Groupe de travail Réseau
Request for Comments: 3489
Catégorie : Standards Track
Mars 2003

J. Rosenberg
J. Weinberger
Dynamicsoft
C. Huitema : Microsoft
R. Mahy : Cisco

STUN – Traversée simple du protocole de datagrammes d'utilisateur (UDP) à travers les traducteurs d'adresse réseau (NAT)

Statut du présent Mémo

Le présent document spécifie un protocole de normalisation Internet pour la communauté Internet, et appelle à discussion et suggestions en vue de son amélioration. Prière de se rapporter à l'édition en cours des "Internet Official Protocol Standards" (*normes officielles du protocole Internet*) (STD 1) pour connaître l'état de la normalisation et le statut du présent protocole. La distribution du présent mémo n'est pas soumise à restrictions.

Déclaration de copyright

Résumé

La traversée simple du protocole de datagrammes d'utilisateur (UDP) à travers les traducteurs d'adresse réseau (NAT) (STUN, *Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*) est un protocole léger qui permet aux applications de découvrir la présence et le type des NAT et pare-feu entre eux et l'Internet public. Il fournit aussi aux applications la capacité de déterminer les adresses du protocole Internet (IP) public qui leur sont allouées par le NAT. STUN fonctionne avec de nombreux NAT existants, et n'exige d'eux aucun comportement particulier. Il en résulte qu'il permet à une grande variété d'applications de fonctionner sur l'infrastructure de NAT existante.

Table des matières

1. Déclaration d'applicabilité

Le présent protocole n'est pas une panacée pour tous les problèmes associés aux NAT. Il n'active pas les connexions TCP entrantes à travers le NAT. Il permet aux paquets UDP de passer à travers le NAT, mais seulement par un sous-ensemble des types de NAT existants. En particulier, STUN ne permet pas le passage des paquets UDP entrants à travers les NAT symétriques (définis plus loin), qui sont courants dans les grandes entreprises. Les procédures de découverte de STUN se fondent sur l'hypothèse que le NAT traite UDP ; une telle hypothèse peut s'avérer infondée dans l'avenir avec le développement de nouveaux appareils. STUN ne fonctionne pas lorsque il est utilisé pour obtenir une adresse afin de communiquer avec un homologue qui se trouve être derrière le même NAT. STUN ne fonctionne pas lorsque le serveur STUN n'est pas dans un domaine d'adresse partagé commun. Voir à la section 14 une discussion plus complète des limites de STUN.

2. Introduction

Les traducteurs d'adresse réseau (NAT, *Network Address Translator*), bien qu'ils procurent de nombreux avantages, ont également de nombreux inconvénients. Le plus gênant de ces inconvénients est le fait qu'ils cassent de nombreuses applications IP existantes, et rendent difficile le développement de nouvelles. Des lignes directrices ont été développées [8] qui décrivent comment construire des protocoles "compatibles NAT", mais de nombreux protocoles ne peuvent tout simplement pas être construits en conformité à ces lignes directrices. Comme exemples de tels protocoles, on trouve presque tous les protocoles d'homologue à homologue, tels que les communications multimédia, le partage de fichiers et les jeux.

Pour essayer de résoudre ce problème, les passerelles de couche d'application (ALG, *Application Layer Gateways*) ont été incorporées dans les NAT. Les ALG effectuent les fonctions de couche d'application nécessaires pour permettre à un protocole particulier de traverser un NAT. Normalement, cela implique de réécrire les messages de couche d'application pour qu'ils contiennent les adresses traduites, plutôt que ceux insérés par l'expéditeur du message. Les ALG rencontrent des difficultés sérieuses, en matière de mesurabilité, de fiabilité et de vitesse de développement de nouvelles applications. Pour résoudre ces problèmes, le protocole de communications par boîtier de médiation (MIDCOM, *Middlebox Communications*) est en cours de développement [9]. MIDCOM permet à une entité d'application, telle qu'un client final ou un serveur réseau de quelque sorte (tel qu'un mandataire de protocole d'initialisation de session (SIP, *Session Initiation Protocol*) [10]) de contrôler un NAT (ou pare-feu), afin d'obtenir des liens de NAT et des micro-sas *(pinholes)* ouverts ou fermés. De cette façon, les NAT et les applications peuvent être séparés une fois de plus, éliminant la nécessité d'incorporer les ALG dans les NAT, ce qui résout les limitations imposées par les architectures courantes. Malheureusement, MIDCOM exige des mises à niveau sur les NAT et pare-feu existants, en plus des composants d'application. La mise à niveau complète de ces produits NAT et pare-feu prendra beaucoup de temps, peut-être des années. Ceci est dû, en partie, au fait que les développeurs de NAT et pare-feu ne sont pas les mêmes que ceux qui développent et utilisent les applications. Il en résulte que l'incitation à l'amélioration de ces appareils sera faible dans de nombreux cas. Considérons, par exemple, un bar Internet dans un aéroport, qui fournit un accès par un NAT. Un utilisateur qui se connecte à un réseau muni d'un NAT peut vouloir utiliser un service d'homologue à homologue, mais il ne le peut pas, parce que le NAT ne le prend pas en charge. Comme les administrateurs du bar ne sont

pas ceux qui fournissent le service, ils ne sont pas motivés pour mettre à niveau leur équipement de NAT pour qu'il le prenne en charge, en utilisant un ALG, ou MIDCOM.

Un autre problème est que le protocole MIDCOM exige que l'agent qui contrôle le boîtier de médiation (*middlebox*) connaisse l'identité de ces boîtiers, et ait avec eux des relations qui permettent le contrôle. Dans de nombreuses configurations, ceci ne sera pas possible. Par exemple, de nombreux fournisseurs d'accès par câble utilisent un NAT en frontal de tout leur réseau d'accès. Ce NAT pourrait s'ajouter à un NAT résidentiel acheté par l'utilisateur final qui le fait fonctionner. L'utilisateur final n'aura probablement pas de relation de contrôle sur le NAT du réseau d'accès par câble, et peut même n'en pas connaître l'existence.

De nombreux protocoles propriétaires existants, tels que ceux pour les jeux en ligne (comme les jeux décrits dans la RFC 3027 [11]) et la voix sur IP, ont développé des astuces qui leur permettent de fonctionner à travers les NAT sans les changer. Le présent document essaye de reprendre certaines de ces idées, et de les codifier en un protocole d'interfonctionnement qui puisse satisfaire les besoins de nombreuses applications.

Le protocole décrit ici, Traversée simple d'UDP à travers un NAT (STUN), permet aux entités qui sont derrière un NAT de découvrir d'abord la présence d'un NAT et le type du NAT, puis d'apprendre les liens d'adresses alloués par le NAT. STUN ne requiert pas de changements sur les NAT, et fonctionne avec un nombre arbitraire de NAT en tandem entre l'entité d'application et l'Internet public.

3. Terminologie

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" doivent être interprétés comme décrit par la [RFC2119] [1] et indiquent les niveaux d'exigence pour les mises en œuvre conformes à STUN.

4. Définitions

Client STUN : un client STUN (aussi appelé simplement client) est une entité qui génère des demandes STUN. Un client STUN peut travailler sur un système terminal, tel qu'un micro-ordinateur de particulier, ou fonctionner dans un élément de réseau, tel qu'un serveur de conférence.

Serveur STUN : un serveur STUN (aussi appelé simplement serveur) est une entité qui reçoit des demandes STUN, et envoie des réponses STUN. Les serveurs STUN sont normalement reliés à l'Internet public.

5. Variantes de NAT

On suppose que les NAT sont familiers aux lecteurs. On a observé que le traitement d'UDP par un NAT varie selon les mises en œuvre. Les quatre traitements observés dans les mises en œuvre sont :

Plein cône : un NAT en plein cône est celui où toutes les demandes provenant de la même adresse et port IP internes sont transposées sur la même adresse et port IP externes. De plus, tout hôte externe peut envoyer un paquet à l'hôte interne, en envoyant un paquet à l'adresse externe transposée.

Cône restreint : un NAT en cône restreint est celui où toutes les demandes provenant de la même adresse et port IP internes sont transposées sur la même adresse et port IP externes. A la différence du NAT en plein cône, un hôte externe (avec l'adresse IP X) ne peut envoyer un paquet à l'hôte interne que si l'hôte interne avait préalablement envoyé un paquet à l'adresse IP X.

Cône à restriction de port : un NAT en cône à restriction de port est comme un NAT à cône restreint, mais la restriction inclut les numéros de port. Précisément, un hôte externe ne peut envoyer un paquet, avec l'adresse IP de source X et le port de source P, à l'hôte interne que si l'hôte interne avait préalablement envoyé un paquet à l'adresse IP X et au port P.

Symétrique : un NAT symétrique est celui dans lequel toutes les demandes provenant de la même adresse et port IP internes, à une adresse et port IP de destination spécifique, sont transposées sur la même adresse et port IP externes. Si le même hôte envoie un paquet avec la même adresse et port de source, mais une destination différente, on utilise une transposition différente. De plus, seul l'hôte externe qui reçoit un paquet peut renvoyer un paquet UDP à l'hôte interne

La détermination du type de NAT est importante dans de nombreux cas. Selon ce que l'application veut faire, il peut être nécessaire de tenir compte des particularités du comportement du NAT.

6. Généralités sur le fonctionnement

La présente section est seulement descriptive. Le comportement normatif est décrit dans les Sections 8 et 9.

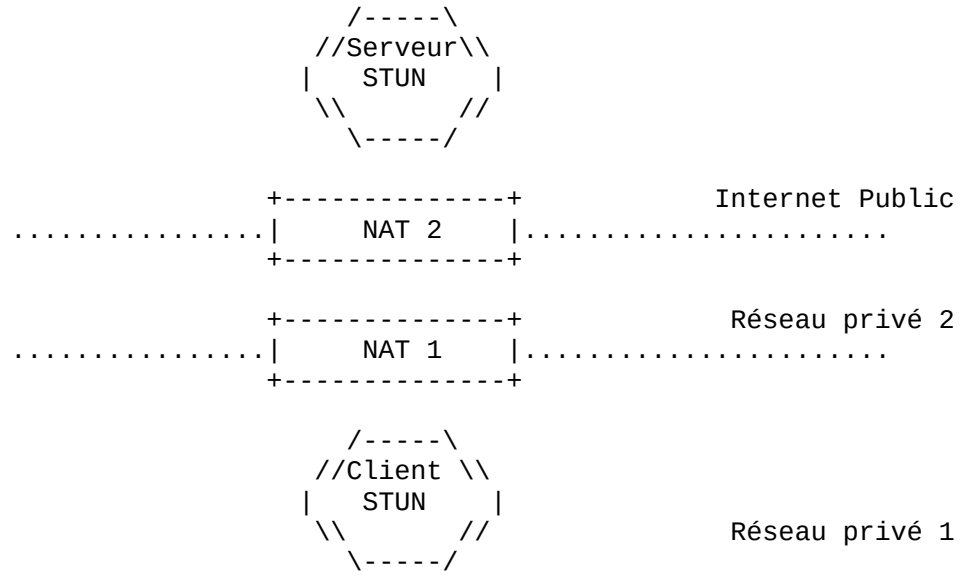


Figure 1 : Configuration de STUN

La configuration STUN typique est indiquée à la Figure 1. Un client STUN est connecté au réseau privé 1. Ce réseau se connecte au réseau privé 2 à travers le NAT 1. Le réseau privé 2 se connecte à l'Internet public à travers le NAT 2. Le serveur STUN demeure dans l'Internet public.

STUN est un simple protocole client-serveur. Un client envoie une demande à un serveur, et le serveur retourne une réponse. Il y a deux types de demandes, les demandes de lien, envoyées sous UDP, et les demandes de secret partagé, envoyées sous TLS [2] sur TCP. Les demandes de secret partagé demandent au serveur de retourner un nom d'utilisateur (*username*) et un mot de passe (*password*) temporaires. Ce nom d'utilisateur et ce mot de passe servent dans une demande de lien et réponse de lien ultérieures, pour les besoins de l'authentification et de l'intégrité du message.

Les demandes de lien servent à déterminer les liens alloués par les NAT. Le client envoie une demande de lien au serveur, sous UDP. Le serveur examine l'adresse et le port IP de source de la demande, et les copie dans une réponse qui est renvoyée au client. Il y a certains paramètres dans la demande qui permettent au client de demander que la réponse soit envoyée ailleurs, ou que le serveur envoie la réponse à partir d'un port et adresse différents. Il y a des attributs pour l'intégrité du message et l'authentification.

L'astuce est d'utiliser STUN pour découvrir la présence d'un NAT, apprendre et utiliser les liens qu'il alloue.

Le client STUN est normalement incorporé dans une application qui a besoin d'obtenir une adresse et port IP publics qui puissent être utilisés pour recevoir des données. Par exemple, il pourrait avoir besoin d'obtenir une adresse et un port IP pour recevoir du trafic de protocole de transport en temps réel (RTP, *Real Time Transport Protocol*) [12]. Lorsque l'application débute, le client STUN dans l'application envoie une demande STUN de secret partagé à son serveur, obtient un nom d'utilisateur et un mot de passe, et lui envoie alors une demande de lien (*Binding Request*). Les serveurs STUN peuvent être découverts à travers les enregistrements SRV de DNS [3], et on suppose généralement que le client est configuré avec le domaine à utiliser pour trouver le serveur STUN. Généralement, ce sera le domaine du fournisseur du service qu'utilise l'application (un tel fournisseur est incité à déployer des serveurs STUN afin de permettre aux consommateurs d'utiliser son application à travers un NAT). Bien sûr, un client peut déterminer l'adresse ou le nom de domaine d'un serveur STUN par d'autres moyens. Un serveur STUN peut toujours être incorporé dans un système terminal.

La demande de lien STUN sert à découvrir la présence d'un NAT, et à découvrir les transpositions d'adresse et port IP publics générées par le NAT. Les demandes de lien sont envoyées au serveur STUN à l'aide de UDP. Lorsqu'une demande de lien arrive au serveur STUN, elle peut être passée à travers un ou plusieurs NAT entre le client STUN et le serveur STUN. Il en résulte que l'adresse de source de la demande reçue par le serveur sera l'adresse transposée créée par le NAT le plus proche du serveur. Le serveur STUN copie cette adresse et port IP de source dans une réponse de lien STUN, et la renvoie à l'adresse et port IP de source de la demande STUN. Pour tous les types de NAT ci-dessus, cette réponse arrivera au client STUN.

