Network Working Group R. Jesup
Internet-Draft Mozilla
Intended status: Standards Track S. Loreto
Expires: December 11, 2014 Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences
June 9, 2014

WebRTC Data Channels
draft-ietf-rtcweb-data-channel-10.txt

Abstract

The Real-Time Communication in WEB-browsers working group is charged to provide protocol support for direct interactive rich communication using audio, video, and data between two peers' web-browsers. This document specifies the non-SRTP media data transport aspects of the WebRTC framework. It provides an architectural overview of how the Stream Control Transmission Protocol (SCTP) is used in the WebRTC context as a generic transport service allowing WEB-browsers to exchange generic data from peer to peer.

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 December 11, 2014.

Copyright Notice

Copyright (c) 2014 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

Non-SRTP media data types in the context of WebRTC are handled by using SCTP [RFC4960] encapsulated in DTLS [RFC6347].

```
+----------+
|   SCTP   |
+----------+
|   DTLS   |
+----------+
| ICE/UDP  |
```

```
                              +----------+

                     Figure 1: Basic stack diagram
```


The encapsulation of SCTP over DTLS (see [I-D.ietf-tsvwg-sctp-dtls-encaps]) over ICE/UDP (see [RFC5245]) provides a NAT traversal solution together with confidentiality, source authentication, and integrity protected transfers. This data transport service operates in parallel to the SRTP media transports, and all of them can eventually share a single transport-layer port number.

SCTP as specified in [RFC4960] with the partial reliability extension defined in [RFC3758] and the additional policies defined in [I-D.ietf-tsvwg-sctp-prpolicies] provides multiple streams natively with reliable, and the relevant partially-reliable delivery modes for user messages. Using the reconfiguration extension defined in [RFC6525] allows to increase the number of streams during the lifetime of an SCTP association and to reset individual SCTP streams. Using [I-D.ietf-tsvwg-sctp-ndata] allows to interleave large messages to avoid the monopolization and adds the support of prioritizing of SCTP streams.

The remainder of this document is organized as follows: Section 3 and Section 4 provide use cases and requirements for both unreliable and reliable peer to peer data channels; Section 5 discusses SCTP over DTLS over UDP; Section 6 provides the specification of how SCTP should be used by the WebRTC protocol framework for transporting non-SRTP media data between WEB-browsers.

2. Conventions

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. Use Cases

This section defines use cases specific to data channels. For general use cases see [I-D.ietf-rtcweb-use-cases-and-requirements].

3.1. Use Cases for Unreliable Data Channels

U-C 1: A real-time game where position and object state information is sent via one or more unreliable data channels. Note that at any time there may be no SRTP media channels, or all SRTP media channels may be inactive, and that there may also be reliable data channels in use.

U-C 2: Providing non-critical information to a user about the reason for a state update in a video chat or conference, such as mute state.

3.2. Use Cases for Reliable Data Channels

U-C 3: A real-time game where critical state information needs to be transferred, such as control information. Such a game may have no SRTP media channels, or they may be inactive at any given time, or may only be added due to in-game actions.

U-C 4: Non-realtime file transfers between people chatting. Note that this may involve a large number of files to transfer sequentially or in parallel, such as when sharing a folder of images or a directory of files.

U-C 5: Realtime text chat during an audio and/or video call with an individual or with multiple people in a conference.

U-C 6: Renegotiation of the configuration of the PeerConnection.

U-C 7: Proxy browsing, where a browser uses data channels of a PeerConnection to send and receive HTTP/HTTPS requests and data, for example to avoid local Internet filtering or monitoring.

4. Requirements

This section lists the requirements for P2P data channels between two browsers.

Req. 1: Multiple simultaneous data channels MUST be supported. Note that there may be 0 or more SRTP media streams in parallel with the data channels in the same PeerConnection, and the number and state (active/inactive) of these SRTP media streams may change at any time.

Req. 2: Both reliable and unreliable data channels MUST be supported.

Req. 3: Data channels of a PeerConnection MUST be congestion controlled; either individually, as a class, or in conjunction with the SRTP media streams of the PeerConnection, to ensure that data channels don't cause congestion problems for these SRTP media streams, and that the WebRTC PeerConnection as a whole is fair with competing traffic such as TCP.

Req. 4: The application SHOULD be able to provide guidance as to the relative priority of each data channel relative to each other, and relative to the SRTP media streams. This will interact with the congestion control algorithms.

Req. 5: Data channels MUST be secured; allowing for confidentiality, integrity and source authentication. See [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch] for detailed info.

Req. 6: Data channels MUST provide message fragmentation support such that IP-layer fragmentation can be avoided no matter how large a message the JavaScript application passes to be sent. It also MUST ensure that large data channel transfers don't unduly delay traffic on other data channels.

