```
Network Working Group                                      H. Alvestrand
Internet-Draft                                                    Google
Intended status: Standards Track                      November 12, 2017
Expires: May 16, 2018
```

Overview: Real Time Protocols for Browser-based Applications
draft-ietf-rtcweb-overview-19

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 an Applicability Statement - it does not itself
   specify any protocol, but specifies which other specifications WebRTC
   compliant implementations are supposed to follow.

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

Status of This Memo

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 video conferencing.

   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, however, proved hard.

   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 use cases document [RFC7478].

Other efforts, for instance the W3C Web Real-Time Communications, Web
Applications Security, and Device and Sensor 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 over the network.

Operators should note that deployment of WebRTC will result in a
change in the nature of signaling for real time media on the network,
and may result in a shift in the kinds of devices used to create and
consume such media. In the case of signaling, WebRTC session setup
will typically occur over TLS-secured web technologies using
application-specific protocols. Operational techniques that involve
inserting network elements to interpret SDP -- either through
endpoint cooperation [RFC3361] or through the transparent insertion
of SIP Application Level Gateways (ALGs) -- will not work with such
signaling. In the case of networks using cooperative endpoints, the
approaches defined in [RFC8155] may serve as a suitable replacement
for [RFC3361]. The increase in browser-based communications may also
lead to a shift away from dedicated real-time-communications
hardware, such as SIP desk phones. This will diminish the efficacy
of operational techniques that place dedicated real-time devices on
their own network segment, address range, or VLAN for purposes such
as applying traffic filtering and QoS. Applying the markings
described in [I-D.ietf-tsvwg-rtcweb-qos] may be appropriate
replacements for such techniques.

This memo uses the term "WebRTC" (note the case used) to refer to the
overall effort consisting of both IETF and W3C efforts.

2. Principles and Terminology

2.1. Goals of this document

The goal of the WebRTC 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 WebRTC
specifications. It defines terms used by other parts of the WebRTC
protocol specifications, lists references to other specifications
that don't need further elaboration in the WebRTC context, and gives
pointers to other documents that form part of the WebRTC suite.

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

2.2. Relationship between API and protocol

The total WebRTC effort consists of two major parts, each consisting
of multiple documents:

o A protocol specification, done in the IETF

o A Javascript API specification, defined in a series of W3C
   documents
   [W3C.WD-webrtc-20120209][W3C.WD-mediacapture-streams-20120628]

Together, these two specifications aim to provide an environment
where Javascript embedded in any page, when suitably authorized by
its user, is able to set up communication using audio, video and
auxiliary data, as long as the browser supports this specification.
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.

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

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

Application Programming Interface (API): 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]. See also
   "WebRTC User Agent".

Data Channel: An abstraction that allows data to be sent between
   WebRTC endpoints in the form of messages. Two endpoints can have
   multiple data channels between them.

ICE Agent: An implementation of the Interactive Connectivity
   Establishment (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 [XEP-0166]).

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 WebRTC
   endpoint 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 Session
   Description Protocol (SDP) offer/answer exchange, as defined in
   [RFC3264] section 3.

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

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

WebRTC Browser: (also called a WebRTC User Agent or WebRTC UA)
    Something that conforms to both the protocol specification and the
    Javascript API cited above.

WebRTC non-Browser: Something that conforms to the protocol
    specification, but does not claim to implement the Javascript API.
    This can also be called a "WebRTC device" or "WebRTC native
    application".

WebRTC Endpoint: Either a WebRTC browser or a WebRTC non-browser.
    It conforms to the protocol specification.

WebRTC-compatible Endpoint: An endpoint that is able to successfully
    communicate with a WebRTC endpoint, but may fail to meet some
    requirements of a WebRTC endpoint. This may limit where in the
    network such an endpoint can be attached, or may limit the
    security guarantees that it offers to others. It is not
    constrained by this specification; when it is mentioned at all, it
    is to note the implications on WebRTC-compatible endpoints of the
    requirements placed on WebRTC endpoints.

WebRTC Gateway: A WebRTC-compatible endpoint that mediates media
    traffic to non-WebRTC entities.

