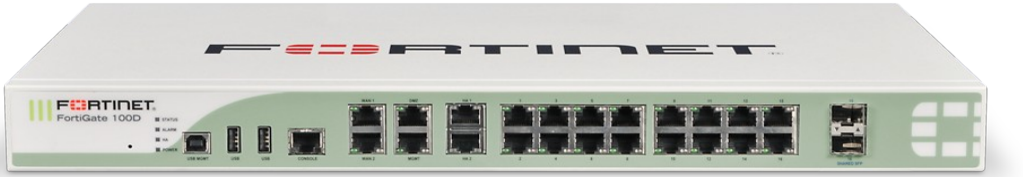
Réseaux
Les pare-feux

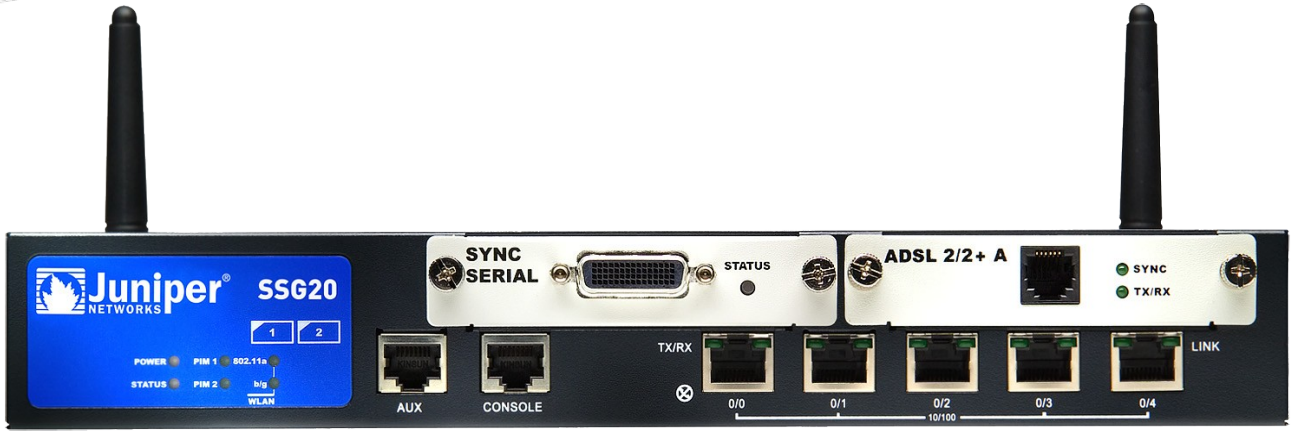
Sommaire

Généralités

Généralités

- Il existe des pare-feux matériels...





- ...et logiciels

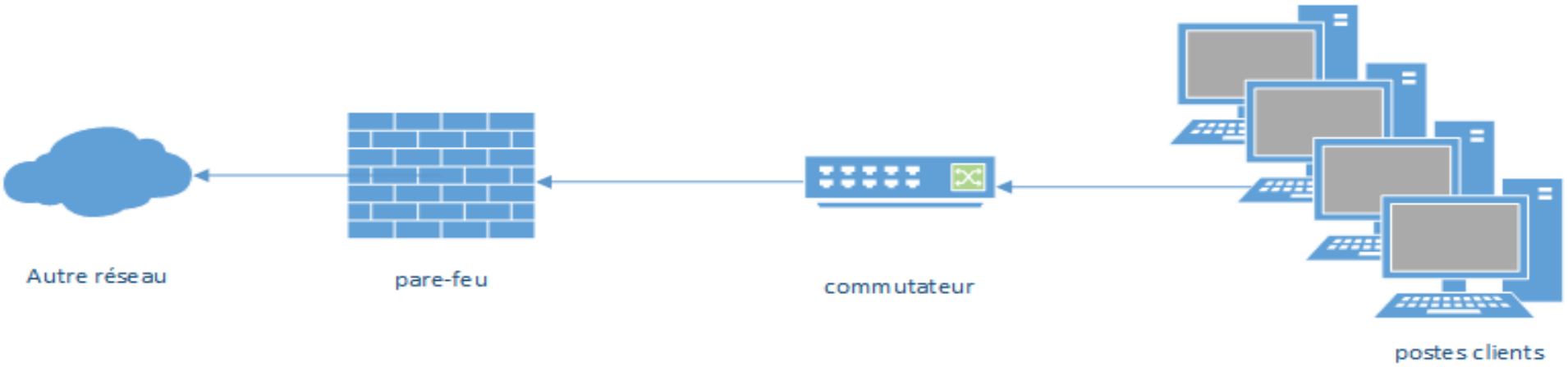
Ils sont utilisés pour protéger un réseau :

- **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 près de machines sensibles
- **Notion de confinement** : empêcher les utilisateurs internes de sortir du domaine protégé sauf par un point précis.

Généralement ils concernent les couches basses (IP/TCP/UDP), mais peuvent également comprendre la couche application (HTTP, FTP, SMTP…)

Ce que peut faire un pare-feu :

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



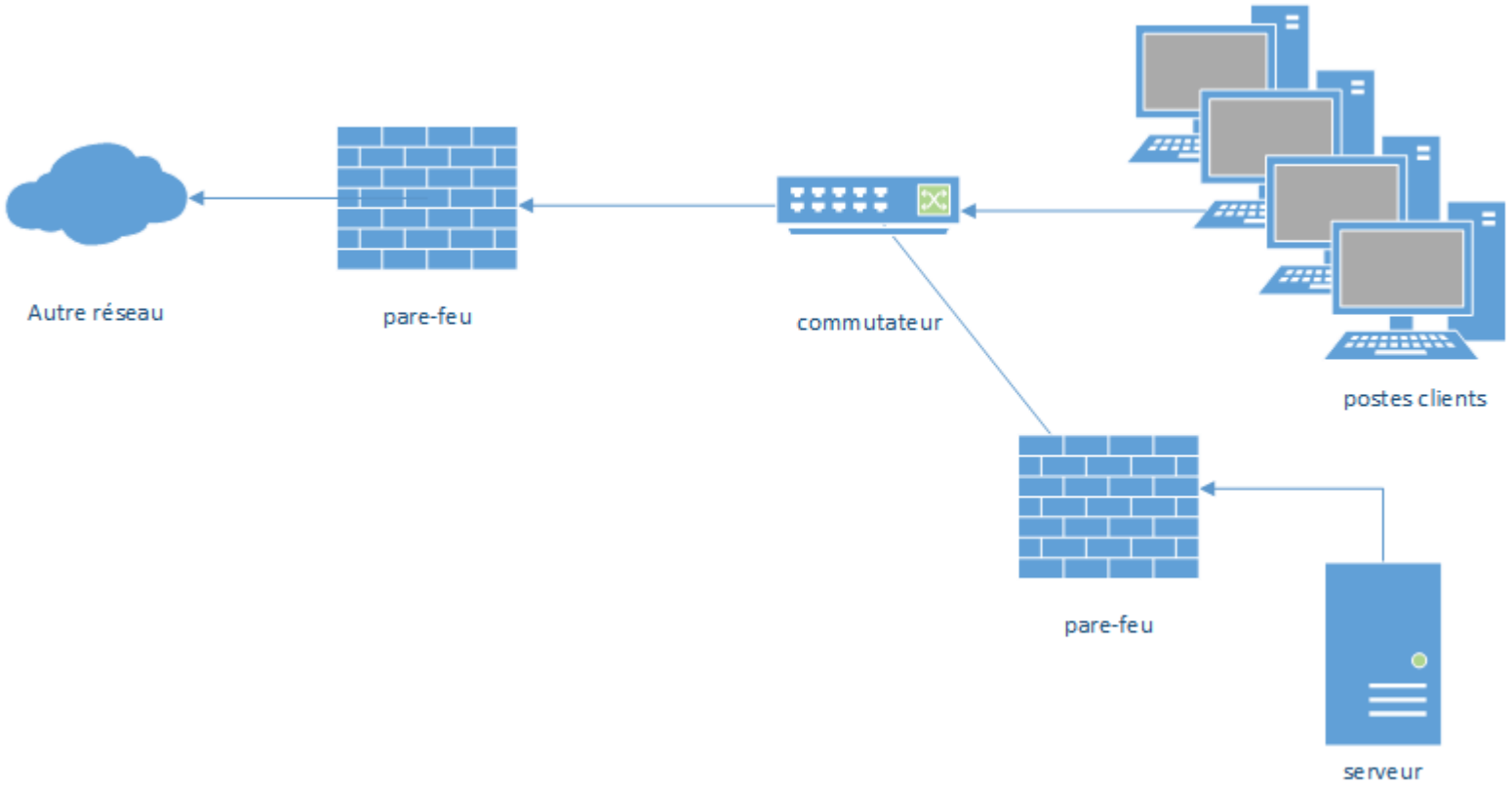
Ce que peut faire un pare-feu :

- 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-feu :

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



Ce que ne peut pas faire un pare-feu :

- 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 (« stateful inspection ») ou dynamique	OSI 3 / 4
• Application (relais applicatifs – « application proxys »)…	OSI 7

Fonctionnalités :

- Principale → filtrage IP (TCP/UDP)

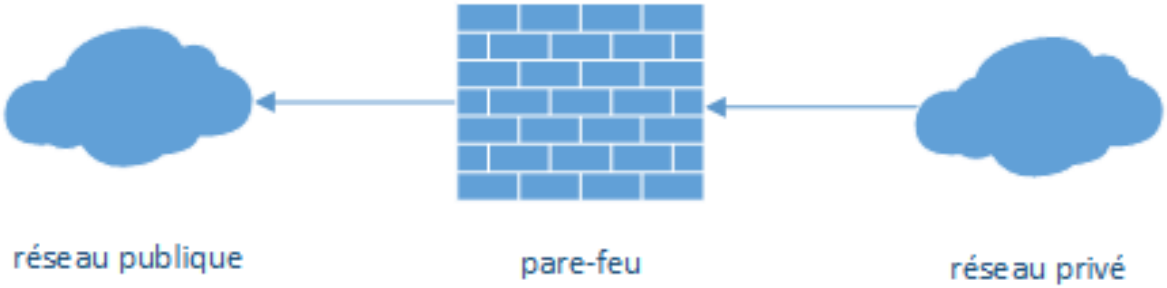
HTTP DNS

DHCP ? FTP

SSH

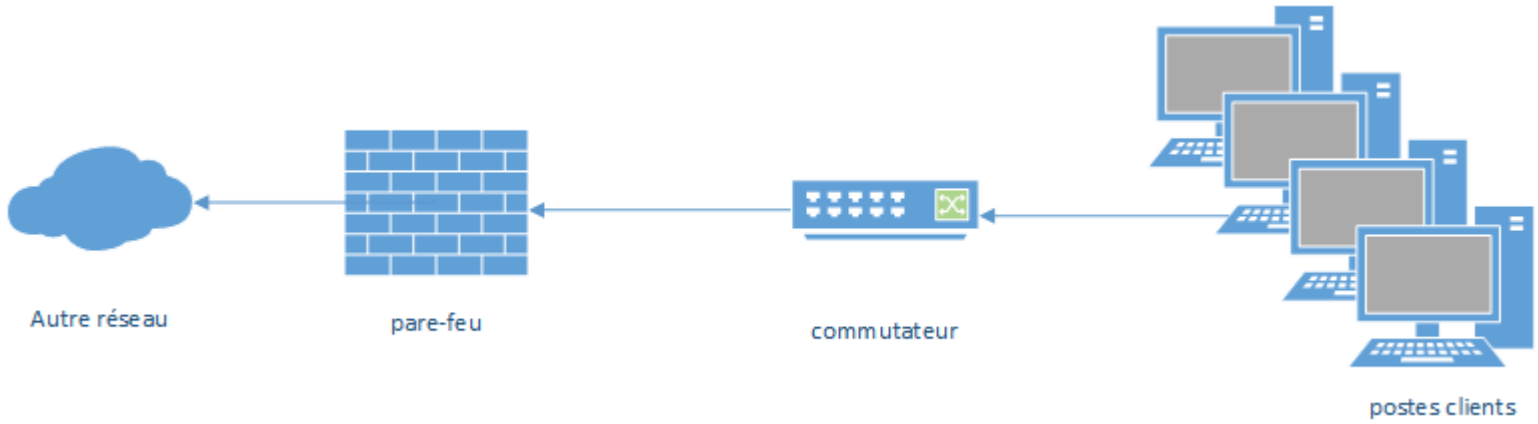
HTTPS

- Supplémentaire → traduction d'adresses et de ports.



