

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2011

H. Alvestrand
Google
March 13, 2011

Overview: Real Time Protocols for Brower-based Applications
draft-alvestrand-dispatch-rtcweb-protocols-01

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 work is an attempt to synthesize the input of many people, but makes no claims to fully represent the views of any of them. All parts of the document should be regarded as open for discussion, with the intended discussion forum being the "RTCWEB" WG (in formation).

Currently, discussion is on the rtc-web@alvestrand.no mailing list.

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 September 14, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. On interoperability and innovation	5
3. Functionality groups	5
4. Data transport	6
5. Data framing and securing	7
6. Data formats	7
7. Connection management	8
8. Presentation and control	9
9. Local system support functions	9
10. IANA Considerations	9
11. Security Considerations	10
12. Acknowledgements	10
13. References	11
13.1. Normative References	11
13.2. Informative References	12
Author's Address	12

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 is 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 are, 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. This memo aims to take a more building-block-oriented approach, and try to find consensus on a set of substrate components that we think will be useful in any real-time communications systems.

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, much promise is seen by the possibility of making those interfaces available in a standardized way within the browser.

Other efforts, for instance the W3C 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. 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 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.

3. Functionality groups

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

- o Data transport: TCP, UDP and the means to securely set up connections between entities.
- o Data framing: RTP 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.
- o 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.
- 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. Cisco/Tandberg's TIP was one attempt at specifying this functionality.
- 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

Datagram transport is the subject of a separate draft, "A Datagram Transport for the RTC-Web"

profile".[I-D.alvestrand-dispatch-rtcweb-datagram] The basic approach is to use ICE as a setup mechanism, and to specify mechanisms to use ICE over connections that utilize UDP and TCP if needed to support a basic datagram-passing function with adequate security. In order to deal with complex NAT/firewall situations, relaying using TURN MUST be supported.

For octet-stream transport, TCP is used. (QUESTION: Do we need a TCP relay specification? The use of TURN over TCP and TLS is specified in the TURN RFC - is it suitable?)

(The role of Web Sockets [I-D.ietf-hybi-thewebsocketprotocol] needs to be clarified.)

The data transport MUST behave reasonably in the presence of congested networks; this is usually interpreted as reducing the send rate when congestion is encountered. TCP, when correctly implemented, does this automatically; this is not the case with UDP, and the RTP framing specification does not contain a congestion control component.

Determining an useful congestion handling mechanism is a high priority for work with this specification suite.

5. Data framing and securing

RTP [RFC3550]and SRTP [RFC3711]. The RTP/SAVP profile, defined as part of SRTP, is supported, and "extended RTCP", RTP/SAVPF [RFC4585], with its secured version RTP/SAVPF [RFC5124]is used in order to support codec functionality that depends on this RTP profile, such as

The implementation of SRTP used MUST support encryption using AES-CM with MIC, on both RTP and RTCP channels. <TODO: Add pointer to appropriate profile here> (Note that like for all mandatory-to-implement, there is no requirement that these protocols be used, just that it is possible to negotiate them.)

[OPEN ISSUE; We need to specify a securable format of passing data that is not RTP. This should probably be a profile on using TLS and/or DTLS, although specifying a "data codec" and using SRTP has been proposed too.]

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.

NOTE IN DRAFT: The particular codecs named are NOT A DECISION. They are included to illustrate possible choices, and to check with the group that the references given are necessary and sufficient for the purpose of specifying an interoperable codec suite.

In audio, the OPUS codec [I-D.ietf-codec-opus] MUST be supported. For ease of interoperability with gateways to older equipment, G.711 U-law, audio/PCMU, defined in RFC 1890 [RFC1890] section 4.4.12, is also mandatory to implement. There is no third mandatory to implement.

In video, the VP8 codec [I-D.westin-payload-vp8] MUST be supported.

The Theora codec is also freely available. H.264/AVC and H.264/SVC [I-D.ietf-avt-rtp-svc] are widely enough used that it gives a wider range of communications partners if they are supported.

7. Connection management

This specification is silent on the definition of connection management protocols. It envisions that implementors will make a choice on whether to implement connection management protocols as a downloadable component, as a browser plug-in, or as a frontend/backend split, where a part of the protocol machinery is downloaded into the browser and uses some mechanism (for instance WebSockets) to communicate back to a backend implementing the rest of the connection management protocol.

XMPP, and its Jingle component, has proved a versatile tool in building interoperable communities, and so has SIP. This suite requires that the browser support establishing and describing connections using a data format capable of representing the information needed by these two protocols, such as one that can be one-to-one transformed into SDP. The exact specification of this API is done elsewhere <insert reference when available>; this API is powerful enough that all interesting parameters of the transport mechanisms specified above are settable, and clear enough that how to connect the API to the protocols is obvious.

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 in <insert reference here when available>.

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:

- o Echo cancellation should be good enough that feedback (defined as a rising volume of sound with no local sound input) does not occur.
- 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 to 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 <whatever dB metrics makes sense here - most important that we have one only>

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).

This specification addresses some, but not all, of these concerns, and makes some assumptions about the security considerations of other parts of the environment; it is up to the implementor to see that these security assumptions are warranted. In particular:

- o We assume that the ICE security mechanism is a necessary and sufficient criterion for accepting that a connection attempt is from a communications partner. This means that we trust the randomness of ICE "usernames" and the security of ICE "passwords".
- o We assume that the SRTP key exchange mechanisms and security profiles specified provide an adequate level of protection for audio and video media.

(there needs to be more text here)

12. Acknowledgements

13. References

13.1. Normative References

- [I-D.alvestrand-dispatch-rtcweb-datagram]
Alvestrand, H., "A Datagram Transport for the RTC-Web profile", draft-alvestrand-dispatch-rtcweb-datagram-01 (work in progress), February 2011.
- [I-D.ietf-codec-opus]
Valin, J. and K. Vos, "Definition of the Opus Audio Codec", draft-ietf-codec-opus-04 (work in progress), March 2011.
- [I-D.ietf-hybi-thewebsocketprotocol]
Fette, I., "The WebSocket protocol", draft-ietf-hybi-thewebsocketprotocol-06 (work in progress), February 2011.
- [I-D.westin-payload-vp8]
Westin, P. and H. Lundin, "Proposal for the IETF on "RTP Payload Format for VP8 Video"", draft-westin-payload-vp8-02 (work in progress), March 2011.
- [RFC1890] Schulzrinne, H., "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 1890, January 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

13.2. Informative References

- [I-D.ietf-avt-rtp-svc]
Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis,
"RTP Payload Format for Scalable Video Coding",
draft-ietf-avt-rtp-svc-27 (work in progress),
February 2011.
- [RFC3935] Alvestrand, H., "A Mission Statement for the IETF",
BCP 95, RFC 3935, October 2004.

Author's Address

Harald T. Alvestrand
Google
Kungsbron 2
Stockholm, 11122
Sweden

Email: harald@alvestrand.no

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2012

C. Bran
Plantronics
C. Jennings
Cisco
JM. Valin
Mozilla
March 12, 2012

WebRTC Codec and Media Processing Requirements
draft-cbran-rtcweb-codec-02

Abstract

This document outlines the codec and media processing requirements for WebRTC client application and endpoint devices.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 September 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Codec Requirements	3
3.1. Audio Codec Requirements	3
3.2. Video Codec Requirements	3
4. Audio Level	4
5. Acoustic Echo Cancellation (AEC)	5
6. Legacy VoIP Interoperability	6
7. IANA Considerations	6
8. Security Considerations	6
9. Acknowledgements	6
10. Normative References	6
Authors' Addresses	7

1. Introduction

An integral part of the success and adoption of the Web Real Time Communications (WebRTC) will be the voice and video interoperability between WebRTC applications. This specification will outline the media processing and codec requirements for WebRTC client implementations.

2. 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 RFC 2119 [RFC2119].

3. Codec Requirements

This section covers the audio and video codec requirements for WebRTC client applications. To ensure a baseline level of interoperability between WebRTC clients, a minimum set of required codecs are specified below. While this section specifies the codecs that will be mandated for all WebRTC client implementations, it leaves the question of supporting additional codecs to the will of the implementer.

3.1. Audio Codec Requirements

WebRTC clients are REQUIRED to implement the following audio codecs.

- o PCMA/PCMU - 1 channel with a rate of 8000 Hz and a ptime of 20 - see section 4.5.14 of [RFC3551]
- o Telephone Event - [RFC4734]
- o Opus [draft-ietf-codec-opus]

For all cases where the client is able to process audio at a sampling rate higher than 8 kHz, it is RECOMMENDED that Opus be offered before PCMA/PCMU. For Opus, all modes MUST be supported, for all ptime values up to 120 ms. Clients MAY use the offer/answer mechanism to signal a preference for a particular mode or ptime.

3.2. Video Codec Requirements

The following feature list applies to all required video codecs.

Required video codecs:

- o MUST support at least 10 frames per second (fps) and SHOULD support 30 fps
- o If VP8 is supported, then it MUST support the bilinear and none reconstruction filters
- o OPTIONALLY offer support for additional color spaces
- o MUST support a minimum resolution of 320X240
- o SHOULD support resolutions of 1280x720, 720x480, 1024x768, 800x600, 640x480, 640 x 360 , 320x240

4. Audio Level

It is desirable to standardize the "on the wire" audio level for speech transmission to avoid users having to manually adjust the playback and to facilitate mixing in conferencing applications. It is also desirable to be consistent with ITU-T recommendations G.169 and G.115, which recommend an active audio level of -19 dBm0. However, unlike G.169 and G.115, the audio for WebRTC is not constrained to have a passband specified by G.712 and can in fact be sampled at any sampling rate from 8 kHz to 48 kHz and up. For this reason, the level SHOULD be normalized by only considering frequencies above 300 Hz, regardless of the sampling rate used. The level SHOULD also be adapted to avoid clipping, either by lowering the gain to a level below -19 dBm0, or through the use of a compressor.

AUTHORS' NOTE: The idea of using the same level as what the ITU-T recommends is that it should improve inter-operability while at the same time maintaining sufficient dynamic range and reducing the risk of clipping. The main drawbacks are that the resulting level is about 12 dB lower than typical "commercial music" levels and it leaves room for ill-behaved clients to be much louder than a normal client. While using music-type levels is not really an option (it would require using the same compressor-limitors that studios use), it would be possible to have a level slightly higher (e.g. 3 dB) than what is recommended above without causing interoperability problems.

Assuming 16-bit PCM with a value of +/-32767, -19 dBm0 corresponds to a root mean square (RMS) level of 2600. Only active speech should be considered in the RMS calculation. If the client has control over the entire audio capture path, as is typically the case for a regular phone, then it is RECOMMENDED that the gain be adjusted in such a way that active speech have a level of 2600 (-19 dBm0) for an average speaker. If the client does not have control over the entire audio

capture, as is typically the case for a software client, then the client SHOULD use automatic gain control (AGC) to dynamically adjust the level to 2600 (-19 dBm0) +/- 6 dB. For music or desktop sharing applications, the level SHOULD NOT be automatically adjusted and the client SHOULD allow the user to set the gain manually.

The RECOMMENDED filter for normalizing the signal energy is a second-order Butterworth filter with a 300 Hz cutoff frequency.

It is common for the audio output on some devices to be "calibrated" for playing back pre-recorded "commercial" music, which is typically around 12 dB louder than the level recommended in this section. Because of this, clients MAY increase the gain before playback.

5. Acoustic Echo Cancellation (AEC)

It is plausible that the dominant near to mid-term WebRTC usage model will be people using the interactive audio and video capabilities to communicate with each other via web browsers running on a notebook computer that has built-in microphone and speakers. The notebook-as-communication-device paradigm presents challenging echo cancellation problems, the specific remedy of which will not be mandated here. However, while no specific algorithm or standard will be required by WebRTC compatible clients, echo cancellation will improve the user experience and should be implemented by the endpoint device.

SHOULD include an AEC and if not, SHOULD ensure that the speaker-to-microphone gain is below unity at all frequencies to avoid instability when none of the client has echo cancellation. For clients that do not control the audio capture and playback devices directly, it is RECOMMENDED to support echo cancellation between devices running at slight different sampling rates, such as when a webcam is used for microphone.

The client SHOULD allow either the entire AEC or the non-linear processing (NLP) to be turned off for applications, such as music, that do not behave well with the spectral attenuation methods typically used in NLPs. It SHOULD have the ability to detect the presence of a headset and disable echo cancellation.

For some applications where the remote client may not have an echo canceller, the local client MAY include a far-end echo canceller, but if that is the case, it SHOULD be disabled by default.

Call control event notification to connected devices such as headsets (what's that exactly?)

6. Legacy VoIP Interoperability

The codec requirements above will ensure, at a minimum, voice interoperability capabilities between WebRTC client applications and legacy phone systems.

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

The codec requirements have no additional security considerations other than those captured in [I-D.ekr-security-considerations-for-rtc-web].

9. Acknowledgements

This draft incorporates ideas and text from various other drafts. In particular we would like to acknowledge, and say thanks for, work we incorporated from Harald Alvestrand.

10. Normative References

- [I-D.ekr-security-considerations-for-rtc-web]
Rescorla, E., "Security Considerations for RTC-Web",
May 2011.
- [I-D.webm]
Google, Inc., "VP8 Data Format and Decoding Guide",
July 2010.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and
Video Conferences with Minimal Control", STD 65, RFC 3551,
July 2003.
- [RFC4734] Schulzrinne, H. and T. Taylor, "Definition of Events for
Modem, Fax, and Text Telephony Signals", RFC 4734,

December 2006.

Authors' Addresses

Cary Bran
Plantronics
345 Encinial Street
Santa Cruz, CA 95060
USA

Phone: +1 206 661-2398
Email: cary.bran@plantronics.com

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Jean-Marc Valin
Mozilla
650 Castro Street
Mountain View, CA 94041
USA

Email: jmvalin@jmvalin.ca

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2012

C. Bran
C. Jennings
Cisco
July 4, 2011

RTC-Web Non-Media Data Transport Requirements
draft-cbran-rtcweb-data-00

Abstract

This document outlines a protocol and requirements for RTC-Web client application to transmit real-time, non-media data.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 January 5, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Non-Media Data Transport Requirements	3
4. Solution Overview	4
5. Specification	4
6. IANA Considerations	4
7. Security Considerations	4
8. Acknowledgements	5
9. Normative References	5
Authors' Addresses	5

1. Introduction

This specification will focus on the transport of real-time non-media data requirements for RTC-Web client applications. An example of real-time non-media data, would be a characters screen position within an multiplayer HTML5 video game.

The non-media data transport requirements fit into a series of specifications have been created to address RTC-Web negotiation and signaling protocols, security requirements, media transmission and use cases. More information on the RTC-Web can be found here:

[TODO put links to supporting drafts here]

2. 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 RFC 2119 [RFC2119].

3. Non-Media Data Transport Requirements

The RTC-WEB will enable for rich voice and video communications from client applications, such as a web browser. One of the natural extensions of the RTC-WEB and the work emerging from the HTML5 community is video games. Video games have a similar stringent real-time requirement for exchanging non-media data types such as a player's screen position.

The question of how best to handle non-media data types has been raised. There have been proposals to address this problem. Common to all proposals is how the data transport session is set up, using ICE [RFC5245] in a similar manner to that of RTP [RFC3550]. The proposals vary from once the session is set up; one proposal is just to use a thin shim on top of UDP or DTLS to de-multiplex the packets from other packets such as RTP on the same connection. Another proposal is DTLS over DCCP over UDP with some appropriate congestion control scheme chosen for DCCP. Lastly there has been a proposal to define a data codec to carry the data in RTP.

Of all the solutions proposed to date there have been issues with implementation maturity and availability, congestion control, high overhead and NAT traversal. The solution outlined within this specification proposes a lightweight, simple to implement approach for RTC-Web client applications to transmit real-time non-media data.

4. Solution Overview

Each application datagram is sent with a single byte header to help with de-multiplexing issues. The combined datagram and header are sent either over UDP or DTLS to the receiver. The receiver sends an acknowledgment for every packet it receives. The sender computes a packet loss rate based upon the number of packets sent, and number of acknowledgements it has received inside of given time window. The browser, or other RTC-Web client application implementation, then enforces a maximum bandwidth usage using the TFRC-SP[RFC4828].

Practically this can be implemented with a simple lookup table such as Table 1 in [RFC4828] that limits the data rate. A JavaScript application running in the browser can implement more complex fragmentation, reliability, and congestion control algorithms, but it is the browser, or other RTC-Web client application, that is responsible for enforcing the basic congestion safety with the TFRC-SP algorithm.

5. Specification

When sending a datagram, a single byte with the value 62 MUST be prepended to the data to be sent. The data is then sent over the UDP or a DTLS flow that has been set up by ICE. The receiver MUST send an acknowledgment for each packets it receives. This acknowledgment is a UDP or DTLS datagram containing a single byte with the value of 63. The packet loss rate is computed by comparing the number of packet sent to the of acknowledgements received within the past 2 seconds. The packet loss rate and amount of data sent is used with the TFRC-SP algorithm to compute a maximum safe bandwidth. The sender MUST not exceed this bandwidth. If an application requests the browser, or other RTC-Web client application, to send a packet that would exceed the bandwidth, the RTC-Web client application MUST signal an error to the requesting application and drop the packet.

6. IANA Considerations

This document makes no request of IANA.

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

7. Security Considerations

Because there are a number of security issues, considerations and

requirements for RTC-WEB client applications there is a draft that specifically addresses the RTC-WEB application security considerations. This draft defers it's security considerations and requirements to the security considerations for RTC-Web draft [I-D.ekr-security-considerations-for-rtc-web].

8. Acknowledgements

You too can see your name here, just send us comments.

9. Normative References

- [I-D.ekr-security-considerations-for-rtc-web]
Rescorla, E., "Security Considerations for RTC-Web",
May 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
Jacobson, "RTP: A Transport Protocol for Real-Time
Applications", STD 64, RFC 3550, July 2003.
- [RFC4828] Floyd, S. and E. Kohler, "TCP Friendly Rate Control
(TFRC): The Small-Packet (SP) Variant", RFC 4828,
April 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment
(ICE): A Protocol for Network Address Translator (NAT)
Traversal for Offer/Answer Protocols", RFC 5245,
April 2010.

Authors' Addresses

Cary Bran
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 206 256-3502
Email: cbran@cisco.com

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2011

C. Bran
C. Jennings
Cisco
June 29, 2011

RTC-Web Media Transport Requirements
draft-cbran-rtcweb-media-00

Abstract

This document outlines the media transport protocols and requirements for RTC-Web client applications.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 31, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Real-Time Media Transport Requirements	3
3.1. RTP Profile	3
3.1.1. Profile Encryption Mechanism	4
3.2. RTP Optimizations	4
3.2.1. RTP and RTCP Multiplexing	4
3.2.2. Reduced-Size RTCP	4
3.2.3. Symmetric RTP/RTCP	5
3.2.4. CNAME Generation	5
3.3. RTP Extensions	5
3.3.1. Conferencing Extensions	6
3.3.2. Header Extensions	6
3.4. RTP Transport Robustness	7
3.4.1. RTP Retransmission	7
3.4.2. Forward Error Correction	8
3.4.3. Multicast	8
3.5. RTP Rate Control	8
4. Legacy VoIP Interoperability	8
5. IANA Considerations	8
6. Security Considerations	8
7. Acknowledgements	9
8. Normative References	9
Authors' Addresses	10

1. Introduction

An integral part of the success and adoption of the Real-Time Communications Web (RTC-WEB) will be the interoperability between RTC-Web applications. This specification will focus on the media transport requirements for RTC-Web client applications.

The media transport requirements fit into a series of specifications have been created to address RTC-Web negotiation and signaling protocols, security requirements, non-media data transmission and use cases. More information on the RTC-Web can be found here:

[TODO put links to supporting drafts here]

2. 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 RFC 2119 [RFC2119].

3. Real-Time Media Transport Requirements

This section defines the real-time media transport requirements for RTC-Web client application implementation. This section breaks down the RTC-WEB RTP requirements into several sections. The sections cover the RTP requirements for: profile, optimizations, extensions, transport robustness and rate control.

[OPEN ISSUE: identify missing requirements]

3.1. RTP Profile

RTC-Web applications will need to provide a secure, interoperable, bandwidth friendly, media transport profile. The Secure Audio-visual Profile Feedback (SAVPF) as defined in [RFC5124] will meet the needs of RTC-Web applications by providing media encryption, interoperability and a flexible, bandwidth conscious RTCP packet transmission model. All RTC-Web applications are REQUIRED to implement SAVPF. Requiring the implementation of SAVPF also means that RTC-Web applications MUST implicitly support Audio-visual Profile Feedback (AVPF) [RFC4585], Audio-visual Profile (AVP) [RFC3551] and Secure Audio-visual Profile (SAVP) [RFC3711].

3.1.1. Profile Encryption Mechanism

SAVPF supports SRTP by providing media encryption, integrity protection, replay protection and a limited form of source authentication. Though the SAVPF profile does support secure media transport, it does not specify an encryption keying mechanism. To support keying for SRTP, WEB-RTC application implementors are REQUIRED to implement DTLS-SRTP [RFC5764].

3.2. RTP Optimizations

This section describes the optimization requirements for RTP within RTC-Web applications.

3.2.1. RTP and RTCP Multiplexing

Historically, RTP and RTCP have been run on separate UDP ports. With the increased use of Network Address Port Translation (NAPT) so have the problems increased for maintaining multiple, costly NAT bindings for each UDP port. This dual UDP port paradigm also complicates firewall administration, since multiple ports must be opened to allow for RTP traffic. To reduce these costs and session setup times, support for multiplexing multiple RTP streams on a single UDP port [I-D.rosenberg-jennings-rtp-mux] is REQUIRED.

Note that the use of RTP and RTCP multiplexed on a single port ensures that there is occasional traffic sent on that port, even if there is no active media traffic. This may be useful to keep-alive NAT bindings.

3.2.2. Reduced-Size RTCP

RTCP packets are usually sent as compound RTCP packets and [RFC3550] demands that the RTCP compound packets always start with a Sender Report (SR) or Receiver Report (RR) packet. The SR and RR packets provide reception quality statistics and increase the mean RTCP packet size. Because the mean compound RTCP packet size is larger, the frequency at which RTCP packets can be sent within the RTCP bandwidth share decreases. The decreased transmission frequency creates a performance bottleneck that is especially noticeable when using frequent feedback messages.

As mentioned in section [Add ref] RTC-Web applications will be required to implement SAVPF, which implicitly requires feedback. [RFC5506] specifies how to reduce the mean RTCP message and allow for more frequent feedback. Frequent feedback, in turn, is essential to make real-time application quickly aware of changing network conditions and allow them to adapt their transmission and encoding

behavior. Support for [RFC5506] is REQUIRED

3.2.3. Symmetric RTP/RTCP

RTP entities choose the RTP and RTCP transport addresses (IP addresses and port numbers), to bind to and receive packets on. However when sending RTP and RTCP packets, senders may use an IP address or port number that is different than the one specified for receiving packets. Using different transport addresses is problematic with regards to NAT traversal. The NAT traversal problem can be alleviated using symmetric RTP/RTCP [RFC4961]. Symmetric RTP/RTCP requires that the transport addresses for sending and receiving RTP/RTCP packets are identical. All RTC-WEB client applications are REQUIRED to implement Symmetric RTP/RTCP [RFC4961].

3.2.4. CNAME Generation

The RTCP Canonical Name (CNAME) provides a persistent transport-level identifier for an RTP endpoint. While the Synchronization Source (SSRC) identifier for an RTP endpoint may change if a collision is detected, or when the RTP application is restarted, it's RTCP CNAME is meant to stay unchanged, so that RTP endpoints can be uniquely identified and associated with their RTP media streams. For proper functionality, RTCP CNAMEs should be unique within the participants of an RTP session.

The RTP specification [RFC3550] includes guidelines for choosing a unique RTP CNAME. These guidelines are not sufficient in the presence of NAT devices or with regards to addressing privacy concerns resulting from the long-term, persistent identifiers.

To address the shortcomings of CNAME selection in [RFC3550], it is RECOMMENDED that RTP CNAME generation follows the approach specified in section 5 of [RFC6222].

For RTC-WEB client applications, such as a web browser, it may not be possible to retrieve the EUI-64 identifier or the host system's MAC address which is needed to fulfill the CNAME generation procedure outlined in section 5 of [RFC6222]. As an alternative to the EUI-64/MAC address, RTC-WEB client applications MAY generate and use a random number for the unique CNAME generation procedure.

3.3. RTP Extensions

This section describes the RTP extensions that could be very useful within the RTC-WEB context.

3.3.1. Conferencing Extensions

RTC-Web applications will support conferencing capabilities. While this document remains silent regarding what conferencing topology should be supported for RTC-Web applications, the following section will provide guidance around RTP extensions to support centralized conferencing.

For more information on RTP conferencing topologies please refer to [RFC5117]

3.3.1.1. FIR RTCP Feedback Message

The Full Intra Request (FIR) command and message are defined in sections 3.5.1 and 4.3.1 of [RFC5104]. FIR messages will request that the currently distributed session participants send new intra coded pictures to the mixer. FIR is used when switching between sources to ensure that the receivers can decode the video or other predicted media encoding with long prediction chains. It is RECOMMENDED that the FIR message is supported.

3.3.1.2. PLI RTCP Feedback Message

The Picture Loss Indicator (PLI) is defined in Section 6.3.1 of [RFC4585]. PLI messages tell the encoder that a receiver has lost the decoder context and would like it repaired. It is RECOMMENDED that the PLI message is supported.

3.3.1.3. TMMBR RTCP Feedback Message

The Temporary Maximum Media Stream Bit Rate Request (TMMBR, "timber") message is defined in sections 3.5.4 and 4.2.1 of [RFC5104]. A receiver, translator, or mixer uses the TMMBR to request a sender to limit the maximum bit rate for a media stream to, or below, the provided value. An example of using TMMBR would be for an RTP mixer to constrain the media sender's bit rate to fit within the lower bit rate range of other session participants. It is RECOMMENDED that the TMMBR message be supported.

3.3.2. Header Extensions

This section describes the requirements for RTC-WEB RTP header extensions. For all RTC-WEB RTP header extensions it is REQUIRED that they are formatted and signaled according to the general mechanism defined in [RFC5285].

[Open Issue: should any of the following headers be added to the list:

- o Transmission time offsets[RFC5450]
- o Associating time-codes with RTP streams [RFC5484] [Remove?]

