Réseaux
Les pare-feux

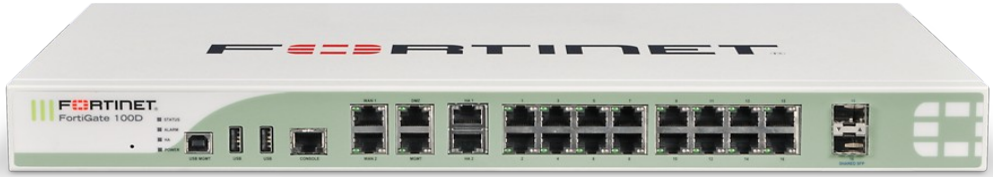
Sommaire

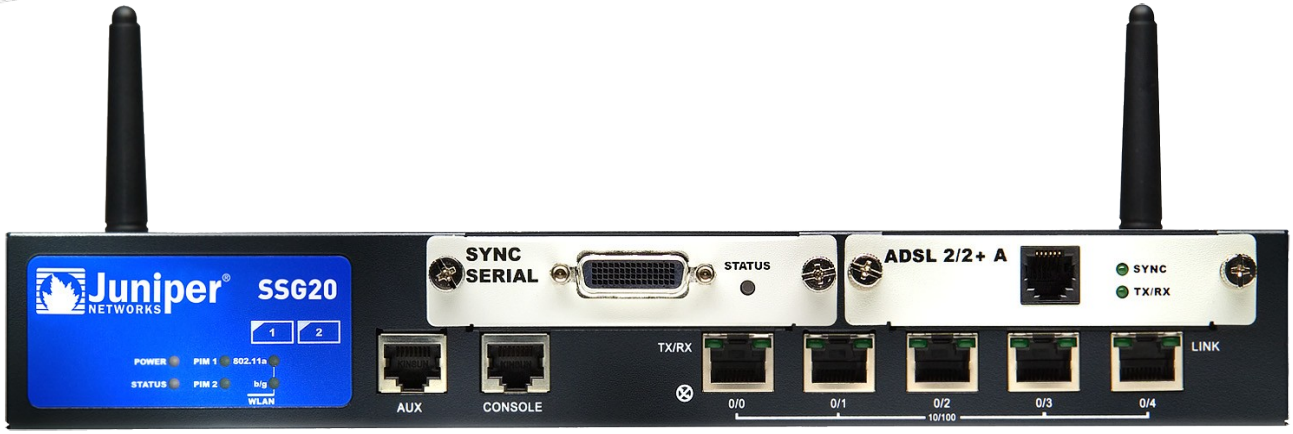
Généralités

Réseaux : Les pare-feux

Généralités

- Il existe des pare-feux matériels...





- ...et logiciels

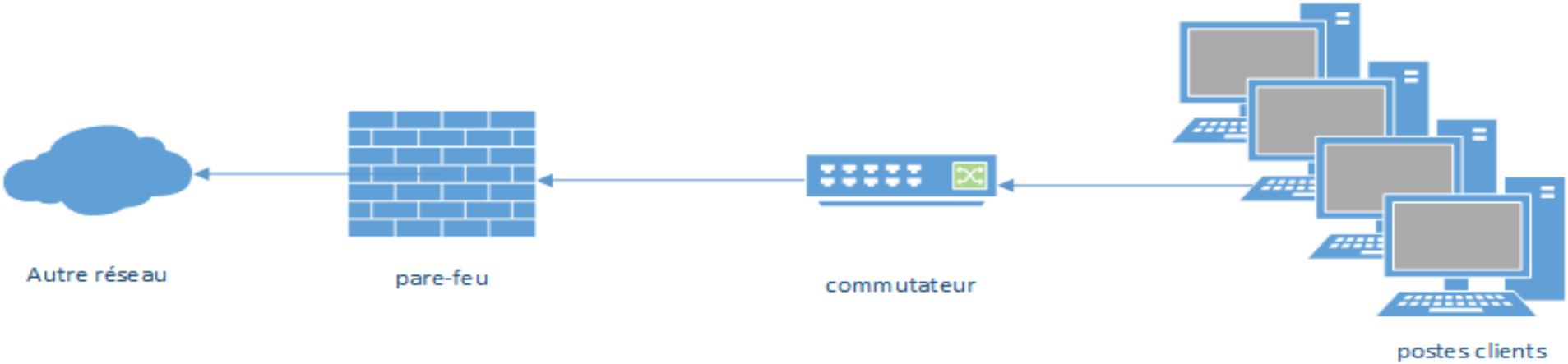
Il est utilisé pour protéger un réseau :

- **Notion de guichet** : restriction de passage en un point précis et contrôle des requêtes ;
- **Notion d'éloignement** : empêcher un attaquant d'entrer dans un réseau protégé ou même de s'approcher de trop prés de machines sensibles ;
- **Notion de confinement** : empêcher les utilisateurs internes de sortir du domaine protégé sauf par un point précis.

Généralement il concerne les couches basses (IP/TCP/UDP), mais peut également comprendre la couche application (HTTP, FTP, SMTP…)

Ce que peut faire un pare-feux :

- Être un point central de contrôle de sécurité plutôt que de multiples contrôles dans différents logiciels clients ou serveurs ;



Ce que peut faire un pare-feux :

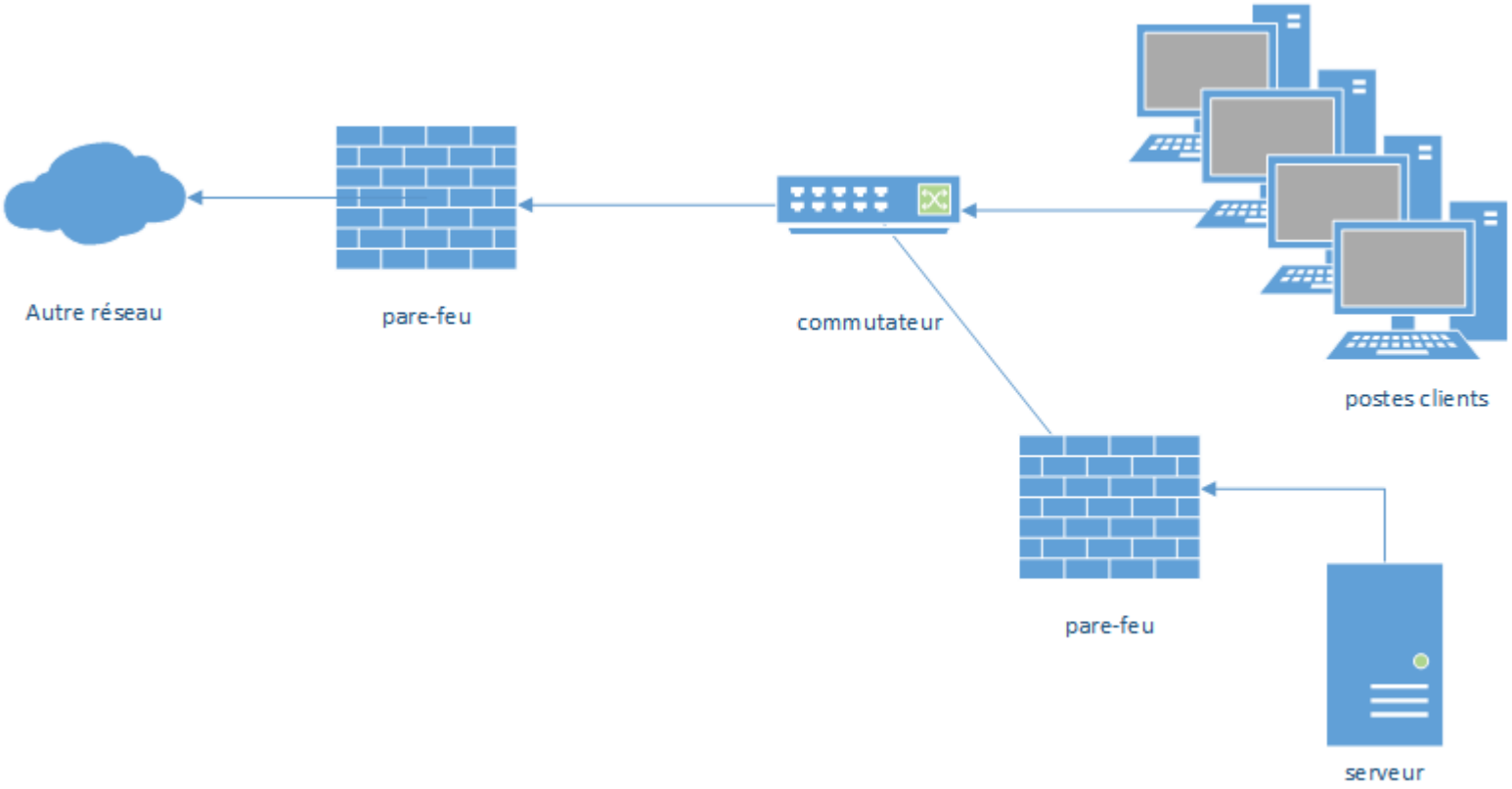
- Appliquer une politique de contrôle d'accès ;

- Enregistrer le trafic (journaux de sécurité / logs) ;

