# Filtrage IP

Nicolas Ollinger, Université d'Orléans

**M2 SIR** Sécurité des réseaux — **S4** 2017/2018

# Contrôler les frontières

- Un **routeur** assure l'interconnexion de différents LAN au niveau **IP** (couche 3).

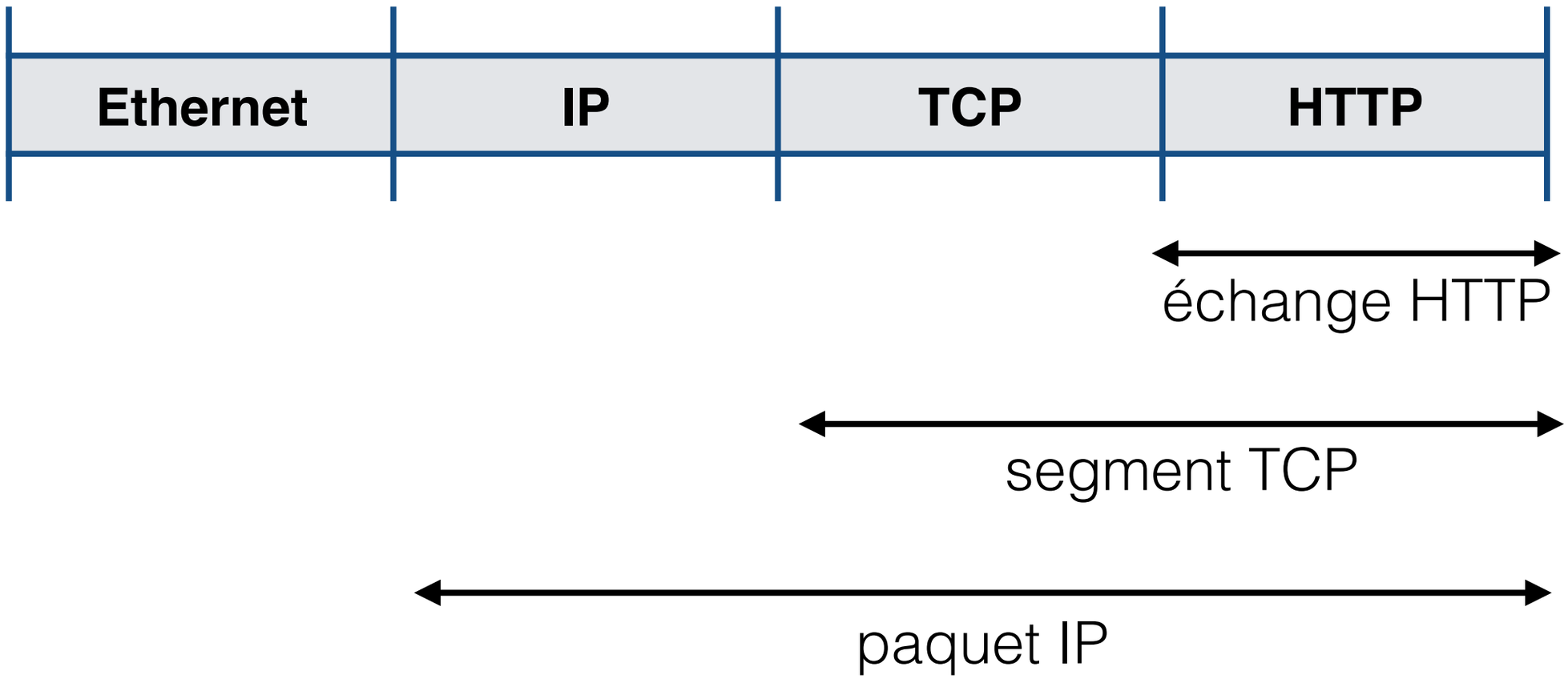
- Un **pare-feu** (*firewall*) est chargé de mettre en œuvre une politique de **contrôle** à la frontière :
  - **faire suivre** ou **effacer** les paquets, les **tracer** ;
  - selon les entêtes de couche 2, 3 ou 4 ;
  - selon la direction de traversée.

- Mise en œuvre d'une politique de sécurité réseau, en s'appuyant sur le découpage en sous-réseaux : **DMZ**, etc

# Pourquoi filtrer ?

- **Restreindre** l'accès à certains services/machines ;
- **Empêcher** l'accès à des services critiques depuis l'extérieur/certains sous-réseaux ;
- S'inscrit dans une politique de sécurité bien définie qui identifie :
  - les ressources **critiques** à protéger ;
  - les **sources** possibles d'attaques ;
  - le **niveau de sécurité** souhaité.

