Stream: Internet Engineering Task Force (IETF)
RFC: 8835
Category: Standards Track
Published: June 2020
ISSN: 2070-1721
Author: H. Alvestrand
Google

RFC 8835
Transports for WebRTC

Abstract

This document describes the data transport protocols used by Web Real-Time Communication (WebRTC), including the protocols used for interaction with intermediate boxes such as firewalls, relays, and NAT boxes.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8835.

Copyright Notice

Table of Contents

1. Introduction

WebRTC is a protocol suite aimed at real-time multimedia exchange between browsers, and between browsers and other entities.

WebRTC is described in the WebRTC overview document [RFC8825], which also defines terminology used in this document, including the terms "WebRTC endpoint" and "WebRTC browser".

Terminology for RTP sources is taken from [RFC7656].

This document focuses on the data transport protocols that are used by conforming implementations, including the protocols used for interaction with intermediate boxes such as firewalls, relays, and NAT boxes.

This protocol suite is intended to satisfy the security considerations described in the WebRTC security documents, [RFC8826] and [RFC8827].

This document describes requirements that apply to all WebRTC endpoints. When there are requirements that apply only to WebRTC browsers, this is called out explicitly.

2. Requirements Language

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Transport and Middlebox Specification

3.1. System-Provided Interfaces

The protocol specifications used here assume that the following protocols are available to the implementations of the WebRTC protocols:

UDP [RFC0768]: This is the protocol assumed by most protocol elements described.

TCP [RFC0793]: This is used for HTTP/WebSockets, as well as TURN/TLS and ICE-TCP.

For both protocols, IPv4 and IPv6 support is assumed.

For UDP, this specification assumes the ability to set the Differentiated Services Code Point (DSCP) of the sockets opened on a per-packet basis, in order to achieve the prioritizations described in [RFC8837] (see Section 4.2 of this document) when multiple media types are multiplexed. It does not assume that the DSCPs will be honored and does assume that they may be zeroed or changed, since this is a local configuration issue.

Platforms that do not give access to these interfaces will not be able to support a conforming WebRTC endpoint.

This specification does not assume that the implementation will have access to ICMP or raw IP.

The following protocols may be used, but they can be implemented by a WebRTC endpoint and are therefore not defined as "system-provided interfaces":

TURN: Traversal Using Relays Around NAT [RFC5766]

STUN: Session Traversal Utilities for NAT [RFC5389]

ICE: Interactive Connectivity Establishment [RFC8445]

TLS: Transport Layer Security [RFC8446]

DTLS: Datagram Transport Layer Security [RFC6347]

3.2. Ability to Use IPv4 and IPv6

Web applications running in a WebRTC browser **MUST** be able to utilize both IPv4 and IPv6 where available -- that is, when two peers have only IPv4 connectivity to each other, or they have only IPv6 connectivity to each other, applications running in the WebRTC browser **MUST** be able to communicate.

When TURN is used, and the TURN server has IPv4 or IPv6 connectivity to the peer or the peer's TURN server, candidates of the appropriate types **MUST** be supported. The "Happy Eyeballs" specification for ICE [RFC8421] **SHOULD** be supported.

3.3. Usage of Temporary IPv6 Addresses

The IPv6 default address selection specification [RFC6724] specifies that temporary addresses [RFC4941] are to be preferred over permanent addresses. This is a change from the rules specified by [RFC3484]. For applications that select a single address, this is usually done by the IPV6_PREFER_SRC_TMP preference flag specified in [RFC5014]. However, this rule, which is intended to ensure that privacy-enhanced addresses are used in preference to static addresses, doesn't have the right effect in ICE, where all addresses are gathered and therefore revealed to the application. Therefore, the following rule is applied instead:

> When a WebRTC endpoint gathers all IPv6 addresses on its host, and both nondeprecated temporary addresses and permanent addresses of the same scope are present, the WebRTC endpoint **SHOULD** discard the permanent addresses before exposing addresses to the application or using them in ICE. This is consistent with the default policy described in [RFC6724].
>
> If some, but not all, of the temporary IPv6 addresses are marked deprecated, the WebRTC endpoint **SHOULD** discard the deprecated addresses, unless they are used by an ongoing connection. In an ICE restart, deprecated addresses that are currently in use **MAY** be retained.

3.4. Middlebox-Related Functions

The primary mechanism for dealing with middleboxes is ICE, which is an appropriate way to deal with NAT boxes and firewalls that accept traffic from the inside, but only from the outside if it is in response to inside traffic (simple stateful firewalls).