Req. 7: The data channel transport protocol MUST NOT encode local IP addresses inside its protocol fields; doing so reveals potentially private information, and leads to failure if the address is depended upon.

Req. 8: The data channel transport protocol SHOULD support unbounded-length "messages" (i.e., a virtual socket stream) at the application layer, for such things as image-file-transfer; Implementations might enforce a reasonable message size limit.

Req. 9: The data channel transport protocol SHOULD avoid IP fragmentation. It MUST support PMTU (Path MTU) discovery and MUST NOT rely on ICMP or ICMPv6 being generated or being passed back, especially for PMTU discovery.

Req. 10: It MUST be possible to implement the protocol stack in the user application space.

5. SCTP over DTLS over UDP Considerations

The important features of SCTP in the WebRTC context are:

- Usage of a TCP-friendly congestion control.

- The congestion control is modifiable for integration with the SRTP media stream congestion control.

- Support of multiple unidirectional streams, each providing its own notion of ordered message delivery.

- Support of ordered and out-of-order message delivery.

- Supporting arbitrary large user messages by providing fragmentation and reassembly.

- Support of PMTU-discovery.

- Support of reliable or partially reliable message transport.

SCTP multihoming will not be used in WebRTC. The SCTP layer will simply act as if it were running on a single-homed host, since that is the abstraction that the lower layer (a connection oriented, unreliable datagram service) exposes.

The encapsulation of SCTP over DTLS defined in [I-D.ietf-tsvwg-sctp-dtls-encaps] provides confidentiality, source authenticated, and integrity protected transfers. Using DTLS over UDP in combination with ICE enables middlebox traversal in IPv4 and IPv6 based networks. SCTP as specified in [RFC4960] MUST be used in combination with the extension defined in [RFC3758] and provides the following features for transporting non-SRTP media data between browsers:

o Support of multiple unidirectional streams.

o Ordered and unordered delivery of user messages.

o Reliable and partial-reliable transport of user messages.

Each SCTP user message contains a Payload Protocol Identifier (PPID) that is passed to SCTP by its upper layer on the sending side and provided to its upper layer on the receiving side. The PPID can be used to multiplex/demultiplex multiple upper layers over a single SCTP association. In the WebRTP context, the PPID is used to distinguish between UTF-8 encoded user data, binary encoded userdata and the Data Channel Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol]. Please note that the PPID is not accessible via the Javascript API.

The encapsulation of SCTP over DTLS, together with the SCTP features listed above satisfies all the requirements listed in Section 4.

The layering of protocols for WebRTC is shown in the following Figure 2.


```
                        +------+------+------+
                        | DCEP | UTF-8|Binary|
                        |      | data | data |
                        +------+------+------+
                        |        SCTP        |
          +----------------------------------+
          | STUN | SRTP |        DTLS        |
          +----------------------------------+
```

```
|          ICE                        |
+-------------------------------------+
| UDP1 | UDP2 | ...                   |
+-------------------------------------+
```

Figure 2: WebRTC protocol layers

This stack (especially in contrast to DTLS over SCTP [RFC6083] in combination with SCTP over UDP [RFC6951]) has been chosen because it

o supports the transmission of arbitrary large user messages.

o shares the DTLS connection with the SRTP media channels of the PeerConnection.

o provides privacy for the SCTP control information.

Considering the protocol stack of Figure 2 the usage of DTLS over UDP is specified in [RFC6347], while the usage of SCTP on top of DTLS is specified in [I-D.ietf-tsvwg-sctp-dtls-encaps]. Please note that the demultiplexing STUN vs. SRTP vs. DTLS is done as described in Section 5.1.2 of [RFC5764] and SCTP is the only payload of DTLS.

Since DTLS is typically implemented in user-land, the SCTP stack also needs to be a user-land stack.

When using DTLS as the lower layer, only single homed SCTP associations are supported, since DTLS does not expose any address management to its upper layer. The ICE/UDP layer can handle IP address changes during a session without needing interaction with the DTLS and SCTP layers. However, SCTP SHOULD be notified when an address changes has happened. In this case SCTP SHOULD retest the Path MTU and reset the congestion state to the initial state. In case of a window based congestion control like the one specified in [RFC4960], this means setting the congestion window and slow start threshold to its initial values.

Incoming ICMP or ICMPv6 messages can't be processed by the SCTP layer, since there is no way to identify the corresponding association. Therefore SCTP MUST support performing Path MTU

discovery without relying on ICMP or ICMPv6 as specified in [RFC4821] using probing messages specified in [RFC4820]. The initial Path MTU at the IP layer SHOULD NOT exceed 1200 bytes for IPv4 and 1280 for IPv6.