3.3.2.1. Rapid Synchronization

Basic RTP session synchronization as described in [RFC3550] can be slow. To improve synchronization performance and maintain relative backwards compatibility it is RECOMMENDED that the rapid RTP synchronization extensions described in [RFC6051] be implemented.

3.3.2.2. Audio Levels

These RTP header extensions provide a mechanism to indicate the audio level within the same RTP packets as the audio data they pertain to.

In large conferences, when clients send audio levels of the audio sample contained within the RTP packet to the mixer, it can reduce the load on the audio mixer as resources for decoding and measuring audio streams are not needed. Because of the performance gains at scale, it is RECOMMENDED that the extension described in [I-D.ietf-avtext-client-to-mixer-audio-level] be implemented.

Clients can also optimize performance if the RTP packets sent from the mixer contain the audio levels. It is OPTIONAL for mixers to implement the extension described in [I-D.ietf-avtext-mixer-to-client-audio-level].

3.4. RTP Transport Robustness

This section identifies tools that can be used to add robustness to the RTP flows. Adding robustness to the RTP flow can reduce packet loss and thus have a positive impact upon media quality.

3.4.1. RTP Retransmission

The retransmission scheme in RTP allows for flexibility of retransmissions. From the receiving side, only selected missing packets can be requested. From the sending side, packets can be prioritized based upon the senders knowledge of the receiver's missing packets. It has been proposed that RTC-WEB applications could use the RTP retransmission as defined by [RFC4588], this retransmission scheme is problematic for RTC-Web applications on two fronts. The first problem area is the additional latency added by [RFC4588] will exceed the latency threshold for interactive voice and video. The second issue is involves the undesirable increase in packet transmission at the point when congestion occurs. Until the two issues are addressed, implementing [RFC4588] for RTC-Web

applications is NOT RECOMMENDED.

3.4.2. Forward Error Correction

[Open issue - should there be a FEC scheme recommendation?]

3.4.3. Multicast

RTC-WEB client applications support for multicast RTP is NOT REQUIRED.

3.5. RTP Rate Control

[OPEN ISSUE - There are currently no available, standardized RTP rate control mechanism that uses media adaptation. Having a mechanism in place will be REQUIRED for RTC-WEB applications and which means there is a need for the IETF to produce this specification.

A potential starting point for defining a solution is "RTP with TCP Friendly Rate Control" [rtp-tfrc].]

4. Legacy VoIP Interoperability

The use of RTP as specified above will maximize the interoperability capabilities between RTC-Web client applications and legacy VoIP systems.

5. IANA Considerations

This document makes no request of IANA.

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

6. Security Considerations

Because there are a number of security issues, considerations and requirements for RTC-WEB client applications there is a draft that specifically addresses the RTC-WEB application security considerations. This draft defers it's security considerations and requirements to the security considerations for RTC-Web draft [I-D.ekr-security-considerations-for-rtc-web].

7. Acknowledgements

This draft incorporates ideas and text from various other drafts. In particular we would like to acknowledge, and say thanks for, work we incorporated from Harald Alvestrand, Magnus Westerlund, Colin Perkins, and Joerg Ott.

8. Normative References

- [I-D.ekr-security-considerations-for-rtc-web]
Rescorla, E., "Security Considerations for RTC-Web",
May 2011.
- [I-D.ietf-avtext-client-to-mixer-audio-level]
Lennox, J., Ivov, E., and E. Marocco, "A Real-Time
Transport Protocol (RTP) Header Extension for Client-to-
Mixer Audio Level Indication", March 2011.
- [I-D.ietf-avtext-mixer-to-client-audio-level]
Ivov, E., Marocco, E., and J. Lennox, "A Real-Time
Transport Protocol (RTP) Header Extension for Mixer-to-
Client Audio Level Indication", May 2011.
- [I-D.rosenberg-jennings-rtp-mux]
Rosenberg, J. and C. Jennings, "Multiplexing of Real-Time
Transport Protocol (RTP) Traffic for Browser based Real-
Time Communications (RTC)", June 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
Jacobson, "RTP: A Transport Protocol for Real-Time
Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and
Video Conferences with Minimal Control", STD 65, RFC 3551,
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.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey,
"Extended RTP Profile for Real-time Transport Control
Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585,
July 2006.

- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, July 2007.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.
- [RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, March 2009.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [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, May 2010.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

Authors' Addresses

Cary Bran
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 206 256-3502
Email: cbran@cisco.com

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2012

C. Bran
Plantronics
M. Kaufman
Skype
C. Jennings
Cisco
J. Rosenberg
Skype
October 29, 2011

WebRTC Network Address Translation
draft-cbran-rtcweb-nat-02

Abstract

This document outlines the network address translation (NAT) traversal requirements and for WebRTC client applications.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 May 1, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Connection Management Requirements	3
3.1. NAT Traversal Requirements	3
3.2. Data Transmission Requirements	3
3.3. IPv4 to IPv6 Transition Requirements	4
3.4. Legacy Phone System Interoperability Requirements	4
4. Connection Management Mechanism	4
4.1. ICE	4
4.1.1. ICE as a Consent Mechanism	4
4.2. Web Browsers and ICE	5
4.2.1. Native ICE Support	5
4.2.2. STUN Configuration	5
5. IANA Considerations	6
6. Security Considerations	6
7. Acknowledgements	6
8. Normative References	6
Authors' Addresses	7

1. Introduction

An integral part of the of the Web Real Time Communications (WebRTC) will be the ability for client application implementations to have native, secure Network Address Translation (NAT) [RFC1631] traversal capabilities. This document provides requirements and implementation specifications WebRTC client NAT traversal.

2. 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 RFC 2119 [RFC2119].

3. Connection Management Requirements

This section identifies the requirements for RTC-Web client applications to connection requirements.

3.1. NAT Traversal Requirements

A majority of WebRTC clients will be web browsers and used behind a NAT and or firewall. WebRTC clients will use a UDP-based data transmission scheme for multimedia sessions [Open Issue: what draft should be cited for this requirement?]. UDP has well know NAT traversal problems and without native capabilities to traverse a NAT, WebRTC clients will be extremely limited in their functionality. Fortunately NAT traversal for UDP is a solved problem, but solutions require that clients transmitting media between each other need to use the same NAT traversal algorithms. Without a consistent, well specified NAT traversal mechanism WebRTC client implementations would likely be inoperable with each other. To address the identified problems WebRTC clients are REQUIRED to implement the NAT traversal mechanism as defined in Section 4.

3.2. Data Transmission Requirements

Whenever a calling WebRTC client attempts to establish a connection, the receiving WebRTC client MUST provide consent before the calling client can transmit data to the receiver. Providing consent on the receiving end before data transmission commence is needed to help to prevent malicious attacks by the calling client. All WebRTC clients are REQUIRED to implement connection management that provides a consent mechanism for media transmission. Furthermore it is REQUIRED that consent be given by the recipient before an WebRTC client can transmit media.

As a note providing consent to open a media connection does not involve user-level consent, rather it is the responsibility of the WebRTC client application (e.g. web browser) to enforce this requirement.

3.3. IPv4 to IPv6 Transition Requirements

RTC-Web clients MUST support IPv4 to IPv6 transition.

3.4. Legacy Phone System Interoperability Requirements

There is no way to meet all the connection management requirements and maintain compatibility with all legacy phone systems. It is highly desirable that the WebRTC connection management mechanism be interoperable with legacy phone systems such as a VOIP endpoints, PSTN gateways and SIP trunks.

4. Connection Management Mechanism

This section specifies the connection management system that will address the identified requirements.

4.1. ICE

To address the NAT traversal, data transmission, and interoperability requirements all WebRTC client applications are REQUIRED to implement ICE [RFC5245]. Implicit to ICE, and listed here for clarity, WebRTC client implementations will are also REQUIRED to implement STUN [RFC3489] and TURN [RFC5766].

Additional ICE requirements:

- o Support of ICE's Aggressive Nomination is REQUIRED
- o Support of ICE's Regular Nomination is OPTIONAL
- o WebRTC media gateways MAY implement ICE-Lite instead of full ICE

4.1.1. ICE as a Consent Mechanism

Of the connection management requirements listed above, the least obvious is how ICE will satisfy being a consent mechanism for data transmission Section 3.2. The reason ICE can satisfy this requirement is due to its reliance on STUN transactions to succeed in order to establish a connection. The success of a STUN transaction can be viewed as semantically the same thing as a recipient providing consent to transmit data. Conversely the failure of the STUN

transaction would semantically map to the recipient rejecting the request to transmit data.

4.2. Web Browsers and ICE

This section specifies the web browser implementation requirements for WebRTC client connection management.

4.2.1. Native ICE Support

To meet the WebRTC connectivity requirements, web browser vendors MUST natively support ICE [RFC5245]. Access to the web browser's ICE implementation will be defined in the W3C WebRTC-API specification[I.D.w3c-webrtc].

Alternate proposals have been made that advocate for natively exposing STUN[RFC3489] APIs in the web browser. The ICE implementation would be realized via a JavaScript library that uses the browser's native STUN API. After reviewing the alternate proposals the solution several issues were identified.

1. JavaScript running within "real world" web applications cannot reliably handle the ICE timing and pacing requirements. An example of this is long running JavaScript code from embedded advertisers. A big JavaScript file can take a significant amount of time to execute and can prevent web application timers from firing in correctly. Given the pacing requirements for ICE are in the 20ms range, it is highly likely that ICE will break if it is implemented in a JavaScript library.
2. Multiple implementations of a JavaScript ICE library is a logistical nightmare. Coordinating updates, bug fixes, enhancements and a testing matrix for interoperability at Internet scale will simply be impossible.

4.2.2. STUN Configuration

Web browsers MUST provide a mechanism to configure access to a STUN server.

Below are some proposed mechanisms by which the STUN server could be configured:

- o A preference page, similar to the what web browser's use for configuring web proxy settings
- o Exposed as a JavaScript API and added to the W3C WebRTC-API specification

Regardless of the mechanism adopted by the web browser vendor, the following configuration data is REQUIRED to be exposed and settable through the web browsers configuration mechanism:

- o STUN Server Address - the IPv4 or IPv6 address of the STUN server
- o STUN Server Port
- o Credentials to access the STUN server (these are not STUN generated credentials)

5. IANA Considerations

This document makes no request of IANA.

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

6. Security Considerations

To guard against spoofing RTC-Web client applications are REQUIRED to:

- o Internally encapsulate the generation of STUN transaction IDs
- o Block read/write access to the generated STUN transaction IDs

7. Acknowledgements

This draft incorporates ideas and text from the IETF mailing list. In particular we would like to acknowledge, and say thanks for, work we incorporated from Timothy Terriberry and Christopher Blizzard.

8. Normative References

[I.D.w3c-webrtc]

Bergkvist, A., Burnett, D., Jennings, C., and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers", October 2011.

[RFC1631] Egevang, K. and P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- [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.

Authors' Addresses

Cary Bran
Plantronics
345 Encinal Street
Santa Cruz, CA 95060
USA

Phone: +1 206 661-2398
Email: cary.bran@plantronics.com

Matthew Kaufman
Skype
3210 Porter Drive
Palo Alto, California 94304
US

Phone: +1 831 440 8771
Email: matthew.kaufman@skype.net

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Jonathan Rosenberg
Skype
3210 Porter Drive
Palo Alto, California 94304
US

Email: jdrosen@skype.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2012

C. Bran
C. Jennings
Cisco
July 4, 2011

RTC-Web Negotiation and Signaling
draft-cbran-rtcweb-negotiation-00

Abstract

This document outlines the negotiation and signaling protocols for RTC-Web client application implementation.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 January 5, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Introduction	3
4. Negotiation Requirements	3
4.1. ICE	4
5. Signaling Protocol Requirements	4
5.1. Client Application SIP Requirements	4
5.2. Client Application Optional SIP Support	5
5.3. Required SIP Methods	5
5.4. Multipart SIP Message Requirements	6
5.5. Identity Requirements	6
5.6. Network Address Traversal	6
6. Legacy VoIP Interoperability	6
7. IANA Considerations	7
8. Security Considerations	7
9. Acknowledgements	7
10. Normative References	7
Authors' Addresses	8

1. Introduction

An integral part of the success and adoption of the Real-Time Communications Web (RTC-WEB) will be the interoperability between RTC-Web applications running on different browsers or different versions of a browser. As browser features evolve, new codecs are deployed, and more advanced features are added, it is critical to have a negotiation framework between browsers to facilitate evolution of real time communications on the web. It is also important for negotiation in a backwards compatible way with legacy VoIP equipment. This specification proposes negotiation and signaling requirements for enabling broad interoperability capabilities for RTC-Web client applications.

The negotiation and signaling requirements fit into a series of specifications have been created to address RTC-Web codec, NAT traversal, security, data transmission and use case requirements. More information on the RTC-Web can be found here:

[TODO put links to supporting drafts here]

2. 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 RFC 2119 [RFC2119].

3. Introduction

This proposal supports an architecture where the negotiations happen directly between two browsers (or other RTC-Web client applications) or a model where the browser routes the negotiation via a third server that helps facilitate the negotiation. In the first model where communications are direct, it is assumed that ICE has already been used to set up a communication channel between the browsers that can be used for negotiation.

While SIP is used in this proposal, it is a restricted subset of the SIP functionally for initiating and setting up RTP streams and rendezvous services for these messages.

4. Negotiation Requirements

This section details the secure channel negotiation requirements for RTC-Web client applications. The requirements below originate from

the RTC-Web NAT draft [I-D.cbran-jennings-rtc-web-nat].

4.1. ICE

It is plausible that many RTC-WEB client applications, such as web browsers will be deployed behind a NAT. To set up secure data plane sessions, and address the security requirements identified in Section 8 all RTC-WEB client application implementations are REQUIRED to natively expose an implementation of either ICE [RFC5245] or ICE-Lite (Section 2.7 of [RFC5245]). Implicit to implementing ICE, all RTC-WEB client applications are REQUIRED to implement Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs) (STUN) [RFC3489] and Traversal Using Relays around NAT (TURN) [RFC5766].

Echoing from the RTC-Web NAT draft [I-D.cbran-jennings-rtc-web-nat], ICE is REQUIRED be implemented and available natively from the RTC-Web client application. What this means via example is; if the RTC-Web client application happens to be a web browser, the web browser MUST implement ICE such that adheres to the RTC-Web NAT draft [I-D.cbran-jennings-rtc-web-nat].

5. Signaling Protocol Requirements

This section covers the signaling protocol to be used by RTC-WEB applications. To ensure interoperability not just between RTC-WEB applications, but with legacy PBX phone systems as well, a small subset of SIP will be REQUIRED for all RTC-WEB client application implementations. In addition to the subset of SIP specification [RFC3261], RTC-WEB client application implementations will be REQUIRED to support DNS resolutions as specified in [RFC3263] and the offer/answer model with SDP as specified in [RFC3264].

Because the browsers only need to use a small subset of SIP, the specification assumes that a majority of SIP implementations will interoperate with the browsers.

5.1. Client Application SIP Requirements

This section focuses on the subset of SIP functionality that will exist within all RTC-WEB client applications. The following User Agent Client (UAC) subset of the SIP specification [RFC3261] is REQUIRED.

- o General User Agent Behavior - [Section 8]

- o Registration - [Section 10]
- o Client Transaction - [Section 17.1]
- o Canceling a Request - [Section 9.1]
- o Terminating a Session - [Section 15.1], [Section 15.1.1]
- o 3.XX Redirect Responses - [Section 8.1.3.4]
- o TLS - [Section 26.3.1]
- o Outbound Proxy

5.2. Client Application Optional SIP Support

In the SIP specification [RFC3261], the SIP features listed below are required for all UAC implementations. RTC-WEB client applications are not a fully featured SIP UAC and will only be implementing a subset of the SIP specification. Thusly, unlike SIP UACs, the following list of SIP features is to be considered OPTIONAL for RTC-WEB client application implementations.

- o INVITEs without an offer
- o re-INVITEs - [Section 14.1]
- o forking - [Section 19.3]
- o S/MIME - [Section 23]
- o SIPS URI Scheme - [Section 26.2.2]

5.3. Required SIP Methods

This section outlines the REQUIRED SIP methods for all RTC-WEB client applications.

- o INVITE - [RFC3261] - Section 13
- o REGISTER - [RFC3261] - Section 10
- o ACK - [RFC3261] - Section 13.2
- o CANCEL - [RFC3261] - Section 13.2
- o BYE - [RFC3261] - Section 13.2

- o UPDATE - [RFC3311]

5.4. Multipart SIP Message Requirements

For handling SIP messages RTC-WEB client applications are required to implement the multipart MIME handling scheme as specified in [RFC5621].

5.5. Identity Requirements

Identity, for the purposes of this section, is defined as a SIP URI. There are two areas concerning SIP identity this specification will address.

The first area covers validation of the message originator. To securely validate a the identity of a SIP message originator, all RTC-WEB client applications are REQUIRED to implement the mechanism specified in [RFC4474].

To support cases where the identity of a caller/callee may change, such as when a call is parked and transferred from the original callee to another party, all RTC-WEB client applications are REQUIRED to implement the identity mechanism specified in [RFC4916]. [RFC3261] implicitly REQUIRES the implementation of the UPDATE method as specified in [RFC3311]

5.6. Network Address Traversal

RTC-WEB client applications MUST support Network Address Translator (NAT) traversal. This section will address SIP-related areas to support NAT traversal.

As called for in the negotiation requirements; RTC-WEB client applications will implement STUN. To support client-managed connections, STUN-based keep-alives as specified in [RFC5626] are REQUIRED.

When SIP is used with UDP, responses to requests are returned to the source address the request came from, and to the port written into the topmost Via header field value of the request. This behavior is not desirable when the RTC-WEB client application is behind a Network Address Translator (NAT). To address UDP traversal problem the "rport" extension as specified in [RFC3581] is REQUIRED.

6. Legacy VoIP Interoperability

The RTC-Web negotiation requirements specify a discrete subset of the

SIP specification. The discrete subset was chosen to implicitly enable broad interoperability between RTC-Web client applications and legacy VoIP systems.

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

Because there are a number of security issues, considerations and requirements for RTC-WEB client applications there is a draft that specifically addresses the RTC-WEB application security considerations. This draft defers it's security considerations and requirements to the security considerations for RTC-Web draft [I-D.ekr-security-considerations-for-rtc-web].

9. Acknowledgements

[TODO - Are there any yet for this area?]

10. Normative References

- [I-D.cbran-jennings-rtc-web-nat]
Bran, C. and C. Jennings, "RTC-Web Network Address Translation", July 2011.
- [I-D.ekr-security-considerations-for-rtc-web]
Rescorla, E., "Security Considerations for RTC-Web", May 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263,

June 2002.

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.
- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- [RFC3581] Rosenberg, J. and H. Schulzrinne, "An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing", RFC 3581, August 2003.
- [RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.
- [RFC4916] Elwell, J., "Connected Identity in the Session Initiation Protocol (SIP)", RFC 4916, June 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5621] Camarillo, G., "Message Body Handling in the Session Initiation Protocol (SIP)", RFC 5621, September 2009.
- [RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.
- [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.

Authors' Addresses

Cary Bran
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 206 256-3502
Email: cbran@cisco.com

Cullen Jennings
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 16, 2018

H. Alvestrand
Google
November 12, 2017

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

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 <https://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 May 16, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Principles and Terminology	4
2.1. Goals of this document	4
2.2. Relationship between API and protocol	5
2.3. On interoperability and innovation	7
2.4. Terminology	8
3. Architecture and Functionality groups	8
4. Data transport	12
5. Data framing and securing	13
6. Data formats	13
7. Connection management	13
8. Presentation and control	14
9. Local system support functions	14
10. IANA Considerations	15
11. Security Considerations	15
12. Acknowledgements	16
13. References	16
13.1. Normative References	16
13.2. Informative References	18
Appendix A. Change log	20
A.1. Changes from draft-alvestrand-dispatch-rtcweb-datagram-00 to -01	20
A.2. Changes from draft-alvestrand-dispatch-01 to draft-alvestrand-rtcweb-overview-00	20
A.3. Changes from draft-alvestrand-rtcweb-00 to -01	20
A.4. Changes from draft-alvestrand-rtcweb-overview-01 to draft-ietf-rtcweb-overview-00	21
A.5. Changes from -00 to -01 of draft-ietf-rtcweb-overview	21
A.6. Changes from -01 to -02 of draft-ietf-rtcweb-overview	21
A.7. Changes from -02 to -03 of draft-ietf-rtcweb-overview	21
A.8. Changes from -03 to -04 of draft-ietf-rtcweb-overview	22
A.9. Changes from -04 to -05 of draft-ietf-rtcweb-overview	22
A.10. Changes from -05 to -06	22
A.11. Changes from -06 to -07	22
A.12. Changes from -07 to -08	22
A.13. Changes from -08 to -09	22

A.14. Changes from -09 to -10	22
A.15. Changes from -10 to -11	23
A.16. Changes from -11 to -12	23
A.17. Changes from -12 to -13	23
A.18. Changes from -13 to -14	23
A.19. Changes from -14 to -15	23
A.20. Changes from -15 to -16	23
A.21. Changes from -16 to -17	24
A.22. Changes from -17 to -18	24
A.23. Changes from -18 to -19	24
Author's Address	24

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.

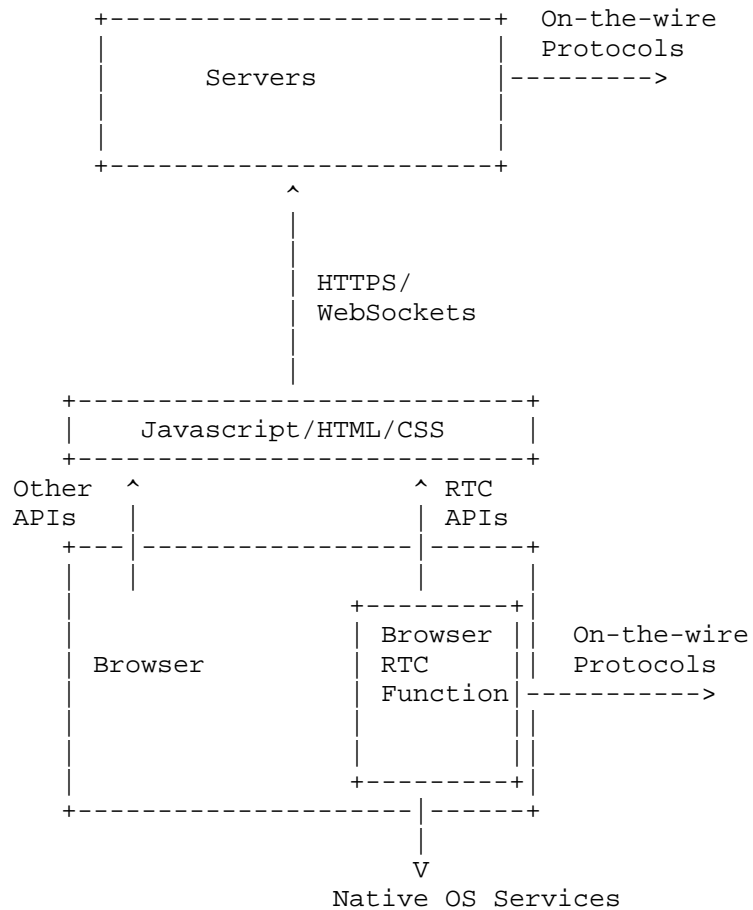


Figure 1: Browser Model

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.

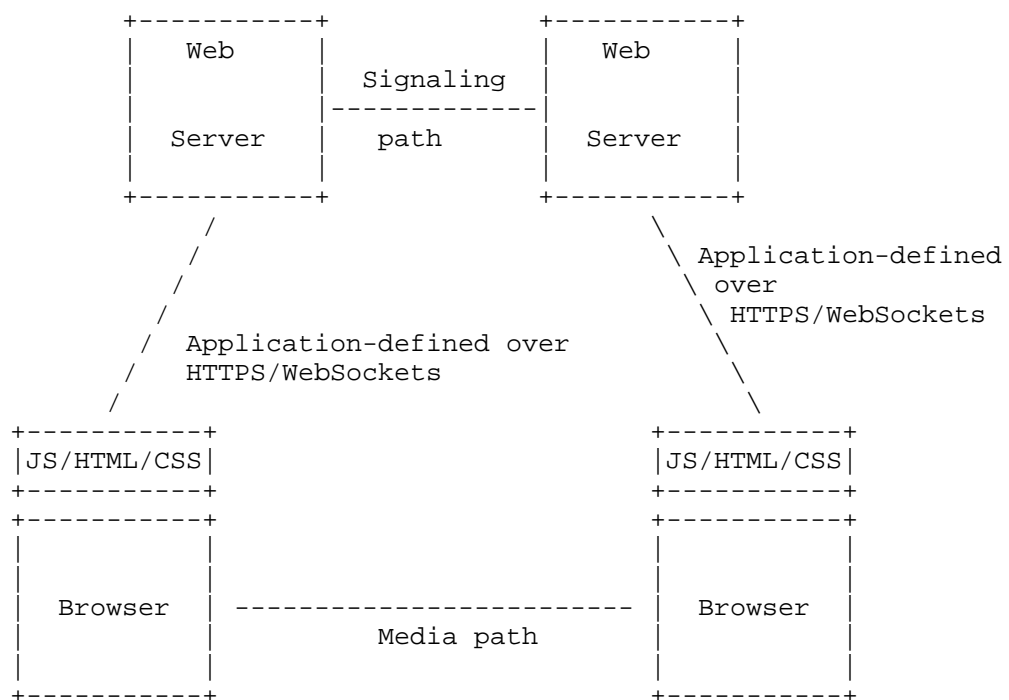


Figure 2: Browser RTC Trapezoid

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-gateways 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

RTCWEB Working Group
Internet-Draft
Intended status: Informational
Expires: July 27, 2015

C. Holmberg
S. Hakansson
G. Eriksson
Ericsson
January 23, 2015

Web Real-Time Communication Use-cases and Requirements
draft-ietf-rtcweb-use-cases-and-requirements-16.txt

Abstract

This document describes web based real-time communication use-cases. Requirements on the browser functionality are derived from the use-cases.

This document was developed in an initial phase of the work with rather minor updates at later stages. It has not really served as a tool in deciding features or scope for the WGs efforts so far. It is being published to record the early conclusions of the working group. It will not be used as a set of rigid guidelines that specifications and implementations will be held to in the future.