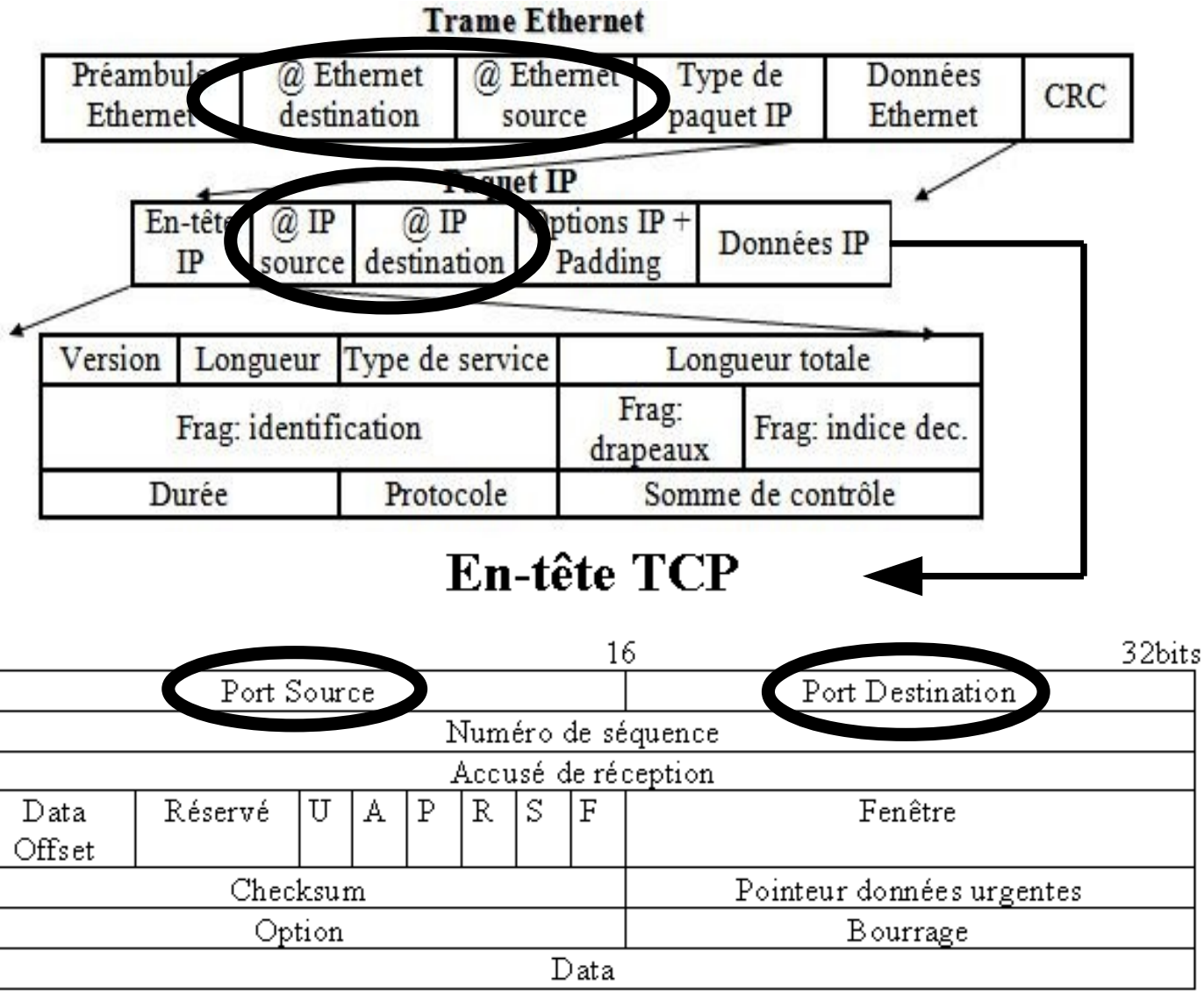
Filtrage IP (TCP/UDP)

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



- Intérêt → sécuriser de manière globale !

Filtrage sur les en-têtes :



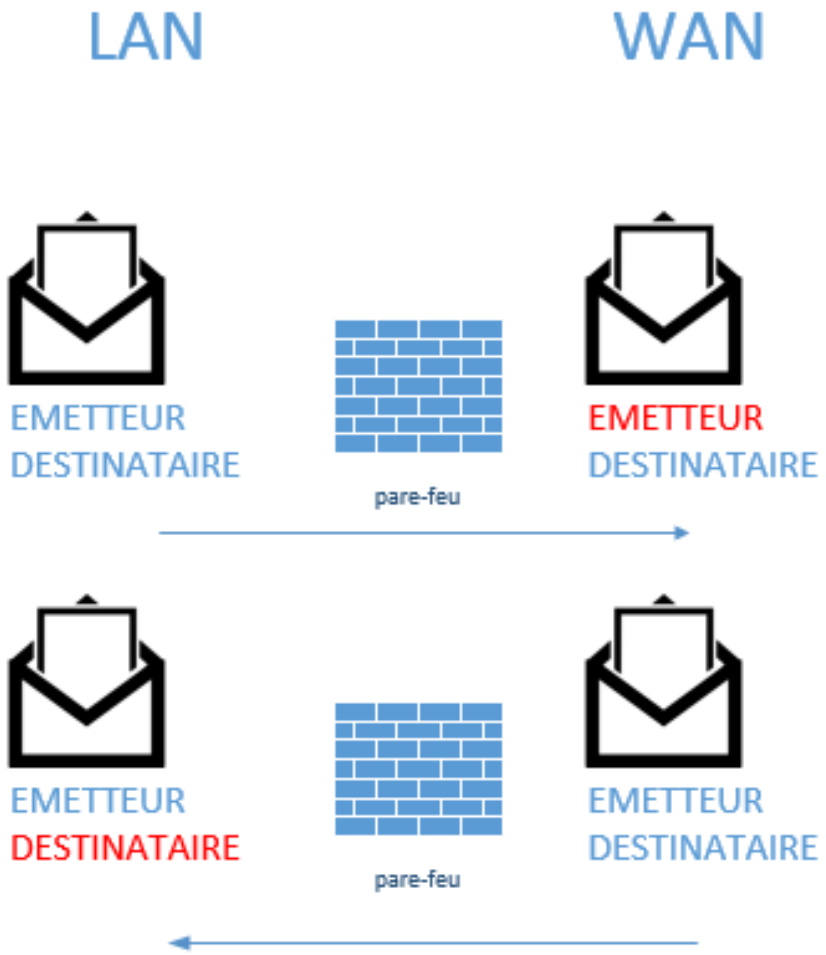
OSI 2

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)

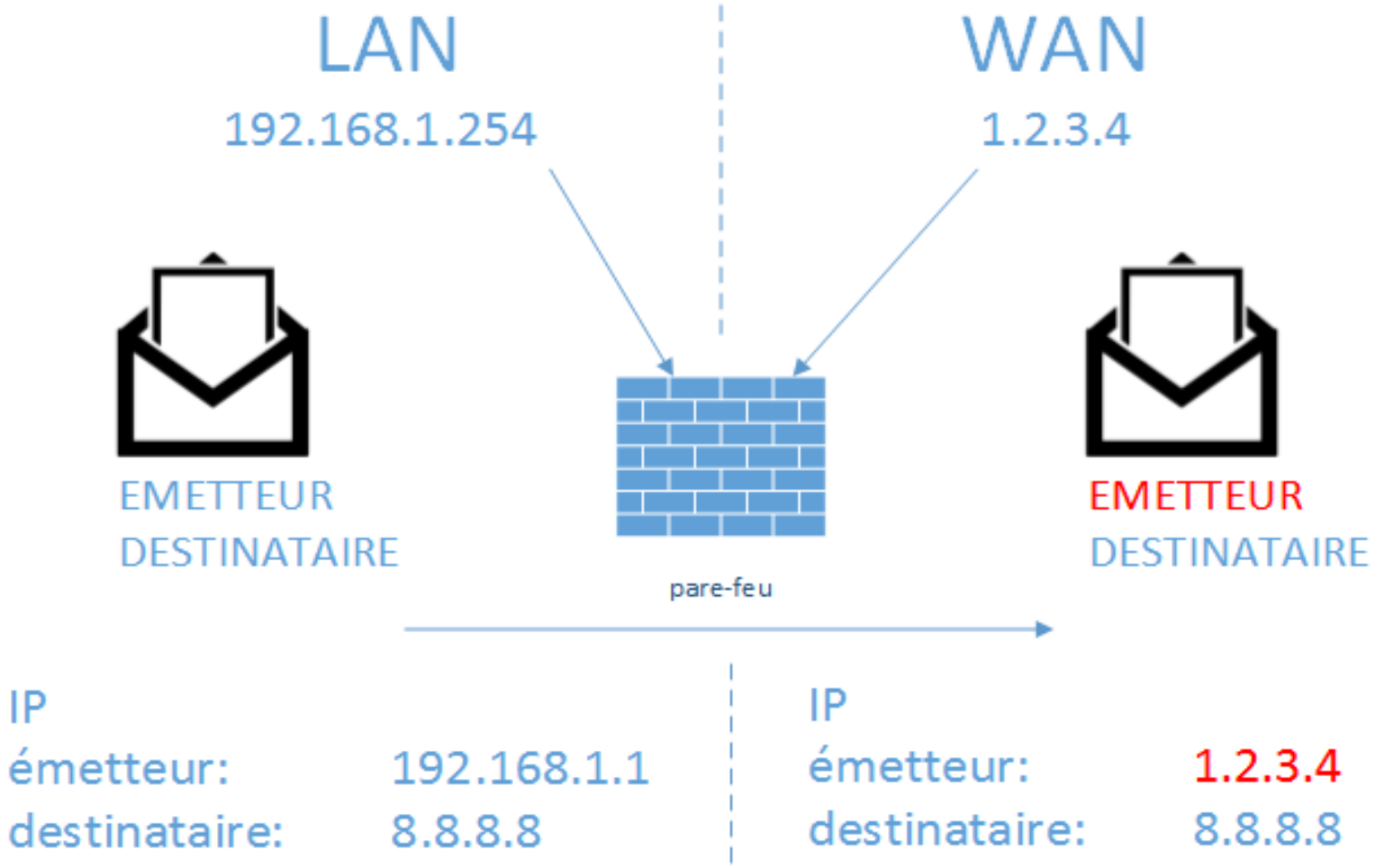
Traduction d'adresses :