ICE [RFC8445] **MUST** be supported. The implementation **MUST** be a full ICE implementation, not ICE-Lite. A full ICE implementation allows interworking with both ICE and ICE-Lite implementations when they are deployed appropriately.

In order to deal with situations where both parties are behind NATs of the type that perform endpoint-dependent mapping (as defined in [RFC5128], Section 2.4), TURN [RFC5766] **MUST** be supported.

WebRTC browsers **MUST** support configuration of STUN and TURN servers, from both browser configuration and an application.

Note that other work exists around STUN and TURN server discovery and management, including [RFC8155] for server discovery, as well as [RETURN].

In order to deal with firewalls that block all UDP traffic, the mode of TURN that uses TCP between the WebRTC endpoint and the TURN server **MUST** be supported, and the mode of TURN that uses TLS over TCP between the WebRTC endpoint and the TURN server **MUST** be supported. See Section 2.1 of [RFC5766], for details.

In order to deal with situations where one party is on an IPv4 network and the other party is on an IPv6 network, TURN extensions for IPv6 [RFC6156] **MUST** be supported.

TURN TCP candidates, where the connection from the WebRTC endpoint's TURN server to the peer is a TCP connection, [RFC6062] **MAY** be supported.

However, such candidates are not seen as providing any significant benefit, for the following reasons.

First, use of TURN TCP candidates would only be relevant in cases where both peers are required to use TCP to establish a connection.

Second, that use case is supported in a different way by both sides establishing UDP relay candidates using TURN over TCP to connect to their respective relay servers.

Third, using TCP between the WebRTC endpoint's TURN server and the peer may result in more performance problems than using UDP, e.g., due to head of line blocking.

ICE-TCP candidates [RFC6544] **MUST** be supported; this may allow applications to communicate to peers with public IP addresses across UDP-blocking firewalls without using a TURN server.

If TCP connections are used, RTP framing according to [RFC4571] **MUST** be used for all packets. This includes the RTP packets, DTLS packets used to carry data channels, and STUN connectivity check packets.

The ALTERNATE-SERVER mechanism specified in Section 11 of [RFC5389] (300 Try Alternate) **MUST** be supported.

The WebRTC endpoint **MAY** support accessing the Internet through an HTTP proxy. If it does so, it **MUST** include the "ALPN" header as specified in [RFC7639], and proxy authentication as described in Section 4.3.6 of [RFC7231] and [RFC7235] **MUST** also be supported.

3.5. Transport Protocols Implemented

For transport of media, secure RTP is used. The details of the RTP profile used are described in "Media Transport and Use of RTP in WebRTC" [RFC8834], which mandates the use of a circuit breaker [RFC8083] and congestion control (see [RFC8836] for further guidance).

Key exchange **MUST** be done using DTLS-SRTP, as described in [RFC8827].

For data transport over the WebRTC data channel [RFC8831], WebRTC endpoints **MUST** support SCTP over DTLS over ICE. This encapsulation is specified in [RFC8261]. Negotiation of this transport in the Session Description Protocol (SDP) is defined in [RFC8841]. The SCTP extension for I-DATA [RFC8260] **MUST** be supported.

The setup protocol for WebRTC data channels described in [RFC8832] **MUST** be supported.

> Note: The interaction between DTLS-SRTP as defined in [RFC5764] and ICE as defined in [RFC8445] is described in Section 6 of [RFC8842]. The effect of this specification is that all ICE candidate pairs associated with a single component are part of the same DTLS association. Thus, there will only be one DTLS handshake, even if there are multiple valid candidate pairs.

WebRTC endpoints **MUST** support multiplexing of DTLS and RTP over the same port pair, as described in the DTLS-SRTP specification [RFC5764], Section 5.1.2, with clarifications in [RFC7983]. All application-layer protocol payloads over this DTLS connection are SCTP packets.

Protocol identification **MUST** be supplied as part of the DTLS handshake, as specified in [RFC8833].

4. Media Prioritization

In the WebRTC prioritization model, the application tells the WebRTC endpoint about the priority of media and data that is controlled from the API.

In this context, a "flow" is used for the units that are given a specific priority through the WebRTC API.

For media, a "media flow", which can be an "audio flow" or a "video flow", is what [RFC7656] calls a "media source", which results in a "source RTP stream" and one or more "redundancy RTP streams". This specification does not describe prioritization between the RTP streams that come from a single media source.

