicle in ITS is equipped with a sensor-gateway comprising sensors like GPS and Accelerometer. The sensor-gateway connects to the Internet using a low-bandwidth cellular (e.g. GPRS) connection. The GPS co-ordinates are periodically updated to the backend server by the gateway. In case of ITS the update rate is adaptive to the motional-state of the vehicle. If the vehicle moves fast the update rate is high as the position of the vehicle changes rapidly. If the vehicle is static or moves slowly then the update rate is low. This ensures that bandwidth and energy is not consumed unnecessarily. The motional-state of the vehicle is inferred by a local analytics running on the sensor-gateway which uses the accelerometer data and the rate of change in GPS co-ordinates. The back-end server hosts applications which use the updates for each vehicle and produce necessary information for remote users.

Retransmitting a location co-ordinate which is already passed by a vehicle is not efficient as it adds redundant traffic to the network. So, the updates are done in NON mode. However, given the thousands of vehicles updating frequently, the NON exchange will also trigger huge number of status responses from the backend. Each response in the air is of 4 bytes of application layer plus several bytes originating from the lower layers. Thus the cumulative load on the network will be quite significant.

On the contrary, if the edge devices explicitly declare that they do not need any status response then significant load will be reduced from the network and the server as well. The assumption is that since the update rate is high stray losses in geo-locations will be compensated with the large update rate and thereby not affecting the end applications.

3.1.1. Benefits using No-Response

Thus mapping the above scenario to the benefits mentioned in section 2 reveals that use of 'No-Response' will help in:

   * Reduction in network clogging

   * Reduction in server-side loading

   * Help in achieving real-time requirements as the application is
       not bound by any delay due to closed loop latency

3.2. Multicasting traffic congestion information to PDAs/ smart-phones
   (Category 2)

   The ITS might have an application which runs some analytics at the
   backend and determines the instantaneous traffic congestion spots in
   a city. The analytics is done based on the real-time geo-location
   updates received from the vehicles registered in the system. The
   backend application multicasts the instantaneous results of the
   analytics to the constrained handheld devices which registered to
   the city authority for real-time updates on congestion points. The
   backend is not really interested in the delivery status of these
   updates. In this case the backend uses No-Response option along with
   NON updates to reduce the traffic generated due to simultaneous
   status responses from hundreds of subscribed handheld devices.

3.2.1. Using granular response suppression

   However, an intelligent application may use the granularity feature
   of this option such that the responses are fed-back to the backend
   when updates to particular devices cause errors. So the updates may
   contain 'No-Response' option indicating that a response is to be
   suppressed only in success conditions and all responses in case of
   errors should be fed back. The server might eventually stop sending
   updates to the devices which responded with consecutive 'bad'
   responses. This will indirectly help saving network bandwidth.

3.2.2. Benefits using No-Response

   Thus mapping the above scenario to the benefits mentioned in section
   2 reveals that use of 'No-Response' will help in:

      * Reduction in network clogging

      * Reduction in battery consumption at the constrained end-point

      * Reduction in communication cost at the constrained end-point

4. Option Definition

   The properties of this option are as in Table 1.

```
+--------+---+---+---+---+-------------+--------+--------+---------+
| Number | C | U | N | R |    Name     | Format | Length | Default |
+--------+---+---+---+---+-------------+--------+--------+---------+
|  TBD   |   | X | - |   | No-Response |  uint  |   1    |    0    |
+--------+---+---+---+---+-------------+--------+--------+---------+
```
                         Table 1: Option Properties

This option is Elective and Non-Repeatable. If a proxy happens to
encounter this option it should not forward. Hence caching is not
applicable. The assumption here is that if an application needs a
proxy in between an unconstrained backend and a constrained node
then in most cases the leg between the constrained node and the
proxy will be constrained in nature. So by restricting this option
up to the proxy we can reap the benefits of this option in
constrained environment without increasing overall complexity.

This option is presently intended for update requests (e.g., PUT) in
NON mode and should have no effect if used with a CON request. This
option contains values to indicate interest/ disinterest in all or a
particular class or combination of classes of responses as described
in the next sub-section.

