CoRE
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2015

C. Bormann
Universitaet Bremen TZI

July 04, 2014

A TCP transport for CoAP
draft-bormann-core-coap-tcp-01

Abstract

CoAP (RFC 7252) is defined to be transported over datagram transports such as UDP or DTLS. For a number of applications, it may be useful to channel CoAP messages in a TCP connection. This draft discusses different ways to do that.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2015.

Copyright Notice

Table of Contents

1. Introduction

(See Abstract)

The primary use case addressed by this specification is:

- Aggregation of CoAP streams behind proxies, e.g.:
  - Behind a DTLS terminator/load balancer on the cloud side
  - As a wide-area interface to a proxy that speaks CoAP over UDP on the constrained side

1.1. Objectives

(TBD)

1.2. Terminology

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].

The term "byte" is used in its now customary sense as a synonym for "octet".

All multi-byte integers in this protocol are interpreted in network byte order.

2. Framing

The TCP stream needs to be structured into frames in order to delimit CoAP messages.

As the size of CoAP messages is limited, there is no need to split a single CoAP message into multiple frames (no interleaving).

Several alternative frame formats are possible. The current version of this specifications proposes several alternatives, with the understanding that a single one of these is likely to be chosen.

One desirable characteristic of a framing scheme is detection of premature termination of the TCP connection. While TCP in principle distinguishes orderly (FIN) and destructive (RST) termination of a connection, the difference is not always visible from the socket interface; also, a crashing process gives the impression of orderly termination. All schemes proposed here provide this detection.

2.1. Length prefix

A popular form of framing for TCP starts each frame with a length indication [RFC1006].

A simple form of length prefix would be an SDNV [RFC6256], which is efficient for large numbers of mostly small (< 128 B) messages. Alternatively, a two-byte prefix could always be used, or the length could be embedded in the CoAP message by using the unused Message-ID field.

The main disadvantage of a length prefix is that the sender needs to know the length before sending the message proper. The main advantage of a length prefix is that the receiver knows the length at the start of receiving the message.

2.2. Delimiter-based

Another form of message delimiting uses special byte values for delimiting protocol elements, e.g. CRLF for lines in a text stream. Since CoAP requires full data transparency, introducing a delimiter byte requires escaping occurrences of the delimiter in the data stream, which in turn requires escaping the escape mechanism. In traditional byte-stuffing (called "octet-stuffing" in [RFC1662]), the overhead of this escaping can be up to 100 % on top of the actual data. Cheshire has shown how to combine delimiter-based and length

prefix based encoding in "Consistent Overhead Byte Stuffing" [COBS]; however, this requires at least two bytes per message, achieving full efficiency only for relatively large messages and only if the length of the remaining message is known (see Section 2.1 before). A scheme such as the FSE scheme in [RFC2687] might be simpler to implement (the efficiency of an FSE-style scheme can be quite high by exploiting the fact that CoAP frames never start with a value below 0x40). A good value for FSE-style (or even a non-zero COBS-style) delimiter can be determined by examining a corpus of CoAP messages (TBD).

A major advantage of a COBS-like scheme would be compatibility with schemes that synchronize TCP packet boundaries with message boundaries [MINION].

Requiring a delimiter at the _end_ of a frame fulfills the requirement for detection of premature TCP connection termination, except for an FSE-style scheme where the FSE starting an escape sequence happens to fall on a packet boundary.

2.3. Self-delimiting

Currently, CoAP messages are not self-delimiting, as the payload delimiter is optional and does not contain a payload length.

In the scheme proposed here, the payload delimiter is made required; the payload length is then encoded exactly as in CoAP options. For example:

- 0xF0 would indicate that a zero-length (absent) payload follows
- 0xF1 would indicate a single-byte payload
- 0xFD 0x47 would indicate a 84-byte payload
- 0xFE 0xFF 0xFF would indicate a 65804-byte payload

One advantage of implementing this scheme is that it could also be used to aggregate multiple CoAP messages into one datagram of a datagram-based transport such as UDP or DTLS, if that is desired, without increasing the overhead for unaggregated messages. For this application, 0xFF could still be used in order to efficiently encode "payload delimited by message boundary" in the final CoAP message in the datagram.