Lorsque le client STUN reçoit la réponse de lien STUN, il compare l'adresse et le port IP du paquet avec l'adresse et le port IP locaux qu'il y a lié lors de l'envoi de la demande. Si elles ne correspondent pas, le client STUN est derrière un ou plusieurs NAT. Dans le cas d'un NAT de plein cône, l'adresse et port IP du corps de la réponse STUN sont publics, et peuvent être utilisés par tout hôte sur l'Internet public pour envoyer des paquets à l'application qui a envoyé la demande

STUN. Une application a seulement besoin d'écouter l'adresse et le port IP à partir desquels la demande STUN a été envoyée. Tout paquet envoyé par un hôte sur l'Internet public à l'adresse et port publics appris par STUN sera reçu par l'application.

Bien sûr, l'hôte peut n'être pas derrière un NAT en plein cône, et il ne sait pas encore derrière quel type de NAT il se trouve. Pour le déterminer, le client utilise des demandes de lien STUN supplémentaires. La procédure exacte est souple, mais devrait normalement fonctionner comme suit. Le client devrait envoyer une seconde demande de lien STUN, cette fois à une adresse IP différente, mais à partir de la même adresse et port IP de source. Si l'adresse et le port IP de la réponse sont différents de ceux de la première réponse, le client saura qu'il est derrière un NAT symétrique. Pour déterminer s'il est derrière un NAT en plein cône, le client peut envoyer une demande de lien STUN avec des fanions qui disent au serveur STUN d'envoyer une réponse à partir d'une adresse et port IP différents de ceux sur lesquels la réponse a été reçue. En d'autres termes, si le client a envoyé une demande de lien à l'adresse/port IP A/B en utilisant une adresse/port IP X/Y, le serveur STUN devrait envoyer la réponse de lien à X/Y en utilisant l'adresse/port IP de source C/D. Si le client reçoit cette réponse, il sait qu'il est derrière un NAT en plein cône.

STUN permet aussi au client de demander au serveur d'envoyer la réponse de lien à partir de la même adresse IP que celle où la demande a été reçue, mais avec un port différent. Ceci peut être utilisé pour détecter si le client est derrière un NAT en cône à restriction de port ou simplement un NAT en cône restreint.

Il convient de noter que la configuration de la Figure 1 n'est pas la seule configuration permise. Le serveur STUN peut être localisé partout, y compris chez un autre client. La seule exigence est que le serveur STUN puisse être atteint par le client, et si le client essaye d'obtenir une adresse d'acheminement public, que le serveur réside sur l'Internet public.

7. Vue d'ensemble des messages

Les messages STUN sont codés en TLV (type-longueur-valeur) en utilisant le binaire gros boutien (dans l'ordre du réseau). Tous les messages STUN commencent par un en-tête STUN, suivi par une charge utile STUN. La charge utile est une série d'attributs STUN, dont l'ensemble dépend du type de message. L'en-tête STUN contient un type de message STUN, un identifiant de transaction, et la longueur. Le type de message peut être Demande de lien, Réponse de lien, Réponse d'erreur de lien, Demande de secret partagé, Réponse de secret partagé, ou Réponse d'erreur de secret partagé. L'ID de transaction sert à corréler demandes et réponses. La longueur indique la longueur totale de la charge utile STUN, en-tête non inclus. Cela permet à STUN de fonctionner sur TCP. Les demandes de secret partagé sont toujours envoyées sur TCP (en utilisant, bien sûr, TLS sur TCP).

Plusieurs attributs STUN sont définis. Le premier est un attribut MAPPED-ADDRESS (*adresse transposée*), qui est une adresse et un port IP. Il est toujours placé dans la Réponse de lien, et il indique l'adresse et le port IP de source que le serveur a vu dans la demande de lien. Il y a aussi un attribut RESPONSE-ADDRESS (*adresse de réponse*), qui contient une adresse et un port IP. L'attribut RESPONSE-ADDRESS peut être présent dans la demande de lien, et indique où la réponse de lien doit être envoyée. Il est facultatif, et lorsqu'il n'est pas présent, la réponse de lien est envoyée à l'adresse et port IP de source de la demande de lien.

Le troisième attribut est CHANGE-REQUEST (*demande de changement*), et il contient deux fanions pour contrôler l'adresse et port IP utilisés pour envoyer la réponse. Ces fanions s'appellent "change IP" et "change port". L'attribut CHANGE-REQUEST n'est admis que dans la demande de lien. Les fanions "change IP" et "change port" sont utiles pour déterminer si le client est derrière un NAT en cône restreint ou un NAT en cône à restriction de port. Ils ordonnent au serveur d'envoyer les réponses de lien à partir de différentes adresses et ports IP de source. L'attribut CHANGE-REQUEST est facultatif dans la demande de lien.

Le quatrième attribut est CHANGED-ADDRESS (*adresse changée*). Il est présent dans les réponses de lien. Il informe le client de l'adresse et port IP de source qui seraient utilisés si le client demandait le comportement "change IP" et "change port".

Le cinquième attribut est SOURCE-ADDRESS (*adresse de source*). Il n'est présent que dans les réponses de lien. Il indique l'adresse et le port IP de source d'où la réponse a été envoyée. Il est utile pour détecter les configurations de NAT double.

Le sixième attribut est USERNAME (*nom d'utilisateur*). Il est présent dans la Réponse de secret partagé, qui attribue au client un nom d'utilisateur et un mot de passe temporaires (codés dans l'attribut PASSWORD). Le USERNAME est aussi présent dans les demandes de lien, et sert d'indice pour le secret partagé utilisé pour la protection de l'intégrité de la demande de lien. Le septième attribut, PASSWORD (*mot de passe*), ne se trouve que dans les messages de réponse de secret partagé. Le huitième attribut est MESSAGE-INTEGRITY (*intégrité du message*), qui contient une vérification d'intégrité du message sur la demande de lien ou la réponse de lien.

Le neuvième attribut est ERROR-CODE (*code d'erreur*). Il est présent dans la réponse d'erreur de lien et dans la réponse d'erreur de secret partagé. Il indique l'erreur qui est survenue. Le dixième attribut est UNKNOWN-ATTRIBUTES (*attributs inconnus*), qui n'est présent ni dans la réponse d'erreur de lien ni dans la réponse d'erreur de secret partagé. Il indique les attributs obligatoires qui étaient inconnus dans la demande. Le onzième attribut est REFLECTED-FROM (*répercuté de*), qui est présent dans les réponses de lien. Il indique l'adresse et port IP de l'expéditeur d'une demande de lien, et sert aux fins de traçabilité pour empêcher certaines attaques de déni de service.

8. Comportement du serveur

Le comportement du serveur varie selon que la demande est une demande de lien ou une demande de secret partagé.

8.1 Demandes de liens

Un serveur STUN DOIT être prêt à recevoir des demandes de lien sur quatre combinaisons d'adresse/port - (A1, P1), (A2, P1), (A1, P2), et (A2, P2). (A1, P1) représente l'adresse et port primaires, et ce sont ceux qui sont obtenus à travers les procédures de découverte du client ci-dessous. Normalement, P1 sera le port 3478, le port STUN par défaut. A2 et P2 sont arbitraires. A2 et P2 sont publiés par le serveur grâce à l'attribut CHANGED-ADDRESS, comme décrit ci-dessous.

Il est RECOMMANDÉ que le serveur vérifie l'attribut MESSAGE-INTEGRITY de la demande de lien. S'il n'est pas présent, et si le serveur exige les vérifications d'intégrité sur la demande, il

génère une réponse d'erreur de lien avec un attribut ERROR-CODE (*code d'erreur*) avec le code de réponse 401. Si l'attribut MESSAGE-INTEGRITY est présent, le serveur calcule le HMAC sur la demande comme décrit au paragraphe 11.2.8. La clé à utiliser dépend du mécanisme de secret partagé. Si la demande de secret partagé STUN a été utilisée, la clé DOIT être celle associée à l'attribut USERNAME présent dans la demande. Si l'attribut USERNAME n'était pas présent, le serveur DOIT générer une réponse d'erreur de lien.

La réponse d'erreur de lien DOIT inclure un attribut ERROR-CODE avec le code de réponse 432. Si le USERNAME est présent, mais que le serveur ne se souvient pas du secret partagé pour ce USERNAME (à cause, par exemple, d'une fin de temporisation), le serveur DOIT générer une réponse d'erreur de lien. La réponse d'erreur de lien DOIT inclure un attribut ERROR-CODE avec le code de réponse 430. Si le serveur ne connaît pas le secret partagé, mais que le HMAC calculé diffère de celui de la demande, le serveur DOIT générer une réponse d'erreur de lien avec un attribut ERROR-CODE au code de réponse de 431. La réponse d'erreur de lien est envoyée à l'adresse et port IP d'où vient la demande de lien, et elle est envoyée de l'adresse et port IP à laquelle la demande de lien a été envoyée.

En supposant que le message de vérification d'intégrité a réussi, le traitement continue. Le serveur DOIT chercher tous les attributs de la demande qui ont des valeurs inférieures ou égales à 0x7fff qu'il ne comprend pas. Si il en rencontre, le serveur DOIT générer une réponse d'erreur de lien, et il DOIT inclure un attribut ERROR-CODE avec le code de réponse 420. Cette réponse DOIT contenir un attribut UNKNOWN-ATTRIBUTES faisant la liste des attributs qui ont des valeurs inférieures ou égales à 0x7fff qu'il ne comprend pas. La réponse d'erreur de lien est envoyée à l'adresse et port IP d'où est venue la demande de lien, et envoyée de l'adresse et port IP d'où a été envoyée la demande de lien.

En supposant que la demande a été formée correctement, le serveur DOIT générer une seule réponse de lien. La réponse de lien DOIT contenir le même ID de transaction que celui contenu dans la demande de lien. La longueur dans l'en-tête de message DOIT contenir la longueur totale du message en octets, non compris l'en-tête. La réponse de lien DOIT avoir le type de message "Réponse de lien".

Le serveur DOIT ajouter un attribut MAPPED-ADDRESS à la réponse de lien. Le composant adresse IP de cet attribut DOIT être réglé à l'adresse IP de source observée dans la demande de lien. Le composant port de cet attribut DOIT être réglé au port de source observé dans la demande de lien.

Si l'attribut RESPONSE-ADDRESS était absent de la demande de lien, l'adresse et port de destination de la réponse de lien DOIT être la même que celle de l'adresse et port de source de la demande de lien. Autrement, l'adresse et port de destination de la réponse de lien DOIT être la valeur de l'adresse et port IP dans l'attribut RESPONSE-ADDRESS.

L'adresse et port de source de la réponse de lien dépendent de la valeur de l'attribut CHANGE-REQUEST et de l'adresse et port sur lesquels la demande de lien a été reçue. Elles sont résumées au Tableau 1.

Soit Da représentant l'adresse IP de destination de la demande de lien (qui sera A1 ou A2), et Dp représentant le port de destination de la demande de lien (qui sera P1 ou P2). Soit Ca représentant l'autre adresse, de sorte que si Da est A1, Ca est A2. Si Da est A2, Ca est A1. De même, soit Cp représentant l'autre port, de sorte que si Dp est P1, Cp est P2. Si Dp est P2, Cp est P1. Si le fanion

"change port" était mis dans l'attribut CHANGE-REQUEST de la demande de lien, et si le fanion "change IP" n'était pas mis, l'adresse IP de source de la réponse de lien DOIT être Da et le port de source de la réponse de lien DOIT être Cp. Si le fanion "change IP" était mis dans la demande de lien, et si le fanion "change port" n'était pas mis, l'adresse IP de source de la réponse de lien DOIT être Ca et le port de source de la réponse de lien DOIT être Dp. Lorsque les deux fanions sont mis, l'adresse IP de source de la réponse de lien DOIT être Ca et le port de source de la réponse de lien DOIT être Cp. Si aucun des fanions n'est mis, ou si l'attribut CHANGE-REQUEST est entièrement absent, l'adresse IP de source de la réponse de lien DOIT être Da et le port de source de la réponse de lien DOIT être Dp.

Fanions	**Adresse de source**	**Port de source**	**CHANGED-ADDRESS**
aucun	Da	Dp	Ca:Cp
Change IP	Ca	Dp	Ca:Cp
Change port	Da	Cp	Ca:Cp
Change IP et Change port	Ca	Cp	Ca:Cp

Tableau 1 : Impact des fanions sur la source du paquet et CHANGED-ADDRESS

Le serveur DOIT ajouter un attribut SOURCE-ADDRESS à la réponse de lien, contenant l'adresse et port de source utilisés pour envoyer la réponse de lien. Le serveur DOIT ajouter un attribut CHANGED-ADDRESS à la réponse de lien. Celui-ci contient l'adresse et port IP de source qui seraient utilisés si le client avait établi les fanions "change IP" et "change port" dans la demande de lien. Comme résumé au Tableau 1, ce sont respectivement Ca et Cp, indépendamment de la valeur des fanions de CHANGE-REQUEST.

Si la demande de lien contenait les deux attributs USERNAME et MESSAGE-INTEGRITY, le serveur DOIT ajouter un attribut MESSAGE-INTEGRITY à la réponse de lien. L'attribut contient un HMAC [13] sur la réponse, comme décrit au paragraphe 11.2.8. La clé à utiliser dépend du mécanisme de secret partagé. Si la demande de secret partagé STUN a été utilisée, la clé DOIT être celle associée à l'attribut USERNAME présent dans la demande de lien.

Si la demande de lien contenait un attribut RESPONSE-ADDRESS, le serveur DOIT ajouter l'attribut REFLECTED-FROM à la réponse. Si la demande de lien a été authentifiée en utilisant un nom d'utilisateur obtenu d'une demande de secret partagé, l'attribut REFLECTED-FROM DOIT contenir l'adresse et port IP de source d'où venait cette demande de secret partagé. Si le nom d'utilisateur présent dans la demande n'a pas été alloué en utilisant une demande de secret partagé, l'attribut REFLECTED-FROM DOIT contenir l'adresse et port de source de l'entité qui a obtenu le nom d'utilisateur, au mieux qu'il puisse être vérifié avec le mécanisme utilisé pour allouer le nom d'utilisateur. Si le nom d'utilisateur n'était pas présent dans la demande, et si le serveur avait l'intention de traiter la demande, l'attribut REFLECTED-FROM DEVRAIT contenir l'adresse et port IP de source d'où venait la demande.

