RFC 8489 : Session Traversal Utilities for NAT (STUN)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 février 2020

Date de publication du RFC : Février 2020

https://www.bortzmeyer.org/8489.html

Le NAT a toujours été une des plaies de l'Internet, entre autres parce qu'il perturbe les applications qui veulent transmettre une référence à leur adresse IP. STUN, décrit dans ce RFC, est un des protocoles qui permet de limiter les dégâts. (Il s'agit ici d'une mise à jour du RFC 5389 [1].)

Pour plusieurs raisons, dont le manque d'adresses IPv4, de nombreuses machines sont connectées à l'Internet derrière un routeur qui fait du NAT. Leur adresse étant alors privée, elles ne peuvent pas la transmettre à l'extérieur, ce qui est une gêne considérable pour tous les protocoles qui reposent sur la référence à des adresses IP, comme SIP. SIP lui-même (RFC 3261) marche à travers le NAT mais les données audio ou vidéo transmises (typiquement par RTP) le sont à une adresse IP que l'appelant donne à l'appelé et, si cette adresse est privée, le flux multi-média n'arrivera jamais.

La bonne solution serait de déployer IPv6, qui fournit des adresses en quantité illimitée, ou à la rigueur de permettre un meilleur contrôle du routeur NAT avec PCP (RFC 6887) mais, en attendant, STUN est une solution simple et qui marche souvent.

STUN s'inscrit donc dans la catégorie des protocoles de « réparation » du NAT, catégorie décrite dans le RFC 3424.

STUN résout partiellement le problème des NAT de la manière suivante : un serveur STUN doit être installé quelque part sur l'Internet public (il existe des serveurs STUN publics <http://olegh.ftp.sh/public-stun.txt>) et reçoit des demandes envoyées en UDP (ou TCP, voir sections 6.2.2 et 6.2.3, ou encore TLS ou DTLS) au port 3478. Le serveur STUN peut alors connaitre l'adresse publique, celle mise par le routeur NAT et, en répondant au client, il pourra informer celui-ci « Voilà à quoi tes paquets ressemblent, vus de l'extérieur ».

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5389.txt

Le nom du serveur STUN est typiquement configuré dans le client, puis le serveur recherché dans le DNS via les enregistrements SRV (RFC 2782), comme détaillé dans la section 8. (Le traditionnel client Unix en ligne de commande, montré plus loin, ne connait pas les SRV.)

Le principe est simple, mais le RFC est plus compliqué que cela, notamment en raison des problèmes de sécurité (la section 16 du RFC est très détaillée sur ce point). Par exemple, le client STUN doit en fait commencer par une connexion TCP pour obtenir un mot de passe temporaire. (D'autres méthodes d'authentification sont possibles.)

Lorsque STUN fonctionne sur UDP, ou bien sur DTLS, le client STUN doit également être préparé à rémettre les paquets qui seraient perdus (section 6.2.1).

Un autre problème est qu'il y a en fait plusieurs sortes de NAT (voir par exemple le RFC 2663). Par exemple, certains routeurs NAT ("*Symmetric NAT*" pour le RFC 3489) n'attribuent pas l'adresse externe uniquement en fonction de la **source** mais aussi en fonction de la **destination**. Ceux-ci ne fonctionnent pas avec STUN, qui a besoin de NAT "*cone*", c'est-à-dire où les paquets d'une même machine auront toujours la même adresse IP externe.