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
1232K  389M f2b-SSH    tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:22
  18M   44G ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0            state RELATED,ESTABLISHED
  238 13380 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:1194
 355K   16M ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80
1723K  113M ACCEPT     udp  --  enp2s0 *       0.0.0.0/0            0.0.0.0/0            udp dpt:53
 459K   36M ACCEPT     udp  --  enp0s6f1u1 *       0.0.0.0/0            0.0.0.0/0            udp dpt:53
15340 5181K ACCEPT     udp  --  enp2s0 *       0.0.0.0/0            0.0.0.0/0            udp dpt:67
  363  119K ACCEPT     udp  --  enp0s6f1u1 *       0.0.0.0/0            0.0.0.0/0            udp dpt:67
  124  6528 ACCEPT     tcp  --  enp2s0 *       0.0.0.0/0            0.0.0.0/0            tcp dpt:445
    0     0 ACCEPT     tcp  --  enp2s0 *       0.0.0.0/0            0.0.0.0/0            tcp dpt:139
    0     0 ACCEPT     tcp  --  enp2s0 *       0.0.0.0/0            0.0.0.0/0            tcp dpt:10000
    1    52 ACCEPT     tcp  --  enp2s0 *       0.0.0.0/0            0.0.0.0/0            tcp dpt:9091
 337K   16M ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state NEW tcp dpt:22
14102 2307K ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
1050K   76M ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
 6264  207K ACCEPT     all  --  tun0   *       0.0.0.0/0            0.0.0.0/0
  15M 1699M DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Ce que peut faire un pare-feux :

- Appliquer une défense en profondeur (multiples pare-feux) ;



Ce que ne peut pas faire un pare-feux :

- Protéger contre les utilisateurs internes (selon leurs droits) ;
- Protéger un réseau d'un trafic qui ne passe pas par le pare-feu (eg. Modems) ;
- Protéger contre les virus ;
- ~~Être gratuit~~ et se configurer tout seul.

Il dispose de divers niveaux de filtrage :

• Liaison (adresse MAC, …) ;	OSI 2
• Réseau (en têtes IP, ICMP, …) ;	OSI 3
• Transport (ports TCP, UDP) ;	OSI 4
• Filtrage adaptatif (« statefull inspection ») ou dynamique ;	OSI 3 / 4
• Application (relais applicatifs – « application proxys »)…	OSI 7

Fonctionnalité:

- Principale → filtrage IP (TCP/UDP) ;

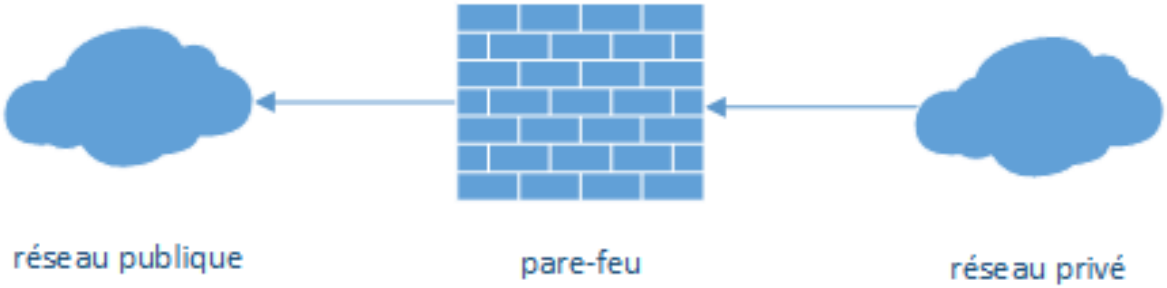
HTTP DNS

DHCP ? FTP

SSH

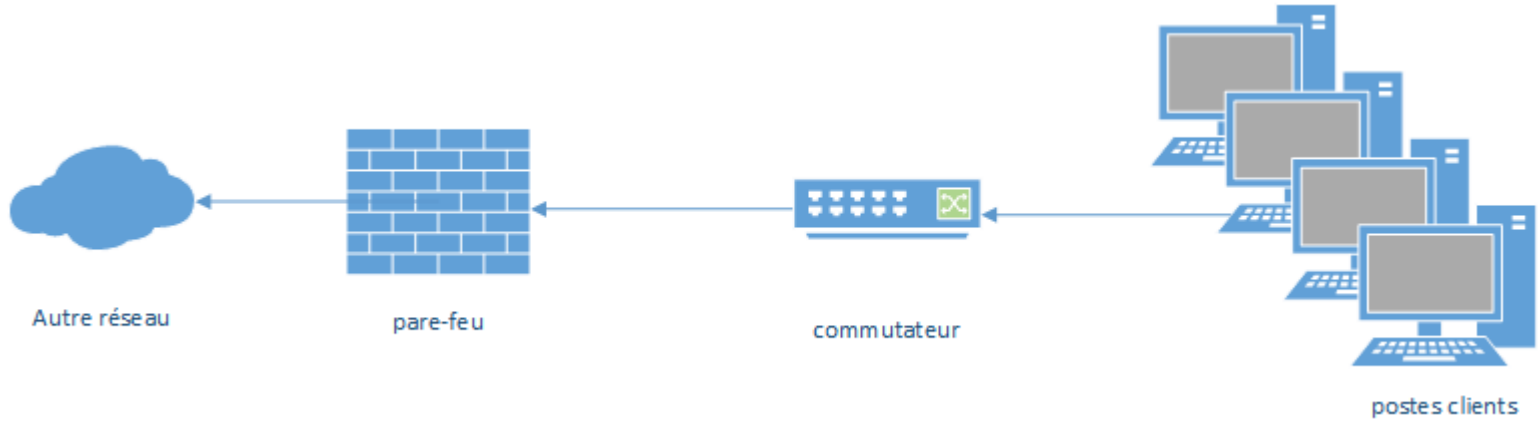
HTTPS

- Supplémentaire → traduction d'adresses et de ports.



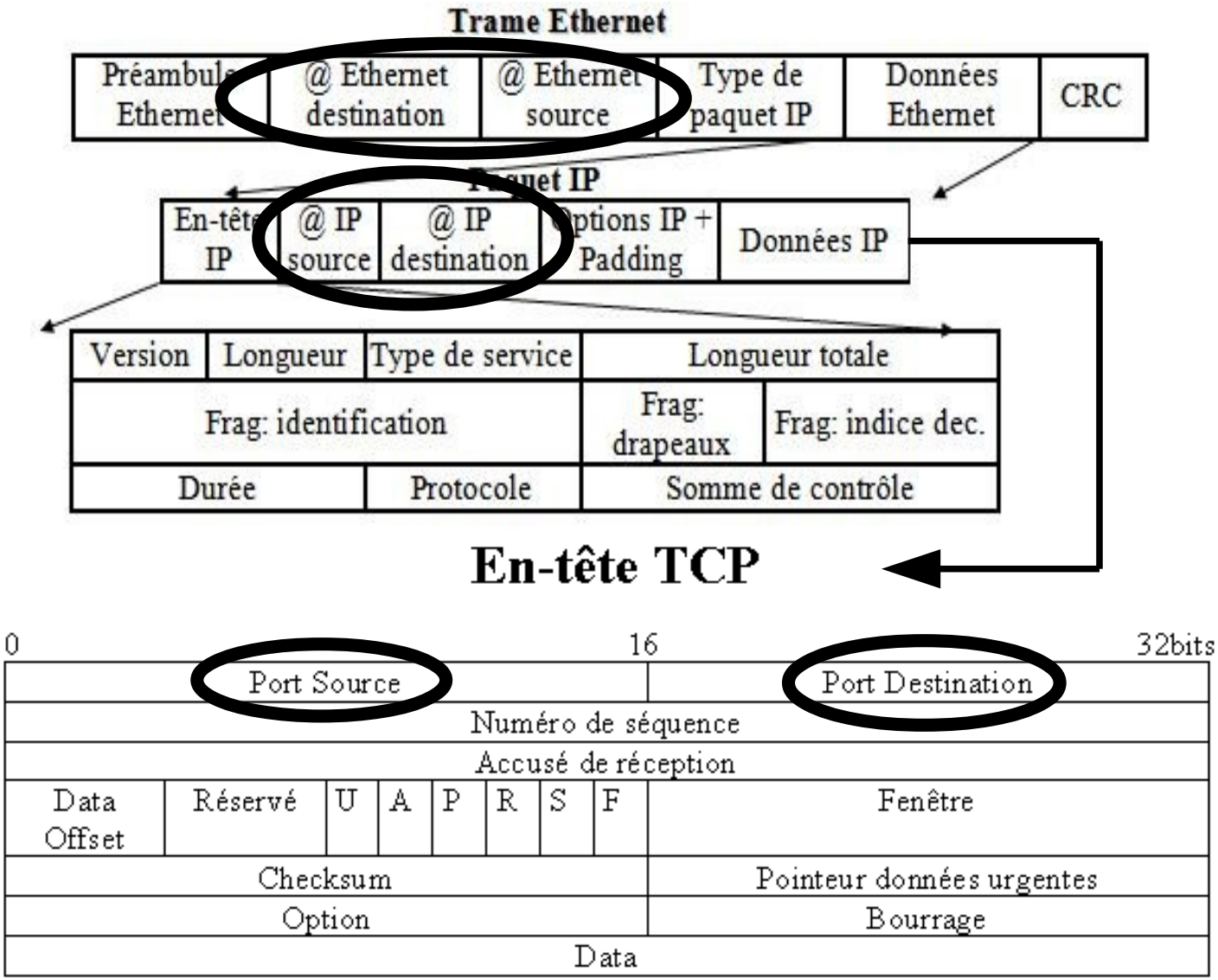
Filtrage IP (TCP/UDP)

- Idée → contrôler les paquets IP (TCP/UDP) autorisés à atteindre un groupe de machine ;



- Intérêt → sécuriser de manière globale !