Le serveur NE DEVRAIT PAS retransmettre la réponse. La fiabilité est obtenue par le renvoi périodique de la demande par le client, chaque renvoi déclanchant une réponse su serveur.

8.2 Demandes de secret partagé

Les demandes de secret partagé sont toujours reçues sur des connexions TLS. Lorsque le serveur reçoit du client une demande d'établissement d'une connexion TLS, il DOIT continuer avec TLS,

et DEVRAIT présenter un certificat de site. On devrait utiliser la suite chiffrante TLS TLS_RSA_WITH_AES_128_CBC_SHA [4]. L'authentification du client TLS NE DOIT PAS être effectuée, car le serveur n'alloue aucune ressource aux clients, et la charge du calcul peut être une source d'attaques.

Si le serveur reçoit une demande de secret partagé, il DOIT vérifier que la demande est arrivée sur une connexion TLS. S'il n'a pas reçu la demande sur TLS, il DOIT générer une Réponse d'erreur de secret partagé, et il DOIT inclure un attribut ERROR-CODE avec un code de réponse de 433. La destination pour la réponse d'erreur dépend du transport sur lequel la demande a été reçue. Si la demande de secret partagé a été reçue sur TCP, la Réponse d'erreur de secret partagé est envoyée sur la même connexion que celle sur laquelle la demande a été reçue. Si la demande de secret partagé a été reçue sur UDP, la Réponse d'erreur de secret partagé est envoyée à l'adresse et port IP de source d'où venait la demande.

Le serveur DOIT chercher tous les attributs de la demande qui ont des valeurs inférieures ou égales à 0x7fff qu'il ne comprend pas. S'il en rencontre, le serveur DOIT générer une Réponse d'erreur de secret partagé, et il DOIT inclure un attribut ERROR-CODE avec un code de réponse de 420. Cette réponse DOIT contenir un attribut UNKNOWN-ATTRIBUTES faisant la liste des attributs qui ont une valeur inférieure ou égale à 0x7fff qui n'ont pas été compris. La Réponse d'erreur de secret partagé est envoyée sur la connexion TLS.

Toutes les Réponses d'erreur de secret partagé DOIVENT contenir le même ID de transaction que celui contenu dans la demande de secret partagé. La longueur dans l'en-tête de message DOIT contenir la longueur totale du message en octets, non compris l'en-tête. La Réponse d'erreur de secret partagé DOIT avoir un type de message de "Réponse d'erreur de secret partagé" (0x0112).

En supposant que la demande était construite de façon appropriée, le serveur crée une Réponse de secret partagé. La Réponse de secret partagé DOIT contenir le même ID de transaction que celui contenu dans la demande de secret partagé. La longueur de l'en-tête de message DOIT contenir la longueur totale du message en octets, non compris l'en-tête. La Réponse de secret partagé DOIT avoir un type de message de "Réponse de secret partagé". La Réponse de secret partagé DOIT contenir un attribut USERNAME et un attribut PASSWORD. L'attribut USERNAME sert d'indice au mot de passe, qui est contenu dans l'attribut PASSWORD. Le serveur peut utiliser tout mécanisme de son choix pour générer le nom d'utilisateur. Cependant, le nom d'utilisateur DOIT être valide pour une durée d'au moins 10 minutes. Validité signifie que le serveur peut calculer le mot de passe pour ce nom d'utilisateur. Il DOIT être un mot de passe unique pour chaque nom d'utilisateur. En d'autres termes, le serveur ne peut pas, 10 minutes plus tard, allouer un mot de passe différent au même nom d'utilisateur. Le serveur DOIT saisir un nom d'utilisateur différent pour chaque demande de secret partagé distincte. Distincte, dans ce cas, implique un ID de transaction différent. Il est RECOMMENDÉ que le serveur invalide explicitement le nom d'utilisateur après dix minutes. Il DOIT invalider le nom d'utilisateur après 30 minutes. Le PASSWORD contient le mot de passe lié à ce nom d'utilisateur. Le mot de passe DOIT avoir au moins 128 bits. La probabilité que le serveur alloue le même mot de passe à deux noms d'utilisateurs différents DOIT être extrêmement faible, et les mots de passe DOIVENT être non devinables. En d'autres termes, ils DOIVENT être une fonction cryptographique aléatoire du nom d'utilisateur.

Ces exigences peuvent toujours être satisfaites en utilisant un serveur sans états, en calculant intelligemment le USERNAME et le PASSWORD. Une approche possible est de construire le USERNAME comme :

USERNAME = <prefix,rounded-time,clientIP,hmac>

Où prefix est une chaîne de texte aléatoire (différente pour chaque demande de secret partagé), rounded-time est l'heure courante modulo 20 minutes, clientIP est l'adresse IP de source d'où vient la demande de secret partagé, et hmac est un HMAC [13] sur prefix, rounded-time, et client IP, en utilisant une clé privée du serveur.

Le mot de passe est alors calculé comme :

password = <hmac(USERNAME,autrecléprivée)>

Avec cette structure, le nom d'utilisateur lui -même, qui sera présent dans la demande de lien, contient l'adresse IP de source d'où vient la demande de secret partagé. Cela permet au serveur de satisfaire aux exigences spécifiées au paragraphe 8.1 pour la construction de l'attribut REFLECTED-FROM. Le serveur peut vérifier que le nom d'utilisateur n'a pas été falsifié, en utilisant le hmac présent dans le nom d'utilisateur.

La Réponse de secret partagé est envoyée sur la même connexion TLS que celle sur laquelle la demande a été reçue. Le serveur DEVRAIT garder la connexion ouverte, et laisser le client la fermer.

9. Comportement du client

Le comportement du client est très simple. Sa tâche est de découvrir le serveur STUN, obtenir un secret partagé, formuler la demande de lien, traiter la fiabilité de la demande, et traiter les réponses de lien.

9.1 Découverte

Généralement, le client sera configuré avec un nom de domaine du fournisseur des serveurs STUN. Ce nom de domaine se résout en adresse et port IP en utilisant les procédures SRV spécifiées dans la RFC 2782 [3].

Précisément, le nom de service est "stun". Le protocole est "udp" pour l'envoi des demandes de lien, ou "tcp" pour l'envoi des demandes de secret partagé. Les procédures de la RFC 2782 sont suivies pour déterminer le serveur à contacter. La RFC 2782 décrit les détails de la façon de trier un ensemble d'enregistrements de SRV et ensuite de les mettre à l'épreuve. Cependant, elle établit seulement que le client devrait "essayer de se connecter au (protocole, adresse, service)" sans donner aucun détail sur ce qui se passe en cas d'échec. Ces détails sont décrits ici pour STUN.

Pour les demandes STUN, l'échec survient si il y a une défaillance du transport de quelque sorte (généralement, due à des erreurs fatales d'ICMP dans UDP ou des défaillances de connexion dans TCP). L'échec survient aussi si la transaction échoue à cause d'un dépassement de temporisation. Cela survient 9,5 secondes après l'envoi de la première demande, à la fois pour la demande de secret partagé et pour la demande de lien. Voir au paragraphe 9.3 les détails sur les dépassements de temporisation de transaction pour les demandes de lien. Si un échec survient, le client DEVRAIT créer une nouvelle demande, identique à la précédente, mais avec un ID de transaction

et un attribut MESSAGE INTEGRITY différents (le HMAC changera parce que l'ID de transaction a changé). Cette demande est envoyée au prochain élément de la liste comme spécifié par la RFC 2782.

Le port par défaut pour les demandes STUN est 3478, pour TCP et UDP. Les administrateurs DEVRAIENT utiliser ce port dans leurs enregistrements SRV, mais PEUVENT en utiliser d'autres.

Si on ne trouve pas d'enregistrement SRV, le client effectue une recherche d'enregistrement A sur le nom de domaine. Le résultat sera une liste d'adresses IP, dont chacune peut être contactée au port par défaut.

Cela devrait permettre à un administrateur de pare-feu d'ouvrir le port STUN, de façon que les hôtes au sein de l'entreprise puissent accéder à de nouvelles applications. Savoir s'ils le feront ou non est une bonne question.

9.2 Obtention d'un secret partagé

Comme exposé à la Section 12, il y a plusieurs attaques possibles sur les systèmes STUN. Beaucoup d'entre elles sont prévenues à travers l'intégrité des demandes et des réponses. Pour fournir cette intégrité, STUN utilise un secret partagé entre le client et le serveur, qui sert de matériau de clé pour un HMAC utilisé à la fois dans la demande de lien et la réponse de lien. STUN permet que le secret partagé soit obtenu de n'importe quelle façon (par exemple, Kerberos [14]). Cependant, il DOIT avoir au moins 128 bits aléatoires. Afin d'assurer l'interopérabilité, la présente spécification décrit un mécanisme fondé sur TLS. Ce mécanisme, décrit dans la présente section, DOIT être mis en œuvre par les clients et les serveurs.

D'abord, le client détermine l'adresse et port IP auxquels il va ouvrir une connexion TCP. Cela est fait en utilisant les procédures de découverte du paragraphe 9.1. Le client ouvre la connexion à cette adresse et port, et commence immédiatement la négociation TLS [2]. Le client DOIT vérifier l'identité du serveur. Pour ce faire, il suit les procédures d'identification définies au paragraphe 3.1 de la RFC 2818 [5]. Ces procédures supposent que le client déréférence un URI. Pour les besoins de l'utilisation avec la présente spécification, le client traite le nom de domaine ou adresse IP utilisée au paragraphe 9.1 comme la portion hôte de l'URI qui a été déréférencé.

Une fois que la connexion est ouverte, le client envoie une demande de secret partagé. Cette demande n'a pas d'attributs, seulement l'en-tête. L'identifiant de transaction dans l'en-tête DOIT satisfaire aux exigences formulées pour l'identifiant de transaction dans une demande de lien, décrites au paragraphe 9.3 ci-dessous. Le serveur génère une réponse, qui peut être une Réponse de secret partagé ou une Réponse d'erreur de secret partagé.

Si la réponse est une Réponse d'erreur de secret partagé, le client vérifie le code de réponse dans l'attribut ERROR-CODE. L'interprétation de ces codes de réponse est identique au traitement du paragraphe 9.4 pour la réponse d'erreur de lien.

Si un client reçoit une Réponse de secret partagé avec un attribut dont le type est supérieur à 0x7fff, l'attribut DOIT être ignoré. Si le client reçoit une Réponse de secret partagé avec un attribut dont le type est inférieur ou égal à 0x7fff, la réponse est ignorée.

Si la réponse est une Réponse de secret partagé, elle contiendra un nom d'utilisateur et mot de passe à courte durée de vie, codés respectivement dans les attributs USERNAME et PASSWORD.

Le client PEUT générer plusieurs demandes de secret partagé sur la connexion, et il PEUT faire ainsi avant de recevoir des Réponses de secret partagé à des demandes de secret partagé précédentes. Le client DEVRAIT fermer la connexion dès qu'il a fini d'obtenir les noms d'utilisateur et les mots de passe.

Le paragraphe 9.3 décrit comment ces mots de passe sont utilisés pour fournir la protection de l'intégrité sur les demandes de lien et le paragraphe 8.1 décrit comment ils sont utilisés dans les réponses de lien.

9.3 Formulation de la demande de lien

Une demande de lien formulée par le client suit les règles de syntaxe définies au paragraphe 11. Si deux demandes ne sont pas identiques au bit près et ne sont pas envoyées au même serveur à partir de la même adresse et port IP, elles DOIVENT porter des identifiants de transaction différents. L'identifiant de transaction DOIT être distribué uniformément et aléatoirement entre 0 et 2**128 - 1. Cette large gamme est nécessaire parce que l'identifiant de transaction sert de forme aléatoire, qui sert à prévenir le rejeu de réponses signées antérieures à partir du serveur. Le type de message de la demande DOIT être "Binding Request".

L'attribut RESPONSE-ADDRESS est facultatif dans la demande de lien. Il est utilisé si le client souhaite que la réponse soit envoyée à une adresse et port IP différents de ceux d'où la demande a été envoyée. Ceci est utile pour déterminer si le client est derrière un pare-feu, et pour les applications qui ont des composants de contrôle et de données séparés. Voir pour plus de détails au paragraphe 10.3. L'attribut CHANGE-REQUEST est aussi facultatif. Sa présence dépend de ce que l'application essaye de faire. Voir des exemples d'utilisation à la Section 10.

Le client DEVRAIT ajouter un attribut MESSAGE-INTEGRITY et USERNAME à la demande de lien. Cet attribut MESSAGE-INTEGRITY contient un HMAC [13]. La valeur du nom d'utilisateur, et la clé à utiliser dans l'attribut MESSAGE-INTEGRITY dépendent du mécanisme de secret partagé. Si la demande de secret partagé STUN a été utilisée, le USERNAME DOIT être un nom d'utilisateur valide obtenu d'une Réponse de secret partagé dans les neuf dernières minutes. Le secret partagé pour le HMAC est la valeur de l'attribut PASSWORD obtenu de la même Réponse de secret partagé.

Une fois formulée, le client envoie la demande de lien. La fiabilité se réalise par les retransmissions du client. Les clients DEVRAIENT retransmettre la demande en commençant par un intervalle de 100 ms, doublé à chaque retransmission jusqu'à ce que l'intervalle atteigne 1,6 s. Les retransmissions continuent à un intervalle de 1,6 s jusqu'à ce qu'une réponse soit reçue, ou qu'un total de 9 demandes aient été envoyées. Si aucune réponse n'est reçue dans les 1,6 secondes après l'envoi de la dernière demande, le client DEVRAIT considérer que la transaction a échoué. En d'autres termes, les demandes devraient être envoyées au temps 0 ms, 100 ms, 300 ms, 700 ms, 1 500 ms, 3 100 ms, 4 700 ms, 6 300 ms, et 7 900 ms. A 9 500 ms, le client considère que la transaction a échoué si aucune réponse n'est reçue.

