

Introduction	2
Partie 1 : Eléments d'architecture	3
Dans le TP 1	3
Dans le cas général	3
Architecture « académique »	4
Architecture « industrielle »	4
Autres utilisations des composants de filtrage et de translation d'adresses	4
Filtrages	4
Translations d'adresses	6
Les technologies disponibles	6
Partie 2 : Protocoles, implémentations	8
La translation statique	8
La translation par port (NATP)	9
La table d'état d'un composant NAT	9
La translation de port source déterministe (TP n°3)	11
TP n°2 - Les connexions entrantes	11
TP n°3 - Les connexions entrantes- La table de NAT	13
La traversée du NAT	13
Difficultés réseaux	13
Traversée des applications	13
Le facteur d'échelle – les opérateurs et leurs infrastructures d'accès à Internet	18
Le contexte	18
Les impacts pour l'opérateur	19
La sécurité	22
Synthèse et conséquences pour les ingénieurs de l'école	23
<a href="#">Exercice n°1 - Réflexion sur les métiers</a>	<a href="#">23</a>
Indications	23
Partie 3 : Exemples de cas d'usages (sortant) :	24
Depuis des PC derrière une box opérateur SOHO	24
Connexion à un site web externe depuis un PC de l'école	24
Connexion d'une application sur Smartphone à un service sur internet	24
Bibliographie	25
<a href="#">Exercice n°2 - Réalisation d'une bibliographie</a>	<a href="#">25</a>
Indications	25

# Introduction

Ce document vient en complément des trois TP qui ont eu lieu pour le cours NE372 pendant la période de confinement due au COVID-19. Il replace les sujets des TP dans le cadre général du NAT, et il devrait permettre de répondre aux questions qui sont couramment posées par les élèves dans ce cours depuis plusieurs années. Il préfigure également les enseignements de 4<sup>ème</sup> et 5<sup>ème</sup> années, notamment sur les questions de sécurité.

Les lecteurs ciblés par le texte sont des apprenants dans le domaine des systèmes qui possèdent des prérequis de base sur le sujet ; typiquement des élèves ingénieurs en cours d'apprentissage. Le texte peut intéresser également des ingénieurs confirmés mais qui ne sont pas des spécialistes de la question.

Ce document est synthétique mais dense. Vous pouvez considérer qu'il fait le tour de tout ce que vous devez savoir sur le sujet.

Nous aborderons d'abord des questions d'architecture, puis de protocole et d'implémentation (comment ça marche), et enfin à titre d'exemple quelques cas d'usage.

Deux **exercices** sont proposés pour ce cours, leurs énoncés sont inclus dans ce document. Ils sont repérés en **bleu** dans le texte.

Un QCM sera proposé sur Chamilo pour chaque partie de ce document.

## Vocabulaire :

*Il n'y a pas de terminologie établie sur le sujet du NAT.*

*« Bidirectionnel/unidirectionnel », « source/destination-nat (Snat)/Dnat », « full / restricted-cone » ... font partie des termes employés, mais ce n'est jamais satisfaisant, parfois confus. C'est assez logique : le sujet est compliqué et il recouvre beaucoup de points (archi, protocole, technos...) de niveaux différents. Il vaut mieux donc ne pas s'en préoccuper dans un premier temps, pour se concentrer sur la compréhension des concepts.*

*Les **mots-clés** de ce document sont mis en valeur en **rouge**.*

## Partie 1 : Eléments d'architecture

### Dans le TP 1

- Vous avez vu dans le sujet qu'il y avait deux composants distincts pour gérer les connexions : un routeur (qui fait le lien avec internet), et un firewall (une linux-box dans ce cas) qui s'intercale entre le routeur et le réseau local.
- Dans le sujet, on demandait de traiter deux questions : celle de la translation d'adresse, puis celle du filtrage des flux applicatifs.
- Dans le cadre pédagogique du TP, ces deux fonctions étaient traitées par un firewall dit « **full-state** » avec un système linux de base, des **Iptables** et du suivi de connexion (**conntrack**).

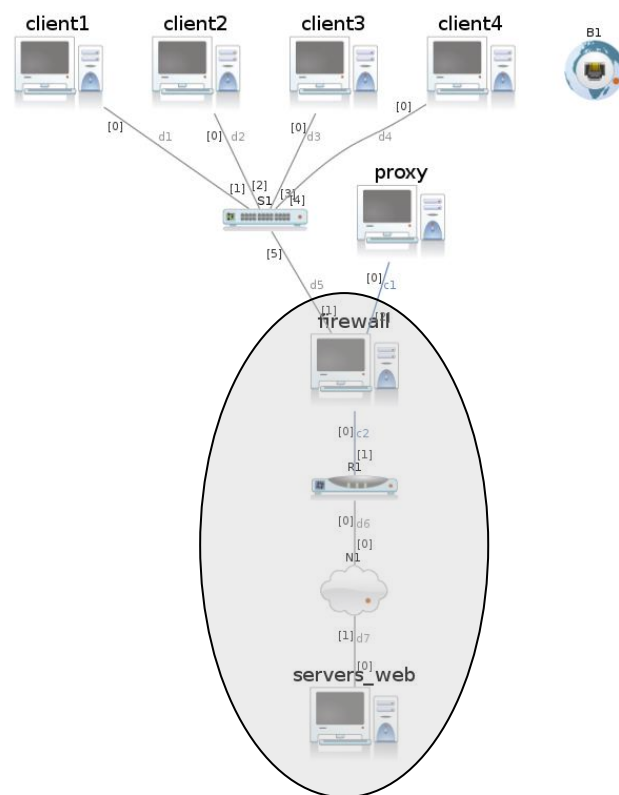


Figure 1 : Architecture du sujet du TP

### Dans le cas général

En réalité ces deux fonctions (NAT/filtrage) peuvent être prises en charge par l'un ou l'autre des composants : routeur ou firewall.

- Les routeurs sont en général bien équipés pour faire du NAT, et ils disposent de procédés de filtrage de flux (en général des **ACL**<sup>1</sup>). Ils sont en revanche moins riches en filtrage applicatif (ce n'est pas leur rôle de base).
- Les firewalls sous forme de « host » (serveurs) comme dans le TP, peuvent faire les deux, mais ne sont pas forcément taillés spécifiquement pour gérer des flux réseaux en grande quantité.

<sup>1</sup> Access Control List

## **Architecture « académique »**

- D'après ce qui a été dit précédemment, le bon modèle d'architecture consisterait à séparer les fonctions sur les composants : NAT sur le routeur, et filtrages sur le firewall, ce qui a d'ailleurs été un modèle appliqué largement pendant plusieurs années.

## **Architecture « industrielle »**

A l'heure actuelle, la pratique consiste pourtant à regrouper les fonctions sur un seul composant : le –ou les firewall(s), essentiellement pour les raisons suivantes :

- les serveurs qui servent de socles aux firewalls ont désormais une capacité de traitement suffisante pour assurer les deux fonctions ;
- le traitement correct du NAT (« traversée du NAT ») nécessite une inspection en profondeur des données, y compris applicatives (nous verrons ça plus loin), opérations pour lesquelles les routeurs ne sont pas adaptés ;
- le découpage des fonctions sur des composants différents (techno, modes opératoires, flux) complique le travail d'administration et de supervision ;
- enfin cela permet de centraliser les politiques de sécurité et leurs traitements (log par ex.) sur un seul composant, ce qui diminue les risques d'erreurs ou de trous de configurations ;
- mais une conséquence de cette centralisation c'est la création de **SPOF**<sup>2</sup> (le firewall), qui doit être pris en compte dans l'étude de la résilience du système. C'est pour cette raison qu'en règle générale les firewalls sont redondés et embarquent des protocoles de **Fail-over**<sup>3</sup>.

