Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: October 4, 2014

O. Kleine
University of Luebeck, ITM

April 2, 2014

CoAP Endpoint Identification
draft-kleine-core-coap-endpoint-id-00.txt

Abstract

The Constrained Application Protocol (CoAP) is an application layer protocol for constrained devices (e.g.low power, few memory) and networks (e.g. lossy, low bandwidth) which relies on UDP on the transport layer. With CoAP it is often the case, that message exchanges need to extend the common request/response pattern, e.g. for seperate responses. This holds, e.g. for CON requests that are confirmed by the server with an empty ACK and answered later with a seperate response. According to the CoAP specification the request/response matching is realized using a unique pair of server address and token per client.

Due to the mobile nature of some devices. e.g. smartphones, they are often assigned new IP addresses because of a network change. Thus, the IP address of a CoAP server might change during an ongoing conversion. This draft proposes a method to assign each communication partner with an identifier (endpoint ID) which replaces the IP address as (partial) key to relate requests and responses.

Besides the common seperate responses, the proposed method is also useful to handle IP address changes, e.g. during an ongoing observation ([observe]) or a blockwise transfer ([block]).

Requirements Language

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 inRFC 2119 [RFC2119].

Status of This Memo

Table of Contents

1. Introduction

The concept of confirmable messages (CON) introduced in the main CoAP specification provides reliability in terms of message reception by the remote endpoint, i.e. the recipient of a confirmable message MUST confirm the reception with an acknowledgement (ACK) within 2 seconds. The absence of an ACK causes the sender of the CON message to retransmit the CON message. However, an (empty) ACK just confirms the reception and for confirmable requests this might cause the server to send a separate response containing the actual result of the request processing, i.e. a third message within a single conversation.

According to the CoAP specification the key to match incoming responses with open requests is a token which is defined by the client. This token is set as a part of the requests header and sent to the server. The server includes the same token in the response and by this means enables the client to match the response with a request. The token is unique per communication partner, i.e. a client would use 2 different tokens for 2 parallel requests to a server but may use the same token for 2 parallel requests to different servers. Thus, the client must use the combination of the server address and the token to match incoming responses with open requests.

CoAP servers may run on mobile devices, e.g. smartphones, that are often assigned new IP addresses due to network changes. The assignment of a new IP could happen within an ongoing conversion, i.e. after an empty ACK was sent but before the actual (separate) response. In this case, the client can not match the response with the open request. This draft introduces 2 new CoAP options to deal with this issue and enable ongoing conversations to continue even if one of the endpoints changes its IP address.

Besides the common separate responses, the proposed method is also useful to handle IP address changes, e.g. during an ongoing observation [observe] or a blockwise transfer [block].

2. A "Message Exchange"

A message exchange is considered to consist of all messages that are sent between two endpoints as direct consequence of the first message plus this first message. Thus, according to the CoAP specification (without extensions) a message exchange consists of either 1, 2, 3, or 4 messages.

As NON request do not require a response, it is possible, that a message exchange consists only of a single message (see Figure 1).

```
CLIENT                        SERVER
  |                              |
  | ------ NON (request) ------> |
  |                              |
```

Figure 1: NON request without any response

There are 2 possible types of Message Exchange that consist of 2 messages. Those are depicted in Figure 2 and Figure 3.

```
CLIENT                        SERVER
  |                              |
  | ----- NON (request) -------> |
  |                              |
  | <------ NON (response) ----- |
  |                                |
```

Figure 2: NON requests and NON response

```
CLIENT                                      SERVER
  |                                            |
  | ----- CON (request) ---------------------> |
  |                                            |
  | <------ ACK (piggy-backed response) ----- |
  |                                            |
```

Figure 3: CON request and ACK response (piggy-backed)

Those were the types of Message Exchange that match the common request/response pattern. However, due to the reliability concept of CoAP there are also types of Message Exchange that extends this pattern by consisting of 3 or even 4 messages. The 2 possible types of Message Exchange that consist of 3 messages are depicted in Figure 4 and Figure 5.

```
CLIENT                           SERVER
  |                                |
  | ----- NON (request) --------> |
  |                                |
  | <------- CON (response) ----- |
  |                                |
  | ----- ACK (empty) ----------> |
  |                                |
```

Figure 4: NON requests and CON response

```
CLIENT                                   SERVER
  |                                        |
  | ----- CON (request) ---------------> |
  |                                        |
  | <----------------- ACK (empty) ----- |
  |                                        |
  | <----- NON (seperate response) ----- |
  |                                        |
```

Figure 5: CON request, empty ACK and NON response (seperate)