# Filtrage et encapsulation



# Typologie pare-feux

- Filtrage **stateless**
  effectué au niveau switch/routeur sur l'entête IP

- Pare-feu **stateful**
  prend en compte la couche transport et toute la durée de vie d'un échange (connexion TCP par exemple), son sens (client ou serveur ?)

- Pare-feu **applicatif**
  inspecte en profondeur jusqu'à la couche application (HTTP, FTP, DNS, LDAP, *etc*) le trafic non chiffré.

# Marché des pare-feux

- Check Point (VPN-1, FW-1, et successeurs) ;
- Cisco (PIX, ASA) ;
- Juniper (NetScreen) ;
- SonicWALL
- Open Source :
  **iptables** sous Linux, **packet filter** sous OpenBSD

# Contraintes en production

- Performances suffisantes :
  - bande passante (> débit du LAN) ;
  - paquets traités par seconde ;
  - nombre de sessions TCP (stateful).
- Sûreté de fonctionnement : bascule transparente sur un équipement de secours sans perte d'état.

# (1) sous Debian GNU/Linux avec **iptables**

# Netfilter

**http://netfilter.org**

- Un système de filtrage réseau au niveau du noyau Linux : permet de mettre en œuvre pare-feux, NAT et toute sorte de modification des paquets.

- Un utilitaire `iptables` en espace utilisateur pour configurer le filtrage.

- paquet `iptables`

- commandes `iptables-save`, `iptables-restore`

# Concepts

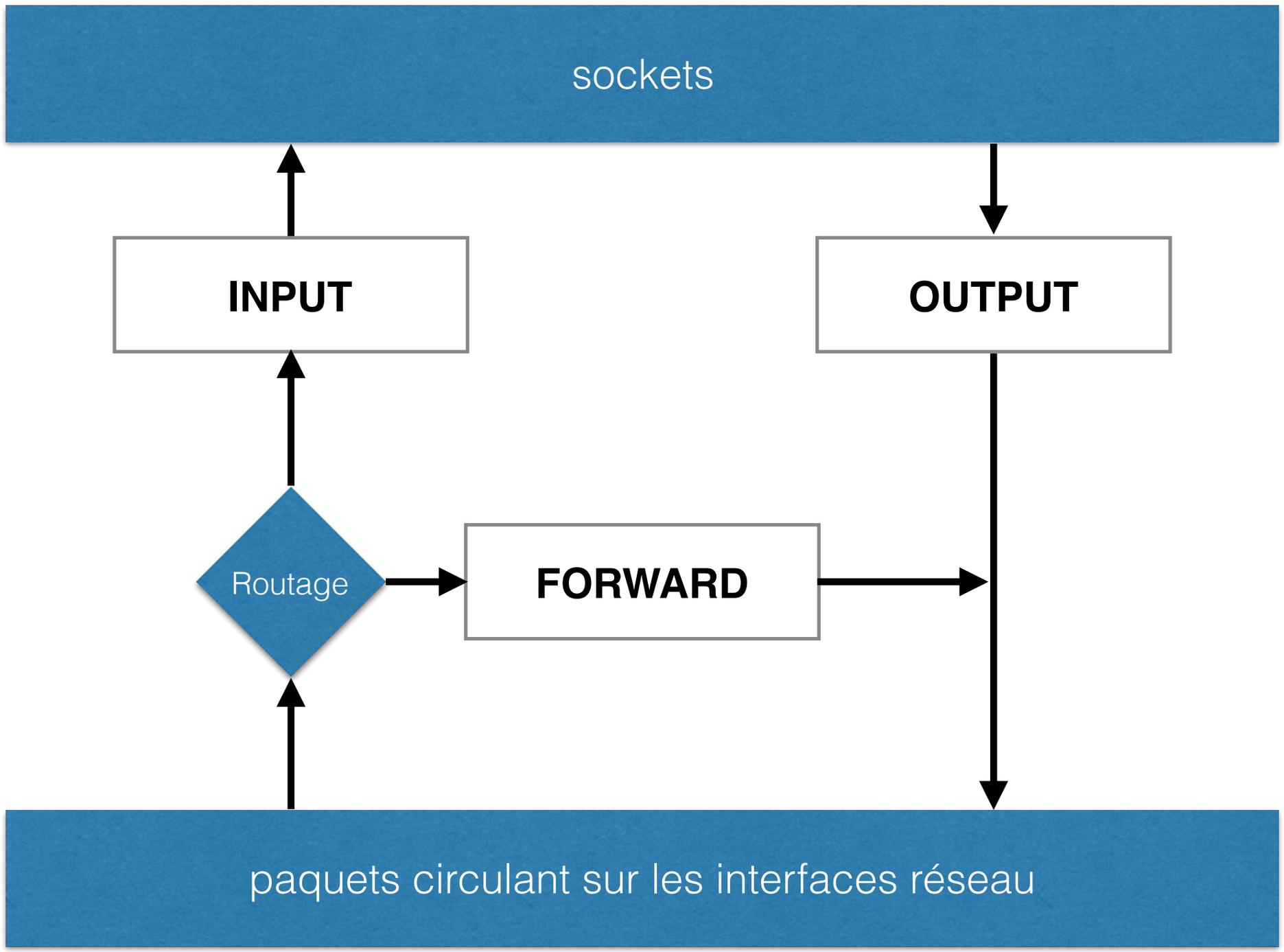
- **Table** (*table*) : contexte de filtrage (pare-feu, nat, …)
- **Chaîne** (*chain*) : chaque table est une collection de chaînes prédéfinies + de chaînes définies par l'utilisateur.
- **Règle** (*rule*) : une chaîne est une liste (ordonnée) de règles de filtrage avec conditions et cible.
- **Cible** (*target*) : action à entreprendre si la règle s'applique, peut-être le nom d'une chaîne ou une action spéciale **`ACCEPT`**, **`REJECT`**, **`DROP`**, **`QUEUE`**, **`RETURN`**, **…**