## **Autres utilisations des composants de filtrage et de translation d'adresses**

Dans le TP 1, les fonctions de NAT et de filtrage étaient intégrées sur un firewall «périmétrique», au sens où ce composant était placé en coupure entre le système et le monde extérieur, mais on retrouve ces fonctions à d'autres endroits de l'architecture :

### **Filtrages**

#### Firewalls du même type que celui du TP

Les firewalls du même type que celui du TP sont utilisés pour cloisonner le système en plusieurs sous-systèmes et sécuriser les flux. Les cloisonnements visés ici peuvent découler d'une nécessité de séparer des composants dédiés aux métiers de l'organisation, ou d'un besoin technique pour cloisonner des composants systèmes entre eux.

---

<sup>2</sup> Single Point Of Failure

<sup>3</sup> Basculement automatique

Exemples :

**Cas de cloisonnements métiers :**

- Pour les organisations qui ont une activité de production, on utilise un firewall pour séparer le sous-système d'information (gestion commerciale, RH, etc...) du sous-système industriel (automates, réseaux industriels, **SCADA**<sup>4</sup>)
- A l'école, nous utilisons un firewall interne pour séparer les flux des projets industriels de ceux l'école, ce qui permet de gérer des politiques de sécurité différentes pour les deux activités de l'organisation.

**Cas de cloisonnements systèmes :**

- Pour gérer la traçabilité du système, on peut utiliser un sous-système spécifique de gestion de traces (log), qui doit répondre à des règles de sécurité particulières. Ce sous-système peut être cloisonné et protégé par un firewall dédié.
- Dans le même objectif que dans le cas précédent, des firewalls internes sont utilisés pour cloisonner des sous-systèmes de supervision ou de sauvegarde des données.

Firewalls sur les clients

Pour atteindre des objectifs de défense en profondeur, il est possible d'activer des firewalls locaux sur les postes clients. Sur des stations Windows on peut utiliser le firewall de Microsoft (il en existe d'autres), et sur des stations Linux les mêmes mécanismes que ceux du TP (**Iptables**).

Filtrages par des équipements actifs du réseau

Les équipements actifs du réseau (routeurs, switches) sont capables de faire du filtrage par des règles de type **ACL**.

Ces **ACL** sont généralement utilisées sur des switches de cœur de réseau pour cloisonner des sous-systèmes réseaux (VLAN notamment). La tendance d'architecture actuelle est de concentrer ces contrôles sur un firewall (« router-on-a-stick » chez Cisco), dans l'objectif de centraliser le déploiement des politiques de sécurité.

Filtrages applicatifs : **WAF**<sup>5</sup>

Moins visibles dans les infrastructures (barrières logiques), les firewalls applicatifs pour les applications web sont des composants logiciels qui se greffent sur les serveurs http (comme Apache ou Nginx). Ils servent à filtrer les flux en amont des applications sur des critères spécifiques au web, comme la reconnaissance de formes d'attaques du type SQL-injection ou des codes JavaScripts réputés dangereux.

Les **WAF** sont déployés sur les serveurs (pas sur les clients). Ils contribuent à la défense en profondeur d'un système, en rajoutant une barrière à franchir pour accéder aux données.

---

<sup>4</sup> Système de supervision : Supervisory Control and Data Acquisition

<sup>5</sup> Web Application Firewall

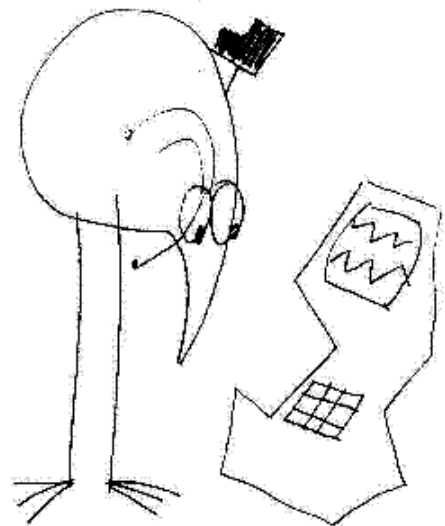
## Translations d'adresses

Les composants NAT se trouvent par nature intercalés entre le système (administration privée) et le monde extérieur (public), de la même manière que les firewalls.

- A l'intérieur du système, on n'a normalement pas besoin d'utiliser le NAT puisque la contrainte du nombre d'adresse IP est levée. Dans certains cas liés à la grande échelle du système, on peut avoir le besoin de réaliser du NAT à l'intérieur même du système mais dans la vraie vie ces cas restent marginaux (voir **CGN**<sup>6</sup> plus loin dans le document).
- D'autre part, il y a une utilisation moins visible mais très fréquente du NAT à l'intérieur du système. La situation se présente tout simplement quand un poste héberge des systèmes virtuels (VM). Quand les hosts hébergés ont besoin de connectivité IP, une des options possibles est de partager l'adresse IP du poste physique à l'aide du NAT. C'est le même principe que l'on a vu. Les machines virtuelles se partagent l'adresse du PC. Logiquement, cette option se configure dans le player, pour chaque VM.

## Les technologies disponibles

- Il y a un grand choix de produits (libres ou commerciaux) qui permettent de réaliser ces fonctions, sous des formes physiques différentes : serveurs standard / OS standards ou durcis, appliances constructeurs, ...
- Au niveau de l'administration, ce que vous avez vu en TP (**Iptables**, mode commande Linux...) vous paraît certainement très ésotérique.
  - C'est bien le cas. Bon nombre d'administrateurs de terrain ne sont pas formés à ça. D'autre part, dans un système d'entreprise, même simple, le nombre de « règles » dépasse facilement la centaine, ce qui rend la compréhension et la maintenance de ces règles forcément hasardeuses.
  - Il existe pourtant des solutions (payantes pour les meilleures) qui permettent une administration assez « sûre » de ces fonctions. Au-delà de l'interface graphique –qui n'est pas le point important, ces applications permettent surtout un contrôle strict de la pertinence, de la robustesse de la configuration et du contrôle en général de la production.



---

<sup>6</sup> Carrier Grade NAT

Mais on l'aura compris, cela a un prix dont il faut tenir compte lors de l'étude de l'architecture.

---

Ceux qui veulent aller plus loin à propos des architectures sécurisées peuvent consulter le guide de **l'ANSSI** :

« *Recommandations relatives à l'interconnexion d'un système d'information à internet* ».

Ce document est disponible sur le site de **l'ANSSI** ou sur chamilo (dossier documentation)

C'est un peu compliqué à votre stade, mais nous aurons l'occasion de revenir ensemble en détail sur ces questions pendant le cours SE501 et le projet SE502 de 5<sup>ème</sup> année.

---

## Partie 2 : Protocoles, implémentations

Dans ce chapitre, nous allons rester dans le périmètre du NAT « sortant ». Pour simplifier, nous ne pencherons par défaut que sur les services utilisant au niveau 4 TCP ou UDP (comme HTTP ou DNS)

Les autres cas (ICMP par ex.) seront abordés au cas par cas dans la partie « **traversée du NAT** », ou les études de cas (partie 3).

Dans ce qui suit, un « *composant NAT* » représentera soit un routeur soit un firewall, selon ce qui a été expliqué dans la première partie.

