RFC 8825 : Overview: Real Time Protocols for Browser-Based Applications

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 19 janvier 2021

Date de publication du RFC : Janvier 2021

https://www.bortzmeyer.org/8825.html

WebRTC sert à faire communiquer en temps réel (avec audio et vidéo) deux navigateurs Web. Il est déjà largement déployé, ce RFC arrive bien après la bataille, et décrit les généralités sur WebRTC. Ce protocole est complexe et offre beaucoup de choix au concepteur d'applications.

En fait, WebRTC n'est pas vraiment un protocole. Deux applications utilisant WebRTC peuvent parfaitement être dans l'incapacité de communiquer. WebRTC est plutôt un cadre ("*framework*") décrivant des composants, parmi lesquels le concepteur d'applications choisira. Vu cette complexité, ce RFC 8825 [1] est un point de départ nécessaire, pour comprendre l'articulation des différents composants. D'ailleurs, il y a un abus de langage courant : WebRTC désigne normalement uniquement l'API JavaScript normalisée par le W3C <https://www.w3.org/2011/04/webrtc/>. La communication sur le réseau se nomme rtcweb, et c'est le nom de l'ex-groupe de travail à l'IETF <https://datatracker.ietf.org/wg/rtcweb/>. Mais tout le monde fait cet abus de langage et moi aussi, j'utiliserai WebRTC pour désigner l'ensemble du système.

L'auteur du RFC est un « vieux de la vieille » et a vu passer beaucoup de protocoles. Il est donc logique qu'il commence par quelques rappels historiques en section 1. L'idée de faire passer de l'audio et de la vidéo sur l'Internet est aussi ancienne que l'Internet lui-même, malgré les prédictions pessimistes des telcos, qui avaient toujours prétendu que cela ne marcherait jamais. (La variante d'aujourd'hui est « si on ne nous laisse pas violer la neutralité du réseau <https://www.bortzmeyer.org/neutralite.html>, la vidéo ne marchera jamais ».)

Mais c'est vrai que les débuts ont été laborieux, la capacité <https://www.bortzmeyer.org/capacite.html> du réseau étant très insuffisante, et les processeurs trop lents. Ça s'est amélioré par

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8825.txt

la suite. Mais si le matériel a progressé, les protocoles continuaient à manquer : il existe bien sûr des protocoles standards pour traiter certains aspects de la communication (comme SIP - RFC 3261 - pour la signalisation) mais, pour une communication complète, il faut un **jeu de protocoles** couvrant tous les aspects, de la signalisation à l'encodage du flux vidéo, en passant par les identificateurs à utiliser (comme les adresses de courrier électronique), et qu'on puisse compter sur ce jeu, de même qu'on peut compter sur la disponibilité de clients et serveurs HTTP, par exemple. Comme le note l'auteur, « la Solution Universelle s'est avérée difficile à développer ». (Cela a d'ailleurs laissé un boulevard à des entreprises privées, proposant un produit très fermé et complètement intégré comme le Skype de Microsoft.)

Depuis quelques années, la disponibilité très répandue de navigateurs Web ayant des possibilités très avancées a changé la donne. Plus besoin d'installer tel ou tel "*plugin*", telle ou telle application, désormais, les possibilités qu'offre HTML5 sont largement répandues, et accessibles en JavaScript via une API standard. Un groupe de travail `<https://www.w3.org/2011/04/webrtc/>` du W3C développe de nouvelles possibilités et WebRTC peut donc être vu comme un effort commun du W3C et de l'IETF. C'est sur ces nouvelles possibilités du navigateur que se fonde WebRTC (son cahier des charges figure dans le RFC 7478).

Notre RFC note que WebRTC change pas mal l'architecture des techniques de communication temps-réel et multimédia. Autant les solutions fondées sur SIP comptaient sur des éléments intermédiaires dans le réseau pour travailler (par exemple des ALG), autant WebRTC est nettement plus « bout en bout », et utilise la cryptographie pour faire respecter ce principe, bien menacé par les FAI.

Question histoire, WebRTC a lui-même eu une histoire compliquée. La première réunion du groupe à l'IETF a eu lieu en 2011 `<https://www.ietf.org/proceedings/80/rtcweb.html>`, pour une fin prévue en 2012. En fait, il y a eu des retards, et même une longue période d'arrêt du travail. Ce RFC de survol de WebRTC, par exemple, a vu sa première version publiée il y a huit ans. Le premier RFC du projet n'a été publié qu'en 2015. Globalement, ce fut l'un des plus gros efforts de l'IETF `<https://datatracker.ietf.org/wg/rtcweb/documents/>`, mais pas le plus efficace en terme de rapidité. Le groupe de travail rtcweb a finalement été clos en août 2019, le travail étant terminé.