All WebRTC browsers are WebRTC endpoints, so any requirement on a
WebRTC endpoint also applies to a WebRTC browser.

A WebRTC non-browser may be capable of hosting applications in a
similar way to the way in which a browser can host Javascript
applications, typically by offering APIs in other languages. For
instance it may be implemented as a library that offers a C++ API
intended to be loaded into applications. In this case, similar
security considerations as for Javascript may be needed; however,
since such APIs are not defined or referenced here, this document
cannot give any specific rules for those interfaces.

WebRTC gateways are described in a separate document,
[I-D.ietf-rtcweb-gateways].

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:

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

o 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 is 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". The "and then some" is usually 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 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. Architecture and Functionality groups

For browser-based applications, the model for real-time support does
not assume that the browser will contain all the functions needed for
an application such as a telephone or a video conference. The vision
is that the browser will have the functions 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 use to 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.

```
              +------------------------+  On-the-wire
              |                        |  Protocols
              |        Servers         |--------->
              |                        |
              |                        |
              +------------------------+
                         ^
                         |
                         |
                         | HTTPS/
                         | WebSockets
                         |
                         |
           +------------------------------+
           |     Javascript/HTML/CSS      |
           +------------------------------+
          Other  ^                ^ RTC
          APIs   |                | APIs
           +---|-----------------|------+
           |   |                 |      |
           |               +---------+|
           |               | Browser ||  On-the-wire
           | Browser       | RTC     ||  Protocols
           |               | Function|----------->
           |               |         ||
           |               |         ||
           |               +---------+|
           +--------------------|------+
                               |
                               V
                        Native OS Services
```
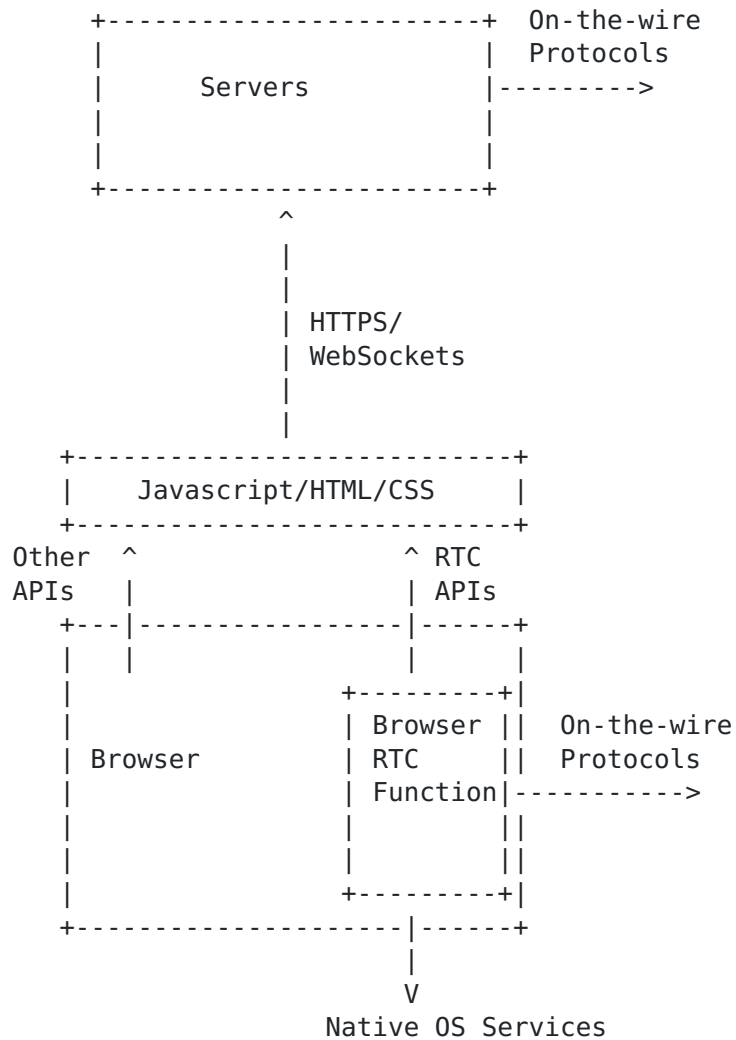
                        Figure 1: Browser Model