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

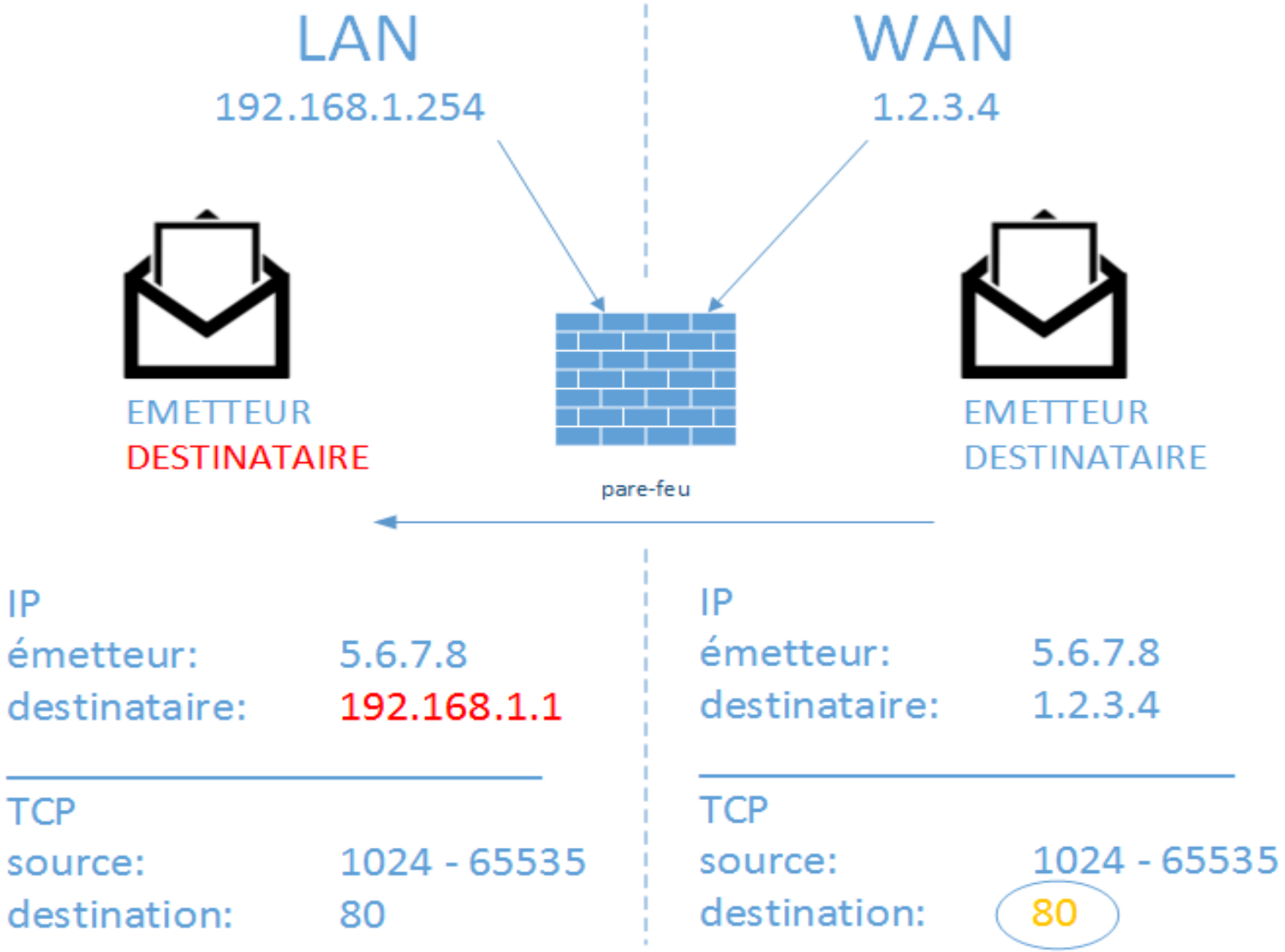
- **Bidirectionnelle** : traduction d'une adresse publique en une adresse privée et réciproquement

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

Translation d'adresses uni-directionnelles (NAT) :



Traduction de ports (PAT) :



Avantages :

- gestion de la sécurité concentrée
- capacité d'audit du trafic réseau
- concentrer la maintenance sur une machine 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'adresses, …)

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

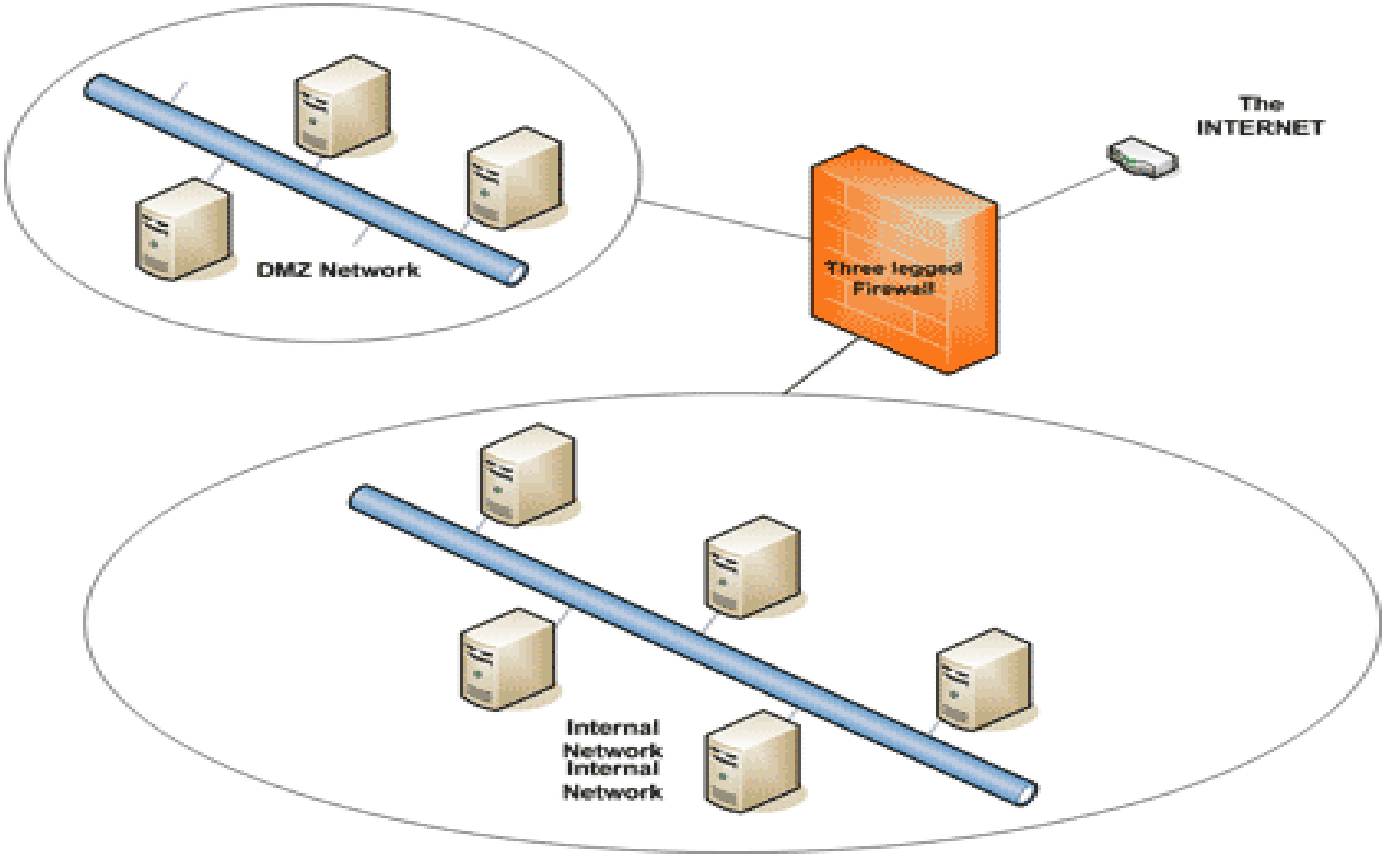
Packet Filter (Daniel Hartmeier)