# Debug

La cible **`LOG`** permet d'écrire un résumé du paquet dans le journal système (`dmesg`).

```
iptables -t nat -I PREROUTING -j LOG \
            --log-prefix "BLOP: "
```

# iptables [-t filter]



# iptables

**`iptables -N machaine`** : créer une chaîne

**`iptables -X machaine`** : effacer une chaîne

**`iptables -P machaine cible`** : cible par défaut

**`iptables -L`** : lister les règles

**`iptables -A`** : ajouter une règle

**`iptables -I`** : insérer une règle

**`iptables -D`** : supprimer une règle

# Cibles

**`-j ACCEPT`** : fin de traitement, le paquet est accepté.

**`-j REJECT`** : fin de traitement, le paquet est refusé avec un envoi d'un message d'erreur ICMP.

**`-j DROP`** : fin de traitement, le paquet est refusé.

**`-j RETURN`** : termine le parcours de la chaîne courante.

**`-j machaîne`** : parcourt `machaîne` (jump).

**`-g machaîne`** : continue le traitement dans `machaîne` (goto).

# Conditions simples

**`-p [!] {icmp,udp,tcp,all}`** : teste le protocole

**`-s [!] address[/mask]`** : teste l'adresse source

**`-d [!] address[/mask]`** : teste l'adresse destination

**`-i [!] interface`** : teste l'interface d'entrée

**`-o [!] interface`** : teste l'interface de sortie

## `-p icmp`

**`[!] --icmp-type typename`** : teste le type de message ICMP (`echo-request`, `destination-unreachable`, ...)

# -p udp

**[!] --sport port** : teste le port UDP source

**[!] --dport port** : teste le port UDP destination

# -p tcp

**[!] --sport port** : teste le port TCP source

**[!] --dport port** : teste le port TCP destination

**[!] --tcp-flags mask comp** : teste les drapeaux parmi **SYN ACK FIN RST URG PSH ALL NONE**.

**[!] --syn** : équivalent à
**--tcp-flags SYN,ACK,FIN,RST SYN**.

# -m state

**[!] --state état** : teste l'état d'un échange (TCP, ICMP, UDP) parmi

- **INVALID** : oups !
- **ESTABLISHED** : connexion en cours ;
- **NEW** : début de connexion ;
- **RELATED** : associé à une connexion existante.

Ex: **iptables -A INPUT -m state **
**--state ESTABLISHED,RELATED -j ACCEPT**

# -m recent

Maintenir des listes d'adresses IP (par défaut l'adresse source).