The following table provides a 'ready-reckoner' on possible
applicability of this option for all the four REST methods. This
table is prepared in view of the type of application scenarios
foreseen so far.

```
+-------------+----------------------------------------------------+
| Method Name |             Remarks on applicability               |
+-------------+----------------------------------------------------+
|     GET     |                  Not applicable.                   |
+-------------+----------------------------------------------------+
|             | Applicable for frequent updates in NON mode on     |
|     PUT     | existing fixed resources. Might not be useful when |
|             | PUT 'creates' a new resource.                      |
+-------------+----------------------------------------------------+
|             | If POST is used just to update a target resource   |
|             | then No-Response can be used in the same manner as |
|             | in NON-PUT. May also be applicable when POST       |
|             | performs resource creation and the client does not |
|             | refer to the resource in future. For example, than |
|             | updating a fixed resource, POST API may rather     |
|    POST     | contain a query-string with name/value pairs for a |
|             | defined action (ex. insertion into a database as   |
|             | part of frequent updates). The resources created   |
|             | this way may be 'short-lived' resources which the  |
|             | client will not refer to in future (see section    |
|             | 5.1.2.2).                                          |
+-------------+----------------------------------------------------+
|             | Not applicable. Deletion is usually a permanent    |
|    DELETE   | action and the client should make sure that the    |
|             | deletion actually happened.                        |
+-------------+----------------------------------------------------+
```
        Table 2: Applicability of No-Response for different methods

4.1. **Achieving granular suppression**

   This option is defined as a bit-map (Table 3) to achieve granular
   suppression.

```
+-------+----------------------+-------------------------------+
| Value | Binary Representation |          Description          |
+-------+----------------------+-------------------------------+
|   0   |       00000000       | Suppress all responses (same as |
|       |                      | empty value).                 |
+-------+----------------------+-------------------------------+
|   2   |       00000010       |   Allow 2.xx success responses. |
+-------+----------------------+-------------------------------+
|   8   |       00001000       |    Allow 4.xx client errors.  |
+-------+----------------------+-------------------------------+
|  16   |       00010000       |    Allow 5.xx server errors.  |
+-------+----------------------+-------------------------------+
```
                     Table 3: Option values

   XOR of the values defined for allowing particular classes will
   result in allowing a combination of classes of responses. So, a
   value of 18 (binary: 00010010) will result in allowing all 2.xx and
   5.xx classes of responses. It is to be noted that a value of 26 will
   indicate that all types of responses are to be allowed (which is as
   good as not using No-Response at all).

   Implementation Note: The use of No-Response option is very much
       driven by the application scenario and the characteristics of the
       information to be updated. Judicious use of this option benefits
       the overall system as explained in sections 2 and 3.

       When No-Response is used with empty or 0 value, the updating
       end-point should cease the listening activity for response
       against the particular request. On the contrary, opening up at
       least one class of responses means that the updating end-point
       can no longer stop listening and must be configured to listen up
       to some application specific time-out period for the particular
       request. The updating end-point never knows whether the present
       update will be a success or a failure. Thus, if the client
       decides to open up the responses for errors (4.xx & 5.xx) then it
       has to wait for the entire time-out period even for the instances
       where the request is successful (and the server is not supposed
       to send back a response). This kind of situation may arise for
       the scenario in section 3.2.1. Under such circumstances the use
       of No-Response may not help improving the performance in terms of

overall latency. However, the advantages in terms of saving
network and energy resources will still hold.

A point to be noted in view of the above example is that there
may be situations when the response on errors might get lost. In
such a situation the sender would wait up to the time-out period
but will not receive any response. But this should not lead to
the impression to the sender that the request was successful. The
situation will worsen if the receiver is no longer active. The
application designer needs to tackle such situation. Since this
option is conceived for frequent updates, the sender may
strategically insert requests without No-Response after N numbers
of requests with No-Response (similar to the way [I-D.ietf-core-
observe] 'weaves' CON notifications within series of NON
notifications to check if the observer is alive).

5. Example

This section illustrates few examples of exchanges based on the
scenario narrated in section 3.1. Examples for other scenarios can
be easily conceived based on these illustrations.