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

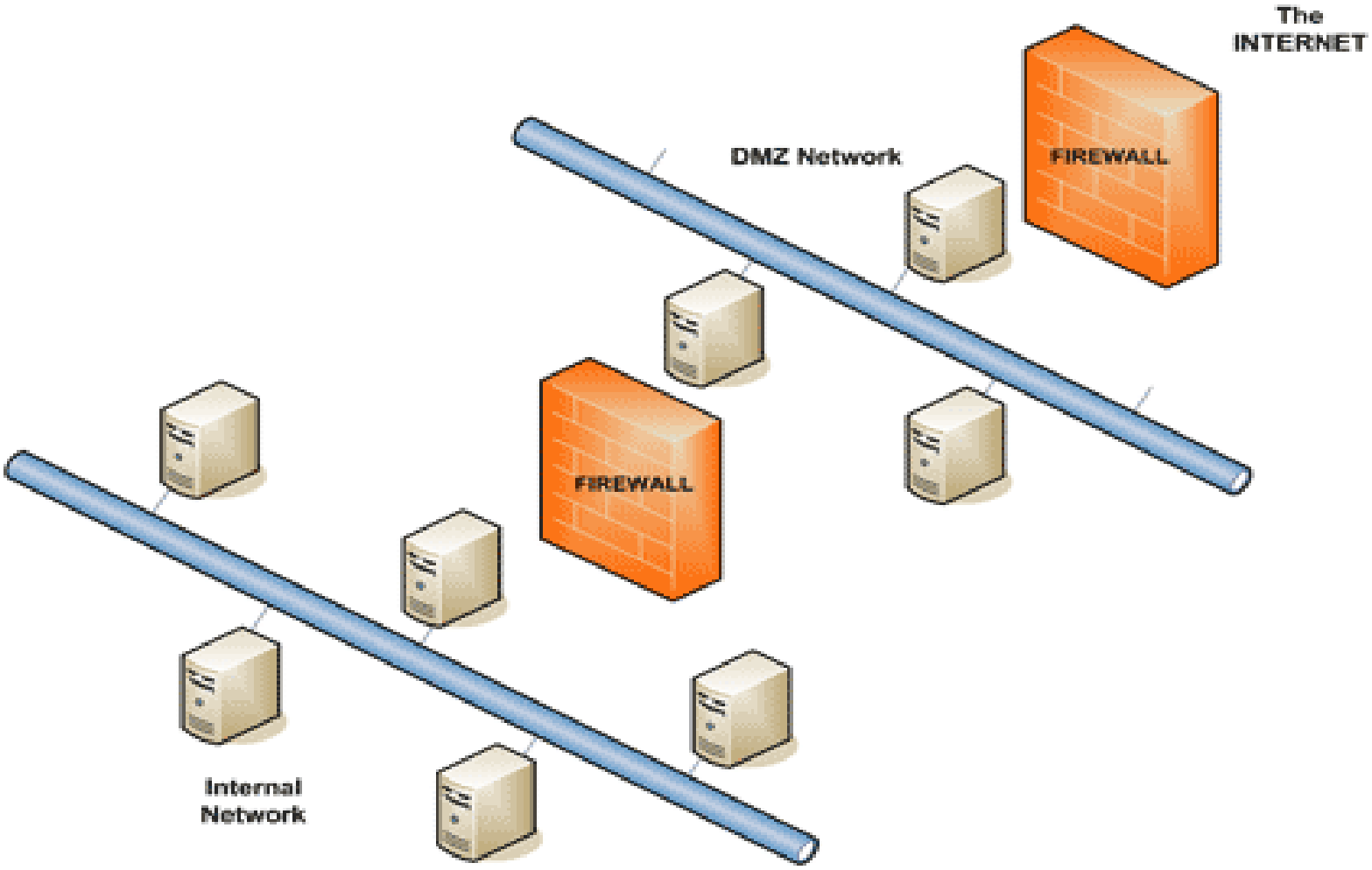
Architecture / utilisation

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)



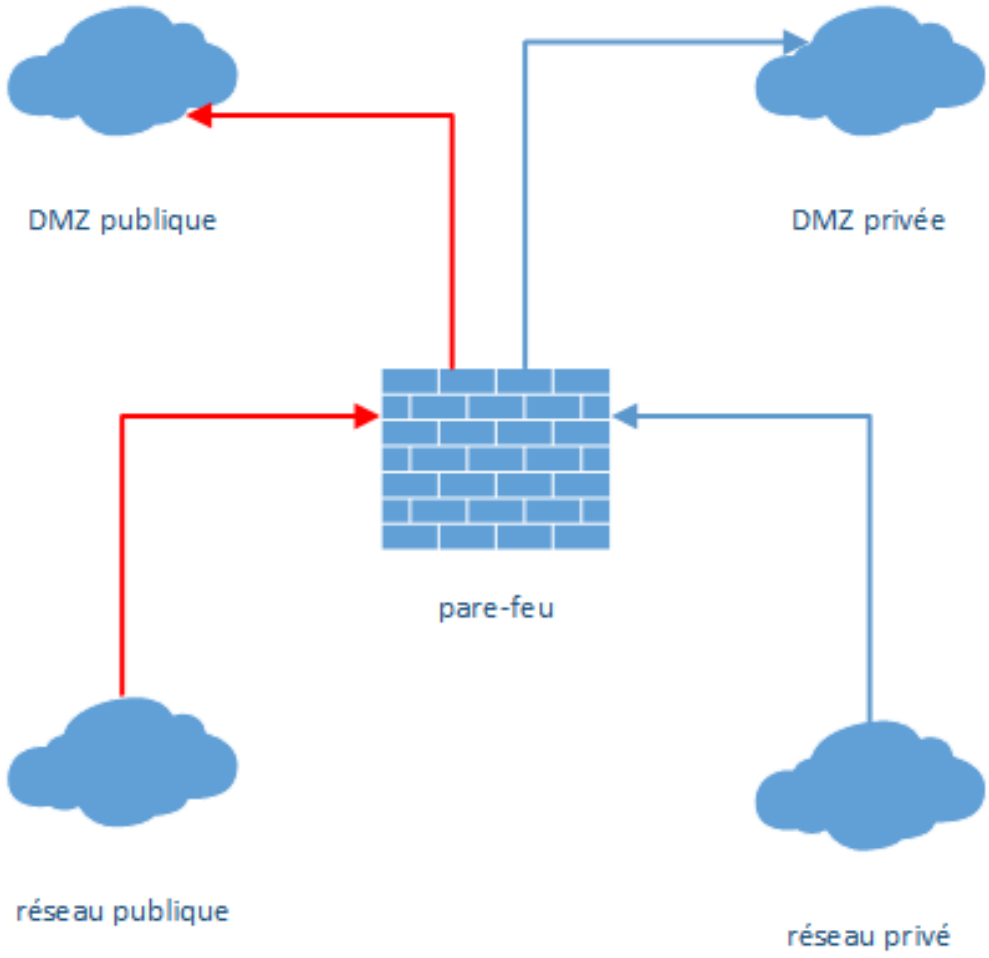
Il existe deux types de DMZ : publiques et privées

Les flux autorisés sont :

WAN → DMZ publique
(DMZ publique → WAN)

LAN → DMZ privée
(DMZ privée → LAN)

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



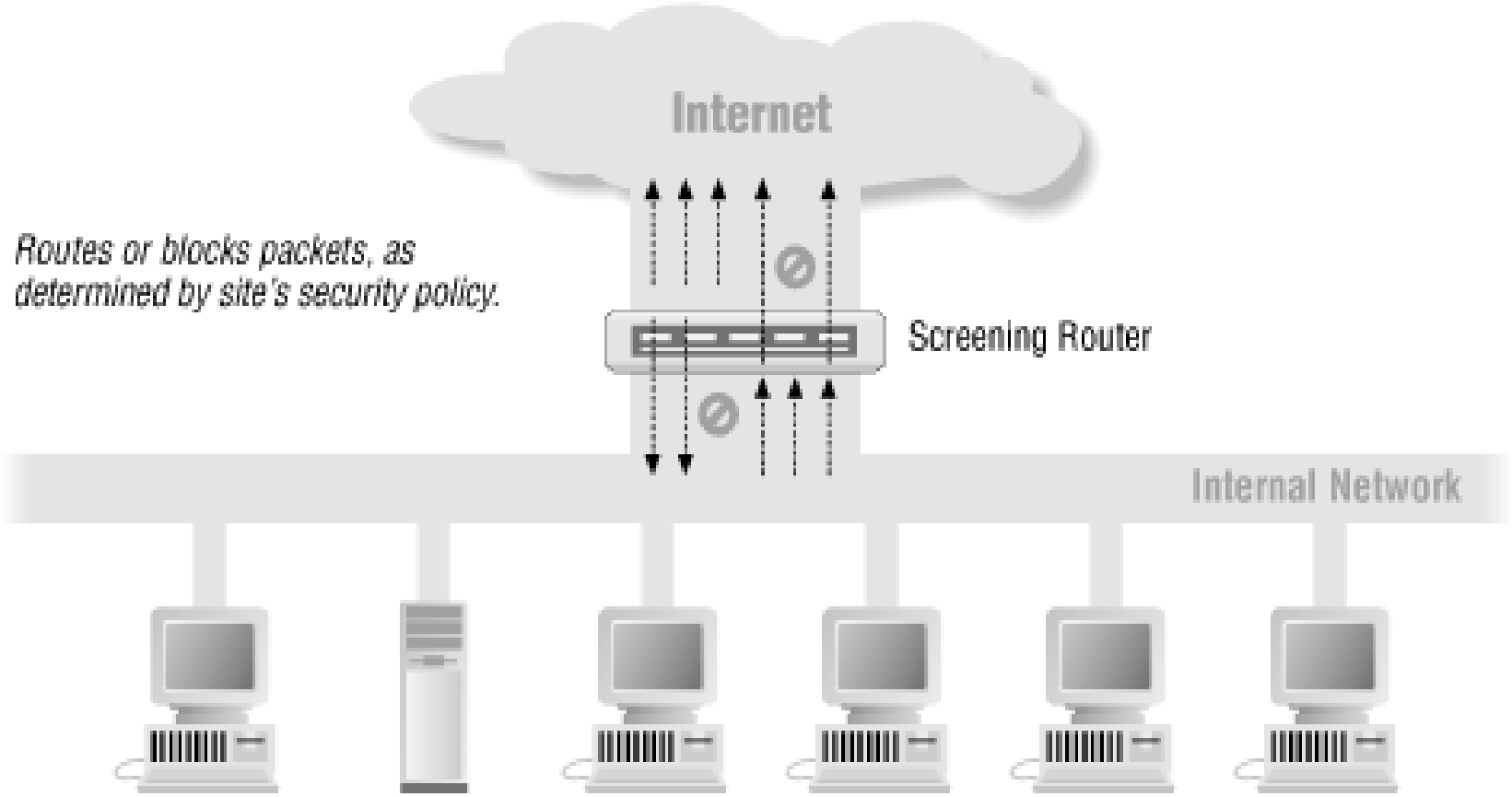
Que mettre dans la DMZ ?

- Publique : les services joignables depuis Internet (HTTP, DNS, …)
- Privée : les services joignables 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.

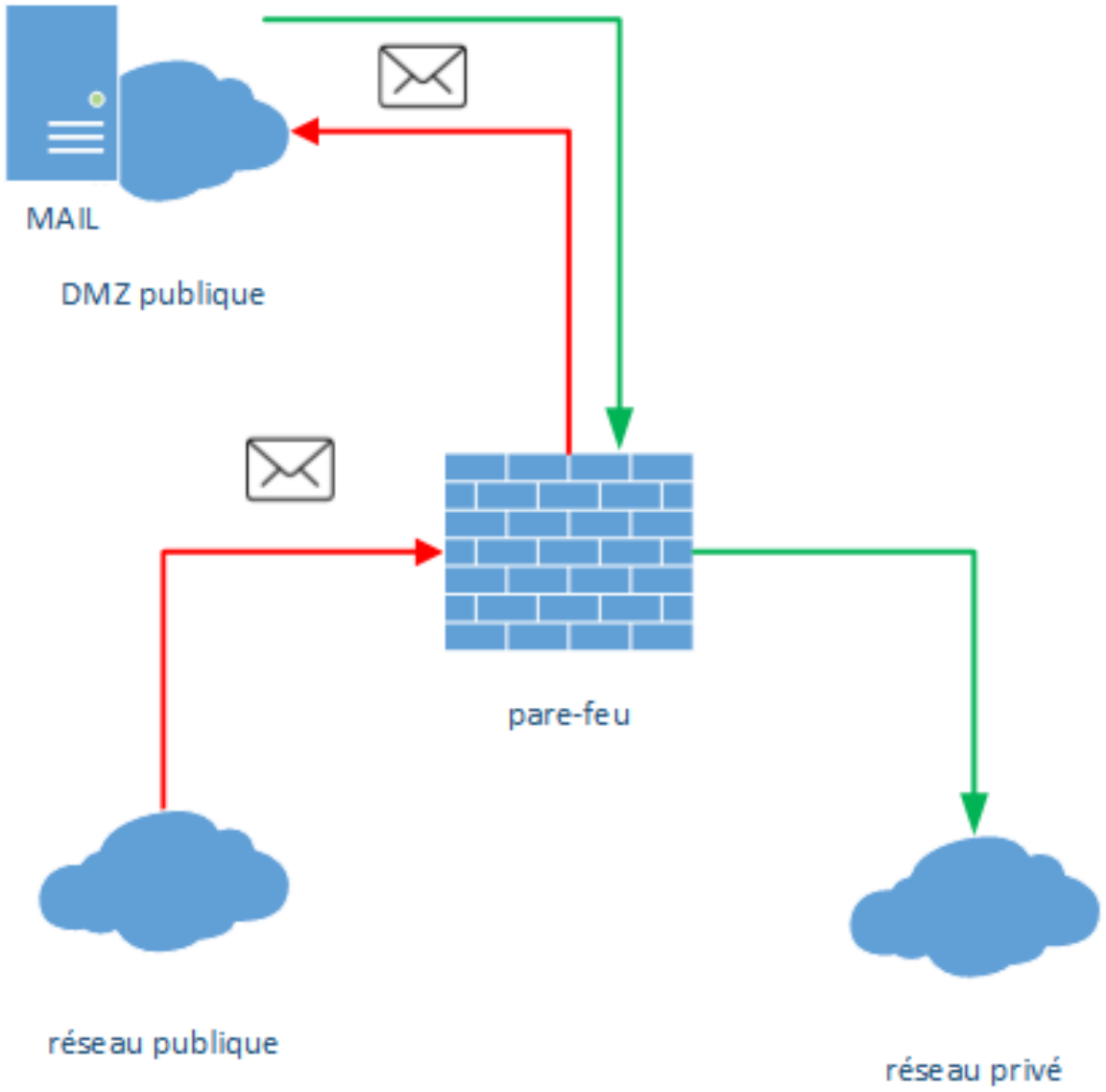
Architecture de pare-feu avec routeur filtrant : « screening router »