9.4 Traitement des réponses de lien

La réponse peut être une Réponse de lien ou une Réponse d'erreur de lien. Les Réponses d'erreur de lien sont toujours reçues à l'adresse et port de source d'où la demande a été envoyée. Une Réponse de lien sera reçue à l'adresse et port placés dans l'attribut RESPONSE-ADDRESS de la demande. Si aucune n'était présente, la réponse de liens sera reçue à l'adresse et port de source d'où la demande a été envoyée.

Si la réponse est une Réponse d'erreur de lien, le client vérifie le code de réponse de l'attribut ERROR-CODE de la réponse. Pour un code de réponse 400, le client DEVRAIT afficher le motif du rejet à l'usager. Pour un code de réponse 420, le client DEVRAIT réessayer la demande, en omettant cette fois tous les attributs dont la liste est donnée dans l'attribut UNKNOWN-ATTRIBUTES de la réponse. Pour un code de réponse 430, le client DEVRAIT obtenir un nouveau secret partagé, et réessayer la demande de lien avec une nouvelle transaction. Pour les codes de réponse 401 et 432, si le client a omis les attributs USERNAME ou MESSAGE-INTEGRITY comme indiqué par l'erreur, il DEVRAIT essayer à nouveau avec ces attributs. Pour un code de réponse 431, le client DEVRAIT alerter l'utilisateur, et PEUT essayer la demande à nouveau après avoir obtenu un nouveau nom d'utilisateur et mot de passe. Pour un code de réponse 500, le client PEUT attendre plusieurs secondes et réessayer alors la demande. Pour un code de réponse 600, le client NE DOIT PAS réessayer la demande, et DEVRAIT afficher le texte de la cause à l'usager. Les attributs inconnus entre 400 et 499 sont traités comme un 400, les attributs inconnus entre 500 et 599 sont traités comme un 500, et les attributs inconnus entre 600 et 699 sont traités comme un 600. Toute réponse entre 100 et 399 DOIT avoir pour résultat la cessation des retransmissions de demandes, mais est mise à l'écart pour le reste.

Si un client reçoit une réponse avec un attribut dont le type est supérieur à 0x7fff, l'attribut DOIT être ignoré. Si le client reçoit une réponse avec un attribut dont le type est inférieur ou égal à 0x7fff, les retransmissions de demandes DOIVENT cesser, mais la réponse en elle-même est ignorée.

Si la réponse est une Réponse de lien, le client DEVRAIT vérifier la réponse à la recherche d'un attribut MESSAGE-INTEGRITY. S'il n'en est pas présent, et que le client a placé un attribut MESSAGE-INTEGRITY dans la demande, il DOIT écarter la réponse. S'il en est un présent, le client calcule le HMAC sur la réponse comme décrit au paragraphe 11.2.8. La clé à utiliser dépend du mécanisme de secret partagé. Si la demande de secret partagé STUN a été utilisée, la clé DOIT être la même que celle utilisée pour calculer l'attribut MESSAGE-INTEGRITY dans la demande. Si le HMAC calculé diffère de celui de la réponse, le client DOIT écarter la réponse, et DEVRAIT alerter l'utilisateur d'une attaque possible. Si le HMAC calculé correspond à celui de la réponse, le processus continue.

La réception d'une réponse (Réponse d'erreur de lien ou Réponse de lien) à une Demande de lien terminera les retransmissions de cette demande. Cependant, les clients DOIVENT continuer à guetter pendant 10 secondes, après la première réponse, les réponses à une Demande de lien. S'il reçoit d'autres réponses dans cet intervalle avec des types de message différents (Réponses de lien et Réponses d'erreur de lien, par exemple) ou des MAPPED-ADDRESS différentes, c'est l'indication d'une attaque possible. Le client NE DOIT PAS utiliser les MAPPED-ADDRESS, d'aucune des réponses reçues (ni la première ni les suivantes), et DEVRAIT alerter l'usager.

De plus, si un client reçoit plus de deux fois plus de Réponses de lien que le nombre de Demandes de lien qu'il a envoyé, il NE DOIT PAS utiliser la MAPPED-ADDRESS, d'aucune de ces réponses, et DEVRAIT alerter l'usager d'une attaque potentielle.

Si la réponse de lien est authentifiée, et si la MAPPED-ADDRESS n'a pas été écartée à cause d'une attaque potentielle, le CLIENT PEUT utiliser les attributs MAPPED-ADDRESS et SOURCE-ADDRESS.

10. Cas d'utilisation

Les règles des Sections 8 et 9 décrivent exactement la façon dont interagissent un client et un serveur pour envoyer des demandes et recevoir les réponses. Cependant, elles n'indiquent pas l'utilisation du protocole STUN pour accomplir les tâches utiles. Cela est à la discrétion du client. Ici, nous allons donner quelques scénarios utiles pour l'application de STUN.

10.1 Processus de découverte

Dans ce scénario, un usager fait tourner une application multimédia qui a besoin de déterminer lequel des scénarios suivants s'applique à lui :

- sur l'Internet ouvert
- un pare-feu bloque UDP
- un pare-feu permet UDP en sortie, et les réponses doivent revenir sur la source de la demande (comme un NAT symétrique, mais sans traduction. On appelle cela un pare-feu UDP symétrique)
- un NAT en plein cône
- un NAT symétrique
- un NAT en cône restreint ou en cône à restriction de port

L'organigramme de la Figure 2 permet de déterminer lequel des six scénarios s'applique. Le diagramme ne se réfère qu'aux séquences de Demande de lien ; les Demandes de secret partagé seront, bien sûr, nécessaires pour authentifier chaque Demande de lien utilisée dans la séquence.

Le flux utilise trois essais. Dans l'essai I, le client envoie une Demande de lien STUN à un serveur, sans aucun fanion établi dans l'attribut CHANGE-REQUEST, et sans l'attribut RESPONSE-ADDRESS. Cela conduit le serveur à renvoyer la réponse à l'adresse et port d'où est venue la demande. Dans l'essai II, le client envoie une Demande de lien avec les deux fanions "change IP" et "change port" établis dans l'attribut CHANGE-REQUEST. Dans l'essai III, le client envoie une Demande de lien avec seulement le fanion "change port" établi.

Le client commence par lancer l'essai I. Si l'essai ne donne pas de réponse, le client sait tout de suite qu'il n'y a pas de capacité de connectivité UDP. Si l'essai produit une réponse, le client examine l'attribut MAPPED-ADDRESS. Si cette adresse et port sont les mêmes que l'adresse et port IP local de la prise utilisée pour envoyer la demande, le client sait qu'il n'est pas derrière un NAT. Il exécute l'essai II.

S'il reçoit une réponse, le client sait qu'il a un accès ouvert à l'Internet (ou, au moins, qu'il est derrière un pare-feu qui se comporte comme un NAT en plein cône, mais sans la traduction). S'il ne reçoit pas de réponse, le client sait qu'il est derrière un pare-feu UDP symétrique.

Dans le cas où l'adresse et port IP de la prise ne correspondent pas à l'attribut MAPPED-ADDRESS de la réponse à l'essai I, le client sait qu'il est derrière un NAT. Il effectue l'essai II. S'il reçoit une réponse, le client sait qu'il est derrière un NAT en plein cône. S'il ne reçoit aucune réponse, il effectue à nouveau l'essai I, mais cette fois, il fait de même avec l'adresse et le port de l'attribut CHANGED-ADDRESS provenant de la réponse à l'essai I. Si l'adresse et port IP retournés dans l'attribut MAPPED-ADDRESS ne sont pas les mêmes que ceux provenant du premier essai I, le client sait qu'il est derrière un NAT symétrique. Si l'adresse et le port sont les mêmes, le client est derrière un NAT restreint ou à restriction de port. Pour déterminer derrière lequel il est, le client lance l'essai III. S'il reçoit une réponse, il est derrière un NAT restreint, et s'il ne reçoit pas de réponse, il est derrière un NAT à restriction de port.

Cette procédure donne des informations substantielles sur les conditions de fonctionnement de l'application du client. Dans le cas où plusieurs NAT se trouvent entre le client et l'Internet, le type découvert sera le type du NAT le plus restrictif entre le client et l'Internet. Les types de NAT, par ordre de restriction, du plus restrictif ou moins restrictif sont symétrique, cône à restriction de port, cône restreint, et plein cône.

Normalement, un client va refaire périodiquement ce processus de découverte pour détecter les changements, ou chercher des résultats incohérent. Il est important de noter que lorsque le processus de découverte est refait, il ne devrait généralement pas l'être à partir du même port et adresse local qui a été utilisé dans les processus de découverte précédents. Si le même port et adresse local est réutilisé, les liens provenant de l'essai précédent peuvent être toujours existants, et cela invalidera les résultats de l'essai. On résout ce problème en utilisant une adresse et port local différents pour les essais suivants. Une solution de remplacement est d'attendre suffisamment longtemps pour être sûr que les vieux liens sont arrivés à expiration (une demi-heure devrait être plus que suffisant).

10.2 Découverte de la durée de vie du lien

On peut aussi se servir de STUN pour découvrir la durée de vie des liens créés par le NAT. Dans de nombreux cas, le client aura besoin de rafraîchir le lien, à l'aide d'une nouvelle demande STUN, ou par un paquet d'application, afin que l'application continue à utiliser le lien. En découvrant la durée de vie du lien, le client peut déterminer la fréquence à laquelle il doit être rafraîchi.

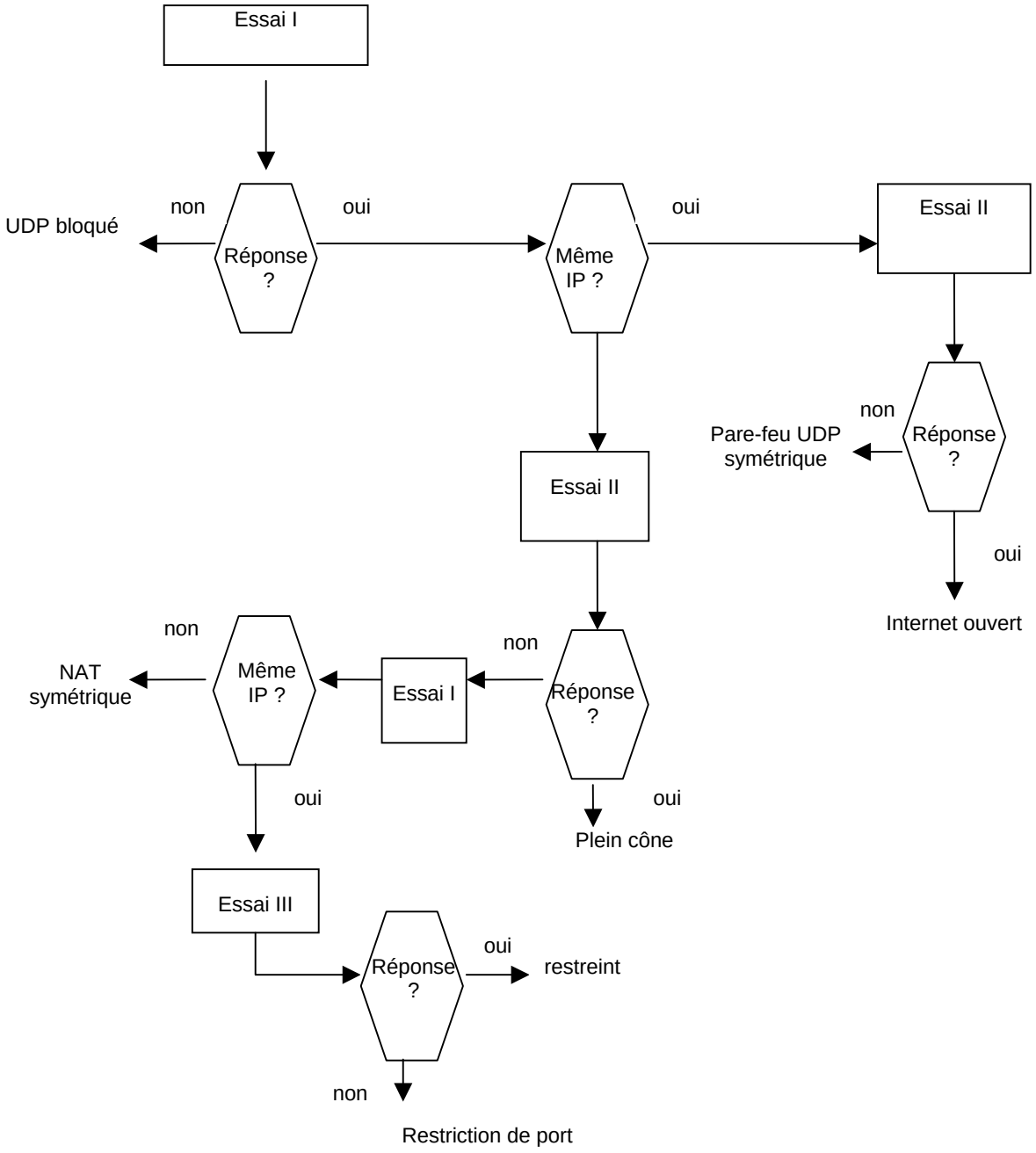


Figure 2 : Flux du processus de découverte du type

Pour déterminer la durée de vie d'un lien, le client envoie d'abord une Demande de lien au serveur à partir d'une prise déterminée, X. Cela crée un lien dans le NAT. La réponse du serveur contient un attribut MAPPED-ADDRESS, qui fournit l'adresse et port publics sur le NAT. Appelons cela respectivement Pa et Pp. Le client lance alors une temporisation d'une valeur de T secondes. Lorsque ce temporisateur arrive à expiration, le client envoie une autre Demande de lien au serveur, en utilisant les mêmes adresse et port de destination, mais d'une prise différente, Y. Cette demande contient un attribut d'adresse RESPONSE-ADDRESS, réglé à (Pa,Pp). Cela va créer un nouveau lien sur le NAT, et causer l'envoi par le serveur STUN d'une Réponse de lien qui devrait correspondre au vieux lien, s'il existe toujours. Si le client reçoit la réponse de lien sur la prise X, il sait que le lien n'est pas arrivé à expiration. Si le client reçoit la réponse de lien sur la prise Y (ce

qui est possible si le vieux lien est arrivé à expiration, et que le NAT a alloué les mêmes adresse et port publics au nouveau lien), ou ne reçoit pas de réponse du tout, il sait que le lien a expiré.