En préalable, ce qu'il faut bien comprendre : c'est que même si c'est le client qui a l'initiative de la connexion (cas sortant), cela va générer très probablement une réponse de son correspondant, donc des datagrammes vont arriver de l'extérieur sur le composant NAT, qui doit donc être capable de rediriger les données entrantes vers les initiateurs du dialogue (les clients).

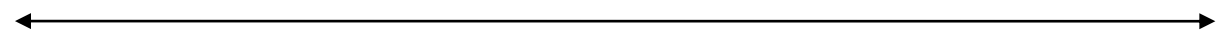
### La translation statique

C'est la question la moins compliquée (client n°3 dans le TP).

Il s'agit ici d'attribuer de façon permanente à un poste client une adresse IP publique unique.

Le composant NAT change à la volée l'adresse IP privée de client 3 en une adresse publique reconnue par Internet, et ce dans les deux sens.

Pour mémoriser la manip, le composant NAT doit maintenir une table du genre :



Adresse IP source (privée)	Adresse IP Destination (publique)	Adresse source traduite (publique)	Adresse IP Destination (publique)	Etat
192.168.0.3	195.220.30.60	100.0.0.12	195.220.30.60	Permanent

Ce n'est pas plus compliqué que ça. En revanche, il faut bien prendre garde à deux choses :

1. l'adresse IP publique attribuée au client 3 (100.0.0.12) par le composant NAT doit être exclusive : il ne faut pas qu'il y ait d'autres règles qui redirigent d'autres flux vers elle. D'où l'intérêt d'utiliser une application d'administration qui contrôle la pertinence de la configuration. (Voir les « *technologies disponibles* » en partie 1 ci-dessus)
2. pour que ça fonctionne, il faut que le client 3 ait toujours la même adresse IP privée (192.168.0.3). Cela peut se faire de deux manières : soit son adresse est fixée de façon permanente par l'administrateur « à la mano » (cas des serveurs par ex.) ; soit son adresse est attribuée dynamiquement par un serveur DHCP. Dans ce cas il faut que le serveur DHCP soit paramétré pour reconnaître le client <sup>37</sup> et lui réserve systématiquement un bail pour l'adresse IP privée convenue.

---

<sup>7</sup> Le serveur DHCP peut reconnaître le client 3 par son adresse MAC



Derrière ce mécanisme, même simple, surgissent plusieurs questions de sécurité. Je vous laisse y réfléchir. Ces points sortent du périmètre de ce document, elles seront abordées plus loin dans le cursus.

## La translation par port (NATP)

Mais alors comment faire quand plusieurs clients (1 et 2 dans le TP) partagent la même adresse publique ?

On ne peut pas « surcharger » les protocoles existants (IP, TCP), ni profiter de champs « inutilisés », il n'y en a pas ou sont trop petits.

Ce qu'il faut comprendre en préalable :

**Une « connexion » entre deux extrémités peut être identifiée de façon unique par le quadruplé suivant :**

( (Adresse IP_Source, Port_Source), (Adresse IP_Destination, Port_Destination) )
--

Pour faire son travail le composant NAT va utiliser cette propriété :

En plus de traduire les adresses IP Publiques/Privées, le composant va attribuer un n° de port source unique (pris dans une liste interne) à chaque connexion, ce qui lui permettra de retrouver ses petits quand il va recevoir des données de l'extérieur.

Pour ne pas être perdu dans la translation, le composant a besoin d'une table plus élaborée...

## La table d'état d'un composant NAT

On comprend (j'espère !) qu'avec cette nouvelle gestion, ça fonctionne dans tous les cas grâce à l'unicité du quadruplet ci-dessus, même si les clients consultent le même serveur, ou même si un client a plusieurs sessions ouvertes simultanément sur un même serveur.

IP source (privée)	Port source	IP Dest. (publique)	Port dest .	IP source traduite (publique)	Port source traduit	IP Dest (publique)	Port dest (inchangé)	Etat
192.168.0.3	x	194.220.6.250	Y	100.0.0.12	x	194.220.6.250	Y	Permanent
192.168.0.1	6731	199.5.9.56	80	100.0.0.13	2000	199.5.9.56	80	Temporaire
192.168.0.2	5543	195.220.30.60	80	100.0.0.13	2001	195.220.30.60	80	Temporaire

Oui mais alors dans ces cas dynamiques :

*Combien de temps le composant conserve t-il l'entrée de la table correspondante ?*

C'est une bonne question, je vous remercie. On ne peut pas conserver ces entrées indéfiniment : la table n'a pas une taille illimitée, et cela laisserait ouverts trop de trous de sécurité.

Il y a alors deux possibilités : soit l'échange de bout en bout entre le client et son correspondant suit un protocole à état (comme dans TCP, il y a une notion de session), soit ce n'est pas le cas (UDP ou ICMP par exemple).

- Dans le cas d'un protocole à état, on peut alors limiter la durée de vie de l'entrée dans la table à celle de la session applicative. Encore faut-il que le composant NAT soit capable de reconnaître et d'interpréter ce type de protocole, par exemple TCP.
- Dans les autres cas, il est indiqué (RFC) que l'on doit positionner un timer sur inactivité sur l'entrée. Les RFC ne sont pas précises sur l'implémentation, en pratique ce timer peut être de l'ordre de 4 à 10 minutes. Le TP n°3 permet d'observer la table et le cycle de vie des connexions.

Au final, la table ressemble à ça :

IP source (privée)	Port source	IP Dest. (publique)	Port dest.	IP source traduite (publique)	Port source traduit	IP Dest (publique)	Port dest	Proto	Etat
192.168.0.3	x	194.220.6.250	Y	100.0.0.12	x	194.220.6.250	Y	*	Permanent
192.168.0.1	6731	199.5.9.56	80	100.0.0.13	2000	199.5.9.56	80	TCP	<i>connecté</i>
192.168.0.2	5543	195.220.30.60	30123	100.0.0.13	2001	195.220.30.60	30123	UDP	<i>10'</i>

En résumé :

- Le composant NAT, pour être efficace doit interpréter les données contenues dans les datagrammes IP et notamment reconnaître l'état de connexion, quand il existe, des sessions utilisateurs (qualité « full-state » d'un composant).

Au-delà des questions de performances (qui ne sont plus une difficulté en pratique aujourd'hui), se pose donc le problème de la reconnaissance en profondeur des données qui transitent par un composant NAT. En effet, si certains protocoles sont bien connus (TCP) et stables, d'autres dérogent au principe d'indépendance entre couches de communication en utilisant des adresses réseau ou des numéros de port au niveau applicatif (FTP par exemple), ou bien sont des protocoles récents ou non normalisés (BiTorrent, Flexlm).

C'est une des difficultés dénombrées dans la question générale de la « **traversée du NAT** », objet du chapitre suivant.

## **La translation de port source déterministe (TP n°3)**

Avec le mécanisme de **NATP**, on a vu (TP n°3) que le choix d'un port source traduit devrait<sup>8</sup> être régulé par une approche déterministe, re-calculable. Les raisons de ce choix proviennent de plusieurs besoins :

- D'abord, pour « dérégler » le moins possible le fonctionnement de services (souvent anciens) qui ne fonctionnent que sur des ports fixes, notamment des ports réservés (1...1024) ;
- Ce déterminisme (même partiel), est également utile aux opérateurs de réseaux d'accès, pour qui cette approche revêt deux intérêts : celui de la qualité de service, et celui de la sécurité. Ce sujet est développé plus loin dans ce document.

## **TP n°2 - Les connexions entrantes**

Ce sujet s'applique lorsque l'initiative de la connexion provient de l'extérieur du système. C'est typiquement le cas quand un client veut atteindre depuis l'extérieur un serveur http, dns (TP n°2) ... qui est dans l'espace privé.