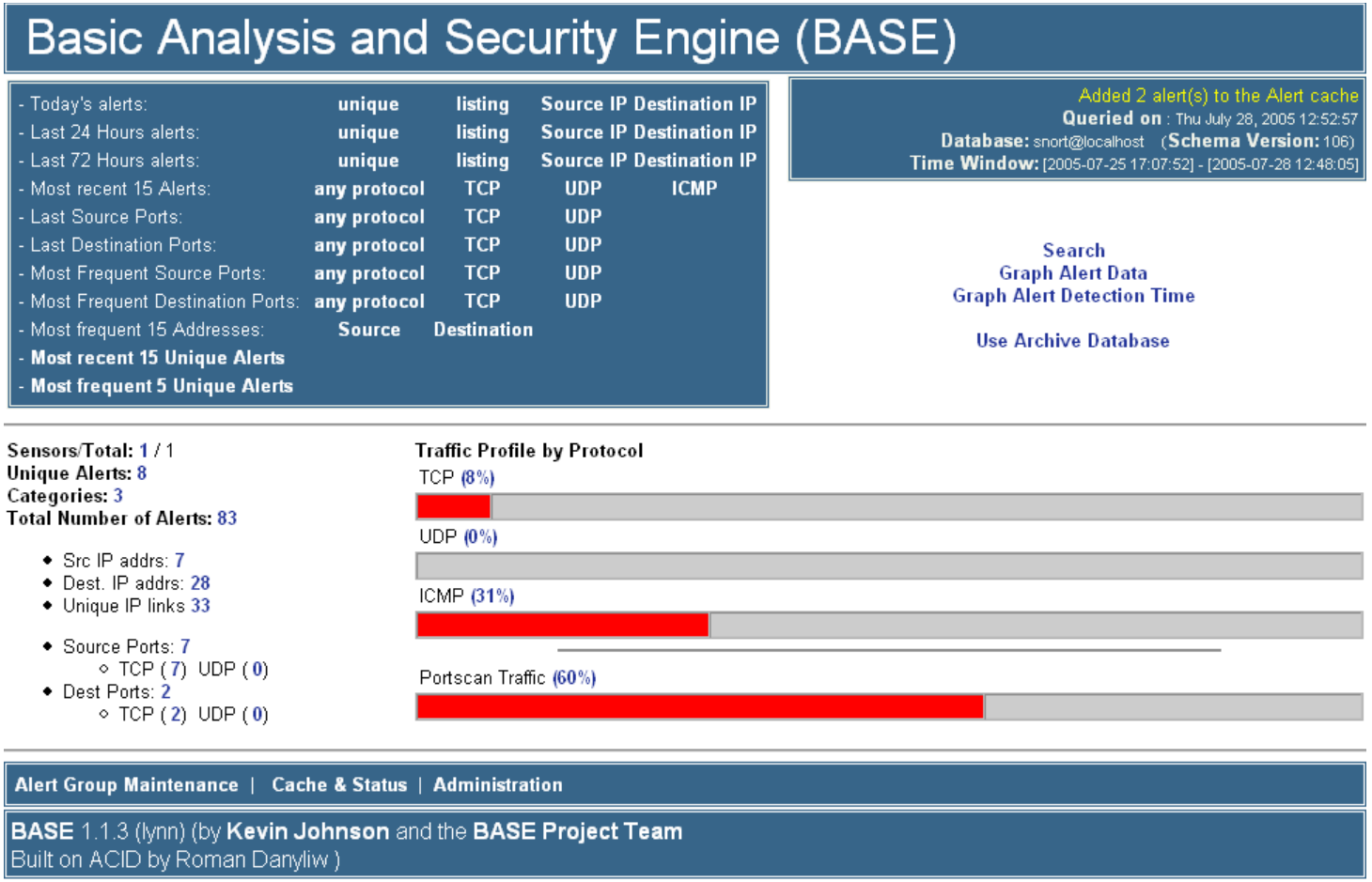
Bastion (hôte exposé au réseau externe)

- 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)

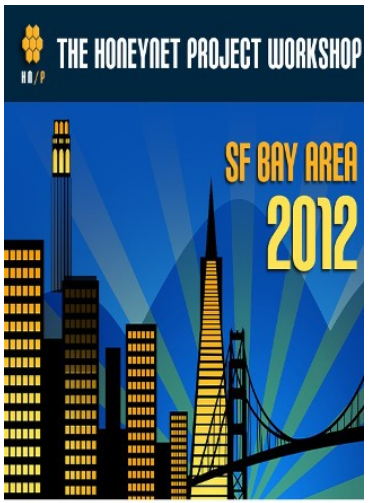
Bastion (hôte exposé au réseau externe)

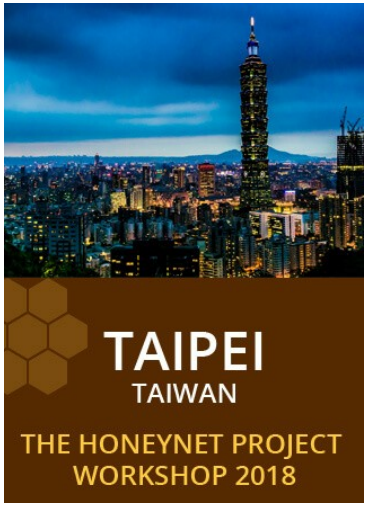


Le bastion peut servir dans la détection d'intrusions, il analyse des communications pour détecter des attaques : IDS (« Intrusion Detection System »).



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





Filtrage

Filtrage : standard

Il existe différents critères de filtrage :

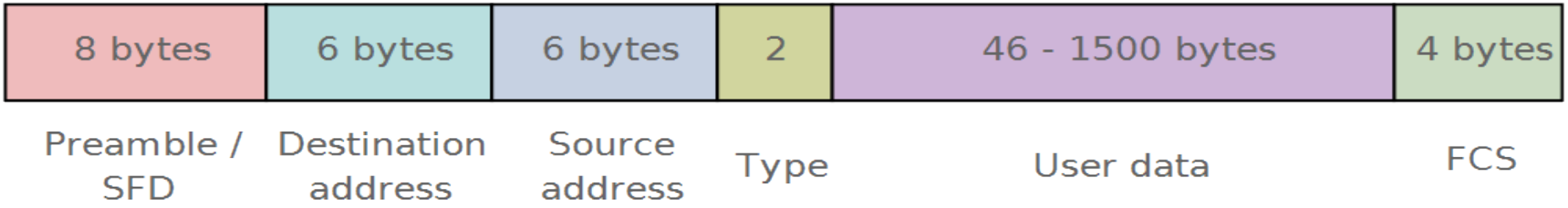
- interface réseau :
  - entrée
  - sortie