Filtrage sur les entêtes :



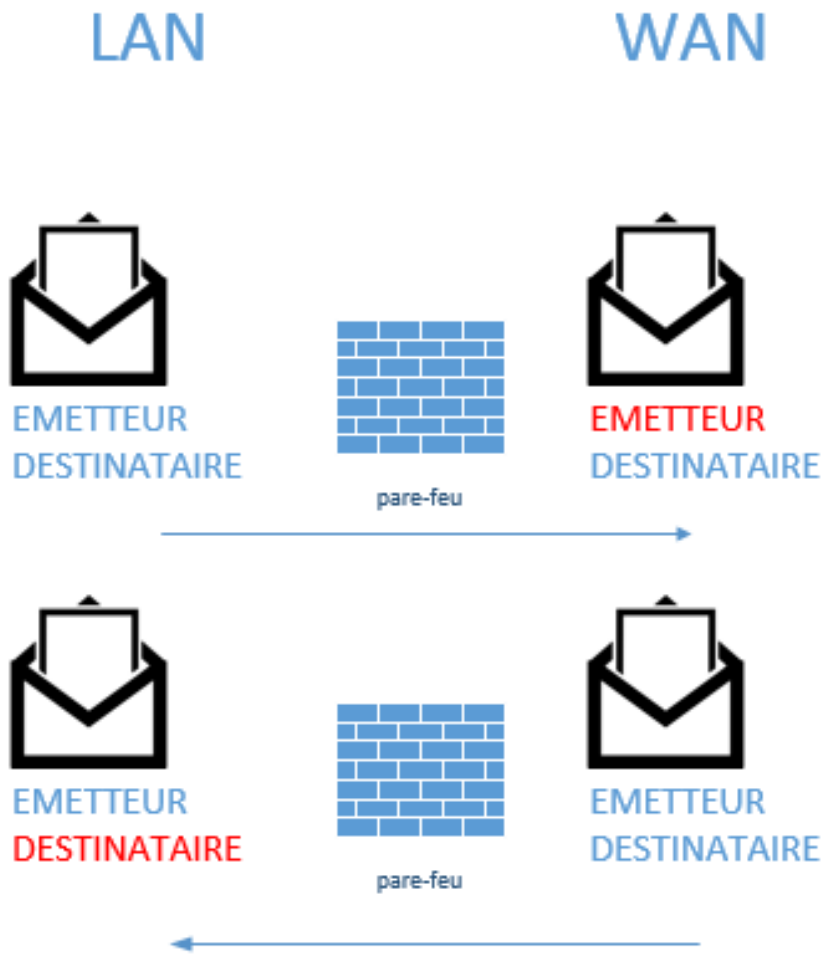
OSI 2

OSI 3

OSI 4

Traduction d'adresses et de ports

- Idée → réécrire les en-têtes des paquets ;



Intérêt :

→ faire face à la pénurie d'adresses routables sur l'Internet (NAT)

→ laisser entrer certains paquets (PAT)

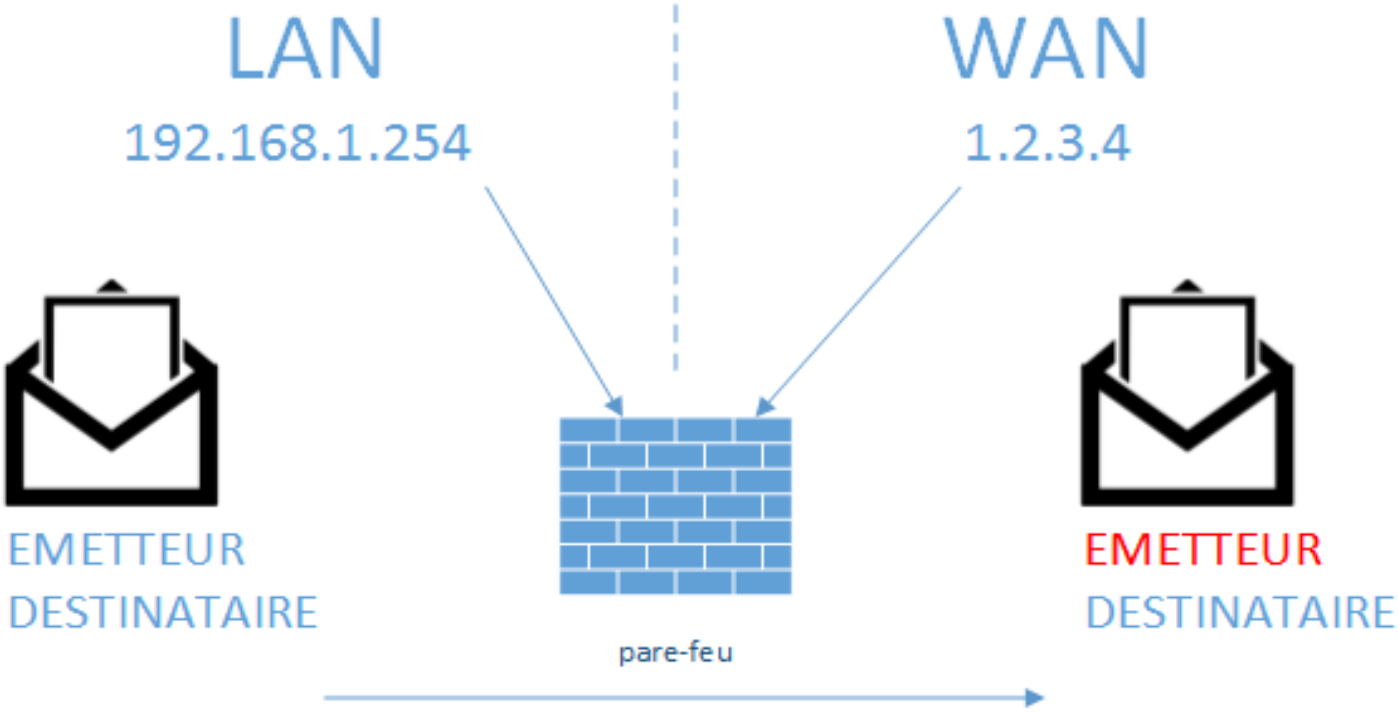
Traduction d'adresses :

- **<u>Uni-directionnelle</u>** :
  - Traduction en sortie d'adresses (typiquement adresses privées en publiques)
  - Possibilité de changer le port source

- **<u>Bidirectionnelle</u>** : traduction d'une adresse publique en une adresse privée et réciproquement

- **<u>Redirection de ports</u>** en entrée vers une adresse privée, en modifiant le port de destination ou non (**PAT**)

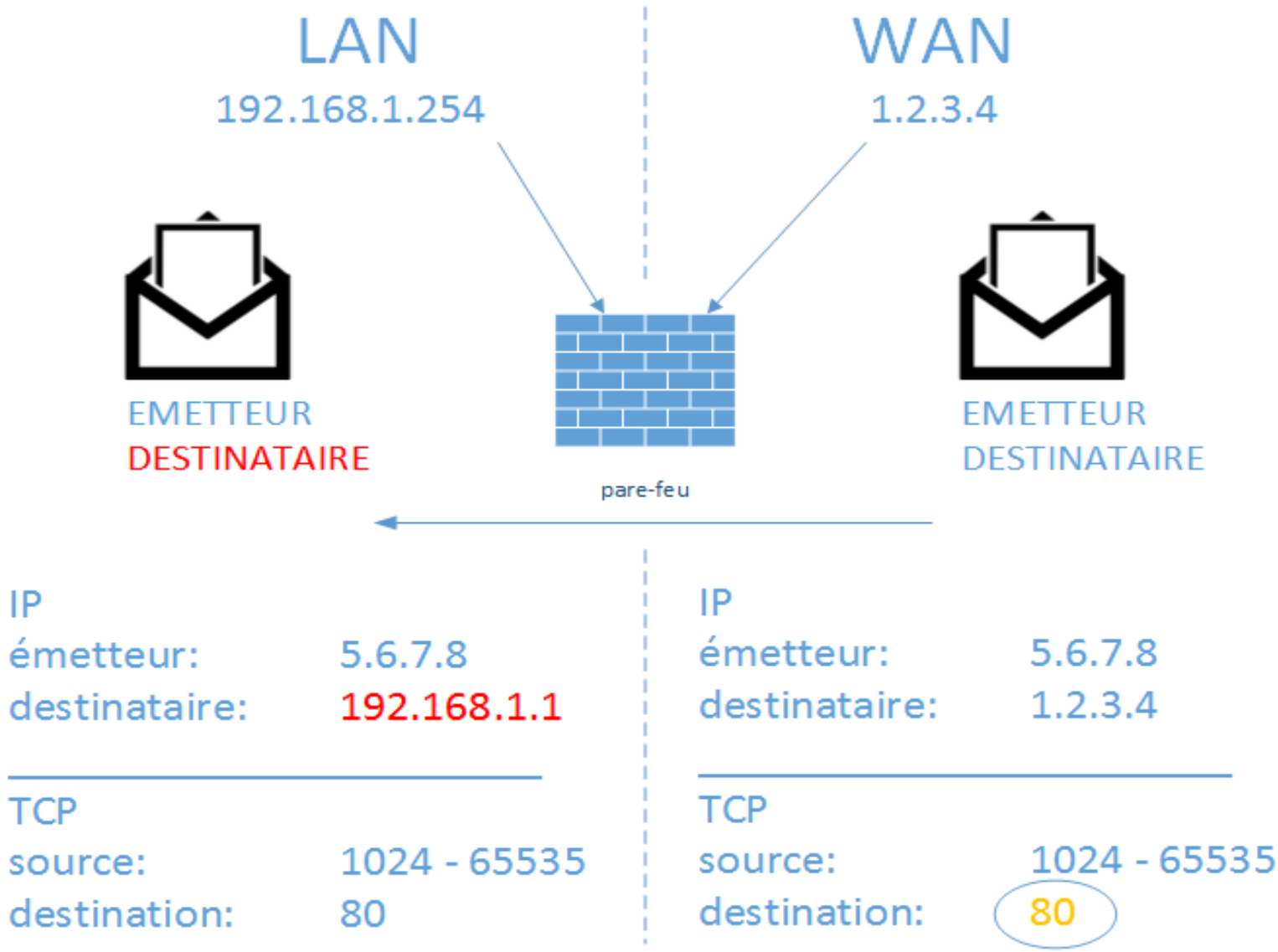
Translation d'adresses uni-directionnelle (NAT) :