All media flows in WebRTC are assumed to be interactive, as defined in [RFC4594]; there is no browser API support for indicating whether media is interactive or noninteractive.

A "data flow" is the outgoing data on a single WebRTC data channel.

The priority associated with a media flow or data flow is classified as "very-low", "low", "medium", or "high". There are only four priority levels in the API.

The priority settings affect two pieces of behavior: packet send sequence decisions and packet markings. Each is described in its own section below.

4.1. Local Prioritization

Local prioritization is applied at the local node, before the packet is sent. This means that the prioritization has full access to the data about the individual packets and can choose differing treatment based on the stream a packet belongs to.

When a WebRTC endpoint has packets to send on multiple streams that are congestion controlled under the same congestion control regime, the WebRTC endpoint **SHOULD** cause data to be emitted in such a way that each stream at each level of priority is being given approximately twice the transmission capacity (measured in payload bytes) of the level below.

Thus, when congestion occurs, a high-priority flow will have the ability to send 8 times as much data as a very-low-priority flow if both have data to send. This prioritization is independent of the media type. The details of which packet to send first are implementation defined.

For example, if there is a high-priority audio flow sending 100-byte packets and a low-priority video flow sending 1000-byte packets, and outgoing capacity exists for sending > 5000 payload bytes, it would be appropriate to send 4000 bytes (40 packets) of audio and 1000 bytes (one packet) of video as the result of a single pass of sending decisions.

Conversely, if the audio flow is marked low priority and the video flow is marked high priority, the scheduler may decide to send 2 video packets (2000 bytes) and 5 audio packets (500 bytes) when outgoing capacity exists for sending > 2500 payload bytes.

If there are two high-priority audio flows, each will be able to send 4000 bytes in the same period where a low-priority video flow is able to send 1000 bytes.

Two example implementation strategies are:

- When the available bandwidth is known from the congestion control algorithm, configure each codec and each data channel with a target send rate that is appropriate to its share of the available bandwidth.
- When congestion control indicates that a specified number of packets can be sent, send packets that are available to send using a weighted round-robin scheme across the connections.

Any combination of these, or other schemes that have the same effect, is valid, as long as the distribution of transmission capacity is approximately correct.

For media, it is usually inappropriate to use deep queues for sending; it is more useful to, for instance, skip intermediate frames that have no dependencies on them in order to achieve a lower bitrate. For reliable data, queues are useful.

Note that this specification doesn't dictate when disparate streams are to be "congestion controlled under the same congestion control regime". The issue of coupling congestion controllers is explored further in [RFC8699].

4.2. Usage of Quality of Service -- DSCP and Multiplexing

When the packet is sent, the network will make decisions about queueing and/or discarding the packet that can affect the quality of the communication. The sender can attempt to set the DSCP field of the packet to influence these decisions.

Implementations **SHOULD** attempt to set QoS on the packets sent, according to the guidelines in [RFC8837]. It is appropriate to depart from this recommendation when running on platforms where QoS marking is not implemented.

The implementation **MAY** turn off use of DSCP markings if it detects symptoms of unexpected behavior such as priority inversion or blocking of packets with certain DSCP markings. Some examples of such behaviors are described in [ANRW16]. The detection of these conditions is implementation dependent.

A particularly hard problem is when one media transport uses multiple DSCPs, where one may be blocked and another may be allowed. This is allowed even within a single media flow for video in [RFC8837]. Implementations need to diagnose this scenario; one possible implementation is to send initial ICE probes with DSCP 0, and send ICE probes on all the DSCPs that are intended to be used once a candidate pair has been selected. If one or more of the DSCP-marked probes fail, the sender will switch the media type to using DSCP 0. This can be carried out simultaneously with the initial media traffic; on failure, the initial data may need to be resent. This switch will, of course, invalidate any congestion information gathered up to that point.

Failures can also start happening during the lifetime of the call; this case is expected to be rarer and can be handled by the normal mechanisms for transport failure, which may involve an ICE restart.

Note that when a DSCP causes nondelivery, one has to switch the whole media flow to DSCP 0, since all traffic for a single media flow needs to be on the same queue for congestion control purposes. Other flows on the same transport, using different DSCPs, don't need to change.

All packets carrying data from the SCTP association supporting the data channels **MUST** use a single DSCP. The code point used **SHOULD** be that recommended by [RFC8837] for the highest-priority data channel carried. Note that this means that all data packets, no matter what their relative priority is, will be treated the same by the network.