   Note that HTTPS 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 endpoint 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.
   In the figure below JS is Javascript.


```
              +-----------+                +-----------+
              |   Web     |                |   Web     |
              |           |  Signaling     |           |
              |           |------------|   |           |
              |  Server   |   path     |   |  Server   |
              |           |            |   |           |
              +-----------+                +-----------+
                   /                            \
                  /                              \ Application-defined
                 /                                \ over
                /                                  \ HTTPS/WebSockets
               /  Application-defined over          \
              /   HTTPS/WebSockets                    \
             /                                          \
        +-----------+                            +-----------+
        |JS/HTML/CSS|                            |JS/HTML/CSS|
        +-----------+                            +-----------+
        +-----------+                            +-----------+
        |           |                            |           |
        |           |                            |           |
        |  Browser  | -------------------------  |  Browser  |
        |           |        Media path          |           |
        |           |                            |           |
        +-----------+                            +-----------+
```
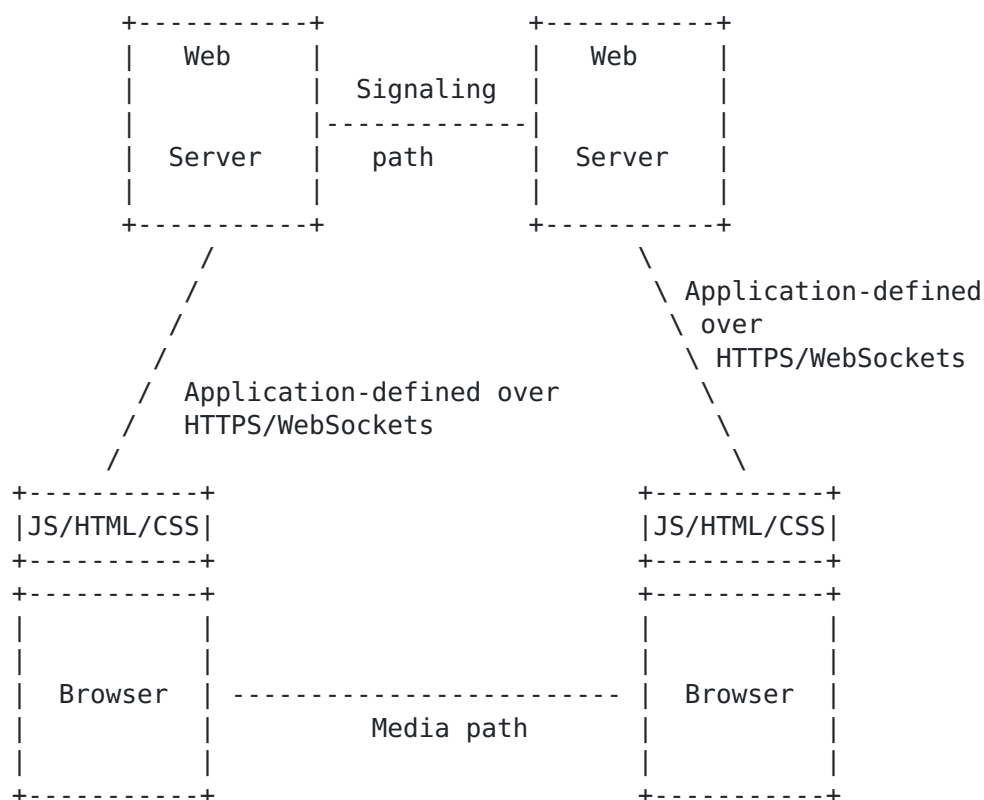
                    Figure 2: Browser RTC Trapezoid

   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 WebRTC protocol suite; the
   signaling path ("high path") goes via servers that can modify,
   translate or manipulate the signals as needed.