IP émetteur:	192.168.1.1
destinataire:	8.8.8.8

IP émetteur:	1.2.3.4
destinataire:	8.8.8.8

Traduction de ports (PAT) :



Avantages :

- gestion de la sécurité concentrée
- capacité d'audit du trafic réseau
- concentrer la maintenance sur une machines plutôt qu'un parc

Inconvénients :

- nécessite une connaissance des protocoles filtrés (TCP/IP, HTTP, FTP, SQL, …)
- Compréhension du fonctionnement du pare-feu (divers niveaux de filtrage, traduction d'adresse, …)

Implémentations logicielles :

Netfilter (moteur d'Iptables) (Paul Russell)

- Filtre de paquets du noyau Linux 2.4
- Successeur d'IPChains (Linux 2.2)

IP Filter (Darren Reed)

- Filtre de paquets fonctionnant sous Unix libres et propriétaires
- Intégré dans FreeBSD et NetBSD

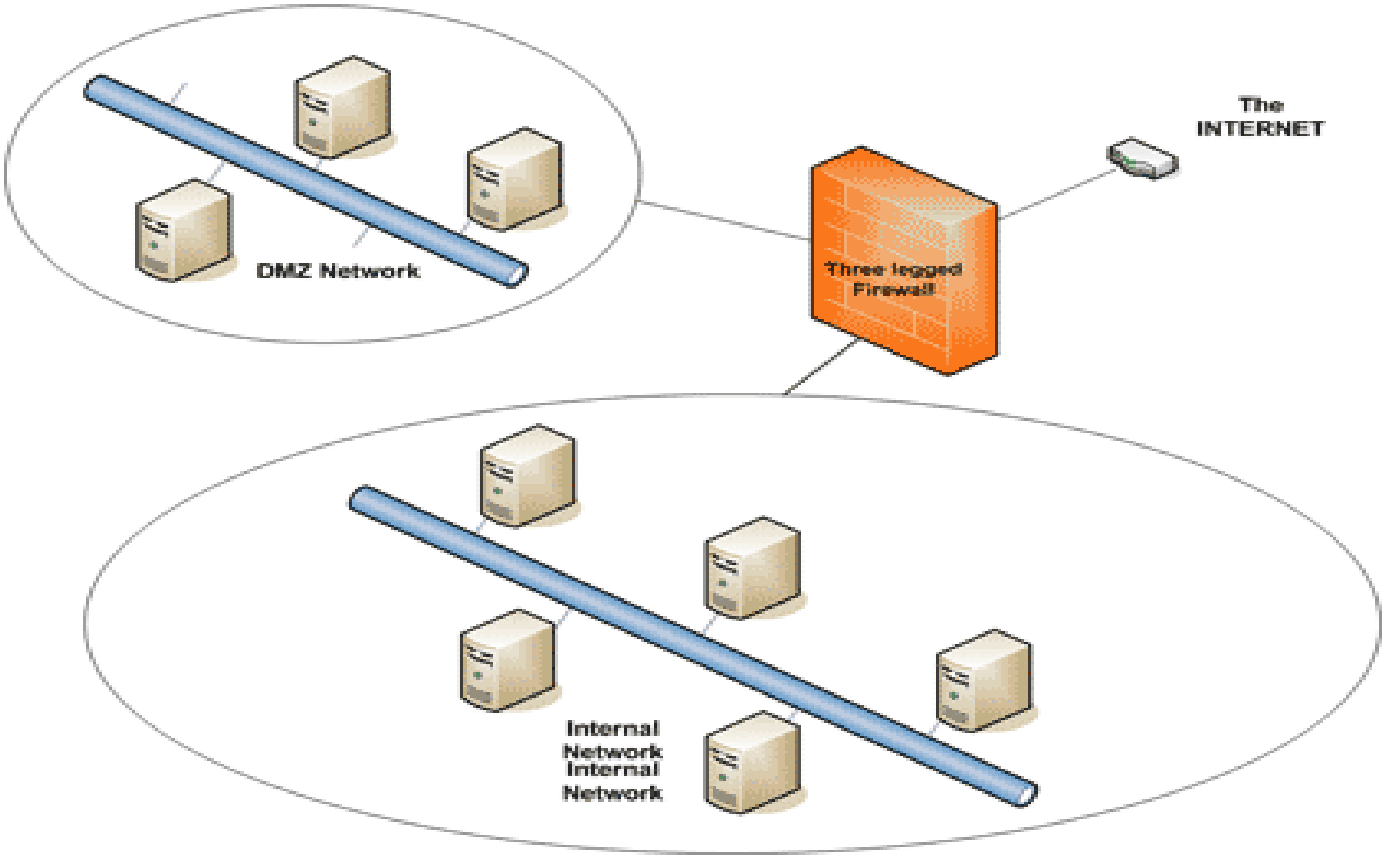
Packet Filter (Daniel Hartmeier)

- Filtre de paquets dans OpenBSD (à partir de la version 3.0)

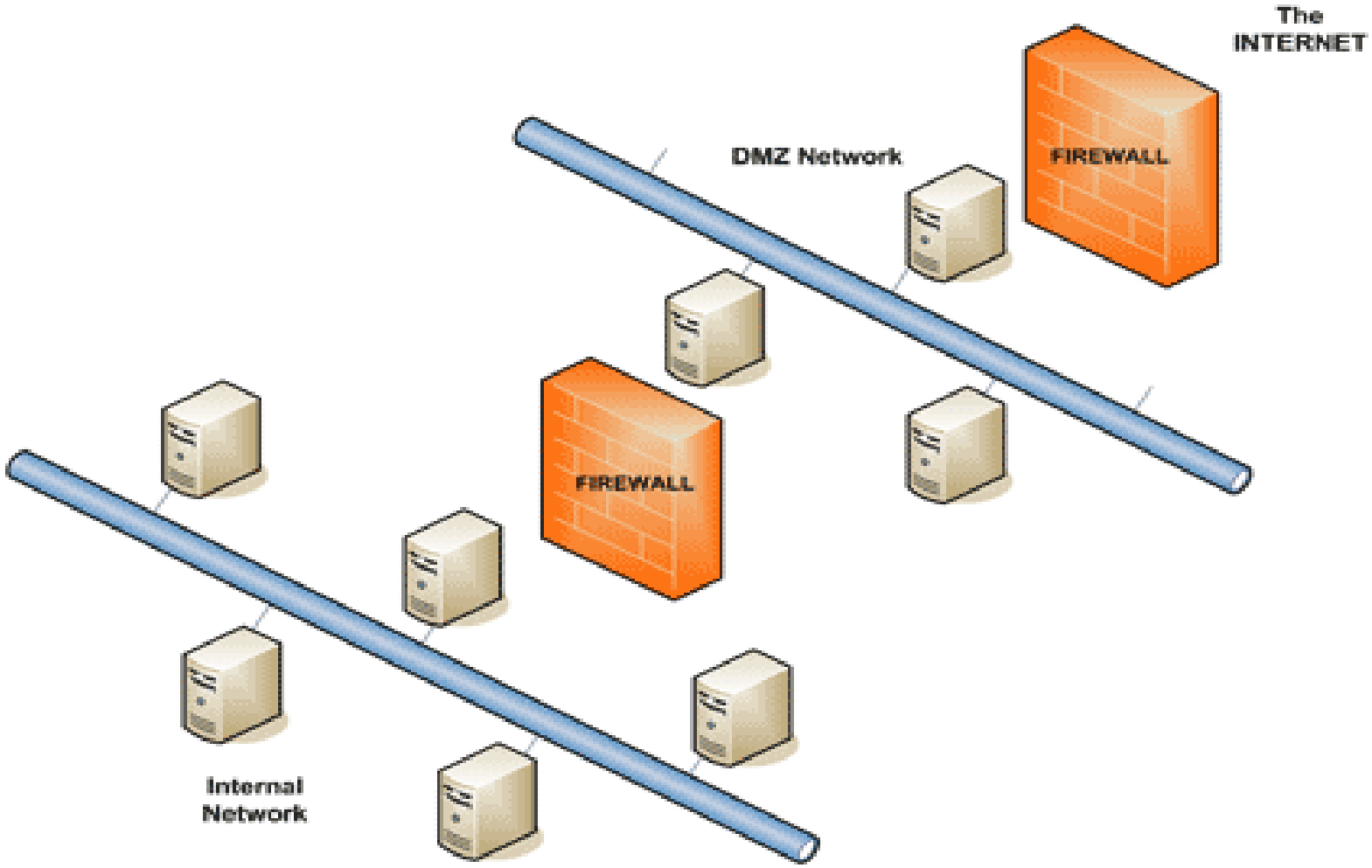
Architecture / utilisation

DMZ (demilitarized zone) : est un sous-réseau séparé du réseau local et d'Internet par un pare-feu.