Important :

Il s'agit bien de l'initiative de la connexion : quand elle est établie (une entrée est créée dans la table), la translation se fait de façon bidirectionnelle, pendant un temps déterminé (défini dans l'entrée de la table).

La première question est la suivante : comment une machine *X1* de l'espace privé peut-elle être atteinte par une machine *Y1* de l'extérieur ? En effet :

- La machine *X1* n'a pas d'adresse publique ;
- Les seules adresses joignables depuis *Y1* sont les adresses publiques définies dans les interfaces externes du composant NAT.

En résumé, il y a deux cas :

1. Le premier cas simple d'une translation statique : on a attribué de façon permanente à *X1* une adresse publique *Pub1* exclusive, et donc *Y1* peut contacter *X1* via *Pub1*. Le composant NAT ne fait que traduire les adresses dans les deux sens, sans changer autre chose (les ports par ex.).
2. En cas d'adresse publique partagée (*Pub2*), *Y1* ne peut s'adresser à *X1* que par *Pub2*. Comment alors reconnaître les connexions qui visent *X1* plutôt qu'un autre serveur (*X2*, ...) ? La plupart du temps, on souhaite que n'importe quel *Y* puisse atteindre *X1*, donc l'adresse publique IP source *Y1* ne suffit pas à résoudre la demande ; idem pour le port source *Y1'*.  
Il ne reste plus que le port de destination *P* pour déterminer la destination finale. On appelle ce procédé la « **redirection de port** ».

---

<sup>8</sup> RFC 4787 pour UDP, 5382 pour TCP

### Précisions sur la **redirection de ports** :

Prenons l'exemple classique de la mise à disposition d'un site web pour des clients extérieurs (dans le TP n°2, vous avez fait la même chose, mais pour un serveur DNS)

- On installe un serveur http sur la machine interne *X1* ;
- Le serveur http écoute sur le port 80 de la machine *X1* ;
- L'adresse publique *Pub2* est partagée ;
- On crée une règle sur le composant NAT qui va signifier :
  - Accepter les connexions entrantes sur *Pub2* avec un port de destination égal à 80 ;
  - Rediriger ces connexions vers le serveur *X1*, c'est-à-dire traduire les paquets entrants :
$$(Y,Y') (Pub2,80) \rightarrow (Y,Y') (X1,80)$$
  - Dans l'autre sens traduire la réponse de *X1* à *Y* :
$$(X,80) (Y,Y') \rightarrow (Pub2,80) (Y,Y')$$

⇒ Point d'attention : pour créer la règle, il faut faire attention à activer les capacités « state full » du composant en lui indiquant le service utilisé (http, dns...) et ne pas se contenter d'accepter sans restriction une communication bidirectionnelle, ce qui créerait un trou de sécurité.

### Limites de la **redirection de port** :

On a vu que l'on pouvait rediriger toutes les connexions entrantes sur un port donné (autrement dit pour un port standard, vers une application : dns, http, ftp...) vers une machine désignée (par son adresse privée). Mais que fait-on quand on a besoin de plusieurs serveurs privés qui répondent au même service ? par exemple deux serveurs web ?

La réponse est : ça dépend du service.

Pour être plus précis, cela dépend des capacités de l'implémentation du service à se «multiplexer» en plusieurs instances.

Prenons l'exemple des serveurs **http** :

Si l'on veut mettre à disposition plusieurs « serveurs Web » derrière un **NATP** avec une **redirection de port** (le 80 en l'occurrence), il y a plusieurs solutions, par exemple :

- Sur une même machine, on peut créer des serveur virtuels (« *Virtual Hosts* ») pour Apache et Nginx) qui hébergeront plusieurs sites web différenciés par le champ « *Host* » de la requête http (projet NE302 !) ;
- Pour des architectures plus exigeantes, on peut installer un reverse-proxy qui se chargera de rediriger les requêtes http vers plusieurs serveurs différents.

En revanche, ce « multiplexage » de services n'est pas toujours aussi simple que pour http. Il n'a pas toujours de sens au niveau fonctionnel, et il est même parfois techniquement impossible. Il faut étudier ça au cas par cas.

## **TP n°3 - Les connexions entrantes- La table de NAT**

Le TP n°3 permet d'observer sur une linux-box avec **Netfilter** la table de NAT et le cycle de vie des connexions dans le cas d'UDP. Il permet notamment d'observer l'implémentation des règles<sup>9</sup> à propos de l'attribution des ports substitués par le composant NAT aux ports d'origine. La raison d'être de ces règles et les conséquences qu'elles emportent seront développées plus loin (sécurité et facteurs d'échelle)

### **La traversée du NAT**

Terminologie : Dans ce document, la « **Traversée du NAT** » (« NAT traversal ») désigne l'ensemble des questions qui se posent pour qu'un service existant puisse fonctionner quand de la translation d'adresses (et de ports) intervient.

Nous nous concentrerons sur la traversée des applications. Il y a d'autres difficultés, notamment au niveau réseau dans certains cas de figure, que nous allons nous contenter d'évoquer rapidement :

#### **Difficultés réseaux**

- Recouvrement des plans d'adressages

Quand on interconnecte deux réseaux privés (situation survenant généralement suite à une fusion de réseaux), il est fréquent qu'il y ait un recouvrement des deux plans d'adressages privés (c'est logique, les adresses privées sont toujours les 10.0.0... ou 192.168...). Concrètement on peut trouver deux machines ayant la même adresse sur les deux réseaux.

- Liaisons chiffrées

Dans certains cas, des réseaux sont reliés par des liaisons chiffrées au niveau 3 (**IPsec**). Si le composant NAT se trouve sur ce type de liaison, il ne pourra pas fonctionner : il ne peut pas déchiffrer les datagrammes IP, et donc faire de la translation, c'est logique.

#### **Traversée des applications**

- Les applications « sales »

On qualifie une application de « sale » quand cette dernière ne respect pas le principe d'indépendance des couches de communications et par conséquent peut troubler le principe de bout en bout. C'est un cas qui est assez fréquent, notamment quand les applications utilisent des éléments de niveau 3 (adresses réseau) ou 4 (transport) au sein de la couche 7, c'est-à-dire par l'application elle-même.

ICMP fait partie de ce cas de figure, mais c'est un service particulier utilisé pour le contrôle du réseau. A ce titre il est traité spécifiquement par les composants NAT (détails hors de portée de ce document).

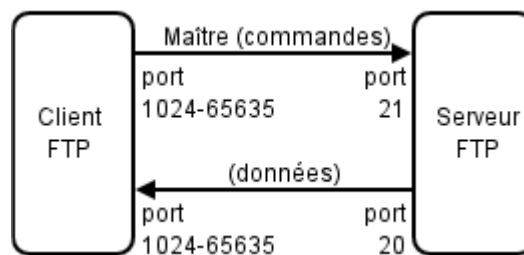
---

<sup>9</sup> RFC 4787 pour UDP, il existe la RFC 5382 pour TCP

Parmi les applications sales les plus connues, **FTP**<sup>10</sup> est un exemple classique de ce problème :

**FTP** est un service de type client/serveur qui permet de transférer des fichiers. **FTP** utilise deux canaux TCP : un pour le transfert des blocs de données du fichier, et un pour les commandes (envoyer, recevoir, start, stop...).

Dans le mode « actif » du service, c'est le client **FTP** qui détermine le port de connexion à utiliser pour permettre le transfert des données. Ainsi, pour que l'échange des données puisse se faire, le serveur **FTP** initialisera la connexion de son port de données (port 20) vers le port spécifié par le client.



