RFC 8445 : Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 janvier 2019

Date de publication du RFC : Juillet 2018

https://www.bortzmeyer.org/8445.html

Le problème de la traversée des routeurs NAT est traité dans plusieurs RFC, de façon à réparer la grossière erreur qu'avait été le déploiement massif du NAT plutôt que celui de IPv6. ICE est un « méta-protocole », orchestrant plusieurs protocoles comme STUN et TURN pour arriver à découvrir un canal de communication malgré le NAT. ICE avait été normalisé il y a huit ans et le RFC 5245 [1], que notre nouveau RFC remplace. Le protocole ne change guère, mais sa description normalisée est sérieusement modifiée.

L'objectif principal est la session multimédia, transmise entre deux terminaux, en général en UDP. ICE est donc utilisé, par exemple, par les solutions de téléphonie sur IP. Ces solutions ont déjà dû affronter le problème du NAT (RFC 3235) et ont développé un certain nombre de techniques, dont la bonne utilisation est l'objet principal de notre RFC.

En effet, les protocoles de téléphonie sur IP, comme SIP (RFC 3261) ont en commun d'avoir une session de **contrôle** de la connexion, établie par l'appelant en TCP (et qui passe en général assez bien à travers le NAT, si l'appelé utilise un relais public) et une session de transport des **données**, en général au-dessus d'UDP. C'est cette session qui est en général perturbée par le NAT. La session de contrôle ("*signaling session*") transmet au partenaire SIP l'adresse IP de son correspondant mais, s'il y a deux domaines d'adressage séparés (par exemple un partenaire sur le domaine public et un autre dans le monde des adresses privées du RFC 1918), cette adresse IP ainsi communiquée ne sert pas à grand'chose.

On voit donc que le non-déploiement d'IPv6, qui aurait permis de se passer du NAT, a coûté très cher <https://www.bortzmeyer.org/ipv6-et-l-echec-du-marche.html> en temps de développement. Notre RFC fait 100 pages (et d'autres RFC doivent être lus en prime comme ceux de STUN, TURN, et le

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5245.txt

futur RFC décrivant l'utilisation de ICE par SIP et SDP), et se traduira par du code réseau très complexe, uniquement pour contourner une mauvaise décision.

Des solutions au NAT, soit standard comme STUN (RFC 5389) soit purement spécifiques à un logiciel fermé comme Skype ont été développées. Mais aucune n'est parfaite, car tous les cas sont spécifiques. Par exemple, si deux machines sont sur le même réseau local, derrière le même routeur NAT, faire appel à STUN est inutile, et peut même échouer (si le routeur NAT ne supporte pas le routage en épingle à cheveux, où un paquet d'origine interne retourne vers le réseau interne). Le principe d'ICE est donc de décrire comment utiliser plusieurs protocoles de traversée de NAT, pour trouver la solution optimale pour envoyer les paquets de la session de données. Et ceci alors que les machines qui communiquent ne savent pas en général si elles sont derrière un NAT, ni, si oui, de quel type de NAT il s'agit.

Le principe d'ICE, exposé dans la section 2 et détaillé dans les sections 5 et suivantes, est donc le suivant : chaque partenaire va déterminer une liste de paires d'adresses de transport candidates (en utilisant leurs adresses IP locales, STUN et TURN), les tester et se mettre d'accord sur la « meilleure paire ». Une fois ces trois étapes passées, on va pouvoir communiquer. Une adresse de transport est un couple (adresse IP, port).

La première étape (section 5) est donc d'établir la liste des paires, et de la trier par ordre de priorité. La section 5.1.2.1 recommande une formule pour calculer la priorité, formule qui fait intervenir une préférence pour le type d'adresses IP (une adresse locale à la machine sera préférée à une adresse publique obtenue par STUN, et celle-ci sera préférée à l'adresse d'un serveur relais TURN, puisque STUN permettra ensuite une communication directe, sans relais, donc plus rapide) et une préférence locale, par exemple pour les adresses IPv6 par rapport à IPv4.

La deuxième étape est de tester les paires (sections 2.2 et 7) pour déterminer celles qui marchent. (Les paires ont d'abord été triées par priorité à l'étape précédente.) ICE poursuit les tests, même en cas de succès, pour voir si un meilleur RTT peut être trouvé (c'est un sérieux changement par rapport au précédent RFC).