The last type of Message Exchange consists of 4 messages to be sent and includes reliability for both, request and response (see Figure 6).

```
CLIENT                                   SERVER
  |                                        |
  | ----- CON (request) ---------------> |
  |                                        |
  | <----------------- ACK (empty) ----- |
  |                                        |
  | <----- CON (seperate response) ----- |
  |                                        |
  | ----- ACK (empty) -----------------> |
  |                                        |
```

Figure 6: CON request, empty ACK, CON response, empty ACK

3. Endpoint Identification Options

The Endpoint Identification Extension introduces 2 new (opaque) options. The first option (ENDPOINT_ID_1) is used to assign the communication partner, i.e. the remote CoAP endpoint, a unique ID. The recipient of a CoAP message that contains an ENDPOINT_ID_1 option repeats its value in every follow-up message as value of the ENDPOINT_ID_2 option.

Furthermore, the sender of a CoAP message with ENDPOINT_ID_1 option uses the value as (partial) key for the duration of the conversation instead of the remote endpoints IP address, e.g. for request/response matching in combination with a token. However, this approach does not include CoAPs reliability "layer" as empty ACKs MUST not include any options. Thus, the CON/ACK matching still bases on the combination of remote IP and message ID.

3.1. ENDPOINT_ID_1 option

The ENDPOINT_ID_1 option is set by the sender of a CoAP message to assign the remote endpoint an ID which is supposed to be used to identify this endpoint for the remaining duration of the actual message exchange (see Section 3.2) whenever possible (this does explicitly not include empty messages, e.g. ACK or RST).

If the recipient of a CoAP message with ENDPOINT_ID_1 option does not support the option it MAY ignore that option. As the recipient is supposed to repeat the value of the ENDPOINT_ID_1 option as value of the ENDPOINT_ID_2 option in every follow-up message within a message exchange, the first message origin can derive the lack of support for that option from the missing ENDPOINT_ID_2 option in the follow-up messages.

3.2. ENDPOINT_ID_2 option

The ENDPOINT_ID_2 option is set by the sender of a CoAP message to identify itself as the message origin. The value of the ENDPOINT_ID_2 option repeats the value of the latest ENDPOINT_ID_1 option that was received from the intended recipient of the message to be sent.

Thus, a ENDPOINT_ID_2 option MUST not be set in a CoAP message if the intended recipient did not send a ENDPOINT_ID_1 option in a previous message. If the ENDPOINT_ID_2 option is not supported the message MUST be rejected via RST message. Also ENDPOINT_ID_2 option values that are unknown to the recipient MUST be rejected with a RST message.

3.3. Option syntax and semantics

Type	C	U	N	R	Name	Format	Length	Default
124	E	U	-	-	ENDPOINT_ID_1	opaque	0-4 B	(none)
189	C	U	-	-	ENDPOINT_ID_2	opaque	0-4 B	(none)

Table 1: The endpoint ID option numbers

3.4. Endpoint IDs for obervations

Observing a CoAP resource means to retrieve multiple responses as a consequence of a single request. If the observe option is set in a request and observing is supported by the addressed resource, the client receives another response (update notification) whenever the status of the observed resource changes [observe].

This leads to a new type of Message Exchange consisting of an arbitrary number of messages. Within the duration of an observation relationship between a client and a server, both, the IP of the client and the IP of the server may change.

3.4.1. Client IP changes during observation

The server MUST set the ENDPOINT_ID_1 option in every update notification. By this means, the client is assigned an ID which is independent from its IP address. Whenever the IP address of the client changes during an ongoing observation, the client resends its initial request and adds the assigned ID as value of the ENDPOINT_ID_2 option.

By this means, the server is able to update its internal "client ID/ IP address - mapping" and continue the observation. However, if the request was a CON request, a server MAY only respond with an empty ACK instead of a full response if the observed resource did not change since the last update notification.

3.4.2. Server IP changes during observation

Due to the ENDPOINT_ID_1 option in the request starting the observation, the server is assigned an ID that is independet from its IP address. This ID is to be set as ENDPOINT_ID_2 value in every follow-up response (update notification) within this observation relationship.

By this means, a client is able to update its internal "server ID/IP address - mapping" with every update notification.

4. Examples

Within the figures in this section MID refers to message ID, whereas ID_x refers to the value of the ENDPOINT_ID_x option.