Le client peut trouver la valeur de la durée de vie du lien en faisant une recherche binaire à travers T, arrivant finalement à la valeur où la réponse n'est pas reçue pour tout temps supérieur à T, mais est reçue pour tout temps inférieur à T.

Ce processus de découverte prend un certain temps, et c'est quelque chose qui va normalement se dérouler en arrière plan sur un appareil lorsqu'il s'amorce.

Il est possible que le client obtienne des résultats incohérents à chaque fois que ce processus se déroule. Par exemple, si le NAT devrait se réamorcer, ou être remis à zéro pour une raison quelconque, le processus peut découvrir une durée de vie qui est plus courte que celle en cours. Pour cette raison, il est conseillé que les mises en œuvre effectuent l'essai un grand nombre de fois, et elles doivent s'attendre à obtenir des résultats incohérents.

10.3 Acquisition de lien

Considérons encore une fois le cas d'un téléphone VoIP. Il a utilisé le processus de découverte ci-dessus au démarrage, pour découvrir son environnement. Maintenant, il veut effectuer un appel. Au titre du processus de découverte, il a déterminé qu'il est derrière un NAT en plein cône.

Considérons de plus que ce téléphone comporte deux composants séparés logiquement – un composant de contrôle qui traite la signalisation, et un composant de support qui traite l'audio, la vidéo, et RTP [12]. Tous deux sont derrière le même NAT. A cause de cette séparation du contrôle et du support, on souhaite minimiser les communications nécessaires entre eux. En fait, ils peuvent même ne pas fonctionner sur le même hôte. Pour passer un appel vocal, le téléphone a besoin d'obtenir une adresse et un port IP qu'il puisse mettre dans le message d'établissement d'appel comme la destination de réception audio.

Pour obtenir une adresse, le composant de contrôle envoie une demande de secret partagé au serveur, obtient un secret partagé, et envoie alors une Demande de lien au serveur. Aucun attribut CHANGE-REQUEST n'est présent dans la demande de lien, pas plus que l'attribut RESPONSE-ADDRESS. La Réponse de lien contient une adresse transposée. Le composant de contrôle formule alors une seconde Demande de lien. Cette demande contient une RESPONSE-ADDRESS, qui est réglée à l'adresse transposée apprise de la Réponse de lien précédente. Cette Demande de lien est passée au composant de support, avec l'adresse et port IP du serveur STUN. Le composant support envoie alors la Demande de lien. La demande va au serveur STUN, qui renvoie la réponse de lien au composant de contrôle. Le composant de contrôle la reçoit, et il a maintenant appris une adresse et un port IP qui seront réacheminés sur le composant support qui a envoyé la demande.

Le client sera à même de recevoir des média de partout sur cette adresse transposée.

Dans le cas de suppression de silence, il peut y avoir des périodes pendant lesquelles le client ne reçoit pas de média. Dans ce cas, les liens UDP peuvent arriver en fin de temporisation (les liens UDP dans les NAT sont normalement courts ; 30 secondes est courant). Pour s'accommoder de cela, l'application peut retransmettre périodiquement la demande pour rafraîchir la liaison.

Il est possible que les deux participants à la session multimédia soient derrière le même NAT. Dans ce cas, ils répéteront tous deux la procédure ci-dessus, et tous deux obtiendront des liens d'adresse publics. Quand l'un envoie des média à l'autre, le média est acheminé au NAT, et aussitôt retourné pour revenir dans l'entreprise, où il est traduit à l'adresse privée du récepteur. Ceci n'est pas particulièrement efficace, et malheureusement, ne fonctionne pas dans de nombreux NAT commerciaux. Dans de tels cas, le clients peut être obligé de réessayer en utilisant les adresses privées.

11. Détails du protocole

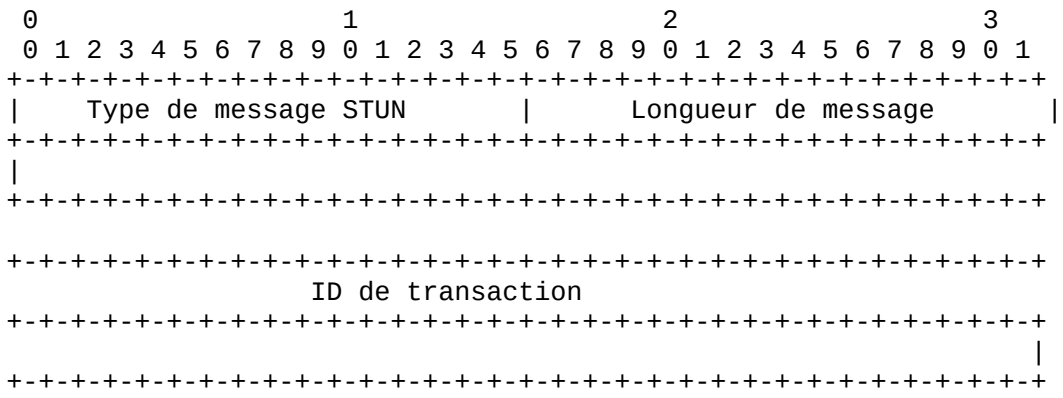
La présente section présente le codage détaillé d'un message STUN.

STUN est un protocole de demande-réponse. Les clients envoient une demande, et le serveur envoie une réponse. Il y a deux demandes, Demande de lien, et Demande de secret partagé. La réponse à une Demande de lien peut être la Réponse de lien ou la Réponse d'erreur de lien. La réponse à une Demande de secret partagé peut être une Réponse de secret partagé ou une Réponse d'erreur de secret partagé.

Les messages STUN sont codés à l'aide de champs binaires. Tous les champs entiers sont portés dans l'ordre d'octets du réseau, c'est-à-dire, avec l'octet de plus fort poids d'abord. Cet ordre des octets est appelé communément gros boutien. L'ordre de transmission est décrit en détail dans l'Appendice B de la RFC 791 [6]. Sauf notation contraire, les constantes numériques sont décimales (base 10).

11.1 En-tête de message

Tous les messages STUN comportent un en-tête de 20 octets :



Les types de message peuvent prendre les valeurs suivantes :

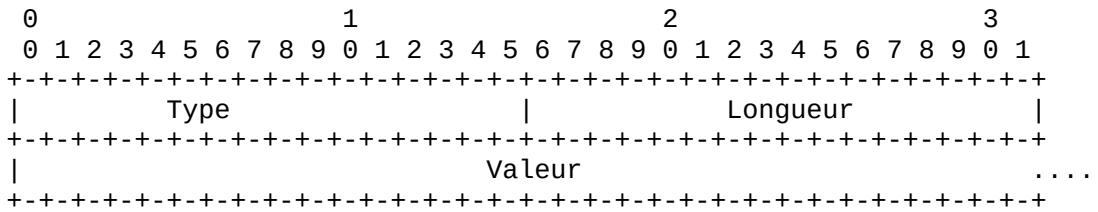
0x0001 : Demande de lien
0x0101 : Réponse de lien
0x0111 : Réponse d'erreur de lien
0x0002 : Demande de secret partagé
0x0102 : Réponse de secret partagé
0x0112 : Réponse d'erreur de secret partagé

La longueur du message est le compte, en octets, de la taille du message, non inclus l'en-tête de 20 octets.

L'ID de transaction est un identifiant de 128 bits. Il sert aussi d'élément pour rendre aléatoire la demande et la réponse. Toutes les réponses portent le même identifiant que la demande à laquelle elles correspondent.

11.2 Attributs du message

Après l'en-tête se trouvent 0 ou plusieurs attributs. Chaque attribut est codé en TLV, avec un type de 16 bits, 16 bits de longueur, et une valeur variable :



Les types suivants sont définis :

0x0001: MAPPED-ADDRESS *(adresse transposée)*
0x0002: RESPONSE-ADDRESS *(adresse de réponse)*
0x0003: CHANGE-REQUEST (*demande de modification)*
0x0004: SOURCE-ADDRESS *(adresse de source)*
0x0005: CHANGED-ADDRESS *(adresse modifiée)*
0x0006: USERNAME *(nom d'utilisateur)*
0x0007: PASSWORD *(mot de passe)*
0x0008: MESSAGE-INTEGRITY *(intégrité du message)*
0x0009: ERROR-CODE *(code d'erreur)*
0x000a: UNKNOWN-ATTRIBUTES *(attributs inconnus)*
0x000b: REFLECTED-FROM *(réfléchi de)*

Pour permettre à des révisions ultérieures de la présente spécification d'ajouter de nouveaux attributs si nécessaire, l'espace d'attribut est divisé en espaces facultatifs et espaces obligatoires. Les attributs avec des valeurs supérieures à 0x7fff sont facultatives, ce qui signifie que le message peut être traité par le client ou le serveur même si l'attribut n'est pas compris. Les attributs qui ont une valeur inférieure ou égale à 0x7fff doivent obligatoirement être compris, ce qui signifie que le client ou le serveur ne peut traiter le message que si l'attribut est compris.

L'attribut MESSAGE-INTEGRITY DOIT être le dernier attribut au sein d'un message. Tout attribut qui est connu, mais non supposé être présent dans un message (MAPPED-ADDRESS dans une demande, par exemple) DOIT être ignoré.

Le Tableau 2 indique quels attributs sont présents dans chaque message. Un O indique que l'inclusion de l'attribut dans le message est obligatoire, F signifie qu'il est facultatif, C qu'il est conditionnel sur la base d'un autre aspect du message, et N/A signifie que l'attribut n'est pas applicable à ce type de message.

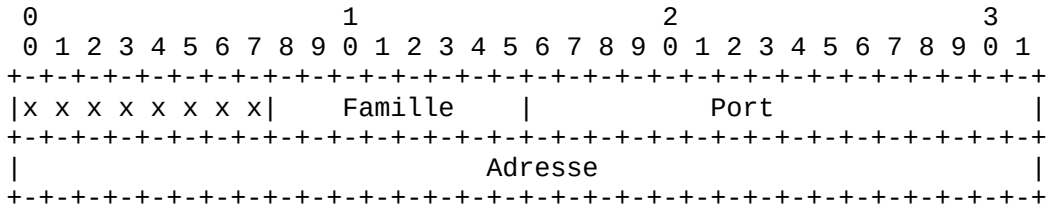
Attribut	Demande de lien	Réponse de lien	Réponse d'erreur de lien	Demande de secret partagé	Réponse de secret partagé	Réponse d'erreur de secret partagé
MAPPED-ADDRESS	N/A	O	N/A	N/A	N/A	N/A
RESPONSE-ADDRESS	F	N/A	N/A	N/A	N/A	N/A
CHANGE-REQUEST	F	N/A	N/A	N/A	/A	N/A
SOURCE-ADDRESS	N/A	O	N/A	N/A	N/A	N/A
CHANGED-ADDRESS	N/A	O	N/A	N/A	N/A	N/A
USERNAME	F	N/A	N/A	N/A	O	N/A
PASSWORD	N/A	N/A	N/A	N/A	O	N/A
MESSAGE-INTEGRITY	F	F	N/A	N/A	N/A	N/A
ERROR-CODE	N/A	N/A	O	N/A	N/A	O
UNKNOWN-ATTRIBUTES	N/A	N/A	C	N/A	N/A	C
REFLECTED-FROM	N/A	C	N/A	N/A	N/A	N/A

Table 2 : Résumé des attributs

La longueur se réfère à la longueur de l'élément valeur, exprimée comme un nombre entier arithmétique d'octets.

11.2.1 MAPPED-ADDRESS

L'attribut MAPPED-ADDRESS indique l'adresse et port IP transposés. Il consiste en une famille d'adresses de huit bits, et un port de seize bits, suivis par une valeur de longueur fixe représentant l'adresse IP.



Le port est une représentation en octets dans l'ordre du réseau du port transposé. La famille d'adresse est toujours 0x01, correspondant à IPv4. Les huit premiers bits de la MAPPED-ADDRESS sont ignorés, pour les besoins du verrouillage des paramètres sur les frontières naturelles. L'adresse IPv4 est de 32 bits.

11.2.2 RESPONSE-ADDRESS

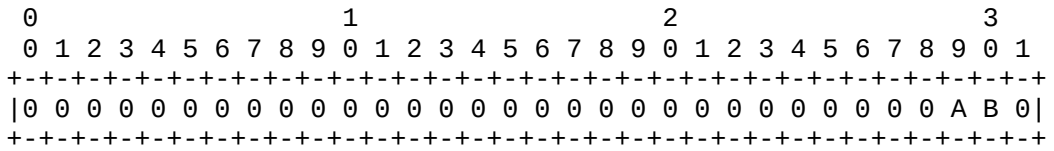
L'attribut RESPONSE-ADDRESS indique où devrait être envoyée la réponse à une Demande de lien. Sa syntaxe est identique à celle de MAPPED-ADDRESS.

11.2.3 CHANGED-ADDRESS

L'attribut CHANGED-ADDRESS indique l'adresse et port IP où les réponses seraient envoyées si les fanions "change IP" et "change port" avaient été mis dans l'attribut CHANGE-REQUEST de la demande de lien. L'attribut est toujours présent dans une Réponse de lien, indépendamment de la valeur des fanions. Sa syntaxe est identique à celle de MAPPED-ADDRESS.

11.2.4 CHANGE-REQUEST

L'attribut CHANGE-REQUEST est utilisé par le client pour demander que le serveur utilise une adresse et/ou port différents lors de l'envoi de la réponse. L'attribut a une longueur de 32 bits, bien que seuls deux bits (A et B) soient utilisés :



La signification des fanions est:

A : C'est le fanion "change IP". S'il est vrai, il demande au serveur d'envoyer la réponse de lien avec une adresse IP différente de celle sur laquelle la demande de lien a été reçue.

B : C'est le fanion "change port". S'il est vrai, il demande au serveur d'envoyer la réponse de lien avec un port différent de celui sur lequel la demande de lien a été reçue.

11.2.5 SOURCE-ADDRESS

L'attribut SOURCE-ADDRESS est présent dans les Réponses de lien. Il indique l'adresse et port IP de source d'où le serveur envoie la réponse. Sa syntaxe est identique à celle de MAPPED-ADDRESS.

11.2.6 USERNAME

L'attribut USERNAME sert à l'intégrité du message. Il sert comme moyen d'identification du secret partagé utilisé dans la vérification d'intégrité du message. Le USERNAME est toujours présent dans une Réponse de secret partagé, avec le PASSWORD. Il est facultativement présent dans une Demande de lien lorsque l'intégrité de message est utilisée.