All packets on one TCP connection, no matter what it carries, **MUST** use a single DSCP.

More advice on the use of DSCPs with RTP, as well as the relationship between DSCP and congestion control, is given in [RFC7657].

There exist a number of schemes for achieving quality of service that do not depend solely on DSCPs. Some of these schemes depend on classifying the traffic into flows based on 5-tuple (source address, source port, protocol, destination address, destination port) or 6-tuple (5-tuple + DSCP). Under differing conditions, it may therefore make sense for a sending application to choose any of the following configurations:

- Each media stream carried on its own 5-tuple
- Media streams grouped by media type into 5-tuples (such as carrying all audio on one 5-tuple)
- All media sent over a single 5-tuple, with or without differentiation into 6-tuples based on DSCPs

In each of the configurations mentioned, data channels may be carried in their own 5-tuple or multiplexed together with one of the media flows.

More complex configurations, such as sending a high-priority video stream on one 5-tuple and sending all other video streams multiplexed together over another 5-tuple, can also be envisioned. More information on mapping media flows to 5-tuples can be found in [RFC8834].

A sending implementation **MUST** be able to support the following configurations:

- Multiplex all media and data on a single 5-tuple (fully bundled)
- Send each media stream on its own 5-tuple and data on its own 5-tuple (fully unbundled)

The sending implementation **MAY** choose to support other configurations, such as bundling each media type (audio, video, or data) into its own 5-tuple (bundling by media type).

Sending data channel data over multiple 5-tuples is not supported.

A receiving implementation **MUST** be able to receive media and data in all these configurations.

5. IANA Considerations

This document has no IANA actions.

6. Security Considerations

WebRTC security considerations are enumerated in [RFC8826].

Security considerations pertaining to the use of DSCP are enumerated in [RFC8837].

7. References

7.1. Normative References

[RFC0768]

Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, DOI 10.17487/RFC4571, July 2006, <https://www.rfc-editor.org/info/rfc4571>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <https://www.rfc-editor.org/info/rfc4594>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <https://www.rfc-editor.org/info/rfc4941>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/info/rfc5389>.

[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, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <https://www.rfc-editor.org/info/rfc5766>.

[RFC6062] Perreault, S., Ed. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, DOI 10.17487/RFC6062, November 2010, <https://www.rfc-editor.org/info/rfc6062>.

[RFC6156] Camarillo, G., Novo, O., and S. Perreault, Ed., "Traversal Using Relays around NAT (TURN) Extension for IPv6", RFC 6156, DOI 10.17487/RFC6156, April 2011, <https://www.rfc-editor.org/info/rfc6156>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B. B., and A. B. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, DOI 10.17487/RFC6544, March 2012, <https://www.rfc-editor.org/info/rfc6544>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <https://www.rfc-editor.org/info/rfc6724>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <https://www.rfc-editor.org/info/rfc7235>.

[RFC7639] Hutton, A., Uberti, J., and M. Thomson, "The ALPN HTTP Header Field", RFC 7639, DOI 10.17487/RFC7639, August 2015, <https://www.rfc-editor.org/info/rfc7639>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <https://www.rfc-editor.org/info/rfc7656>.

[RFC7983] Petit-Huguenin, M. and G. Salgueiro, "Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)", RFC 7983, DOI 10.17487/RFC7983, September 2016, <https://www.rfc-editor.org/info/rfc7983>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https://www.rfc-editor.org/info/rfc8083>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8260] Stewart, R., Tuexen, M., Loreto, S., and R. Seggelmann, "Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol", RFC 8260, DOI 10.17487/RFC8260, November 2017, <https://www.rfc-editor.org/info/rfc8260>.

[RFC8261] Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "Datagram Transport Layer Security (DTLS) Encapsulation of SCTP Packets", RFC 8261, DOI 10.17487/RFC8261, November 2017, <https://www.rfc-editor.org/info/rfc8261>.

[RFC8421] Martinsen, P., Reddy, T., and P. Patil, "Guidelines for Multihomed and IPv4/IPv6 Dual-Stack Interactive Connectivity Establishment (ICE)", BCP 217, RFC 8421, DOI 10.17487/RFC8421, July 2018, <https://www.rfc-editor.org/info/rfc8421>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, June 2020, <https://www.rfc-editor.org/info/rfc8825>.

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/RFC8826, June 2020, <https://www.rfc-editor.org/info/rfc8826>.