   If the two Web servers are operated by different entities, the inter-
   server signaling mechanism needs to be agreed upon, either by

standardization or by other means of agreement. Existing protocols
(e.g. SIP [RFC3261] or XMPP [RFC6120]) 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 Extensible Messaging and Presence Protocol (XMPP),
XMPP could be used for communication between XMPP servers, with
either a standardized signaling mechanism (e.g. XMPP over WebSockets
or BOSH [XEP-0124] or a proprietary signaling mechanism used between
the application running in the browser and the web server.

The choice of protocols for client-server and inter-server
signalling, and definition of the translation between them, is
outside the scope of the WebRTC protocol 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:

o Data transport: such as 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.

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

o 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.

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

o 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.

   o 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,
   retransmission, and in-order delivery.

   WebRTC endpoints MUST implement the transport protocols described in
   [I-D.ietf-rtcweb-transports].

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 WebRTC 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 a supporting
   protocol for establishing individual data channels is described in
   [I-D.ietf-rtcweb-data-protocol]. WebRTC endpoints MUST implement
   these two specifications.

   WebRTC endpoints MUST implement [I-D.ietf-rtcweb-rtp-usage],
   [I-D.ietf-rtcweb-security], [I-D.ietf-rtcweb-security-arch], and the
   requirements they include.

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.

   WebRTC endpoints that support audio and/or video MUST implement the
   codecs and profiles required in [RFC7874] and [RFC7742].

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 WebRTC media negotiations will be capable of representing the
       same SDP offer/answer semantics [RFC3264] that are used in SIP,
       in such a way that it is possible to build a signaling gateway
       between SIP and the WebRTC 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 an SDP for a new codec is specified, no other
    standardization 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 WebRTC, and their implications for
the API offered by a browser implementing WebRTC, are described in
[I-D.ietf-rtcweb-jsep].

WebRTC browsers MUST implement [I-D.ietf-rtcweb-jsep].

WebRTC endpoints MUST implement the functions described in that
document that relate to the network layer (e.g. Bundle
[I-D.ietf-mmusic-sdp-bundle-negotiation], RTCP-mux [RFC5761] and
Trickle ICE [I-D.ietf-ice-trickle]), but do not need to support the
API functionality described there.

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 specified in the peer connection API
[W3C.WD-webrtc-20120209], and the media capture API
[W3C.WD-mediacapture-streams-20120628].

WebRTC browsers MUST implement these two specifications.

9. Local system support functions

These are characterized by the fact that the quality of these
functions strongly influence 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.

One would want to see certain parts of the system conform to certain
properties, for instance:

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

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

o Automatic gain control, if present, should normalize a speaking
   voice into a reasonable dB range.

The requirements on WebRTC systems with regard to audio processing
are found in [RFC7874] and includes more guidance about echo
cancellation and AGC; the proposed API for control of local devices
are found in [W3C.WD-mediacapture-streams-20120628].

WebRTC endpoints MUST implement the processing functions in
[RFC7874]. (Together with the requirement in Section 6, this means
that WebRTC endpoints MUST implement the whole document.)

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:

o 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.

o 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.

o 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].

It is also important to read the security sections of
[W3C.WD-mediacapture-streams-20120628] and [W3C.WD-webrtc-20120209].

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 Alissa Cooper, Bjoern Hoehrmann, Colin Perkins, Colton
Shields, Eric Rescorla, Heath Matlock, Henry Sinnreich, Justin
Uberti, Keith Drage, Magnus Westerlund, Olle E. Johansson, Sean
Turner and Simon Leinen for document review.

13. References

13.1. Normative References

[I-D.ietf-rtcweb-data-channel]
          Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data
          Channels", draft-ietf-rtcweb-data-channel-13 (work in
          progress), January 2015.

   [I-D.ietf-rtcweb-data-protocol]
              Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel
              Establishment Protocol", draft-ietf-rtcweb-data-
              protocol-09 (work in progress), January 2015.

   [I-D.ietf-rtcweb-jsep]
              Uberti, J., Jennings, C., and E. Rescorla, "JavaScript
              Session Establishment Protocol", draft-ietf-rtcweb-jsep-24
              (work in progress), October 2017.

   [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-26 (work in progress), March
              2016.