La valeur de USERNAME est une valeur opaque de longueur variable. Sa longueur DOIT être un multiple de 4 (mesurée en octets) afin de garantir le verrouillage des attributs sur les limites de mots.

11.2.7 PASSWORD

L'attribut PASSWORD sert dans les Réponses de secret partagé. Il est toujours présent dans une Réponse de secret partagé, avec USERNAME.

La valeur de PASSWORD est de longueur variable et elle est utilisée comme un secret partagé. Sa longueur DOIT être un multiple de 4 (mesuré en octets) afin de garantir le verrouillage des attributs sur les limites de mots.

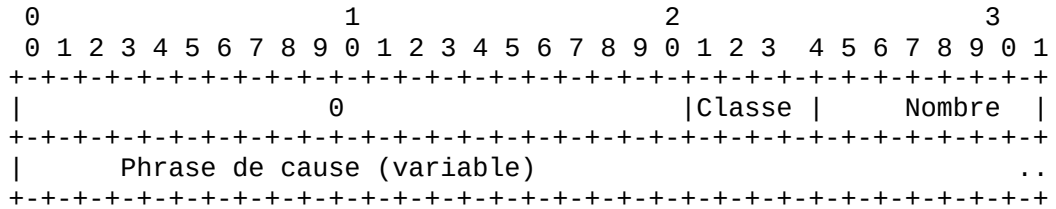
11.2.8 MESSAGE-INTEGRITY

L'attribut MESSAGE-INTEGRITY contient un HMAC-SHA1 [13] du message STUN. Il peut être présent dans les Demandes de lien ou les Réponses de lien. Comme il utilise le hachage SHA1, le HMAC sera de 20 octets. Le texte utilisé comme entrée au HMAC est le message STUN, y compris l'en-tête, jusqu'à et y inclus l'attribut qui précède l'attribut MESSAGE-INTEGRITY. Ce texte est alors bourré avec des zéros de sorte qu'il soit un multiple de 64 octets. Il en résulte que l'attribut MESSAGE-INTEGRITY DOIT être le dernier attribut de tout message STUN. La clé utilisée comme entrée au HMAC dépend du contexte.

11.2.9 ERROR-CODE

L'attribut ERROR-CODE est présent dans la Réponse d'erreur de lien et dans la Réponse d'erreur de secret partagé. C'est une valeur numérique dans la gamme de 100 à 699, plus une phrase textuelle de cause codée en UTF-8, et qui est cohérente dans ses allocations de code et sa sémantique avec SIP [10] et HTTP [15]. La phrase de cause est destinée à l'utilisateur, et peut être tout ce qui est approprié au code de réponse. La longueur des phrases de cause DOIT être un multiple de 4 (mesuré en octets). Ceci peut être accompli par l'ajout d'espaces à la fin du texte, si nécessaire. Les phrases de cause recommandées pour les codes de réponse définis sont présentés ci-dessous.

Pour faciliter le traitement, la classe du code d'erreur (le chiffre des centaines) est codée séparément du reste du code.



La classe représente le chiffre des centaines du code de réponse. La valeur DOIT être entre 1 et 6. Le nombre représente le code de réponse modulo 100, et sa valeur DOIT être entre 0 et 99.

Les codes de réponse suivants, avec leurs phrases de cause recommandées (entre parenthèses) sont définis comme suit pour l'instant :

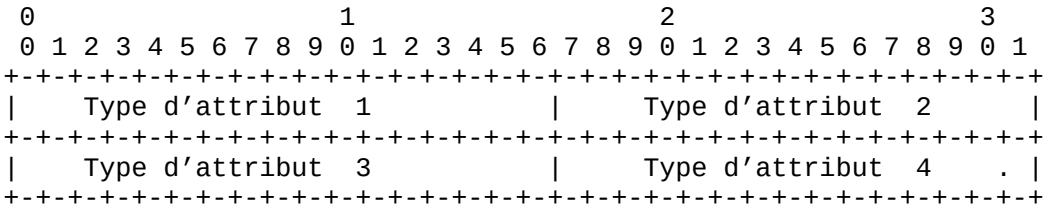
400 (mauvaise demande) : La demande est mal formée. Le client ne devrait pas réessayer la demande sans modification par rapport à l'essai précédent.
401 (non autorisée) : La Demande de lien ne contient pas d'attribut MESSAGE- INTEGRITY.

420 (attribut inconnu) : Le serveur n'a pas compris un attribut obligatoire de la demande.
430 (accréditation périmée) : La Réponse de lien contient un attribut MESSAGE- INTEGRITY, mais il utilise un secret partagé arrivé à expiration. Le client devrait obtenir un nouveau secret partagé et réessayer.
431 (échec de vérification d'intégrité) : La Demande de lien contient un attribut MESSAGE-INTEGRITY, mais la vérification HMAC a échoué. Cela pourrait signaler une attaque potentielle, ou une erreur de mise en œuvre du client.
432 (nom d'utilisateur manquant) : La Demande de lien contient un attribut MESSAGE-INTEGRITY, mais pas d'attribut USERNAME. Tous deux DOIVENT être présents pour les vérifications d'intégrité.
433 (utiliser TLS) : La demande de secret partagé doit être envoyée sur TLS, mais n'a pas été reçue sur TLS.
500 (erreur de serveur) : Le serveur a subi une erreur temporaire. Le client devrait réessayer.
600 (défaillance globale) : Le serveur refuse de remplir la demande. Le client ne devrait pas réessayer.

11.2.10 UNKNOWN-ATTRIBUTES

L'attribut UNKNOWN-ATTRIBUTES n'est présent que dans une Réponse d'erreur de lien ou une Réponse d'erreur de secret partagé lorsque le code de réponse dans l'attribut ERROR-CODE est 420.

L'attribut contient une liste de valeurs à 16 bits, chacune d'elles représentant un type d'attribut qui n'a pas été compris par le serveur. Si le nombre d'attributs inconnus est un nombre impair, un des attributs DOIT être répété dans la liste, pour que la longueur totale de la liste soit un multiple de 4 octets.



11.2.11 REFLECTED-FROM

L'attribut REFLECTED-FROM n'est présent que dans les Réponses de lien, lorsque la demande de lien contenait un attribut RESPONSE-ADDRESS. L'attribut contient l'identité (en termes d'adresse IP) de la source d'où provient la demande. Son objet est de procurer la traçabilité, de sorte qu'un serveur STUN ne puisse être utilisé comme réflecteur d'attaques de déni de service.

Sa syntaxe est identique à celle de l'attribut MAPPED-ADDRESS.

12. Considérations sur la sécurité

12.1 Attaques contre STUN

D'une façon générale, on peut classer les attaques contre STUN en attaques de déni de service et en attaques d'espionnage. Les attaques de déni de service peuvent être lancées contre un serveur STUN lui-même, ou contre d'autres éléments utilisant le protocole STUN.

Les serveurs STUN créent un état à travers le mécanisme de demande de secret partagé. Pour empêcher d'être submergé par le trafic, un serveur STUN DEVRAIT limiter le nombre de connexions TLS simultanées qu'il va tenir en laissant tomber une connexion existante lorsqu'une nouvelle demande de connexion arrive (sur la base de la politique de la moins récemment utilisée (LRU, *Least Recently Used*), par exemple). De même, il DEVRAIT limiter le nombre de secrets partagés qu'il va stocker, dans le cas où le serveur stocke les secrets partagés.

Les attaques les plus intéressantes sont celles dans lesquelles le serveur STUN et le client sont utilisés pour lancer des attaques en DoS *(déni de service)* contre d'autres entités, y compris le client lui-même.

De nombreuses attaques exigent que l'attaquant génère une réponse à une demande STUN légitime, afin de fournir au client une fausse MAPPED-ADDRESS. L'attaque qui peut être lancée en utilisant une telle technique inclut :

12.1.1 Attaque I : DDOS contre une cible

Dans ce cas, l'attaquant fournit à un grand nombre de clients la même fausse MAPPED-ADDRESS qui renvoie à la cible visée. Cela va amener tous les clients STUN à penser que leurs adresses sont égales à celle de la cible. Les clients donnent alors cette adresse afin de recevoir du trafic dessus (par exemple, dans des messages SIP ou H.323). Cependant, tout ce trafic se concentre sur la cible visée. L'attaque peut subir une amplification substantielle, spécialement quand elle est utilisée avec des clients qui se servent de STUN pour activer des applications multimédia.

12.1.2 Attaque II : Réduire un client au silence

Dans cette attaque, l'agresseur cherche à empêcher un client d'accéder aux services activés par STUN (par exemple, un client utilisant STUN pour activer du trafic multimédia fondé sur SIP). Pour faire cela, l'agresseur fournit une fausse MAPPED-ADDRESS au client. La MAPPED-ADDRESS qu'il fournit est une adresse IP qui ne mène nulle part. Il en résulte que le client ne recevra aucun des paquets qu'il s'attend à recevoir lorsqu'il passe sur la MAPPED-ADDRESS.

Cette exploitation n'est pas très intéressante pour l'agresseur. Elle n'impacte qu'un seul client, ce qui est rarement la cible souhaitée. De plus, tout agresseur qui peut monter ce type d'attaque peut aussi dénier le service au client par d'autres moyens, tels qu'empêcher le client de recevoir de réponse du serveur STUN, ou même d'un serveur DHCP.

12.1.3 Attaque III : Emprunt de l'identité d'un client

Cette attaque est similaire à l'attaque II. Cependant, la fausse MAPPED-ADDRESS donne l'adresse de l'agresseur lui-même. Cela lui permet de recevoir du trafic qui était destiné au client.

12.1.4 Attaque IV : Espionnage

Dans cette attaque, l'agresseur force le client à utiliser une MAPPED- ADDRESS qui achemine sur lui-même. Il transmet alors tout paquet reçu au client. Cette attaque permet à l'agresseur d'observer tous les paquets envoyés au client. Cependant, afin de lancer l'attaque, l'agresseur DOIT déjà avoir eu la possibilité d'observer les paquets allant du client au serveur STUN. Dans la plupart des cas (comme lorsque l'attaque est lancée à partir d'un réseau d'accès), cela signifie que l'agresseur pouvait déjà observer les paquets envoyés au client. Cette attaque n'est utile, par conséquent, que pour observer du trafic sur le chemin du client au serveur STUN, mais généralement pas sur le chemin de paquets acheminés vers le client.

12.2 Lancement des attaques

Il est important de noter que les attaques de cette nature (injecter des réponses avec de fausses MAPPED-ADDRESS) exige que l'agresseur soit capable d'espionner les demandes envoyées du client au serveur (ou d'agir comme un MITM pour de telles attaques). Cela se produit parce que les demandes STUN contiennent un identifiant de transaction, choisi par le client, qui est aléatoire avec une entropie de 128 bits. Le serveur fait écho à cette valeur dans la réponse, et le client ignore toute réponse qui n'a pas l'ID de transaction correspondant. Donc, pour qu'un agresseur fournisse une fausse réponse qui soit acceptée par le client, l'agresseur doit connaître l'ID de transaction de la demande. La grande quantité d'incertitude, combinée au besoin de savoir quand le client envoie une demande, empêche les attaques qui impliquent de deviner l'ID de transaction.

Dans la mesure où toutes les attaques ci-dessus reposent sur cette seule prémisse – injecter une réponse avec une fausse MAPPED-ADDRESS – la prévention des attaques est réalisée en empêchant cette seule opération. Pour l'empêcher, nous devons considérer les diverses façons dont elle peut s'accomplir. Elles sont plurielles :

12.2.1 Approche I : Compromission d'un serveur STUN légitime

Dans cette attaque, l'agresseur compromet un serveur STUN légitime au moyen d'un virus ou d'un cheval de Troie. Cela permettra vraisemblablement à l'agresseur de subvertir le serveur STUN, et de contrôler les types de réponses qu'il génère.

Compromettre un serveur STUN peut aussi conduire à découvrir ses ports ouverts. La connaissance d'un port ouvert crée une opportunité pour des attaques de DoS sur ces ports (ou des attaques de DDoS si le NAT traversé est un NAT en plein cône). Découvrir des ports ouverts est déjà assez trivial en utilisant la preuve de port, et donc cela ne représente pas une menace majeure.

12.2.2 Approche II : Attaques DNS

Les serveurs STUN sont découverts en utilisant les enregistrements SRV de DNS. Si un agresseur peut compromettre le DNS, il peut injecter de faux enregistrements qui transposent un nom de domaine en l'adresse IP d'un serveur STUN contrôlé par l'agresseur. Cela lui permet d'injecter de fausses réponses pour lancer l'une des attaques ci-dessus.

12.2.3 Approche III : Routeur ou NAT illégal

Plutôt que de compromettre le serveur STUN, un agresseur peut amener un serveur STUN à générer des réponses avec la mauvaise MAPPED-ADDRESS en compromettant un routeur ou un NAT sur le chemin allant du client au serveur STUN. Lorsque la demande STUN passe à travers le routeur ou NAT falsifié, il réécrit l'adresse source du paquet pour qu'elle soit celle de la MAPPED-ADDRESS désirée. Cette adresse ne peut pas être arbitraire. Si l'agresseur est sur l'Internet public (c'est-à-dire qu'il n'y a pas de NAT entre lui et le serveur STUN), et si l'agresseur ne modifie pas la demande STUN, l'adresse doit avoir la propriété que les paquets envoyés du serveur STUN à cette adresse vont être acheminés à travers le routeur compromis. Cela parce que le serveur STUN va renvoyer les réponses à l'adresse de source de la demande. Avec une adresse de source modifiée, la seule façon d'atteindre le client est que le routeur compromis les dirige sur ce point. Si l'agresseur est sur l'Internet public, mais qu'il peut modifier la demande STUN, il peut insérer un attribut RESPONSE-ADDRESS dans la demande, qui contiendra l'adresse de source réelle de la demande STUN. Cela amènera le serveur à envoyer la réponse au client, indépendamment de l'adresse de source que voit le serveur STUN. Cela donne à l'agresseur la capacité de forger une adresse de source arbitraire lorsqu'il fait suivre la demande STUN.

