Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2012

H. Alvestrand
Google

June 20, 2012

Overview: Real Time Protocols for Brower-based Applications draft-ietf-rtcweb-overview-04

Abstract

This document gives an overview and context of a protocol suite intended for use with real-time applications that can be deployed in browsers - "real time communication on the Web".

It intends to serve as a starting and coordination point to make sure all the parts that are needed to achieve this goal are findable, and that the parts that belong in the Internet protocol suite are fully specified and on the right publication track.

This document is a work item of the RTCWEB working group.

Requirements Language

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 RFC 2119 [RFC2119].

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 22, 2012.

Copyright Notice

Table of Contents

1. Introduction

The Internet was, from very early in its lifetime, considered a possible vehicle for the deployment of real-time, interactive applications - with the most easily imaginable being audio conversations (aka "Internet telephony") and videoconferencing.

The first attempts to build this were dependent on special networks, special hardware and custom-built software, often at very high prices or at low quality, placing great demands on the infrastructure.

As the available bandwidth has increased, and as processors and other hardware has become ever faster, the barriers to participation have decreased, and it has become possible to deliver a satisfactory experience on commonly available computing hardware.

Still, there are a number of barriers to the ability to communicate universally - one of these is that there is, as of yet, no single set of communication protocols that all agree should be made available for communication; another is the sheer lack of universal identification systems (such as is served by telephone numbers or email addresses in other communications systems).

Development of The Universal Solution has proved hard, however, for all the usual reasons.

The last few years have also seen a new platform rise for deployment of services: The browser-embedded application, or "Web application". It turns out that as long as the browser platform has the necessary interfaces, it is possible to deliver almost any kind of service on it.

Traditionally, these interfaces have been delivered by plugins, which had to be downloaded and installed separately from the browser; in the development of HTML5, application developers see much promise in the possibility of making those interfaces available in a standardized way within the browser.

This memo describes a set of building blocks that can be made accessible and controllable through a Javascript API in a browser, and which together form a sufficient set of functions to allow the use of interactive audio and video in applications that communicate directly between browsers across the Internet. The resulting protocol suite is intended to enable all the applications that are described as required scenarios in the RTCWEB use cases document [I-D.ietf-rtcweb-use-cases-and-requirements].

Other efforts, for instance the W3C WebRTC, Web Applications and

Device API working groups, focus on making standardized APIs and interfaces available, within or alongside the HTML5 effort, for those functions; this memo concentrates on specifying the protocols and subprotocols that are needed to specify the interactions that happen across the network.

2. Principles and Terminology

2.1. Goals of this document

The goal of the RTCWEB protocol specification is to specify a set of protocols that, if all are implemented, will allow an implementation to communicate with another implementation using audio, video and data sent along the most direct possible path between the participants.

This document is intended to serve as the roadmap to the RTCWEB specifications. It defines terms used by other pieces of specification, lists references to other specifications that don't need further elaboration in the RTCWEB context, and gives pointers to other documents that form part of the RTCWEB suite.

By reading this document and the documents it refers to, it should be possible to have all information needed to implement an RTCWEB compatible implementation.

2.2. Relationship between API and protocol

The total RTCWEB/WEBRTC effort consists of two pieces:

- A protocol specification, done in the IETF
- A Javascript API specification, done in the W3C [W3C.WD-webrtc-20120209]

Together, these two specifications aim to provide an environment where Javascript embedded in any page, viewed in any compatible browser, when suitably authorized by its user, is able to set up communication using audio, video and auxiliary data, where the browser environment does not constrain the types of application in which this functionality can be used.

The protocol specification does not assume that all implementations implement this API; it is not intended to be necessary for interoperation to know whether the entity one is communicating with is a browser or another device implementing this specification.

The goal of cooperation between the protocol specification and the API specification is that for all options and features of the protocol specification, it should be clear which API calls to make to exercise that option or feature; similarly, for any sequence of API calls, it should be clear which protocol options and features will be invoked. Both subject to constraints of the implementation, of course.

2.3. On interoperability and innovation

The "Mission statement of the IETF" [RFC3935] states that "The benefit of a standard to the Internet is in interoperability - that multiple products implementing a standard are able to work together in order to deliver valuable functions to the Internet's users."