Dans une configuration **en étoile**, nécessite un pare-feu à trois interfaces (pattes) minimum



Dans une configuration **en sas**, nécessite deux pare-feux à deux interfaces (pattes)



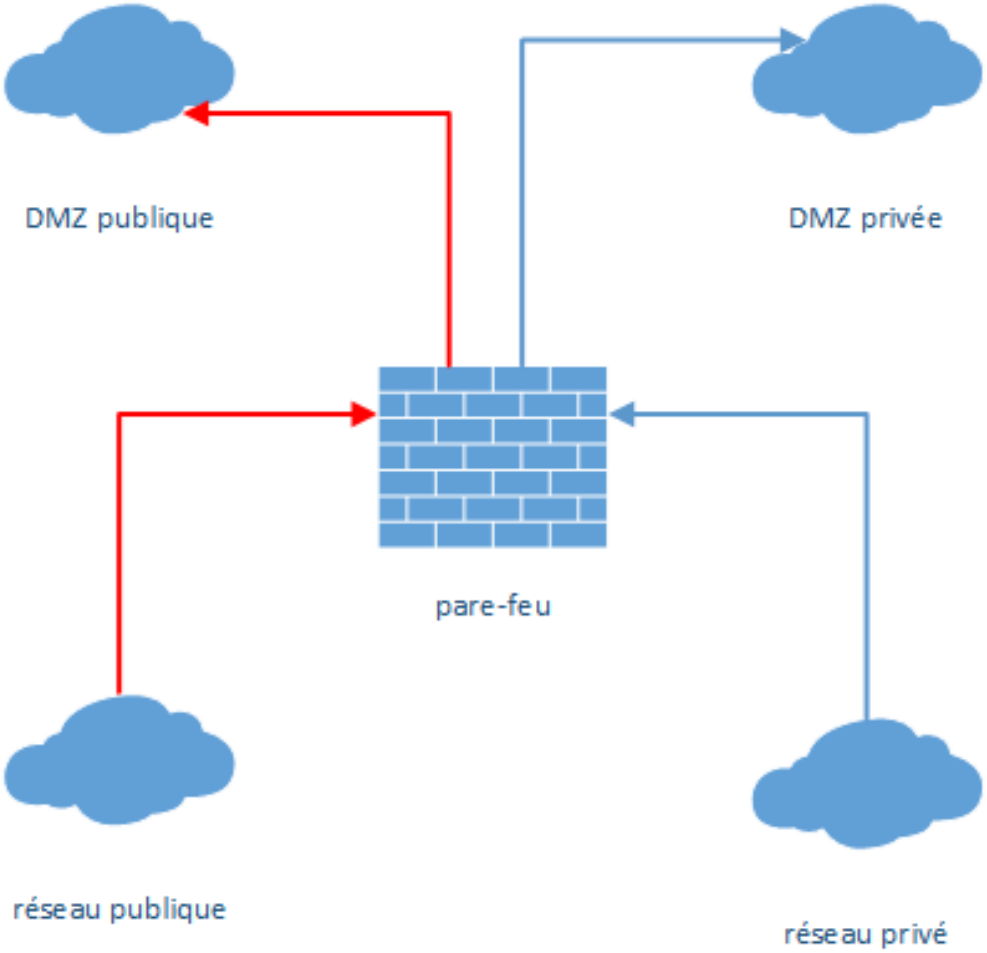
Il existe deux types de DMZ : publiques et privées

Les flux autorisés sont :

WAN → DMZ publique
(DMZ publique → WAN)

LAN → DMZ privée
(DMZ privée → LAN)

DMZ privée → DMZ publique
(DMZ publique → DMZ privée)



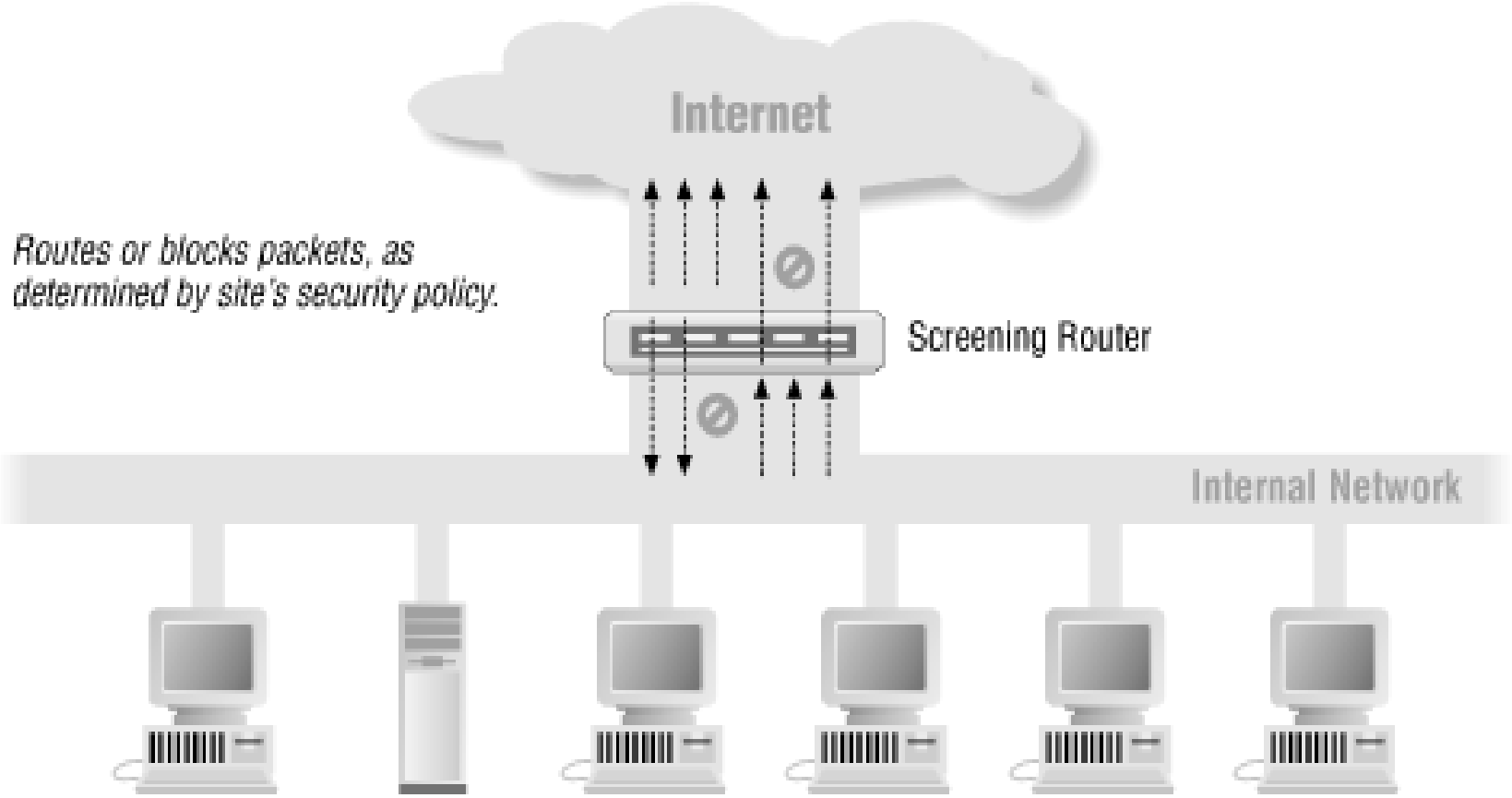
Que mettre dans la DMZ ?

- Publique : les services accédés depuis Internet (HTTP, DNS, …) ;
- Privée : les services accédés depuis le LAN (DNS, DHCP, SQL, …) ;

Pourquoi la DMZ ?

- En cas de compromission d'un des services dans la DMZ, le pirate n'aura accès qu'aux machines de la DMZ et non au réseau local.