   [I-D.ietf-rtcweb-security]
              Rescorla, E., "Security Considerations for WebRTC", draft-
              ietf-rtcweb-security-09 (work in progress), October 2017.

   [I-D.ietf-rtcweb-security-arch]
              Rescorla, E., "WebRTC Security Architecture", draft-ietf-
              rtcweb-security-arch-13 (work in progress), October 2017.

   [I-D.ietf-rtcweb-transports]
              Alvestrand, H., "Transports for WebRTC", draft-ietf-
              rtcweb-transports-17 (work in progress), October 2016.

   [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>.

   [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model
              with Session Description Protocol (SDP)", RFC 3264,
              DOI 10.17487/RFC3264, June 2002,
              <https://www.rfc-editor.org/info/rfc3264>.

   [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
              Jacobson, "RTP: A Transport Protocol for Real-Time
              Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550,
              July 2003, <https://www.rfc-editor.org/info/rfc3550>.

   [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
              Norrman, "The Secure Real-time Transport Protocol (SRTP)",
              RFC 3711, DOI 10.17487/RFC3711, March 2004,
              <https://www.rfc-editor.org/info/rfc3711>.

   [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment
              (ICE): A Protocol for Network Address Translator (NAT)
              Traversal for Offer/Answer Protocols", RFC 5245,
              DOI 10.17487/RFC5245, April 2010,
              <https://www.rfc-editor.org/info/rfc5245>.

   [RFC7742] Roach, A., "WebRTC Video Processing and Codec
              Requirements", RFC 7742, DOI 10.17487/RFC7742, March 2016,
              <https://www.rfc-editor.org/info/rfc7742>.

   [RFC7874] Valin, JM. and C. Bran, "WebRTC Audio Codec and Processing
              Requirements", RFC 7874, DOI 10.17487/RFC7874, May 2016,
              <https://www.rfc-editor.org/info/rfc7874>.

   [W3C.WD-mediacapture-streams-20120628]
              Burnett, D. and A. Narayanan, "Media Capture and Streams",
              World Wide Web Consortium WD WD-mediacapture-streams-
              20120628, June 2012, <http://www.w3.org/TR/2012/
              WD-mediacapture-streams-20120628>.

   [W3C.WD-webrtc-20120209]
              Bergkvist, A., Burnett, D., Jennings, C., and A.
              Narayanan, "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>.

13.2. Informative References

   [I-D.ietf-ice-trickle]
              Ivov, E., Rescorla, E., Uberti, J., and P. Saint-Andre,
              "Trickle ICE: Incremental Provisioning of Candidates for
              the Interactive Connectivity Establishment (ICE)
              Protocol", draft-ietf-ice-trickle-14 (work in progress),
              September 2017.

   [I-D.ietf-mmusic-sdp-bundle-negotiation]
              Holmberg, C., Alvestrand, H., and C. Jennings,
              "Negotiating Media Multiplexing Using the Session
              Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-
              negotiation-39 (work in progress), August 2017.

   [I-D.ietf-rtcweb-gateways]
              Alvestrand, H. and U. Rauschenbach, "WebRTC Gateways",
              draft-ietf-rtcweb-gateways-02 (work in progress), January
              2016.

   [I-D.ietf-tsvwg-rtcweb-qos]
              Jones, P., Dhesikan, S., Jennings, C., and D. Druta, "DSCP
              Packet Markings for WebRTC QoS", draft-ietf-tsvwg-rtcweb-
              qos-18 (work in progress), August 2016.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <https://www.rfc-editor.org/info/rfc3261>.

   [RFC3361] Schulzrinne, H., "Dynamic Host Configuration Protocol
              (DHCP-for-IPv4) Option for Session Initiation Protocol
              (SIP) Servers", RFC 3361, DOI 10.17487/RFC3361, August
              2002, <https://www.rfc-editor.org/info/rfc3361>.

   [RFC3935] Alvestrand, H., "A Mission Statement for the IETF",
              BCP 95, RFC 3935, DOI 10.17487/RFC3935, October 2004,
              <https://www.rfc-editor.org/info/rfc3935>.