5.1. Request/response Scenario

5.1.1. Using No-Response with PUT

Figure 1 shows a typical request with this option. The depicted
scenario occurs when the vehicle#n moves very fast and update rate
is high. The vehicle is assigned a dedicated resource: vehicle-stat-
<n>, where <n> can be any string uniquely identifying the vehicle.
The update requests are in NON mode. The No-Response option causes
the server not to reply with any status code.

```
   Client Server
    |      |
    |      |
    +----->| Header: PUT (T=NON, Code=0.03, MID=0x7d38)
    | PUT  | Token: 0x53
    |      | Uri-Path: "vehicle-stat-00"
    |      | Content Type: text/plain
    |      | No-Response: 0
    |      | Payload:
    |      | "VehID=00&RouteID=DN47&Lat=22.5658745&Long=88.4107966667&
    |      | Time=2013-01-13T11:24:31"
    |      |
   [No response from the server. Next update in 20 secs.]
    |      |
    +----->| Header: PUT (T=NON, Code=0.03, MID=0x7d39)
    | PUT  | Token: 0x54
    |      | Uri-Path: "vehicle-stat-00"
    |      | Content Type: text/plain
    |      | No-Response: 0
    |      | Payload:
    |      | "VehID=00&RouteID=DN47&Lat=22.5649015&Long=88.4103511667&
    |      | Time=2013-01-13T11:24:51"
```

             Figure 1: Exemplary unreliable update with No-Response option using
                                   PUT.

5.1.2. Using No-Response with POST

   POST "usually results in a new resource being created or the target
   resource being updated". Exemplary uses of 'No-Response' for both
   these 'usual' actions of POST are given below.

5.1.2.1. POST updating a target resource

   In this case POST acts the same way as PUT. The exchanges are same
   as above. The updated values are carried as payload of POST as shown
   in Figure 2.

```
   Client Server
    |      |
    |      |
    +----->| Header: POST (T=NON, Code=0.02, MID=0x7d38)
    | POST | Token: 0x53
    |      | Uri-Path: "vehicle-stat-00"
    |      | Content Type: text/plain
    |      | No-Response: 0
    |      | Payload:
    |      | "VehID=00&RouteID=DN47&Lat=22.5658745&Long=88.4107966667&
    |      | Time=2013-01-13T11:24:31"
    |      |
   [No response from the server. Next update in 20 secs.]
    |      |
    +----->| Header: PUT (T=NON, Code=0.02, MID=0x7d39)
    | POST | Token: 0x54
    |      | Uri-Path: "vehicle-stat-00"
    |      | Content Type: text/plain
    |      | No-Response: 0
    |      | Payload:
    |      | "VehID=00&RouteID=DN47&Lat=22.5649015&Long=88.4103511667&
    |      | Time=2013-01-13T11:24:51"
```

          Figure 2: Exemplary unreliable update with No-Response option using
                          POST as the update-method.

5.1.2.2. POST performing updates through resource creation

   In most practical implementations the backend of section 3.1 will
   have a dedicated database to store the location updates. In such a
   case the client would send an update string as the POST URI which
   contains the name/value pairs for each update. Thus frequent updates
   may be performed through POST by creating such 'short-lived'
   resources which the client would not refer to in future. Hence 'No-
   Response' can be used in same manner as for updating fixed
   resources. The scenario is depicted in Figure 3.

```
   Client Server
   |      |
   |      |
   +----->| Header: POST (T=NON, Code=0.02, MID=0x7d38)
   | POST | Token: 0x53
   |      | Uri-Path: "insertInfo"
   |      | Uri-Query: "VehID=00"
   |      | Uri-Query: "RouteID=DN47"
   |      | Uri-Query: "Lat=22.5658745"
   |      | Uri-Query: "Long=88.4107966667"
   |      | Uri-Query: "Time=2013-01-13T11:24:31"
   |      | No-Response: 0
   |      |
   [No response from the server. Next update in 20 secs.]
   |      |
   +----->| Header: POST (T=NON, Code=0.02, MID=0x7d39)
   | POST | Token: 0x54
   |      | Uri-Path: "insertInfo"
   |      | Uri-Query: "VehID=00"
   |      | Uri-Query: "RouteID=DN47"
   |      | Uri-Query: "Lat=22.5649015"
   |      | Uri-Query: "Long=88.4103511667"
   |      | Uri-Query: "Time=2013-01-13T11:24:51"
   |      | No-Response: 0
   |      |
```

Figure 3: Exemplary unreliable update with No-Response option using
POST with a query-string to insert update information to backend
database.