**`--name maliste`** : liste concernée

**`[!] --set`** : ajoute l'adresse à la liste

**`[!] --rcheck`** : teste si l'adresse est dans la liste

**`[!] --update`** : teste la présence + ajoute un hit

**`[!] --remove`** : supprime de la liste

**`[!] -seconds ticks`** : limite la rechercher dans le temps

**`[!] -hitcount hits`** : compte le nombre d'occurrences

# (2) sous Cisco/IOS avec les **ACLs**

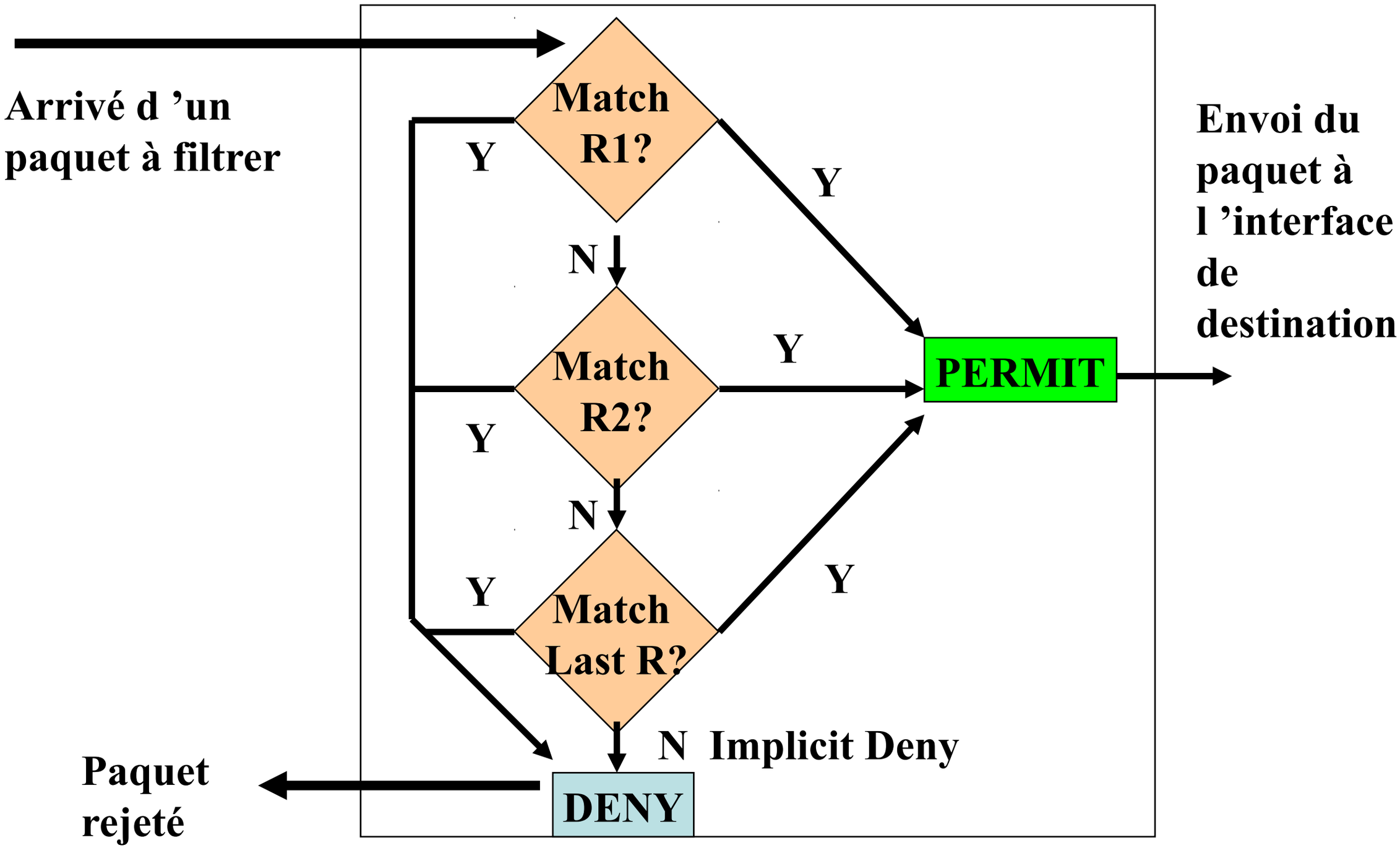
# Access Control Lists

- Une liste de contrôle d'accès (ACL) est un objet **nommé** définissable sous IOS qui décrit des règles simples de filtrage.

