Séminaire Cr@ns
Routage et pare-feu sous Linux avec `NFTables` et `ip`

DsAc

Jeudi 2 décembre 2021

1 Préambule

L'objectif de ce séminaire est de discuter de mise en place d'un pare-feu basique sous Linux. Pour cela, rappelons nous comment votre ordinateur communique avec les autres dans les couches trois et quatre du modèle OSI :

7	Application
6	Pas utile ici
5	Pas utile ici
4	Transport (UDP, TCP)
3	Réseau (IP)
2	Liaison (Adressage physique MAC))
1	Physique (e- dans un câble de cuivre, photons dans une fibre, ...)

Figure 1 – Modèle OSI simplifié

Remarque :

Le TCP et l'UDP sont des protocoles bâtis au-dessus de la couche réseau permettant de transmettre des données sur Internet. Leur détail n'est pas difficile, mais pas nécessaire dans le cadre de ce séminaire. Il est possible de voir le TCP comme une lettre suivie —établissant une connexion stable et conservant l'ordre des envois— et l'UDP comme une carte postale.

Dans la couche IP, les différentes machines d'un réseau sont représentées par des adresses IPs. Lors de ce séminaire, je parlerai d'IPv4, mais le fonctionnement que je présente est valable en IPv6.

Lorsqu'un paquet (couche 4) est envoyé sur Internet, votre machine se pose deux questions :

1. Est-ce que je possède l'IP que l'on cherche à joindre ?
2. Où transférer le paquet pour le rapprocher de sa destination ?
   Pour la plupart de vos machines, cette seconde question se reformule ainsi :
   i Est-ce qu'il s'agit d'une IP du même sous-réseau que moi (auquel cas, je peux l'envoyer au réseau local et elle arrivera sur la bonne machine) ?
   ii Autrement, quelle est la passerelle à utiliser (typiquement la box Internet de votre fournisseur d'accès) ?

Pour visualiser les routes que votre ordinateur connait, n'hésitez pas à regarder la sortie de la commande `ip route`. Dans mon cas, j'observe :

— `default via` *`192.168.1.4`* `dev` *`wlp1s0`* `proto dhcp metric 600`
Cette route m'indique que si aucune route n'est connue pour le paquet traité, il sera envoyé par *192.168.1.4*, qui est ici l'adresse de la box fournie par mon FAI.

— *`192.168.1.0/24`* `dev` *`wlp1s0`* `proto kernel scope link src` *`192.168.1.124`* `metric 600`
Cette ligne m'indique que lorsque je cherche à joindre une IP contenue dans l'intervalle `192.168.1.0/24`, le paquet est envoyé par ma carte wifi —dont l'interface s'appelle *wlp1s0*— et que la manière de joindre l'appareil est de passer par la couche «`link`» (couche 2).

La différence entre cet exemple et le cas d'un vrai routeur est le nombre de route. En effet, un routeur possède toutes les routes d'Internet. L'idée est que pour toute adresse, le routeur doit choisir de faire transiter le paquet par son voisin le plus proche de la destination. Ainsi, Internet fonctionne en faisant faire de petits bonds aux paquets, chaque bond rapprochant le paquet de sa destination.

Pour visualiser cela, vous pouvez effectuer un `traceroute` en utilisant l'utilitaire éponyme ou l'utilitaire `mtr` (*e.g.* `mtr crans.org`).

Remarque :

Si plusieurs routes sont en conflit, celle qui correspond au préfixe commun le plus long de l'adresse de destination est utilisée. Par exemple, s'il faut router 1.1.1.1 et que nous connaissons des règles pour 1.1.1.0/24 et 1.1.1.0/26, la seconde règle sera appliquée. Si vous avez besoin de plus d'information sur les masques tel que `/24`, la page Wikipédia intitulée «Sous-réseau» peut vous être utile.