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 July 27, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions	3
3. Use-cases	3
3.1. Introduction	4
3.2. Common requirements	4
3.3. Browser-to-browser use-cases	4
3.3.1. Simple Video Communication Service	4
3.3.2. Simple Video Communication Service, NAT/Firewall that blocks UDP	7
3.3.3. Simple Video Communication Service, Firewall that only allows traffic via a HTTP Proxy	7
3.3.4. Simple Video Communication Service, global service provider	7
3.3.5. Simple Video Communication Service, enterprise aspects	8
3.3.6. Simple Video Communication Service, access change . .	9
3.3.7. Simple Video Communication Service, QoS	10
3.3.8. Simple Video Communication Service with screen sharing	10
3.3.9. Simple Video Communication Service with file exchange	11
3.3.10. Hockey Game Viewer	11
3.3.11. Multiparty video communication	12
3.3.12. Multiparty on-line game with voice communication . .	14
3.4. Browser - GW/Server use cases	15
3.4.1. Telephony terminal	15
3.4.2. Fedex Call	16
3.4.3. Video conferencing system with central server	17
4. Requirements summary	18
4.1. General	18
4.2. Browser requirements	18
5. IANA Considerations	22
6. Security Considerations	22
6.1. Introduction	22
6.2. Browser Considerations	22
6.3. Web Application Considerations	23
7. Acknowledgements	23
8. Change Log	23
9. Normative References	30
Appendix A. API requirements	30

Authors' Addresses	33
------------------------------	----

1. Introduction

This document presents a few use-cases of web applications that are executed in a browser and use real-time communication capabilities. In most of the use-cases all end-user clients are web applications, but there are some use-cases where at least one of the end-user clients is of another type (e.g. a mobile phone or a SIP User Agent (UA)).

Based on the use-cases, the document derives requirements related to browser functionality. These requirements are named "Fn", where n is an integer, and are listed in conjunction with the use-cases. A summary is provided in Section 4.2.

This document was developed in an initial phase of the work with rather minor updates at later stages. It has not really served as a tool in deciding features or scope for the WGs efforts so far. It is proposed to be used in a later phase to evaluate the protocols and solutions developed by the WG.

This document also lists requirements related to the API to be used by web applications as an appendix. The reason is that the W3C WebRTC WG has decided to not develop its own use-case/requirement document, but instead use this document. These requirements are named "An", where n is an integer, and are described in Appendix A.

This document was developed in an initial phase of the work with rather minor updates at later stages. It has not really served as a tool in deciding features or scope for the WGs efforts so far. It is being published to record the early conclusions of the working group. It will not be used as a set of rigid guidelines that specifications and implementations will be held to in the future.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

3. Use-cases

3.1. Introduction

This section describes web based real-time communication use-cases, from which requirements are derived.

The following considerations are applicable to all use cases:

- o Clients can be on IPv4-only
- o Clients can be on IPv6-only
- o Clients can be on dual-stack
- o Clients can be connected to networks with different throughput capabilities
- o Clients can be on variable-media-quality networks (wireless)
- o Clients can be on congested networks
- o Clients can be on firewalled networks with no UDP allowed
- o Clients can be on networks with a NAT or IPv4-IPv6 translation devices using any type of Mapping and Filtering behaviors (as described in RFC4787).

3.2. Common requirements

The requirements retrieved from the Simple Video Communication Service use-case (Section 3.3.1) by default apply to all other use-cases, and are considered common. For each individual use-case, only the additional requirements are listed.

3.3. Browser-to-browser use-cases

3.3.1. Simple Video Communication Service

3.3.1.1. Description

Two or more users have loaded a video communication web application into their browsers, provided by the same service provider, and logged into the service it provides. The web service publishes information about user login status by pushing updates to the web application in the browsers. When one online user selects a peer online user, a 1-1 audiovisual communication session between the browsers of the two peers is initiated. The invited user might accept or reject the session.

During session establishment a self-view is displayed, and once the session has been established the video sent from the remote peer is displayed in addition to the self-view. During the session, each user can select to remove and re-insert the self-view as often as desired. Each user can also change the sizes of his/her two video displays during the session. Each user can also pause sending of media (audio, video, or both) and mute incoming media.

It is essential that media and data be encrypted, authenticated and integrity protected on a per IP packet basis and that media and data packets failing the integrity check not be delivered to the application.

The application gives the users the opportunity to stop it from exposing the host IP address to the application of the other user.

Any session participant can end the session at any time.

The two users may be using communication devices with different operating systems and browsers from different vendors.

The web service monitors the quality of the service (focus on quality of audio and video) the end-users experience.

3.3.1.2. Common Requirements

REQ-ID	DESCRIPTION
F1	The browser must be able to use microphones and cameras as input devices to generate streams.
F2	The browser must be able to send streams and data to a peer in the presence of NATs.
F3	Transmitted streams and data must be rate controlled (meaning that the browser must, regardless of application behavior, reduce send rate when there is congestion).
F4	The browser must be able to receive, process and render streams and data ("render" does not apply for data) from peers.
F5	The browser should be able to render good quality audio and video even in the presence of reasonable levels of jitter and packet losses.

- F6 The browser must detect when a stream from a peer is not received anymore.
-
- F7 When there are both incoming and outgoing audio streams, echo cancellation must be made available to avoid disturbing echo during conversation.
-
- F8 The browser must support synchronization of audio and video.
-
- F9 The browser should use encoding of streams suitable for the current rendering (e.g. video display size) and should change parameters if the rendering changes during the session.
-
- F10 The browser must support a baseline audio and video codec.
-
- F11 It must be possible to protect streams and data from wiretapping [RFC2804][RFC7258].
-
- F12 The browser must enable verification, given the right circumstances and by use of other trusted communication, that streams and data received have not been manipulated by any party.
-
- F13 The browser must encrypt, authenticate and integrity protect media and data on a per IP packet basis, and must drop incoming media and data packets that fail the per IP packet integrity check. In addition, the browser must support a mechanism for cryptographically binding media and data security keys to the user identity (see R-ID-BINDING in [RFC5479]).
-
- F14 The browser must make it possible to set up a call between two parties without one party learning the other party's host IP address.
-
- F15 The browser must be able to collect statistics, related to the transport of audio and video between peers, needed to estimate quality of experience.
-

A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A25, A26

3.3.2. Simple Video Communication Service, NAT/Firewall that blocks UDP

3.3.2.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1). The difference is that one of the users is behind a NAT/Firewall that blocks UDP traffic.

3.3.2.2. Additional Requirements

REQ-ID	DESCRIPTION
F18	The browser must be able to send streams and data to a peer in the presence of NATs and Firewalls that block UDP traffic.

3.3.3. Simple Video Communication Service, Firewall that only allows traffic via a HTTP Proxy

3.3.3.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1). The difference is that one of the users is behind a Firewall that only allows traffic via a HTTP Proxy.

3.3.3.2. Additional Requirements

REQ-ID	DESCRIPTION
F21	The browser must be able to send streams and data to a peer in the presence of Firewalls that only allows traffic via a HTTP Proxy, when Firewall policy allows WebRTC traffic.

3.3.4. Simple Video Communication Service, global service provider

3.3.4.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1).

What is added is that the service provider is operating over large geographical areas (or even globally).

Assuming that the Interactive Connectivity Establishment (ICE) mechanism [RFC5245] will be used, this means that the service provider would like to be able to provide several STUN and TURN servers (via the app) to the browser; selection of which one(s) to use is part of the ICE processing. Other reasons for wanting to provide several STUN and TURN servers include support for IPv4 and IPv6, load balancing and redundancy.

Note that ICE support being mandatory does not preclude a WebRTC endpoint from supporting more traversal mechanisms than ICE using STUN and TURN.

3.3.4.2. Additional Requirements

REQ-ID	DESCRIPTION
F19	The browser must be able to use several STUN and TURN servers
A22	

3.3.5. Simple Video Communication Service, enterprise aspects

3.3.5.1. Description

This use-case is similar to the Simple Video Communication Service use-case (Section 3.3.1).

What is added is aspects when using the service in enterprises. ICE is assumed in the further description of this use-case.

An enterprise that uses a RTCWEB based web application for communication desires to audit all RTCWEB based application sessions used from inside the company towards any external peer. To be able to do this they deploy a TURN server that straddles the boundary between the internal and the external network.

The firewall will block all attempts to use STUN with an external destination unless they go to the enterprise auditing TURN server. In cases where employees are using RTCWEB applications provided by an external service provider they still want the traffic to stay inside their internal network and in addition not load the straddling TURN server, thus they deploy a STUN server allowing the RTCWEB client to

determine its server reflexive address on the internal side. Thus enabling cases where peers are both on the internal side to connect without the traffic leaving the internal network. It must be possible to configure the browsers used in the enterprise with network specific STUN and TURN servers. This should be possible to achieve by auto-configuration methods. The RTCWEB functionality will need to utilize both network specific STUN and TURN resources and STUN and TURN servers provisioned by the web application.

3.3.5.2. Additional Requirements

REQ-ID	DESCRIPTION
F20	The browser must support the use of STUN and TURN servers that are supplied by entities other than the web application (i.e. the network provider).

3.3.6. Simple Video Communication Service, access change

3.3.6.1. Description

This use-case is almost identical to the Simple Video Communication Service use-case (Section 3.3.1). The difference is that the user changes network access during the session.

The communication device used by one of the users has several network adapters (Ethernet, WiFi, Cellular). The communication device is accessing the Internet using Ethernet, but the user has to start a trip during the session. The communication device automatically changes to use WiFi when the Ethernet cable is removed and then moves to cellular access to the Internet when moving out of WiFi coverage. The session continues even though the access method changes.

3.3.6.2. Additional Requirements

REQ-ID	DESCRIPTION
F17	The communication session must survive across a change of the network interface used by the session

3.3.7. Simple Video Communication Service, QoS

3.3.7.1. Description

This use-case is almost identical to the Simple Video Communication Service, access change use-case (Section 3.3.6). The use of Quality of Service (QoS) capabilities is added:

The user in the previous use case that starts a trip is behind a common residential router that supports differentiation of traffic. In addition, the user's provider of cellular access has QoS support enabled. The user is able to take advantage of the QoS support both when accessing via the residential router and when using cellular.

3.3.7.2. Additional Requirements

REQ-ID	DESCRIPTION
F17	The communication session must survive across a change of the network interface used by the session
F22	The browser should be able to take advantage of available capabilities (supplied by network nodes) to differentiate voice, video and data appropriately.

3.3.8. Simple Video Communication Service with screen sharing

3.3.8.1. Description

This use-case has the audio and video communication of the Simple Video Communication Service use-case (Section 3.3.1).

But in addition to this, one of the users can share what is being displayed on her/his screen with a peer. The user can choose to share the entire screen, part of the screen (part selected by the user) or what a selected application displays with the peer.

3.3.8.2. Additional Requirements

REQ-ID	DESCRIPTION
F36	The browser must be able to generate streams using the entire user display, a specific area of the user's display or the information being displayed by a specific application.

A21

3.3.9. Simple Video Communication Service with file exchange

3.3.9.1. Description

This use-case has the audio and video communication of the Simple Video Communication Service use-case (Section 3.3.1).

But in addition to this, the users can send and receive files stored in the file system of the device used.

3.3.9.2. Additional Requirements

REQ-ID	DESCRIPTION
F35	The browser must be able to send reliable data traffic to a peer browser.

A21, A24

3.3.10. Hockey Game Viewer

3.3.10.1. Description

An ice-hockey club uses an application that enables talent scouts to, in real-time, show and discuss games and players with the club manager. The talent scouts use a mobile phone with two cameras, one front facing and one rear facing.

The club manager uses a desktop, equipped with one camera, for viewing the game and discussing with the talent scout.

Before the game starts, and during game breaks, the talent scout and the manager have a 1-1 audiovisual communication session. On the mobile phone, only the camera facing the talent scout is used. On the user display of the mobile phone, the video of the club manager

is shown with a picture-in-picture thumbnail of the rear facing camera (self-view). On the display of the desktop, the video of the talent scout is shown with a picture-in-picture thumbnail of the desktop camera (self-view).

When the game is on-going, the talent scout activates the use of the front facing camera, and that stream is sent to the desktop (the stream from the rear facing camera continues to be sent all the time). The video stream captured by the front facing camera (that is capturing the game) of the mobile phone is shown in a big window on the desktop screen, with picture-in-picture thumbnails of the rear facing camera and the desktop camera (self-view). On the display of the mobile phone the game is shown (front facing camera) with picture-in-picture thumbnails of the rear facing camera (self-view) and the desktop camera. As the most important stream in this phase is the video showing the game, the application used in the talent scout's mobile sets higher priority for that stream.

3.3.10.2. Additional Requirements

REQ-ID	DESCRIPTION
F22	The browser should be able to take advantage of available capabilities (supplied by network nodes) to differentiate voice, video and data appropriately.
F25	The browser must be able to render several concurrent audio and video streams.

A17, A23

3.3.11. Multiparty video communication

3.3.11.1. Description

In this use-case, the Simple Video Communication Service use-case (Section 3.3.1) is extended by allowing multiparty sessions. No central server is involved - the browser of each participant sends and receives streams to and from all other session participants. The web application in the browser of each user is responsible for setting up streams to all receivers.

In order to enhance the user experience, the web application renders the audio coming from different participants so that it is

experienced to come from different spatial locations. This is done automatically, but users can change how the different participants are placed in the (virtual) room. In addition the levels in the audio signals are adjusted before mixing.

Another feature intended to enhance the use experience is that the video window that displays the video of the currently speaking peer is highlighted.

Each video stream received is by default displayed in a thumbnail frame within the browser, but users can change the display size.

Note: What this use-case adds in terms of requirements is capabilities to send streams to and receive streams from several peers concurrently, as well as the capabilities to render the video from all received streams and be able to spatialize, level adjust and mix the audio from all received streams locally in the browser. It also adds the capability to measure the audio level/activity.

3.3.11.2. Additional Requirements

REQ-ID	DESCRIPTION
F23	The browser must be able to transmit streams and data to several peers concurrently.
F24	The browser must be able to receive streams and data from multiple peers concurrently.
F25	The browser must be able to render several concurrent audio and video streams.
F26	The browser must be able to mix several audio streams.
F27	The browser must be able to apply spatialization effects to audio streams.
F28	The browser must be able to measure the voice activity level in audio streams.
F29	The browser must be able to change the voice activity level in audio streams.
A13, A14, A15, A16	

3.3.12. Multiparty on-line game with voice communication

3.3.12.1. Description

This use case is based on the previous one. In this use-case, the voice part of the multiparty video communication use case is used in the context of an on-line game. The received voice audio media is rendered together with game sound objects. For example, the sound of a tank moving from left to right over the screen must be rendered and played to the user together with the voice media.

Quick updates of the game state is required, and have higher priority than the voice.

Note: the difference regarding local audio processing compared to the "Multiparty video communication" use-case is that other sound objects than the streams must be possible to be included in the spatialization and mixing. "Other sound objects" could for example be a file with the sound of the tank; that file could be stored locally or remotely.

3.3.12.2. Additional Requirements

REQ-ID	DESCRIPTION
F22	The browser should be able to take advantage of available capabilities (supplied by network nodes) to differentiate voice, video and data appropriately.
F23	The browser must be able to transmit streams and data to several peers concurrently.
F24	The browser must be able to receive streams and data from multiple peers concurrently.
F25	The browser must be able to render several concurrent audio and video streams.
F26	The browser must be able to mix several audio streams.
F27	The browser must be able to apply spatialization effects when playing audio streams.
F28	The browser must be able to measure the voice activity level in audio streams.
F29	The browser must be able to change the voice activity level in audio streams.
F30	The browser must be able to process and mix sound objects (media that is retrieved from another source than the established media stream(s) with the peer(s) with audio streams.
F34	The browser must be able to send short latency unreliable datagram traffic to a peer browser [RFC5405].

A13, A14, A15, A16, A17, A18, A23

3.4. Browser - GW/Server use cases

3.4.1. Telephony terminal

3.4.1.1. Description

A mobile telephony operator allows its customers to use a web browser to access their services. After a simple log in the user can place and receive calls in the same way as when using a normal mobile phone. When a call is received or placed, the identity is shown in the same manner as when a mobile phone is used.

Note: With "place and receive calls in the same way as when using a normal mobile phone" it is meant that you can dial a number, and that your mobile telephony operator has made available your phone contacts on line, so they are available and can be clicked to call, and be used to present the identity of an incoming call. If the callee is not in your phone contacts the number is displayed. Furthermore, your call logs are available, and updated with the calls made/received from the browser. And for people receiving calls made from the web browser the usual identity (i.e. the phone number of the mobile phone) will be presented.

3.4.1.2. Additional Requirements

REQ-ID	DESCRIPTION
F31	The browser must support an audio media format (codec) that is commonly supported by existing telephony services.
F33	The browser must be able to initiate and accept a media session where the data needed for establishment can be carried in SIP.

3.4.2. Fedex Call

3.4.2.1. Description

Alice uses her web browser with a service that allows her to call PSTN numbers. Alice calls 1-800-gofedex. Alice should be able to hear the initial prompts from the fedex Interactive Voice Responder (IVR) and when the IVR says press 1, there should be a way for Alice to navigate the IVR.

3.4.2.2. Additional Requirements

REQ-ID	DESCRIPTION
F31	The browser must support an audio media format (codec) that is commonly supported by existing telephony services.
F32	There should be a way to navigate a Dual-tone multi-frequency signaling (DTMF) based Interactive voice response (IVR) System

3.4.3. Video conferencing system with central server

3.4.3.1. Description

An organization uses a video communication system that supports the establishment of multiparty video sessions using a central conference server.

The browser of each participant sends an audio stream (type in terms of mono, stereo, 5.1, ... depending on the equipment of the participant) to the central server. The central server mixes the audio streams (and can in the mixing process naturally add effects such as spatialization) and sends towards each participant a mixed audio stream which is played to the user.

The browser of each participant sends video towards the server. For each participant one high resolution video is displayed in a large window, while a number of low resolution videos are displayed in smaller windows. The server selects what video streams to be forwarded as main- and thumbnail videos respectively, based on speech activity. As the video streams to display can change quite frequently (as the conversation flows) it is important that the delay from when a video stream is selected for display until the video can be displayed is short.

All participants are authenticated by the central server, and authorized to connect to the central server. The participants are identified to each other by the central server, and the participants do not have access to each others' credentials such as e-mail addresses or login IDs.

Note: This use-case adds requirements on support for fast stream switches F16. There exist several solutions that enable the server to forward one high resolution and several low resolution video streams: a) each browser could send a high resolution, but scalable

stream, and the server could send just the base layer for the low resolution streams, b) each browser could in a simulcast fashion send one high resolution and one low resolution stream, and the server just selects or c) each browser sends just a high resolution stream, the server transcodes into low resolution streams as required.

3.4.3.2. Additional Requirements

REQ-ID	DESCRIPTION
F16	The browser must support insertion of reference frames in outgoing media streams when requested by a peer.
F25	The browser must be able to render several concurrent audio and video streams.

4. Requirements summary

4.1. General

This section contains the requirements on the browser derived from the use-cases in Section 3.

NOTE: It is assumed that the user applications are executed on a browser. Whether the capabilities to implement specific browser requirements are implemented by the browser application, or are provided to the browser application by the underlying operating system, is outside the scope of this document.

4.2. Browser requirements

Common, basic requirements	
REQ-ID	DESCRIPTION
F1	The browser must be able to use microphones and cameras as input devices to generate streams.
F2	The browser must be able to send streams and data to a peer in the presence of NATs.
F3	Transmitted streams and data must be rate controlled (meaning that the browser must, regardless of application behavior, reduce send rate when there is congestion).

-
- F4 The browser must be able to receive, process and render streams and data ("render" does not apply for data) from peers.
-
- F5 The browser should be able to render good quality audio and video even in the presence of reasonable levels of jitter and packet losses.
-
- F6 The browser must detect when a stream from a peer is not received anymore
-
- F7 When there are both incoming and outgoing audio streams, echo cancellation must be made available to avoid disturbing echo during conversation.
-
- F8 The browser must support synchronization of audio and video.
-
- F9 The browser should use encoding of streams suitable for the current rendering (e.g. video display size) and should change parameters if the rendering changes during the session
-
- F10 The browser must support a baseline audio and video codec
-
- F11 It must be possible to protect streams and data from wiretapping [RFC2804][RFC7258].
-
- F12 The browser must enable verification, given the right circumstances and by use of other trusted communication, that streams and data received have not been manipulated by any party.
-
- F13 The browser must encrypt, authenticate and integrity protect media and data on a per-packet basis, and must drop incoming media and data packets that fail the per-packet integrity check. In addition, the browser must support a mechanism for cryptographically binding media and data security keys to the user identity (see R-ID-BINDING in [RFC5479]).
-
- F14 The browser must make it possible to set up a call between two parties without one party

learning the other party's host IP address.

F15 The browser must be able to collect statistics,
 related to the transport of audio and video
 between peers, needed to estimate quality of
 experience.

Requirements related to network and topology

REQ-ID DESCRIPTION

F16 The browser must support insertion of reference frames
 in outgoing media streams when requested by a peer.

F17 The communication session must survive across a
 change of the network interface used by the
 session

F18 The browser must be able to send streams and
 data to a peer in the presence of NATs and
 Firewalls that block UDP traffic.

F19 The browser must be able to use several STUN
 and TURN servers

F20 The browser must support the use of STUN and TURN
 servers that are supplied by entities other than
 the web application (i.e. the network provider).

F21 The browser must be able to send streams and
 data to a peer in the presence of Firewalls that only
 allows traffic via a HTTP Proxy, when Firewall policy
 allows WebRTC traffic.

F22 The browser should be able to take advantage
 of available capabilities (supplied by network
 nodes) to differentiate voice, video and data
 appropriately.

Requirements related to multiple peers and streams

REQ-ID DESCRIPTION

F23 The browser must be able to transmit streams and
 data to several peers concurrently.

F24 The browser must be able to receive streams and
 data from multiple peers concurrently.

F25 The browser must be able to render several
 concurrent audio and video streams.

F26 The browser must be able to mix several
 audio streams.

Requirements related to audio processing

REQ-ID DESCRIPTION

F27 The browser must be able to apply spatialization
 effects when playing audio streams.

F28 The browser must be able to measure the
 voice activity level in audio streams.

F29 The browser must be able to change the
 voice activity level in audio streams.

F30 The browser must be able to process and mix
 sound objects (media that is retrieved from
 another source than the established media
 stream(s) with the peer(s) with audio streams.

Requirements related to legacy interop

REQ-ID DESCRIPTION

F31 The browser must support an audio media format
 (codec) that is commonly supported by existing
 telephony services.

F32 There should be a way to navigate
 a Dual-tone multi-frequency signaling (DTMF)
 based Interactive voice response (IVR) System

F33 The browser must be able to initiate and
 accept a media session where the data needed
 for establishment can be carried in SIP.

Other requirements

REQ-ID DESCRIPTION

F34 The browser must be able to send short
 latency unreliable datagram traffic to a
 peer browser [RFC5405].

-
- F35 The browser must be able to send reliable data traffic to a peer browser.
-
- F36 The browser must be able to generate streams using the entire user display, a specific area of the user's display or the information being displayed by a specific application.
-

5. IANA Considerations

There are no IANA actions in this document.

6. Security Considerations

6.1. Introduction

A malicious web application might use the browser to perform Denial Of Service (DOS) attacks on NAT infrastructure, or on peer devices. Also, a malicious web application might silently establish outgoing, and accept incoming, streams on an already established connection.

Based on the identified security risks, this section will describe security considerations for the browser and web application.

6.2. Browser Considerations

The browser is expected to provide mechanisms for getting user consent to use device resources such as camera and microphone.

The browser is expected to provide mechanisms for informing the user that device resources such as camera and microphone are in use ("hot").

The browser must provide mechanisms for users to revise and even completely revoke consent to use device resources such as camera and microphone.

The browser is expected to provide mechanisms for getting user consent to use the screen (or a certain part of it) or what a certain application displays on the screen as source for streams.

The browser is expected to provide mechanisms for informing the user that the screen, part thereof or an application is serving as a stream source ("hot").

The browser must provide mechanisms for users to revise and even completely revoke consent to use the screen, part thereof or an application is serving as a stream source.

The browser is expected to provide mechanisms in order to assure that streams are the ones the recipient intended to receive.

The browser is expected to provide mechanisms that allows the users to verify that the streams received have not be manipulated (F12).

The browser needs to ensure that media is not sent, and that received media is not rendered, until the associated stream establishment and handshake procedures with the remote peer have been successfully finished.

The browser needs to ensure that the stream negotiation procedures are not seen as Denial Of Service (DOS) by other entities.

6.3. Web Application Considerations

The web application is expected to ensure user consent in sending and receiving media streams.

7. Acknowledgements

The authors wish to thank Bernard Aboba, Gunnar Hellstrom, Martin Thomson, Lars Eggert, Matthew Kaufman, Emil Ivov, Eric Rescorla, Eric Burger, John Leslie, Dan Wing, Richard Barnes, Barry Dingle, Dale Worley, Ted hardie, Mary Barnes, Dan Burnett, Stephan Wenger, Harald Alvestrand, Cullen Jennings, Andrew Hutton and everyone else in the RTCWEB community that have provided comments, feedback, text and improvement proposals on the document. A big thank you to everyone that provided comments as part of the IESG evaluation, and to everyone else that provided comments and input in order to improve the document.

8. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-ietf-rtcweb-use-cases-and-requirements-15

- o Changes based on comment from Stephen Farrell:

- o - A1 modified, to also cover access to the local file system.
- o Changes based on comments from Benoit Claise:
- o - RFC 5245 added to references.
- o - Note added to Annex A, indicating that the API requirements are not normative.
- o Changes based on comments from Brian Carpenter:
- o - RFC 7258 added to references.
- o - Terminology fixes:
- o -- 'prioritize' -> 'differentiate'.
- o -- 'prioritization' -> 'differentiation'.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-14

- o Changes based on comments from the ops-dir:
- o - Editorial fixes.
- o - F13: 'per-packet basis' -> 'per IP packet basis'.
- o - F22: Text corrected in one occurrence.
- o - F25: 'audio' added.
- o Changes based on comments from IESG
- o - Editorial fixes.
- o - Disclaimer text suggested by Alissa Cooper added.
- o - F11: Reference to RFC 7258 added.
- o - F27: 'when playing' removed.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-10

- o Described that the API requirements are really from a W3C perspective and are supplied as an appendix in the introduction. Moved API requirements to an Appendix.