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group d
ault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
3: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast stat
UP group default qlen 1000
    link/ether e0:cb:4e:df:83:dd brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.253/24 brd 192.168.3.255 scope global noprefixroute enp4
       valid_lft forever preferred_lft forever
4: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast stat
UP group default qlen 1000
    link/ether 68:05:ca:1e:1b:86 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.254/24 brd 192.168.1.255 scope global noprefixroute enp2
       valid_lft forever preferred_lft forever
5: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP g
up default qlen 1000
    link/ether 68:05:ca:1e:1b:86 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::6a05:caff:fe1e:1b86/64 scope link
       valid_lft forever preferred_lft forever
6: enp2s0.100@enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqu
e master br0 state UP group default qlen 1000
    link/ether 68:05:ca:1e:1b:86 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::6a05:caff:fe1e:1b86/64 scope link
       valid_lft forever preferred_lft forever
7: enp4s0.100@enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqu
e master br0 state UP group default qlen 1000
    link/ether e0:cb:4e:df:83:dd brd ff:ff:ff:ff:ff:ff
    inet6 fe80::e2cb:4eff:fedf:83dd/64 scope link
       valid_lft forever preferred_lft forever
8: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fas
state UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::4775:1d08:7df7:7ff2/64 scope link flags 800
       valid_lft forever preferred_lft forever
9: enp0s6f1u2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
ate UP group default qlen 1000
    link/ether 00:0e:c6:c2:d3:10 brd ff:ff:ff:ff:ff:ff
```

Filtrage : standard

Il existe différents critères de filtrage :

- adresses mac (source / destination)
- protocoles de niveau 3 (IP, ICMP, …)

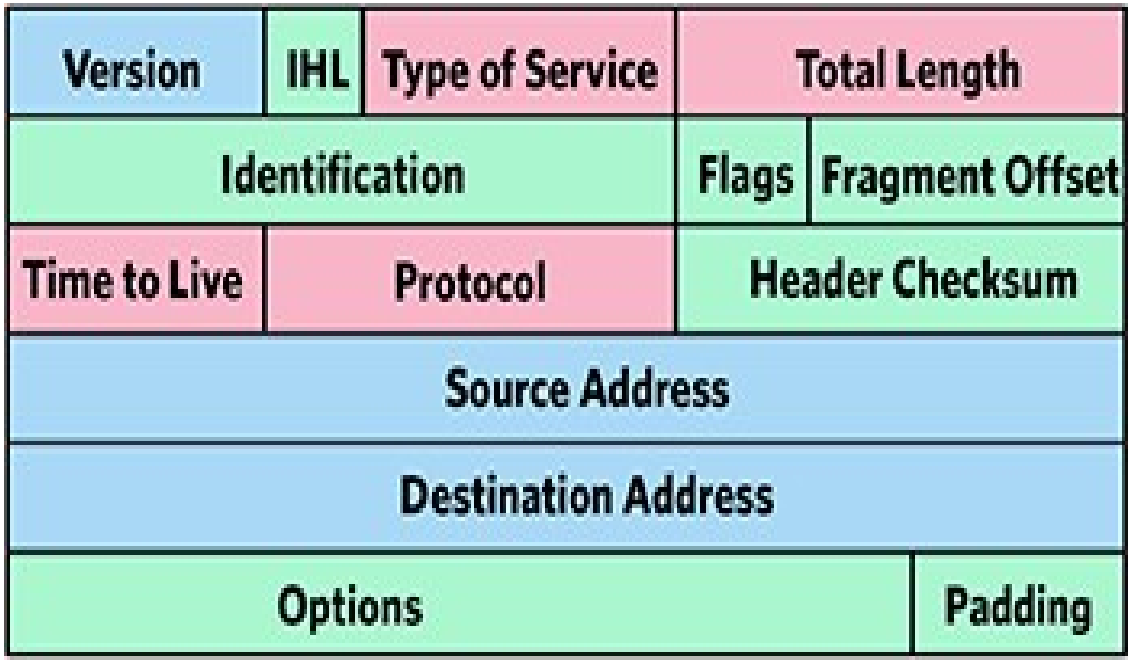


Filtrage : standard

Il existe différents critères de filtrage :

- adresses IP (source / destination) :
  - hôte (eg. 192.168.1.1)
  - sous-réseau (eg. 192.168.1.0/24)
- protocoles de niveau 4 (TCP, UDP, …)
- champs de l'en-tête IP (TOS / TTL, codes ICMP, …)

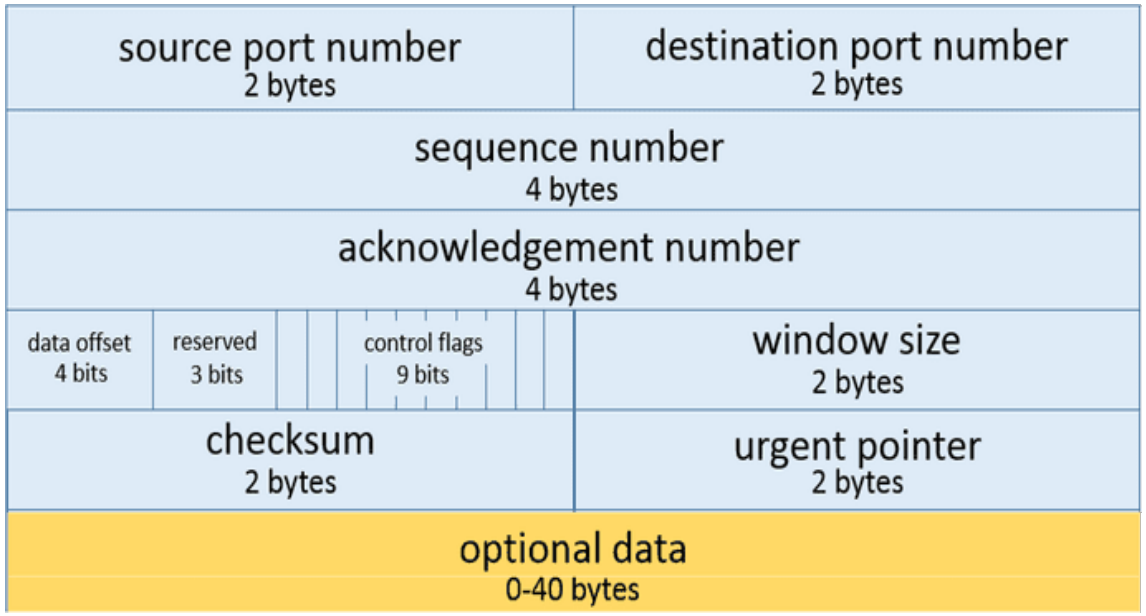
…



Filtrage : standard

Il existe différents critères de filtrage :

- ports source et destination (TCP et UDP)
- drapeaux (TCP)
- …



entête TCP

entête UDP

Filtrage : standard

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

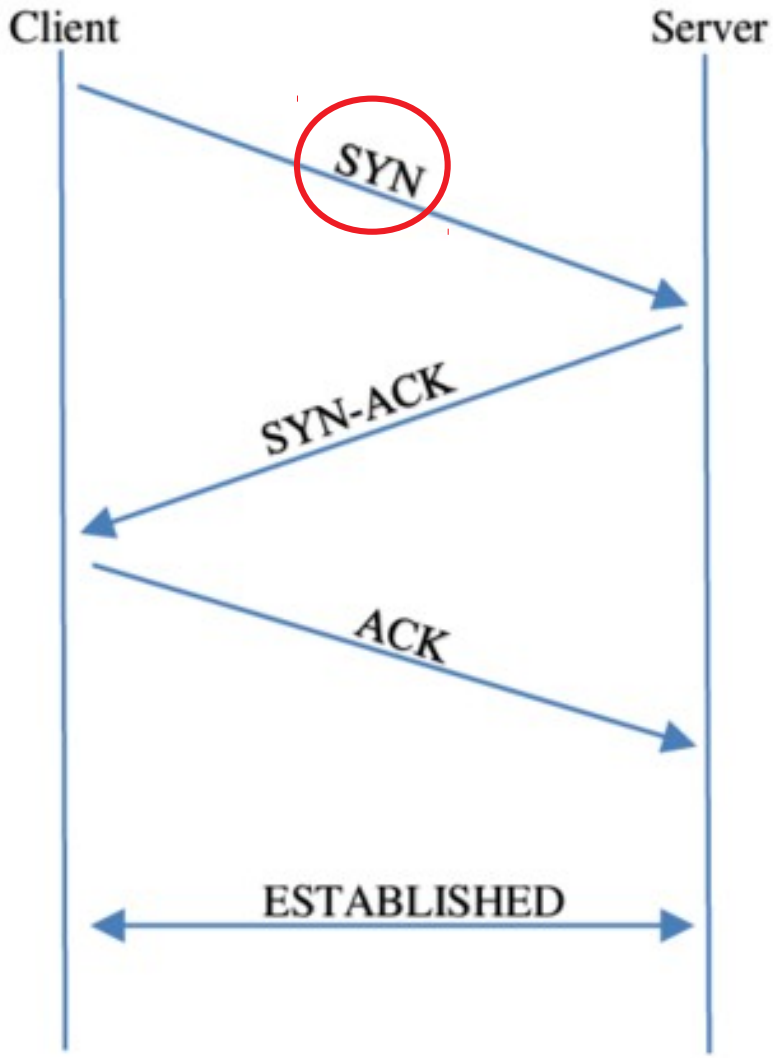
Le filtrage « stateful » ou filtrage dynamique, permet de suivre les états des connexions en cours .

Seuls des paquets correspondants à un état pré-existant sont acceptés.

Intérêt :

- simplifier l'écriture des règles de filtrage
- améliorer la sécurité, en n'autorisant que le trafic licite

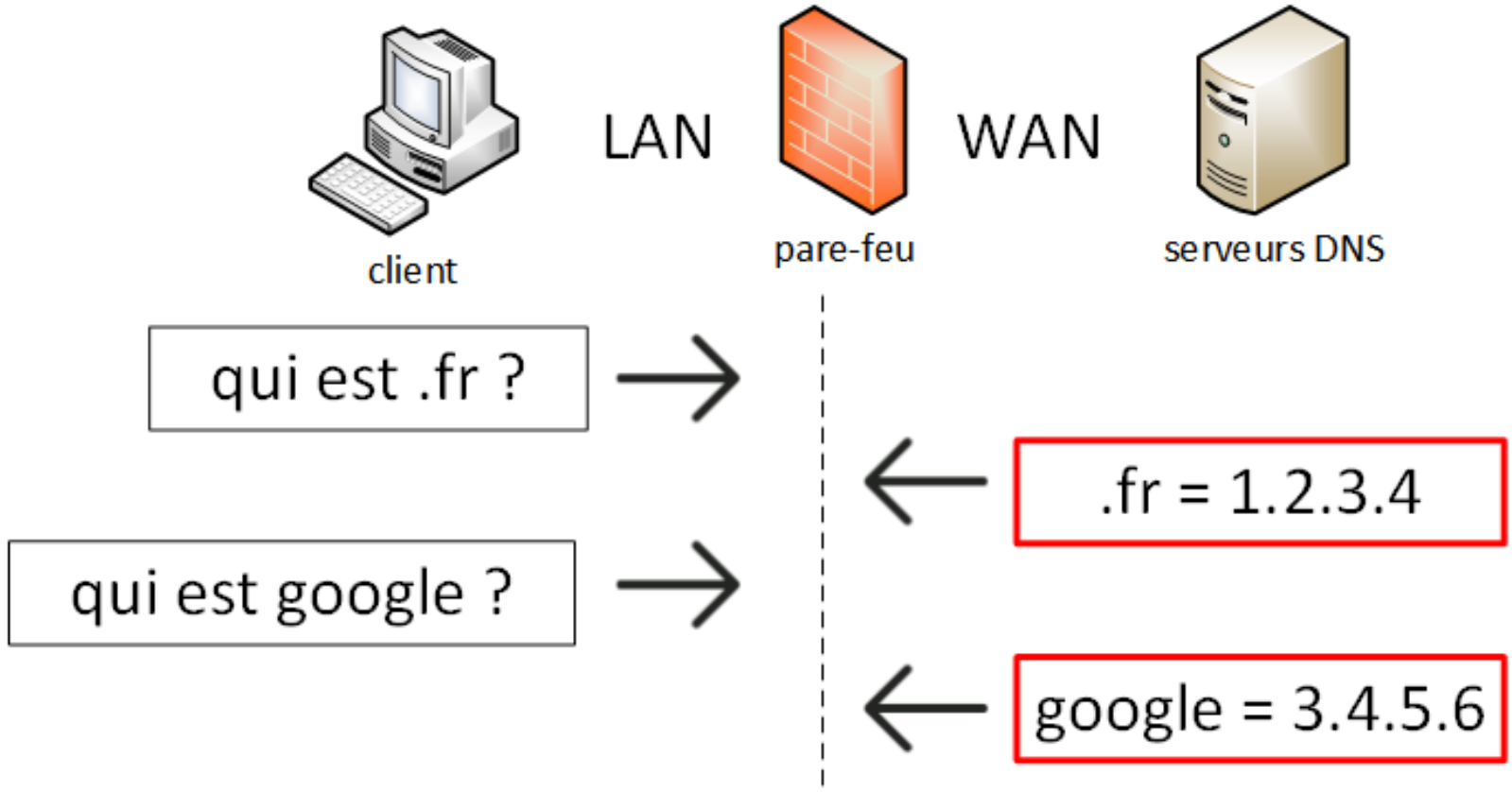
Cela permet en TCP de contrôler un segment appartenant à une connexion en cours



Three-Way Handshake

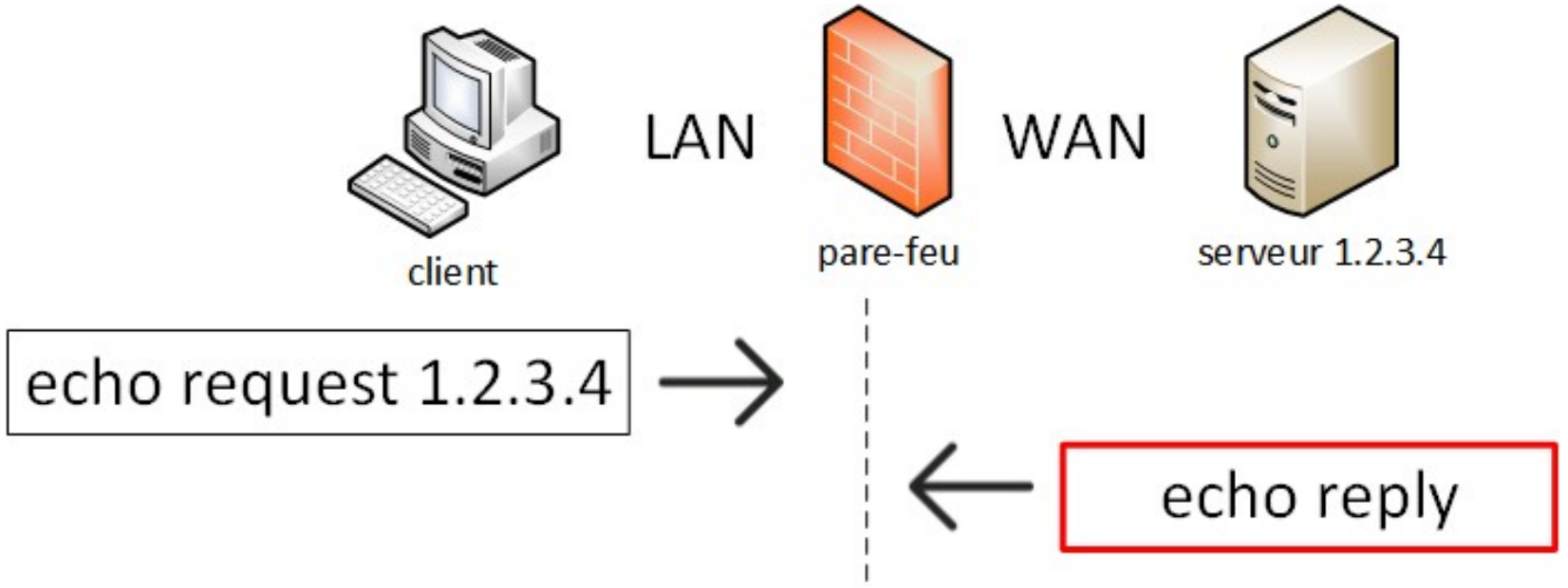
Cela permet en UDP d'autoriser un datagramme en réponse à un datagramme précédemment émis.

Exemple avec une requête DNS (UDP 53), les paquets en rouges sont dynamiquement acceptés :



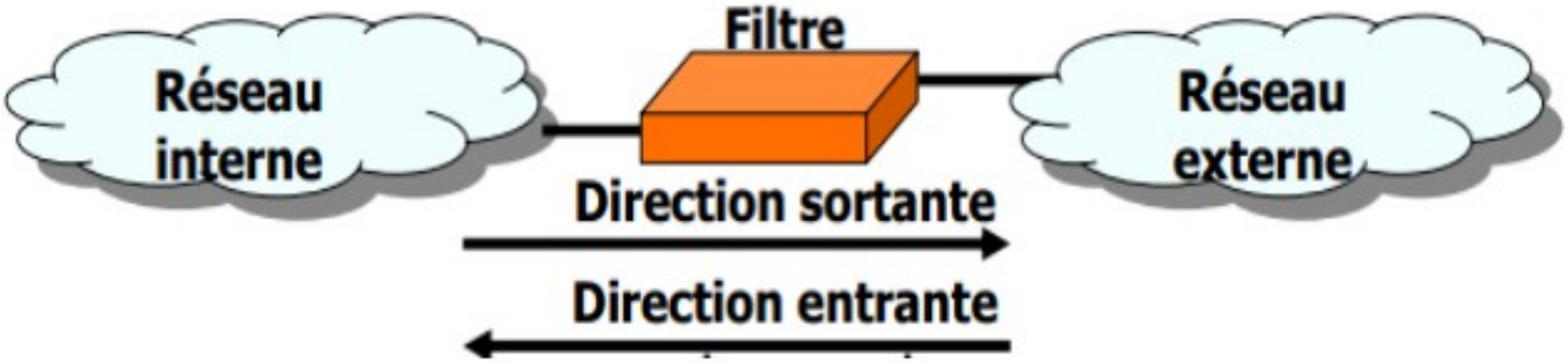
Cela permet en ICMP d'autoriser la réponse à un message ICMP émis.

Exemple avec un ping (echo request), les paquets en rouges sont dynamiquement acceptés :



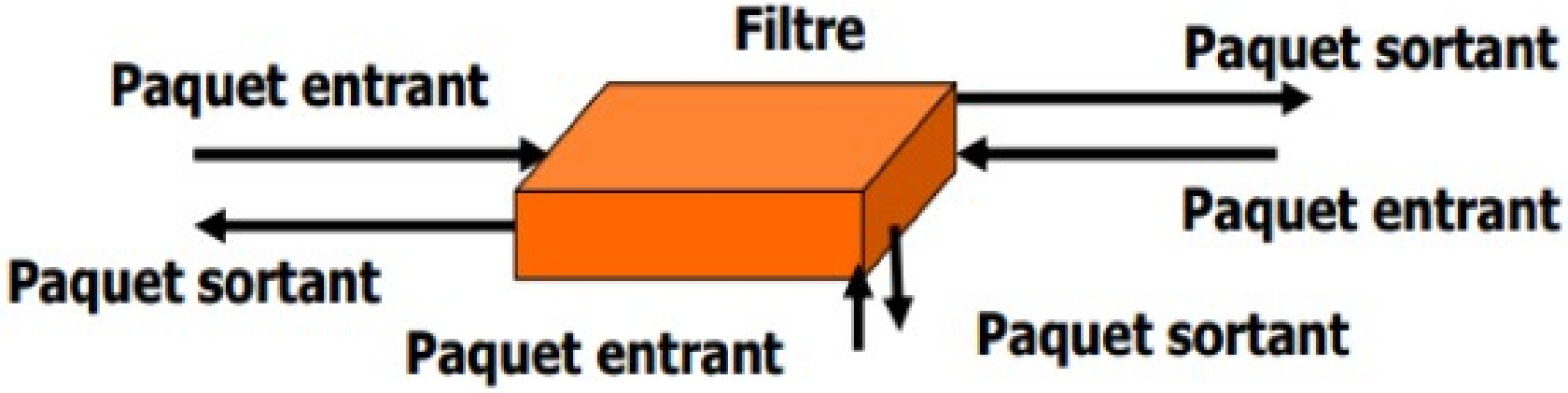
Les règles sont souvent exprimées selon une notion de direction : entrant ou sortant.

Par exemple en fonction du réseau à protéger



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

Par exemple en fonction de la position du filtrage dans un routeur selon l'interface



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.

Filtrage : politique par défaut

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 → Autoriser tout par défaut
Solution 2 → Interdire tout par défaut