2 Présentation des commandes utilisées dans ce séminaire

2.1 Présentation de `NFTables`

2.1.1 Présentation générale

`NFTables` est un outil permettant de modifier le traitement des paquets réseau par votre noyau et dont le support est présent dans les noyaux Linux modernes. Du point de vue de `NFTables`, les paquets vus par votre machine suivent le cheminement suivant :

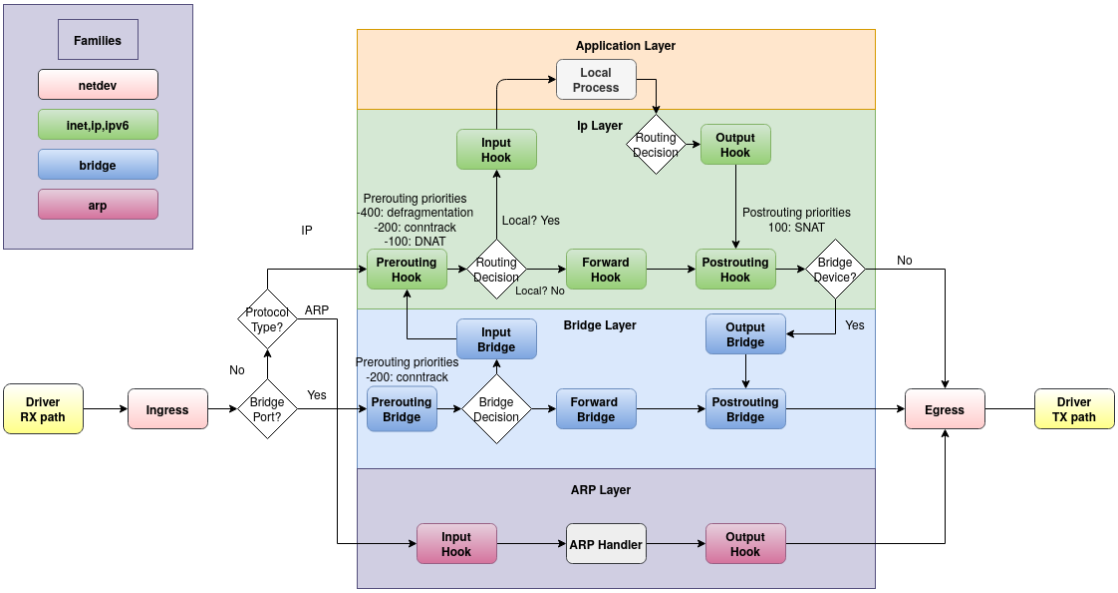


FIGURE 2 – Traitement des paquets par `NFTables`

Ce schéma dépasse de loin les objectifs de ce séminaire. Seul l'étage vert (Network Layer) nous intéresse aujourd'hui.

Une version plus précise de ce schéma est :

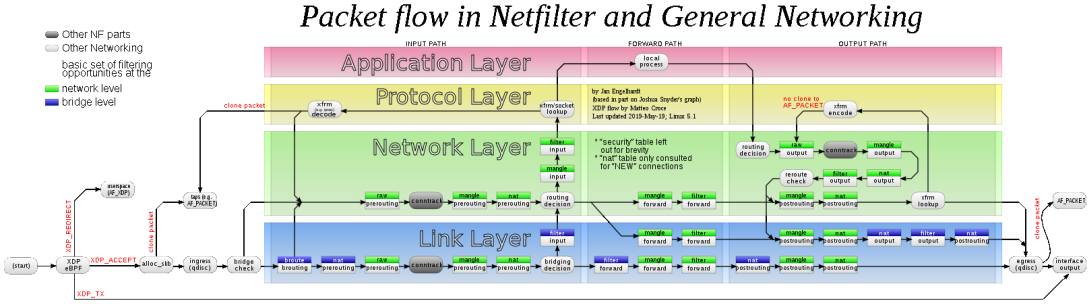


FIGURE 3 – Traitement des paquets par `NFTables`

Sur ce schéma, chaque case représente un crochet d'exécution de `NFTables`, *i.e.* une étape dans le traitement des paquets. Il en existe de plusieurs sortes :

filter Permet de filtrer les paquets (*Cf.* section sur le pare-feu)

nat permet de natter les paquets (*Cf.* section sur le routage)

mangle non utilisé ici (permet notamment de changer les ports des paquets)

raw non utilisé ici

2.1.2 Structure de `NFTables`

`NFTables` examine chaque paquet passant par, sortant de ou arrivant sur votre machine. L'examen des paquets est effectué par application de règles ordonnées dans des tables.