Mode actif

A l'ouverture de la session, le client envoie donc au serveur dans ses données applicatives, non seulement son adresse IP (privée !), mais aussi le numéro d'un port local avec lequel il veut communiquer. Ce port étant négocié dynamiquement entre les extrémités, il ne peut pas être ouvert à priori sur les composants NAT (par une commande **Iptable**, par exemple).

Pour que cela fonctionne, il faut que le composant NAT :

1. translate, à l'intérieur de la donnée, les adresses et les ports ;
2. ouvre automatiquement les entrées de la table de NAT nécessaires.

Ce qui implique que le composant :

1. connaisse les spécifications de l'application et son format de données ;
2. sache ouvrir automatiquement des entrées.

A l'origine du NAT, ces applications « sales » étaient bien connues : **FTP, IRC-DCC, DNS, SNMP, LDAP, H.323, SIP** (téléphonie sur IP) ...

Du coup, les concepteurs ont spécifié le comportement d'un composant NAT avec ces services, en précisant deux concepts :

- Celui d'**ALG**<sup>11</sup>, qui sont des composants logiciels qui gèrent spécifiquement ces services (des sortes de plug-in) ;
- Et le « **déclenchement de port** » (« port triggering »), qui permet à l'**ALG** de demander au composant NAT de traduire et de rediriger des ports.

<sup>10</sup> File Transfer Protocol

<sup>11</sup> Application Layer Gateway

A niveau des implémentations, les composants NAT embarquent en général une batterie des **ALG** les plus connus. Il est en général difficile de savoir lesquels à priori, ce n'est que rarement décrit dans les documentations, ou difficile à trouver.

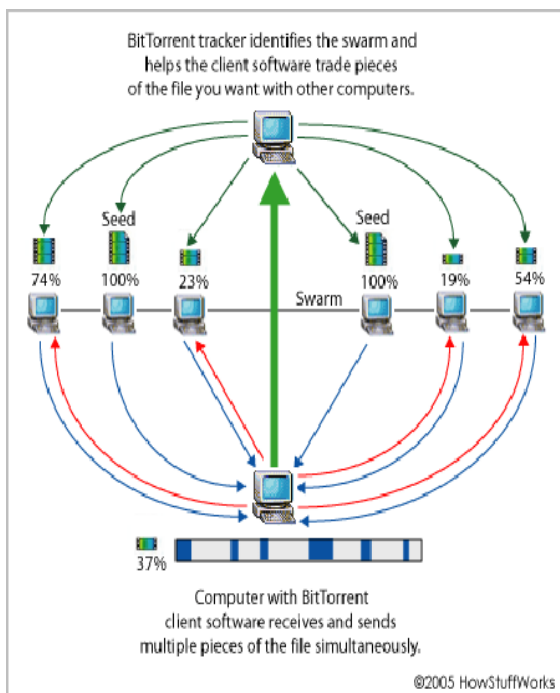
A noter que, point positif, ces mécanismes permettent également des IHM de haut niveau, avec lesquelles l'administrateur peut plus facilement gérer des règles d'accès aux services. (« J'autorise A à faire du LDAP avec B » par ex.)

- Les autres applications

Les applications précédentes sont qualifiées de « sales », mais pour autant elles sont bien connues. Elles sont souvent anciennes (mais très utilisées), stables, et leurs spécifications ne changent pratiquement pas.

*Mais alors, quid des autres applications ? Les jeux ? Les applications de téléchargement **P2P**<sup>12</sup> ? Les applications commerciales ? ; d'une façon générale, les applications nouvelles ?*

Prenons l'exemple des applications de **P2P**. Un des buts de ces applications, c'est de pouvoir récupérer sur un poste client des contenus (très souvent des fichiers multimédias : sons, images, vidéos) et en parallèle de partager des contenus avec d'autres « pairs ». D'une façon générale, ces applications fonctionnent de la manière suivante :



- Le contenu C est découpé en blocs ( $C_1, C_2 \dots C_n$ ) ; les blocs sont stockés sur les clients (les « pairs » qui participent au réseau **P2P**)
- Lorsqu'il veut récupérer un contenu C, le client se connecte à un réseau de serveurs spécialisés (généralement dénommés les « super-nœuds ») ;
- Ces serveurs lui indiquent sur quels clients « pairs »<sup>13</sup> il peut trouver tout ou partie des blocs de C, c'est à dire une liste de type :  $(C_1, IP_1, Port_1), (C_2, IP_2, Port_2) \dots (C_k, IP_k, Port_k)$  ; en espérant que  $k=n$  pour obtenir la totalité du contenu.
- Le client se connecte alors sur chacun de ses « pairs », et demande l'envoi du bloc correspondant (pour le transfert, BitTorrent utilise **MFTP**<sup>14</sup> qui est un protocole de transfert de fichiers dérivé de FTP). Le client devient alors un « leecher » ;
- En parallèle, le client C déclare au réseau de serveurs de quels contenus il dispose (même s'ils ne sont pas complets) pour les partager avec ses pairs (certains réseaux **P2P** exigent des ratios d'upload/download pour l'équité des transactions).

Pour que cela fonctionne pour un client situé derrière un composant NAT, il faut que ce dernier ouvre dans sa table de translation toutes les entrées nécessaires durant la session

<sup>12</sup> Peer To Peer

<sup>13</sup> Les « seeders »

<sup>14</sup> Multi-sources File Transmission Protocol

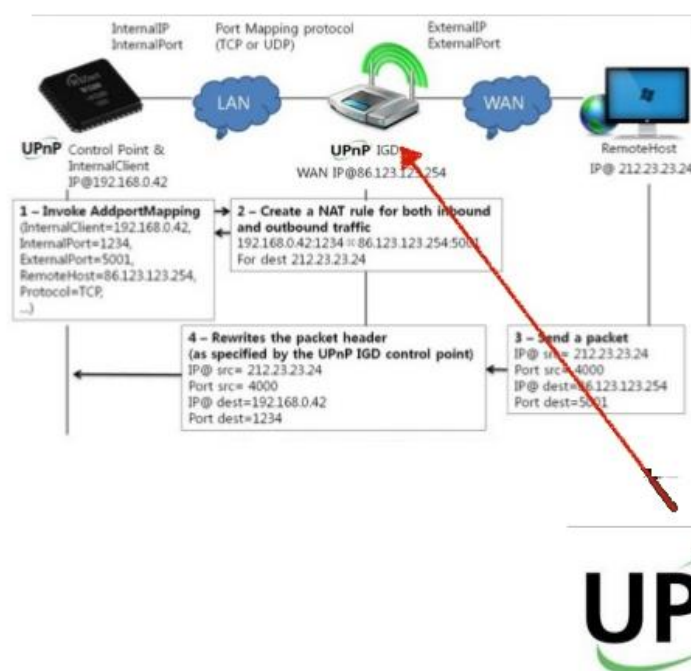


de 'partage' du client (il peut y en avoir des centaines !). Ce n'est pas possible de le faire à priori<sup>15</sup>, et il n'existe pas d'**ALG** pour ça. C'est la même chose pour certaines applications de jeux en ligne, ou par exemple pour jouer avec la Xbox.

Comment faire ? Heureusement, les ingénieurs ont de la ressource : ils ont créé des nouveaux procédés, (plus ou moins normalisés) pour pouvoir mettre à jour dynamiquement les tables des composants NAT. Le principe général est le suivant :

- Les composants NAT offrent une interface avec un service d'administration automatique qui permet la mise à jour dynamique de la table de translation ;
- Sur les systèmes qui supportent le composant NAT, on déploie en parallèle un 'mini-serveur', qui en entrée prend les demandes des clients (par exemple un client BiTorrent), et en sortie les envoie au composant NAT. Celui-ci exécute la demande, typiquement une redirection de port (et selon le cas il peut renvoyer une réponse).

