

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2013

R. Jesup
Mozilla
S. Loreto
Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences
October 23, 2012

RTCWeb Datagram Connection
draft-ietf-rtcweb-data-channel-02.txt

Abstract

The Web Real-Time Communication (WebRTC) 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 describes the non-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 Browser 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 April 26, 2013.

Copyright Notice

Copyright (c) 2012 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

Internet-Draft

RTCWeb Datagram Connection

October 2012

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 | 3 |
| 2. | Conventions | 3 |
| 3. | Requirements | 3 |
| 4. | Use Cases | 5 |
| 4.1. | Use Cases for Unreliable Datagram Based Channels | 5 |
| 4.2. | Use Cases for Reliable Channels (Datagram or Stream Based) | 5 |
| 5. | SCTP over DTLS over UDP Considerations | 6 |
| 6. | The Usage of SCTP in the RTCWeb Context | 8 |
| 6.1. | Association Setup | 9 |
| 6.2. | SCTP Streams | 9 |
| 6.3. | Channel Definition | 9 |
| 6.4. | Usage of Payload Protocol Identifier | 10 |
| 7. | Security Considerations | 10 |
| 8. | IANA Considerations | 10 |
| 9. | Acknowledgments | 10 |
| 10. | Informative References | 11 |
| | Authors' Addresses | 12 |

1. Introduction

Non-media data types in the context of RTCWEB are handled by using SCTP [[RFC4960](#)] encapsulated in DTLS [[RFC6347](#)].

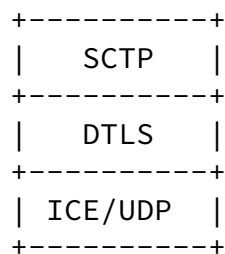


Figure 1: Basic stack diagram

The encapsulation of SCTP over DTLS over ICE/UDP provides a NAT traversal solution together with confidentiality, source authenticated, integrity protected transfers. This data transport service operates in parallel to the media transports, and all of them can eventually share a single transport-layer port number.

SCTP as specified in [[RFC4960](#)] with the extension defined in [[RFC3758](#)] provides multiple streams natively with reliable, and partially-reliable delivery modes.

The remainder of this document is organized as follows: [Section 3](#) and [Section 4](#) provide requirements and use cases for both unreliable and reliable peer to peer datagram base channel; [Section 5](#) arguments SCTP over DTLS over UDP; [Section 6](#) provides an overview of how SCTP should be used by the RTCWeb protocol framework for transporting non-media data between 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.](#) Requirements

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

Jesup, et al.

Expires April 26, 2013

[Page 3]

Internet-Draft

RTCWeb Datagram Connection

October 2012

- Req. 1 Multiple simultaneous datagram streams MUST be supported. Note that there may 0 or more media streams in parallel with the data streams, and the number and state (active/inactive) of the media streams may change at any time.
- Req. 2 Both reliable and unreliable datagram streams MUST be supported.
- Req. 3 Data streams MUST be congestion controlled; either individually, as a class, or in conjunction with the media streams, to ensure that datagram exchanges don't cause congestion problems for the media streams, and that the rtcweb PeerConnection as a whole is fair with competing streams such as TCP.
- Req. 4 The application SHOULD be able to provide guidance as to the relative priority of each datagram stream relative to each other, and relative to the media streams. [TBD: how this is encoded and what the impact of this is.] This will interact with the congestion control algorithms.
- Req. 5 Datagram streams MUST be encrypted; 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 Consent and NAT traversal mechanism: These are handled by the PeerConnection's ICE [[RFC5245](#)] connectivity checks and optional TURN servers.

- Req. 7 Data streams MUST provide message fragmentation support such that IP-layer fragmentation does not occur no matter how large a message the Javascript application passes to be sent.
- Req. 8 The data stream 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. 9 The data stream protocol SHOULD support unbounded-length "messages" (i.e., a virtual socket stream) at the application layer, for such things as image-file-transfer; or it MUST support a maximum application-layer message size of at least 2GB.

- Req. 10 The data stream packet format/encoding MUST be such that it is impossible for a malicious Javascript to generate an application message crafted such that it could be interpreted as a native protocol over UDP - such as UPnP, RTP, SNMP, STUN, etc.
- Req. 11 The data stream transport protocol MUST start with the assumption of a PMTU of 1280 [*** need justification ***] bytes until measured otherwise.
- Req. 12 The data stream transport protocol MUST NOT rely on ICMP or ICMPv6 being generated or being passed back, such as for PMTU discovery.
- Req. 13 It MUST be possible to implement the protocol stack in the user application space.

