RFC 7478 : Web Real-Time Communication Use-cases and Requirements

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 30 mars 2015

Date de publication du RFC : Mars 2015

https://www.bortzmeyer.org/7478.html

Le système WebRTC permet une communication interactive et multimédia entre deux applications tournant dans un navigateur Web (d'où le nom de **Web**RTC). La vision est qu'Alice et Bob lanceront leur navigateur, iront sur une page qui chargera le code du logiciel (typiquement du JavaScript) et qu'ils communiqueront ensuite. Contrairement à Skype, WebRTC ne nécessitera pas l'installation d'un logiciel, et, surtout, il s'agira d'un protocole ouvert. Ce RFC, le premier du groupe de travail rtcweb <https://tools.ietf.org/wg/rtcweb>, décrit certains scénarios d'usage de WebRTC, pour aider à comprendre quel rôle il joue exactement. Il ne prétend pas être un cahier des charges pour WebRTC et, bien que publié tardivement, il représente plutôt l'état d'esprit au début du projet. (WebRTC a finalement été décrit dans le RFC 8825 [1].)

WebRTC peut permettre la communication avec des systèmes extérieurs et au moins un des scénarios décrits envisage une communication avec un téléphone SIP. Les demandes mises par ce document sur le (à cette époque) futur protocole WebRTC sont marquées Fn (où n est un entier) mais, on l'a vu, ce document ne prétend pas être un cahier des charges et WebRTC ne suivra pas forcément rigoureusement ces demandes. Les demandes marquées An concernent uniquement l'API de WebRTC (le protocole est développé par l'IETF, l'API par le W3C, oui, c'est un peu compliqué). L'annexe A contient ces exigences concernant l'API mais il faut voir le W3C <http://www.w3.org/TR/webrtc/> pour avoir l'information fiable.

Les clients WebRTC peuvent être connectés en IPv4, en IPv6, les deux, avoir des capacités réseau très différentes, être sur des réseaux dont la capacité effective varie (liens radio, liens congestionnés), être

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

derrière des pare-feux fascistes, et, bien sûr, être coincés derrière des NAT, des NAT de toutes les sortes (RFC 4787).

Attaquons maintenant les scénarios en commençant par le plus simple : ces chers Alice et Bob veulent discuter par vidéo. Ils ont le même opérateur WebRTC, sont informés en temps réel de la disponibilité de l'autre par une notification au navigateur, et, en cliquant sur le nom du correspondant, peuvent lui parler, l'écouter, le voir et se montrer. On a bien sûr les fonctions classiques, une petite vignette avec un retour de sa propre caméra, la capacité de couper le son pour crier une remarque à son conjoint sans que le correspondant l'entende, etc. Chaque partie peut raccrocher quand elle le veut. Évidemment, Bob et Alice ont des navigateurs de modèles différents, tournant sur des systèmes d'exploitation différents. Bien sûr, on veut de la sécurité, tout le trafic audio et vidéo doit évidemment être chiffré, authentifié et protégé contre toute modification.

De ce premier scénario d'usage, le cas le plus simple et le plus courant, on en déduit les exigences de base. Je ne vais pas toutes les citer mais en voici un échantillon, avec leur identificateur Fn :

— F1 : le navigateur doit avoir accès au micro et à la caméra,
— F2 : le navigateur doit pouvoir envoyer des données, même en présence de NAT (la bonne solution serait bien sûr de déployer IPv6 mais l'opposition des opérateurs rend les choses difficiles),
— Diverses exigences sur la qualité des flux multimédia (F3, réagir à la congestion, F6, détecter que le pair ne reçoit plus rien, F8, pouvoir synchroniser audio et vidéo),
— F11 : Protection contre les écoutes et autres « interceptions de sécurité » (RFC 2804 et RFC 7258),
— F13 : authentification et intégrité,
— F14 : vie privée, il doit exister un mode où Alice communique avec Bob sans lui révéler son adresse IP (possibilité de passer par un serveur pour se protéger).