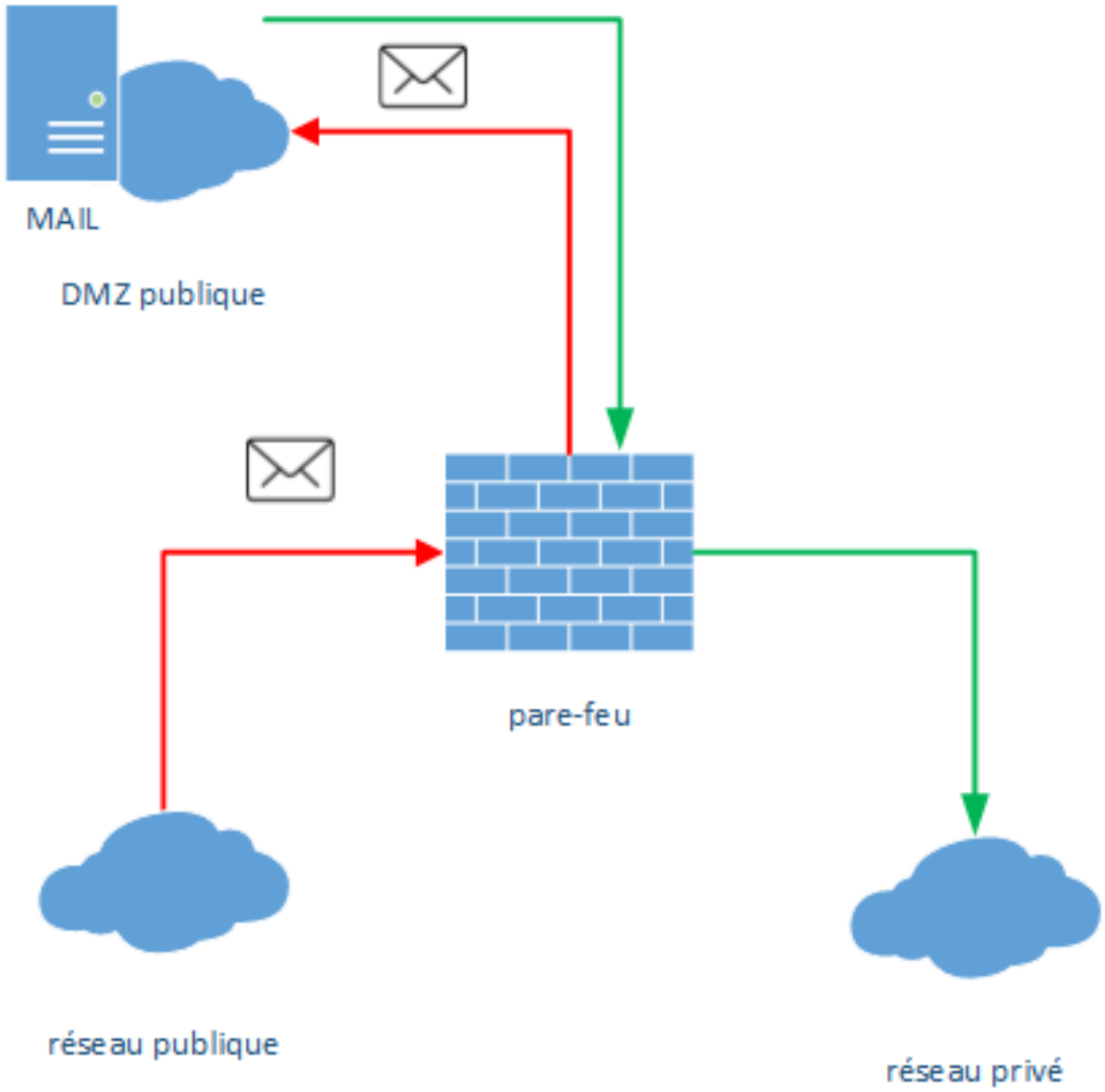
Architecture de pare-feu avec routeur filtrant : « screening router »



Bastion (hôte exposé au réseau externe)

- Il constitue un point de contact entre réseau externe et réseau interne ;
- S'occupe d'un ensemble de services prédéfinis HTTP, DNS, SMTP, … ;
- Il peut agir :
  - en rendant directement le service concerné ;
  - en relayant les requêtes vers d’autres serveurs après avoir effectué un contrôle d’accès applicatif (proxy) ;

Bastion (hôte exposé au réseau externe)



Le bastion peut servir dans la détection d'intrusion:

- Analyse des communications pour détecter des attaques : IDS (« Intrusion Detection System ») ;

Fonction pot de miel (Honeypot): un service semblant attractif pour un attaquant et qui n'est en réalité qu'un piège pour le détecter.

Filtrage

Il existe différents critères de filtrage :

- Interface réseau : entrée ou sortie ;
- Adresses IP (source / destination) : hôte ou sous-réseau ;
- Protocoles de niveau 3 : IP, ICMP, …
- Champs de l'entête IP / ICMP : fragmentation, TOS / TTL, options, types et codes (ICMP) ... ;
- Protocoles de niveau 4 : TCP, UDP, …
- Ports source et destination (TCP et UDP)
- Drapeaux (TCP) ;
- ...

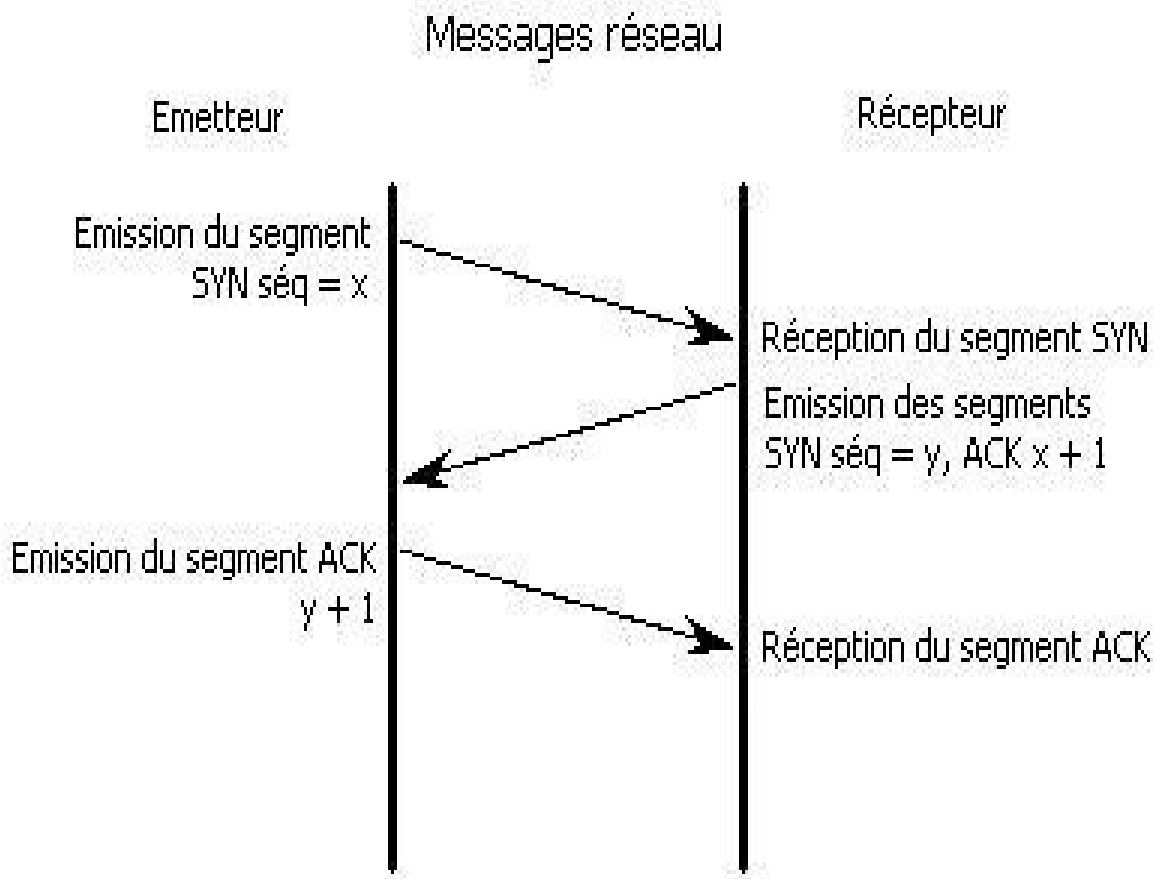
Il est possible d'appliquer différentes actions :

- Laisser passer (ACCEPT) ;
- Bloquer (DENY ou DROP) ;
- Rejeter (REJECT) => message ICMP ou segment TCP avec drapeau RST.

Filtrage : stateful

- Le filtrage « stateful », également appelé filtrage dynamique, permet de suivre les états des connexions en cours ;
- Seuls des paquets correspondants à un état pré-existant sont acceptés ;
- L'intérêt est une simplification de l’écriture des règles de filtrage ;
- Il améliore la sécurité, en n’autorisant que le trafic effectivement licite ;

Établissement d'une connexion TCP :



Filtrage : stateful

Il permet:

- En TCP de contrôler un segments appartenant à une connexion en cours ;
- En UDP d'autoriser un datagrammes en réponse à un datagramme précédemment émis ou d'envoyer un messages ICMP d’erreur ;
- En ICMP d'envoyer un messages en réponse à un message ICMP émis

Filtrage : stateful

Mise en œuvre :

- Création d’un état lors de la traversée du premier paquet ;
- Mémorisation de paramètres identifiant de façon unique une connexion ;
- Validation des paquets par comparaison des états courants ;
- Expiration des états après un temps paramétrable.

Filtrage : stateful

Paramètres conservés :

- Adresses source et destination ;
- Ports source et destination ;
- Type, code, identifiant et numéro de séquence (ICMP) ;

Pour TCP, il s’assure que les segments reçus font partie d’une connexion en cours.

Filtrage : notion de flux

Les règles sont souvent exprimées selon une notion de direction : entrant ou sortant.