Voici une démonstration d'un client STUN <http://sourceforge.net/projects/stun/> (assez ancien mais qui marche) qui, en se connectant à un serveur STUN public <https://www.voip-info.org/stun> va apprendre son adresse IP publique, et le port, les deux ensemble formant l'**adresse de transport** (cf. section 4) :

```
% ifconfig -a
hme0: [...] inet 172.19.1.2
(Adresse privée, probablement NATée...)

% stun stun.1und1.de -v
STUN client version 0.97
...
MappedAddress = 192.0.2.29:30189
SourceAddress = 212.227.67.33:3478
ChangedAddress = 212.227.67.34:3479
Unknown attribute: 32800
ServerName = Vovida.org 0.96
...
Primary: Independent Mapping, Port Dependent Filter, preserves ports, no hairpin
...
(Mon adresse telle que vue de l'extérieur est donc 192.0.2.29, à noter que le terme
MappedAddress n'est plus utilisé, le RFC parle de ReflexiveAddress.)
```

Une fois l'adresse extérieure détectée, tout n'est pas terminé car rien n'indique, par exemple, que les paquets vont effectivement passer. C'est pour cela que la section 13 insiste que STUN n'est qu'un outil, devant s'insérer dans une solution plus globale, comme ICE (RFC 8445). À lui seul, STUN ne permet pas la traversée de tous les NAT. La section 13 décrit en détail ce concept d'utilisation de STUN et les règles que doivent suivre les protocoles qui utilisent STUN.

La section 5 décrit le format des paquets. Un paquet STUN doit comprendre l'indication d'une **méthode**, qui indique le genre de services que désire le client. Notre RFC 8489 ne décrit qu'une seule méthode, `Binding`, qui permet d'obtenir son adresse IP extérieure mais TURN (RFC 8656), par exemple, en définit d'autres. Après la méthode et un identificateur de transaction (qui sert au serveur STUN à séparer ses clients), suivent les **attributs**, encodés en TLV (la liste des attributs figure en section 14). Les numéros des attributs sont compris entre 0x0000 et 0x7FFF s'ils doivent être reconnus par le serveur (autrement, la requête est rejetée) et entre 0x8000 et 0xFFFF si leur compréhension est facultative (cette

notion d'attributs obligatoires ou facultatifs facilite les évolutions ultérieures du protocole). Il y a aussi un "*magic cookie*", une valeur fixe (`0x2112A442`) qui sert à reconnaitre les agents STUN conformes (il n'existait pas dans les vieilles versions de STUN).

