core
Internet-Draft
Intended status: Standards Track
Expires: March 29, 2013

B. Greevenbosch
Huawei Technologies

September 25, 2012

CoAP Minimum Request Interval
draft-greevenbosch-core-minimum-request-interval-00

Abstract

This document defines an "MinimumRequestInterval" option for CoAP, which can be used to negotiate the minimum time between two subsequent requests within a single client and server pair. It can be used for flow and congestion control, reducing the consumption of server and network resources when needed.

Note

Discussion and suggestions for improvement are requested, and should be sent to core@ietf.org.

Status of this Memo

This Internet-Draft will expire on March 29, 2013.

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [I-D.ietf-core-coap] is a RESTful protocol for constrained nodes and networks.

This document defines a "MinimumRequestInterval" option, which can be used to negotiate the minimum time between two subsequent requests within a single client and server pair.

Negotiating the minimum time between the requests can be used to limit the associated traffic, thereby reducing network congestion. In addition, it allows constrained servers to limit the number of requests they receive within a certain time period, preventing them from becoming overloaded.

The mechanism is especially useful for a block transaction, as defined in [I-D.ietf-core-block]. However it can also be used for other transactions involving multiple requests from the client, for example when the client browses the server’s resources.

2. Requirements notation

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 [RFC2119].

3. Definitions

transaction
A series of request/response pairs within a unique client and server pair.

block transaction
A transaction which consists of the transfer of a single source using the block mechanism.

two subsequent requests
Two requests within a single transaction, in which one request follows the other request, without a third request from the transaction in between.

request interval
The time between two subsequent requests.

request speed
The multiplicative inverse of the request interval.

ms
Milliseconds or mibiseconds, depending on the implementation.

mibisecond
1/1024 of a second.

4. Motivation

It would be beneficial for the server to control the amount of requests it receives from the client within a certain time period. In this way, the server can achieve better usage of its internal resources, such as memory, processor load and message buffers. Limiting the number of incoming requests increases the reliability in responding to them, and decreases the chance on server overload.

One method to reduce the client’s request speed is for the server to delay sending its ACKs. This indeed can slow down the client, especially in case the client only issues a new request after receipt of the ACK of the previous request. However, it has the disadvantage that the server has to keep the transaction open, and needs to use resources for delaying the ACK that could have been used to perform other tasks.

If, however, the server can explicitly signal the client’s request speed, then the server does not need to keep track of its own minimum time to respond to each request, and can handle requests as soon as possible. This allows the server to use its resources for other tasks sooner. Since all clients will have a better probability that their requests are handled and that they will receive responses, the overall system’s reliability is increased.

5. The "MinimumRequestInterval" option

Type	C/E	Name	Format	Length	Default
TBD	E	MinimumRequestInterval	uint	0-2B	0

Table 1: The "MinimumRequestInterval" option

The "MinimumRequestInterval" option is an elective option, which is used to negotiate the minimum time in ms that a client needs to wait between sending two subsequent requests.

In the remainder of this section, it is assumed that both the client and the server support the "MinimumRequestInterval" option.

If the client plans to perform a transaction consisting of multiple requests, it SHOULD include the "MinimumRequestInterval" option in the first request of the transaction.

The server MUST include the "MinimumRequestInterval" option in a response to a request that contained a "MinimumRequestInterval" option.

If a client receives a response with the "MinimumRequestInterval" option, it MUST include the "MinimumRequestInterval" in its subsequent request.

In the request, the option’s value T_C is the request interval the client is currently using. An exception is the first request in the transaction, in which case the value T_C is a proposed request interval.

In a response, the option’s value T_S indicates the minimum request interval in ms that the server can support at that particular moment. Depending on its workload, the server MAY increase or decrease the latest value of T_C to form T_S.

The client SHALL wait at least T_S ms between sending two subsequent requests. It MAY also send at a slower speed.

The "MinimumRequestInterval" option has a default value 0. A value T_S=0 indicates the server does not put any restrictions on the transaction speed. Similiarly value T_C=0 in the first request indicates that the client prefers to send the following requests as quickly as possible.

6. Legacy behaviour

It is possible that either the client or server does not support the "MinimumRequestInterval" option. If the client does not support the option, then obviously it cannot take the server’s preference into account. Similarly if the server does not support the option, it cannot use it to restrict the transaction speed.