- o Removed the "Conventions" section with the key-words and reference to RFC2119. Also changed uppercase MUST's/SHOULD's to lowercase.
- o Added a note on the proposed use of the document to the introduction.
- o Removed the note talking about WS from the "Firewall that only allows http" use-case.
- o Removed the word "Skype" that was used as example in one of the use-cases.
- o Clarified F3 (the req saying the everything the browser sends must be rate controlled).
- o Removed the TBD saying we need to define reasonable levels from the requirement saying that quality must be good even in presence of packet losses (F5), and changed "must" to "should" (Based on a list discussion involving Bernard).
- o Removed F6 ("The browser must be able to handle high loss and jitter levels in a graceful way."), also after a list discussion.
- o Clarified F7 (used to say that the browser must support fast stream switches, now says that reference frames must be inserted when requested).
- o Removed the questions from F9 (echo cancellation), F10 (synchronization), F21 (telephony codec).
- o Exchanged "restrictive firewalls" for "limited middleboxes" in F19 (as proposed by Martin).
- o Expanded DTMF and IVR in F22 (proposed by Martin)
- o Added ref to RFC5405 in F23 (proposed by Lars Eggert).
- o Exchanged "service provided" for "web application" in F32.
- o Changed the text in 3.2.1 that motivates F36 (new text "It is essential that media and data be encrypted, authenticated ... bound to the user identity."); and rewrote F36, included a ref to RFC5479.
- o Changed "quality of service" to "quality of experience" in F38.
- o Added F39.

- o Used new formulation of A17 (proposed by Martin).
- o Updated A20.
- o Updated A25.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-09

- o Changed "video communication session" to "audiovisual communication session."

Changes from draft-ietf-rtcweb-use-cases-and-requirements-08

- o Changed "eavesdropping" to "wiretapping" and referenced RFC2804.
- o Removed informal ref webrtc_req; that document has been abandoned by the W3C webrtc WG.
- o Added use-case where one user is behind a Firewall that only allows http; derived req. F37.
- o Changed F24 slightly; MUST-> SHOULD, inserted "available".
- o Added a clause to "Simple video communication service" saying that the service provider monitors the quality of service, and derived reqs F38 and A26.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-07

- o Added "and data exchange" to 1. Introduction.
- o Removed cone and symmetric NAT from 4.1 Introduction, refers to RFC4787 instead.
- o Added text on enabling verification of that the media has not been manipulated by anyone to use-case "Simple Video Communication Service", derived req. F35
- o Added text on that the browser should reject media (data) that has been created/injected/modified by non-trusted party, derived req. F36
- o Added text on enabling the app to refrain from revealing IP address to use-case "Simple Video Communication Service", derived req. A25
- o Added use-case "Simple Video Communication Service with file exchange", derived reqs F33 and A24

- o Added priority of video streams to "Hockey game viewer" use case, added priority of data to "on-line game use-case", derived reqs F34 and A23
- o In F22, "the IVR" -> "a DTMF based IVR".
- o Updated req F23 to clarify that requirements such as NAT traversal, protection from eavesdropping, rate control applies also to datagram.

Changes from draft-ietf-rtcweb-use-cases-and-requirements-06

- o Renaming of requirements (FaI1 -> F31), (FaI2 -> F32) and (AaI1 -> A22)

Changes from draft-ietf-rtcweb-use-cases-and-requirements-05

- o Added use-case "global service provider", derived reqs associated with several STUN/TURN servers
- o Added use-case "enterprise aspects", derived req associated with enabling the network provider to supply STUN and TURN servers
- o The requirements from the above are ICE specific and labeled accordingly
- o Separated the requirements phrased like "processing such as pan, mix and render" for audio to be specific reqs on spatialization, level measurement, level adjustment and mixing (discussed on the lists in <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01648.html> and <http://lists.w3.org/Archives/Public/public-webrtc/2011Sep/0102.html>)
- o Added use-case on sharing as decided in <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01700.html>, derived reqs F30 and A21
- o Added the list of common considerations proposed in mail <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01562.html> to the Introduction of the use-case section

Changes from draft-ietf-rtcweb-use-cases-and-requirements-04

- o Most changes based on the input from Dan Burnett <http://www.ietf.org/mail-archive/web/rtcweb/current/msg00948.html>
- o Many editorial changes
- o 4.2.1.1 Clarified

- o Some clarification added to 4.3.1.1 as a note
- o F-requirements updated (see reply to Dan's mail).
- o Almost all A-requirements updated to start "The Web API MUST provide ..."
- o A8 removed, A9 rephrased to cover A8 and old A9
- o A15 rephrased
- o For more details, and discussion, look at the response to Dan's mail <http://www.ietf.org/mail-archive/web/rtcweb/current/msg01177.html>

Changes from draft-ietf-rtcweb-use-cases-and-requirements-03

- o Editorials
- o Changed when the self-view is displayed in 4.2.1.1, and added words about allowing users to remove and re-insert it.
- o Clarified 4.2.6.1
- o Removed the "mono" stuff from 4.2.7.1
- o Added that communication should not be possible to eavesdrop to most use cases - and req. F17
- o Re-phrased 4.3.3.1 to not describe the technical solution so much, and removed "stereo" stuff. Solution possibilities are now in a note.
- o Re-inserted API requirements after discussion in the W3C webrtc WG. (Re-phrased A15 and added A18 compared to version -02).

Changes from draft-ietf-rtcweb-use-cases-and-requirements-02

- o Removed description/list of API requirements, instead
- o Reference to W3C webrtc_reqs document for API requirements

Changes from draft-ietf-rtcweb-ucreqs-01

- o Changed Intended status to Information
- o Changed "Ipr" to "trust200902"

- o Added use case "Simple video communication service, NAT/Firewall that blocks UDP", and derived new req F26
- o Added use case "Distributed Music Band" and derived new req A17
- o Added F24 as requirement derived from use case "Simple video communication service with inter-operator calling"
- o Added section "Additional use cases"
- o Added text about ID handling to multiparty with central server use case
- o Re-phrased A1 slightly

Changes from draft-ietf-rtcweb-ucreqs-00

- o - Reshuffled: Just two main groups of use cases (b2b and b2GW/Server); removed some specific use cases and added them instead as flavors to the base use case (Simple video communication)
- o - Changed the formulation of F19
- o - Removed the requirement on an API for DTMF
- o - Removed "FX3: There SHOULD be a mapping of the minimum needed data for setting up connections into SIP, so that the restriction to SIP-carriable data can be verified. Not a rew on the browser but rather on a document"
- o - (see <http://www.ietf.org/mail-archive/web/rtcweb/current/msg00227.html> for more details)
- o -Added text on informing user of that mic/cam is being used and that it must be possible to revoke permission to use them in section 7.

Changes from draft-holmberg-rtcweb-ucreqs-01

- o - Draft name changed to draft-ietf-rtcweb-ucreqs
- o - Use-case grouping introduced
- o - Additional use-cases added
- o - Additional reqs added (derived from use cases): F19-F25, A16-A17

Changes from draft-holmberg-rtcweb-ucreqs-00

- o - Mapping between use-cases and requirements added (Harald Alvestrand, 090311)
- o - Additional security considerations text (Harald Alvestrand, 090311)
- o - Clarification that user applications are assumed to be executed by a browser (Ted Hardie, 080311)
- o - Editorial corrections and clarifications

9. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.
- [RFC5479] Wing, D., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, April 2009.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, May 2014.

Appendix A. API requirements

This section contains the requirements on the API derived from the use-cases in Section 3.

NOTE: As W3C is responsible for the API, the API requirements in this specification are not normative.

REQ-ID	DESCRIPTION

A1	The Web API must provide means for the application to ask the browser for permission

to use cameras and microphones as input devices,
and to have access to the local file system.

-
- A2 The Web API must provide means for the web application to control how streams generated by input devices are used.
-
- A3 The Web API must provide means for the web application to control the local rendering of streams (locally generated streams and streams received from a peer).
-
- A4 The Web API must provide means for the web application to initiate sending of stream/stream components to a peer.
-
- A5 The Web API must provide means for the web application to control the media format (codec) to be used for the streams sent to a peer.
- NOTE: The level of control depends on whether the codec negotiation is handled by the browser or the web application.
-
- A6 The Web API must provide means for the web application to modify the media format for streams sent to a peer after a media stream has been established.
-
- A7 The Web API must provide means for informing the web application of whether the establishment of a stream with a peer was successful or not.
-
- A8 The Web API must provide means for the web application to mute/unmute a stream or stream component(s). When a stream is sent to a peer mute status must be preserved in the stream received by the peer.
-
- A9 The Web API must provide means for the web application to cease the sending of a stream to a peer.
-
- A10 The Web API must provide means for the web application to cease processing and rendering of a stream received from a peer.
-

- A11 The Web API must provide means for informing the web application when a stream from a peer is no longer received.
-
- A12 The Web API must provide means for informing the web application when high loss rates occur.
-
- A13 The Web API must provide means for the web application to apply spatialization effects to audio streams.
-
- A14 The Web API must provide means for the web application to detect the level in audio streams.
-
- A15 The Web API must provide means for the web application to adjust the level in audio streams.
-
- A16 The Web API must provide means for the web application to mix audio streams.
-
- A17 The Web API must provide a way to identify streams such that an application is able to match streams on a sending peer with the same stream on all receiving peers.
-
- A18 The Web API must provide a mechanism for sending and receiving isolated discrete chunks of data.
-
- A19 The Web API must provide means for the web application to indicate the type of audio signal (speech, audio) for audio stream(s)/stream component(s).
-
- A20 It must be possible for an initiator or a responder web application to indicate the types of media it is willing to accept incoming streams for when setting up a connection (audio, video, other). The types of media to be accepted can be a subset of the types of media the browser is able to accept.
-
- A21 The Web API must provide means for the application to ask the browser for permission to the screen, a certain area on the screen or what a certain application displays on the

screen as input to streams.

-
- A22 The Web API must provide means for the application to specify several STUN and/or TURN servers to use.
-
- A23 The Web API must provide means for the application to specify the priority to apply for outgoing streams and data.
-
- A24 The Web API must provide a mechanism for sending and receiving files.
-
- A25 It must be possible for the application to instruct the browser to refrain from exposing the host IP address to the application
-
- A26 The Web API must provide means for the application to obtain the statistics (related to transport, and collected by the browser) needed to estimate quality of service.
-

Authors' Addresses

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Stefan Hakansson
Ericsson
Laboratoriegrend 11
Lulea 97128
Sweden

Email: stefan.lk.hakansson@ericsson.com

Goran AP Eriksson
Ericsson
Farogatan 6
Stockholm 16480
Sweden

Email: goran.ap.eriksson@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 2, 2011

A. Johnston
Avaya
P. Zimmermann
Zfone Project
May 31, 2011

RTCWEB Media Privacy
draft-johnston-rtcweb-media-privacy-00

Abstract

RTCWEB is the joint effort between the IETF and the W3C to add real-time voice, video, and communication capabilities to browsers. This document looks at the requirements for media privacy and existing mechanisms.

Status of this Memo

This Internet-Draft is submitted to IETF 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 2, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Media Security Requirements	3
3. Security Mechanism Discussion	4
4. Summary	5
5. Security Considerations	6
6. Informative References	6
Authors' Addresses	7

1. Introduction

The requirements for real-time communications in web browsers or RTCWEB are currently being discussed and developed. For both the IETF and the W3C, there are significant challenges due to the unique architecture of browsers.

The same is true for security as well - the requirements are evolving, but starting to come into focus. This draft raises a few issues relating to media security and privacy, something the authors have spent considerable time and effort thinking, writing, and deploying code over the past seven years.

2. Media Security Requirements

One possible model for RTCWEB is described in [I-D.alvestrand-dispatch-rtcweb-protocols], which is summarized here. To implement RTCWEB, both signaling and media is needed. Media for audio and video will likely use RTP [RFC3550], and using some NAT traversal/media authorization approach, will ideally go end-to-end between the two browsers, bypassing the web server and any intermediaries. ICE [RFC5245] is commonly mentioned as a potential protocol for both the NAT traversal method and also for the media authorization method.

Signaling is quite a different story, however. This approach does not standardize any signaling between the browser and the web server. Instead, the state machine and capability negotiation abilities will be downloaded from the web server into the browser - the same way other features and functionality are provided in web apps and pages today. For example, Javascript could be used for this purpose.

There are two interesting side effects of this. It means that there will be many different signaling protocols used for RTCWEB. Also, it means that will not be any single security or trust model for the signaling - it will depend on the web page, the application, and the way in which the signaling works.

Both of these side effects pose significant challenges for media security. The best way to secure RTP streams is to use SRTP [RFC3711]. However, SRTP requires a key management protocol. A key management protocol generates and distributes the symmetric secret keys to the sender and receiver of the stream. The simplest SRTP key management is to have the SRTP sender generate the key and use the signaling channel to send it to the receiver. However, in order to do this, the signaling protocol must meet some security requirements relating to confidentiality. In the RTCWEB architecture, there is no

standardization of the signaling protocol possible, and hence this approach can not be used. Another approach which does not rely on security properties of the signaling protocol is to use a key management system backed by a PKI. In this way, secret keys can be transported by any signaling protocol. However, this requires public private key pairs on every browser, and the infrastructure to manage them.

Instead, what is needed is a key management protocol for SRTP that does not place any reliance on the signaling protocol.

3. Security Mechanism Discussion

The issues and requirements in the previous section are very different from those normally encountered in telephony systems. Nearly all key management protocols for SRTP either rely on a PKI-backed certificate or place strict requirements and trust on the signaling layer. However, the ZRTP key management protocol [RFC6189] does not. In fact, to read its design principles, one might come to the conclusion that it was specifically designed for RTCWEB, despite the fact that it predates the RTCWEB effort by five years.

ZRTP is an entirely self-contained key management protocol for SRTP that places no requirements or reliance on the signaling path. It was originally designed as an extension to RTP, but is now a separate protocol that runs over the same ports and IP addresses as an RTP stream. Today, ZRTP is used with SIP, Jingle, and even proprietary VoIP (Voice over IP) and video systems - the only requirement is that they use RTP for media. This flexibility is something that is simply not possible for other key management protocols. It implements its own discovery mechanism, having first applied the concept of "Best Effort Encryption" to VoIP as defined in [RFC5479]. It uses an in-band Diffie Hellman exchange to generate the secret keying material for SRTP. ZRTP avoids the need for PKI backed certificates by using techniques borrowed from SSH and key continuity.

ZRTP has been widely discussed in the IETF, and has been published by the IETF as an informational RFC, to document an existing and deployed security model. Through this process, ZRTP benefited from significant review from the IETF and security community. ZRTP inspired other media-path keying protocols such as DTLS/SRTP [RFC5764]. However, DTLS-SRTP missed the mark on many of the key advantages of ZRTP and has seen little or no deployment or interest in the marketplace despite being published as a proposed standard. The single most important failure is its reliance on either PKI backed endpoint certificates, or on an end-to-end integrity protected signaling path. While there are SIP mechanisms that have been

published to implement an end-to-end integrity protected signaling path [RFC4474], this approach also has no deployment and no traction in the industry. As discussed earlier, there is way to place any requirements on the signaling protocol, let alone one as difficult as end-to-end integrity protection.

In the development of ZRTP, it was realized that there are scenarios in which the media can not be encrypted end-to-end. For example, when a client has a trusted server or PBX which provides media services in the path. For these cases, ZRTP developed mechanisms for handling a "trusted MiTM" which can terminate than reoriginate the SRTP encryption. This is done without compromising the basic security of the protocol, or allowing arbitrary MiTM entities in the media path. With RTCWEB applications, there may be cases where the web server application is providing media services and hence needs access to the media path. ZRTP can support these scenarios, allowing for a user to explicitly authorize this, while still having all the benefits of ZRTP. ZRTP also handles cases where each endpoint of a communications session have a trusted MiTM. In this case, there will actually be three separately encrypted media paths. These types of scenarios could easily be encountered where each user has a trusted MiTM web server.

To take full advantage of ZRTP, a voice path is needed in order for users to compare the Short Authentication String (SAS). However, ZRTP still provides security similar to SSH in its key continuity. Also, ZRTP normally requires a display for rendering the SAS, but this is not an issue for a browser.

RFC 6189 documents the ZRTP protocol as it is deployed today in VoIP systems. For the RTCWEB application, it is likely that modifications and enhancements might need to be made. It is the hope of the authors that these modifications could be done by the working group in a way that does not compromise the core principles of ZRTP, and also perhaps provides fallback interoperability between browsers and existing ZRTP VoIP devices and systems.

4. Summary

In summary, this draft has discussed some of the unique requirements of RTCWEB media security and shown that ZRTP actually meets these. In fact, the authors believe that ZRTP is ideally architected for providing media security, privacy, and even some identity services for RTCWEB. ZRTP is not perfect, but it has the correct architecture that other protocols do not have, and can be adapted to meet the needs of the RTCWEB effort.

5. Security Considerations

This whole document is about security. In the RTCWEB effort, we are hoping to provide a browser based real-time communication platform that can be trusted and used by Internet users worldwide. The privacy of the browser to browser media path should be our most important concern. Choosing the wrong media security approach will hurt users of the Internet and limit the usefulness of the HTML5 RTCWEB extensions. It is the hope of the authors that the IETF will take this responsibility seriously and give users of RTCWEB the best options for media security and privacy.

6. Informative References

- [I-D.alvestrand-dispatch-rtcweb-protocols]
Alvestrand, H., "Overview: Real Time Protocols for Brower-based Applications",
draft-alvestrand-dispatch-rtcweb-protocols-01 (work in progress), March 2011.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC6189] Zimmermann, P., Johnston, A., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, April 2011.
- [RFC5479] Wing, D., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, April 2009.
- [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, May 2010.
- [RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session

Initiation Protocol (SIP)", RFC 4474, August 2006.

Authors' Addresses

Alan Johnston
Avaya
St. Louis, MO 63124

Email: alan.b.johnston@gmail.com

Philip Zimmermann
Zfone Project

Email: prz@mit.edu

RTCWEB
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2012

M. Kaufman
Skype
June 30, 2011

Client Security User Interface Requirements for RTCWEB
draft-kaufman-rtcweb-security-ui-00

Abstract

This document calls for a requirement to be imposed on RTCWEB client user interfaces whereby the user may inspect the current media security status.

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 January 1, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3

2. Security Inspector Requirements for Clients 3

3. Other Advantages 4

4. Security Considerations 4

Author's Address 4

1. Introduction

RTCWEB clients - including, but not limited to web browsers - should transmit and receive audio and video media over an encrypted channel whenever practical. It is important for a user to be able to determine the level of security provided for the currently-active media channel(s). This document provides a set of requirements that - if implemented - provide the user with that ability.

2. Security Inspector Requirements for Clients

A client **MUST** provide a user interface through which a user may determine the security characteristics for the currently-audible audio stream(s).

A client **MUST** provide a user interface through which a user may determine the security characteristics for currently-visible video stream(s).

A client **MUST** provide a user interface through which a user may determine the security characteristics for transmissions of their microphone audio.

A client **MUST** provide a user interface through which a user may determine the security characteristics for transmissions of their camera video.

The "security characteristics" **MUST** include an indication as to whether or not the transmission is encrypted, and if so, a brief description of the cipher in use. (For example: "AES-CBC" or "Null Cipher".)

If the transmission is encrypted, the "security characteristics" **MUST** include an indication as to the source of the keying material, particularly whether the keying material was delivered out-of-band (from a server) or was generated as a result of a pairwise negotiation.

If possible for the cryptosystem in use, the "security characteristics" **MUST** include information regarding the authenticity of the far station identity. (For example, in the case of a self-signed certificate with RSA key the contents of the certificate and the key fingerprint.)

If possible for the cryptosystem in use, the "security characteristics" **SHOULD** include a Short Authentication String which may be used by the user to authenticate the far station identity and

keying integrity (specifically, the presence or lack of a man-in-the-middle that may be in collusion with the service provider to attempt to bypass authentication tests) by communicating this string out-of-band with the far party.

If the transmission is encrypted, the "security characteristics" SHOULD indicate whether or not the keying algorithm is able to provide perfect forward secrecy.

In the case of a web browser client, the "display of security characteristics" MUST take the form of an inspection panel or dialog provided by the browser chrome, as any user interface rendered in-browser cannot be sufficiently trusted.

3. Other Advantages

In addition to the security advantages provided to users, this requirement will simplify debugging, particularly when building interoperable clients.

4. Security Considerations

These requirements enhance the communication security experienced by "interested users", that is to say users who are sufficiently careful that they utilize these mechanisms to actually inspect the security of their communications. Like the ability to inspect SSL certificates for HTTPS/TLS connections, this ability is of little use to those who do not actively choose to use it, but is critical to a subset of the user population.

Author's Address

Matthew Kaufman
Skype
3210 Porter Drive
Palo Alto, California 95060
US

Phone: +1 831 440 8771
Email: matthew.kaufman@skype.net

RTCWEB
Internet-Draft
Intended status: Standards Track
Expires: December 8, 2011

M. Kaufman
J. Rosenberg
Skype
June 6, 2011

NAT Traversal Requirements for RTCWEB
draft-kaufman-rtcweb-traversal-00

Abstract

This document describes a minimal set of requirements to enable NAT traversal (and satisfy one of the security requirements) for media channels within browser-based real-time communications (RTCWEB).

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 8, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Drawbacks of a Full ICE Implementation	3
2.1. Limits Adaptability	3
2.2. Hampers Innovation	4
2.3. Unneccesary Cost in some Cases	4
3. Proposed Model	5
4. Proposed API	6
4.1. Before Test	6
4.2. Send Test	6
4.3. Receipt of Test	7
4.4. Receipt of Response	7
5. Security Considerations	7
6. Informative References	8
Authors' Addresses	8

1. Introduction

RTCWEB clients - including, but not limited to web browsers - should be able to send and receive real-time media directly to and from other RTCWEB clients without sending the media through an application-layer intermediary. This will serve to reduce media latency, decrease packet loss, and reduce the operational cost of deploying the application.

There is general agreement that Interactivity Connectivity Establishment (ICE) RFC 5245 [RFC5245] represents a reasonable choice for meeting this need. ICE provides firewall and NAT traversal, creating direct peer-to-peer connections for media when possible, and falling back to media relays (typically established with TURN) when not possible.

Consequently, the natural inclination is to simply embed a full ICE implementation inside of the browser. However, there are drawbacks to doing so. This document proposes an alternative model, based on the concept of browser minimalism - embedding only the minimum necessary functionality into the browser itself, and then allowing application developers flexibility to use those tools as needed.

Section 2 first discusses the drawbacks of a full ICE implementation in the browser. Section 3 then outlines an alternative model where only STUN is present in the browser, and argues why it addresses the limitations discussed in Section 2. Section 4 then proposes a concrete extension to the PeerConnection API to enable STUN.

2. Drawbacks of a Full ICE Implementation

There are several drawbacks to including a full ICE implementation in the browser.

2.1. Limits Adaptability

ICE was not the IETFs first attempt at techniques for firewall and NAT traversal. Basic STUN [RFC3489] was defined in 2003, and it solved the problem by attempting to characterize NATs. It failed for a variety of reasons. However, one of the key lessons of STUN was that its technique for classifying NATs - breaking them into four different NAT varieties - proved brittle. In reality, the market saw changes in the types of implementations, and NATs appeared which met none of the classifications. For this reason, ICE abandoned the classification approach and instead moved towards a model of connectivity checking.

As a consequence, ICE has greater reliability than pure STUN, but its effectiveness in achieving direct p2p connections is still based on some underlying assumptions around NAT types. Its design is most effective for NATs whose behavior is endpoint-independent mapping, and whose filtering policy is either endpoint-independent or address-dependent [RFC4787].

With the ongoing exhaustion of the IPv4 address space, we can anticipate even further reliance on NAT and the likely appearance of carrier NATs of differing varieties. This is likely to change the nature of NAT behaviors seen in the real world. The right way to deal with this is to adapt ICE's behavior, using differing allocation techniques and assigning different priorities. For example, ICE currently does not enable direct p2p connections in cases where NATs have mapping policies which are endpoint dependent but utilize sequential port allocation. If, despite the recommendations of RFC4787, such NAT types become increasingly prevalent, ICE's effectiveness will decline and more connections will be relayed. With ICE literally baked into web browsers, it will become harder to adapt its algorithms to work best under the conditions of the modern Internet.

2.2. Hampers Innovation

One of the benefits of ICE is that it allows local implementation flexibility in the way candidates are gathered, offered and prioritized. However, once ICE is baked into the browser, it is no longer possible for that innovation to take place - or at least, it leaves the hands of the voice application providers. To date, there has been variability in this aspect of implementation, with different providers tuning it to tweak their needs and deployments.

2.3. Unnecesary Cost in some Cases

There is a broad array of use cases for VoIP. It is used for everything from consumer Internet services (like Skype) to small business phone systems. Though clearly global consumer Internet services require the kind of traversal technology provided by full ICE, it is not needed in other cases. One such use case is, in fact, enterprise telephony, where users make calls within the confines of their corporate network, and remote access is supported through VPN. Today, VoIP endpoints in these environments do not generally use ICE.

As such, if an enterprise communications application wanted to utilize browser RTC, it would need to support ICE even though it was not strictly required. Is there a penalty to support of ICE? The enterprise would need to deploy STUN and TURN servers, which would not actually be needed. ICE also typically increases call setup

delay (though the degree to which it does it is dependent on the network conditions the users are in), those increases would be for no benefit in the enterprise deployment scenario.

3. Proposed Model

The model proposed here is that the browser itself support STUN only. APIs are provided which allow for initiation of a STUN transaction. The results of this transaction are then passed to the browser application (notably, the reflexive address). The browser API allows the browser application to set attribute/value pairs in the message. Similarly, on the receive-side, APIs are defined for allowing an application to register callbacks for receipt of a STUN request. Those callbacks provide the application information on the source IP and port, amongst other information.

For security purposes, the browser will refuse to send, or accept, media to or from a peer to which a STUN transaction has not completed successfully. This ensures that the browser cannot be used as a DoS tool to launch a voice hammer attack.

What about TURN? In this model, TURN is mostly implemented on top of the browsers STUN implementation. The Javascript code in the browser can generate Allocate requests, and be informed of the results. The only exception to this is that the browser has to be told whether or not to encapsulate media in Send transactions, or to use an allocated channel. The browser API provides a switch which allows the application to tell the browser which encapsulation to use for media.