[RFC8827] Rescorla, E., "WebRTC Security Architecture", RFC 8827, DOI 10.17487/RFC8827, June 2020, <https://www.rfc-editor.org/info/rfc8827>.

[RFC8831] Jesup, R., Loreto, S., and M. Tüxen, "WebRTC Data Channels", RFC 8831, DOI 10.17487/RFC8831, June 2020, <https://www.rfc-editor.org/info/rfc8831>.

[RFC8832] Jesup, R., Loreto, S., and M. Tüxen, "WebRTC Data Channel Establishment Protocol", RFC 8832, DOI 10.17487/RFC8832, June 2020, <https://www.rfc-editor.org/info/rfc8832>.

[RFC8833] Thomson, M., "Application-Layer Protocol Negotiation (ALPN) for WebRTC", RFC 8833, DOI 10.17487/RFC8833, June 2020, <https://www.rfc-editor.org/info/rfc8833>.

[RFC8834] Perkins, C., Westerlund, M., and J. Ott, "Media Transport and Use of RTP in WebRTC", RFC 8834, DOI 10.17487/RFC8834, June 2020, <https://www.rfc-editor.org/info/rfc8834>.

[RFC8836] Jesup, R. and Z. Sarker, Ed., "Congestion Control Requirements for Interactive Real-Time Media", RFC 8836, DOI 10.17487/RFC8836, June 2020, <https://www.rfc-editor.org/info/rfc8836>.

[RFC8837] Jones, P., Dhesikan, S., Jennings, C., and D. Druta, "Differentiated Services Code Point (DSCP) Packet Markings for WebRTC QoS", RFC 8837, DOI 10.17487/RFC8837, June 2020, <https://www.rfc-editor.org/info/rfc8837>.

[RFC8841] Holmberg, C., Shpount, R., Loreto, S., and G. Camarillo, "Session Description Protocol (SDP) Offer/Answer Procedures for Stream Control Transmission Protocol (SCTP) over Datagram Transport Layer Security (DTLS) Transport", RFC 8841, DOI 10.17487/RFC8841, June 2020, <https://www.rfc-editor.org/info/rfc8841>.

[RFC8842] Holmberg, C. and R. Shpount, "Session Description Protocol (SDP) Offer/Answer Considerations for Datagram Transport Layer Security (DTLS) and Transport Layer Security (TLS)", RFC 8842, DOI 10.17487/RFC8842, June 2020, <https://www.rfc-editor.org/info/rfc8842>.

7.2. Informative References

[ANRW16] Barik, R., Welzl, M., and A. Elmokashfi, "How to say that you're special: Can we use bits in the IPv4 header?", ANRW '16: Proceedings of the 2016 Applied Networking Research Workshop, pages 68-70, DOI 10.1145/2959424.2959442, July 2016, <https://irtf.org/anrw/2016/anrw16-final17.pdf>.

[RETURN] Schwartz, B. and J. Uberti, "Recursively Encapsulated TURN (RETURN) for Connectivity and Privacy in WebRTC", Work in Progress, Internet-Draft, draft-ietf-rtcweb-return-02, 27 March 2017, <https://tools.ietf.org/html/draft-ietf-rtcweb-return-02>.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, DOI 10.17487/RFC3484, February 2003, <https://www.rfc-editor.org/info/rfc3484>.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <https://www.rfc-editor.org/info/rfc5014>.

[RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March 2008, <https://www.rfc-editor.org/info/rfc5128>.

[RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <https://www.rfc-editor.org/info/rfc7657>.

[RFC8155] Patil, P., Reddy, T., and D. Wing, "Traversal Using Relays around NAT (TURN) Server Auto Discovery", RFC 8155, DOI 10.17487/RFC8155, April 2017, <https://www.rfc-editor.org/info/rfc8155>.

[RFC8699] Islam, S., Welzl, M., and S. Gjessing, "Coupled Congestion Control for RTP Media", RFC 8699, DOI 10.17487/RFC8699, January 2020, <https://www.rfc-editor.org/info/rfc8699>.

Acknowledgements

This document is based on earlier draft versions embedded in [RFC8825], which were the result of contributions from many RTCWEB Working Group members.

Special thanks for reviews of earlier draft versions of this document go to Eduardo Gueiros, Magnus Westerlund, Markus Isomaki, and Dan Wing; the contributions from Andrew Hutton also deserve special mention.

Author's Address

Harald Alvestrand
Google
Email: harald@alvestrand.no