Communication on the Internet frequently occurs in two phases:

- Two parties communicate, through some mechanism, what functionality they both are able to support

- They use that shared communicative functionality to communicate, or, failing to find anything in common, give up on communication.

There are often many choices that can be made for communicative functionality; the history of the Internet is rife with the proposal, standardization, implementation, and success or failure of many types of options, in all sorts of protocols.

The goal of having a mandatory to implement function set is to prevent negotiation failure, not to preempt or prevent negotiation.

The presence of a mandatory to implement function set serves as a strong changer of the marketplace of deployment - in that it gives a guarantee that, as long as you conform to a specification, and the other party is willing to accept communication at the base level of that specification, you can communicate successfully.

The alternative - that of having no mandatory to implement - does not mean that you cannot communicate, it merely means that in order to be part of the communications partnership, you have to implement the standard "and then some" - that "and then some" usually being called a profile of some sort; in the version most antithetical to the Internet ethos, that "and then some" consists of having to use a specific vendor's product only.

2.4. Terminology

The following terms are used in this document, and as far as possible across the documents specifying the RTCWEB suite, in the specific meanings given here. Not all terms are used in this document. Other terms are used in their commonly used meaning.

The list is in alphabetical order.

Agent: Undefined term. See "SDP Agent" and "ICE Agent".

API: Application Programming Interface - a specification of a set of calls and events, usually tied to a programming language or an abstract formal specification such as WebIDL, with its defined semantics.

Browser: Used synonymously with "Interactive User Agent" as defined in the HTML specification [W3C.WD-html5-20110525].

ICE Agent: An implementation of the ICE [RFC5245] protocol. An ICE Agent may also be an SDP Agent, but there exist ICE Agents that do not use SDP (for instance those that use Jingle).

Interactive: Communication between multiple parties, where the expectation is that an action from one party can cause a reaction by another party, and the reaction can be observed by the first party, with the total time required for the action/reaction/observation is on the order of no more than hundreds of milliseconds.

Media: Audio and video content. Not to be confused with "transmission media" such as wires.

Media path: The path that media data follows from one browser to another.

Protocol: A specification of a set of data units, their representation, and rules for their transmission, with their defined semantics. A protocol is usually thought of as going between systems.

Real-time media: Media where generation of content and display of content are intended to occur closely together in time (on the order of no more than hundreds of milliseconds). Real-time media can be used to support interactive communication.

SDP Agent: The protocol implementation involved in the SDP offer/ answer exchange, as defined in [RFC3264] section 3.

Signaling: Communication that happens in order to establish, manage and control media paths.

Signaling Path: The communication channels used between entities participating in signalling to transfer signaling. There may be more entities in the signaling path than in the media path.

NOTE: Where common definitions exist for these terms, those definitions should be used to the greatest extent possible.

TODO: Extend this list with other terms that might prove slippery.

3. Architecture and Functionality groups

The model of real-time support for browser-based applications does not envisage that the browser will contain all the functions that need to be performed in order to have a function such as a telephone or a videoconferencing unit; the vision is that the browser will have the functions that are needed for a Web application, working in conjunction with its backend servers, to implement these functions.

This means that two vital interfaces need specification: The protocols that browsers talk to each other, without any intervening servers, and the APIs that are offered for a Javascript application to take advantage of the browser's functionality.