Si l'agresseur est sur un réseau privé (c'est-à-dire qu'il y a des NAT entre lui et le serveur STUN), l'agresseur ne pourra pas forcer le serveur à générer des MAPPED-ADRESS arbitraires dans les réponses. Il sera seulement capable de forcer le serveur STUN à générer des MAPPED-ADDRESS qui acheminent sur le réseau privé. Cela parce que le NAT entre l'agresseur et le serveur STUN va réécrire l'adresse de source de la demande STUN, la transposant en une adresse publique qui achemine sur le réseau privé. A cause de cela, l'agresseur peut seulement forcer le serveur à générer de fausses adresses transposées qui acheminent au réseau privé. Malheureusement, il est possible qu'un NAT de mauvaise qualité accepte de transposer une adresse allouée publique en une autre adresse publique (par opposition à une adresse privée interne), auquel cas l'agresseur peut fabriquer l'adresse de source dans une demande STUN comme étant une adresse publique arbitraire. Ce type de comportement de la part d'un NAT n'est pas rare.

12.2.4 Approche IV : MITM

Comme alternative à l'approche III, si l'agresseur peut placer un élément sur le chemin allant du client au serveur, l'élément peut agir comme un intermédiaire. Dans ce cas, il peut intercepter une demande STUN, et générer une réponse STUN directement avec toute valeur désirée du champ MAPPED-ADDRESS. Autrement, il peut transmettre la demande STUN au serveur (après une possible modification), recevoir la réponse, et la transmettre au client. Lors de la transmission de la demande et de la réponse, cette attaque est soumise aux mêmes limitations sur la MAPPED-ADDRESS que celles décrites au paragraphe 12.2.3.

12.2.5 Approche V : Injection de réponse plus DoS

Dans cette approche, l'agresseur n'a pas besoin d'être un MITM (comme dans les approches III et IV). En fait, il doit seulement être capable d'espionner sur une segment de réseau qui transporte des demandes STUN. Cela est facile dans de nombreux réseaux d'accès comme Ethernet ou du 802.11 non protégé. Pour injecter la réponse falsifiée, l'agresseur écoute sur le réseau l'arrivée d'une demande STUN. Quand il en voit une, il lance simultanément une attaque de DoS sur le serveur STUN et génère sa propre réponse STUN avec la valeur de MAPPED-ADDRESS désirée. La réponse STUN générée par l'agresseur va atteindre le client, et l'attaque de DoS contre le serveur vise à empêcher la réponse légitime du serveur d'atteindre le client. On peut objecter que l'agresseur peut le faire sans l'attaque de DoS sur le serveur, à condition que la réponse falsifiée batte de vitesse la vraie réponse au client, que le client utilise la première réponse, et ignore la seconde (bien qu'elles soient différentes).

12.2.6 Approche VI : Duplication

Cette approche est semblable à l'approche V. L'agresseur cherche une demande STUN sur le réseau. Quand il en voit une, il génère sa propre demande STUN au serveur. Cette demande STUN est identique à celle qu'il a vu, mais avec une adresse IP de source falsifiée. L'adresse falsifiée est égale à celle que l'agresseur souhaite placer dans la MAPPED-ADDRESS de la réponse STUN. En fait, l'agresseur génère un flux de tels paquets. Le serveur STUN va recevoir la demande d'origine plus une inondation de duplicata des faux. Il génère des réponses à chacun. Si le flux est suffisamment important pour que les réponses congestionnent les routeurs ou quelque autre équipement, il y a une probabilité raisonnable pour que la seule bonne réponse soit perdue (avec un grand nombre des fausses), mais le résultat net est que seules les réponses falsifiées sont reçues par le client STUN. Ces réponses sont toutes identiques et contiennent toutes la MAPPED-ADDRESS que l'agresseur veut qu'utilise le client.

Le flot de paquets dupliqués n'est pas nécessaire (c'est-à-dire qu'une seule demande falsifiée est envoyée), tant que la réponse falsifiée bat de vitesse la vraie pour atteindre le client, que le client utilise la première réponse, et ignore la seconde (bien qu'elles soient différentes).

Noter que, dans cette approche, le lancement d'une attaque de DoS contre le serveur STUN ou le réseau IP pour empêcher la réponse valide d'être envoyée ou reçue pose problème. L'agresseur a besoin que le serveur STUN soit disponible pour traiter sa propre demande. Du fait des retransmissions périodiques de la demande provenant du client, il ne reste qu'une très petite fenêtre d'opportunité. L'agresseur DOIT commencer l'attaque de DoS immédiatement après la demande réelle du client, pour causer la mise à l'écart de la réponse correcte, puis cesser l'attaque de DoS afin d'envoyer sa propre demande, tout cela avant la prochaine retransmission du client. Du fait du faible espacement des retransmissions (100 ms à quelques secondes), c'est très difficile à faire.

A côté des attaques de DoS, il peut y avoir d'autres façons d'empêcher que la demande réelle du client n'atteigne le serveur. Des manipulations de couche 2, par exemple, peuvent arriver à ce résultat.

Heureusement, l'approche IV subit les mêmes limitations que celles évoquées au paragraphe 12.2.3, qui limitent la gamme des MAPPED-ADDRESS que l'agresseur peut amener le serveur STUN à générer.

12.3 Contre-mesures

STUN fournit des mécanismes pour contrer les approches décrites ci-dessus, et d'autres techniques, non-STUN, peuvent aussi bien être utilisées.

Tout d'abord, il est RECOMMENDÉ que les réseaux avec des clients STUN mettent en œuvre un filtrage de source d'entrée (RFC 2827 [7]). Ceci est particulièrement important pour les NAT eux-mêmes. Comme expliqué au paragraphe 12.2.3, les NAT qui n'effectuent pas cette vérification peuvent être utilisés comme "réflecteurs" dans les attaques de DDoS. La plupart des NAT font effectivement cette vérification au titre du mode de fonctionnement par défaut. Nous recommandons fortement aux acheteurs de NAT de s'assurer que cette capacité est présente et activée.

Ensuite, il est RECOMMENDÉ que les serveurs STUN fonctionnent sur des hôtes dédiés à STUN, tous les ports UDP et TCP étant désactivés à l'exclusion des ports STUN. Cela pour empêcher les virus et chevaux de Troie d'infecter les serveurs STUN, et éviter leur compromission. Cela aide à atténuer l'approche I (paragraphe 12.2.1).

Troisièmement, pour empêcher les attaques de DNS du paragraphe 12.2.2, le paragraphe 9.2 recommande que le client confronte les lettres de créance fournies par le serveur avec le nom utilisé dans la consultation DNS.

Finalement, toutes les attaques ci-dessus reposent sur l'idée que le client va prendre l'adresse transposée qu'il a apprise de STUN, et qu'il va l'utiliser dans des protocoles de couche d'application. Si le cryptage et l'intégrité de message sont fournis au sein de ces protocoles, les attaques d'espionnage et d'usurpation d'identité peuvent être empêchées. Comme telles, les applications qui font usage d'adresses STUN dans des protocoles d'application DEVRAIENT utiliser l'intégrité et le cryptage, même si le niveau d'obligation DEVRAIT n'est pas spécifié pour ce protocole. Par exemple, les applications multimédia qui utilisent des adresses STUN pour recevoir du trafic RTP utiliseraient RTP sécurisé [16].

Les trois techniques ci-dessus sont des mécanismes non-STUN. STUN fournit lui-même plusieurs contre-mesures.

Les approches IV (paragraphe 12.2.4), quand on génère la réponse localement, et V (paragraphe 12.2.5) exigent d'un agresseur qu'il génère une réponse falsifiée. On empêche cette attaque en utilisant le mécanisme d'intégrité de message fourni dans STUN, décrit au paragraphe 8.1.

Les approches III (paragraphe 12.2.3) IV (paragraphe 12.2.4), quand on utilise la technique de relais, et VI (12.2.6), ne peuvent cependant pas être empêchées au moyen des signatures de serveur. Les deux approches sont plus puissantes quand l'agresseur peut modifier la demande, en insérant une RESPONSE-ADDRESS qui achemine sur le client. Heureusement, de telles modifications peuvent être empêchées en utilisant les techniques d'intégrité de message décrites au paragraphe 9.3. Cependant, ces trois approches sont encore fonctionnelles lorsque l'agresseur ne modifie que l'adresse de source de la demande STUN. Malheureusement, c'est la seule chose qui ne puisse être protégée par des moyens cryptographiques, car c'est le changement même que STUN cherche à détecter pour en faire rapport. C'est donc une faiblesse inhérente des NAT, non soluble dans STUN. Pour aider à atténuer ces attaques, le paragraphe 9.4 fournit plusieurs méthodes heuristiques que le client peut suivre. Le client cherche des réponses incohérentes ou surnuméraires, qui sont toutes deux des signes des attaques décrites plus haut. Cependant, ces méthodes heuristiques ne

sont que cela – des méthodes heuristiques, et il ne peut être garanti qu'elles empêchent les attaques. La méthode heuristique paraît empêcher les attaques dont nous savons aujourd'hui comment elles sont lancées. Les développeurs de mises en œuvre devraient rester aux aguets d'informations sur les nouvelles méthodes heuristiques qui pourraient être nécessaires à l'avenir. De telles informations seront distribuées sur la liste des destinataires du MIDCOM de l'IETF, midcom@ietf.org.

12.4 Menaces résiduelles

Aucune des contre-mesures dont la liste figure ci-dessus ne peut empêcher les attaques décrites au paragraphe 12.2.3 si l'agresseur est dans les chemins de réseau appropriés. En particulier, considérons le cas où l'agresseur veut convaincre le client C qu'il a l'adresse V. L'agresseur a besoin d'avoir un élément de réseau sur le chemin entre A et le serveur (afin de modifier la demande) et sur le chemin entre le serveur et V de sorte qu'il puisse transmettre la réponse à C. De plus, s'il y a un NAT entre l'agresseur et le serveur, V DOIT aussi être derrière le même NAT. Dans une telle situation, l'agresseur peut gagner l'accès à tout le trafic de couche application ou monter l'attaque de DDOS décrite au paragraphe 12.1.1. Noter que tout hôte existant dans la relation topologique adéquate peut se faire "DDOSer". Il n'est pas nécessaire d'utiliser STUN.

13. Considérations sur l'IANA

STUN ne peut pas faire l'objet d'une extension. Les changements au protocole sont faits au moyen d'une révision standard de la présente spécification. Il en résulte qu'il n'est pas nécessaire de faire de réservations. Toutes les extensions à venir établiront les réservations nécessaires.

14. Considérations sur l'IAB

L'IAB a étudié le problème de la "fixation unilatérale de sa propre adresse" (UNSAF, *Unilateral Self Address Fixing*), qui est le processus général par lequel un client tente de déterminer son adresse dans un autre domaine de l'autre côté d'un NAT à travers un mécanisme de réflexion de protocole collaboratif (RFC 3424 [17]). STUN est un exemple de protocole effectuant ce type de fonction. L'IAB a demandé que tout protocole développé dans ce but fasse apparaître un ensemble spécifique de considérations. La présente section répond à cette exigence.

14.1 Définition du problème

D'après la RFC 3424 [17], toute proposition d'UNSAF DOIT fournir :

Une définition précise d'un problème spécifique d'une portée limitée qui est à résoudre à l'aide de la proposition d'UNSAF. Une solution à court terme ne devrait pas être généralisée pour résoudre d'autres problèmes ; c'est pourquoi "les solutions à court terme n'en sont pas".

Les problèmes spécifiques résolus par STUN sont :

- Fournir à un client un moyen de détecter la présence d'un ou plusieurs NAT entre lui et un serveur géré par un fournisseur de service sur l'Internet public. L'objet d'une telle détection est de déterminer les étapes supplémentaires qui pourraient être nécessaires afin de recevoir les services de ce fournisseur particulier.
- Fournir à un client le moyen de détecter la présence d'un ou plusieurs NAT entre lui et un autre client, lorsque le second client peut être atteint depuis le premier, mais qu'on ne sait pas si le second client réside sur l'Internet public.
- Fournir à un client le moyen d'obtenir une adresse sur l'Internet public de la part d'un NAT non symétrique, dans le but exprès de recevoir du trafic UDP entrant d'un autre hôte, logé à cette adresse.

STUN ne vise pas TCP, entrant ou sortant, et ne vise pas les communications UDP sortantes.

14.2 Stratégie de sortie

A partir de [17], toute proposition UNSAF DOIT fournir :

La description de toute stratégie/plan de transition de sortie. Les meilleures solutions à court terme sont celles qui seront de moins en moins utilisées avec le développement de la technologie appropriée.

STUN a sa propre stratégie de sortie incorporée. Cette stratégie est l'opération de détection qui est effectuée en préambule de l'opération réelle UNSAF de fixation d'adresse. Cette opération de découverte, expliquée au paragraphe 10.1, essaye de révéler l'existence, et le type, de tous NAT entre le client et le réseau du fournisseur de service. Alors que la détection du type spécifique de NAT peut être délicate, la découverte de l'existence de NAT est elle même assez solide. Comme les NAT sont éliminés progressivement par le développement de IPv6, l'opération de découverte donnera immédiatement le résultat qu'il n'y a pas de NAT, et aucune opération ultérieure n'est nécessaire. Bien sûr, l'opération de découverte elle-même peut être utilisée pour aider au développement de IPv6 ; si un utilisateur détecte un NAT entre lui et l'Internet public, il peut appeler son fournisseur d'accès et s'en plaindre.

STUN peut aussi faciliter l'introduction de midcom. Comme des NAT à capacité midcom sont déployés, les applications, au lieu d'utiliser STUN (qui réside aussi à la couche application), vont d'abord allouer un lien d'adresse utilisant midcom. Cependant, une limitation bien connue de midcom est qu'il ne fonctionne que lorsque l'agent connaît le boîtier de médiation à travers lequel s'écoule son trafic. Une fois que des liens ont été alloués à partir de ces boîtiers de médiation, une procédure de détection STUN peut valider qu'il n'y a pas de boîtier de médiation supplémentaire sur le chemin entre l'Internet public et le client. Si c'est le cas, l'application peut continuer de fonctionner en utilisant les liens d'adresse alloués depuis midcom. Si ce n'est pas le cas, STUN donne un mécanisme d'auto-fixation d'adresse à travers les boîtiers de médiation restants sans capacité midcom. Et donc, STUN fournit un moyen d'aider à la transition vers les réseaux à pleine capacité midcom.