- **standard ACL** : couche réseau uniquement et ne vérifie qu'une seule adresse IP ;

- **extended ACL** : filtrage selon adresses IP source et destination mais aussi ports UDP/TCP.

# Processus de traitement d ’une ACL



# Application des ACLs

- Une ACL est associée à une interface par un **access-group** qui spécifie un sens (in/out) d'application.

```
interface Ethernet0/1
ip address 172.16.1.2 255.255.255.0
ip access-group 101 in
access-list 101 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 101 permit ip any 10.1.1.0 0.0.0.255
```

- Chaque interface ne dispose que d'un access-group entrant et d'un access-group sortant.

# Exemples simples de définitions d ’ACLs sur routers CISCO (1)

- **Utiliser la commande *ip access-list extended* pour créer une ACL étendue:**

*router1#  conf t*

*Enter configuration commands, one per line.  End with CNTL/Z.*

*router1(config)#ip access-list extended ACL_Ethernet0_in*

router1(config-ext-nacl)*# permit ip any any log*

- **Appliquer l’ACL en entrée ( inbound ) sur l ’interface Ethernet0**

*router1#  conf t*

*Enter configuration commands, one per line.  End with CNTL/Z.*

*router1(config)#int e0*

*router1(config-if)#ip access-group ACL_Ethernet0_in in*

# Exemples simples de définitions d ’ACLs sur routers CISCO (2)

- **Une ACL un peu plus utile**

*router1(config)#ip access list ACL_Ethernet1_in*
*router1(config -ext-nacl)#permit tcp any host 104.3.20.3 eq www*
*router1(config -ext-nacl)#permit tcp any host 104.3.20.3 eq smtp*
*router1(config -ext-nacl)#deny ip any any log*
*router1(config -ext-nacl)#exit*
*router1(config)#exit*
*router1#sh access list ACL_Ethernet1_in*
*router1#sh access-list ACL_Ethernet1_in*
*Extended IP access list ACL_Ethernet1_in*
*permit tcp any host 104.3.20.3 eq www*
*permit tcp 10.3.1.0 0.0.0.255 host 104.3.20.3 eq smtp*
*deny ip any any log*

# Exemples simples de définitions d ’ACLs sur périphériques CISCO (3) : une ACL plus complète

permit ip 126.11.1.0 0.0.0.255 10.1.6.0 0.0.0.255 (320929 matches)

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.10 eq telnet

permit tcp 126.11.1.0 0.0.0.255 host 10.1.68.17 eq 5631 (66454 matches)

permit ip 126.11.1.0 0.0.0.255 host 10.1.1.7 (8853 matches)

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.1 eq telnet (299 matches)

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.10 range 1700 1799

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.20 eq 2505 (6754 matches)

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.19 eq 5631

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.20 eq 2504 (18183 matches)

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.1 eq 389 (872 matches)

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.3 eq 389

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.20 eq 2503 (544 matches)