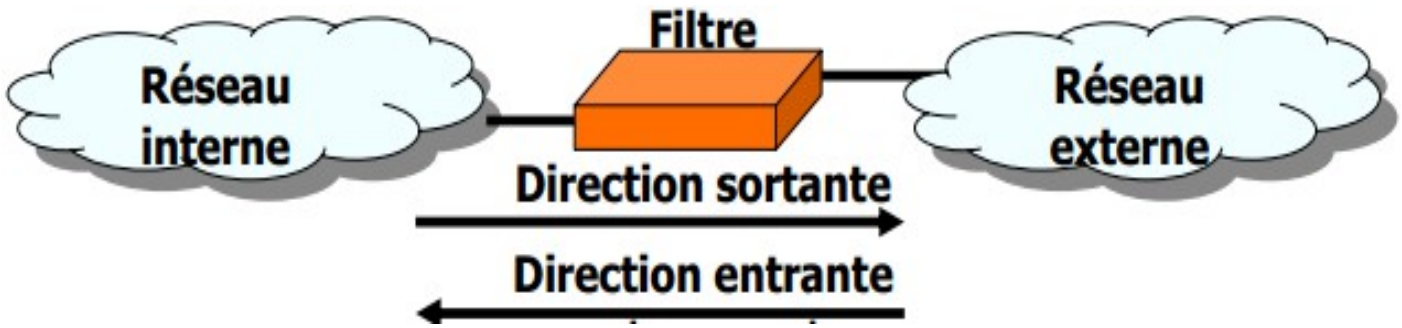
- Direction associée au réseau à protéger ;
- Les règles ne sont pas symétriques: on donne des droits à un hôte interne vers l'extérieur et on refuse ces droits de l'extérieur vers l'intérieur ;
- Position du filtrage dans un routeur filtrant selon l'interface.
- Application d'une règle de filtrage : soit dès la réception ou avant l'émission ;

Filtrage : notion de flux

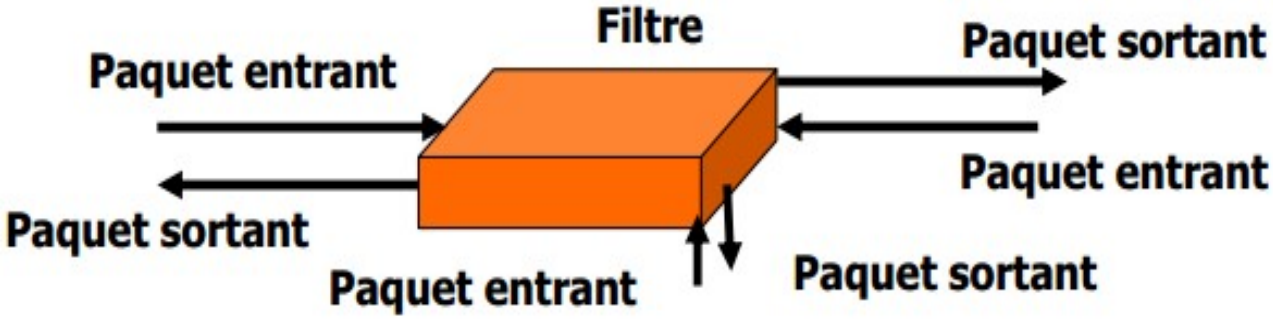
Les règles sont souvent exprimées selon une notion de direction : entrant ou sortant.

- Direction associée au réseau à protéger ;
- Les règles ne sont pas symétriques: on donne des droits à un hôte interne vers l'extérieur et on refuse ces droits de l'extérieur vers l'intérieur ;
- Position du filtrage dans un routeur filtrant selon l'interface.
- Application d'une règle de filtrage : soit dès la réception ou avant l'émission ;

Direction associée au réseau à protéger :



Position du filtrage dans un routeur filtrant selon l'interface.



Notion de direction selon le sens de connexion

- La direction concerne des connexions :

  ouvreur actif TCP vers ouvreur passif (client/serveur)

- Règles concernant des services sortants :

  la connexion est à la demande d'un ouvreur ou d'un client interne (eg. SSH sortant).

- Règles concernant des services entrants :

  la connexion est à la demande d'un ouvreur ou d'un client externe (eg.: http entrant).

Notion de direction selon le sens de connexion

Conclusion : Introduction dans certains filtres d'une sémantique de sens dans l'expression des règles

→ Bien connaître la sémantique de la direction dans le filtre utilisé concernant les notions d'entrant et de sortant.

Filtrage : politique par défaut

Autoriser tout par défaut

On permet tout sauf ce qui est considéré comme dangereux
→ Tout ce qui n’est pas explicitement interdit est autorisé.

On analyse les différents risques des applications qui doivent s’exécuter.

→ On en déduit les interdictions à appliquer, on autorise tout le reste.

Avantages/inconvénients :

- inconfortable pour l’administrateur de la sécurité ;
- facilite l’accès des usagers au réseau ;
- peu recommandée et peu utilisée.

Interdire tout par défaut

→ Tout ce qui n’est pas explicitement permis est interdit.

Avantages/inconvénients :

- plus sécuritaire et plus confortable pour l’administrateur de la sécurité ;
- limite considérablement les droits des usagers ;
- recommandée et plus souvent utilisée.

Filtrage : Spécification abstraite

Définir de manière abstraite la politique de sécurité :

ce qui est autorisé et ce qui est interdit

- Choisir une politique d'ensemble:

Solution 1 → Tout ce qui n'est pas explicitement autorisé est interdit.

Solution 2 → Tout ce qui n'est pas explicitement interdit est autorisé.

- Énoncer les règles :

Exemple 1 : Autoriser un hôte interne à recevoir du courrier électronique parce que c'est un serveur de courrier SMTP.

Exemple 2 : Interdire à un hôte externe précis d'envoyer du courrier SMTP parce qu'il est en liste noire.

Filtrage : Spécification abstraite

Traduire la politique de sécurité en règles précises concernant des communications IP

Règles concernant des datagrammes IP : adresses source/destination, port TCP source/destination, indicateurs TCP, …

Réunir ses informations dans une matrice de flux :

	Internet	DMZ	Zone Interconnexion	Zone supervision	Zone Serveur
Internet	Tout	SMTP sortant + DNS + PROXY	rien	rien	rien
DMZ	SMTP entrant + web + dns	Tout	SMTP sortant	VNC + TSE + TELNET+SNMP	rien
Zone Interconnexion	rien	SMTP + WEB	Tout	VNC + TSE + TELNET+SNMP	rien
Zone supervision	rien	rien	rien	tout	rien
Zone Serveur	rien	rien	rien	VNC + TSE + TELNET+SNMP	Tout
Zone Administratif	rien	rien	rien	TELNET+SNMP	rien
Zone Cursus	rien	rien	rien	TELNET+SNMP	rien

Iptables

- Iptables est un outil Linux pour gérer le pare-feu qui est intégré au noyau Linux 2.4 (et supérieur) ;

- L'architecture du noyau pour le système de pare-feu s'appelle 'Netfilter' ;

- Netfilter a de très nombreuses fonctionnalités: filtrage de paquets, suivi de connexions, NAT, … ;

Le principe de fonctionnement est simple :

- lorsque la carte réseau reçoit un paquet celui-ci est transmis à la partie **Netfilter** du noyau ;

- **Netfilter** va ensuite étudier ce paquet (les entêtes et le contenu) et en se basant sur des règles que l'administrateur aura défini ;

Il va choisir de :

- laisser passer le paquet intact ;
- de le modifier ;
- de le transmettre à une autre machine ;
- d'interdire son passage.

Iptables: fonctionnement

- Iptables s'organise par tables ;
- Chaque table contient une série de règles appelées chaînes ;
- Il existe 3 tables principales :
  - **FILTER** :correspond aux notions de filtrage réseaux (accepter/refuser un paquet) contient les chaînes INPUT, FORWARD et OUTPUT (table par défaut)
  - **NAT** : correspond à des fonctions de routage et s’occupe de la conversion d’adresse réseau. Contient les chaînes PREROUTING, OUTPUT et POSTROUTING.
  - **MANGLE** : est utilisée pour modifier les paquets à la volée. Contient les chaînes PREROUTING, INPUT, FORWARD, OUTPUT et POSTROUTING.

