RTCWeb Working Group
Internet-Draft
Intended status: Informational
Expires: March 11, 2013

R. Jesup
Mozilla
S. Loreto
Ericsson
M. Tuexen
Muenster Univ. of Appl. Sciences

September 7, 2012

RTCWeb Datagram Connection
draft-ietf-rtcweb-data-channel-01.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 March 11, 2013.

Copyright Notice

Table of Contents

1. Introduction

The issue of how best to handle non-media data types in the context of RTCWEB has reached a general consensus on the usage of SCTP [RFC4960] encapsulated on DTLS [RFC6347]:

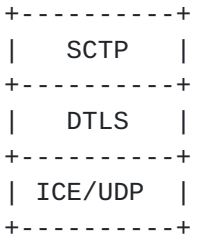

Figure 1: Basic stack diagram

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 provides multiple streams natively with reliable, unreliable and partially-reliable delivery modes.

The remainder of this document is organized as follows: Section 2 and Section 3 provide requirements and use cases for both unreliable and reliable peer to peer datagram base channel; Section 4 arguments SCTP over DTLS over UDP; Section 5 provides an overview of how SCTP should be used by the RTCWeb protocol framework for transporting non-media data between browsers.

2. Requirements

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

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 else it must support at least a maximum application-layer message size of 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.

3. Use Cases

3.1. 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.

3.2. 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.

4. Datagrams over SCTP over DTLS over UDP

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 provides also natively several interesting features for transporting non-media data between browsers:

- Support of multiple streams.
- Ordered and unordered delivery of user messages.
- 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 demultiplexes 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 in Section 2.

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

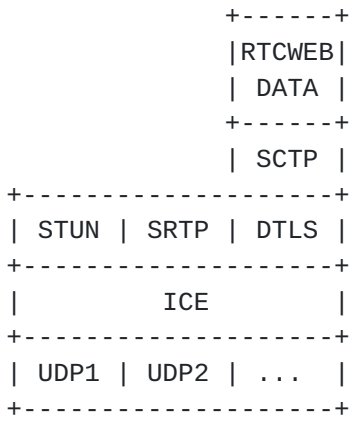

Figure 2: WebRTC protocol layers

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

- supports the transmission of arbitrary large user messages.
- shares the DTLS connection with the media channels.
- 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, an SCTP user-land implementation must also be used.

When using DTLS as the lower layer, only single homed SCTP associations can 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 must 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. This is required for performing path MTU discovery. The DTLS implementation must also support sending user messages exceeding the path MTU.

When supporting multiple SCTP associations over a single DTLS connection, incoming ICMP or ICMPv6 messages can't be processed by the SCTP layer, since there is no way to identify the corresponding association. Therefore the number of SCTP associations should be limited to one or ICMP and ICMPv6 messages should be ignored. In general, the lower layer interface of an SCTP implementation has to 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 supports the usage of multiple SCTP streams. A user message can be sent ordered or unordered and, if the SCTP implementations support [RFC3758], with partial reliability. When using partial reliability, it might make sense to use a policy limiting the number of retransmissions by time or number. Limiting

the number of retransmissions to zero provides a UDP-like service where each user message is sent exactly once.

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, it will be advantageous to select a delay-sensitive congestion control algorithm or to at least coordinate congestion control 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 have an internal negotiaton mechanism for selecting a congestion control algorithm, the algorithm should be negotiated before establishment of the SCTP associaton.

5. The Envisioned Usage of SCTP in the RTCWeb Context

The appealing features of SCTP in the RTCWeb context are:

- TCP-friendly congestion control.
- The congestion control is modifiable for integration with media stream congestion control.
- Support for multiple channels with different characteristics.
- Support for out-of-order delivery.
- Support for large datagrams and PMTU-discovery and fragmentation.
- Reliable or partial reliability support.
- Support of multiple streams.

Multihoming will not be used in this scenario. The SCTP layer would 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) would expose.

5.1. Association Setup

The SCTP association would 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 would use the DTLS connection created at the start of the PeerConnection connection.

The application should indicate the number of simultaneous 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. There can be up to 65535 SCTP streams per SCTP association in each direction.

5.2. 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. 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.

5.3. 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.

A possible realization of a bidirectional Data Channel is a pair of one incoming stream and one outcoming SCTP stream.

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 and used to provide this functionality is up to the protocol.

Closing of a Data Channel can be signalled 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.

5.4. Usage of Payload Protocol Identifier

The SCTP Payload Protocol Identifiers (PPIDs) can be used to signal the interpretation of the "Payload data", like a string, ASCII or

binary data.

RFC 4960 [RFC4960] creates the registry from which these identifiers have been assigned. Eventual PPIDs defined within the RTCWeb Context have to be registered with IANA.

6. Minor Protocol and Message Format

A separate draft (draft-jesup-rtcweb-data-protocol) proposes the minor protocol to set up and manage the bidirectional data channels needed to satisify the requirements in this document for WebRTC.

Masking of the protocol is not needed if the lower layer always encrypts with DTLS.

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 Cullen Jennings, Eric Rescorla, Randall Stewart, Justin Uberti, Adam Bergkvist and Harald Alvestrand.

10. Informative References

[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-03 (work in progress), July 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-tsvwg-sctp-udp-encaps]
Tuexen, M. and R. Stewart, "UDP Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-udp-encaps-04 (work in progress), July 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