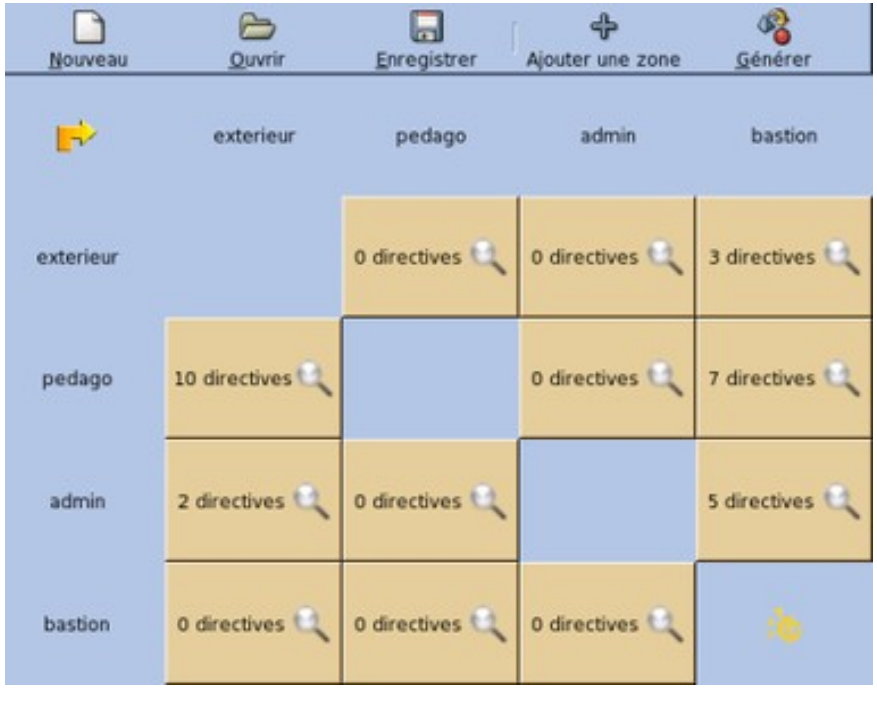
- Énoncer les règles :

Exemple 1 : Autoriser un hôte interne à recevoir des mails parce que c'est un serveur de mail (SMTP / POP / IMAP).

Exemple 2 : Interdire les mails en provenance d'un hôte externe précis parce qu'il est en liste noire (blacklist).

Traduire la politique de sécurité en règles précises et les réunir 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



Exemple du pare-feu de l'éducation nationale (AMON avec ERA) basé sur Linux Eole

Iptables

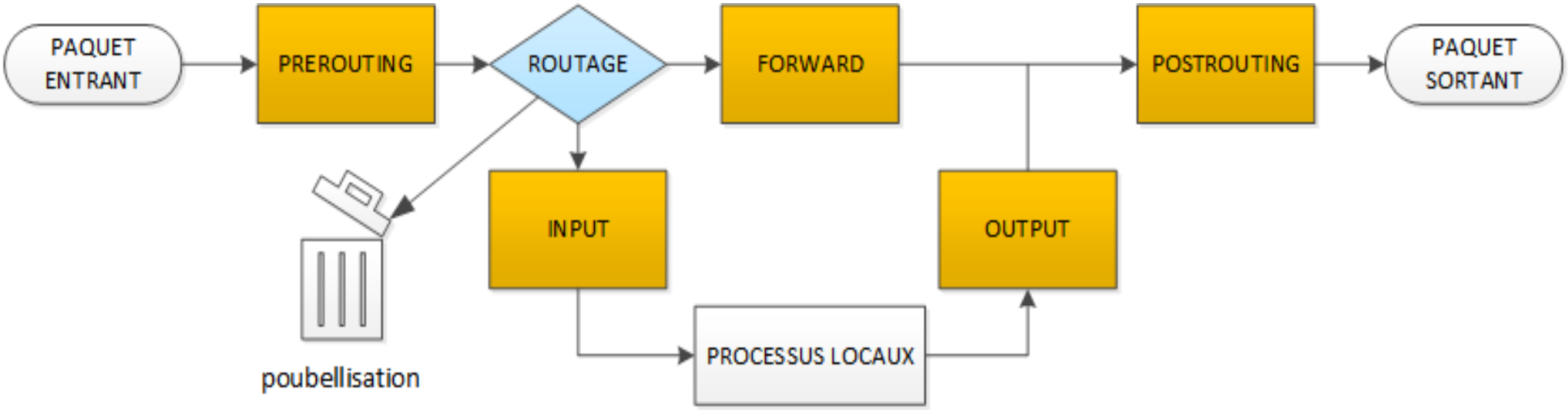
Le principe de fonctionnement est simple :

- lorsque la carte réseau reçoit un paquet celui-ci est transmis à la partie **Netfilter** du noyau
- **Netfilter** va ensuite étudier ce paquet (les en-têtes et le contenu)
- En se basant sur les règles que l'administrateur aura définies, il va choisir :
  - de laisser passer le paquet intact
  - de le modifier
  - de le transmettre à une autre machine
  - d'interdire son passage.

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

Réseaux : Les pare-feux

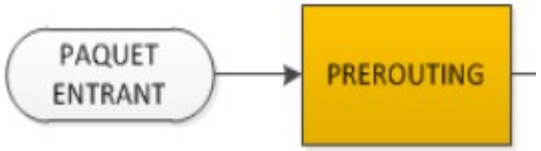
Iptables: fonctionnement



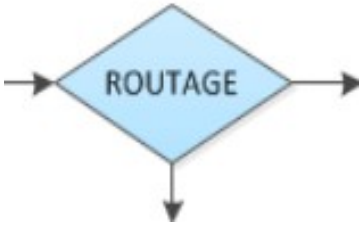
Tables	Chaînes
FILTER	INPUT, FORWARD, OUPUT
NAT	PREROUTING, POSTROUTING, OUTPUT
MANGLE	PREROUTING, INPUT, FORWARD, OUTPUT, PORTROUTING

Voici le chemin d'un paquet à **destination** de la machine:

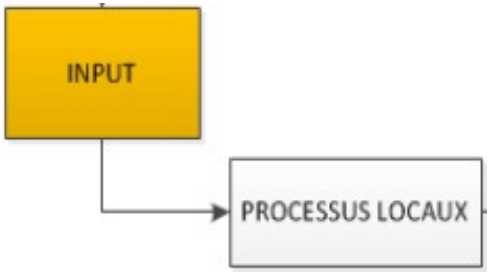
1. Netfilter vérifie les règles de la chaîne **PREROUTING** et effectue le changement d'adresse destination si une règle de **PAT** est trouvée



2. Le noyau regarde la destination du paquet



3. Si ce paquet est destiné à cette machine, le paquet est **filtré** par la chaîne INPUT et les processus qui attendent le paquet le recevront.



Sinon, si :

4. 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 'dropé' (supprimé).

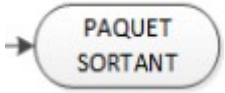
poubellisation

Sinon, si :

5. le **forwarding** est autorisé et que le paquet est destiné à un autre réseau, le paquet va directement à la chaîne **FORWARD**.

6. l'adresse source pourra être modifiée par la chaîne POSTROUTING

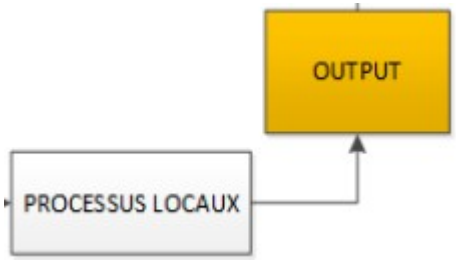
7. il sera envoyé vers le réseau de destination ;



Iptables: fonctionnement

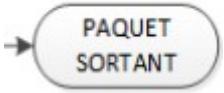
Voici le chemin d'un paquet **originaire** de la machine :

1. Netfilter vérifie la chaîne **OUTPUT** pour voir si le paquet a le droit de sortir

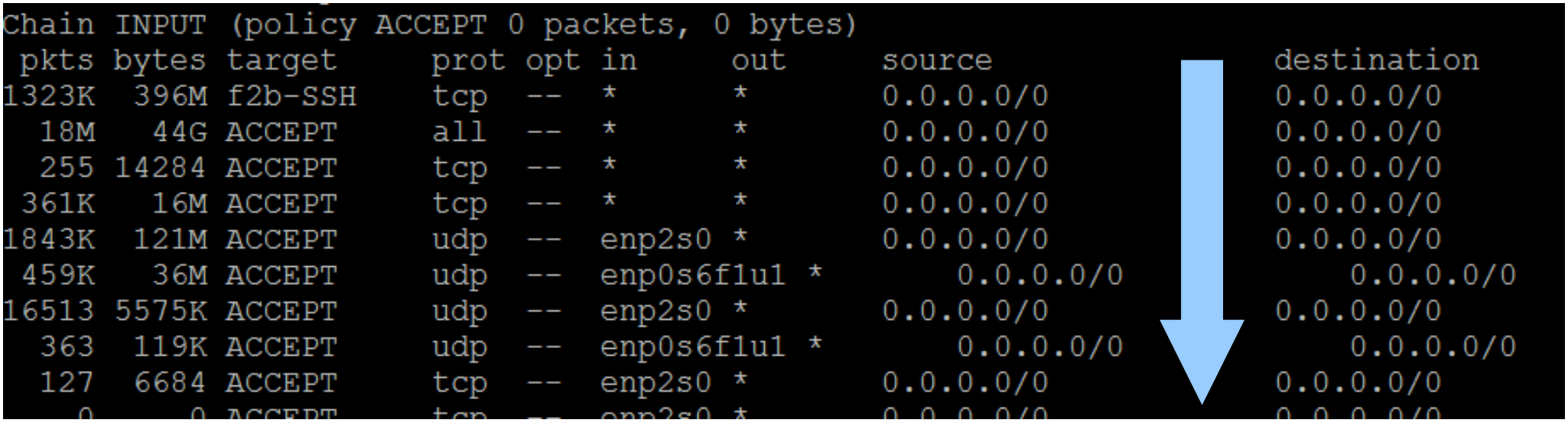


2. l'adresse source pourra être modifiée par la chaîne POSTROUTING

3. il sera envoyé vers le réseau de destination ;



Les règles de chaque chaîne dépendent de leur ordre (du haut vers le bas)



Iptables: fonctionnement

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.

Par exemple pour un paquet traversant (routé)

1. table **NAT**, chaîne **PREROUTING**
2. table **FILTER**, chaîne **FORWARD**
3. table **NAT**, chaîne **POSTROUTING**

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

Chacune de ces chaînes permet de « sauter sur une cible » (jump at target), ci-dessous un exemple pour **FILTER** :

- **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**

pkts	bytes	target	prot	opt	in	out
1323K	396M	f2b-SSH	tcp	--	*	*
18M	44G	ACCEPT	all	--	*	*
255	14284	ACCEPT	tcp	--	*	*
361K	16M	ACCEPT	tcp	--	*	*
1843K	121M	ACCEPT	udp	--	enp2s0	*
459K	36M	ACCEPT	udp	--	enp0s6f1u1	*
16513	5575K	ACCEPT	udp	--	enp2s0	*
363	119K	ACCEPT	udp	--	enp0s6f1u1	*
127	6684	ACCEPT	tcp	--	enp2s0	*
0	0	ACCEPT	tcp	--	enp2s0	*
0	0	ACCEPT	tcp	--	enp2s0	*
1	52	ACCEPT	tcp	--	enp2s0	*
341K	16M	ACCEPT	tcp	--	*	*
15558	2384K	ACCEPT	icmp	--	*	*
1126K	82M	ACCEPT	all	--	lo	*
6332	211K	ACCEPT	all	--	tun0	*
16M	1814M	DROP	all	--	*	*

Chaque chaîne possède une règle « policy » qui définit la cible par défaut lorsqu'un paquet ne correspond à aucune règle.

Cela permet de définir si l'on accepte ou refuse les paquets par défaut.

Elle est généralement toujours à ACCEPT.

On peut la modifier grâce à l'option '-P' :

iptables -P INPUT ACCEPT