In general, the lower layer interface of an SCTP implementation SHOULD be adapted to address the differences between IPv4 and IPv6 (being connection-less) or DTLS (being connection-oriented).

When the protocol stack of Figure 2 is used, DTLS protects the complete SCTP packet, so it provides confidentiality, integrity and source authentication of the complete SCTP packet.

This SCTP stack and its upper layer MUST support the usage of multiple SCTP streams. A user message can be sent ordered or unordered and with partial or full reliability. The partial reliability extension MUST support policies to limit

- o the transmission and retransmission by time.
- o the number of retransmissions.

Limiting the number of retransmissions to zero combined with unordered delivery provides a UDP-like service where each user message is sent exactly once and delivered in the order received.

SCTP provides congestion control on a per-association base. This means that all SCTP streams within a single SCTP association share the same congestion window. Traffic not being sent over SCTP is not covered by the SCTP congestion control. Using a congestion control different from than the standard one might improve the impact on the parallel SRTP media streams.

6. The Usage of SCTP for Data Channels

6.1. SCTP Protocol Considerations

The DTLS encapsulation of SCTP packets as described in [I-D.ietf-tsvwg-sctp-dtls-encaps] MUST be used.

The following SCTP protocol extensions are required:

- o The stream reset extension defined in [RFC6525] MUST be supported. It is used for closing channels.
- o The dynamic address reconfiguration extension defined in [RFC5061] MUST be used to signal the support of the stream reset extension

defined in [RFC6525], other features of [RFC5061] are not REQUIRED to be implemented.

- The partial reliability extension defined in [RFC3758] MUST be supported. In addition to the timed reliability PR-SCTP policy defined in [RFC3758], the limited retransmission policy defined in [I-D.ietf-tsvwg-sctp-prpolicies] MUST be supported.

The support for message interleaving as defined in [I-D.ietf-tsvwg-sctp-ndata] SHOULD be used.

6.2. Association Setup

The SCTP association will be set up when the two endpoints of the WebRTC PeerConnection agree on opening it, as negotiated by JSEP (typically an exchange of SDP) [I-D.ietf-rtcweb-jsep]. It will use the DTLS connection selected via ICE; typically this will be shared via BUNDLE or equivalent with DTLS connections used to key the SRTP media streams.

The number of streams negotiated during SCTP association setup SHOULD be 65535, which is the maximum number of streams that can negotiated during the association setup.

6.3. SCTP Streams

SCTP defines a stream as a unidirectional logical channel existing within an SCTP association to another SCTP endpoint. The streams are used to provide the notion of in-sequence delivery and for multiplexing. Each user message is sent on a particular stream, either ordered or unordered. Ordering is preserved only for ordered messages sent on the same stream.

6.4. Channel Definition

The W3C has consensus on defining the application API for WebRTC DataChannels to be bidirectional. They also consider the notions of in-sequence, out-of-sequence, reliable and unreliable as properties of Channels. One strong wish is for the application-level API to be close to the API for WebSockets, which implies bidirectional streams of data and waiting for onopen to fire before sending, a textual label used to identify the meaning of the stream, among other things.

Each data channel also has a priority, which is an 2 byte unsigned integer value. These priorities MUST be interpreted as weighted-fair-queuing scheduling priorities per the definition of the corresponding stream scheduler supporting interleaving in [I-D.ietf-tsvwg-sctp-ndata]. For use in WebRTC, the values used

SHOULD be one of 128 ("below normal"), 256 ("normal"), 512 ("high") or 1024 ("extra high").

The realization of a bidirectional Data Channel is a pair of one incoming stream and one outgoing SCTP stream having the same stream SCTP identifier.

How stream values are selected is protocol and implementation dependent.

6.5. Opening a Channel

Data channels can be opened by using negotiation within the SCTP association, called in-band negotiation, or out-of-band negotiation. Out-of-band negotiation is defined as any method which results in an agreement as to the parameters of a channel and the creation thereof. The details are out of scope of this document.

A simple protocol for in-band negotiation is specified in [I-D.ietf-rtcweb-data-protocol].

When one side wants to open a channel using out-of-band negotiation, it picks a stream. Unless otherwise defined or negotiated, the streams are picked based on the DTLS role (the client picks even stream identifiers, the server odd stream identifiers). However, the application is responsible for avoiding collisions with existing streams. If it attempts to re-use a stream which is part of an existing Channel, the addition SHOULD fail. In addition to choosing a stream, the application SHOULD also determine the options to use for sending messages. The application MUST ensure in an application-specific manner that the application at the peer will also know the selected stream to be used, and the options for sending data from that side.

6.6. Transferring User Data on a Channel

All data sent on a Channel in both directions MUST be sent over the underlying stream using the reliability defined when the Channel was opened unless the options are changed, or per-message options are specified by a higher level.

