CORE
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2015

C. Bormann, Ed.
Universitaet Bremen TZI
S. Lemay
V. Solorzano Barboza
Zebra Technologies
H. Tschofenig
ARM Ltd.

June 10, 2015

A TCP and TLS Transport for the Constrained Application Protocol (CoAP)
draft-tschofenig-core-coap-tcp-tls-04.txt

Abstract

The Hypertext Transfer Protocol (HTTP) has been designed with TCP as an underlying transport protocol. The Constrained Application Protocol (CoAP), which has been inspired by HTTP, has on the other hand been defined to make use of UDP. Therefore, reliable delivery and a simple congestion control and flow control mechanism are provided by the message layer of the CoAP protocol.

A number of environments benefit from the use of CoAP directly over a reliable byte stream that already provides these services. This document defines the use of CoAP over TCP as well as CoAP over TLS.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] was designed for Internet of Things (IoT) deployments, assuming that UDP can be used freely - UDP [RFC0768], or DTLS [RFC6347] over UDP, is a good choice for transferring small amounts of data in networks that follow the IP architecture. Some CoAP deployments, however, may have to integrate well with existing enterprise infrastructure, where the use of UDP-based protocols may not be well-received or may even be blocked by firewalls. Middleboxes that are unaware of CoAP usage for IoT can make the use of UDP brittle.

Where NATs are still present, CoAP over TCP can also help with their traversal. NATs often calculate expiration timers based on the transport layer protocol being used by application protocols. Many NATs are built around the assumption that a transport layer protocol such as TCP gives them additional information about the session life cycle and keep TCP-based NAT bindings around for a longer period. UDP on the other hand does not provide such information to a NAT and timeouts tend to be much shorter, as research confirms [HomeGateway].

Some environments may also benefit from the more sophisticated congestion control capabilities provided by many TCP implementations. (Note that there is ongoing work to add more elaborate congestion control to CoAP as well, see [I-D.bormann-core-cocoa].)

Finally, CoAP may be integrated into a Web environment where the front-end uses CoAP from IoT devices to a cloud infrastructure but the CoAP messages are then transported in TCP between the back-end services. A TCP-to-UDP gateway can be used at the cloud boundary to talk to the UDP-based IoT.

To make both IoT devices work smoothly in these demanding environments, CoAP needs to make use of a different transport protocol, namely TCP [RFC0793] and in some situations even TLS [RFC5246].

The present document document describes a shim header that conveys length information about each CoAP message included. Modifications to CoAP beyond the replacement of the message layer (e.g., to introduce further optimizations) are intentionally avoided.

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 [RFC2119].

3. Constrained Application Protocol

The interaction model of CoAP over TCP is very similar to the one for CoAP over UDP with the key difference that TCP voids the need to provide certain transport layer protocol features, such as reliable delivery, fragmentation and reassembly, as well as congestion control, at the CoAP level. The protocol stack is illustrated in Figure 1 (derived from [RFC7252], Figure 1).

```
           +----------------------+
           |      Application     |
           +----------------------+
           +----------------------+
           |  Requests/Responses  |  CoAP (RFC7252)
           |----------------------|
           |    Message adapter   |  this document
           +----------------------+
           +-----------+    ^
           |    TLS    |    |
           +-----------+    v
           +----------------------+
           |          TCP         |
           +----------------------+
```

Figure 1: The CoAP over TLS/TCP Protocol Stack

TCP offers features that are not available in UDP and consequently have been provided in the message layer of CoAP. Since TCP offers reliable delivery, there is no need to offer a redundant acknowledgement at the CoAP messaging layer.

Hence, the only message type transported when using CoAP over TCP is the Non-Confirmable message (NON). By nature of TCP, a NON over TCP is still transmitted reliably. Figure 2 (derived from [RFC7252], Figure 3) shows this message exchange graphically. A UDP-to-TCP gateway will therefore discard all empty messages, such as empty ACKs (after operating on them at the message layer), and re-pack the contents of all non-empty CON, NON, or ACK messages (i.e., those ACK messages that have a piggy-backed response) into NON messages.