In a server-mediated environment, TURN might also be unnecessary. A call setup service can communicate directly with the relay service to establish a transparent UDP tunnel through one or more relays, the STUN connectivity checks may be sent through this tunnel, and no TURN encapsulation support is needed in the browser. The Javascript-initiated STUN connectivity tests may also be used to authenticate the browser to the tunnel service.

With this model, there is now a great deal of flexibility in how NAT traversal can be done. Some of the models which can now be supported are:

ICE in Javascript: A full ICE implementation is possible in Javascript itself. Because the implementation resides in Javascript, it is trivially changed at any time.

Server-Based ICE: A full ICE implementation can execute in the server, using remote-control commands to inform the browser to send STUN transactions, and passing the results from the browser back to the server. In essence - MGCP for ICE.

STUN-Only: For deployments where the peer is always publically reachable from clients - such as enterprises or PSTN termination services - the Javascript can do a single STUN transaction to create a permission in the browser, and then proceed to send media.

Non-ICE: Protocols similar to ICE, but not otherwise compliant, can also be implemented. Negotiation of which NAT traversal mechanism is needed, is done by the application outside of the browser.

This model addresses all of the concerns outlined in Section 2. Now, if changes in NAT types occur over time, new Javascript or server code can be deployed which uses different prioritizations, or even performs new allocation models. For example, port-predictive allocations can be added in this model, without upgrading the browser. Since the browser has the barest minimum necessary for security and functional purposes, innovation is possible to a greater degree. Finally, implementations can be only as complex as is needed for the task at hand.

4. Proposed API

The following makes the assumption that a PeerConnection object exists and is bound to a single local UDP port.

4.1. Before Test

There must be an API which allows the PeerConnection's local credentials to be determined, and a way to send these via a signaling service to the other party. The browser SHOULD generate the credentials itself and provide an API for read-only access.

4.2. Send Test

A function must be provided in order to initiate a STUN connectivity test. This function MUST allow the specification of a far address and port number, the far username and password, and (if necessary) additional ICE attributes to be included in the STUN Binding Request message. This function causes a single STUN RFC 3489 [RFC3489] Binding Request with short-term credentials to be sent to the far address from the initiating client. The client MUST enforce a rate limit of the transmission of these requests. Username concatenation

is performed as per ICE 7.1.2.3. The client (in the case of a browser) MUST NOT allow the user of the API to specify or examine the transaction ID for this request, in order to prevent spoofing of successful replies from an attacking host. Using a STUN request ensures that the packet will begin with the STUN magic cookie, and therefore is very unlikely to simulate other traffic. The STUN request is sent from the exact same IP address and port that the PeerConnection object will use for subsequent media traffic.

4.3. Receipt of Test

Upon receipt of a STUN Binding Request with valid credentials, the responding client SHOULD automatically generate and send the STUN transaction response. (If it does not, an API for sending the transaction response MUST be provided.) The responding client MUST also locally call a callback function that delivers the attribute/value pairs received in the Binding Request as well as the locally derived (reflexive) address from which the Binding Request was received.

4.4. Receipt of Response

Upon receipt of a valid STUN transaction response from the responding client, the initiating client MUST call a callback function that delivers the attribute/value pairs received in the response, one of which is the reflexive address. The response MUST be ignored if the receivedSocketAddress does not match the socket address to which the matching transaction ID was sent, as per ICE 7.1.3.2. Upon receipt of a valid response the client also adds the now-verified address to the Transmit Whitelist, a list of socket addresses to which sending of media is now permissible. The client MUST NOT allow media to be sent to any address/port combination that has not been added to the Transmit Whitelist. Note that the client must appropriately time out any state associated with pending tests.

5. Security Considerations

The Transmit Whitelist function serves to prevent a client from sending media to an endpoint which has not properly responded to a STUN request.

The requirement that the client internally generate the transaction ID and not allow it to be explicitly set or read back prevents spoofing of the STUN test replies.

6. Informative References

- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

Authors' Addresses

Matthew Kaufman
Skype
3210 Porter Drive
Palo Alto, California 94304
US

Phone: +1 831 440 8771
Email: matthew.kaufman@skype.net

Jonathan Rosenberg
Skype
3210 Porter Drive
Palo Alto, California 94304
US

Email: jdrosen@skype.net

RTCWEB
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2012

M. Kaufman
Skype
June 30, 2011

draft-kaufman-rtp-compatible-data-00

Abstract

This document describes a method for sending unreliable datagrams that are "compatible" with RTP and RTCP, and STUN usage of a shared UDP address and port.

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 January 1, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Constraints	3
3. Protocol Description	3
4. Alternatives Considered	4
5. Security Considerations	4
6. Informative References	5
Author's Address	5

1. Introduction

There is a desire to support the transmission of unreliable datagrams to a UDP receiver that is also receiving RTP and RTCP [RFC3550] (and STUN [RFC5389]) datagrams at the same address and port. One possible application is to add this ability to RTCWEB clients for direct peer-to-peer transmission of arbitrary data between clients.

The desire to share the same UDP port is driven, in part, by the reduction in the number of NAT bindings that would be required to simultaneously support media and data transmission. It is also driven by the desire to reuse STUN [RFC5389] connectivity checks and ICE [RFC5245] as the mechanism whereby receiver consent is obtained (confirmation that the receiver is willing to receive data from this sender) and as the mechanism for NAT traversal.

2. Constraints

The format of these datagrams have several constraints:

RTP: These datagrams must not be decoded as RTP datagrams.

RTCP: These datagrams must not be decoded as RTCP datagrams.

Future Versions: These datagrams must not prevent the future deployment of new versions of RTP or RTCP.

STUN: These datagrams must not be decoded as valid STUN messages.

3. Protocol Description

All such messages MUST start with a 4-byte header as follows:

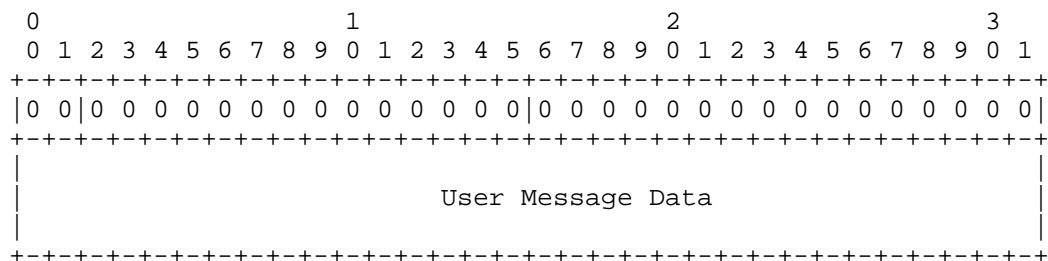


Figure 1

The most significant 2 bits of every such message MUST be zero. This can be used to differentiate these messages from all protocols other than STUN that are multiplexed on the same port (e.g., RTP, RTCP).

The next 14 bits of every such message MUST be zero. If interpreted by a STUN implementation, this will encode a message type of 0x000, which is reserved, and class (if using RFC5389) of 0b00 (Request). The reserved message type is not an allowed STUN message, thus differentiating such messages from STUN messages.

In order to further prevent STUN implementations from interpreting these datagrams the next 16 bits of every such message MUST be zero. This results in a field of all zeros at the position of the STUN length field. This ensures that even if the arbitrary data that follows were to simulate the STUN cookie (which is unlikely) the STUN receiver would find no attributes.

The choice of setting the high two bits to zero does require that the receiver that wishes to receive these datagrams must intercept them prior to them being received and then discarded by the STUN implementation, if present.

4. Alternatives Considered

One alternative considered was to encapsulate the arbitrary datagrams within RTP as an RTP payload type. However the semantics associated with RTP are not always appropriate, depending on the type of arbitrary data being transmitted.

5. Security Considerations

The contents of these messages SHOULD be encrypted. Reuse of the keying used for an associated RTP session that is using the same IP address and port MAY be an acceptable method of key specification and cryptosystem choice.

These messages MUST only be sent by RTCWEB clients that have already verified that the recipient is willing to receive data via a mechanism such as ICE.

If consent is received and because these messages have an inflexible (that is to say, not settable by an untrusted client such as a browser) header of 4 zero bytes, it is unlikely that these messages could be used to spoof other types of protocols, even if software running within an untrusted client (i.e., web browser) is able to specify the entire content of the message after the header. If this

guarantee is not strong enough, an additional header with a magic cookie similar to (but different in value from) the STUN cookie could be added.

Because these messages are transmitted as UDP datagrams, it is unlikely that proxy devices will misinterpret the contents of the messages, and thus masking techniques (such as those discussed for WebSockets) are probably not necessary even over unencrypted channels.

6. Informative References

- [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

Author's Address

Matthew Kaufman
Skype
3210 Porter Drive
Palo Alto, California 95060
US

Phone: +1 831 440 8771
Email: matthew.kaufman@skype.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 29, 2012

C. Perkins
University of Glasgow
M. Westerlund
Ericsson
J. Ott
Aalto University
August 28, 2011

RTP Requirements for RTC-Web
draft-perkins-rtcweb-rtp-usage-03

Abstract

This memo discusses use of RTP in the context of the RTC-Web activity. It discusses important features of RTP that need to be considered by other parts of the RTC-Web framework, describes which RTP profile to use in this environment, and outlines what RTP extensions should be supported.

This document is a candidate to become a work item of the RTCWEB working group as <WORKING GROUP DRAFT "MEDIA TRANSPORTS">.

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 February 29, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Expected Topologies	3
2. Requirements from RTP	6
2.1. Signalling for RTP sessions	6
2.2. (Lack of) Signalling for Payload Format Changes	7
3. RTP Profile	7
4. RTP and RTCP Guidelines	8
5. RTP Optimisations	8
5.1. RTP and RTCP Multiplexing	8
5.2. Reduced Size RTCP	8
5.3. Symmetric RTP/RTCP	9
5.4. Generation of the RTCP Canonical Name (CNAME)	9
6. RTP Extensions	10
6.1. RTP Conferencing Extensions	10
6.1.1. Full Intra Request	11
6.1.2. Picture Loss Indication	11
6.1.3. Slice Loss Indication	11
6.1.4. Reference Picture Selection Indication	11
6.1.5. Temporary Maximum Media Stream Bit Rate Request	11
6.2. RTP Header Extensions	12
6.3. Rapid Synchronisation Extensions	13
6.4. Client to Mixer Audio Level	13
6.5. Mixer to Client Audio Level	13
7. Improving RTP Transport Robustness	13
7.1. RTP Retransmission	14
7.2. Forward Error Correction (FEC)	15
7.2.1. Basic Redundancy	15
7.2.2. Block Based	16
7.2.3. Recommendations for FEC	17
8. RTP Rate Control and Media Adaptation	17
9. RTP Performance Monitoring	18
10. IANA Considerations	18
11. Security Considerations	18
12. Acknowledgements	18
13. References	19
13.1. Normative References	19
13.2. Informative References	21
Authors' Addresses	21

1. Introduction

This memo discusses the Real-time Transport Protocol (RTP) [RFC3550] in the context of the RTC-Web activity. The work in the IETF Audio/Video Transport Working Group, and its successors, has been about providing building blocks for real-time multimedia transport, and has not specified who should use which building blocks. The selection of building blocks and functionalities can really only be done in the context of some application, for example RTC-Web. We have selected a set of RTP features and extensions that are suitable for a number of applications that fit the RTC-Web context. Thus, applications such as VoIP, audio and video conferencing, and on-demand multimedia streaming are considered. Applications that rely on IP multicast have not been considered likely to be applicable to RTC-Web, thus extensions related to multicast have been excluded. We believe that RTC-Web will greatly benefit in interoperability if a reasonable set of RTP functionalities and extensions are selected. This memo is intended as a starting point for discussion of those features in the RTC-Web framework.

This memo is structured into different topics. For each topic, one or several recommendations from the authors are given. When it comes to the importance of extensions, or the need for implementation support, we use three requirement levels to indicate the importance of the feature to the RTC-Web specification:

REQUIRED: Functionality that is absolutely needed to make the RTC-Web solution work well, or functionality of low complexity that provides high value.

RECOMMENDED: Should be included as it brings significant benefit, but the solution can potentially work without it.

OPTIONAL: Something that is useful in some cases, but not always a benefit.

When this memo discusses RTP, it includes the RTP Control Protocol (RTCP) unless explicitly stated otherwise. RTCP is a fundamental and integral part of the RTP protocol, and is **REQUIRED** to be implemented.

1.1. Expected Topologies

As RTC-Web is focused on peer to peer connections established from clients in web browsers the following topologies further discussed in RTP Topologies [RFC5117] are primarily considered. The topologies are depicted and briefly explained here for ease of the reader.

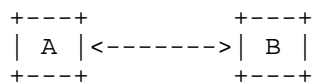


Figure 1: Point to Point

The point to point topology (Figure 1) is going to be very common in any single user to single user applications.

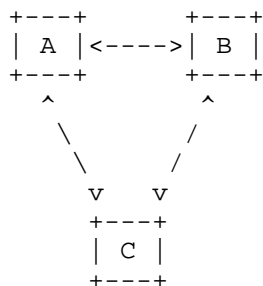


Figure 2: Multi-unicast

For small multiparty sessions it is practical enough to create RTP sessions by letting every participant send individual unicast RTP/UDP flows to each of the other participants. This is called multi-unicast and is unfortunately not discussed in the RTP Topologies [RFC5117]. This topology has the benefit of not requiring central nodes. The downside is that it increases the used bandwidth at each sender by requiring one copy of the media streams for each participant that are part of the same session beyond the sender itself. Thus this is limited to scenarios with few end-points unless the media is very low bandwidth.

It needs to be noted that, if this topology is to be supported by the RTC-Web framework, it needs to be possible to connect one RTP session to multiple established peer to peer flows that are individually established.

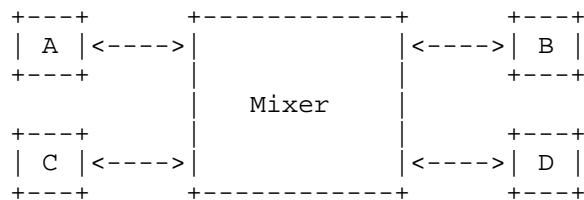


Figure 3: RTP Mixer with Only Unicast Paths

An RTP mixer (Figure 3) is a centralised point that selects or mixes content in a conference to optimise the RTP session so that each end-point only needs connect to one entity, the mixer. The mixer also reduces the bit-rate needs as the media sent from the mixer to the end-point can be optimised in different ways. These optimisations include methods like only choosing media from the currently most active speaker or mixing together audio so that only one audio stream is required in stead of 3 in the depicted scenario. The downside of the mixer is that someone is required to provide the actual mixer.

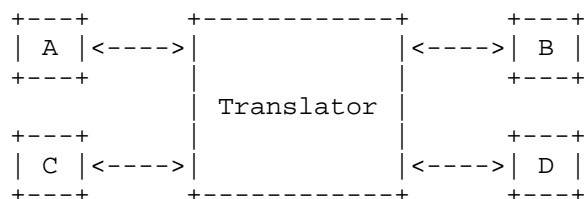


Figure 4: RTP Translator (Relay) with Only Unicast Paths

If one wants a less complex central node it is possible to use an relay (called an Transport Translator) (Figure 4) that takes on the role of forwarding the media to the other end-points but doesn't perform any media processing. It simply forwards the media from all other to all the other. Thus one endpoint A will only need to send a media once to the relay, but it will still receive 3 RTP streams with the media if B, C and D all currently transmits.

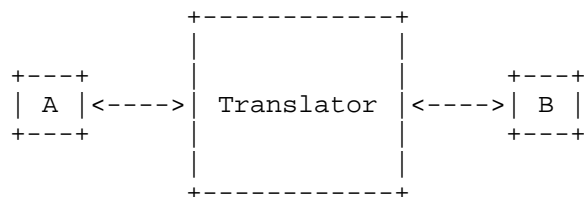


Figure 5: Translator towards Legacy end-point

To support legacy end-point (B) that don't fulfil the requirements of RTC-Web it is possible to insert a Translator (Figure 5) that takes on the role to ensure that from A's perspective B looks like a fully compliant end-point. Thus it is the combination of the Translator and B that looks like the end-point B. The intention is that the presence of the translator is transparent to A, however it is not certain that is possible. Thus this case is include so that it can be discussed if any mechanism specified to be used for RTC-Web results in such issues and how to handle them.

2. Requirements from RTP

This section discusses some requirements RTP and RTCP [RFC3550] place on their underlying transport protocol, the signalling channel, etc.

2.1. Signalling for RTP sessions

RTP is built with the assumption of an external to RTP/RTCP signalling channel to configure the RTP sessions and its functions. The basic configuration of an RTP session consists of the following parameters:

RTP Profile: The name of the RTP profile to be used in session. The RTP/AVP [RFC3551] and RTP/AVPF [RFC4585] profiles can interoperate on basic level, as can their secure variants RTP/SAVP [RFC3711] and RTP/SAVPF [RFC5124]. The secure variants of the profiles do not directly interoperate with the non-secure variants, due to the presence of additional header fields in addition to any cryptographic transformation of the packet content.

Transport Information: Source and destination address(s) and ports for RTP and RTCP must be signalled for each RTP session. If RTP and RTCP multiplexing [RFC5761] is to be used, such that a single port is used for RTP and RTCP flows, this must be signalled.

RTP Payload Types, media formats, and media format parameters: The mapping between media type names (and hence the RTP payload formats to be used) and the RTP payload type numbers must be signalled. Each media type may also have a number of media type parameters that must also be signalled to configure the codec and RTP payload format (the "a=fmtp:" line from SDP).

RTP Extensions: The RTP extensions one intends to use need to be agreed upon, including any parameters for each respective extension. At the very least, this will help avoiding using bandwidth for features that the other end-point will ignore. But for certain mechanisms there is requirement for this to happen as interoperability failure otherwise happens.

RTCP Bandwidth: Support for exchanging RTCP Bandwidth values to the end-points will be necessary, as described in "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth" [RFC3556], or something semantically equivalent. This also ensures that the end-points have a common view of the RTCP bandwidth, this is important as too different view of the bandwidths may lead to failure to interoperate.

These parameters are often expressed in SDP messages conveyed within

an offer/answer exchange. RTP does not depend on SDP or on the offer/answer model, but does require all the necessary parameters to be agreed somehow, and provided to the RTP implementation. We note that in RTCWEB context it will depend on the signalling model and API how these parameters need to be configured but they will be need to either set in the API or explicitly signalled between the peers.

2.2. (Lack of) Signalling for Payload Format Changes

As discussed in Section 2.1, the mapping between media type name, and its associated RTP payload format, and the RTP payload type number to be used for that format must be signalled as part of the session setup. An endpoint may signal support for multiple media formats, or multiple configurations of a single format, each using a different RTP payload type number. If multiple formats are signalled by an endpoint, that endpoint is REQUIRED to be prepared to receive data encoded in any of those formats at any time. RTP does not require advance signalling for changes between formats that were signalled during the session setup. This is needed for rapid rate adaptation.

3. RTP Profile

The "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)" [RFC5124] is REQUIRED to be implemented. This builds on the basic RTP/AVP profile [RFC3551], the RTP/AVPF feedback profile [RFC4585], and the secure RTP/SAVP profile [RFC3711].

The RTP/AVPF part of RTP/SAVPF is required to get the improved RTCP timer model, that allows more flexible transmission of RTCP packets in response to events, rather than strictly according to bandwidth. This also saves RTCP bandwidth and will commonly only use the full amount when there is a lot of events on which to send feedback. This functionality is needed to make use of the RTP conferencing extensions discussed in Section 6.1.

The RTP/SAVP part of RTP/SAVPF is for support for Secure RTP (SRTP) [RFC3711]. This provides media encryption, integrity protection, replay protection and a limited form of source authentication. It does not contain a specific keying mechanism, so that, and the set of security transforms, will be required to be chosen. It is possible that a security mechanism operating on a lower layer than RTP can be used instead and that should be evaluated. However, the reasons for the design of SRTP should be taken into consideration in that discussion.

4. RTP and RTCP Guidelines

RTP and RTCP are two flexible and extensible protocols that allow, on the one hand, choosing from a variety of building blocks and combining those to meet application needs, and on the other hand, create extensions where existing mechanisms are not sufficient: from new payload formats to RTP extension headers to additional RTCP control packets.

Different informational documents provide guidelines to the use and particularly the extension of RTP and RTCP, including the following: Guidelines for Writers of RTP Payload Format Specifications [RFC2736] and Guidelines for Extending the RTP Control Protocol [RFC5968].

5. RTP Optimisations

This section discusses some optimisations that makes RTP/RTCP work better and more efficient and therefore are considered.

5.1. RTP and RTCP Multiplexing

Historically, RTP and RTCP have been run on separate UDP ports. With the increased use of Network Address/Port Translation (NAPT) this has become problematic, since maintaining multiple NAT bindings can be costly. It also complicates firewall administration, since multiple ports must be opened to allow RTP traffic. To reduce these costs and session setup times, support for multiplexing RTP data packets and RTCP control packets on a single port [RFC5761] is REQUIRED. Supporting this specification is generally a simplification in code, since it relaxes the tests in [RFC3550].

Note that the use of RTP and RTCP multiplexed on a single port ensures that there is occasional traffic sent on that port, even if there is no active media traffic. This may be useful to keep-alive NAT bindings.

5.2. Reduced Size RTCP

RTCP packets are usually sent as compound RTCP packets; and RFC 3550 demands that those compound packets always start with an SR or RR packet. However, especially when using frequent feedback messages, these general statistics are not needed in every packet and unnecessarily increase the mean RTCP packet size and thus limit the frequency at which RTCP packets can be sent within the RTCP bandwidth share.

RFC5506 "Support for Reduced-Size Real-Time Transport Control

Protocol (RTCP): Opportunities and Consequences" [RFC5506] specifies how to reduce the mean RTCP message and allow for more frequent feedback. Frequent feedback, in turn, is essential to make real-time application quickly aware of changing network conditions and allow them to adapt their transmission and encoding behaviour.

Support for RFC5506 is REQUIRED.

5.3. Symmetric RTP/RTCP

RTP entities choose the RTP and RTCP transport addresses, i.e., IP addresses and port numbers, to receive packets on and bind their respective sockets to those. When sending RTP packets, however, they may use a different IP address or port number for RTP, RTCP, or both; e.g., when using a different socket instance for sending and for receiving. Symmetric RTP/RTCP requires that the IP address and port number for sending and receiving RTP/RTCP packets are identical.

The reasons for using symmetric RTP is primarily to avoid issues with NAT and Firewalls by ensuring that the flow is actually bi-directional and thus kept alive and registered as flow the intended recipient actually wants. In addition it saves resources in the form of ports at the end-points, but also in the network as NAT mappings or firewall state is not unnecessary bloated. Also the number of QoS state are reduced.

Using Symmetric RTP and RTCP [RFC4961] is REQUIRED.

5.4. Generation of the RTCP Canonical Name (CNAME)

The RTCP Canonical Name (CNAME) provides a persistent transport-level identifier for an RTP endpoint. While the Synchronisation Source (SSRC) identifier for an RTP endpoint may change if a collision is detected, or when the RTP application is restarted, it's RTCP CNAME is meant to stay unchanged, so that RTP endpoints can be uniquely identified and associated with their RTP media streams. For proper functionality, RTCP CNAMEs should be unique among the participants of an RTP session.

The RTP specification [RFC3550] includes guidelines for choosing a unique RTP CNAME, but these are not sufficient in the presence of NAT devices. In addition, some may find long-term persistent identifiers problematic from a privacy viewpoint. Accordingly, support for generating a short-term persistent RTCP CNAMEs following method (b) as specified in Section 4.2 of "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)" [RFC6222] is RECOMMENDED, since this addresses both concerns.

6. RTP Extensions

There are a number of RTP extensions that could be very useful in the RTC-Web context. One set is related to conferencing, others are more generic in nature.

6.1. RTP Conferencing Extensions

RTP is inherently defined for group communications, whether using IP multicast, multi-unicast, or based on a centralised server. In today's practice, however, overlay-based conferencing dominates, typically using one or a few so-called conference bridges or servers to connect endpoints in a star or flat tree topology. Quite diverse conferencing topologies can be created using the basic elements of RTP mixers and translators as defined in RFC 3550.

An number of conferencing topologies are defined in [RFC5117] out of the which the following ones are the more common (and most likely in practice workable) ones:

- 1) RTP Translator (Relay) with Only Unicast Paths (RFC 5117, section 3.3)
- 2) RTP Mixer with Only Unicast Paths (RFC 5117, section 3.4)
- 3) Point to Multipoint Using a Video Switching MCU (RFC 5117, section 3.5)
- 4) Point to Multipoint Using Content Modifying MCUs (RFC 5117, section 3.6)

We note that 3 and 4 are not well utilising the functions of RTP and in some cases even violates the RTP specifications. Thus we recommend that one focus on 1 and 2.

RTP protocol extensions to be used with conferencing are included because they are important in the context of centralised conferencing, where one RTP Mixer (Conference Focus) receives a participants media streams and distribute them to the other participants. These messages are defined in the Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF) [RFC4585] and the "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)" (CCM) [RFC5104] and are fully usable by the Secure variant of this profile (RTP/SAVPF) [RFC5124].

6.1.1. Full Intra Request

The Full Intra Request is defined in Sections 3.5.1 and 4.3.1 of CCM [RFC5104]. It is used to have the mixer request from a session participants a new Intra picture. This is used when switching between sources to ensure that the receivers can decode the video or other predicted media encoding with long prediction chains. It is RECOMMENDED that this feedback message is supported.

6.1.2. Picture Loss Indication

The Picture Loss Indication is defined in Section 6.3.1 of the RTP/AVPF profile [RFC4585]. It is used by a receiver to tell the encoder that it lost the decoder context and would like to have it repaired somehow. This is semantically different from the Full Intra Request above. It is RECOMMENDED that this feedback message is supported as a loss tolerance mechanism.

6.1.3. Slice Loss Indication

The Slice Loss Indicator is defined in Section 6.3.2 of the RTP/AVPF profile [RFC4585]. It is used by a receiver to tell the encoder that it has detected the loss or corruption of one or more consecutive macroblocks, and would like to have these repaired somehow. The use of this feedback message is OPTIONAL as a loss tolerance mechanism.