No more than one message should be put into an SCTP user message.

The SCTP Payload Protocol Identifiers (PPIDs) are used to signal the interpretation of the "Payload data". For identifying a JavaScript string encoded in UTF-8 the PPID "WebRTC String" MUST be used, for JavaScript binary data (ArrayBuffer or Blob) the PPID "WebRTC Binary" MUST be used (see Section 8).

The usage of the PPIDs "WebRTC String Partial" and "WebRTC Binary Partial" is deprecated. They were used for a PPID-based fragmentation and reassembly of user messages belonging to reliable and ordered data channels.

If a message with an unsupported PPID is received or some error is detected by the receiver (for example, illegal ordering), the receiver SHOULD close the corresponding channel.

The SCTP base protocol specified in [RFC4960] does not support the interleaving of user messages. Therefore sending a large user message can monopolize the SCTP association. To overcome this limitation, [I-D.ietf-tsvwg-sctp-ndata] defines an extension to support message interleaving, which SHOULD be used. As long as message interleaving is not supported, the sender SHOULD limit the maximum message size to 16 KB to avoid monopolization.

It is recommended that the message size be kept within certain size bounds as applications will not be able to support arbitrarily-large single messages. This limit has to be negotiated, for example by using [I-D.ietf-mmusic-sctp-sdp].

The sender SHOULD disable the Nagle algorithm to minimize the latency.

6.7. Closing a Channel

Closing of a Data Channel MUST be signaled by resetting the corresponding outgoing streams [RFC6525]. This means that if one side decides to close the channel, it resets the corresponding outgoing stream. When the peer sees that an incoming stream was reset, it also resets its corresponding outgoing stream. Once this is completed, the channel is closed. Resetting a stream sets the Stream Sequence Numbers (SSNs) of the stream back to 'zero' with a corresponding notification to the application layer that the reset has been performed. Streams are available to reuse after a reset has been performed.

[RFC6525] also guarantees that all the messages are delivered (or abandoned) before resetting the stream.

7. Security Considerations

This document does not add any additional considerations to the ones given in [I-D.ietf-rtcweb-security] and [I-D.ietf-rtcweb-security-arch].

8. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

This document uses four already registered SCTP Payload Protocol Identifiers (PPIDs): "DOMString Last", "Binary Data Partial", "Binary Data Last", and "DOMString Partial". [RFC4960] creates the registry "SCTP Payload Protocol Identifiers" from which these identifiers were assigned. IANA is requested to update the reference of these four assignments to point to this document and change the names of the PPIDs. Therefore these four assignments should be updated to read:

Value	SCTP PPID	Reference
WebRTC String	51	[RFCXXXX]
WebRTC Binary Partial (Deprecated)	52	[RFCXXXX]
WebRTC Binary	53	[RFCXXXX]
WebRTC String Partial (Deprecated)	54	[RFCXXXX]

9. Acknowledgments

Many thanks for comments, ideas, and text from Harald Alvestrand, Adam Bergkvist, Christer Holmberg, Cullen Jennings, Paul Kyzivat, Eric Rescorla, Irene Ruengeler, Randall Stewart, Justin Uberti, and Magnus Westerlund.

10. References

10.1. Normative References

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

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

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

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, February 2012.

[I-D.ietf-tsvwg-sctp-ndata]
Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "A New Data Chunk for Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-ndata-00 (work in progress), February 2014.

[I-D.ietf-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Establishment Protocol", draft-ietf-rtcweb-data-protocol-05 (work in progress), May 2014.

[I-D.ietf-tsvwg-sctp-dtls-encaps]
Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "DTLS Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-dtls-encaps-04 (work in progress), May 2014.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-06 (work in progress), January 2014.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-09 (work in progress), February 2014.

[I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-06 (work in progress), February 2014.

[I-D.ietf-tsvwg-sctp-prpolicies]
Tuexen, M., Seggelmann, R., Stewart, R., and S. Loreto, "Additional Policies for the Partial Reliability Extension of the Stream Control Transmission Protocol", draft-ietf-tsvwg-sctp-prpolicies-03 (work in progress), May 2014.

[I-D.ietf-mmusic-sctp-sdp]
Loreto, S. and G. Camarillo, "Stream Control Transmission Protocol (SCTP)-Based Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-06 (work in progress), February 2014.

10.2. Informative References

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013.

[I-D.ietf-rtcweb-use-cases-and-requirements]
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-14 (work in progress), February 2014.

Authors' Addresses

Randell Jesup
Mozilla
US

Email: randell-ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420

FI

Email: salvatore.loreto@ericsson.com

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Steinfurt 48565
DE

Email: tuexen@fh-muenster.de