Similarly, there is no need to detect duplicate delivery of a message. In UDP CoAP, the Message ID is used for relating acknowledgements to Confirmable messages as well as for duplicate detection. Since the Message ID thus is not meaningful over TCP, it is elided (as indicated by the dashes in Figure 2).

```
           Client              Server
              |                   |
              |   NON [------]    |
              +------------------>|
              |                   |
```

Figure 2: NON Message Transmission over TCP.

As a result of removing the message layer in CoAP over TCP, there is no longer a need to distinguish message types. Since the two-bit field for the message needs to be filled with something, all messages are marked with the bit combination indicating the NON type (no message layer acknowledgement is expected or even possible). A response is sent back as defined in [RFC7252], as illustrated in Figure 3 (derived from [RFC7252], Figure 6).

```
        Client              Server
           |                  |
           |   NON [------]   |
           | GET /temperature |
           |   (Token 0x74)   |
           +----------------->|
           |                  |
           |   NON [------]   |
           |   2.05 Content   |
           |   (Token 0x74)   |
           |     "22.5 C"     |
           |<-----------------+
           |                  |
```

Figure 3: NON Request/Response.

4. Message Format

The CoAP message format defined in [RFC7252], as shown in Figure 4, relies on the datagram transport (UDP, or DTLS over UDP) for keeping the individual messages separate.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Ver| T |  TKL  |      Code     |          Message ID           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Token (if any, TKL bytes) ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Options (if any) ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 1 1 1 1 1 1 1|    Payload (if any) ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: RFC 7252 defined CoAP Message Format.

In a stream oriented transport protocol such as TCP, some other form of delimiting messages is needed. For this purpose, CoAP over TCP introduces a length field. Figure 5 shows a 1-byte shim header carrying length information prepending the CoAP message header.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Length Shim  |Ver| T |  TKL  |      Code     |   Token (TKL
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   bytes) ...  |  Options (if any) ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 1 1 1 1 1 1 1|    Payload (if any) ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: CoAP Header with prepended Shim Header.

-- Alternative L1 --

The ’Message Length’ field is a 16-bit unsigned integer in network byte order.

-- Alternative L2 --

The ’Message Length’ field starts with an 8-bit unsigned integer. Length encoding follows the same mechanism as "Major type 0" from the CBOR specification [RFC7049]. The length field is indicated by the 5 least significant bits of the byte. Values are used as such:

o between 0b000_00001 and 0b000_10111 (1 to 23) indicates the actual length of the following message

o 0b000_11000 (24) means an additional 8-bit unsigned Integer is appended to the initial length field indicating the total length

o 0b000_11001 (25) means an additional 16-bit unsigned Integer (in network byte order) is appended to the initial length field indicating the total length

o 0b000_11010 (26) means an additional 32-bit unsigned Integer (in network byte order) is appended to the initial length field indicating the total length

The 3 most significant bits in the initial length field are reserved for future use. If a recipient gets a message larger than it can handle, it SHOULD if possible send back a 4.13 in accordance with [RFC7252] section on error code.

-- Common for L1 and L2 Alternatives --

The "length" field provides the length of the subsequent CoAP message (including the CoAP header but excluding this message length field) in bytes. T is always the code for NON (1).

-- Alternative L3 --

The initial byte of the frame contains two nibbles, in a similar way to the CoAP option encoding (Section 3.1 of [RFC7252]). The first nibble is used to indicate the length of the options (including any option delimiter), and the payload (if any); it does not include the Code byte or the Token bytes. The first nibble is interpreted as a 4-bit unsigned integer. A value between 0 and 12 directly indicates the length of the options/payload, in bytes. The other three values have a special meaning:

13: An 8-bit unsigned integer follows the initial byte and indicates the length of options/payload minus 13.

14: A 16-bit unsigned integer in network byte order follows the initial byte and indicates the length of options/payload minus 269.

15: A 32-bit unsigned integer in network byte order follows the initial byte and indicates the length of options/payload minus 65805.

The second nibble of the initial byte indicates the token length.

Example: 01 43 7f is a frame just containing a 2.03 code with the token 7f.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Len  |  TKL  | Len+ bytes... |      Code     | TKL bytes ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Options (if any) ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 1 1 1 1 1 1 1|    Payload (if any) ...
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: CoAP Header with prepended Shim Header (L3).