In either case, or their combination, the client will choose the transaction speed as it prefers. This corresponds to the case T_S=0.

To allow the server to distinguish between a client that supports the "MinimumRequestInterval" option but wants to signal T_C=0, and a client that does not support the "MinimumRequestInterval" option, it is RECOMMENDED for the compliant client to include the option in the requests of a multiple request transaction, even when the client wants to signal T_C=0.

7. Example

Figure 1 contains an example of a block transaction with the "MinimumRequestInterval" option.

In the first request, the client proposes a minimum request interval of T_C=150ms. As the server is too busy, it wants to slow down the client and returns a minimum request interval of T_S=200ms.

The client uses this request interval for the timing of the next requests, and keeps informing the server of its current request speed. Likewise, in the first several messages the server echos the T_C in T_S, signalling that it is comfortable with the current request speed.

After sending three blocks, the server becomes less busy. It therefore increases the allowed request speed by signalling a new T_S=150ms. The client uses this speed until the end of the transaction.

```
           CLIENT                                                 SERVER
              |                                                     |
        /     | CON [MID=1234], GET, /status, N=0, T_C=150 -------> |
        |     |                                                     |
      200ms   | <------- ACK [MID=1234], 2.05 Content, N=0, T_S=200 |
        |     |                                                     |
        \  /  | CON [MID=1235], GET, /status, N=1, T_C=200 -------> |
           |  |                                                     |
         200ms| <------- ACK [MID=1235], 2.05 Content, N=1, T_S=200 |
           |  |                                                     |
        /  \  | CON [MID=1234], GET, /status, N=2, T_C=200 -------> |
        |     |                                                     |
      200ms   | <------- ACK [MID=1234], 2.05 Content, N=2, T_S=200 |
        |     |                                                     |
        \  /  | CON [MID=1235], GET, /status, N=3, T_C=200 -------> |
           |  |                                                     |
         150ms| <------- ACK [MID=1235], 2.05 Content, N=3, T_S=150 |
           |  |                                                     |
        /  \  | CON [MID=1234], GET, /status, N=4, T_C=150 -------> |
        |     |                                                     |
      150ms   | <------- ACK [MID=1234], 2.05 Content, N=4, T_S=150 |
        |     |                                                     |
        \     | CON [MID=1235], GET, /status, N=5, T_C=150 -------> |
              :                                                     :
              :                        ...                          :
              :                                                     :
```

Figure 1: Example of transaction with "MinimumRequestInterval"

8. Security Considerations

By modifying the value of the "MinimumRequestInterval" option in a response to a higher value, a man-in-the-middle could increase the time used to perform a transaction. When the client encounters a response with a too high "MinimumRequestInterval" value, it MAY abort the transaction, and try to reinitiate it. However, to prevent overloading the server, the client MUST limit the number of these reinitiations.

By decreasing the value of the "MinimumRequestInterval" option in a response, the man-in-the-middle can induce the client to send requests at a speed too high for the server. The server should be prepared for this, for example by discarding requests that cannot be processed. This is similar to the case where the server or client does not support the "MinimumRequestInterval" option.

By altering the value of the "MinimumRequestInterval" option in a request, the man-in-the-middle can induce the server to believe that the client is using another transaction speed than it really is. This could lead to a false adjustment of the request interval.

All these attacks depend on the man-in-the-middle being able to modify multiple messages, as the speed would otherwise stabilise again after several adjustments by the server.

9. IANA Considerations

This draft adds the following option numbers to the CoAP Option Numbers registry of [I-D.ietf-core-coap].

Number	Name	Reference
TBD (elective)	MinimumRequestInterval	[RFCXXXX]

Table 2: CoAP option numbers

10. Acknowledgements

The author would like to thank Carsten Borrman, Esko Dijk and Kepeng Li for their feedback.

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.ietf-core-block] Shelby, Z. and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-block-08 (work in progress), February 2012.

[I-D.ietf-core-coap] Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-10 (work in progress), March 2012.

Author's Address

Bert Greevenbosch
Huawei Technologies Co., Ltd.
Huawei Industrial Base
Bantian, Longgang District
Shenzhen 518129
P.R. China

Phone: +86-755-28978088
Email: bert.greevenbosch@huawei.com