6.1.4. Reference Picture Selection Indication

Reference Picture Selection Indication (RPSI) is defined in Section 6.3.3 of the RTP/AVPF profile [RFC4585]. Some video coding standards allow the use of older reference pictures than the most recent one for predictive coding. If such a codec is in used, and if the encoder has learned about a loss of encoder-decoder synchronicity, a known-as-correct reference picture can be used for future coding. The RPSI message allows this to be signalled. The use of this RTCP feedback message is OPTIONAL as a loss tolerance mechanism.

6.1.5. Temporary Maximum Media Stream Bit Rate Request

This feedback message is defined in Section 3.5.4 and 4.2.1 in CCM [RFC5104]. This message and its notification message is used by a media receiver, to inform the sending party that there is a current limitation on the amount of bandwidth available to this receiver. This can be for various reasons, and can for example be used by an RTP mixer to limit the media sender being forwarded by the mixer (without doing media transcoding) to fit the bottlenecks existing towards the other session participants. It is RECOMMENDED that this feedback message is supported.

6.2. RTP Header Extensions

The RTP specification [RFC3550] provides a capability to extend the RTP header with in-band data, but the format and semantics of the extensions are poorly specified. Accordingly, if header extensions are to be used, it is REQUIRED that they be formatted and signalled according to the general mechanism of RTP header extensions defined in [RFC5285].

As noted in [RFC5285], the requirement from the RTP specification that header extensions are "designed so that the header extension may be ignored" [RFC3550] stands. To be specific, header extensions must only be used for data that can safely be ignored by the recipient without affecting interoperability, and must not be used when the presence of the extension has changed the form or nature of the rest of the packet in a way that is not compatible with the way the stream is signalled (e.g., as defined by the payload type). Valid examples might include metadata that is additional to the usual RTP information.

The RTP rapid synchronisation header extension [RFC6051] is recommended, as discussed in Section 6.3 we also recommend the client to mixer audio level [I-D.ietf-avtext-client-to-mixer-audio-level], and consider the mixer to client audio level [I-D.ietf-avtext-mixer-to-client-audio-level] as optional feature.

It is RECOMMENDED that the mechanism to encrypt header extensions [I-D.ietf-avtcore-srtp-encrypted-header-ext] is implemented when the client-to-mixer and mixer-to-client audio level indications are in use in SRTP encrypted sessions, since the information contained in these header extensions may be considered sensitive.

Currently the other header extensions are not recommended to be included at this time. But we do include a list of the available ones for information below:

Transmission Time offsets: [RFC5450] defines a format for including an RTP timestamp offset of the actual transmission time of the RTP packet in relation to capture/display timestamp present in the RTP header. This can be used to improve jitter determination and buffer management.

Associating Time-Codes with RTP Streams: [RFC5484] defines how to associate SMPTE times codes with the RTP streams.

6.3. Rapid Synchronisation Extensions

Many RTP sessions require synchronisation between audio, video, and other content. This synchronisation is performed by receivers, using information contained in RTCP SR packets, as described in the RTP specification [RFC3550]. This basic mechanism can be slow, however, so it is RECOMMENDED that the rapid RTP synchronisation extensions described in [RFC6051] be implemented. The rapid synchronisation extensions use the general RTP header extension mechanism [RFC5285], which requires signalling, but are otherwise backwards compatible.

6.4. Client to Mixer Audio Level

The Client to Mixer Audio Level [I-D.ietf-avtext-client-to-mixer-audio-level] is an RTP header extension used by a client to inform a mixer about the level of audio activity in the packet the header is attached to. This enables a central node to make mixing or selection decisions without decoding or detailed inspection of the payload. Thus reducing the needed complexity in some types of central RTP nodes.

Assuming that the Client to Mixer Audio Level [I-D.ietf-avtext-client-to-mixer-audio-level] is published as a finished specification prior to RTCWEB's first RTP specification then it is RECOMMENDED that this extension is included.

6.5. Mixer to Client Audio Level

The Mixer to Client Audio Level header extension [I-D.ietf-avtext-mixer-to-client-audio-level] provides the client with the audio level of the different sources mixed into a common mix from the RTP mixer. Thus enabling a user interface to indicate the relative activity level of a session participant, rather than just being included or not based on the CSRC field. This is a pure optimisations of non critical functions and thus optional functionality.

Assuming that the Mixer to Client Audio Level [I-D.ietf-avtext-client-to-mixer-audio-level] is published as a finished specification prior to RTCWEB's first RTP specification then it is OPTIONAL that this extension is included.

7. Improving RTP Transport Robustness

There are some tools that can make RTP flows robust against Packet loss and reduce the impact on media quality. However they all add extra bits compared to a non-robust stream. These extra bits needs

to be considered and the aggregate bit-rate needs to be rate controlled. Thus improving robustness might require a lower base encoding quality but has the potential to give that quality with fewer errors in it.

7.1. RTP Retransmission

Support for RTP retransmission as defined by "RTP Retransmission Payload Format" [RFC4588] is RECOMMENDED.

The retransmission scheme in RTP allows flexible application of retransmissions. Only selected missing packets can be requested by the receiver. It also allows for the sender to prioritise between missing packets based on senders knowledge about their content. Compared to TCP, RTP retransmission also allows one to give up on a packet that despite retransmission(s) still has not been received within a time window.

"RTC-Web Media Transport Requirements" [I-D.cbran-rtcweb-data] raises two issues that they think makes RTP Retransmission unsuitable for RTCWEB. We here consider these issues and explain why they are in fact not a reason to exclude RTP retransmission from the tool box available to RTCWEB media sessions.

The additional latency added by [RFC4588] will exceed the latency threshold for interactive voice and video: RTP Retransmission will require at least one round trip time for a retransmission request and repair packet to arrive. Thus the general suitability of using retransmissions will depend on the actual network path latency between the end-points. In many of the actual usages the latency between two end-points will be low enough for RTP retransmission to be effective. Interactive communication with end-to-end delays of 400 ms still provide a fair quality. Even removing half of that in end-point delays allows functional retransmission between end-points on the continent. In addition in some applications one may accept temporary delay spikes to allow for retransmission of crucial codec information such as parameter sets, intra picture etc, rather than getting no media at all.

The undesirable increase in packet transmission at the point when congestion occurs: Congestion loss will impact the rate controls view of available bit-rate for transmission. When using retransmission one will have to prioritise between performing retransmissions and the quality one can achieve with ones adaptable codecs. In many use cases one prefer error free or low rates of error with reduced base quality over high degrees of error at a higher base quality.

The RTCWEB end-point implementations will need to both select when to enable RTP retransmissions based on API settings and measurements of the actual round trip time. In addition for each NACK request that a media sender receives it will need to make a prioritisation based on the importance of the requested media, the probability that the packet will reach the receiver in time for being usable, the consumption of available bit-rate and the impact of the media quality for new encodings.

To conclude, the issues raised are implementation concerns that an implementation needs to take into consideration, they are not arguments against including a highly versatile and efficient packet loss repair mechanism.

7.2. Forward Error Correction (FEC)

Support of some type of FEC to combat the effects of packet loss is beneficial, but is heavily application dependent. However, some FEC mechanisms are encumbered.

The main benefit from FEC is the relatively low additional delay needed to protect against packet losses. The transmission of any repair packets should preferably be done with a time delay that is just larger than any loss events normally encountered. That way the repair packet isn't also lost in the same event as the source data.

The amount of repair packets needed are also highly dynamically and depends on two main factors, the amount and pattern of lost packets to be recovered and the mechanism one use to derive repair data. The later choice also effects the the additional delay required to both encode the repair packets and in the receiver to be able to recover the lost packet(s).

7.2.1. Basic Redundancy

The method for providing basic redundancy is to simply retransmit an some time earlier sent packet. This is relatively simple in theory, i.e. one saves any outgoing source (original) packet in a buffer marked with a timestamp of actual transmission, some X ms later one transmit this packet again. Where X is selected to be longer than the common loss events. Thus any loss events shorter than X can be recovered assuming that one doesn't get an another loss event before all the packets lost in the first event has been received.

The downside of basic redundancy is the overhead. To provide each packet with once chance of recovery, then the transmission rate increases with 100% as one needs to send each packet twice. It is possible to only redundantly send really important packets thus

reducing the overhead below 100% for some other trade-off is overhead.

In addition the basic retransmission of the same packet using the same SSRC in the same RTP session is not possible in RTP context. The reason is that one would then destroy the RTCP reporting if one sends the same packet twice with the same sequence number. Thus one needs more elaborate mechanisms.

RTP Payload for Redundant Audio Data: This audio and text redundancy format defined in [RFC2198] allows for multiple levels of redundancy with different delay in their transmissions, as long as the source plus payload parts to be redundantly transmitted together fits into one MTU. This should work fine for most interactive use cases as both the codec bit-rates and the framing intervals normally allow for this requirement to hold. This payload format also don't increase the packet rate, as original data and redundant data are sent together. This format does not allow perfect recovery, only recovery of information deemed necessary for audio, for example the sequence number of the original data is lost.

RTP Retransmission Format: The RTP Retransmission Payload format [RFC4588] can be used to pro-actively send redundant packets using either SSRC or session multiplexing. By using different SSRCs or a different session for the redundant packets the RTCP receiver reports will be correct. The retransmission payload format is used to recover the packets original data thus enabling a perfect recovery.

Duplication Grouping Semantics in the Session Description Protocol: This [I-D.begen-mmusic-redundancy-grouping] is proposal for new SDP signalling to indicate media stream duplication using different RTP sessions, or different SSRCs to separate the source and the redundant copy of the stream.

7.2.2. Block Based

Block based redundancy collects a number of source packets into a data block for processing. The processing results in some number of repair packets that is then transmitted to the other end allowing the receiver to attempt to recover some number of lost packets in the block. The benefit of block based approaches is the overhead which can be lower than 100% and still recover one or more lost source packet from the block. The optimal block codes allows for each received repair packet to repair a single loss within the block. Thus 3 repair packets that are received should allow for any set of 3 packets within the block to be recovered. In reality one commonly

don't reach this level of performance for any block sizes and number of repair packets, and taking the computational complexity into account there are even more trade-offs to make among the codes.

One result of the block based approach is the extra delay, as one needs to collect enough data together before being able to calculate the repair packets. In addition sufficient amount of the block needs to be received prior to recovery. Thus additional delay are added on both sending and receiving side to ensure possibility to recover any packet within the block.

The redundancy overhead and the transmission pattern of source and repair data can be altered from block to block, thus allowing a adaptive process adjusting to meet the actual amount of loss seen on the network path and reported in RTCP.

The alternatives that exist for block based FEC with RTP are the following:

RTP Payload Format for Generic Forward Error Correction: This RTP payload format [RFC5109] defines an XOR based recovery packet. This is the simplest processing wise that an block based FEC scheme can be. It also results in some limited properties, as each repair packet can only repair a single loss. To handle multiple close losses a scheme of hierarchical encodings are need. Thus increasing the overhead significantly.

Forward Error Correction (FEC) Framework: This framework [I-D.ietf-fecframe-framework] defines how not only RTP packets but how arbitrary packet flows can be protected. Some solutions produced or under development in FECFRAME WG are RTP specific. There exist alternatives supporting block codes such as Reed-Salomon and Raptor.

7.2.3. Recommendations for FEC

(tbd)

8. RTP Rate Control and Media Adaptation

It is REQUIRED to have an RTP Rate Control mechanism using Media adaptation to ensure that the generated RTP flows are network friendly, and maintain the user experience in the presence of network problems.

The biggest issue is that there are no standardised and ready to use mechanism that can simply be included in RTC-Web. Thus there will be

need for the IETF to produce such a specification. A potential starting point for defining a solution is "RTP with TCP Friendly Rate Control" [rtp-tfrc].

9. RTP Performance Monitoring

RTCP does contains a basic set of RTP flow monitoring points like packet loss and jitter. There exist a number of extensions that could be included in the set to be supported. However, in most cases which RTP monitoring that is needed depends on the application, which makes it difficult to select which to include when the set of applications is very large.

10. IANA Considerations

This memo makes no request of IANA.

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

11. Security Considerations

RTP and its various extensions each have their own security considerations. These should be taken into account when considering the security properties of the complete suite. We currently don't think this suite creates any additional security issues or properties. The use of SRTP [RFC3711] will provide protection or mitigation against all the fundamental issues by offering confidentiality, integrity and partial source authentication. The guidelines in [I-D.ietf-avtcore-srtp-vbr-audio] apply when using variable bit rate (VBR) audio codecs, for example Opus.

We don't discuss the key-management aspect of SRTP in this memo, that needs to be done taking the RTC-Web communication model into account.

In the context of RTC-Web the actual security properties required from RTP are currently not fully understood. Until security goals and requirements are specified it will be difficult to determine what security features in addition to SRTP and a suitable key-management, if any, that are needed.

12. Acknowledgements

13. References

13.1. Normative References

- [I-D.ietf-avtcore-srtp-encrypted-header-ext]
Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)",
draft-ietf-avtcore-srtp-encrypted-header-ext-00 (work in progress), June 2011.
- [I-D.ietf-avtcore-srtp-vbr-audio]
Perkins, C. and J. Valin, "Guidelines for the use of Variable Bit Rate Audio with Secure RTP",
draft-ietf-avtcore-srtp-vbr-audio-03 (work in progress), July 2011.
- [I-D.ietf-avtext-client-to-mixer-audio-level]
Lennox, J., Ivov, E., and E. Marocco, "A Real-Time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication",
draft-ietf-avtext-client-to-mixer-audio-level-03 (work in progress), July 2011.
- [I-D.ietf-avtext-mixer-to-client-audio-level]
Ivov, E., Marocco, E., and J. Lennox, "A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication",
draft-ietf-avtext-mixer-to-client-audio-level-03 (work in progress), July 2011.
- [RFC2736] Handley, M. and C. Perkins, "Guidelines for Writers of RTP Payload Format Specifications", BCP 36, RFC 2736, December 1999.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, 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.

- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, July 2007.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.
- [RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, March 2009.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

13.2. Informative References

- [I-D.begen-mmusic-redundancy-grouping]
Begen, A., Cai, Y., and H. Ou, "Duplication Grouping Semantics in the Session Description Protocol", draft-begen-mmusic-redundancy-grouping-01 (work in progress), June 2011.
- [I-D.cbran-rtcweb-data]
Bran, C. and C. Jennings, "RTC-Web Non-Media Data Transport Requirements", draft-cbran-rtcweb-data-00 (work in progress), July 2011.
- [I-D.ietf-fecframe-framework]
Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", draft-ietf-fecframe-framework-15 (work in progress), June 2011.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5968] Ott, J. and C. Perkins, "Guidelines for Extending the RTP Control Protocol (RTCP)", RFC 5968, September 2010.
- [rtp-tfrc]
Gharai, L., "RTP with TCP Friendly Rate Control (draft-gharai-avtcore-rtp-tfrc-00)", March 2011.

Authors' Addresses

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csp Perkins.org

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Joerg Ott
Aalto University
School of Electrical Engineering
Espoo 02150
Finland

Email: jorg.ott@aalto.fi

RTC-Web
Internet-Draft
Intended status: Standards Track
Expires: March 7, 2012

E. Rescorla
RTFM, Inc.
September 4, 2011

Security Considerations for RTC-Web
draft-rescorla-rtcweb-security-01

Abstract

The Real-Time Communications on the Web (RTC-Web) working group is tasked with standardizing protocols for real-time communications between Web browsers. The major use cases for RTC-Web technology are real-time audio and/or video calls, Web conferencing, and direct data transfer. Unlike most conventional real-time systems (e.g., SIP-based soft phones) RTC-Web communications are directly controlled by some Web server, which poses new security challenges. For instance, a Web browser might expose a JavaScript API which allows a server to place a video call. Unrestricted access to such an API would allow any site which a user visited to "bug" a user's computer, capturing any activity which passed in front of their camera. This document defines the RTC-Web threat model and defines an architecture which provides security within that threat model.

Legal

THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN ARE PROVIDED ON AN "AS IS" BASIS AND THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST, AND THE INTERNET ENGINEERING TASK FORCE, DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 7, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. Terminology	5
3. The Browser Threat Model	5
3.1. Access to Local Resources	6
3.2. Same Origin Policy	6
3.3. Bypassing SOP: CORS, WebSockets, and consent to communicate	7
4. Security for RTC-Web Applications	7
4.1. Access to Local Devices	7
4.1.1. Calling Scenarios and User Expectations	8
4.1.1.1. Dedicated Calling Services	8
4.1.1.2. Calling the Site You're On	8
4.1.1.3. Calling to an Ad Target	9
4.1.2. Origin-Based Security	9
4.1.3. Security Properties of the Calling Page	11
4.2. Communications Consent Verification	12
4.2.1. ICE	12
4.2.2. Masking	12
4.2.3. Backward Compatibility	12
4.3. Communications Security	13
4.3.1. Protecting Against Retrospective Compromise	14
4.3.2. Protecting Against During-Call Attack	15
4.3.2.1. Key Continuity	15
4.3.2.2. Short Authentication Strings	16
5. Security Considerations	17
6. Acknowledgements	17
7. References	17
7.1. Normative References	17
7.2. Informative References	17
Author's Address	19

1. Introduction

The Real-Time Communications on the Web (RTC-Web) working group is tasked with standardizing protocols for real-time communications between Web browsers. The major use cases for RTC-Web technology are real-time audio and/or video calls, Web conferencing, and direct data transfer. Unlike most conventional real-time systems, (e.g., SIP-based[RFC3261] soft phones) RTC-Web communications are directly controlled by some Web server. A simple case is shown below.

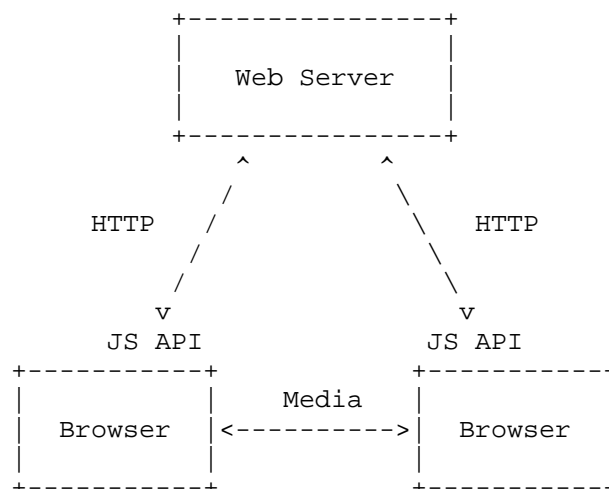


Figure 1: A simple RTC-Web system

In the system shown in Figure 1, Alice and Bob both have RTC-Web enabled browsers and they visit some Web server which operates a calling service. Each of their browsers exposes standardized JavaScript calling APIs which are used by the Web server to set up a call between Alice and Bob. While this system is topologically similar to a conventional SIP-based system (with the Web server acting as the signaling service and browsers acting as softphones), control has moved to the central Web server; the browser simply provides API points that are used by the calling service. As with any Web application, the Web server can move logic between the server and JavaScript in the browser, but regardless of where the code is executing, it is ultimately under control of the server.

It should be immediately apparent that this type of system poses new security challenges beyond those of a conventional VoIP system. In particular, it needs to contend with malicious calling services. For example, if the calling service can cause the browser to make a call at any time to any callee of its choice, then this facility can be

used to bug a user's computer without their knowledge, simply by placing a call to some recording service. More subtly, if the exposed APIs allow the server to instruct the browser to send arbitrary content, then they can be used to bypass firewalls or mount denial of service attacks. Any successful system will need to be resistant to this and other attacks.

2. 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 RFC 2119 [RFC2119].

3. The Browser Threat Model

The security requirements for RTC-Web follow directly from the requirement that the browser's job is to protect the user. Huang et al. [huang-w2sp] summarize the core browser security guarantee as:

Users can safely visit arbitrary web sites and execute scripts provided by those sites.

It is important to realize that this includes sites hosting arbitrary malicious scripts. The motivation for this requirement is simple: it is trivial for attackers to divert users to sites of their choice. For instance, an attacker can purchase display advertisements which direct the user (either automatically or via user clicking) to their site, at which point the browser will execute the attacker's scripts. Thus, it is important that it be safe to view arbitrarily malicious pages. Of course, browsers inevitably have bugs which cause them to fall short of this goal, but any new RTC-Web functionality must be designed with the intent to meet this standard. The remainder of this section provides more background on the existing Web security model.

In this model, then, the browser acts as a TRUSTED COMPUTING BASE (TCB) both from the user's perspective and to some extent from the server's. While HTML and JS provided by the server can cause the browser to execute a variety of actions, those scripts operate in a sandbox that isolates them both from the user's computer and from each other, as detailed below.

Conventionally, we refer to either WEB ATTACKERS, who are able to induce you to visit their sites but do not control the network, and NETWORK ATTACKERS, who are able to control your network. Network attackers correspond to the [RFC3552] "Internet Threat Model". In

general, it is desirable to build a system which is secure against both kinds of attackers, but realistically many sites do not run HTTPS [RFC2818] and so our ability to defend against network attackers is necessarily somewhat limited. Most of the rest of this section is devoted to web attackers, with the assumption that protection against network attackers is provided by running HTTPS.

3.1. Access to Local Resources

While the browser has access to local resources such as keying material, files, the camera and the microphone, it strictly limits or forbids web servers from accessing those same resources. For instance, while it is possible to produce an HTML form which will allow file upload, a script cannot do so without user consent and in fact cannot even suggest a specific file (e.g., /etc/passwd); the user must explicitly select the file and consent to its upload. [Note: in many cases browsers are explicitly designed to avoid dialogs with the semantics of "click here to screw yourself", as extensive research shows that users are prone to consent under such circumstances.]

Similarly, while Flash SWFs can access the camera and microphone, they explicitly require that the user consent to that access. In addition, some resources simply cannot be accessed from the browser at all. For instance, there is no real way to run specific executables directly from a script (though the user can of course be induced to download executable files and run them).

3.2. Same Origin Policy

Many other resources are accessible but isolated. For instance, while scripts are allowed to make HTTP requests via the XMLHttpRequest() API those requests are not allowed to be made to any server, but rather solely to the same ORIGIN from whence the script came. [I-D.abarth-origin] (although CORS [CORS] and WebSockets [I-D.ietf-hybi-thewebsocketprotocol] provides a escape hatch from this restriction, as described below. This SAME ORIGIN POLICY (SOP) prevents server A from mounting attacks on server B via the user's browser, which protects both the user (e.g., from misuse of his credentials) and the server (e.g., from DoS attack).

More generally, SOP forces scripts from each site to run in their own, isolated, sandboxes. While there are techniques to allow them to interact, those interactions generally must be mutually consensual (by each site) and are limited to certain channels. For instance, multiple pages/browser panes from the same origin can read each other's JS variables, but pages from the different origins--or even iframes from different origins on the same page--cannot.

3.3. Bypassing SOP: CORS, WebSockets, and consent to communicate

While SOP serves an important security function, it also makes it inconvenient to write certain classes of applications. In particular, mash-ups, in which a script from origin A uses resources from origin B, can only be achieved via a certain amount of hackery. The W3C Cross-Origin Resource Sharing (CORS) spec [CORS] is a response to this demand. In CORS, when a script from origin A executes what would otherwise be a forbidden cross-origin request, the browser instead contacts the target server to determine whether it is willing to allow cross-origin requests from A. If it is so willing, the browser then allows the request. This consent verification process is designed to safely allow cross-origin requests.

While CORS is designed to allow cross-origin HTTP requests, WebSockets [I-D.ietf-hybi-thewebsocketprotocol] allows cross-origin establishment of transparent channels. Once a WebSockets connection has been established from a script to a site, the script can exchange any traffic it likes without being required to frame it as a series of HTTP request/response transactions. As with CORS, a WebSockets transaction starts with a consent verification stage to avoid allowing scripts to simply send arbitrary data to another origin.

While consent verification is conceptually simple--just do a handshake before you start exchanging the real data--experience has shown that designing a correct consent verification system is difficult. In particular, Huang et al. [huang-w2sp] have shown vulnerabilities in the existing Java and Flash consent verification techniques and in a simplified version of the WebSockets handshake. In particular, it is important to be wary of CROSS-PROTOCOL attacks in which the attacking script generates traffic which is acceptable to some non-Web protocol state machine. In order to resist this form of attack, WebSockets incorporates a masking technique intended to randomize the bits on the wire, thus making it more difficult to generate traffic which resembles a given protocol.

4. Security for RTC-Web Applications

4.1. Access to Local Devices

As discussed in Section 1, allowing arbitrary sites to initiate calls violates the core Web security guarantee; without some access restrictions on local devices, any malicious site could simply bug a user. At minimum, then, it MUST NOT be possible for arbitrary sites to initiate calls to arbitrary location without user consent. This immediately raises the question, however, of what should be the scope

of user consent.

For the rest of this discussion we assume that the user is somehow going to grant consent to some entity (e.g., a social networking site) to initiate a call on his behalf. This consent may be limited to a single call or may be a general consent. In order for the user to make an intelligent decision about whether to allow a call (and hence his camera and microphone input to be routed somewhere), he must understand either who is request access, where the media is going, or both. So, for instance, one might imagine that at the time access to camera and microphone is requested, the user is shown a dialog that says "site X has requested access to camera and microphone, yes or no" (though note that this type of in-flow interface violates one of the guidelines in Section Section 3). The user's decision will of course be based on his opinion of Site X. However, as discussed below, this is a complicated concept.

4.1.1.1. Calling Scenarios and User Expectations

While a large number of possible calling scenarios are possible, the scenarios discussed in this section illustrate many of the difficulties of identifying the relevant scope of consent.

4.1.1.1.1. Dedicated Calling Services

The first scenario we consider is a dedicated calling service. In this case, the user has a relationship with a calling site and repeatedly makes calls on it. It is likely that rather than having to give permission for each call that the user will want to give the calling service long-term access to the camera and microphone. This is a natural fit for a long-term consent mechanism (e.g., installing an app store "application" to indicate permission for the calling service.) A variant of the dedicated calling service is a gaming site (e.g., a poker site) which hosts a dedicated calling service to allow players to call each other.

With any kind of service where the user may use the same service to talk to many different people, there is a question about whether the user can know who they are talking to. In general, this is difficult as most of the user interface is presented by the calling site. However, communications security mechanisms can be used to give some assurance, as described in Section 4.3.2.