[4.](#) Use Cases

[4.1.](#) Use Cases for Unreliable Datagram Based 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 media channels, or all media channels may be inactive, and that there may also be reliable data channels in use.

U-C 2 Non-critical state updates about a user in a video chat or conference, such as Mute state.

[4.2.](#) Use Cases for Reliable Channels (Datagram or Stream Based)

Note that either reliable datagrams or streams are possible; reliable streams would be fairly simple to layer on top of SCTP reliable datagrams with in-order delivery.

U-C 3 A real-time game where critical state information needs to be transferred, such as control information. Typically this would be datagrams. Such a game may have no 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. This could be datagrams or streaming. 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 while talking with an individual or with multiple people in a conference. Typically this would be datagrams.

U-C 6 Renegotiation of the set of media streams in the PeerConnection. Typically this would be datagrams.

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. Typically this would be streams.

[5.](#) SCTP over DTLS over UDP Considerations

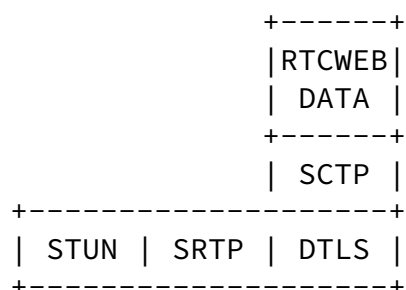
The encapsulation of SCTP over DTLS as defined in [[I-D.tuexen-tsvwg-sctp-dtls-encaps](#)] provides a NAT traversal solution together with confidentiality, source authenticated, integrity protected transfers. SCTP as specified in [[RFC4960](#)] MUST be used in combination with the extension defined in [[RFC3758](#)] and provides the following interesting features for transporting non-media data between browsers:

- o Support of multiple streams.
- o Ordered and unordered delivery of user messages.
- o Reliable and partial-reliable transport of user messages.

Each SCTP user message contains a so called Payload Protocol Identifier (PPID) that is passed to SCTP by its upper layer and sent to its peer. This value represents an application (or upper layer) specified protocol identifier and be used to transport multiple protocols over a single SCTP association. The sender provides for each protocol a specific PPID and the receiver MAY demultiplex the messages based on the received PPID.

The encapsulation of SCTP over DTLS, together with the SCTP features listed above satisfies all the requirements listed in [Section 3](#).

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



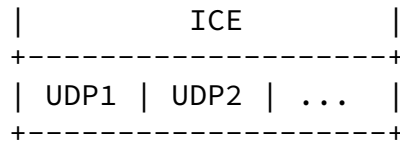


Figure 2: WebRTC protocol layers

This stack (especially in contrast to DTLS over SCTP [[RFC6083](#)]) has been chosen because it

- o supports the transmission of arbitrary large user messages.
- o shares the DTLS connection with the media channels.
- 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.tuexen-tsvwg-sctp-dtls-encaps](#)].

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 SHOULD be used, 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 to notify the DTLS and SCTP layers, though it would be advantageous to retest path MTU on an IP address change.

DTLS implementations used for this stack SHOULD support controlling fields of the IP layer like the Don't fragment (DF)-bit in case of IPv4 and the Differentiated Services Code Point (DSCP) field required for supporting [[I-D.ietf-rtcweb-qos](#)]. Being able to set the (DF)-bit in case of IPv4 is required for performing path MTU discovery. The DTLS implementation SHOULD also support sending user messages exceeding the path MTU.

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

When 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 protocol stack 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. Due to the typical parallel SRTP media streams, a delay-sensitive congestion control algorithm MUST be supported and the congestion control MAY be coordinated between the data channels and the media streams to avoid a data channel transfer ending up with most or all the channel bandwidth.

Since SCTP does not support the negotiation of a congestion control algorithm, the algorithm either MUST be negotiated before establishment of the SCTP association or MUST not require any negotiation because it only requires sender side behavior using existing information carried in the association.

[6.](#) The Usage of SCTP in the RTCWeb Context

The important features of SCTP in the RTCWeb context are:

- o TCP-friendly congestion control.
- o The congestion control is modifiable for integration with media stream congestion control.