Ainsi, il existe les objets – chacun des objets présentés étant hiérarchiquement inclus dans dans le précédent- — suivants dans le système de `NFTables` :

tables Une table est définie pour un type de protocole (`ip`, `ipv6`, `inet` [1] ou encore `arp`).

Chaque table se voit donner un nom.

chaînes Une chaîne correspond à une case du second graphique précédent, *i.e.* à un crochet d'exécution de `NFTables` et à un type de crochet (`filter`, `mangle`, `nat`, `raw`).

Chaque chaîne se voit donner un nom.

règles Une règle correspond à une succession de tests caractérisant les paquets concernés. Dans une règle peuvent également se trouver certaines actions telles que le comptage des paquets concernés et un verdict lié au paquet.

Un verdict correspond à l'acceptation ou rejet des paquets, ou encore à un saut vers une autre chaîne.

La commande `nft` permet d'interagir avec `NFTables`. Il est également possible d'écrire un fichier de configuration complet et de demander à `NFTables` de le filtrer.

Objet à créer	**Commande**
Table	`nft add table inet basse` Crée une table «basse»

Chaîne	`nft add chain inet basse HiFi`
	`'{ type filter hook input priority filter}'`
	Crée une chaîne pour le filtre des paquets entrant dans la table `basse`

Règle	`nft add rule inet basse HiFi 'tcp dport 53 drop'` Crée une règle pour rejeter les connexions tcp à destination du port 53

Il est pratique de pouvoir interagir avec `NFTables` par l'emploi de telles commandes lorsqu'on l'utilise dans des programmes (des interfaces existent en C et Python par exemple). Néanmoins, lorsqu'il s'agit d'une configuration initiale ou d'un pare-feu statique, par exemple dans le cas d'un ordinateur de bureau, il est coutume de placer cette configuration dans un fichier `/etc/nftables.conf`, et d'activer le service `nftables` [2].

Pour la configuration précédente, le fichier `nftables.conf` aurait la forme suivante :

```
flush ruleset # permet de retirer toute règle préexistante
              #(utile si le fichier est lu plusieurs fois d'affilée)
table basse {
    chain HiFi {
        type filter hook input priority filter;
        tcp dport 53 drop
    }
}
```

Les conditions que l'on peut utiliser dans `NFTables` sont multiples, le principe général étant de vérifier si un «type» a une certaine «valeur». Par exemple, plus haut, `tcp dport` est un «type», dont les valeurs sont des entiers sur 16 bits (les ports). Lorsque l'on connait un type,

1. `inet` représente pour `NFTables` l'union des IPv4 et des IPv6.
2. Par exemple, si vous utilisez `systemd`, «`systemctl enable nftables.service`», et si vous utilisez `runit`, «`sv enable nftables`»

savoir quelles valeurs sont acceptables est possible en observant la sortie de la commande `nft describe <type>`.

Les principaux tests sont effectués sur :

— Le protocole de couche 4 avec `ip protocol` (ou `meta l4proto` si l'on souhaite inclure l'IPv6)

— Les ports de destination (`tcp dport` en cas de paquets TCP, `udp dport`, `th dport` si le protocole de couche 4 n'importe pas)

— Les ports source (`tcp sport` en cas de paquets TCP, `udp sport`, `th sport` si le protocole de couche 4 n'importe pas)

— L'état de la connexion `ct state`. Les états intéressants sont :

↪ `new` pour une nouvelle connexion

↪ `established` pour une connexion déjà établie (en TCP notamment)

↪ `related` pour un paquet relié à des paquets déjà émis ou reçus (en UDP notamment)

Les actions principales sont :

— `counter` compte les paquets atteignant ce point de la règle

— `accept` accepte un paquet (crochet de type filtre)

— `drop` jette un paquet (crochet de type filtre)

— `nat` permet d'effectuer des translation d'adresses (crochet de type nat)

Remarque :

L'action `meta nftrace set 1` permet un débogage détaillé. Il est par la suite possible d'exécuter `nft monitor` pour voir les en-tête des paquets ciblés par cette action.