Il existe deux spécifications principales qui sont utilisées par les implémentations : **NAT-PMP**<sup>16</sup>, et **UPnP**<sup>17</sup>. Ces deux mécanismes font partie du paradigme « **Zéroconf** », qui selon Wikipédia « est l'appellation générique d'un ensemble de protocoles permettant de créer automatiquement un réseau IP utilisable sans configuration particulière ou serveurs dédié ». En gros, Mme Michu branche le machin et ça marche sans rien faire d'autre. Apple aime bien ça. Ces procédés sont également très utilisés pour les applications grand public (objets connectés, domotique).



C'est la raison pour laquelle ces technos sont généralement implémentées dans les routeurs **SOHO** (les boxes) des abonnés ADSL, ce qui permet à domicile d'utiliser de nombreuses applications. En revanche, ce n'est pas le cas en général dans les systèmes

<sup>15</sup> Certains composants (freebox par ex.) permettent quand même d'ouvrir à priori une plage entière de ports pour un service BiTorrent (celui-ci n'utilise que des ports dans un range donné), mais c'est plus dangereux.

<sup>16</sup> NAT Port Mapping Protocol, RFC 6886

<sup>17</sup> Universal Plug and Play, adopté par l' UPnP Forum (standard industriel)



d'entreprise : il n'y en a pas besoin, ou bien c'est tout simplement interdit (on ne joue pas au travail !) : heureusement.

Heureusement, parce qu'il y a un problème majeur de sécurité avec ces manières de procéder :

En effet les protocoles **NAT-PMP** et **UPnP**, même si leurs utilisations sont limitées aux clients privés du réseau, ne gèrent pas l'authentification des clients.

Autrement dit : n'importe quel poste du réseau privé peut modifier les règles de NAT. Ainsi, si un client est corrompu, il peut ouvrir des « trous »<sup>18</sup> à volonté dans les flux vers l'extérieur du réseau. Il y a vraiment de quoi être inquiet : imaginez une casserole « connectée » qui, en gros se permet de faire des « **Iptables** » sur le routeur ! Malheureusement c'est bien la réalité, ce sont des failles de sécurité qui sont constatés quotidiennement. Méfiez-vous en particulier des passoires connectées<sup>19</sup>.

Il existe une dernière catégorie d'applications « sales » qui n'ont ni d'**ALG**, ni d'interfaces de type **Zéroconf**. C'est par exemple le cas de Flexlm, le gestionnaire de licence très utilisé par des logiciels commerciaux (par ex. MatLab à l'école). Dans ces cas-là, la traversée du NAT reste impossible. Il faut alors se tourner vers d'autres solutions d'architectures (basées par exemple sur l'utilisation de VPN).

Enfin, les caractéristiques de certaines infrastructures d'accès, notamment les réseaux de téléphonie mobile (voir chapitre **CGN** ci-dessous) ne permettent pas (entre autres) l'utilisation de procédés de traversée du NAT (**Zéroconf** notamment), et par conséquent certaines applications ne fonctionnent pas sur les Smartphones (**P2P** et certains jeux par exemple).

A ce jour le NAT reste un incontournable imparfait, mais qui permet d'attendre le déploiement généralisé d'IP<sub>v6</sub>. Dans cette attente, il existe une alternative au NAT actuel<sup>20</sup>, mais qui n'a pas été standardisée, en tout cas qui n'a pas déployée à ma connaissance.

---

<sup>18</sup> UDP ou TCP « punching holes »

<sup>19</sup> Voir le cours de logique des passoires du professeur Shadoko

<sup>20</sup> RFC 6346 The Address plus Port (A+P) Approach to the IPv4 Address Shortage

## Le facteur d'échelle – les opérateurs et leurs infrastructures d'accès à Internet

### Le contexte

Cette question se pose pour les organisations qui ont un très grand nombre de «clients» internes, notamment si ces clients doivent se connecter à Internet.

C'est typiquement le cas des opérateurs de téléphonie mobile : par exemple en 2020 Orange a 30 millions d'abonnés en France, et Verizon 130 millions aux US, sans compter les autres services offerts par ces opérateurs : abonnements fixes, ADSL, services d'IOT... Ces derniers usages (IOT) ne faisant qu'amplifier les difficultés car ils impliquent souvent de très grandes quantités « d'objets » connectés, même si IP<sub>v6</sub> est couramment utilisé dans ces applications.

Les principaux problèmes sont de deux ordres :

1. Le nombre d'adresses privées disponibles : en effet la RFC 1918 n'en définit pas assez pour couvrir les besoins de l'adressage privé de tous les abonnés : la plage 10/8 est limitée à 16, 7 millions d'adresses ;
2. La limite du nombre de ports sources traduits (NATP). Le champ « port » d'un segment TCP ou UDP est codé sur 2 octets, donc limité à 65.535 valeurs. Cela veut dire que pour une adresse IP publique partagée, le composant NAT ne peut allouer que 65.535 entrées dans sa table pour ses clients.

Le traitement du premier point (nombre d'adresses privées) dépasse du cadre de ce document, il n'est pas directement lié au mécanisme de NAT. Vous pouvez chercher vous-même les solutions utilisées (il en existe) si vous voulez approfondir<sup>21</sup>.

Pour illustrer le deuxième point, faisons un petit calcul avec l'exemple de l'opérateur Orange et ses 30 millions d'abonnés (mobile) :

- *Combien faut-il d'adresses publiques pour servir les abonnés d'Orange en connexions internet ?*

Pour cela il faut avoir une idée de la consommation d'un téléphone en translations d'adresses par port (NATP).

Il faut savoir qu'une « application » sur Smartphone peut consommer beaucoup de connexions simultanées. Un exemple courant est celui de *Google Maps* qui peut utiliser plus de 60 connexions simultanées (l'application *Itunes* est aussi un autre exemple).

Faisons l'hypothèse qu'un abonné utilise 5 applications qui elles-mêmes consomment 20 connexions chacune (c'est une hypothèse raisonnable, même un peu minimaliste<sup>22</sup>). Ce client a donc besoin de 100 connexions NATP simultanées.

L'opérateur Orange a donc besoin de  $30 \cdot 10^6 \times 100 = 3 \cdot 10^9$  translations par port.

Chaque adresse IP publique en permet 65.535.

---

<sup>21</sup> Indication : RFC 6598

<sup>22</sup> Voir cas des applications P2P dans le chapitre « traversée du NAT »

Donc Orange a besoin d'au moins  $(3.10^9/65.535) \approx 46.000$  adresses IP publiques.

C'est beaucoup, même pour un opérateur national : n'oublions pas que celui ci doit en plus assurer le service pour l'ADSL, l'IOT etc... Si l'on extrapole au cas de Verizon aux US on obtient  $46.000 \times 4 \approx 184.000$  adresses IP<sub>v4</sub>...