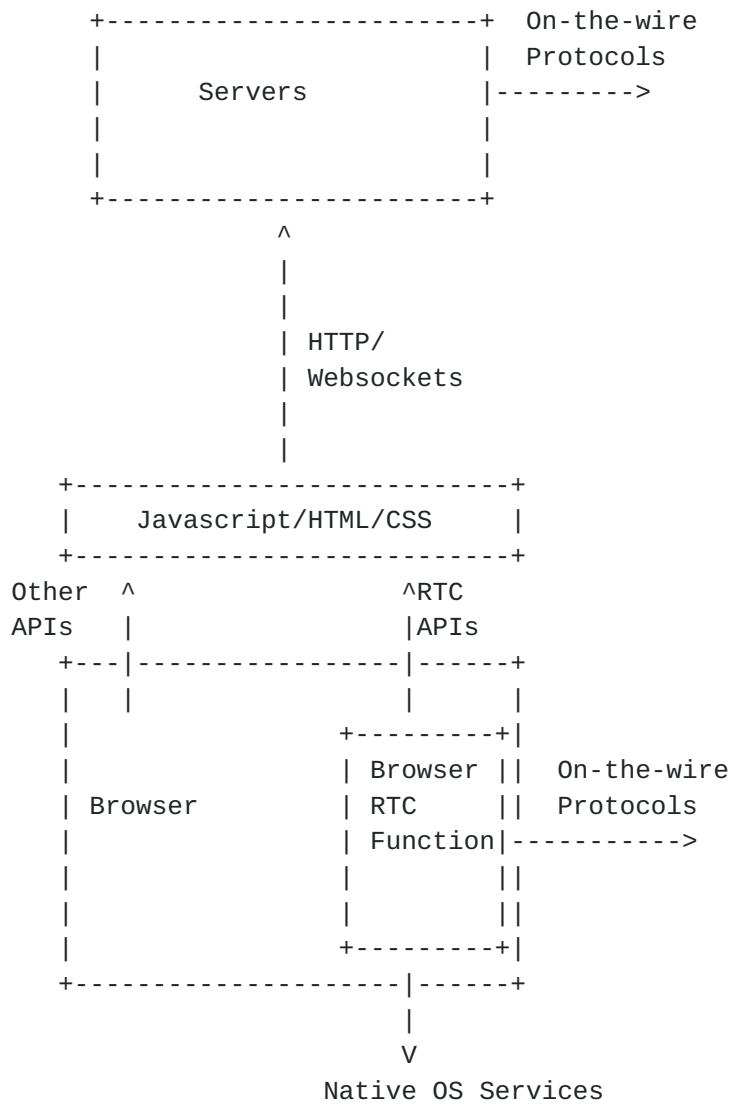

Figure 1: Browser Model

Note that HTTP and Websockets are also offered to the Javascript application through browser APIs.

As for all protocol and API specifications, there is no restriction that the protocols can only be used to talk to another browser; since they are fully specified, any device that implements the protocols faithfully should be able to interoperate with the application

running in the browser.

A commonly imagined model of deployment is the one depicted below.