Les autres exigences apparaissent au fur et à mesure des autres scénarios, qui présentent des cas de plus en plus complexes. Par exemple, Alice ou Bob est derrière un réseau fasciste ou mal configuré qui bloque complètement UDP. On en déduit l'exigence F18, pouvoir passer même quand UDP est bloqué. Variante : un réseau encore plus fasciste ou mal conçu qui n'autorise que HTTP, et via un relais. L'exigence F21 traite cas cas.

Réussir à faire passer une communication malgré les "*middleboxes*", les pare-feux et le NAT est le grand défi des applications réseau modernes. Les solutions nécessitent en général d'utiliser STUN (RFC 8489) et TURN (RFC 8656, le tout avec ICE (RFC 8445). L'exigence F19 impose de pouvoir utiliser ces services, y compris dans le cas où il y a plusieurs serveurs STUN et TURN disponibles. F20 impose de permettre l'utilisation de serveurs STUN et TURN qui ne sont pas ceux du fournisseur WebRTC (par exemple, au sein d'une organisation qui a son propre serveur STUN).

Lorsqu'on utilise des équipements mobiles, le problème peut se compliquer par un changement de réseau en cours de communication, menant presque toujours à un changement d'adresse IP (un "*smartphone*" qui était connecté en 4G et qui rentre au bercail où il bascule vers la WiFi). F17 demande que WebRTC puisse continuer à fonctionner dans ce cas, pour éviter le « Allo, Alice, je vais passer en WiFi, je te rappelle ».

Satisfaits de ces services, Alice et Bob multiplient leur exigence. Voilà tout à coup qu'apparait dans le RFC leur désir de partager l'écran, pendant la communication, avec un tiers, que ce dernier puisse suivre ce qu'il se passe (exigence F36).

Laissons maintenant Alice et Bob tranquilles, ils ont peut-être enfin réussi à communiquer. Le RFC passe ensuite à des scénarios plus collectifs, autour du hockey. Un chercheur de talents est à un match avec son téléphone portable (qui a deux caméras, une devant et une derrière), il filme les joueurs qui lui semblent les plus prometteurs et il veut en discuter en télé-conférence avec le patron du club, à

distance, et qui regarde le match transmis par le chercheur de talents. WebRTC doit donc pouvoir gérer de nombreux flux multimédias (exigence F25).

Ce genre de demandes s'étend à tout système de conférence à plusieurs, loin du cas simple des seuls Alice et Bob. WebRTC doit jongler avec les flux et de nombreux pairs qui veulent les recevoir (exigences F23 à F29).

C'est bien joli de communiquer avec les autres utilisateurs de WebRTC mais, souvent, on voudrait communiquer avec des gens qui n'ont pas WebRTC, en passant par une passerelle vers leur système, par exemple vers la téléphonie classique. Un premier exemple est celui d'un opérateur qui permet à ses abonnés de passer un appel téléphonique depuis un navigateur Web. L'utilisateur se connecte au site de l'opérateur, s'authentifie, puis appelle qui il veut, comme s'il utilisait un téléphone traditionnel. WebRTC doit donc gérer les codecs traditionnels de la téléphonie (exigence F31).

Si le service qu'on appelle a des commandes comme « pour signaler un problème, appuyez sur la touche 1 », il faut évidemment un moyen de le faire (exigence F32).

Voilà, je n'ai pas tout indiqué mais cela vous donne déjà une bonne idée de ce que WebRTC doit permettre. La section 3 de notre RFC résume toutes ces exigences de manière synthétique, si vous ne voulez pas lire tout le RFC.

Notez que, comme avec tout système qui apporte de nouvelles possibilités, il y a aussi de nouveaux risques. La section 4 liste les risques connus de WebRTC. Notamment, il faut obtenir le consentement éclairé de l'utilisateur avant de permettre l'utilisation du micro et de la caméra (songez aux facilités d'espionnage qu'on aurait si un script WebRTC sur un site Web pouvait allumer la caméra...) et il faut prévenir clairement l'utilisateur que micro et caméras sont allumés.