- Les règles de chaque chaîne dépendent de leur ordre ;

- Quand un paquet arrive, il est comparé aux règles de la chaîne pertinente appartenant à la table pertinente, depuis la première jusqu'à la dernière.

iptables [-t table] CMD [chain] [rule-spec|num] [options]

<u>Exemple :</u>

iptables –t filter –A input –i eth0 -s 192.168.0.1 –p TCP –j DENY

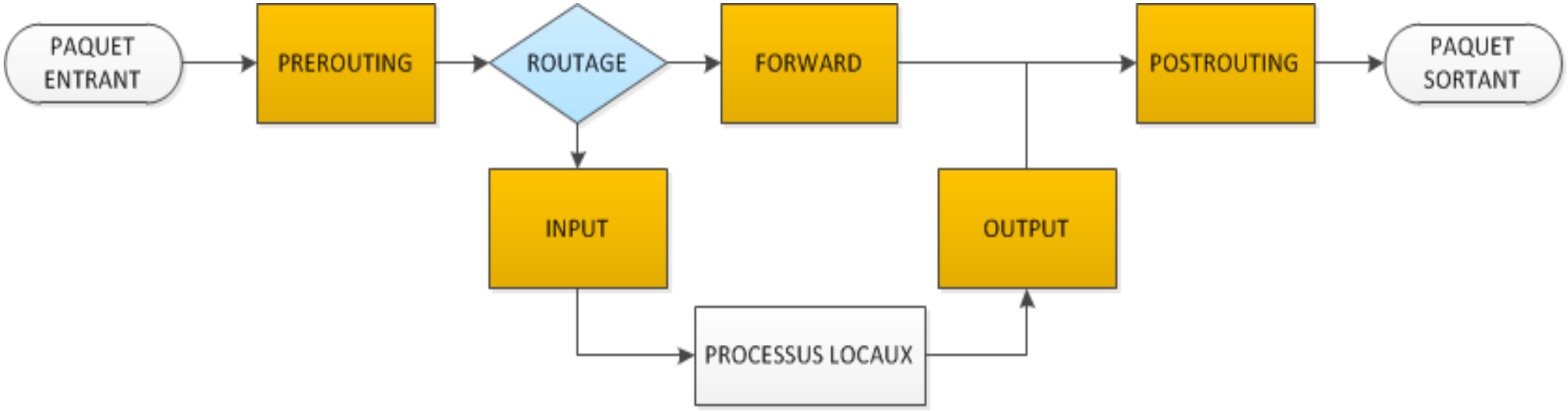
- Quand un paquet arrive, le noyau regarde la destination de ce paquet : c'est appelé le routage.

- Si ce paquet est destiné à cette machine le paquet passe dans la chaîne INPUT et les processus qui attendent le paquet le recevront.
- Sinon, si :
  - le noyau n'a pas le **forwarding** autorisé (machine qui n'est pas configurée en routeur), ou qu'il ne sait pas comment forwarder le paquet, le paquet est 'droper' (supprimé).
  - le **forwarding** est autorisé et que le paquet est destiné à un autre réseau, le paquet va directement à la chaîne **FORWARD**.

- S'il est accepté, il sera envoyé vers le réseau de destination ;

- La chaîne **OUTPUT** concerne les paquets qui ont été créé par la machine locale et y passe immédiatement.

Iptables: filtrage



Tables	Chaînes
FILTER	INPUT, FORWARD, OUPUT
NAT	PREROUTING, POSTROUTING, OUTPUT
MANGLE	PREROUTING, INPUT, FORWARD, OUTPUT, PORTROUTING

Chacune de ces chaînes peut donner plusieurs réponses possibles pour paquet :

- **ACCEPT** : le paquet est accepté
- **DROP** : le paquet est refusé , on l'efface et on ne répond rien
- **REJECT** : le paquet est refusé et on signale le rejet à l'expéditeur

Chaque chaîne possède une règle 'policy' qui définit si l'on accepte ou refuse les paquets par défaut. Elle est généralement toujours à ACCEPT.

Exemple :

```
[root@hades ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

46242 33M ACCEPT all  --  *    *  0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED
    ...
Chain FORWARD (policy ACCEPT 936K packets, 714M bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 105K packets, 13M bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Les cibles sont spécifiées grâce à l'option –j :

- Cibles de FILTER (**ACCEPT**, **DROP**, **REJECT**) ;
- **SNAT** (adresse source convertie → dans chaîne **POSTROUTING** de table **NAT**) ;
- **DNAT** (adresse destination convertie → dans chaîne **PREROUTING** de table **NAT**) ;
- **LOG** (nécessité d'avoir chargé le module concerné dans le noyau)

- Plus d'informations :

http://www.iptables.info/en/iptables-targets-and-jumps.html

Iptables: options

Quelques commandes :

- -N (--new-chain) ;
- -X (--delete-chain) ;
- -F (--flush) : supprime toutes les règles de la chaîne et de la table concernée ;

Quelques options :

- -i [!] interface : spécifie l'interface de réception, valable pour les chaînes **INPUT**, **FORWARD** et **PREROUTING** ;
- -o [!] interface : spécifie l'interface d'expédition, valable pour les chaînes **OUTPUT**, **FORWARD** et **POSTROUTING** ;
- [!] –syn : spécifie que cette règle ne devrait satisfaire que les paquets TCP qui initient la connexion

Iptables: le module state

- Capacité de créer des règles de comparaison fondées sur les états des paquets grâce au module 'state' :

  iptables –m state –state [!] [state1, state2, …]

Voici les états possibles :

- **NEW** : compare les paquets n'appartenant à aucune connexion en cours ;
- **ESTABLISHED** : compare les paquets appartenant à une connexion déjà ouverte ;
- **RELATED** : compare les paquets qui appartiennent à une autre connexion.
  Exemple : messages ICMP d'erreur, trafic d'un protocole applicatif (FTP passif ou actif), … ;
- **INVALID** : compare les paquets qui n'ont aucun sens dans le contexte de la connexion existante, ou ceux qui n'ont pu être reçus pour une raison quelconque.

Protection de la machine locale :

Exécutez la commande :

```
# iptables -nvL
```

(-L liste les règles, -v active le mode 'verbeux', -n pour les sorties numeriques)

Vous devriez voir quelque chose qui ***ressemble*** à :

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target prot opt in out source destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
Pkts bytes target prot opt in out source destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out source
 destination
```

Dans l'ordre : la 'policy' de la chaîne **INPUT** est par défaut sur **ACCEPT**, le nombre de paquets qui a traversé les règles de la chaîne et le nombre d'octets correspondants.

Exemple: protection locale

Nous allons maintenant interdire les connexions sur le port 80, pour cela , il faut interdire les paquets dans la chaîne INPUT sur le port 80 :

```
# iptables -A INPUT -p tcp --dport 80 -j DROP
```

Cette commande signifie :

-A INPUT => ajoute en bas de la liste des règles de la chaîne INPUT

-p tcp => pour les paquets TCP

--dport => à destination du port 80

-j DROP => **DROP** les paquets, le -j peut également servir à rediriger sur une autre chaîne et/ou à effectuer différentes actions

Exemple: protection locale

Nous allons maintenant autoriser uniquement un adresse quelqu'un à se connecter sur votre machine a travers port 80.

Pour cela, nous allons insérer une règle pour cet adresse ***avant*** la règle qui **DROP** les paquets.

Il est important de la placer au début, car les règles sont exécutées dans l'ordre ou elles ont été placées dans la liste :

```
#iptables -I INPUT 2 -p tcp --dport 22 -s @ip -j ACCEPT
```

'-I INPUT 1' insère la règle en 2er dans la liste.
'-s' indique la source des paquets

Exemple: protection locale

Essayez de vous connecter sur votre machine depuis la machine dont l'adresse IP est autorisée, et depuis une autre machine !

N'hésitez pas à utiliser la commande 'iptables -nvL' pour voir les compteurs de paquets qui évoluent en fonction de vos tests ou la commande **watch** :

```
# watch -n 1 'iptables -nvL'
```

NB: vous pouvez utiliser la commande 'iptables -Z' pour remettre à zéro les compteurs de paquets et d'octets

Nous avons vu comment ajouter et insérer des règles, on peut aussi les effacer en utilisant l'option -D, par exemple en faisant :

```
# iptables -D INPUT 2
```

vous effacez la première règle de la chaîne INPUT.

Il peut être pratique de voir les numéros de chaque règle lorsqu'on liste les règles, cela se fait avec l'option line-numbers :

```
# iptables -nvL --line-numbers
```

Exemple: création d'une chaîne

Imaginons, par exemple, que l'on veuille gérer plusieurs règles pour contrôler l'accès au service **SSH**. On peut « ranger » toutes les règles dans une chaîne :

```
# iptables -N SSH
```

Il faut ensuite indiquer à la chaîne INPUT que tout ce qui concerne **SSH** doit être transmis à cette chaîne :

```
# iptables -A INPUT -p tcp --dport 22 -j SSH
```

Enfin, on ajoute nos règles de filtrages :

- on accepte que le voisin se connecte

```
# iptables -A ssh -s @ip -j ACCEPT
```

- on interdit au reste du monde de se connecter

```
# iptables -A ssh -j DROP
```

Exemple: NAT

Nous allons créer une **passerelle**, de la sorte, les machines de la salle vont pouvoir accéder au reste du réseau (Internet) à travers cette dernière.

Étape 1 : il faut activer le 'forwarding':

De manière temporaire :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Ou de manière permanente:

```
# vi /etc/sysctl
# sysctl -p
```

Étape 2: activez la fonction de **masquerading** pour tout le réseau :

```
# iptables -t nat -A POSTROUTING -s @NET -j MASQUERADE
```

Remplacer @NET par l'adresse du réseau

Étape 3 : vider la table FORWARD

```
# iptables -F FORWARD
# iptables -P FORWARD ACCEPT (optionnel sur CentOS)
```

Étape 4 : utilisez « ***tcpdump*** » pour 'voir' le NAT :

Sur un terminal : `# tcpdump -i eth0 -nn -vv proto 1`
Sur un autre terminal : `# tcpdump -i eth1 -nn -vv proto 1`

Exemple: PAT

Étape 1: utilisez le PAT pour rediriger le trafic HTTP sur un hôte

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to
    @hôte:port
```

Remplacer @hôte par l'adresse de l'hôte HTTP et port par le port destination (si besoin)

Attention : la redirection du trafic HTTP sur toutes les interfaces va engendrer des problèmes en sortie, pour cela spécifiez toujours l'interface d'arrivée !

```
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to
    @hôte:port
```