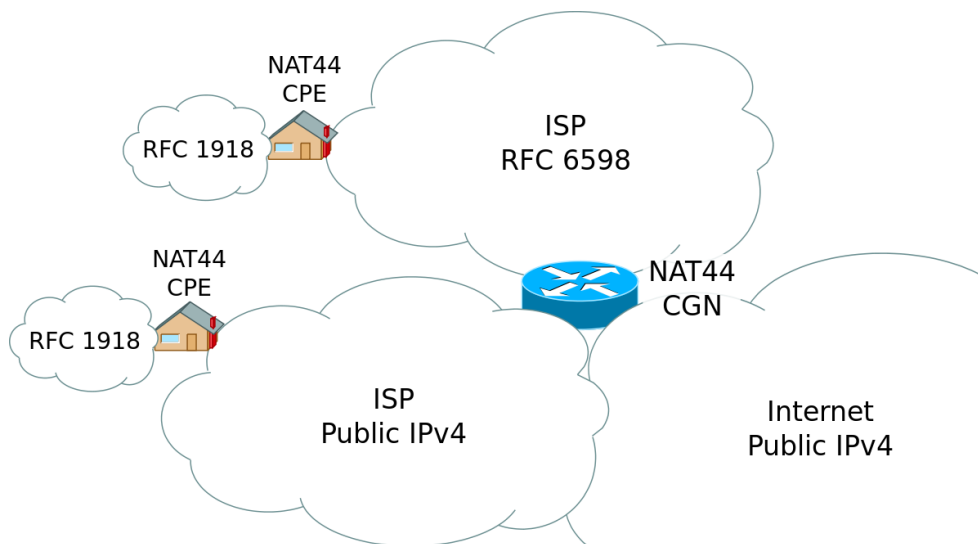
## Les impacts pour l'opérateur

On voit que le système de l'opérateur se complique avec le problème de la translation d'adresse, d'autant plus qu'il faut qu'il gère en parallèle un plan de transition vers IP<sub>v6</sub>.

- *Mais alors comment font les opérateurs pour faire face à ces problèmes ?*

*Et bien, on ne le sait pas exactement...* Les opérateurs sont assez discrets sur le sujet : c'est normal ils doivent protéger leurs infrastructures. La plupart sont d'ailleurs soumis à des réglementations nationales qui leur impose cette discrétion. En France, les opérateurs sont considérés comme des **OIV**<sup>23</sup>, à ce titre ils sont régulés par l'**ANSSI**. On a quand même une idée des procédés qu'ils utilisent, à travers quelques références académiques et d'autres sources plausibles :

1. IP<sub>v6</sub>, pour leur plan de transition et pour acheminer du trafic au sein de leur infrastructure ;
2. Du « NAT de NAT » (mots-clés : **NAT44**<sup>24</sup>, **NAT444**, **CGN**<sup>6</sup>) :



En résumé, les adresses privées des clients sont traduites en sortie une première fois par des composants NAT « proches » des clients (les box pour les abonnés ADSL) vers un plan d'adressage privé de l'opérateur (sur le schéma, RFC 6598 désigne un plan privé comme RFC 1918), puis une deuxième fois vers Internet.

A noter que cette double translation peut poser problème pour des applications qui ont besoin de connaître l'adresse IP publique du composant NAT : un client qui interroge sa

<sup>23</sup> Opérateur d'Importance Vitale

<sup>24</sup> Le '4' de NAT44 fait référence à la version d'IP. Il existe aussi le NAT64, NAT46, ...

box par exemple va obtenir en fait une adresse privée de l'opérateur (RFC 6598, range : 100.64.0.0/10), alors que s'il interroge un serveur distant (« mon-ip.com » par exemple), il obtiendra une vraie adresse publique, celle qui lui a été attribuée en sortie de l'infrastructure de son opérateur.

### 3. La translation de port source « déterministe » :

Avec le mécanisme de **NATP**, on a vu (TP n°3) que le choix d'un port source traduit devrait<sup>25</sup> être régulé par une approche déterministe, re-calculable. Pour les opérateurs, cette approche revêt deux intérêts : celui de la qualité de service, et celui de la sécurité, en termes de disponibilité et surtout de traçabilité :

- Qualité de service et disponibilité :

**Qualité** : les opérateurs sont tenus de garantir un certain niveau de qualité de service et de disponibilité pour leurs abonnés, même si ce niveau de détail ne se trouve pas explicitement dans les contrats (surtout pour les abonnements grand public). Plus concrètement, il faut pouvoir gérer des « ressources » pour les utilisateurs, dans ce cas des entrées dans les pools globaux d'adresses IP publiques et de ports sources disponibles. Par exemple, si l'on reprend le calcul ci-dessus (§ contexte), il faudrait pouvoir allouer au moins 100 ports par utilisateur, ce qui implique un procédé de contrôle global de l'attribution de ces ports. C'est plus facile si on a une stratégie déterministe (par exemple réserver un intervalle de port  $[p_1..p_2]$  toujours le même pour une adresse IP privée donnée). A noter qu'une des contreparties de cette stratégie peut être le gaspillage des ressources disponibles.

**Disponibilité** : il ne faut pas que les tables de translations soient saturées, notamment par des entrées qui ne correspondent plus à des besoins (erreurs), ou qui sont là depuis (trop) longtemps : des applications utilisateurs qui cherchent à maintenir des sessions indéfiniment par exemple. Par l'observation, on a pu constater que certains opérateurs réinitialisaient leurs tables à intervalle régulier (toutes les 24h, par ex.) pour la maintenance de leurs structures de données. Il arrive également que certaines adresses publiques soient blacklistées par certains sites, à cause d'usages inappropriés par certains usagers ; ce qui peut causer une indisponibilité temporaire de service. C'est un vrai souci pour les applications, qui voient leurs sessions coupées régulièrement sans raison apparentes. Cette particularité des réseaux mobiles avec des abonnements destinés au grand public doit être prise en compte dans la construction des applications, et ce n'est pas toujours facile. En tout état de cause, cela a un coût (qui est fréquemment oublié). Quand le contexte le permet, on peut dans cet objectif utiliser des abonnements spécialisés pour ce genre d'application dits : abonnements « **M2M**<sup>26</sup> », qui permettent d'avoir contractuellement des ressources fixes et garanties (adresses IP fixe par ex.).

---

<sup>25</sup> RFC 4787 pour UDP, 5382 pour TCP

<sup>26</sup> Machine to Machine

- **Traçabilité** :

Il faut bien voir que les traces (logs) des événements de translations d'adresses sont nécessaires pour répondre à la question centrale : « *qui-a-fait-quoi-quand ?* ». Par conséquent, le besoin de traçabilité est de plusieurs ordres : économique, techniques et surtout règlementaire. Dans ce sens, ces traces représentent à la fois des *actifs système* mais aussi des *actifs métier* pour l'opérateur, parce qu'elles ont de la valeur pour lui :

- économique : l'opérateur a besoin de ces traces pour sa comptabilité (facturation) et pour gérer sa productivité en général ;
- technique : ces logs sont nécessaires pour la *supervision* du système et pour aider à la résolution d'incidents ;
- règlementaire : dans tous les pays, avec plus ou moins d'exigences, les opérateurs sont tenus par la loi de répondre à des demandes administratives et judiciaires sur les communications de leurs abonnés. Dans notre contexte, des questions classiques qui sont posées sont par exemple : « *qui s'est connecté à tel serveur sur tel port à telle date/heure* », ou « *pour cet utilisateur particulier, enregistrer toutes ses traces de communication pendant cette période* ».

C'est un tracas pour l'opérateur, parce que ces traces peuvent être très lourdes :

- en volume : on estime<sup>27</sup> qu'un opérateur qui a un million d'abonnés a besoin (pour le NAT) de 150 téraoctets par mois de traces, soit 1,8 pétaoctets par an. C'est une approximation, mais qui donne bien l'idée de l'échelle du problème à traiter ;
- en débit : la bande passante nécessaire vers le système de log est estimée à 460 Mb/s pour 1 millions d'utilisateurs, ce qui peut avoir un impact significatif sur l'infrastructure elle-même de l'opérateur.