4.1. NON request and NON response

```
CLIENT                                            SERVER
  |                                                  |
  | ----- NON [MID=1234, ID_1=0xab] ----------> |
  |                                                  |
  |                              (Server IP may change)
  |                                                  |
  | <--------- NON [MID=5678, ID_2=0xab] ------ |
  |                                                  |
```

Figure 7: NON requests and NON response

4.2. NON request, CON response, and empty ACK

```
CLIENT                                                    SERVER
  |                                                          |
  | ----- NON [MID=1234, ID_1=0xab] ------------------> |
  |                                                          |
  |                                      (Server IP may change)
  |                                                          |
  | <----------------- CON [MID=5678, ID_2=0xab] ------ |
  |                                                          |
  | ----- ACK [MID=5678] -----------------------------> |
  |                                                          |
```

Figure 8: NON requests and NON response

4.3. CON request, empty ACK, and NON response

```
           CLIENT                                     SERVER
             |                                          |
             | ----- CON [MID=1234, ID_1=0xab] -------> |
             |                                          |
             | <----------------- ACK [MID=1234] ------ |
             |                                          |
             |                           (Server IP may change)
             |                                          |
             | <------ NON [MID=5678, ID_2=0xab] ------ |
             |                                          |
```

Figure 9: NON requests and NON response

4.4. CON request, empty ACK, CON response, and empty ACK

```
      CLIENT                                                 SERVER
        |                                                      |
        | ----- CON [MID=1234, ID_1=0xab] ------------------> |
        |                                                      |
        | <----------------------------- ACK [MID=1234] ----- |
        |                                                      |
        |                                       (Server IP may change)
        |                                                      |
        | <----------------- CON [MID=5678, ID_2=0xab] ------ |
        |                                                      |
        | ----- ACK [MID=5678] -----------------------------> |
        |                                                      |
```

Figure 10: CON request, empty ACK, CON response, and empty ACK

4.5. Server IP address changes during observation

```
   CLIENT                                                      SERVER
      |                                                           |
      | ----- CON [MID=1234, OBS, ID_1=0xab] -------------------> |
      |                                                           |
      | <---------------------------------- ACK [MID=1234] ----- |
      |                                                           |
      |                                                           |
      | <----------------- CON [MID=999, OBS=1, ID_2=0xab] ----- |
      |                                                           |
      | ----- ACK [MID=999] -----------------------------------> |
      |                                                           |
      |                                                           |
      |                                        (Server IP may change)
      |                                                           |
      |                                                           |
      | <---------------- CON [MID=1000, OBS=2, ID_2=0xab] ----- |
      |                                                           |
      | ----- ACK [MID=1000] ----------------------------------> |
      |                                                           |
```

Figure 11: Server IP address changes during observation

4.6. Client IP address changes during observation

```
    CLIENT                                                     SERVER
       |                                                          |
       | ----- CON [MID=1234, OBS, ID_1=0xab] -------------------> |
       |                                                          |
       | <---------------------------------- ACK [MID=1234] ----- |
       |                                                          |
       |                                                          |
       | <------- CON [MID=999, OBS=1, ID_1=0xef, ID_2=0xab] ----- |
       |                                                          |
       | ----- ACK [MID=999] ------------------------------------> |
       |                                                          |
       |                                                          |
    (Client IP may change)                                        |
       |                                                          |
       |                                                          |
       | ----- CON [MID=1235, OBS, ID_1=0xab, ID_2=0xef] --------> |
       |                                                          |
       | <---------------------------------- ACK [MID=1235] ----- |
       |                                                          |
       |                                                          |
       | <------ CON [MID=1000, OBS=2, ID_1=0xef, ID_2=0xab] ----- |
       |                                                          |
       | ----- ACK [MID=1000] -----------------------------------> |
```

Figure 12: Client IP address changes during observation

5. Acknowledgements

No acknowledgements, yet...

6. IANA Considerations

This memo includes no request to IANA.

(It’s good - indeed pretty much mandatory now - to have an explicit note because otherwise IANA wastes cycles trying to figure out if something is needed..)

7. Security Considerations

To avoid an eval interruption of an ongoing Message Exchange, DTLS SHOULD be used to encrypt the CoAP messages.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://xml.resource.org/public/rfc/html/rfc2119.html>.

[coap] Shelby, et.al., , "Contrained Application Protocol", 2013, <https://datatracker.ietf.org/doc/draft-ietf-core-coap/>.

8.2. Informative References

[block] Borman, et al., , "Blockwise transfers in CoAP", 2013, <https://datatracker.ietf.org/doc/draft-ietf-core-block/>.

[observe] Hartke, , "Observing Resources in CoAP", 2014, <https://datatracker.ietf.org/doc/draft-ietf-core-observe/>.

Author’s Address

Oliver Kleine
University of Luebeck, Institute of Telematics
Ratzeburger Allee 160
Luebeck 23552
DE

Email: kleine@itm.uni-luebeck.de