Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

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

July 15, 2013

RTCWeb Data Channels
draft-ietf-rtcweb-data-channel-05.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 specifies 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

Copyright Notice

Table of Contents

1. Introduction

Non-media data types in the context of RTCWeb 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 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] provides multiple streams natively with reliable, and partially-reliable delivery modes.

The remainder of this document is organized as follows: Section 4 and Section 3 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 specification 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. Use Cases

This section defined 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 media channels, or all 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 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 while talking with an individual or with multiple people in a conference.

U-C 6 Renegotiation of the set of media streams in 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 0 or more media streams in parallel with the data channels, and the number and state (active/inactive) of the media streams may change at any time.

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

Req. 3 Data channels MUST be congestion controlled; either individually, as a class, or in conjunction with the media streams, to ensure that data channels don't cause congestion problems for the media streams, and that the RTCWeb 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 media streams. [TBD: how this is encoded and what the impact of this is.] 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 unduely 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 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 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.

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.

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 NAT traversal in IPv4 based networks. 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 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 so called Payload Protocol Identifier (PPID) that is passed to SCTP by its upper layer and sent to its peer. This value can be used to multiplex multiple protocols over a single SCTP association. The sender provides for each protocol a specific PPID and the receiver can 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 4.

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

```
              +------+
              |RTCWEB|
              | 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 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.ietf-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 MUST 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 as specified in [RFC4821] using probing messages specified in [RFC4820]. The initial Path MTU MUST NOT exceed 1280 [*** need justification ***] bytes until measured otherwise.

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

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:

- The stream reset extension defined in [RFC6525] MUST be supported. It is used for closing channels.

- 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] MUST NOT be used.

- 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.tuexen-tsvwg-sctp-prpolicies] MUST be supported.

- The message interleaving extension defined in [I-D.stewart-tsvwg-sctp-ndata] MUST be supported.

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]. 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 or equivalent 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 an appropriate default. 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.

6.3. 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 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 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 signaled 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. Streams are available to reuse after a reset has been performed.

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

6.5. Usage of Payload Protocol Identifier

The SCTP Payload Protocol Identifiers (PPIDs) can be 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 binary data (ArrayBuffer or Blob) and to provide fragmentation support for large messages that may cause the message to monopolize the SCTP association.

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

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, 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.stewart-tsvwg-sctp-ndata] Stewart, R., Tuexen, M., and S. Loreto, "A New Data Chunk for Stream Control Transmission Protocol", draft-stewart-tsvwg-sctp-ndata-01 (work in progress), February 2013.

[I-D.jesup-rtcweb-data-protocol] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Protocol", draft-jesup-rtcweb-data-protocol-04 (work in progress), February 2013.

[I-D.ietf-tsvwg-sctp-dtls-encaps] Jesup, R., Loreto, S., Stewart, R., and M. Tuexen, "DTLS Encapsulation of SCTP Packets for RTCWEB", draft-ietf-tsvwg-sctp-dtls-encaps-00 (work in progress), February 2013.

[I-D.ietf-rtcweb-security] Rescorla, E., "Security Considerations for RTC-Web", draft-ietf-rtcweb-security-04 (work in progress), January 2013.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "RTCWEB Security Architecture", draft-ietf-rtcweb-security-arch-06 (work in progress), January 2013.

[I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-03 (work in progress), February 2013.

[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.

10.2. Informative References

[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-11 (work in progress), June 2013.

[I-D.tuexen-tsvwg-sctp-prpolicies]
Loreto, S., Seggelmann, R., Stewart, R., and M. Tuexen, "Additional Policies for the Partial Delivery Extension of the Stream Control Transmission Protocol", draft-tuexen-tsvwg-sctp-prpolicies-02 (work in progress), July 2013.

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