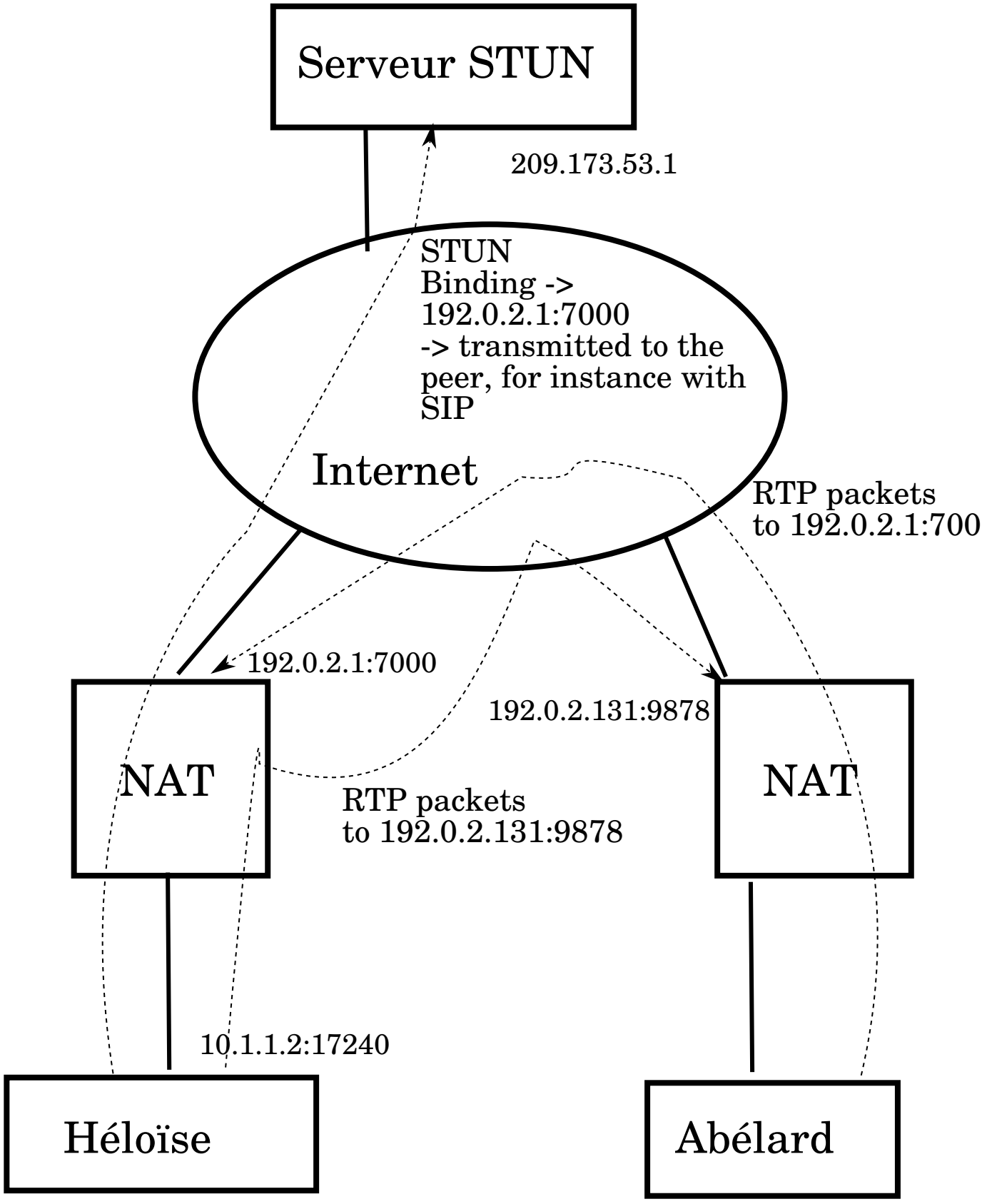
La section 19 de notre RFC résume les changements faits depuis le prédécesseur, le RFC 5389. Rien de bien crucial, d'autant plus que la plupart de ces changements sont l'intégration de RFC qui avaient étendu le protocole STUN après la parution du RFC 5389 :

— DTLS est désormais intégré (RFC 6347),
— L'algorithmes pour calculer le délai avant retransmission a été modifié suivant le RFC 6298,
— Les URI du RFC 7064 sont désormais inclus,
— SHA-256 et des nouveaux algorithmes pour TLS arrivent,
— Les identificateurs et mots de passe peuvent être en Unicode (RFC 8265),
— Possibilité d'anonymat pour les identificateurs (ce qui a nécessité des nouvelles règles dans le numnique envoyé par le serveur, pour éviter les attaques par repli),
— Ajout de l'attribut `ALTERNATE-DOMAIN`, pour indiquer que le serveur proposé en alternative peut avoir un autre nom,
— Et pour les programmeurs, des vecteurs de test en annexe B (venus du RFC 5769).

Il existe de nombreux logiciels clients et serveurs pour STUN (mais pas forcément déjà conforme aux légers changements de ce RFC). Le client est en général embarqué dans une application qui a besoin de contourner le NAT, par exemple un "*softphone*", ou bien un navigateur Web qui fait du WebRTC. On peut citer :

— Le serveur coturn <`https://github.com/coturn/coturn/`>,
— Le serveur Restund <`http://www.creytiv.com/restund.html`>,
— Le client et serveur stuntman <`http://www.stunprotocol.org/`>,
— Le serveur turnserver <`http://turnserver.sourceforge.net/`>.

Et si vous cherchez une liste de serveurs STUN publics, il y a celle-ci <`https://gist.github.com/zziuni/3741933`>.



Héloïse wants to receive RTP packets from Abélard. Both Héloïse and Abélard asks the STUN server for their public addresses, then sends them to each other.