4.1.1.1.2. Calling the Site You're On

Another simple scenario is calling the site you're actually visiting. The paradigmatic case here is the "click here to talk to a representative" windows that appear on many shopping sites. In this

case, the user's expectation is that they are calling the site they're actually visiting. However, it is unlikely that they want to provide a general consent to such a site; just because I want some information on a car doesn't mean that I want the car manufacturer to be able to activate my microphone whenever they please. Thus, this suggests the need for a second consent mechanism where I only grant consent for the duration of a given call. As described in Section 3.1, great care must be taken in the design of this interface to avoid the users just clicking through.

4.1.1.3. Calling to an Ad Target

In both of the previous cases, the user has a direct relationship (though perhaps a transient one) with the target of the call. Moreover, in both cases he is actually visiting the site of the person he is being asked to trust. However, this is not always so. Consider the case where a user is visiting a content site which hosts an advertisement with an invitation to call for more information. When the user clicks the ad, they are connected with the advertiser or their agent.

The relationships here are far more complicated: the site the user is actually visiting has no direct relationship with the advertiser; they are just hosting ads from an ad network. The user has no relationship with the ad network, but desires one with the advertiser, at least for long enough to learn about their products. At minimum, then, whatever consent dialog is shown needs to allow the user to have some idea of the organization that they are actually calling.

However, because the user also has some relationship with the hosting site, it is also arguable that the hosting site should be allowed to express an opinion (e.g., to be able to allow or forbid a call) since a bad experience with an advertiser reflect negatively on the hosting site [this idea was suggested by Adam Barth]. However, this obviously presents a privacy challenge, as sites which host advertisements often learn very little about whether individual users clicked through to the ads, or even which ads were presented.

4.1.2. Origin-Based Security

As discussed in Section 3.2, the basic unit of Web sandboxing is the origin, and so it is natural to scope consent to origin. Specifically, a script from origin A MUST only be allowed to initiate communications (and hence to access camera and microphone) if the user has specifically authorized access for that origin. It is of course technically possible to have coarser-scoped permissions, but because the Web model is scoped to origin, this creates a difficult

mismatch.

Arguably, origin is not fine-grained enough. Consider the situation where Alice visits a site and authorizes it to make a single call. If consent is expressed solely in terms of origin, then at any future visit to that site (including one induced via mash-up or ad network), the site can bug Alice's computer, use the computer to place bogus calls, etc. While in principle Alice could grant and then revoke the privilege, in practice privileges accumulate; if we are concerned about this attack, something else is needed. There are a number of potential countermeasures to this sort of issue.

Individual Consent

Ask the user for permission for each call.

Callee-oriented Consent

Only allow calls to a given user.

Cryptographic Consent

Only allow calls to a given set of peer keying material or to a cryptographically established identity.

Unfortunately, none of these approaches is satisfactory for all cases. As discussed above, individual consent puts the user's approval in the UI flow for every call. Not only does this quickly become annoying but it can train the user to simply click "OK", at which point the consent becomes useless. Thus, while it may be necessary to have individual consent in some case, this is not a suitable solution for (for instance) the calling service case. Where necessary, the user interface must be carefully designed to avoid the risk of the user blindly clicking through.

The other two options are designed to restrict calls to a given target. Unfortunately, Callee-oriented consent does not work well because a malicious site can claim that the user is calling any user of his choice. One fix for this is to tie calls to a cryptographically established identity. While not suitable for all cases, this approach may be useful for some. If we consider the advertising case described in Section 4.1.1.3, it's not particularly convenient to require the advertiser to instantiate an iframe on the hosting site just to get permission; a more convenient approach is to cryptographically tie the advertiser's certificate to the communication directly. We're still tying permissions to origin here, but to the media origin rather than to the Web origin.

Another case where media-level cryptographic identity makes sense is when a user really does not trust the calling site. For instance, I might be worried that the calling service will attempt to bug my

computer, but I also want to be able to conveniently call my friends. If consent is tied to particular communications endpoints, then my risk is limited. However, this is also not that convenient an interface, since managing individual user permissions can be painful.

While this is primarily a question not for IETF, it should be clear that there is no really good answer. In general, if you cannot trust the site which you have authorized for calling not to bug you then your security situation is not really ideal. It is RECOMMENDED that browsers have explicit (and obvious) indicators that they are in a call in order to mitigate this risk.

4.1.3. Security Properties of the Calling Page

Origin-based security is intended to security against web attackers. However, we must also consider the case of network attackers. Consider the case where I have granted permission to a calling service by an origin that has the HTTP scheme, e.g., `http://calling-service.example.com`. If I ever use my computer on an unsecured network (e.g., a hotspot or if my own home wireless network is insecure), and browse any HTTP site, then an attacker can bug my computer. The attack proceeds like this:

1. I connect to `http://anything.example.org/`. Note that this site is unaffiliated with the calling service.
2. The attacker modifies my HTTP connection to inject an IFRAME (or a redirect) to `http://calling-service.example.com`
3. The attacker forges the page from `http://calling-service.example.com/` to inject JS to initiate a call to himself.

Note that this attack does not depend on the media being insecure. Because the call is to the attacker, it is also encrypted to him. Moreover, it need not be executed immediately; the attacker can "infect" the origin semi-permanently (e.g., with a web worker or a popunder) and thus be able to bug me long after I have left the infected network. This risk is created by allowing calls at all from a page fetched over HTTP.

Even if calls are only possible from HTTPS sites, if the site embeds active content (e.g., JavaScript) that is fetched over HTTP or from an untrusted site, because that JavaScript is executed in the security context of the page [finer-grained]. Thus, it is also dangerous to allow RTC-Web functionality from HTTPS origins that embed mixed content. Note: this issue is not restricted to PAGES which contain mixed content. If a page from a given origin ever loads mixed content then it is possible for a network attacker to infect the browser's notion of that origin semi-permanently.

4.2. Communications Consent Verification

As discussed in Section 3.3, allowing web applications unrestricted access to the via the browser network introduces the risk of using the browser as an attack platform against machines which would not otherwise be accessible to the malicious site, for instance because they are topologically restricted (e.g., behind a firewall or NAT). In order to prevent this form of attack as well as cross-protocol attacks it is important to require that the target of traffic explicitly consent to receiving the traffic in question. Until that consent has been verified for a given endpoint, traffic other than the consent handshake **MUST NOT** be sent to that endpoint.

4.2.1. ICE

Verifying receiver consent requires some sort of explicit handshake, but conveniently we already need one in order to do NAT hole-punching. ICE [RFC5245] includes a handshake designed to verify that the receiving element wishes to receive traffic from the sender. It is important to remember here that the site initiating ICE is presumed malicious; in order for the handshake to be secure the receiving element **MUST** demonstrate receipt/knowledge of some value not available to the site (thus preventing it from forging responses). In order to achieve this objective with ICE, the STUN transaction IDs must be generated by the browser and **MUST NOT** be made available to the initiating script, even via a diagnostic interface.

4.2.2. Masking

Once consent is verified, there still is some concern about misinterpretation attacks as described by Huang et al.[huang-w2sp]. As long as communication is limited to UDP, then this risk is probably limited, thus masking is not required for UDP. I.e., once communications consent has been verified, it is most likely safe to allow the implementation to send arbitrary UDP traffic to the chosen destination, provided that the STUN keepalives continue to succeed. However, with TCP the risk of transparent proxies becomes much more severe. If TCP is to be used, then WebSockets style masking **MUST** be employed.

4.2.3. Backward Compatibility

A requirement to use ICE limits compatibility with legacy non-ICE clients. It seems unsafe to completely remove the requirement for some check. All proposed checks have the common feature that the browser sends some message to the candidate traffic recipient and refuses to send other traffic until that message has been replied to. The message/reply pair must be generated in such a way that an

attacker who controls the Web application cannot forge them, generally by having the message contain some secret value that must be incorporated (e.g., echoed, hashed into, etc.). Non-ICE candidates for this role (in cases where the legacy endpoint has a public address) include:

- o STUN checks without using ICE (i.e., the non-RTC-web endpoint sets up a STUN responder.)
- o Use or RTCP as an implicit reachability check.

In the RTCP approach, the RTC-Web endpoint is allowed to send a limited number of RTP packets prior to receiving consent. This allows a short window of attack. In addition, some legacy endpoints do not support RTCP, so this is a much more expensive solution for such endpoints, for which it would likely be easier to implement ICE. For these two reasons, an RTCP-based approach does not seem to address the security issue satisfactorily.

In the STUN approach, the RTC-Web endpoint is able to verify that the recipient is running some kind of STUN endpoint but unless the STUN responder is integrated with the ICE username/password establishment system, the RTC-Web endpoint cannot verify that the recipient consents to this particular call. This may be an issue if existing STUN servers are operated at addresses that are not able to handle bandwidth-based attacks. Thus, this approach does not seem satisfactory either.

If the systems are tightly integrated (i.e., the STUN endpoint responds with responses authenticated with ICE credentials) then this issue does not exist. However, such a design is very close to an ICE-Lite implementation (indeed, arguably is one). An intermediate approach would be to have a STUN extension that indicated that one was responding to RTC-Web checks but not computing integrity checks based on the ICE credentials. This would allow the use of standalone STUN servers without the risk of confusing them with legacy STUN servers. If a non-ICE legacy solution is needed, then this is probably the best choice.

4.3. Communications Security

Finally, we consider a problem familiar from the SIP world: communications security. For obvious reasons, it MUST be possible for the communicating parties to establish a channel which is secure against both message recovery and message modification. (See [RFC5479] for more details.) This service must be provided for both data and voice/video. Ideally the same security mechanisms would be used for both types of content. Technology for providing this service (for instance, DTLS [RFC4347] and DTLS-SRTP [RFC5763]) is

well understood. However, we must examine this technology to the RTC-Web context, where the threat model is somewhat different.

In general, it is important to understand that unlike a conventional SIP proxy, the calling service (i.e., the Web server) controls not only the channel between the communicating endpoints but also the application running on the user's browser. While in principle it is possible for the browser to cut the calling service out of the loop and directly present trusted information (and perhaps get consent), practice in modern browsers is to avoid this whenever possible. "In-flow" modal dialogs which require the user to consent to specific actions are particularly disfavored as human factors research indicates that unless they are made extremely invasive, users simply agree to them without actually consciously giving consent. [abarth-rtcweb]. Thus, nearly all the UI will necessarily be rendered by the browser but under control of the calling service. This likely includes the peer's identity information, which, after all, is only meaningful in the context of some calling service.

This limitation does not mean that preventing attack by the calling service is completely hopeless. However, we need to distinguish between two classes of attack:

Retrospective compromise of calling service.

The calling service is non-malicious during a call but subsequently is compromised and wishes to attack an older call.

During-call attack by calling service.

The calling service is compromised during the call it wishes to attack.

Providing security against the former type of attack is practical using the techniques discussed in Section 4.3.1. However, it is extremely difficult to prevent a trusted but malicious calling service from actively attacking a user's calls, either by mounting a MITM attack or by diverting them entirely. (Note that this attack applies equally to a network attacker if communications to the calling service are not secured.) We discuss some potential approaches and why they are likely to be impractical in Section 4.3.2.

4.3.1. Protecting Against Retrospective Compromise

In a retrospective attack, the calling service was uncompromised during the call, but that an attacker subsequently wants to recover the content of the call. We assume that the attacker has access to the protected media stream as well as having full control of the calling service.

If the calling service has access to the traffic keying material (as in SDES [RFC4568]), then retrospective attack is trivial. This form of attack is particularly serious in the Web context because it is standard practice in Web services to run extensive logging and monitoring. Thus, it is highly likely that if the traffic key is part of any HTTP request it will be logged somewhere and thus subject to subsequent compromise. It is this consideration that makes an automatic, public key-based key exchange mechanism imperative for RTC-Web (this is a good idea for any communications security system) and this mechanism SHOULD provide perfect forward secrecy (PFS). The signaling channel/calling service can be used to authenticate this mechanism.

In addition, the system MUST NOT provide any APIs to extract either long-term keying material or to directly access any stored traffic keys. Otherwise, an attacker who subsequently compromised the calling service might be able to use those APIs to recover the traffic keys and thus compromise the traffic.

4.3.2. Protecting Against During-Call Attack

Protecting against attacks during a call is a more difficult proposition. Even if the calling service cannot directly access keying material (as recommended in the previous section), it can simply mount a man-in-the-middle attack on the connection, telling Alice that she is calling Bob and Bob that he is calling Alice, while in fact the calling service is acting as a calling bridge and capturing all the traffic. While in theory it is possible to construct techniques which protect against this form of attack, in practice these techniques all require far too much user intervention to be practical, given the user interface constraints described in [abarth-rtcweb].

[OPEN ISSUE: What are the best UI recommendations to make?
Proposal: take the text from Matthew Kaufman's draft.]

4.3.2.1. Key Continuity

One natural approach is to use "key continuity". While a malicious calling service can present any identity it chooses to the user, it cannot produce a private key that maps to a given public key. Thus, it is possible for the browser to note a given user's public key and generate an alarm whenever that user's key changes. SSH [RFC4251] uses a similar technique. (Note that the need to avoid explicit user consent on every call precludes the browser requiring an immediate manual check of the peer's key).

Unfortunately, this sort of key continuity mechanism is far less

useful in the RTC-Web context. First, much of the virtue of RTC-Web (and any Web application) is that it is not bound to particular piece of client software. Thus, it will be not only possible but routine for a user to use multiple browsers on different computers which will of course have different keying material (SACRED [RFC3760] notwithstanding.) Thus, users will frequently be alerted to key mismatches which are in fact completely legitimate, with the result that they are trained to simply click through them. As it is known that users routinely will click through far more dire warnings [cranor-wolf], it seems extremely unlikely that any key continuity mechanism will be effective rather than simply annoying.

Moreover, it is trivial to bypass even this kind of mechanism. Recall that unlike the case of SSH, the browser never directly gets the peer's identity from the user. Rather, it is provided by the calling service. Even enabling a mechanism of this type would require an API to allow the calling service to tell the browser "this is a call to user X". All the calling service needs to do to avoid triggering a key continuity warning is to tell the browser that "this is a call to user Y" where Y is close to X. Even if the user actually checks the other side's name (which all available evidence indicates is unlikely), this would require (a) the browser to trusted UI to provide the name and (b) the user to not be fooled by similar appearing names.

4.3.2.2. Short Authentication Strings

ZRTP [RFC6189] uses a "short authentication string" (SAS) which is derived from the key agreement protocol. This SAS is designed to be read over the voice channel and if confirmed by both sides precludes MITM attack. The intention is that the SAS is used once and then key continuity (though a different mechanism from that discussed above) is used thereafter.

Unfortunately, the SAS does not offer a practical solution to the problem of a compromised calling service. "Voice conversion" systems, which modify voice from one speaker to make it sound like another, are an active area of research. These systems are already good enough to fool both automatic recognition systems [farus-conversion] and humans [kain-conversion] in many cases, and are of course likely to improve in future, especially in an environment where the user just wants to get on with the phone call. Thus, even if SAS is effective today, it is likely not to be so for much longer. Moreover, it is possible for an attacker who controls the browser to allow the SAS to succeed and then simulate call failure and reconnect, trusting that the user will not notice that the "no SAS" indicator has been set (which seems likely).

Even were SAS secure if used, it seems exceedingly unlikely that users will actually use it. As discussed above, the browser UI constraints preclude requiring the SAS exchange prior to completing the call and so it must be voluntary; at most the browser will provide some UI indicator that the SAS has not yet been checked. However, it is well-known that when faced with optional mechanisms such as fingerprints, users simply do not check them [whitten-johnny]. Thus, it is highly unlikely that users will ever perform the SAS exchange.

Once users have checked the SAS once, key continuity is required to avoid them needing to check it on every call. However, this is problematic for reasons indicated in Section 4.3.2.1. In principle it is of course possible to render a different UI element to indicate that calls are using an unauthenticated set of keying material (recall that the attacker can just present a slightly different name so that the attack shows the same UI as a call to a new device or to someone you haven't called before) but as a practical matter, users simply ignore such indicators even in the rather more dire case of mixed content warnings.

5. Security Considerations

This entire document is about security.

6. Acknowledgements

7. References

7.1. Normative References

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

7.2. Informative References

[CORS] van Kesteren, A., "Cross-Origin Resource Sharing".

[I-D.abarth-origin]
Barth, A., "The Web Origin Concept",
draft-abarth-origin-09 (work in progress), November 2010.

[I-D.ietf-hybi-thewebsocketprotocol]
Fette, I. and A. Melnikov, "The WebSocket protocol",
draft-ietf-hybi-thewebsocketprotocol-13 (work in

progress), August 2011.

- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC3760] Gustafson, D., Just, M., and M. Nystrom, "Securely Available Credentials (SACRED) - Credential Server Framework", RFC 3760, April 2004.
- [RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5479] Wing, D., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, April 2009.
- [RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.
- [RFC6189] Zimmermann, P., Johnston, A., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, April 2011.
- [abarth-rtcweb] Barth, A., "Prompting the user is security failure", RTC-Web Workshop.

[cranor-wolf]

Sunshine, J., Egelman, S., Almuhiemedi, H., Atri, N., and L. cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness", Proceedings of the 18th USENIX Security Symposium, 2009.

[farus-conversion]

Farrus, M., Erro, D., and J. Hernando, "Speaker Recognition Robustness to Voice Conversion".

[finer-grained]

Barth, A. and C. Jackson, "Beware of Finer-Grained Origins", W2SP, 2008.

[huang-w2sp]

Huang, L-S., Chen, E., Barth, A., Rescorla, E., and C. Jackson, "Talking to Yourself for Fun and Profit", W2SP, 2011.

[kain-conversion]

Kain, A. and M. Macon, "Design and Evaluation of a Voice Conversion Algorithm based on Spectral Envelope Mapping and Residual Prediction", Proceedings of ICASSP, May 2001.

[whitten-johnny]

Whitten, A. and J. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0", Proceedings of the 8th USENIX Security Symposium, 1999.

Author's Address

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Phone: +1 650 678 2350
Email: ekr@rtfm.com

RTCWEB
Internet-Draft
Intended status: Informational
Expires: January 5, 2012

J. Rosenberg
Skype
C. Jennings
Cisco
J. Peterson
Neustar
M. Kaufman
Skype
E. Rescorla
RTFM
T. Terriberry
Mozilla
July 4, 2011

Multiplexing of Real-Time Transport Protocol (RTP) Traffic for Browser
based Real-Time Communications (RTC)
draft-rosenberg-rtcweb-rtpmux-00

Abstract

This document argues that multiplexing of voice and video traffic over a single RTP session should be specified as the baseline mode of operation for multimedia traffic in RTC web.

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 January 5, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. RTP Muxing with SSRC	3
3. Arguments in Favor of Multiplexing	4
3.1. NAT Resource Preservation	4
3.2. Improved Failure Modes	5
3.3. Setup Time	5
3.4. Complexity	5
4. Responding to draft-perkins-rtcweb-rtp-usage	5
4.1. Requires Additional Signaling	6
4.2. QoS and Traffic Engineering	6
4.3. Scalability	7
4.4. RTP Retransmission	7
4.5. Forward Error Correction	8
4.6. RTCP Issues	8
5. Arguing Against a Shim	9
6. Conclusion	10
7. Informative References	10
Authors' Addresses	11

1. Introduction

The RTCweb working group is chartered to specify a framework and protocols for enabling real-time communications services within a browser, without the need for plugins [I-D.rosenberg-rtcweb-framework]. It is envisioned that this will enable many use cases [I-D.ietf-rtcweb-use-cases-and-requirements], the most basic of which is a video call between two users on the web.

In order to enable this functionality, the specifications produced by the IETF will mandate a specific set of protocols that must be implemented within the browser. It is anticipated that these protocols will include the Real-Time Transport Protocol [RFC3550], and either in full or in part, Interactive Connectivity Establishment (ICE) [RFC5245].

The usage of RTP raises the question of multiplexing - whether or not RTCP and RTP should run on the same port, and furthermore, whether or not voice, video, and possibly data, should also run on the same port. To provide guidance on this, Perkins et. al. produced [I-D.perkins-rtcweb-rtp-usage], which recommends that voice and video utilize different RTP sessions, and thus different UDP ports.

This document argues against this conclusion, and advocates that a single transport session (i.e., a single UDP port) is used to carry voice and video traffic, using the SSRC for demux.

2. RTP Muxing with SSRC

This document recommends that all of the associated media content of the call - the voice, video, and RTCP traffic for both the voice and video sessions, utilize a single transport session (i.e., single UDP port). In cases where there are multiple video streams (for example, screen sharing), the single transport session would carry all of the video. Furthermore, that demultiplexing voice and video traffic is done by assigning a different SSRC to each. This recommendation applies to the case of a single unicast communications session between a pair of endpoints (e.g., this document does not consider the case of running a multi-user service like a gateway).

To enable multiplexing, we propose that the 32-bit SSRC value in the RTP header be broken up into the following sub-fields:

```

+-----+
|          Magic Cookie          | Type |      StreamID      | x |
+-----+
```

SSRC Field

The Magic Cookie is two bytes, with a value of 0xf7b3. It is meant to facilitate DPI applications which can use its value to - with high confidence - determine that this RTP packet uses the encoding format defined here. The type is a 3 bit value, corresponding to the top-level MIME type of the media (mapping table TBD). It too is meant to facilitate DPI applications which want to separate voice and video. The streamID is a 12 bit field which represents the unique ID for this stream. It is signaled between participants out of band. The final bit, 'x' is set to zero and is reserved for future usage.

3. Arguments in Favor of Multiplexing

This section outlines several arguments in favor of multiplexing.

3.1. NAT Resource Preservation

Today's Internet is full of Network Address Translators (NAT), a situation which is likely to get worse as IPv4 address exhaustion continues. When NAT is in use, the constraint on the number of endpoints behind the NAT is based on the number of parallel transport sessions that need to be supported. If, for example, a NAT has a single external IP address, it can support 64k UDP sessions while having an endpoint-independent mapping behavior [RFC4787]. Thus, in the presence of NAT, parallel transport sessions becomes the scarce resource.

If rtcweb specifies that audio and video run on a separate port, this will double the number of transport session resources consumed in intervening NATs. While the usage of port as an application layer demux point made sense when RTP was designed back in 1992 (the year the first RTP draft was published), the Internet has changed substantially since then. Continuing to perpetuate this design optimizes pre-severation of legacy against protection of resources in the modern Internet. We feel that this optimizes in the wrong direction.

Given that we anticipate widespread usage of rtcweb, this design choice may create a non-trivial load on the transport session capacity of the Internet at large. Real-time video communications on the Internet has seen huge growth in recent years. For Skype, approximately 40% of its Skype-to-Skype calls are video based. A recent report by Sandvine reports that Skype alone is the third largest source of upload traffic on the Internet as a whole, largely attributed to Skype video calling. <http://www.sandvine.com/downloads/documents/05-17-2011_phenomena/

Sandvine%20Global%20Internet%20Phenomena%20Spotlight%20-%20Netflix%20Rising.pdf>. The conclusion from this is that the costs of a separate voice and video port cannot be ignored.

Simply put, the usage of transport ports for application demultiplexing should be considered harmful for the Internet.

3.2. Improved Failure Modes

The usage of separate transport sessions for the audio, video or other content of the call introduces a variety of partial failure modes. The transport session for one type of media might get established; but a NAT capacity problem might cause the transport session for another type of media to fail. Usage of a single transport session means that the conversation succeeds or fails atomically. We consider this a feature.

3.3. Setup Time

The rtcweb group is considering the usage of ICE to create p2p sessions. ICE provides firewall and NAT traversal in addition to providing a handshake necessary to assure mutual consent for communications.

Unfortunately, ICE requires time to perform its setup operations. This time grows in proportion to the number of transport sessions which must be opened in order to support the call. By using a different port for video traffic, call setup times will increase. The precise amount of this increase depends on the type of NAT and varies depending on packet loss. However, in a simple, ideal case of no packet loss and direct connectivity between endpoints, this value is XXX [[fill in]].

3.4. Complexity

ICE is not a simple protocol. One of its significant complexities is its requirement to support calls for multiple media streams, each of which runs on a separate port, and multiple components for each stream (e.g., RTCP). If the concept of streams and components were eliminated, ICE would be a simpler protocol.

If, within rtcweb, a single transport connection was utilized, browsers could implement a simplified version of the ICE protocol.

4. Responding to draft-perkins-rtcweb-rtp-usage

[I-D.perkins-rtcweb-rtp-usage] outlines several arguments for

continuing to use a separate port for audio and video. In this section, we respond to those arguments.

4.1. Requires Additional Signaling

[I-D.perkins-rtcweb-rtp-usage] argues that multiplexing of voice and video on the same RTP session would require a demux point to be specified (for example, the SSRC), and require additional signaling to be specified to accomplish this.

Firstly, this conclusion is only partly true. For communications sessions between rtcweb users within the same domain, no signaling specifications are required. This is true in general with rtcweb; one of its benefits is that it does not require standardized signaling.

Secondly, it is not yet clear that rtcweb will be able to interoperate with existing VoIP endpoints without a media intermediary to terminate ICE traffic. It is our position that interoperability without media intermediary only be provided for basic voice services, and even then, only when RTCP is supported. In the case of basic voice endpoints, where there is no video, RTP multiplexing of voice and video is irrelevant, and thus no signaling complexity is introduced.

Thirdly, the primary place where there will be a need for signaling enhancements is for inter-domain calling between rtcweb endpoints in different domains. In such a case, an SDP extension is required, and one can be specified. It is trivial to do so.

Finally, this document does recommend that it be possible to utilize a separate transport session for voice and for video, and that, in the worst case, this mode can be used for calls between an rtcweb endpoint and a legacy endpoint.

4.2. QoS and Traffic Engineering

[I-D.perkins-rtcweb-rtp-usage] argues that multiplexing of voice and video on the same RTP session would mean that it would not be possible to apply QoS techniques separately for voice and video which rely on the 5-tuple.

Firstly, the public Internet lacks any QoS mechanism, so this argument is moot on the public Internet.

Secondly, private enterprise networks which do provide QoS most often use diffserv. Diffserv is compatible with utilization of a common port for voice and video traffic. Typically, different DSCPs are

used for voice and video (Cisco recommends EF for audio and AF41 for video in enterprise telephony deployments), and this practice is compatible with usage of the same port - each packet would be marked appropriately. It is also possible to use the same DSCP for voice and video.