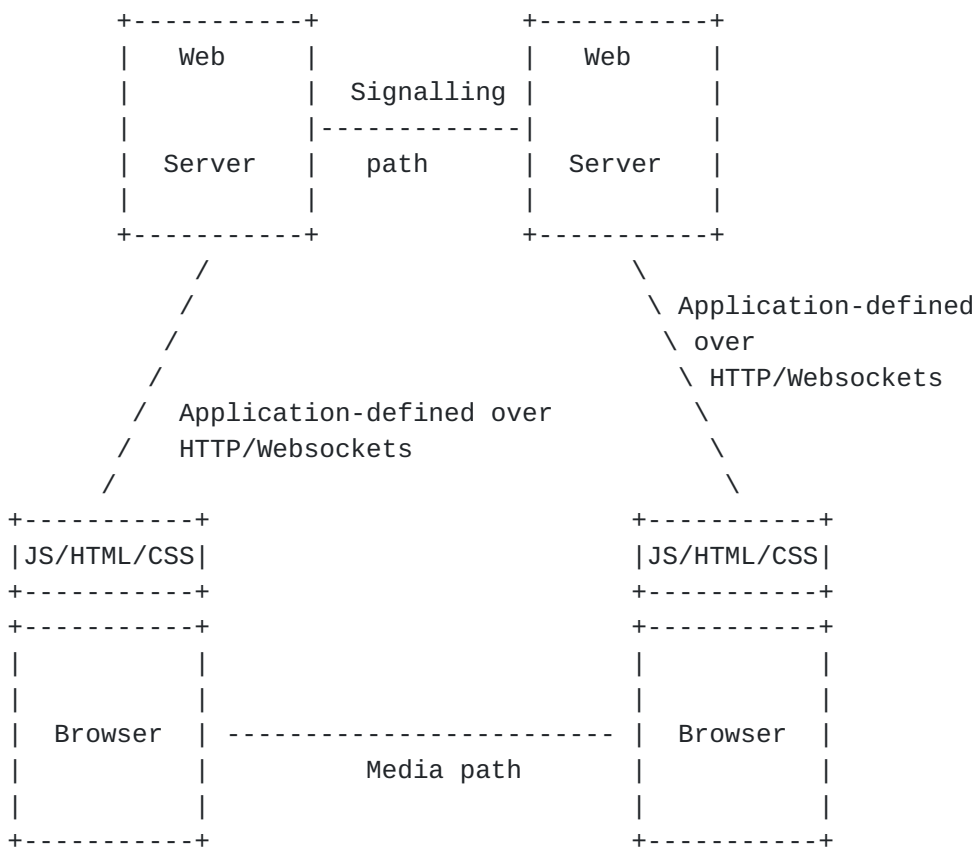

Figure 2: Browser RTC Trapezoid

On this drawing, the critical part to note is that the media path ("low path") goes directly between the browsers, so it has to be conformant to the specifications of the RTCWEB protocol suite; the signalling path ("high path") goes via servers that can modify, translate or massage the signals as needed.

If the two Web servers are operated by different entities, the inter-server signalling mechanism needs to be agreed upon, either by standardization or by other means of agreement. Existing protocols (for example SIP or XMPP) could be used between servers, while either a standards-based or proprietary protocol could be used between the browser and the web server.

For example, if both operators' servers implement SIP, SIP could be used for communication between servers, along with either a standardized signaling mechanism (e.g. SIP over Websockets) or a

proprietary signaling mechanism used between the application running in the browser and the web server. Similarly, if both operators' servers implement XMPP, XMPP couild be used for communication between XMPP servers, with either a standardized signaling mechanism (e.g. XMPP over Websockets or BOSH) or a proprietary signaling mechanism used between the application running in the browser and the web server.

The choice of protocols, and definition of the translation between them, is outside the scope of the RTCWEB standards suite described in the document.

The functionality groups that are needed in the browser can be specified, more or less from the bottom up, as:

- Data transport: TCP, UDP and the means to securely set up connections between entities, as well as the functions for deciding when to send data: Congestion management, bandwidth estimation and so on.

- Data framing: RTP and other data formats that serve as containers, and their functions for data confidentiality and integrity.

- Data formats: Codec specifications, format specifications and functionality specifications for the data passed between systems. Audio and video codecs, as well as formats for data and document sharing, belong in this category. In order to make use of data formats, a way to describe them, a session description, is needed.

- Connection management: Setting up connections, agreeing on data formats, changing data formats during the duration of a call; SIP and Jingle/XMPP belong in this category.

- Presentation and control: What needs to happen in order to ensure that interactions behave in a non-surprising manner. This can include floor control, screen layout, voice activated image switching and other such functions - where part of the system require the cooperation between parties. XCON and Cisco/ Tandberg's TIP were some attempts at specifying this kind of functionality; many applications have been built without standardized interfaces to these functions.

- Local system support functions: These are things that need not be specified uniformly, because each participant may choose to do these in a way of the participant's choosing, without affecting the bits on the wire in a way that others have to be cognizant of. Examples in this category include echo cancellation (some forms of it), local authentication and authorization mechanisms, OS access

control and the ability to do local recording of conversations.

Within each functionality group, it is important to preserve both freedom to innovate and the ability for global communication. Freedom to innovate is helped by doing the specification in terms of interfaces, not implementation; any implementation able to communicate according to the interfaces is a valid implementation. Ability to communicate globally is helped both by having core specifications be unencumbered by IPR issues and by having the formats and protocols be fully enough specified to allow for independent implementation.

One can think of the three first groups as forming a "media transport infrastructure", and of the three last groups as forming a "media service". In many contexts, it makes sense to use a common specification for the media transport infrastructure, which can be embedded in browsers and accessed using standard interfaces, and "let a thousand flowers bloom" in the "media service" layer; to achieve interoperable services, however, at least the first five of the six groups need to be specified.

4. Data transport

Data transport refers to the sending and receiving of data over the network interfaces, the choice of network-layer addresses at each end of the communication, and the interaction with any intermediate entities that handle the data, but do not modify it (such as TURN relays).

It includes necessary functions for congestion control: When not to send data.

T are described in <WORKING GROUP DRAFT "TRANSPORTS">.