5.2. Resource-observe Scenario

This option should be treated transparently if used with NON
notifications. In other words, just like GET and DELETE, this option
will have no effect for observe notifications. The following example
demonstrates how optimizations achieved using No-Response may also
be achieved using resource-observe mode in certain situations at
least in theory.

For example, the scenario of section 3.1 may also be achieved using
resource-observe. In that case the backend will have to subscribe to
each of the in-vehicle sensor gateway. The gateways will notify the
backend with updated geo-locations. However, considering the huge
number of vehicles moving around and several being added to the
system quite often, this kind of arrangement may not be as

   practicable and efficient solution as illustrated in the previous
   examples.

   Figure 4 shows the resource observe variant. The No-Response option
   has been used intentionally both with GET and the notifications to
   illustrate the non-applicability of this option in this situation.

```
   Server Client
   |     |
   |<-----+ Header  : GET (MID=0x5d28)
   |  GET | Token   : 0x53
   |      | No-Response: 0
   |      | Uri-Path: vehicle-stat
   |      | Observe : (empty)
   |      |
   |      |
   +----->| Header: 2.05 (T=NON, MID=0x7d38)
   | 2.05 | Token: 0x53
   |      | Content Type: text/plain
   |      | No-Response: 0
   |      | Payload:
   |      | "VehID=00&RouteID=DN47&Lat=22.5658745&Long=88.4107966667&
   |      | Time=2013-01-13T11:24:31"
   |      |
   [Next update in 20 secs.]
   |      |
   +----->| Header: 2.05 (T=NON, MID=0x7d39)
   | 2.05 | Token: 0x53
   |      | Content Type: text/plain
   |      | No-Response: 0
   |      | Payload:
   |      | "VehID=00&&RouteID=DN47&Lat=22.5649015&Long=88.4103511667&
   |      | Time=2013-01-13T11:24:51"
```

    Figure 4: Exemplary unreliable update in resource-observe mode with
      No-Response option where practically No-Response has no effect.

   Note: The reason for keeping this example is to open up the choice
       to the user when there is a possibility of choosing between
       resource-observe with NON and updates with No-Response and to
       show a possible case where the latter option may sound more
       useful.

6. IANA Considerations

   The IANA is requested to add the following option number entries:

```
+--------+-------------+---------------------------+
| Number |    Name     |         Reference         |
+--------+-------------+---------------------------+
|   TBD  | No-Response | Section 4 of this document |
+--------+-------------+---------------------------+
```

7. Security Considerations

The No-Response option defined in this document presents no security
considerations beyond those in Section 11 of the base CoAP
specification [I-D.ietf-core-coap].

8. Acknowledgments

Thanks to Carsten Bormann, Esko Dijk, Bert Greevenbosch, Akbar
Rahman and Claus Hartke for their valuable inputs.

9. References

9.1. Normative References

[I-D.ietf-core-coap]

Shelby, Z., Hartke, K. and Bormann, C.,"Constrained Application
Protocol (CoAP)", draft-ietf-core-coap-18, June 28, 2013

[I-D.ietf-core-observe]

Hartke, K.,"Observing Resources in CoAP", draft-ietf-core-observe-
09, July 15, 2013

[I-D.ietf-core-groupcomm]

Rahman, A. and Dijk, E.,"Group Communication for CoAP", draft-ietf
core-groupcomm-12, July 30, 2013

Authors' Addresses

    Abhijan Bhattacharyya
    Tata Consultancy Services Ltd.
    Kolkata, India

    Email: abhijan.bhattacharyya@tcs.com

    Soma Bandyopadhyay
    Tata Consultancy Services Ltd.
    Kolkata, India

    Email: soma.bandyopadhyay@tcs.com

    Arpan Pal
    Tata Consultancy Services Ltd.
    Kolkata, India

    Email: arpan.pal@tcs.com