Pour ces tests, ICE utilise STUN, mais avec des extensions spécifiques à ICE (sections 7.1 et 16).

La troisième et dernière étape d'ICE (section 8) est de sélectionner la meilleure paire (en terme de priorité) parmi celles qui ont fonctionné. Les couples (adresse IP, port) de celles-ci seront alors utilisées pour la communication.

Des exemples détaillés d'utilisation d'ICE, avec tous les messages échangés, figurent dans la section 15 du RFC.

L'ensemble du processus est relativement compliqué et nécessite de garder un état sur chaque partenaire ICE, alors qu'ICE est inutile pour des machines qui ont une adresse IP publique. La section 2.5 introduit donc la possibilité de mises en œuvre légères ("*lite implementation*") d'ICE, qui peuvent interagir avec les autres machines ICE, sans avoir à gérer tout le protocole (cf. aussi section 8.2).

Tous ces tests prennent évidemment du temps, d'autant plus de temps qu'il y a de paires d'adresse de transport « nominées ». C'est le prix à payer pour la plus grande souplesse d'ICE : il sera toujours plus lent que STUN seul.

Les protocoles de téléphonie sur IP ayant eu leur part de vulnérabilités, la section 19, sur la sécurité, est très détaillée. Par exemple, une attaque classique est d'établir une communication avec un partenaire,

puis de lui demander d'envoyer le flux de données vers la victime. C'est une attaque par amplification classique, sauf que l'existence d'une session de données séparée de la session de contrôle fait qu'elle ne nécessite même pas de tricher sur son adresse IP (et les techniques du RFC 2827 sont donc inefficaces). Cette attaque, dite « attaque du marteau vocal », peut être combattue grâce à ICE, puisque le test de connectivité échouera, la victime ne répondant pas (puisqu'elle n'a rien demandé). Si tout le monde utilise ICE, cette attaque peut donc complètement disparaitre.

Notez que le lien direct entre participants est excellent pour les performances, mais est mauvais pour la vie privée (voir la section 19.1) puisqu'il révèle les adresses IP de son correspondant à chaque participant. (WebRTC a le même problème.)

D'innombrables détails compliquent les choses et expliquent en partie la taille de ce RFC. Par exemple, la section 11 décrit l'obligation d'utiliser des "*keepalives*", des paquets inutiles mais qui ont pour seul but de rappeler au routeur NAT que la session sert toujours, afin que les correspondances entre une adresse IP interne et une adresse externe restent ouvertes (le flux de données ne suffit pas forcément, car il peut y avoir des périodes d'inactivité).

Enfin, une intéressante section 17 décrit les problèmes pratiques du déploiement. Par exemple, la planification de la capacité des serveurs est discutée en 17.2.1. Un serveur STUN n'a pas besoin de beaucoup de capacité <https://www.bortzmeyer.org/capacite.html>, mais un serveur TURN, oui, puisqu'il relaie tous les paquets, y compris ceux de la session de données.

La première version d'ICE ne gérait que l'UDP mais, depuis la publication du RFC 6544, TCP est également accepté.

La section 18 discute de la sortie, que le RFC 3424 impose à toutes les propositions concernant le NAT : est-ce que le contournement du NAT pourra être supprimé par la suite, ou bien sera-t-il lui-même un élément d'ossification ? En fait, ICE est utile même dans un Internet sans NAT (par exemple grâce à IPv6), pour tester la connectivité.

La section 21 décrit, mais de façon sommaire, les changements depuis le RFC 5245. Le RFC a été pas mal réorganisé et a un plan différent de celui du RFC 5245. Les principaux changements techniques :

— Suppression de la possibilité d'arrêter les tests dès la découverte d'une paire possible,
— Protocole de contrôle ("*signaling protocol*") désormais déplacé dans un RFC à part (pas encore paru).

Pour celles et ceux qui veulent creuser vraiment le RFC, et comprendre tous les choix techniques effectués, l'annexe B est là pour cela.

Il existe plusieurs mises en œuvres d'ICE en logiciel libre, comme pjnath <http://blog.pjsip.org/2007/04/06/introducing-pjnath-open-source-ice-stun-and-turn/>, qui fait partie de la bibliothèque PJSIP <http://www.pjsip.org/>, ou comme Nice <http://nice.freedesktop.org/>.