-- End L Alternatives

The Message ID is meaningless and thus elided. The semantics of the other CoAP header fields is left unchanged.

4.1. Discussion

One might wish that, when CoAP is used over TLS, then the TLS record layer length field could be used in place of the shim header length. Each CoAP message would be transported in a separate TLS record layer message, making the shim header that includes the length information redundant.

However, RFC 5246 says that "Client message boundaries are not preserved in the record layer (i.e., multiple client messages of the same ContentType MAY be coalesced into a single TLSPlaintext record, or a single message MAY be fragmented across several records)." While the Record Layer provides length information about the encapsulated application data and handshaking payloads, TLS implementations typically do not support an API interface that would provide access to the record layer delimiting information. An additional problem with this approach is that this approach would remove the potential optimization of packing several CoAP messages into one record layer message, which is normally a way to amortize the record layer and MAC overhead over all these messages.

In summary, we are not pursuing this idea for an optimization.

One other observation is that the message size limitations defined in Section 4.6 of [RFC7252] are no longer strictly necessary. Consenting [how?] implementations may want to interchange messages with payload sizes than 1024 bytes, potentially also obviating the need for the Block protocol [I-D.ietf-core-block]. It must be noted that entirely getting rid of the block protocol is not a generally applicable solution, as:

- o a UDP-to-TCP gateway may simply not have the context to convert a message with a Block option into the equivalent exchange without any use of a Block option.
- o large messages might also cause undesired head-of-line blocking.

The general assumption is therefore that the block protocol will continue to be used over TCP, even if applications occasionally do exchange messages with payload sizes larger than desirable in UDP.

5. Message Transmission

As CoAP exchanges messages asynchronously over the TCP connection, the client can send multiple requests without waiting for responses. For this reason, and due to the nature of TCP, responses are returned during the same TCP connection as the request. In the event that the connection gets terminated, all requests that have not elicited a response yet are canceled; clients are free to transmit the request again once a connection is reestablished.

Furthermore, since TCP is bidirectional, requests can be sent from both the connecting host or the endpoint that accepted the connection. In other words, who initiated the TCP connection has no bearing on the meaning of the CoAP terms client and server, which are relating only to an individual request and response pair.

6. CoAP URI

CoAP [RFC7252] defines the "coap" and "coaps" URI schemes for identifying CoAP resources and providing a means of locating the resource. RFC 7252 defines these resources for use with CoAP over UDP.

The present specification introduces two new URI schemes, namely "coap+tcp" and "coaps+tcp". The rules from Section 6 of [RFC7252] apply to these two new URI schemes.

[RFC7252], Section 8 (Multicast CoAP), does not apply to the URI schemes defined in the present specification.

Resources made available via one of the "coap+tcp" or "coaps+tcp" schemes have no shared identity with the other scheme or with the "coap" or "coaps" scheme, even if their resource identifiers indicate the same authority (the same host listening to the same port). The schemes constitute distinct namespaces and, in combination with the authority, are considered to be distinct origin servers.

6.1. coap+tcp URI scheme

```
coap-tcp-URI = "coap+tcp:" "//" host [ ":" port ] path-abempty
               [ "?" query ]
```

The semantics defined in [RFC7252], Section 6.1, applies to this URI scheme, with the following changes:

- o The port subcomponent indicates the TCP port at which the CoAP server is located. (If it is empty or not given, then the default port 5683 is assumed, as with UDP.)

6.2. coaps+tcp URI scheme

```
coaps-tcp-URI = "coaps+tcp:" "//" host [ ":" port ] path-abempty
                [ "?" query ]
```

The semantics defined in [RFC7252], Section 6.2, applies to this URI scheme, with the following changes:

- o The port subcomponent indicates the TCP port at which the TLS server for the CoAP server is located. If it is empty or not given, then the default port 443 is assumed (this is different from the default port for "coaps", i.e., CoAP over DTLS over UDP).