   [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and
              Control Packets on a Single Port", RFC 5761,
              DOI 10.17487/RFC5761, April 2010,
              <https://www.rfc-editor.org/info/rfc5761>.

   [RFC6120] Saint-Andre, P., "Extensible Messaging and Presence
              Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120,
              March 2011, <https://www.rfc-editor.org/info/rfc6120>.

   [RFC7478] Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-
              Time Communication Use Cases and Requirements", RFC 7478,
              DOI 10.17487/RFC7478, March 2015,
              <https://www.rfc-editor.org/info/rfc7478>.

   [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>.

   [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>.

   [XEP-0124]
              Paterson, I., Smith, D., Saint-Andre, P., Moffitt, J.,
              Stout, L., and W. Tilanus, "BOSH", XSF XEP 0124, November
              2016.

   [XEP-0166]
               Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan,
               S., and J. Hildebrand, "Jingle", XSF XEP 0166, June 2007.

Appendix A. Change log

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

A.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.

A.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

A.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 acknowledgments section.

   Added change log.

   Spell-checked document.

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

   Changed draft name and document date.

   Removed unused references

A.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.

A.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.

A.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".

A.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 an appendix to hold references and notes that are not yet in a
   separate document.

A.9. **Changes from -04 to -05 of draft-ietf-rtcweb-overview**

   Minor grammatical fixes. This is mainly a "keepalive" refresh.

A.10. **Changes from -05 to -06**

   Clarifications in response to Last Call review comments. Inserted
   reference to draft-ietf-rtcweb-audio.

A.11. **Changes from -06 to -07**

   Added a reference to the "unified plan" draft, and updated some
   references.

   Otherwise, it's a "keepalive" draft.

A.12. **Changes from -07 to -08**

   Removed the appendix that detailed transports, and replaced it with a
   reference to draft-ietf-rtcweb-transports. Removed now-unused
   references.

A.13. **Changes from -08 to -09**

   Added text to the Abstract indicating that the intended status is an
   Applicability Statement.

A.14. **Changes from -09 to -10**

   Defined "WebRTC Browser" and "WebRTC device" as things that do, or
   don't, conform to the API.

   Updated reference to data-protocol draft

   Updated data formats to reference -rtcweb-audio- and not the expired
   -cbran draft.

   Deleted references to -unified-plan

Deleted reference to -generic-idp (draft expired)

Added notes on which referenced documents WebRTC browsers or devices
MUST conform to.

Added pointer to the security section of the API drafts.

A.15. **Changes from -10 to -11**

Added "WebRTC Gateway" as a third class of device, and referenced the
doc describing them.

Made a number of text clarifications in response to document reviews.

A.16. **Changes from -11 to -12**

Refined entity definitions to define "WebRTC endpoint" and "WebRTC-
compatible endpoint".

Changed remaining usage of the term "RTCWEB" to "WebRTC", including
in the page header.

A.17. **Changes from -12 to -13**

Changed "WebRTC device" to be "WebRTC non-browser", per decision at
IETF 91. This led to the need for "WebRTC endpoint" as the common
label for both, and the usage of that term in the rest of the
document.

Added words about WebRTC APIs in languages other than Javascript.

Referenced draft-ietf-rtcweb-video for video codecs to support.

A.18. **Changes from -13 to -14**

None. This is a "keepalive" update.

A.19. **Changes from -14 to -15**

Changed "gateways" reference to point to the WG document.

A.20. **Changes from -15 to -16**

None. This is a "keepalive" publication.

A.21. Changes from -16 to -17

   Addressed review comments by Olle E. Johansson and Magnus Westerlund

A.22. Changes from -17 to -18

   Addressed review comments from Sean Turner and Alissa Cooper

A.23. Changes from -18 to -19

   A number of grammatical issues were fixed.

   Added note on operational impact of WebRTC.

   Unified all definitions into the definitions list.

   Added a reference for BOSH.

   Changed ICE reference from 5245bis to RFC 5245.

Author's Address

   Harald T. Alvestrand
   Google
   Kungsbron 2
   Stockholm 11122
   Sweden

   Email: harald@alvestrand.no