ICE is required for all media paths that use UDP; in addition to the ability to pass NAT boxes, ICE fulfils the need for guaranteeing that the media path is going to an UDP port that is willing to receive the data.

The data transport protocols used by RTCWEB, as well as the details of interactions with intermediate boxes, such as firewalls, relays and NAT boxes, are intended to be described in a separate document; for now, notes are gathered in Appendix A.

5. Data framing and securing

The format for media transport is RTP [RFC3550]. Implementation of SRTP [RFC3711] is required for all implementations.

The detailed considerations for usage of functions from RTP and SRTP are given in [I-D.ietf-rtcweb-rtp-usage]. The security considerations for the RTCWEB use case are in [I-D.ietf-rtcweb-security], and the resulting security functions are described in [I-D.ietf-rtcweb-security-arch].

Considerations for the transfer of data that is not in RTP format is described in [I-D.ietf-rtcweb-data-channel], and the resulting protocol is described in [I-D.jesup-rtcweb-data-protocol] (not yet a WG document)

6. Data formats

The intent of this specification is to allow each communications event to use the data formats that are best suited for that particular instance, where a format is supported by both sides of the connection. However, a minimum standard is greatly helpful in order to ensure that communication can be achieved. This document specifies a minimum baseline that will be supported by all implementations of this specification, and leaves further codecs to be included at the will of the implementor.

The mandatory to implement codecs, as well as any profiling requirements for both mandatory and optional codecs, is described in <WORKING GROUP DRAFT "MEDIA PROCESSING"> (candidate draft: [I-D.cbran-rtcweb-codec].

7. Connection management

The methods, mechanisms and requirements for setting up, negotiating and tearing down connections is a large subject, and one where it is desirable to have both interoperability and freedom to innovate.

The following principles apply:

1. The RTCWEB media negotiations will be capable of representing the same SDP offer/answer semantics that are used in SIP [RFC3264], in such a way that it is possible to build a signalling gateway between SIP and the RTCWEB media negotiation.

2. It will be possible to gateway between legacy SIP devices that support ICE and appropriate RTP / SDP mechanisms, codecs and security mechanisms without using a media gateway. A signaling gateway to convert between the signaling on the web side to the SIP signaling may be needed.

3. When a new codec is specified, and the SDP for the new codec is specified in the MMUSIC WG, no other standardization would should be required for it to be possible to use that in the web browsers. Adding new codecs which might have new SDP parameters should not change the APIs between the browser and javascript application. As soon as the browsers support the new codecs, old applications written before the codecs were specified should automatically be able to use the new codecs where appropriate with no changes to the JS applications.

The particular choices made for RTCWEB, and their implications for the API offered by a browser implementing RTCWEB, are described in [I-D.ietf-rtcweb-jsep]

8. Presentation and control

The most important part of control is the user's control over the browser's interaction with input/output devices and communications channels. It is important that the user have some way of figuring out where his audio, video or texting is being sent, for what purported reason, and what guarantees are made by the parties that form part of this control channel. This is largely a local function between the browser, the underlying operating system and the user interface; this is being worked on as part of the W3C API effort, and will be part of the peer connection API [W3C.WD-webrtc-20120209], and the device control API [getusermedia]. Considerations for the implications of wanting to identify correspondents are described in [I-D.rescorla-rtcweb-generic-idp] (not a WG item).

9. Local system support functions

These are characterized by the fact that the quality of these functions strongly influences the user experience, but the exact algorithm does not need coordination. In some cases (for instance echo cancellation, as described below), the overall system definition may need to specify that the overall system needs to have some characteristics for which these facilities are useful, without requiring them to be implemented a certain way.

Local functions include echo cancellation, volume control, camera

management including focus, zoom, pan/tilt controls (if available), and more.

Certain parts of the system SHOULD conform to certain properties, for instance:

- Echo cancellation should be good enough to achieve the suppression of acoustical feedback loops below a perceptually noticeable level.

- Privacy concerns must be satisfied; for instance, if remote control of camera is offered, the APIs should be available to let the local participant to figure out who's controlling the camera, and possibly decide to revoke the permission for camera usage.

