RFC 5780 : NAT Behavior Discovery Using STUN

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 19 mai 2010

Date de publication du RFC : Mai 2010

https://www.bortzmeyer.org/5780.html

La norme STUN, à son origine en mars 2003 (RFC 3489 [1]), avait mis dans un seul RFC beaucoup de choses, comme la capacité à découvrir son adresse IP publique derrière le NAT, ou la capacité à découvrir le comportement du routeur NAT, et visait à être un mécanisme complet pour permettra la traversée d'un des engins qui enferment les machines situées derrière eux dans des adresses IP privées. Mais « qui trop embrasse mal étreint » et STUN avait été critiqué pour ne pas assez séparer ces différentes fonctions. Notamment, le mécanisme de découverte du comportement du routeur NAT a été jugé, avec l'expérience, trop fragile et peu fiable, vue la variété des comportements possibles (et l'absence de normes respectées sur ce point). Pire, dans certains cas, le comportement du routeur NAT variait dans le temps (par exemple parce qu'une table était pleine, cf. section 2). Le successeur du RFC 3489, le RFC 5389, a donc décidé de se concentrer sur le protocole de base, laissant des RFC comme notre RFC 5780, définir des usages (la section 1 résume ce choix). Notre RFC décrit donc comment utiliser STUN pour essayer de déterminer le comportement du routeur NAT.

Un routeur NAT alloue des "*bindings*" entre un couple {adresse IP, port} interne (l'adresse IP est généralement une adresses IP privée, telle que décrite dans le RFC 1918) et un couple {adresse IP, port} externe. Par exemple, si une machine sur le réseau local écrit depuis le couple {192.171.1.2, 45342}, le routeur dont l'adresse publique est 192.0.2.129, peut créer un "*binding*" de {192.0.2.129, 8662} vers {192.171.1.2, 45342}. Tout paquet envoyé depuis une machine de l'Internet à l'adresse 192.0.2.129, port 8662, va alors être transmis à l'adresse 192.171.1.2, port 45342.

Tous les routeurs NAT font cela. Mais ils diffèrent sur bien des points (la section 4 du RFC 4787 les énumère). Par exemple :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3489.txt

— Combien de temps durera le "*binding*" si aucun paquet IP ne passe ? Dans les réseaux Wifi des hôtels, il est courant qu'il ne dure que quelques dizaines de secondes, ce qui ne pose pas de problème à HTTP mais est insupportable pour SSH. (Section 3.3)
— Si la même machine interne écrit, depuis le même port, à une autre machine sur l'Internet, le même "*binding*" sera t-il utilisé ? Autrement dit, dépend t-il uniquement de l'adresse source ("*Endpoint-Independent Mapping*", dit le RFC 4787) ou bien aussi de la destination ?
— Est-ce que le routeur autorise les virages en épingle à cheveux, c'est-à-dire les cas où un paquet envoyé vers l'extérieur doit en fait être acheminé vers le réseau interne (car les deux machines ne savent pas qu'elles sont derrière le même NAT) ? (Section 3.4)
— Est-ce que le routeur traduit les adresses IP présentes dans le flux de données (une idée catastrophique mais certains le font) ? (Section 3.6).
— Etc

Notre protocole doit donc permettre de découvrir une réponse à ces questions. Il se base sur STUN (avec quelques attributs supplémentaires décrits dans ce RFC, section 7). L'application qui l'utilise peut alors se servir de ses découvertes pour ajuster son comportement. Par exemple, la découverte d'une très courte durée de vue des "*bindings*" peut l'amener à envoyer des paquets "*keepalive*".

Pour tenir compte des critiques qui avaient été faites à STUN, le RFC précise bien (section 1) que ce protocole est expérimental, que l'application ne doit pas prendre au pied de la lettre les informations trouvées et qu'aucun procédé ne peut être complètement fiable dans ce domaine.

La section 2 décrit des cas où la découverte des propriétés du routeur NAT peut être utile, par exemple pour sélectionner les participants à un réseau pair-à-pair qui devront jouer un rôle particulier, par exemple de routage. La section 2.2, sur un protocole de pair-à-pair hypothétique qui utiliserait STUN, est particulièrement détaillée et donne un très bon exemple d'un mécanisme de « survie dans un monde de NAT ». Une lecture à recommander à tous les auteurs d'un protocole P2P.

La section 3 décrit le protocole en détail. Le serveur STUN est recherché dans le DNS via les enregistrements SRV (RFC 2782 et section 5.1). Chaque sous-section traite ensuite d'une question particulière de la liste donnée ci-dessus. Par exemple, pour déterminer si le "*binding*" dépend de l'adresses de destination, le client STUN écrit aux deux adresses IP du serveur STUN, en utilisant l'attribut `OTHER-ADDRESS` (c'est pour cela qu'un serveur STUN pour ce protocole ne peut pas avoir qu'une seule adresse, à noter que l'ancienne version de STUN imposait deux adresses IP dans tous les cas) et regarde si les résultats sont différents (section 3.2). Autre exemple, la section 3.3 est consacrée à la détermination de la durée de vie du "*binding*" en faisant un essai, attendant un certain temps, puis refaisant l'essai pour voir s'il marche toujours (inutile de dire que ce test est un des moins fiables, car la durée de vie d'un "*binding*" peut dépendre du remplissage des tables du routeur, section 4.6). Les virages en épingle à cheveux (paquets envoyés d'une machine située derrière le routeur à une autre machine située sur le même réseau, en passant par le routeur) sont exposés en section 3.4, qui explique comment déterminer si ces virages fonctionnent. La section 3.6 explique comment détecter la réécriture des adresses IP en comparant les attributs `MAPPED-ADDRESS` et `XOR-MAPPED-ADDRESS`. Oui, c'est une **abomination**, qui justifierait un rétablissement de la peine de mort mais certains routeurs examinent tous les octets qui passent et, lorsqu'ils trouvent quatre octets qui sont identiques à l'adresse IP NATée, les remplacent par l'adresse publique... C'est un des plus beaux exemples de délire d'un programmeur réseau.

La section 4 décrit en détail les algorithmes que doivent suivre clients et serveurs. Par exemple, 4.4 parle de la nécessité de faire tourner les tests en parallèle (certains peuvent être longs) et 4.5 explicite le mécanisme de découverte de la durée de vie des "*bindings*" (tout en notant que les clients doivent s'attendre à des résultats incohérents car cette durée de vie peut varier, par exemple selon la charge du routeur).

Pourquoi notre RFC 5780 sort-il si longtemps après le RFC 5389 ? Parce que sa genèse a été difficile : des points comme le tirage au hasard des ports utilisés (sections 4.1 et 9.2), ou comme les nouveaux enregistrements SRV (`stun-behavior` au lieu du traditionnel `stun`), ont suscité de longues discussions.