- o When CoAP is exchanged over TLS port 443 then the "TLS Application Layer Protocol Negotiation Extension" [RFC7301] MUST be used to allow demultiplexing at the server-side unless out-of-band information ensures that the client only interacts with a server that is able to demultiplex CoAP messages over port 443. This would, for example, be true for many Internet of Things deployments where clients are pre-configured to only ever talk with specific servers. [[_1: Shouldn’t we simply always require ALPN? The protocol should not be defined in such a way that it depends on some undefined pre-configuration mechanism. --cabo]]

7. Security Considerations

This document defines how to convey CoAP over TCP and TLS. It does not introduce new vulnerabilities beyond those described already in the CoAP specification. CoAP [RFC7252] makes use of DTLS 1.2 and this specification consequently uses TLS 1.2 [RFC5246]. CoAP MUST NOT be used with older versions of TLS. Guidelines for use of cipher suites and TLS extensions can be found in [I-D.ietf-dice-profile].

8. IANA Considerations

8.1. Service Name and Port Number Registration

IANA is requested to assign the port number 5683 and the service name "coap+tcp", in accordance with [RFC6335].

Service Name.
   coap+tcp

Transport Protocol.
   tcp

Assignee.
   IESG <iesg@ietf.org>

Contact.
   IETF Chair <chair@ietf.org>

Description.

Constrained Application Protocol (CoAP)

Reference.
[RFCthis]

Port Number.
5683

Similarly, IANA is requested to assign the service name "coaps+tcp", in accordance with [RFC6335]. However, no separate port number is used for "coaps" over TCP; instead, the ALPN protocol ID defined in Section 8.3 is used over port 443.

Service Name.
coaps+tcp

Transport Protocol.
tcp

Assignee.
IESG <iesg@ietf.org>

Contact.
IETF Chair <chair@ietf.org>

Description.
Constrained Application Protocol (CoAP)

Reference.
[RFC7301], [RFCthis]

Port Number.
443 (see also Section 8.3 of [RFCthis]})

8.2. URI Schemes

This document registers two new URI schemes, namely "coap+tcp" and "coaps+tcp", for the use of CoAP over TCP and for CoAP over TLS over TCP, respectively. The "coap+tcp" and "coaps+tcp" URI schemes can thus be compared to the "http" and "https" URI schemes.

The syntax of the "coap" and "coaps" URI schemes is specified in Section 6 of [RFC7252] and the present document re-uses their semantics for "coap+tcp" and "coaps+tcp", respectively, with the exception that TCP, or TLS over TCP is used as a transport protocol.

IANA is requested to add these new URI schemes to the registry established with [RFC4395].

8.3. ALPN Protocol ID

This document requests a value from the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" created by [RFC7301]:

Protocol:
CoAP

Identification Sequence:
0x63 0x6f 0x61 0x70 ("coap")

Reference:
[RFCthis]

9. Acknowledgements

We would like to thank Stephen Berard, Geoffrey Cristallo, Olivier Delaby, Michael Koster, Matthias Kovatsch, Szymon Sasin, and Zach Shelby for their feedback.

10. References

10.1. Normative References

[I-D.ietf-dice-profile]
Tschofenig, H. and T. Fossati, "A TLS/DTLS Profile for the Internet of Things", draft-ietf-dice-profile-12 (work in progress), May 2015.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

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

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

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

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, July 2014.

10.2. Informative References

[HomeGateway]
Eggert, L., "An experimental study of home gateway characteristics", Proceedings of the 10th annual conference on Internet measurement, 2010.

[I-D.bormann-core-cocoa]
Bormann, C., Betzler, A., Gomez, C., and I. Demirkol, "CoAP Simple Congestion Control/Advanced", draft-bormann-core-cocoa-02 (work in progress), July 2014.

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Block-wise transfers in CoAP", draft-ietf-core-block-17 (work in progress), March 2015.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

Authors’ Addresses

Carsten Bormann (editor)
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

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

Simon Lemay
Zebra Technologies
820 W. Jackson Blvd.suite 700
Chicago 60607
United States of America

Phone: +1-847-634-6700
Email: slemay@zebra.com

Valik Solorzano Barboza
Zebra Technologies
820 W. Jackson Blvd. suite 700
Chicago 60607
United States of America

Phone: +1-847-634-6700
Email: vsolorzanobarboza@zebra.com

Hannes Tschofenig
ARM Ltd.
110 Fulbourn Rd
Cambridge CB1 9NJ
Great Britain

Email: Hannes.tschofenig@gmx.net
URI: http://www.tschofenig.priv.at