- Automatic gain control, if present, should normalize a speaking voice into <whatever dB metrics makes sense here - most important that we have one only>

The requirements on RTCWEB systems in this category are found in <WORKING GROUP DRAFT "MEDIA PROCESSING">; the proposed API for control of local devices are found in [getusermedia].

10. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

11. Security Considerations

Security of the web-enabled real time communications comes in several pieces:

- Security of the components: The browsers, and other servers involved. The most target-rich environment here is probably the browser; the aim here should be that the introduction of these components introduces no additional vulnerability.

- Security of the communication channels: It should be easy for a participant to reassure himself of the security of his communication - by verifying the crypto parameters of the links he himself participates in, and to get reassurances from the other parties to the communication that they promise that appropriate measures are taken.

- Security of the partners' identity: verifying that the participants are who they say they are (when positive identification is appropriate), or that their identity cannot be uncovered (when anonymity is a goal of the application).

The security analysis, and the requirements derived from that analysis, is contained in [I-D.ietf-rtcweb-security].

12. Acknowledgements

The number of people who have taken part in the discussions surrounding this draft are too numerous to list, or even to identify. The ones below have made special, identifiable contributions; this does not mean that others' contributions are less important.

Thanks to Cary Bran, Cullen Jennings, Colin Perkins, Magnus Westerlund and Joerg Ott, who offered technical contributions on various versions of the draft.

Thanks to Jonathan Rosenberg, Matthew Kaufman and others at Skype for the ASCII drawings in section 1.

Thanks to Eric Rescorla, Justin Uberti, Henry Sinnreich, Colin Perkins and Simon Leinen for document review.

13. References

13.1. Normative References

[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-00 (work in progress), July 2011.

[I-D.ietf-rtcweb-data-channel]
Jesup, R., Loreto, S., and M. Tuexen, "RTCWeb Datagram Connection", draft-ietf-rtcweb-data-channel-00 (work in progress), March 2012.

[I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-00 (work in progress), March 2012.

[I-D.ietf-rtcweb-rtp-usage]

Perkins, C., Westerlund, M., and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP", draft-ietf-rtcweb-rtp-usage-01 (work in progress), October 2011.

[I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for RTC-Web", draft-ietf-rtcweb-security-01 (work in progress), October 2011.

[I-D.ietf-rtcweb-security-arch]
Rescorla, E., "RTCWEB Security Architecture", draft-ietf-rtcweb-security-arch-00 (work in progress), January 2012.

[I-D.nandakumar-rtcweb-stun-uri]
Nandakumar, S., Salgueiro, G., and P. Jones, "URI Scheme for Session Traversal Utilities for NAT (STUN) Protocol", draft-nandakumar-rtcweb-stun-uri-00 (work in progress), October 2011.

[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-00 (work in progress), March 2012.

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

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

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

[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, April 2010.

13.2. Informative References

[I-D.cbran-rtcweb-codec]
Bran, C. and C. Jennings, "WebRTC Codec and Media Processing Requirements", draft-cbran-rtcweb-codec-01 (work in progress), October 2011.

[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-06 (work in progress), October 2011.

[I-D.jesup-rtcweb-data-protocol]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel Protocol", draft-jesup-rtcweb-data-protocol-00 (work in progress), March 2012.

[I-D.rescorla-rtcweb-generic-idp]
Rescorla, E., "RTCWEB Generic Identity Provider Interface", draft-rescorla-rtcweb-generic-idp-00 (work in progress), January 2012.

[RFC3935] Alvestrand, H., "A Mission Statement for the IETF", BCP 95, RFC 3935, October 2004.

[W3C.WD-html5-20110525]
Hickson, I., "HTML5", World Wide Web Consortium LastCall WD-html5-20110525, May 2011, <http://www.w3.org/TR/2011/WD-html5-20110525>.

[W3C.WD-webrtc-20120209]
Bergkvist, A., Burnett, D., Narayanan, A., and C. Jennings, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120209, February 2012, <http://www.w3.org/TR/2012/WD-webrtc-20120209>.

[getusermedia]
Burnett, D. and A. Narayanan, "getusermedia: Getting access to local devices that can generate multimedia streams", December 2011, <http://dev.w3.org/2011/webrtc/editor/getusermedia.html>.