La section 2 de notre RFC résume les principes de WebRTC : permettre une communication audio, vidéo et données la plus directe possible entre les deux participants. Ce RFC 8825 est la description générale, les protocoles effectifs figurent dans d'autres RFC comme le RFC 8832 pour le protocole de création du canal de données ou le RFC 8831 pour la communication via le canal de données. Techniquement, WebRTC a deux grandes parties :

— La spécification du protocole, faite à l'IETF, dans les RFC 8832 et RFC 8831.
— L'API JavaScript permettant le développement des clients tournant dans les navigateurs Web, API normalisée par le W3C dans « "*WebRTC 1.0 : Real-time Communication Between Browsers*" `<https://www.w3.org/TR/webrtc/>` » et « "*Media Capture and Streams*" `<https://www.w3.org/TR/mediacapture-streams/>` ».

La section 2 décrit également le vocabulaire, si vous voulez réviser des concepts comme la notion de protocole ou de signalisation.

La section 3 de notre RFC présente l'architecture générale. Chaque navigateur WebRTC se connecte à un serveur Web où il récupère automatiquement un code JavaScript qui appelle les fonctions WebRTC, qui permettent d'accéder au canal de données, ouvert avec l'autre navigateur, et aux services multimédia de son ordinateur local (le micro, la caméra, etc). On y voit notamment (figure 2) que les données (la voix, la vidéo, etc) passent directement (enfin, presque, voir plus loin) entre les deux navigateurs, alors que la signalisation se fait entre les deux serveurs Web. On note, et c'est très important, que WebRTC ne gère que le canal de données, pas celui de signalisation, qui peut utiliser SIP (RFC 3261), XMPP (RFC 6120) ou même un protocole privé. C'est entre autres pour cette raison que deux services WebRTC différents ne

peuvent pas forcément interopérer. Voici un schéma de WebRTC lorsque les deux parties se connectent au même site Web (la signalisation est alors faite à l'intérieur de ce site) :

Et ici une image d'une fédération, où les deux parties qui communiquent utilisent des services différents, mais qui se sont mis d'accord sur un protocole de signalisation :

Le canal de données nécessite déjà beaucoup de spécifications, notamment le transport et l'encodage du contenu. Il est spécifié dans le RFC 8831.

Le transport, justement (section 4 du RFC). Il faut envoyer les données à l'autre navigateur, et assurer la retransmission des données perdues, le contrôle de congestion, etc. Tout cela est décrit dans le RFC 8835. Ce RFC spécifie, pour transporter les données multimédia, l'utilisation de SRTP sur DTLS, lui-même sur UDP (cf. RFC 8834). Des systèmes de relais comme TURN (RFC 5766) peuvent être nécessaires si les deux malheureux navigateurs Web soint coincés derrière des "*middleboxes*" méchantes, NAT ou pare-feu par exemple.

Ce n'est pas tout d'avoir un protocole de transport. Il faut aussi découper les données d'une manière compréhensible par l'autre bout ("*framing*", section 5 de notre RFC). WebRTC utilise RTP (RFC 3550, et voir aussi le RFC 8083) et SRTP (RFC 3711 et RFC 5764). Le RFC 8834 détaille leur utilisation. La sécurité est, elle, couverte dans les RFC 8826 et RFC 8827. L'établissement du canal de données est normalisé dans le RFC 8832 et le canal de données lui-même dans le RFC 8831.

On le sait, les formats d'audio et de vidéo sont un problème compliqué, le domaine est pourri de brevets souvent futiles, et la variété des formats rend difficile l'interopérabilité. Notre RFC impose au minimum l'acceptation des codecs décrits dans le RFC 7874 pour l'audio (Opus est obligatoire) et RFC 7742 pour la vidéo (H.264 et VP8, après une longue lutte). D'autres codecs pourront s'y ajouter par la suite.

La section 7 mentionne la gestion des connexions, pour laquelle la solution officielle est le JSEP ("*JavaScript Session Establishment Protocol*", RFC 8829).

La section 9 du RFC décrit les fonctions locales (l'accès par le navigateur aux services de la machine qui l'héberge). Le RFC rappelle que ces fonctions doivent inclure des choses comme la suppression d'écho, la protection de la vie privée (demander l'autorisation avant d'accéder à la caméra…) Le RFC 7874 fournit des détails. Ici, un exemple d'une demande d'autorisation par Firefox :

Parmi les logiciels qui permettent de faire du WebRTC, on peut citer Jitsi, qui est derrière le service Framatalk `<https://framatalk.org>`, mais qui est également accessible via d'autres services comme .

Regarder du trafic WebRTC avec un logiciel comme Wireshark est frustrant, car tout est chiffré. Pour Jitsi, on voit beaucoup de STUN, pour contourner les problèmes posés par le NAT, puis du DTLS évidemment impossible à décrypter.