Remarque :

`NFTables` permet d'effectuer bien plus d'autres choses. Il est notamment possible d'utiliser des variables dans un fichier de configuration, de définir des ensembles et des mappings (injections partielles) ou des compteurs nommés. Notons qu'il est possible également d'associer une durée à la présence d'éléments dans un ensemble ou mapping (particulièrement utile dans les cas de blacklist, whitelist ou port knocking).

2.2 Présentation de `ip`

La commande `ip` est bien plus simple que `nft`. Il s'agit d'une commande dont la syntaxe est `ip <flags> <sous-commande> <option>`. Les `<flags>` représentent des options générales sur `ip`, tel que `-c` permettant de colorer les sorties des commandes.

Les sous commandes comprennent :

link permettant de gérer les interfaces réseau de votre machine

address permettant de gérer les adresses IPs de vos interfaces

route permettant de gérer les routes utilisées par votre noyau

neigh permettant de voir vos voisins (connus de votre noyau)

Tout préfixe du nom d'une sous commande d'`ip` permet de désigner cette dite sous commande. Par exemple, `ip a` et `ip address` désignent toutes deux la même sous commande. Il y a une notion de priorité au sein de sous commandes afin qu'il n'y ait jamais de conflit. Par exemple, `ip r` désigne `ip route` et non pas `ip rule`.

Remarque :

Je vous conseille d'utiliser la variable d'environnement `COLORFGBG` permettant de décrire les couleurs du texte et d'arrière plan de votre terminal à `ip` afin de rendre le texte plus simple à lire, par exemple en exécutant `export COLORFGBG="7;0"` avant d'exécuter `ip`. Comparez alors la sortie de `ip -c a` avant et après cela. L'alias `ip='ip -c'` peut également vous aider à y voir plus clair à l'avenir.

Sans argument, une sous commande d'`ip` permet de voir le statut de la commande associée. Par exemple, `ip a` permet de lister des informations sur vos interfaces, dont vos adresses ip.

3 Pare-feu avec `NFTables`

Le principe d'un pare-feu est de filtrer les paquets entrant et sortant de votre machine. D'après le schéma donné en présentation, les crochets utilisé pour cet usage sont :

— le crochet de filtre en entrée (`input, filter`)
— le crochet de filtre en sortie (`output, filter`)

Ainsi, un fichier de configuration de base ressemble à :

```
flush ruleset
table inet filter {
    chain input {
        type filter hook input priority filter;
        # Règles liées aux paquets entrant.
    }
    chain output {
        type filter hook output priority filter;
        # Règles liées aux paquets sortant.
    }
}
```

Les règles mises en place en général sont :

— Autoriser tout paquet à sortir. Cela présuppose que l'on fait confiance aux applications installées sur la machine
— Autoriser les connexions entrantes liées à des connexions déjà acceptées
— Autoriser les paquets `icmp` et `icmpv6` entrant
— Refuser tous les autres paquet entrant

Remarque :

Par défaut, `NFTables` accepte tout paquet. Ainsi, si aucune règle ne concerne un crochet, les paquets sont intouchés et tous acceptés.

Ainsi, nous obtenons le pare-feu suivant :

```
flush ruleset
table inet filter {
    chain input {
        type filter hook input priority filter; policy drop;
        ct state { related, established } accept
        meta l4proto { icmp, icmpv6 } accept
    }
}
```

4 Routage et NAT avec `NFTables` et `ip`

4.1 Routage

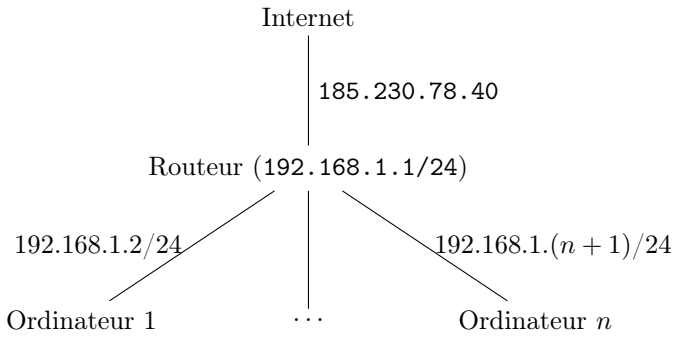
Le routage consiste à :
— autoriser des paquets à traverser votre machine quand bien même elle n'est ni destinataire ni émettrice de ces derniers,
— choisir où envoyer les paquets transférés par votre machine.