```
[root@hades ~]# iptables -nvL INPUT
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
1323K  396M f2b-SSH    tcp  --  *      *       0.0.0.0/0
  18M   44G ACCEPT     all  --  *      *       0.0.0.0/0
```

Voici les cibles les plus utilisées:

- De la table **FILTER** :
    1. ACCEPT
    2. DROP
    3. REJECT

- De la table **NAT** :
    1. chaîne **POSTROUTING** → SNAT (IP source convertie)
    2. chaîne **PREROUTING** → DNAT (IP destination convertie)

- Dans n'importe quelle table :
    1. LOG: permet d'enregistrer le passage d'un paquet

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.

Exemple

Exemple: protection locale

Tout d'abord, listons les règles qui sont présentes sur la machine :

```
# iptables -nvL
```

-L liste les règles
-v active le mode 'verbeux'
-n pour les sorties numériques

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

```
[root@sweb ~]# iptables -nvL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
96036   29M ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
18031 1082K ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
   11   572 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
 132K   16M REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REJECT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 4 packets, 480 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Exemple: protection locale

Nous allons maintenant autoriser les connexions HTTP entrantes :

HTTP = TCP:80

Si on fait naïvement :

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

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 ACCEPT => accepte les paquets

On ajoute donc la règle en bas de la chaîne INPUT, c'est à dire après la règle :

```
REJECT     all  --  *      *  0.0.0.0/0   0.0.0.0/0   reject-with icmp-host-prohibited
```

… et comme les règles sont lues dans l'ordre, aucune des règles ajoutées après le « REJECT ALL » ne seront lues.

Il faut donc les insérer plutôt que de les ajouter :

```
#  iptables -I INPUT 2 -p tcp --dport 80 -j ACCEPT
```

Gardez à l'esprit que vos règles ne sont pas sauvegardées tant que vous n'avez pas exécuté la commande :

```
[root@sweb ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
[root@sweb ~]#
```

Sur les systèmes avec un noyau systemd, il est possible de retrouver cette fonctionnalité en installant le paquet grâce à yum:

yum -y install iptables-sevices

Exemple: protection locale

Pour information, les règles sont sauvegardées dans le fichier /etc/sysconfig/iptables

```
[root@sweb ~]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.4.21 on Wed Mar 13 11:38:18 2019
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Wed Mar 13 11:38:18 2019
```

Exemple: protection locale

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

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  209 16788 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELA
    3   156 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
```

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 deuxième règle de la chaîne INPUT.

Il peut être pratique de voir les numéros de chaque règle lorsqu'on liste les règles:

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