CORE
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

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

March 9, 2015

A TCP and TLS Transport for the Constrained Application Protocol (CoAP)

draft-tschofenig-core-coap-tcp-tls-02.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

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 September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Internet protocol stack is organized in layers, namely link layer, internet layer, transport layer, and the application layer ([RFC1122]).

IP emerged as the waist of the hour glass and supports a variety of link layers and new link layer technologies can be added in the future, without affecting IP.

Combined with the end-to-end principle, the hour glass indicates the level of protocol understanding that intermediaries need to have in order to forward IP packets between a sender and a receiver (absent any specific application layer entities, such as proxies or caches). Having IP as the waist means that anyone can extend the layers above the network layer in the way they want to communicate end-to-end, including defining new transport layer protocols.

Unfortunately, some network deployments depart from this architecture. 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 even supported by firewalls. Middleboxes that are unaware of the IoT can make the use of UDP brittle.

As a separate consideration, some environments benefit from the more advanced congestion control and flow control capabilities provided by TCP. For instance, CoAP back-end processors in a cloud environment may want to connect between each other via TCP instead of UDP; a TCP-to-UDP gateway can be used at the cloud boundary to talk to the UDP-based IoT.

To make both IoT devices and their associated back-end processors 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 supported 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, the only supported message type from the ones CoAP over UDP provides is the

NON type. 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 the 2-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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        Message Length         |Ver| T |  TKL  |      Code     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Token (if any, TKL bytes) ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Options (if any) ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 1 1 1 1 1 1 1|    Payload (if any) ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: CoAP Header with prepended Shim Header.

The 'Message Length' field is a 16-bit unsigned integer in network byte order. It 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). 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 of subsequent 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:

- 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.
- 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. 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.

5.1. 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:

- 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.)

5.2. 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:

- 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).

- 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? --cabo]]

6. 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].

7. IANA Considerations

7.1. 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 7.3 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 7.3 of [RFCthis]})

7.2. 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].

7.3. 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]

8. Acknowledgements

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

9. References

9.1. 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-10 (work in progress), March 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.

9.2. Informative References

[I-D.ietf-core-block] Bormann, C. and Z. Shelby, "Blockwise transfers in CoAP", draft-ietf-core-block-16 (work in progress), October 2014.

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

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[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.

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