Le choix de la destination où envoyer les paquets est effectué par le noyau. Pour cela, il choisit une route parmi décrites par `ip route` comme décrit plus haut. Il existe plusieurs protocoles d'échanges de routes entre routeurs, l'un des plus utilisé étant BGP, qui vous sera présenté dans un séminaire ultérieur.

4.2 NAT

Il n'y a pas assez d'IPv4 dans le monde. En effet, il n'en existe que $2^{32} \approx 4$ milliards d'adresses. Pour pallier à cette pénnurie, il est parfois utile de mettre en place des IPs privées ou locales : les mêmes IPs peuvent être utilisées dans plusieurs contextes privés sans conflit. Il faut néanmoins que les paquets puissent sortir de ce réseau interne si les machines doivent aller sur Internet, ou relier deux réseaux privés. Comme les IPs utilisées ne sont pas publiques, il n'est pas possible pour le routeur de transmettre les paquets intouchés. En effet, le serveur distant ne peux pas accéder à l'adresse émettrice du paquet, celle-ci étant privée au réseau de provenance.

Le routeur va modifier l'adresse émettrice `A` et la remplacer par sa propre adresse IP, laquelle est soit publique, soit accessible par le routeur suivant. Cela est appelé `sNAT` (abrégé de source Network Address Translation). Lorsque cela est fait, le noyau sait que les réponses doivent être modifiées également : l'adresse de destination doit être changée, de l'adresse du routeur et celle de la machine `A`.



Mise en place du NAT L'idée ici est pour le routeur d'autoriser les redirections de paquets, ajouter les adresses locales et ajouter les routes par défaut des ordinateurs «Ordinateur 1», ..., «Ordinateur n».

Plus précisément :
— Le routeur autorise la retransmission de paquets par la commande `sysctl net.ipv4.ip_forward=1`. Cette commande autorise la retransmission sur toutes les interfaces. Pour un contrôle plus précis, utiliser les commandes `sysctl net.ipv4.conf.<interface name>.forwarding=1`.
— Le routeur attribue l'adresse `192.168.1.1/24` à son interface liée au réseau privé.

— Attribution des adresses IPs
  ↪ L'«Ordinateur 1» s'attribue l'adresse `192.168.1.2/24` à l'aide de la commande `ip address add 192.168.1.2/24 dev <interface name>`
  ↪ ...
  ↪ L'«Ordinateur n» s'attribue l'adresse `192.168.1.`$(n+1)$`/24` à l'aide de la commande `ip address add 192.168.1.`$(n+1)$ `dev <interface name>`
— Remarque (non important)
  ↪ L'«Ordinateur 1» constate l'ajout d'une règle de routage concernant le réseau local à l'aide de la commande `ip route`
  ↪ ...
  ↪ L'«Ordinateur n» constate l'ajout d'une règle de routage concernant le réseau local à l'aide de la commande `ip route`
— Attribution des routes : il faut que les paquets sortant du réseau local passent par le routeur
  ↪ L'«Ordinateur 1» s'attribue une route par défaut par `192.168.1.1` à l'aide de la commande `ip route add default via 192.168.1.1`
  ↪ ...
  ↪ L'«Ordinateur n» s'attribue une route par défaut par `192.168.1.1` à l'aide de la commande `ip route add default via 192.168.1.1`
— Permettre de natter les paquets en sortie de la machine à l'aide du morceau de configuration suivante dans `NFTables` :

```
table ip nat {
    chain postrouting {
        type nat hook postrouting priority srcnat;
        ip saddr 192.168.1.0/24 snat to 185.230.78.40
    }
}
```

Remarque :

`snat to 185.230.78.40` peut être remplacé par `masquerade`, qui est un raccourci de `NFTables` pour dire «change l'IP source de ce paquet en une IP de l'interface de sortie correspondant à ce paquet, *i.e.* l'interface de sortie du routeur».

5 Correction des Travaux Pratiques

5.1 TP0 : Écrivez un pare-feu pour votre machine

Correction : *Cf.* plus haut.