Il y a de nombreux autres logiciels avec WebRTC. C'est le cas par exemple de PeerTube mais attention, seul l'échange de vidéo entre pairs qui regardent au même moment utilise WebRTC (plus exactement WebTorrent, qui utilise WebRTC). La récupération de la vidéo depuis le serveur se fait en classique HTTPS.

Si vous voulez voir une session WebRTC, le mieux est d'utiliser un service comme , qui journalise tout, et d'activer la console de Firefox. Vous voyez alors :

```
Console (Webdeveloper tools) Firefox
9.569: Initializing; server= undefined.
```

Puis l'établissement des canaux nécessaires :

```
37.252: Opening signaling channel.
39.348: Joined the room.
39.586: Retrieved ICE server information.
39.927: Signaling channel opened.
39.930: Registering signaling channel.
39.932: Signaling channel registered.
```

Là, Firefox vous demande l'autorisation d'utiliser micro et caméra, que vous acceptez :

```
44.452: Got access to local media with mediaConstraints:
  '{"audio":true,"video":true}'
44.453: User has granted access to local media.
```

Le navigateur peut alors utiliser ICE (RFC 8445) pour trouver le bon moyen de communiquer avec le pair (dans mon test, les deux pairs étaient sur le même réseau local, et s'en aperçoivent, ce qui permet une communication directe par la suite) :

```
44.579: Creating RTCPeerConnnection with:
  config: '{"rtcpMuxPolicy":"require","bundlePolicy":"max-bundle","iceServers":[{"urls":["stun:74.125.140.12
  constraints: '{"optional":[]}'.
44.775: Created PeerConnectionClient
```

On peut alors négocier les codecs à utiliser (ici VP9) :

```
44.979: Set remote session description success.
44.980: Waiting for remote video.
44.993: No preference on audio receive codec.
44.993: Prefer video receive codec: VP9
```

Et, ici, on voit passer le descripteur de session (section 7 de notre RFC), au format SDP (RFC 4566 et RFC 3264, et notez la blague entre SDP et le film 300) :

```
44.997: C->WSS: {"sdp":"v=0\r\no=mozilla...THIS_IS_SDPARTA-60.5.1 6947646294855805442 0 IN IP4 0.0.0.0\r\ns=
```

Et c'est bon, tout marche :

```
45.924: Call setup time: 1399ms.
```

Du fait de son caractère pair-à-pair (les données sont échangées, autant que le NAT le permet, directement entre les navigateurs), WebRTC soulève des questions de vie privée. Votre correspondant va voir votre adresse IP. Le point est discuté plus longuement dans le RFC 8826. (PeerTube met en bas un mot d'avertissement à ce sujet.)

WebRTC est présent depuis pas mal d'années dans tous les navigateurs graphiques comme Firefox, Chrome ou Edge. Côté bibliothèques, il existe de nombreuses mises en œuvre de WebRTC comme OpenWebRTC (abandonné depuis) ou comme l'implémentation de référence <https://webrtc.googlesource.com/>. Il y a également du WebRTC dans des serveurs comme Asterisk, WebEx ou MeetEcho <https://www.meetecho.com/> (ce dernier étant utilisé par l'IETF pour ses réunions à distance). Mais rappelez-vous que WebRTC n'est pas un ensemble cohérent permettant l'interopérabilité. Vous pouvez avoir deux services WebRTC qui n'arrivent pas à interagir, et ce n'est pas une bogue, c'est un choix de conception.

Et, pour finir, quelques lectures et activités supplémentaires :

— Pour tester votre environnement (micro, caméra, etc) et votre réseau (beaucoup de réseaux sont truffés de "*middleboxes*" hostiles), un bon site de test, , c'est vraiment le premier endroit où aller si vous avez des soucis techniques avec un service WebRTC,
— Vous pouvez voir les statistiques d'activité WebRTC dans votre navigateur, en pour Chrome, pour Opera, et pour Firefox,
— Un bon tutoriel sur WebRTC <https://www.html5rocks.com/en/tutorials/webrtc/basics/>, très détaillé, écrit plutôt du point de vue du programmeur JavaScript écrivant un client,
— Un bon résumé de WebRTC <http://www.frafos.com/wp-content/uploads/2014/11/FRAFOS_WebRTC_Deployment.pdf>, plutôt du point de vue de « comment ça marche »,
— Le site « officiel » du projet <https://webrtc.org/>,
— Le groupe de travail au W3C <https://www.w3.org/2011/04/webrtc/>.
— L'annonce officielle <https://www.w3.org/2021/01/pressrelease-webrtc-rec.html.en> de la publication des normes.