Le fait que l'attribution de port soit déterministe simplifie le problème (par exemple si on alloue une plage de ports à chaque utilisateur<sup>27</sup>) en réduisant les traces à enregistrer, quitte à une recherche un peu plus coûteuse quand il y en aura éventuellement besoin. En parallèle, il reste nécessaire pour l'opérateur de pouvoir enregistrer exhaustivement les traces d'un ensemble d'utilisateurs ciblés pour répondre aux demandes spécifiques, mais leur nombre restant en général faible, l'impact s'inscrit à la marge.

On peut imaginer pour finir que certains opérateurs enregistrent systématiquement toutes les traces détaillées, pour répondre à des demandes politiques de surveillances généralisées. Mais on n'est pas sûr ;- ) en tout cas, il y a alors besoin de très gros disques.

---

<sup>27</sup> RFC 7422 Deterministic Address Mapping to Reduce Logging in Carrier-Grade NAT Deployments

## La sécurité

Il y a un débat entre experts sur les vices et vertus de la translation d'adresses en matière de sécurité. Un des impacts du NAT qui est valorisé positivement, c'est le masquage du plan d'adressage privé. Un observateur extérieur ne peut en effet pas identifier un host interne par son adresse réseau. C'est un bon point, mais à mitiger parce qu'il existe bien d'autres façons d'identifier des systèmes (« fingerprinting », empreintes temporelles...).

Si l'on considère l'ensemble des conséquences de la translation d'adresse : ce qui a été vu auparavant : filtrage, traversée, effet d'échelle..., le bilan en termes de sécurité est au final important pour les systèmes :

- Perméabilité : les procédés de translations dynamiques permettent la création de trous (même « petits ») dans la gestion des flux réseau entre l'intérieur et l'extérieur d'un système. Ces vulnérabilités sont souvent difficiles à identifier à cause du nombre d'implémentations différentes ; le « summum » des risques étant atteint dans les cas d'usages des systèmes domestiques<sup>17</sup>. Ces faiblesses sont par ailleurs difficiles à mitiger directement par des mesures de sécurité ;
- Disponibilité : comme on l'a vu dans la première partie, les composants NAT constituent généralement des **SPOF**<sup>2</sup> du système, à ce titre ils doivent faire l'objet d'une attention particulière dans la conception des architectures. Pour les fournisseurs d'accès (voir chapitre sur les opérateurs), le NAT a un impact très fort sur les exigences de qualité de service ;
- Traçabilité : comme elles modifient des données importantes du système : sources, destinations, services utilisés..., les opérations de translations doivent être tracées pour répondre aux exigences réglementaires et pour la gestion technique du système, même si d'autres traces peuvent être générées par d'autres composants (proxies, firewalls...) dans le même objectif. Avec le facteur d'échelle, cette exigence peut avoir des impacts forts sur le système et le réseau interne ;
- Complexité : l'ensemble des dispositifs à concevoir, déployer et exploiter pour gérer le NAT augmente la complexité des systèmes. De la conception des architectures aux applications et au maintien en conditions opérationnelles, le besoin est marquant en ressources humaines (compétences, disponibilité) et en gestion de processus (configuration, déploiement, supervision). Ces points entraînent des coûts pour l'organisation qui ne sont pas toujours bien assumés. Ajouté à la dimension humaine de la question, cette complexité crée au final de nouvelles vulnérabilités significatives pour l'organisation.

En bilan des questions de sécurité, on peut dire que l'usage du NAT amène pour un système des vulnérabilités supplémentaires de tous ordres, y compris non techniques (humaines et organisationnelles).

Selon la posture dans laquelle on est placé, on peut voir dans ces faiblesses une source forte de préoccupation pour protéger les systèmes, ou bien un ensemble d'opportunités intéressantes pour des actions offensives.



## Synthèse et conséquences pour les ingénieurs de l'école

### *Exercice n°1 - Réflexion sur les métiers*

L'idée de cet exercice, c'est d'essayer de voir quels impacts concrets a le sujet étudié sur le travail d'un ingénieur de l'école, qu'il soit en cours de formation lors des projets industriels, ou bien surtout dans ses prochains postes et missions en entreprise.

### **Indications**

- Identification des métiers et missions :  
Pour les projets industriels, vous pouvez vous faire une idée des sujets traités sur l'intranet ; pour approfondir il faut demander. Pour les postes et missions, vous pouvez consulter des référentiels de la profession ; en France, il y a la nomenclature des métiers du SI du **CIGREF**, ou plus spécialisé, le référentiel de l'**ANSSI** pour les métiers de la cybersécurité.  
Vous pouvez regarder aussi ce que font aujourd'hui les anciens élèves, à travers les réseaux ou avec vos connaissances personnelles.  
Au final, vous devriez obtenir quelques catégories de profils du type : ingénieur système et réseaux ; ingénieur développement logiciel, architecte système, expert en cybersécurité... et une description de leurs missions respectives et des problèmes auxquels ils sont confrontés.
- Impacts :  
A partir de la liste précédente, il faut identifier en quoi le sujet (le NAT) a une importance pour l'ingénieur dans ses missions. C'est par exemple évident dans le cas d'un ingénieur qui travaille pour un fabricant ou un intégrateur de composants réseaux, mais pour d'autres cas (postes ou projets) il faut réfléchir.



## Partie 3 : Exemples de cas d'usages (sortant) :

*A suivre dans une version ultérieure de ce document*

### **Depuis des PC derrière une box opérateur **SOHO**<sup>28</sup>**

Thème :

- Des PC fixes connectés à une box qui :
  - se connectent à des sites web externes
  - partagent des fichiers en Peer-to-Peer (P2P) avec des PC extérieurs eux-mêmes derrière une infrastructure privée

### **Connexion à un site web externe depuis un PC de l'école**

Thème :

- Des PC fixes connectés au réseau local de l'école (salles de TP par ex.)

### **Connexion d'une application sur Smartphone à un service sur internet**

Thème :

- Des smartphones qui utilisent des applications web. Depuis une liaison mobile (4G), depuis une connexion wifi. Les deux.

---

<sup>28</sup> Small Office and Home Office



# Bibliographie

## *Exercice n°2 - Réalisation d'une bibliographie*

L'objectif de cet exercice est de **réaliser une bibliographie propre et utile** pour les sujets traités dans le texte.

### *Indications*

- **Propre :**  
Sur la forme, vous devez vous inspirer de ce qui existe déjà en vous inspirant de publications dans le monde académique ou le domaine technique. Il existe également de nombreux tutoriels sur le sujet, de standards (APA<sup>29</sup> par exemple), et d'outils pour réaliser une « bonne » bibliographie.  
En aucun cas, il ne faut faire une liste d'URL dénommée parfois une « webographie », incompréhensible et inexploitable. Webographie = bullshit.
- **Utile :**  
Les lecteurs ciblés par le texte sont des apprenants dans le domaine des systèmes qui possèdent des prérequis de base sur le sujet ; typiquement des élèves ingénieurs en cours d'apprentissage. Le texte peut intéresser également des ingénieurs confirmés mais qui ne sont pas des spécialistes de la question.

Les références doivent être utiles pour les lecteurs visés, afin de :

- comprendre : références générales, synthèses... ;
- apprendre : tutos, schémas, exercices, études de cas... ;
- approfondir : articles, expertises, normes... ; et
- élargir les connaissances : ouverture sur des sujets connexes au texte.

---

<sup>29</sup> Le style « APA » est un format éditorial défini par l'American Psychological Association