5.2 TP1 : Partage de connexion Internet

5.3 Sujet

Ce TP est dessiné pour deux personnes `A` et `B` ou plus :

1. `A` conserve un accès à Internet, mais `B` coupe la sienne
2. `A` et`B` placent un câble Ethernet (RJ-45) entre leurs machines (des adaptateurs USB – RJ-45 peuvent être utilisés)
3. `A` dit à `B` quelle IP s'attribuer et lui donne accès à Internet

Remarque :

Il est possible que `B` ait des difficultés à résoudre les noms de domaines. Si cela arrive, il faut que `B` récupère le contenu de `/etc/resolv.conf` de `A` et le place dans son fichier du même nom.

5.3.1 Mise en place (si vous êtes seul)

Si vous lisez seul ce fichier, le script `TP1.sh` vous permet d'émuler un camarade, vous permettant d'effectuer ce TP. Pour cela, veuillez suivre les étapes suivantes :

1. ouvrir un `tmux` en nommant la session TP1 [3],
2. executer le fichier `TP1.sh`,

Dans le `tmux`, une fenêtre nommée «routeur» (resp. «client») représente la configuration de votre machine (resp. de votre camarade). Vous pouvez ainsi effectuer le TP seul.
Chaque fenêtre est séparée en cinq sections :

haut-gauche Vue de votre configuration `NFTables` auto réactualisée

bas-gauche Shell bash en mode root

haut-droite Une vue de vos adresses IPs auto réactualisée

milieu-droite Une vue de vos routes auto réactualisée

bas-droite Une vue de vos voisins auto réactualisée

Vous pouvez effectuer le TP dans le `tmux`.

Une fois terminé, fermez le tmux et lancez le script `unTP1.sh` qui retirera le nom d'espace réseau temporaire créé pour représenter le client.

Correction : exécuter le script `TP1_correction.sh` qui écrira les lignes de commandes nécessaire dans les fenêtres du `tmux`. Vous pourrez alors vous rendre dans le `tmux` et observer ce qui a été modifié.

5.4 TP2 : Partage de connexion Internet en cascade

5.4.1 Sujet

Ce TP est prévu pour quatre personnes **R1**, **r2**, **c1** et**c2**.

L'objet de ce TP est d'aller un cran plus loin que le simple partage de connexion et permet de s'interroger plus sérieusement sur le chemin pris par les paquets *i.e.* le routage.

L'objectif est que :

— **R1** partage sa connexion Internet avec **c1** et **r2**,
— **r2** partage sa connexion Internet avec **c2**,
— Que **c1** et**R1** soient capable de pinger **c2** (seul **R1** doit adapter sa configuration).

5.4.2 Mise en place (si vous êtes seul)

Si vous lisez seul ce fichier, le script `TP2.sh` vous permet d'émuler un camarade, vous permettant d'effectuer ce TP. Pour cela, veuillez suivre les étapes suivantes :

1. ouvrir un `tmux` en nommant la session TP2 [4],
2. executer le fichier `TP2.sh`,

3. `tmux new-session -s TP1` vous permet de lancer un tel `tmux`
4. `tmux new-session -s TP2` vous permet de lanceer un tel `tmux`

Dans le `tmux`, vous trouverez les fenêtres suivantes :

R1 représentant la machine ayant l'accès direct à Internet

c1 Représentant un client de **R1**, lequel lui fournira Internet

r2 représentant le second routeur, client de **R1**

c2 représentant un client de **r2**, lequel lui fournira Internet

Chaque fenêtre est séparée en cinq sections :

haut-gauche Vue de votre configuration `NFTables` auto-réactualisée

bas-gauche Shell bash en mode root

haut-droite Une vue de vos adresses IPs auto-réactualisée

milieu-droite Une vue de vos routes auto-réactualisée

bas-droite Une vue de vos voisins auto-réactualisée

Vous pouvez effectuer le TP dans ce `tmux`.

Une fois terminé, fermez le `tmux` et lancez le script `unTP2.sh` qui retirera le nom d'espace réseau temporaire créé pour représenter les clients.

Correction : exécuter le script `TP2_correction.sh` qui écrira les lignes de commandes nécessaire dans les fenêtres du `tmux`. Vous pourrez alors vous rendre dans le `tmux` et observer ce qui a été modifié.