RFC 6062 : Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 30 novembre 2010

Date de publication du RFC : Novembre 2010

http://www.bortzmeyer.org/6062.html

Le protocole TURN, qui permet à deux machines coincées derrière des routeurs NAT de communiquer en utilisant un relais tiers, a été normalisé dans le RFC 5766 [1], dans une version minimale, uniquement pour IPv4 et seulement pour UDP, les services « incontestables », qui étaient pressés d'avoir une solution. Depuis, des extensions à TURN sont développées pour d'autres services. C'est ainsi que ce RFC 6062 permet d'utiliser TURN pour établir des liaisons avec TCP.

TURN, tel que normalisé dans le RFC 5766, pouvait déjà utiliser TCP entre le client et le relais (le serveur). Mais, en sortie du relais, vers l'autre pair, c'était forcément de l'UDP. Cela posait un problème (section 1 du RFC) lorsque l'autre pair ne pouvait pas faire d'UDP, ou bien lorsque les propriétés de TCP (notamment de fiabilité) étaient nécessaires. Supposons un logiciel d'audio-conférence : le son est typiquement transmis en UDP (car la rapidité est préférable à la fiabilité, pour ce service) mais si on veut ajouter un système permettant aux participants de se montrer des images, TCP est certainement préférable, afin de transférer le fichier représentant l'image. Avec le protocole TURN, cela se nomme une « allocation TCP » et ce sera désormais possible, aussi bien en demandant une session TCP entrant vers un pair, qu'en acceptant une session TCP sortant d'un pair.

La section 3 rappelle le fonctionnement de TURN : le client TURN ouvre une connexion avec le serveur TURN (qui servira de relais) pour le contrôle (messages TURN, pour ouvrir et fermer des connexions). De plus, pour chaque connexion avec un pair, le client TURN aura une connexion de données pour faire passer ses données (son et image, dans l'exemple précédent), avec divers protocoles (par exemple RTP). Pour chacune de ces connexions de données, il y aura également une connexion entre le relais (le serveur TURN) et le pair avec qui on communique.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5766.txt

Pour avoir une allocation TCP, le client TURN va envoyer, sur la connexion de contrôle, une requête `Allocate` avec un attribut `REQUESTED-TRANSPORT` indiquant TCP. Le serveur, s'il accepte la requête, écoutera alors en TCP. Chaque fois qu'il voudra envoyer des donnée à un pair, le client enverra une requête `Connect` au serveur TURN et celui-ci se connectera à son tour en TCP au pair. Enfin, le client devra ouvrir une nouvelle connexion TCP pour les données (voir section 4.3). Une fois cette dernière connexion établie, le relais « connectera » les deux connexions (client-¿relais et relais-¿pair) puis relaiera les données entre les deux pairs.

Dans le cas où le client accepte que ce soit le pair qui initie la session, il indique d'abord au serveur qu'il accepte les connections entrantes (requête `CreatePermission`), le serveur écoute en TCP et, établit la connexion avec le pair et, lorsque c'est fait, signale au client que le pair est arrivé. Le client peut alors faire une connexion TCP vers le serveur puis tout continue comme dans le cas précédent.

Les détails pratiques figurent dans la section 4 (obligations du client) et 5 (obligations du serveur). On notera que, pour relayer du TCP, la connexion entre le client et le serveur **doit** être en TCP (pour relayer de l'UDP, elle pouvait être en UDP ou TCP). Outre l'attribut `REQUESTED-TRANSPORT` déjà mentionné, qui doit avoir la valeur 6 (pour TCP), le client doit prendre soin de ne **pas** inclure des attributs spécifiques à UDP comme `EVEN-PORT` (qui force l'utilisation d'un port de numéro pair, et qui n'est utile que pour UDP, voir le RFC 5766, section 14.6). TURN utilisera un attribut `CONNECTION-ID` qui sera indiqué lors de l'acceptation d'une allocation TCP, et devra être donné par le client lors de sa connexion TCP ultérieure, pour permettre au relais de mettre les deux opérations en correspondance.

Le serveur, lui, doit accepter toutes les connexions TCP entrantes (section 5.3), du moment qu'une allocation a été faite (et, donc, un port réservé). Un essai avec telnet montrera donc toujours "*Connected to $HOSTNAME*". C'est ensuite que le serveur TURN vérifie que le client avait donné la permission de se connecter. Sinon (ou si le client, bien qu'il ait donné la permission, ne se manifeste pas lorsqu'on lui signale que le pair est arrivé), la session, à peine acceptée, sera close d'autorité.

La section 6 décrit l'enregistrement à l'IANA des nouvelles méthodes (`Connect`, `ConnectionBind` et `ConnectionAttempt`), d'un nouvel attribut `CONNECTION-ID` et de nouveaux codes d'erreur 446 (déjà une connexion pour ce couple adresse/port) et 447 (échec de la connexion avec le pair) dans le registre des paramètres STUN <`https://www.iana.org/assignments/stun-parameters/stun-parameters.xhtml`> (rappelez-vous que TURN est défini comme une extension de STUN).

Enfin, la section 7, sur la sécurité, met en garde contre un petit piège : entre l'acceptation d'une connexion et son branchement de l'autre côté, le relais risque de recevoir des données, qu'il doit tamponner (section 5.2). Le tampon risquant de devenir trop gros, la section 7 demande qu'une taille maximale lui soit fixée.

Une mise en œuvre se trouve dans le service de Viagénie <`http://numb.viagenie.ca/`> mais ne semble pas activée à l'heure actuelle. Pour une implémentation libre, turnserver <`http://turnserver.sourceforge.net/`> a cette option TCP depuis la version 0.4.

Merci à Simon Perreault pour sa relecture et ses remarques.