Carrier networks, such as mobile operator networks, typically provide QoS through traffic engineering, using a combination of MPLS tunnels and diffserv markings. MPLS tunnels do use 5-tuples as classifiers to determine which traffic to put in what kind of tunnel. If there is a need for using separate MPLS tunnels for voice and video, the DSCP codepoint itself can be used as a differentiator.

It is true that it would not be possible to utilize RSVP to separately establish QoS treatment for the voice and the video traffic. However, there is very little real deployment of RSVP. None within the public Internet and relatively little within corporate networks. As such, this argument is mostly theoretical.

Finally, DPI is used within some operator networks to perform traffic classification. It would always be possible to use DPI to assign different treatment to voice and video traffic.

4.3. Scalability

[I-D.perkins-rtcweb-rtp-usage] argues that multiplexing of voice and video on the same RTP session would mean that layered coding using multicast for each layer would not be possible.

Firstly, most layered coding today uses unicast and a switch or mixer of some sort to discard layers. That architecture is completely compatible with the usage of a single transport session for voice and video. The limitation applies only to the use of IP multicast for real-time communications. The usage of multicast on the Internet has substantially diminished over time. There is some usage today in private networks but primarily for streaming media distribution. The usage for real-time communications is quite rare. As such, we find this to be a theoretical corner case.

4.4. RTP Retransmission

[I-D.perkins-rtcweb-rtp-usage] argues that multiplexing of voice and video on the same RTP session would not be interoperable with endpoints doing RTP retransmission per [RFC4588].

As pointed out above, interoperability with existing endpoints without the usage of a media intermediary is not a given at this point, and we argue it should only be supported for the common case -

a basic, voice-only RTP-capable endpoint. There is, to our knowledge, relatively little deployment of RFC4588, at least for real-time communications. It is certainly not a common feature in basic RTP endpoints and never a baseline requirement for interoperability. Consequently, if there is a need to interoperate with an endpoint supporting RFC4588, and it is desired to avoid a media intermediary, RFC4588 can just be turned off for the session.

As such, we find the interoperability argument here not compelling.

4.5. Forward Error Correction

[I-D.perkins-rtcweb-rtp-usage] argues that multiplexing of voice and video on the same RTP session will limit the applicability of FEC [RFC5109] to when the RTP packets are half of the path MTU.

There are two cases to consider - interoperability with existing endpoints and usage for calls between rtcweb endpoints.

For interoperability with existing endpoints, we argue the same thing here as for retransmits. FEC is not commonly used in legacy voice endpoints, and if it is supported, is never a required feature. Consequently, if present, its usage can be disabled when interoperating with an rtcweb endpoint. If FEC is included as part of the rtcweb specifications, the lower bandwidth of voice means that FEC packets could be sent on the same port, using [RFC2198], without approaching the path MTU.

For communications between rtcweb endpoints, this is only an issue if FEC is included as part of the rtcweb specification. If the group decides to do that (there is some value for real-time video), it should define a mechanism which allows for FEC packets to be sent using a separate SSRC.

4.6. RTCP Issues

[I-D.perkins-rtcweb-rtp-usage] argues that multiplexing of voice and video on the same RTP session will introduce complications in the usage of RTCP, primarily when considering RTCP extensions.

It is our belief that normal RTCP operation as defined in the RTCP specification will work fine with multiplexed voice and video traffic. SRs and RRs are already generated per SSRC to handle multiple senders, and RTCP in general supports feedback for multiple SSRC within a session. These mechanisms work as defined when each SSRC happens to represent a different media stream instead of a different user.

The only complication that arises is for RTCP extensions which are defined to be media dependent. [I-D.perkins-rtcweb-rtp-usage] points out, as an example, the usage of RTCP extended report blocks (XR) [RFC3611]. However, XR works fine in conjunction with multiplexing of voice and video within the same port. Each of the seven report blocks defined in [RFC3611] include the SSRC of the source as part of the block, and thus will work. [I-D.perkins-rtcweb-rtp-usage] indicates that "SSRC purpose tagging needs not only to be on the media side, but also on the RTCP reporting". However, we do not believe this to be accurate. Since the XR blocks report the SSRC source already, the specifications provide all that is needed. The XR report is merely included when it is relevant.

Furthermore, the discussion around XR assumes that we need to support them for interoperability with existing VoIP endpoints, or we are utilizing it for rtcweb itself. As with FEC and retransmissions, in the case of interoperability, if there is an issue, XR can simply be disabled in these cases. [RFC3611] does specify that XR can be sent without prior signaling. In the worst case XR are received by an rtcweb endpoint which are then discarded. In terms of usage of RTCP XR for communications between rtcweb endpoints, we would argue that a much more flexible solution would be to provide Javascript APIs which allow the application to have access to the same data used to generate the XR, and then the application itself can use this data as it sees fit, including sending it back to the sender through some kind of application data packet.

5. Arguing Against a Shim

It has been proposed on the mailing list that an alternative approach for multiplexing on the same port would be to specify a new multiplexing protocol that has a small shim, which could then be used to separate voice and video traffic as a layer between UDP and RTP. Such a shim could then also be used to enable non-RTP data traffic as well.

We believe that such a shim would be a mistake, for the same reason that shims have been avoided in the multiplexing of RTCP, STUN, and DTLS on the same port as RTP:

- o The shim would break interoperability with a great deal of existing network inspection gear - firewalls, packet sniffers, traffic analyzers, and so on - which know how to extract, parse, and process RTP packets.
- o The shim would add complexity through yet another layer of multiplexing.

- o The shim would increase packet overhead further.
- o A shim is a mistake which cannot be undone later. If multiplexing on a single port truly causes interoperability issues, clients can fall back to using multiple ports, possibly even in the preponderance of cases. However, once a shim is inserted, interoperability will always require an intermediary to strip it out, forever.

6. Conclusion

In conclusion, we feel that benefits of multiplexing of voice and video on a single RTP session (and thus single transport connection), outweigh the drawbacks. The primary benefit is the impact on NAT capacity, which is becoming an important issue in the modern Internet. Furthermore, the unique nature of backwards compatibility for rtcweb lessens many of the interoperability concerns, and the traditional arguments around multicast and RSVP are simply no longer relevant and those technologies have faded from use.

7. Informative References

- [I-D.perkins-rtcweb-rtp-usage]
Perkins, C., Westerlund, M., and J. Ott, "RTP Requirements for RTC-Web", draft-perkins-rtcweb-rtp-usage-01 (work in progress), June 2011.
- [I-D.rosenberg-rtcweb-framework]
Rosenberg, J., Kaufman, M., Hiie, M., and F. Audet, "An Architectural Framework for Browser based Real-Time Communications (RTC)", draft-rosenberg-rtcweb-framework-00 (work in progress), February 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-01 (work in progress), July 2011.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time

Applications", STD 64, RFC 3550, July 2003.

- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

Authors' Addresses

Jonathan Rosenberg
Skype

Email: jdrosen@skype.net
URI: <http://www.jdrosen.net>

Cullen Jennings
Cisco

Email: fluffy@cisco.com

Jon Peterson
Neustar

Email: jon.peterson@neustar.biz

Matthew Kaufman
Skype

Email: matthew.kaufman@skype.net

Eric Rescorla
RTFM

Email: ekr@rtfm.com

Tim Terriberry
Mozilla

Email: tterriberry@mozilla.com

RTCWEB
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2012

S. Wenger
A. Eleftheriadis
Vidyo
July 4, 2011

The Case for Layered Codecs
draft-wenger-rtcweb-layered-codec-00

Abstract

RTCWEB is in the process of developing a protocol infrastructure and a browser API to support browser-to-browser real-time communication over IP. Real-time communication necessarily requires the use of encoders and decoders (codecs) for media data. The document advocates mandating support for a class of codecs known as scalable or layered codecs, for their superior network adaptivity, error resilience, and application adaptivity. Examples are provided for use cases currently under discussion, focusing on video coding as the, perhaps, most challenging media type currently under consideration.

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 January 5, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Detour: Suggested additional Requirements	3
4. Scalable Codecs in the defined Use Cases	4
4.1. Use case: Browser to browser use-cases	5
4.2. Telephony use cases	7
4.3. Video conferencing use cases	8
4.3.1. Use-case: Multiparty Video Communication	8
4.3.2. Use-case: Video Conferencing w/ Central Server	10
4.4. Embedded voice communication use cases	10
4.5. Bandwidth/QoS/mobility use cases	11
5. Concluding Remarks	11
6. Security Considerations	12
7. References	12
7.1. Normative References	12
7.2. Informative References	12
Authors' Addresses	13

1. Introduction

In this document we advocate the mandatory support of scalable coding techniques for RTCWEB. If consensus towards such a position is not achievable for whatever technical, commercial, or legal reasons, we suggest that not having mandatory codecs may be better than mandating the use of an inferior, non-scalable codec.

The current status of RTCWEB's use case and requirements discussion is captured in [I-D.ietf-rtcweb-use-cases-and-requirements]. (Note: Version 00 of this I-D was used for the comparison in this document.) We use this document as a guide and attempt to compare the broad categories of scalable and non-scalable codecs in terms of how well they can satisfy the requirements specified therein. We use video codecs as an example, and conclude that scalable codecs are considerably better-suited for RTCWEB than non-scalable codecs.

We first address some additional - and, we believe, self-evident - requirements.

2. 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 RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Detour: Suggested additional Requirements

There are certain requirements that, we believe, appear so obvious to the authors of [I-D.ietf-rtcweb-use-cases-and-requirements] that they have not been explicitly captured. One of these requirements is:

Fxx: The browser MUST enable communication with a peer (other browser(s), MCUs, etc.) with a latency adequate for "real-time" communication.

This is a crucial requirement when selecting a (video) codec for reasons that will become apparent shortly. We believe that RTCWEB should strive for solutions allowing latency (not counting long-distance network delays) below 200 msec. Furthermore, any solution that cannot guarantee a latency below 500 msec (before dropping video altogether) should not be considered. At the time of writing there are mailing list discussions ongoing regarding a requirement that may be differently formulated but addresses the same operational aspect. We don't specifically care about the formulation, but highly

recommend to go through the pain of agreeing to fixed millisecond numbers.

An additional requirement that should be considered relates to heterogeneity:

Fxy: The browser must be able to send fully decodable bitstreams, ideally without wasting resources, for a broad range of receivers (handheld to desktop) and, in multi-party cases, for a heterogeneous receiver population.

Seems obvious? Yes, to us. Have the implications, however, been thoroughly considered? Not that we are aware of. We discuss this further when addressing the multiparty use case.

Finally, an additional requirement that relates to the heterogeneity of peers has to do with the heterogeneity of connection characteristics among peers:

Fxz: The browser **MUST** be able to perform error control on multiple streams, with potentially different error characteristics, simultaneously.

In a full-mesh, multipoint scenario, for example, handling of individual peer connection characteristics can be very challenging. Assume, by way of example, that in a five-way conference one of the streams gets damaged or that the corresponding connection is subject to congestion. If the browser chooses to produce a more robust stream, then the other three receiving peers will be penalized through the overhead spent for error protection and resulting inferior picture quality, even if they have perfect connections to the sender. If, on the other hand, the browser chooses to produce separate streams for each of its peers, then it must be able to handle an increasing computational load. Even if the computational load is not a concern, battery power consumption certainly is. In fact, there is only one thing that grows faster than CPU power (according to Moore's law): per-pixel cycle demands of video codecs. Also, we should bear in mind that even when staying with moderately complex (and efficient) video codecs, video resolution constantly goes up. Worse, resolution grows in both horizontal and vertical dimensions, whereas CPU power grows only in one dimension.

4. Scalable Codecs in the defined Use Cases

The concept of scalable coding (for video, audio, or any media) consists of providing a representation of the source at multiple levels of fidelity using a bitstream comprised of a corresponding

number of layers. Typically these layers are formed in a pyramidal fashion, such that a given level of fidelity requires the availability of some or all of the lower layers (i.e., those that correspond to lower levels of fidelity). The dependency exists because the encoding process uses prediction between layers, thus providing increased compression efficiency. If no such prediction is used, then the layers are independent of one another, and we have what is called simulcasting. Indeed, from a media coding point of view, simulcasting is scalable coding without inter-layer prediction. In simulcast, each layer provides a different level of fidelity and is independently decodable.

One example of a standardized codec that uses such pyramidal layering to implement scalable coding is H.264/SVC. MPEG-2 and H.263+ are two other similar examples. The scalable layers in H.264/SVC enhance fidelity in any of three dimensions: temporal rate, spatial resolution, or spatial quality (or signal-to-noise ratio, SNR). More than one such enhancement can exist in a given video at the same time, i.e., scalability enhancements can be combined.

Scalable coding has been a well-known design architecture in the media coding community since at least 1993, but it has only recently been used in products. As a relatively recent addition to the commercial audio-video communication arsenal, it is not yet widely known among people who are not video coding experts. An overview of the H.264/SVC standard can be read in [wiegand2007]. A detailed discussion of how it can be used in videoconferencing systems is provided in [eleftheriadis2006].

In the following we revisit each of the use cases and discuss how scalable coding, and particularly pyramidal scalable video coding, can be used to provide a significantly improved user experience.

4.1. Use case: Browser to browser use-cases

This category has four sub-cases. The sub-cases "simple video communication" and "simple video communication with inter-operator calling" are, from a codec viewpoint, the same, with the exception of the mentioned interoperability argument resulting in a need for a baseline codec. From a (video) codec viewpoint the "Hockey" sub-case appears to consist of two independent video streams being sent by the mobile phone. (We are actually not sure whether this, rather exotic, multi-camera scenario warrants its own use case, but we are not opposed to it, either.) The sub-case "video size change" appears, to us, to introduce a feature that would be relevant to all video-capable use cases. As a result, from a codec viewpoint, all four use cases appear to have similar requirements and can be discussed jointly.

On the surface, these use cases appear to be trivially supported by any video codec as long as basic rate control and error resilience features are utilized (probably with the help of protocol support such as RTCP receiver reports and feedback messages therein, retransmission, and so on).

Let's see, however, how scalable codecs can offer significant advantages over non-scalable codecs.

First, temporal scalability (where pictures coded in an enhancement layer can be decoded in combination with a lower frame rate base layer) can greatly enhance error resilience, through techniques such as the one described in [wenger1998]. Especially in conjunction with feedback messages, virtually latency-neutral repair mechanisms can be devised without relying on latency- and bandwidth-unfriendly intra (I) pictures. It is important to note that such "reactive" techniques do not penalize a system when no errors happen to occur. Also, by being reactive, they are amenable to heterogeneous error control support (thus addressing requirement Fxz).

(Note: sending intra pictures is typically not a good idea. First, when transmitting video over a bandwidth-limited link that is close to capacity (e.g., a 3G link), an I frame can bring the latency up to the seconds range. Second, I frames are much larger than P or B frames and, therefore, statistically much more likely to be hit by a packet loss. Sending I frames for error control is, in almost all cases, a bad idea. It has been done for many years, but only because few, if any, better error control mechanisms were available.)

A second scalability dimension is spatial scalability. Here the enhancement layer enables decoding of the video in a higher resolution than a base layer, but can predict information from the base layer. Spatial scalability can also offer advantages for error control, something that we would gladly elaborate on in the future. In this use case, however, its key advantage lies in the graceful and, if properly implemented, latency-neutral handling of both changes in available network bandwidth (addressing the bandwidth and error characteristics aspects of F23, among others) and receiver-side rendering requirements (addressing F22).

Most, if not all, video coding experts will agree that there is nothing better than shedding pixels when running into a bandwidth issue, and this can be trivially done (without sending an I frame!) when using spatially scalable coding techniques. Similarly, recovering from bandwidth shortages can be achieved without sending an I frame, again in those cases where the sender properly implements spatial scalability. These techniques are generally more flexible and often lead to better-quality pictures compared to the rate-

control techniques used in single-layer codecs (QP adjustment).

Another point that speaks for spatial scalability is the option to gracefully and quickly react to rendering size requirements, i.e., users enlarging or "full-screening" the rendering window (F22). Again adding a high resolution scalable layer can be done without sending an I frame (which, as already pointed out, are evil), and, if implemented properly, can be done without interrupting the smoothness of the video playback experience. (Note that even today's most advanced spatial scalability techniques still encode only a small set of discrete picture sizes in their finite, and small, number of layers. As a result, some form of resampling in the rendering interface will probably still be required.)

It has been remarked that simulcasting low and high resolution representations of the same picture can have very similar positive effects. This is correct. However, simulcast comes at a price, which is bandwidth and compute cycle requirements. The quite extensive subjective evaluation tests performed by MPEG in conjunction with the H.264/SVC verification tests have shown that a 2006-generation scalable video encoder offers between 17% and 40% bitrate savings over a same generation encoder using simulcast [mpeg2007].

(Note that "subjective" evaluation is the Mercedes of media quality testing, whereas "objective" evaluation compares to a pre-"Volt" Chevrolet :-). Subjective tests, when performed properly, are by no means unscientific and they are, in fact, considered a much better way to assess the quality of a codec compared with objective tests (e.g., the PSNR, or less crude metrics devised by groups such as the Video Quality Experts Group). Unfortunately, subjective evaluations are also orders of magnitude more expensive and time-consuming than objective tests, as it takes many person-hours to evaluate a single test case, whereas objective tests take only CPU cycles.)

Further, while a modern scalable encoder creating two spatial layers may require roughly the same number of cycles as two encoders coding the same pixel count, there are savings in the decoder due to what SVC calls "single loop decoding" (i.e., the fact that the receiver only needs to maintain a prediction loop just for the layer that is being decoded, and not for the layers that the to-be-decoded layer may depend on).

4.2. Telephony use cases

As the telephony use cases appear to address interaction with legacy telephone equipment only, there is probably little use for the quality scalability that modern scalable speech/audio codecs offer.

As formulated, from a codec viewpoint, any speech/audio codec used in telco environments (and many that have been specified outside this environment) ought to be acceptable choices.

4.3. Video conferencing use cases

There are two sub-cases in this broad category: full-mesh video conferencing, and centralized conferencing with resolution switching. The use cases also make particular assumptions regarding audio; the first case involves mono audio with panning at the web application, whereas the second involves mono or stereo audio with mixing at the server.

We first point out that these two use cases offer only a small subset of the functionality offered in today's multiparty videoconferencing solutions. In particular, the centralized server sub-case appears to deal with one, not frequently used, aspect of MCU-based communication that is implemented in a crude and unoptimized fashion (see below). We believe that video conferencing use cases in a standard drafted in 2011 should encompass at least as much functionality as is routinely offered in 2005 generation MCUs.

We wonder, specifically, why the group is not (yet?) considering other use cases in the same broad category of centralized conferencing that require more flexible server architectures. For example, sometimes active speaker size-up is not wanted, i.e., in scenarios involving presentations with questions. Or, one wants to be able to up-size more than one (but less than the whole population) of speakers. Or a full continuous presence conference where it is up to the user of each receiving browser to decide how he/she wishes to render the received signals. All of these capabilities are available with today's MCUs. Many similar scenarios could be described. We plan to contribute more detailed descriptions in the future unless we receive pushback.

We now examine each of the two use-case categories.

4.3.1. Use-case: Multiparty Video Communication

On the surface, this use case is not very different from the Simple Video use case, at least from a codec viewpoint. Indeed, it appears that a browser just needs to be able to simultaneously process N incoming independent video bitstreams (as spelled out in F12 and F14). A slightly deeper examination will reveal that this use case appears to be a picture-perfect example of why scalable codecs offer superior performance.

First off, all the arguments that were made for scalable codecs in

the Simple Video use case still apply. Let's consider, however, what happens when more than one stream is received.

A first consideration is the CPU requirements for decoding all the pictures in full resolution. Suppose a browser is receiving, for example, four video pictures in standard TV resolution and in a non-scalable format. The computational load for decoding these pictures in addition to the load of encoding one's outgoing picture would pretty much max out today's desktop CPUs. In typical multipoint layouts, the reconstructed pictures of most of the peers would be shown in thumbnail format, thus throwing away anywhere between 75-95% of the reconstructed pixel count! What a waste.

But waste what? Battery power? Yes, for handheld devices, which do not have the screen real-estate to show all the pictures in full resolution. We come to that in a moment. Electricity? Yes. This is not a joke. A desktop motherboard can easily double its power intake based on CPU load and memory access. Say, a motherboard's power consumption goes up from 100W to 200W, just because video is decoded in unnecessarily high resolution. One kWh of electricity costs at one of the author's home (Bay Area, California, USA, PG&E as the electricity provider) about 26 cents at his consumption level (single family house). This means that 8 hours of unnecessarily decoding of full-resolution video that is not rendered at full resolution will cost \$0.20. This, by coincidence, is the exact same maximum amount that MPEG-LA charges as licensing fees for using H.264/SVC :-). Never mind the numbers of trees that can be saved...

A second consideration is when multiple streams are received, or transmitted, by the browser in a heterogeneous receiver population. Browsers are today ubiquitous and can be found anywhere from handheld devices with QVGA screens to gaming racks with multiple 2K pixel resolution screens. That, incidentally, is a big part of the appeal of the RTCWEB activity. If you are sitting behind a 23-inch 1080p monitor, your display is approximately 120 dpi. A QVGA image, perfectly appropriate for being rendered on the screens of many smartphones, would be just 2-in by 1.5-in on the PC screen. On one of the authors' laptop (a 1080p, 13.3-in model, better than 200 dpi) it would literally be the size of a thumbnail. Not exactly what one expects to see, judging from daily uses of Skype and our own (Vidyo's) desktop video conferencing services. Further, when upsampling to, say, a quarter HD resolution, a QVGA signal simply doesn't look good no matter how well your upsample filter is designed.

This means that a transmitting browser that creates bitstreams for a receiver population including a smartphone and a PC, would have to, ideally, generate at least three resolutions: thumbnail, something

useful for smartphones (e.g., QVGA), and something useful for PC users (VGA or better). More option would be better. There are commercial products available today that ship, using a software codec on a high-end PC, 1080p60 video conferencing. Two years down the road, and a mainstream PC will be able to handle that type of load.

It has already been pointed out that the compute cycle requirements for simulcast and scalable encoding are roughly the same, assuming a common set of coding tools (i.e., H.264 Baseline profile vs. H.264 Scalable Baseline profile). The sending bandwidth requirements, however, are not the same: scalable beats simulcast by several tens of percentage points.

There are additional considerations for simulcast vs. scalable coding that have to do with RTP packetization and access unit alignment, resolution switching, and error resilience, that require much more extensive analysis than what's intended in this document.

4.3.2. Use-case: Video Conferencing w/ Central Server

This particular use case is interesting because, as formulated, it appears to assume that multiple video resolutions are produced by each sending browser and simulcast to the server. As formulated, the receiving browser receives the active speaker in full resolution and the other participants in low resolution. The server selects what to forward based on speech activity. Apparently, what is emulated here is one operation point implemented in many continuous presence MCUs (namely voice activation), without requiring the video transcoding features of a continuous presence MCU.

This use case almost begs for use of scalable video coding. The simulcasting alternative would have all the drawbacks discussed in the Simple Video Communication Service. For example, consider what happens when the active speaker changes. How is the receiving participant to switch resolutions between the two simulcast streams? It appears to require the request (from server to sending browser) of an I frame, with all the drawbacks that entails. Actually, as described, there appears to be no value in simulcasting all resolutions to the server, because of the need of such interaction...

4.4. Embedded voice communication use cases

The benefits discussed in multiparty video communication can apply here for scalable audio codecs. Scalable audio codecs haven't been mentioned prominently before, as in audio-visual systems the cost of video processing (in terms of compute cycles and bit rate, among others) typically outweighs that of audio. The requirements for today's games are so high that the CPU and bandwidth requirements for

a couple of audio channels may fall into the category of background noise. However, note that this ceases to be the case as the number of audio channels increases and the quality and (therefore) complexity of the audio codecs grows. We understand that the audio quality here would probably not be restricted to toll quality, and require something like Opus or MP3 for coding. While one or two of these codecs probably would still qualify as "background noise" on a gaming rack, 20 certainly do not. What appears to be useful here would be scalability both in terms of bitrate scalability and complexity scalability.

4.5. Bandwidth/QoS/mobility use cases

The use cases as formulated appear to have limited impact on the codec choices beyond aspects already raised above. NIC changes (or other changes that materially affect connectivity) are best addressed by codecs that can flexibly change their operation points without essentially restarting the codec (such as sending I frames in video, or user-generated codebooks in audio).

If QoS mechanisms that take advantage of QoS marking are supported in the underlying network - something that is at least questionable in today's Internet, but may be very relevant in the future and in special application fields such as the military - the pyramid structure of scalable codecs makes the selection of bits that are best to be transported at higher QoS trivial: the lower the layer, the higher the desired QoS. We note that scalable systems today emulate such a desirable network behavior by protecting base layers better than enhancement layers, through techniques such as retransmission or FEC.

5. Concluding Remarks

Scalability, while known as a concept for decades, is a relatively new technique in the commercial sphere of video and audio communication products. As a result, a lot of people are not familiar with how it works, and how it can be beneficial to real-time communication systems.

Most significantly, scalable coding has been successfully used to solve several fundamental, decades-old challenges in packet video communication, and it therefore behooves any standardization activity that considers codec design or adoption to take it into serious consideration.

6. Security Considerations

None

7. References

7.1. Normative References

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

7.2. Informative References

- [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-00 (work in progress), June 2011.
- [eleftheriadis2006]
Eleftheriadis, A., Cinvanlar, M., and O. Shapiro, "Multipoint Videoconferencing with Scalable Video Coding, Journal of Zhejiang University SCIENCE A, Vol. 7, Nr. 5, April 2006, pp. 696-705. (Proceedings of the Packet Video 2006 Workshop.)", 2006.
- [mpeg2007]
HHI and MPEG, "MPEG Verification Test for SVCd", 2007, <http://ip.hhi.de/imagecom_G1/savce/MPEG-Verification-Test/MPEG-Verification-Test.htm>.
- [wenger1998]
Wenger, S., "Temporal scalability using P-pictures for low-latency applications, Multimedia Signal Processing Workshop 1998, available from <http://www.stewe.org/papers/mm98/243/index.htm>", 1998.
- [wiegand2007]
Schwarz, H., Marpe, D., and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Scalable Video Coding, vol. 17, no. 9, pp. 1103-1120, September 2007, Invited Paper.", 2006.

Authors' Addresses

Stephan Wenger
Vidyo, Inc.
2400 SKyfarm Dr.
Hillsborough, CA 94010
US

Email: stewe@stewe.org

Alex Eleftheriadis
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: alex@vidyo.com