3. Changes to CoAP

The content of this section is expressed as a delta on [RFC7252].

3.1. One Message Type Only

As reliability is handled by TCP, there is no need for ACK messages. Similarly, rebooting nodes will drop their TCP connections, so there is no need for RST messages (but see Section 3.4).

Cf. [I-D.savolainen-core-coap-websockets].

There may still be a desire to differentiate CON and NON for the intention behind a TCP-to-UDP proxy. In contrast to [I-D.savolainen-core-coap-websockets], this specification proposes to retain the difference between CON and NON messages as a hint for the reliability requirements placed on a message forwarded through a proxy. There are no ACK or RST messages; ACK messages MUST be encoded as CON messages.

3.2. Token

The Token space is local to the TCP connection. In particular, this means that closing down a TCP connection cancels all outstanding requests (the responses are not sent over a new connection, which is handled like a new endpoint).

3.3. Message-ID, Fixed Header Format

As there are no ACKs, there is no need to correlate an ACK to a CON. As a result, there is no need to carry a Message-ID for this. There is also no danger of duplication of a message, so the Message-ID is entirely without function.

If it seems desirable to maintain the frame format, the message-ID could still be sent empty. Alternatively, it could be used as a space for the frame length.

As does [I-D.savolainen-core-coap-websockets], this specification proposes to elide the Message-ID, i.e. to send bytes 0 and 1 of the CoAP message followed directly by byte 4 and following.

3.4. Rejecting messages

[I-D.ietf-core-observe] now supports Observation Cancellation, reducing the need to support Reset-like messages for cancelling an observation relationship.

3.5. Resilient variant

An alternative approach is to treat the TCP connection as ephemeral. If a connection can go away at any point in time, and be replaced by a new one, the delivery of messages is no longer fully reliable. All functions of Message-IDs remain, as well as the functions of ACKs. Tokens retain their meaning beyond a connection.

3.6. Signatures

A new TCP connection can send an identifying signature in both directions to facilitate debugging and protocol evolution and to enable detection of mismatches.

E.g., the side opening a connection could send the seven bytes "CoAP1\r\n" and the answering side similarly "cOap1\r\n".

4. Transport selection

There may be use cases where the TCP transport should be explicitly selected from a URI. This problem should be solved in a way that doesn't cause the available of different transports to generate aliases for the same resource, i.e. the same "coap://" URI should be used for the same resource. Cf. [I-D.silverajan-core-coap-alternative-transports].

5. References

5.1. Normative References

[I-D.ietf-core-observe] Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-14 (work in progress), June 2014.

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

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

5.2. Informative References

[COBS] Cheshire, S. and M. Baker, "Consistent Overhead Byte Stuffing", ToN Vol.7, No. 2, April 1999.

[I-D.savolainen-core-coap-websockets]
Savolainen, T., Hartke, K., and B. Silverajan, "CoAP over WebSockets", draft-savolainen-core-coap-websockets-02 (work in progress), April 2014.

[I-D.silverajan-core-coap-alternative-transports]
Silverajan, B. and T. Savolainen, "CoAP Communication with Alternative Transports", draft-silverajan-core-coap-alternative-transports-05 (work in progress), June 2014.

[MINION] Iyengar, J., Ford, B., Amin, S., Nowlan, M., and N. Tiwari, "Wire-Compatible Unordered Delivery in TCP and TLS", CoRR abs/1103.0463, February 2011, <http://arxiv.org/abs/1103.0463v2>.

[RFC1006] Rose, M. and D. Cass, "ISO transport services on top of the TCP: Version 3", STD 35, RFC 1006, May 1987.

[RFC1662] Simpson, W., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.

[RFC2687] Bormann, C., "PPP in a Real-time Oriented HDLC-like Framing", RFC 2687, September 1999.

[RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", RFC 6256, May 2011.

Author’s Address

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org