14.3 Ce que STUN apporte

D'après [17], toute proposition d'UNSAF DOIT fournir :

La discussion des questions spécifiques qui rendent les systèmes plus "fragiles". Par exemple, les approches qui impliquent d'utiliser des données à plusieurs niveaux de réseau créent plus de dépendance, accroissent les contraintes de mise au point, et rendent les transitions plus difficiles.

STUN introduit la fragilité dans le système de plusieurs façons :

- Le processus de découverte suppose une certaine classification des appareils fondée sur leur traitement d'UDP. Il pourrait y avoir d'autres types de NAT installés qui ne correspondent pas à un de ces modèles. Donc les NAT de l'avenir peuvent n'être pas correctement détectés par STUN. Les clients STUN (mais pas les serveurs) auront besoin de modifications pour s'en accommoder.
- L'utilisation de l'acquisition de lien de STUN ne fonctionne pas pour tous les types de NAT. Elle ne fonctionnera que pour toute application de NAT en plein cône. Pour les NAT en cône restreint et les NAT à restriction de port, elle fonctionnera pour certaines applications en fonction de l'application. Un traitement spécifique de l'application sera généralement nécessaire. Pour les NAT symétriques, l'acquisition de lien ne donnera pas d'adresse utilisable. L'étroite dépendance au type spécifique de NAT rend le protocole fragile.
- STUN suppose que le serveur existe sur l'Internet public. Si le serveur est localisé dans un autre domaine d'adresse privé, l'utilisateur peut être ou n'être pas capable d'utiliser l'adresse découverte pour communiquer avec d'autres utilisateurs. Il n'y a aucune façon de détecter une telle condition.
- Les liens alloués depuis le NAT ont besoin d'être continuellement rafraîchis. Comme les temporisations pour ces liens sont très spécifiques de la mise en oeuvre, l'intervalle de rafraîchissement ne peut être aisément déterminé. Lorsque le lien n'est pas activement utilisé pour recevoir du trafic, mais pour attendre un message entrant, le rafraîchissement de lien va consommer inutilement de la bande passante du réseau.
- L'utilisation du serveur STUN comme élément de réseau supplémentaire introduit un autre point d'attaque potentielle contre la sécurité. Ces attaques sont largement empêchées par les mesures de sécurité fournies par STUN, mais pas entièrement.
- L'utilisation du serveur STUN comme élément de réseau supplémentaire introduit un autre point de faiblesse. Si le client ne peut pas localiser un serveur STUN, ou si le serveur se trouve indisponible du fait d'une défaillance, l'application ne peut pas fonctionner.
- L'utilisation de STUN pour découvrir les liens d'adresse aura pour résultat une augmentation de délai pour les applications. Par exemple, une application de voix sur IP verra une augmentation du délai d'établissement des appels égale à au moins un RTT avec le serveur STUN.
- La découverte des durées de vie de lien favorise l'erreur. Elle suppose que la même durée de vie va exister pour tous les liens. Cela peut n'être pas vrai si le NAT utilise des durées de vie de lien dynamiques pour traiter la surcharge, ou si le NAT lui-même se réamorce durant le processus de découverte.
- STUN impose certaines restrictions à la topologie des réseaux pour un fonctionnement approprié. Si le client A obtient une adresse du serveur STUN X, et l'envoie au client B, B peut n'être pas capable de faire des envois à A en utilisant cette adresse IP. L'adresse ne fonctionnera pas si l'un des éléments qui suit est vrai :
    - Le serveur STUN n'est pas dans un domaine d'adresse qui soit un ancêtre commun (topologiquement) de deux clients A et B. Par exemple, considérons les clients A et B, qui ont tous deux des appareils NAT résidentiels. Les deux appareils se connectent à leurs câblo-opérateurs, mais les deux clients ont des fournisseurs différents. Chaque fournisseur a un NAT en frontal de son réseau entier, le connectant à l'Internet public. Si le serveur STUN utilisé par A est dans le réseau du câblo-opérateur de A, une adresse obtenue par

lui ne sera pas utilisable par B. Le serveur STUN DOIT être dans le réseau dans lequel est un ancêtre commun des deux – dans ce cas, l'Internet public.

- Le serveur STUN est dans un domaine d'adresse qui est un ancêtre commun des deux clients, mais les deux clients sont derrière le même NAT qui les connecte à ce domaine d'adresse. Par exemple, si les deux clients de l'exemple précédent ont le même câblo-opérateur, ce câblo-opérateur à un seul NAT qui connecte leur réseau à l'Internet public, et le serveur STUN était sur l'Internet public, l'adresse obtenue par A ne serait pas utilisable par B. Et cela parce que certains NAT n'accepteront pas un paquet interne envoyé à une adresse IP publique qui serait retransposée sur une adresse interne. Pour faire avec cela, des mécanismes de protocole supplémentaires ou d'autres paramètres de configuration doivent être introduit pour détecter ce cas.

- Plus grave, STUN introduit des menaces potentielles sur la sécurité qui ne peuvent pas être éliminées. La présente spécification décrit des méthodes heuristiques qui peuvent être utilisées pour atténuer le problème, mais il est probablement insoluble étant donné ce que STUN essaye d'accomplir. Ces problèmes de sécurité sont pleinement décrits à la Section 12.

14.4 Exigences pour une solution à long terme

D'après [17], toute proposition d'UNSAF DOIT fournir :

L'identification des exigences pour des solutions à long terme, techniquement valables – contribuer au processus d'élaboration de la bonne solution à plus long terme.

Notre expérience avec STUN nous a conduit à l'exigence suivante pour une solution à long terme au problème du NAT :

Les demandes de liens et de commande des autres ressources d'un NAT doivent être explicites. Beaucoup de la fragilité de STUN découle du fait qu'il devine les paramètres du NAT, plutôt que d'indiquer au NAT les paramètres à utiliser.

Le contrôle doit être "dans la bande". Il y a beaucoup trop de scénarios dans lesquels le client ne saura pas à l'avance la localisation du boîtier de médiation. Au lieu de cela, le contrôle de tels boîtiers doit se faire dans la bande, voyageant sur le même chemin que les données emprunteront elles-mêmes. Cela garantit que le bon ensemble de boîtiers de médiation est sous contrôle. Ceci n'est vrai que pour les contrôles directs, les contrôles de tiers sont mieux traités en utilisant la trame des boîtiers de médiation.

Le contrôle doit être limité. Les utilisateurs auront besoin de communiquer à travers des NAT qui sont en-dehors de leur contrôle administratif. Afin que les fournisseurs ne soient pas réticents à installer des NAT qui puissent être contrôlés par des utilisateurs de différents domaines, la portée de tels contrôles doit être extrêmement limitée - normalement, à allouer un lien pour atteindre l'adresse d'où viennent les paquets de contrôle.

La simplicité est le but à atteindre. Le protocole de contrôle devra être mis en œuvre chez des clients très ordinaires. Les serveurs auront besoin de supporter des charges extrêmement lourdes. Le protocole devra être extrêmement robuste, étant le précurseur d'un hôte de protocoles d'application. Pour cela, la simplicité est la clé.

14.5 Problèmes avec les boîtiers NAPT existants

D'après [17], toute proposition d'UNSAF DOIT fournir :

La discussion de l'impact des questions pratiques notées avec les NA[P]T existants, en place et les rapports d'expérimentation.

Plusieurs des questions pratiques sur STUN impliquent une mise à l'épreuve future – la défaillance du protocole lorsque de nouveaux types de NAT seront installés. Heureusement ce n'est pas un problème actuel, car la plupart de NAT installés suivent les types pris en charge par STUN. La principale utilisation que STUN a trouvée est dans le domaine de VoIP, pour faciliter l'allocation des adresses pour recevoir du trafic RTP [12]. Dans cette application, les messages de réanimation périodiques sont fournis par le trafic RTP lui-même. Cependant, plusieurs problèmes pratiques surgissent pour RTP. D'abord, RTP suppose que le trafic RTCP est sur un port supérieur de un au trafic RTP. Cette propriété d'appariement ne peut pas être garantie à travers des NAT qui ne sont pas directement contrôlables. Il en résulte que le trafic RTCP peut n'être pas correctement reçu. Des extensions de protocole à SDP ont été proposées qui atténuent cela en permettant au client de signaler un port différent pour RTCP [18]. Cependant, il y aura des problèmes d'interopérabilité pendant quelques temps.

Pour VoIP, la suppression de silence peut causer un trou dans la transmission de paquets RTP. Il pourrait en résulter la perte d'un lien au milieu d'un appel, si cette période de silence dépasse la temporisation du lien. Ceci peut être atténué en envoyant des paquets de silence occasionnels pour garder le lien en vie. Cependant, le résultat en est une fragilité accrue, le bon fonctionnement dépendant de l'algorithme de suppression de silence utilisé, de l'utilisation d'un codec de bruit de confort, de la durée de la période de silence, et de la durée de vie du lien dans le NAT.

14.6 En conclusion

Les problèmes de STUN ne sont pas des fautes de conception de STUN. Les problèmes de STUN se rapportent au manque de comportements et contrôles normalisés dans les NAT. Le résultat de ce manque de normalisation est une prolifération d'appareils dont le comportement est totalement imprévisible, extrêmement variable, et incontrôlable. STUN fait de son mieux dans un environnement aussi hostile. Finalement, la solution est de rendre l'environnement moins hostile, et d'introduire des contrôles et des comportements normalisés dans le NAT. Cependant, jusqu'à ce que vienne ce moment, STUN fournit une bonne solution à court terme étant donné les mauvaises conditions dans lesquelles il est forcé de fonctionner.

15. Remerciements

Les auteurs voudraient remercier Cedric Aoun, Pete Cordell, Cullen Jennings, Bob Penfield et Chris Sullivan pour leurs commentaires, et Baruch Sterman et Alan Hawrylyshen pour les mises en œuvre initiales. Merci à Leslie Daigle, Allison Mankin, Eric Rescorla, et Henning Schulzrinne pour les apports d'IESG et IAB à ce travail.

16. Références normatives

[1] Bradner, S., "Key words for use in RFCs to indicate requirement levels" *(Mots clé à utiliser dans les RFC pour indiquer les niveaux d'exigence)*, BCP 14, RFC 2119, mars 1997.

[2] Dierks, T. et C. Allen, "The TLS protocol Version 1.0" *(Protocole TLS, version 1.0)*, RFC 2246, janvier 1999.

[3] Gulbrandsen, A., Vixie, P. et L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)" *(RR DNS pour la spécification de la localisation des services)*, RFC 2782, février 2000.

[4] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)" *(Suites de chiffrement pour norme de chiffrement avancée (AES) pour la sécurité de la couche transport (TLS))*, RFC 3268, juin 2002.

[5] Rescorla, E., "HTTP over TLS" *(http sur TLS)*, RFC 2818, mai 2000.

[6] Postel, J., "Internet Protocol" *(Le protocole Internet)*, STD 5, RFC 791, septembre 1981.

[7] Ferguson, P. et D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing" *(Filtrage à l'entrée du réseau : vaincre les attaques de déni de service qui utilisent le camouflage d'adresse de source IP)*, BCP 38, RFC 2827, mai 2000.

17. Références informatives

[8] Senie, D., "Network Address Translator (NAT)-Friendly Application Design Guidelines" *(Lignes directrices pour la conception d'applications faciles de traducteur d'adresse réseau (NAT))*, RFC 3235, janvier 2002.

[9] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A. et A. Rayhan, "Middlebox Communication Architecture and Framework" (*Architecture et cadre de travail des communications par boîtier de médiation)*, RFC 3303, août 2002.

[10] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. et E. Schooler, "SIP: Session Initiation Protocol" *(SIP :Protocole d'initialisation de session)*, RFC 3261, juin 2002.

[11] Holdrege, M. et P. Srisuresh, "Protocol Complications with the IP Network Address Translator" *(Difficultés du protocole avec le traducteur d'adresse réseau IP)*, RFC 3027, janvier 2001.

[12] Schulzrinne, H., Casner, S., Frederick, R. et V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications" *(RTP : un protocole de transport pour les applications en temps réel)*, RFC 1889, janvier 1996.

[13] Krawczyk, H., Bellare, M. et R. Canetti, "HMAC: Keyed-Hashing for Message Authentication" *(HMAC : hachage de clés pour l'authentification de message)*, RFC 2104, février 1997.

[14] Kohl, J. et C. Neuman, "The kerberos Network Authentication Service (V5)" *(Le service d'authentification de réseau Kerberos (Version 5))*, RFC 1510, septembre 1993.

[15] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. et T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1" *(Protocole de transfert hypertexte – http/1.1)*, RFC 2616, juin 1999.

[16] Baugher M., et al., "The secure real-time transport protocol" *(Protocole sécurisé de transport en temps réel)*, Travail en cours.

[17] Daigle, L., Editeur, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation" *(Considérations de l'IAB sur la fixation unilatérale de sa propre adresse (UNSAF) à travers la traduction d'adresse réseau)*, RFC 3424, novembre 2002.

[18] Huitema, C., "RTCP attribute in SDP" *(Attribut RTCP en SDP)*, Travail en cours.

18. Adresses des auteurs

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Joel Weinberger
Dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jweinberger@dynamicsoft.com

Christian Huitema
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
EMail: huitema@microsoft.com

Rohan Mahy
Cisco Systems
101 Cooper St
Santa Cruz, CA 95060
EMail: rohan@cisco.com

19. Déclaration de copyright

les procédures de copyright définies dans les processus des normes Internet DOIVENT être suivies, ou selon les besoin de la traduction dans d’autres langues que l’anglais.

Les permissions limitées accordées ci-dessus sont perpétuelles et ne seront pas révoquées par la Société Internet ou ses successeurs ou héritiers.

Le présent document et les informations y contenues sont fournies sur une base "EN L’ETAT" et LA INTERNET SOCIETY ET LA INTERNET ENGINEERING TASK FORCE DECLINENT TOUTES GARANTIES, EXPRIMEES OU IMPLICITES, Y COMPRIS MAIS NON LIMITEES A TOUTE GARANTIE QUE L’UTILISATION DES INFORMATIONS CI-ENCLOSES NE VIOLENT AUCUN DROIT OU AUCUNE GARANTIE IMPLICITE DE COMMERCIALISATION OU D’APTITUDE A UN OBJET PARTICULIER.

Remerciement

Le financement de la fonction d’édition des RFC est actuellement fourni par Internet Society.