- o Support for multiple channels with different characteristics.
- o Support for out-of-order delivery.
- o Support for large datagrams and PMTU-discovery and fragmentation.
- o Reliable or partial reliability support.
- o Support of multiple streams.

SCTP multihoming will not be used in RTCWeb. 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.

[6.1.](#) 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](#)]. Additionally, the negotiation SHOULD include some type of congestion control selection. It will use the DTLS connection selected via SDP; typically this will be shared via BUNDLE with DTLS connections used to key the DTLS-SRTP media streams.

The application SHOULD indicate the initial number of streams required when opening the association, and if no value is supplied, the implementation SHOULD provide a default, with a suggested value of 16. If more simultaneous streams are needed, [[RFC6525](#)] allows adding additional (but not removing) streams to an existing association. Note there can be up to 65536 SCTP streams per SCTP association in each direction.

[6.2.](#) SCTP Streams

SCTP defines a stream as an unidirectional logical channel existing within an SCTP association one 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 order or unordered. Ordering is preserved only for all ordered messages sent on the same stream.

[6.3.](#) 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 un-reliable 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.

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

The simple protocol specified in [[I-D.jesup-rtcweb-data-protocol](#)] MUST be used to set up and manage the bidirectional data channels.

Note that there's no requirement for the SCTP streams used to create a bidirectional channel have the same number in each direction. How stream values are selected is protocol and implementation dependent.

Closing of a Data Channel MUST be signalled by resetting the corresponding streams [[RFC6525](#)]. Resetting a stream set 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. Closed streams are available to reuse.

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

[6.4.](#) Usage of Payload Protocol Identifier

The SCTP Payload Protocol Identifiers (PPIDs) MUST used to signal the interpretation of the "Payload data", like the protocol specified in [[I-D.jesup-rtcweb-data-protocol](#)] uses them to identify a Javascript string, a Javascript array or a a Javascript blob.

[7.](#) Security Considerations

To be done.

[8.](#) IANA Considerations

This document does not require any actions by the IANA.

9. Acknowledgments

Many thanks for comments, ideas, and text from Harald Alvestrand, Adam Bergkvist, Cullen Jennings, Eric Rescorla, Randall Stewart, and Justin Uberti.

Jesup, et al.

Expires April 26, 2013

[Page 10]

Internet-Draft

RTCWeb Datagram Connection

October 2012

10. Informative 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.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", [RFC 4960](#), 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.
- [RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", [RFC 6083](#), January 2011.
- [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-rtcweb-security] Rescorla, E., "Security Considerations for RTC-Web", [draft-ietf-rtcweb-security-03](#) (work in progress), June 2012.

[I-D.ietf-rtcweb-security-arch]

Rescorla, E., "RTCWEB Security Architecture",
[draft-ietf-rtcweb-security-arch-05](#) (work in progress),
October 2012.

[I-D.ietf-rtcweb-jsep]

Uberti, J. and C. Jennings, "Javascript Session
Establishment Protocol", [draft-ietf-rtcweb-jsep-01](#) (work
in progress), June 2012.

[I-D.ietf-rtcweb-qos]

Dhesikan, S., Druta, D., Jones, P., and J. Polk, "DSCP and
other packet markings for RTCWeb QoS",
[draft-ietf-rtcweb-qos-00](#) (work in progress), October 2012.

Jesup, et al.

Expires April 26, 2013

[Page 11]

Internet-Draft

RTCWeb Datagram Connection

October 2012

[I-D.ietf-tsvwg-sctp-udp-encaps]

Tuexen, M. and R. Stewart, "UDP Encapsulation of SCTP
Packets", [draft-ietf-tsvwg-sctp-udp-encaps-06](#) (work in
progress), October 2012.

[I-D.jesup-rtcweb-data-protocol]

Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel
Protocol", [draft-jesup-rtcweb-data-protocol-03](#) (work in
progress), September 2012.

[I-D.tuexen-tsvwg-sctp-dtls-encaps]

Jesup, R., Loreto, S., Stewart, R., and M. Tuexen, "DTLS
Encapsulation of SCTP Packets for RTCWEB",
[draft-tuexen-tsvwg-sctp-dtls-encaps-01](#) (work in progress),
July 2012.

Authors' Addresses

Randell Jesup
Mozilla
USA

Email: randell-ietf@jesup.org

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com

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

Email: tuexen@fh-muenster.de