Appendix A. Transport and Middlebox specification

The draft referred to as "transport and middle boxes" in Section 4 has not been written yet. This appendix contains some keywords to what it should say; this also serves the purpose of linking to the drafts-in-progress that are relevant to this specification.

A.1. System-provided interfaces

The protocol specifications used here assume that the following protocols are available as system-level interfaces:

- UDP. This is the protocol assumed by most protocol elements described.

- TCP. This is used for HTTP/WebSockets, as well as for TURN/SSL and ICE-TCP.

For both protocols, we assume the ability to set the DSCP code point of the sockets opened. We do not assume that the DSCP codepoints will be honored, and we do assume that they may be zeroed or changed, since this is a local configuration issue.

We do not assume that the implementation will have access to ICMP or raw IP.

A.2. Middle box related functions

The primary mechanism to deal with middle boxes 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's in response to inside traffic (simple stateful firewalls).

In order to deal with symmetric NATs, TURN MUST be supported.

In order to deal with firewalls that block all UDP traffic, TURN over TCP MUST be supported. (QUESTION: What about ICE-TCP?)

The following specifications MUST be supported:

- ICE [RFC5245]

- TURN, including TURN over TCP [[QUESTION: and TURN over TLS]], [RFC5766].

For referring to ICE servers, we use the STUN URI, [I-D.nandakumar-rtcweb-stun-uri].

A.3. Transport protocols implemented

For data transport, we implement SCTP over DTLS over ICE. This is specified in [I-D.tuexen-tsvwg-sctp-dtls-encaps]. Negotiation of this transport in SCTP is defined in [I-D.ietf-mmusic-sctp-sdp].

Appendix B. Change log

This section may be deleted by the RFC Editor when preparing for publication.

B.1. Changes from draft-alvestrand-dispatch-rtcweb-datagram-00 to -01

Added section "On interoperability and innovation"

Added data confidentiality and integrity to the "data framing" layer

Added congestion management requirements in the "data transport" layer section

Changed need for non-media data from "question: do we need this?" to "Open issue: How do we do this?"

Strengthened disclaimer that listed codecs are placeholders, not decisions.

More details on why the "local system support functions" section is there.

B.2. Changes from draft-alvestrand-dispatch-01 to draft-alvestrand-rtcweb-overview-00

Added section on "Relationship between API and protocol"

Added terminology section

Mentioned congestion management as part of the "data transport" layer in the layer list

B.3. Changes from draft-alvestrand-rtcweb-00 to -01

Removed most technical content, and replaced with pointers to drafts as requested and identified by the RTCWEB WG chairs.

Added content to acknowledgements section.

Added change log.

Spell-checked document.

B.4. Changes from draft-alvestrand-rtcweb-overview-01 to draft-ietf-rtcweb-overview-00

Changed draft name and document date.

Removed unused references

B.5. Changes from -00 to -01 of draft-ietf-rtcweb-overview

Added architecture figures to section 2.

Changed the description of "echo cancellation" under "local system support functions".

Added a few more definitions.

B.6. Changes from -01 to -02 of draft-ietf-rtcweb-overview

Added pointers to use cases, security and rtp-usage drafts (now WG drafts).

Changed description of SRTP from mandatory-to-use to mandatory-to-implement.

Added the "3 principles of negotiation" to the connection management section.

Added an explicit statement that ICE is required for both NAT and consent-to-receive.

B.7. Changes from -02 to -03 of draft-ietf-rtcweb-overview

Added references to a number of new drafts.

Expanded the description text under the "trapezoid" drawing with some more text discussed on the list.

Changed the "Connection management" sentence from "will be done using SDP offer/answer" to "will be capable of representing SDP offer/answer" - this seems more consistent with JSEP.

Added "security mechanisms" to the things a non-gatewayed SIP devices must support in order to not need a media gateway.

Added a definition for "browser".

B.8. Changes from -03 to -04 of draft-ietf-rtcweb-overview

Made introduction more normative.

Several wording changes in response to review comments from EKR

Added Appendix A to hold references and notes that are not yet in a separate document.

Author's Address

Harald T. Alvestrand
Google
Kungsbron 2
Stockholm, 11122
Sweden

Email: harald@alvestrand.no