permit tcp 126.11.1.0 0.0.0.255 host 10.1.63.10 eq 1500

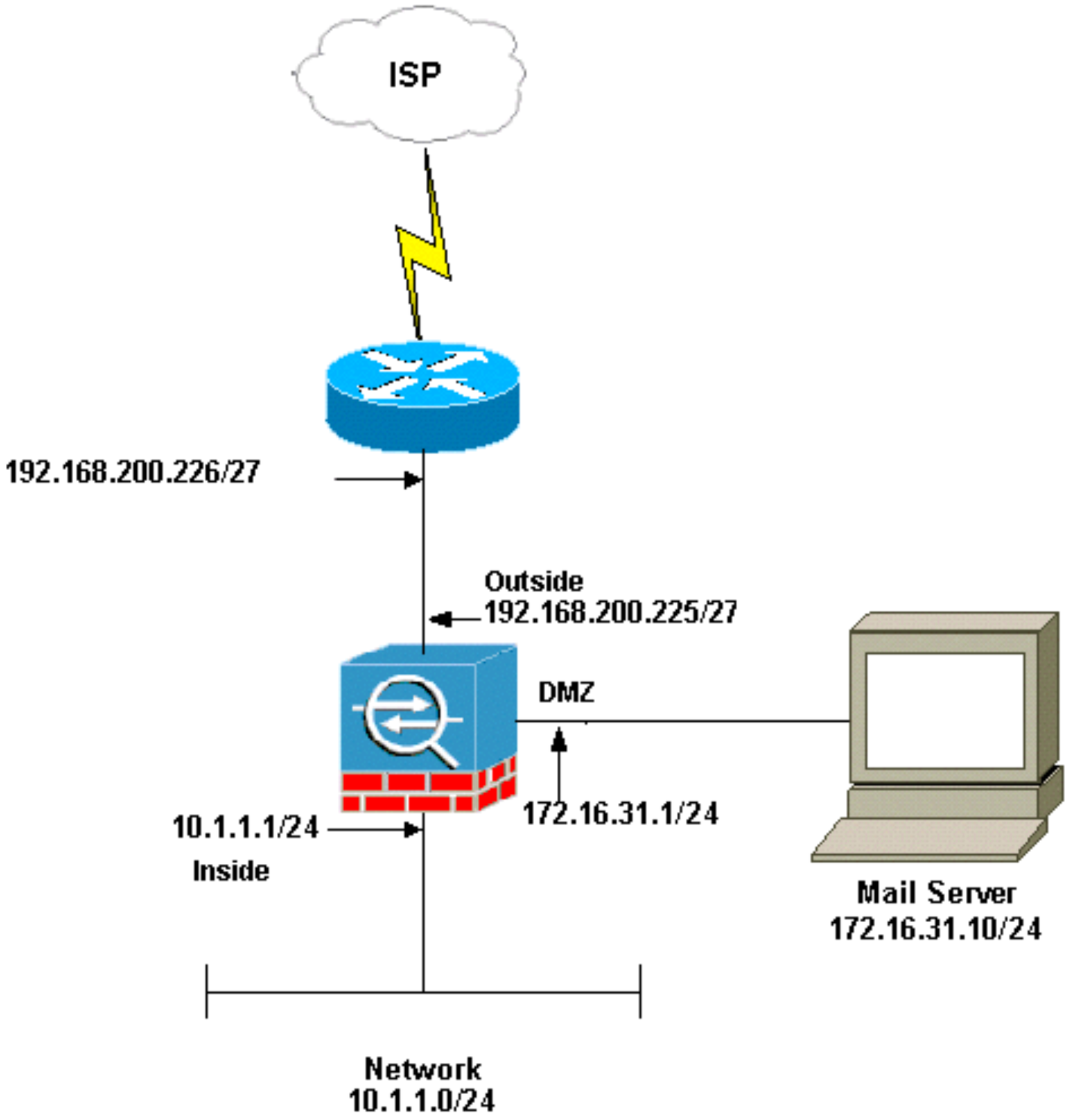
deny ip any any log (40 matches)

# ACLs 101

- Il y a une règle **deny** explicite à la fin ;
- Le parcours s'arrête dès qu'une règle s'applique ;
- Toujours écrire les règles les plus restrictives en tête de liste ;
- L'option **log** permet de journaliser les paquets ;
- Les ports se contrôlent avec **eq** et une plage de ports s'indique avec **range** ;
- Le mask permet de spécifier des plages d'IP ;
- Les ACLs sont **stateless**.

# Cisco PIX/ASA

- Appliance de type **pare-feu** (+ NAT, VPN, *etc*)
- Filtrage **stateful** TCP/UDP/ICMP
- Principe simple : un **security-level** par interface
- Par défaut (sauf ACL contraire) le trafic ne peut qu'aller vers des interfaces de niveau de sécurité inférieure.
- Sûreté de fonctionnement (*failover*)


ISP
192.168.200.226/27
Outside
192.168.200.225/27
DMZ
172.16.31.1/24
10.1.1.1/24
Inside
Mail Server
172.16.31.10/24
Network
10.1.1.0/24

```
!--- Configure the inside interface.

interface Ethernet3
 nameif inside
 security-level 100
 ip address 10.1.1.1 255.255.255.0

!--- Configure the outside interface.

interface Ethernet4
 nameif outside
 security-level 0
 ip address  192.168.200.225 255.255.255.224

!--- Configure dmz interface.

interface Ethernet5
 nameif dmz
 security-level 10
 ip address 172.16.31.1 255.255.255.0

!--- Allow outgoing SMTP connections
access-list dmz_int extended permit tcp host
172.16.31.10 eq smtp any
```

# (3) sous OpenBSD avec **pf**

# OpenBSD **pf**

- Une alternative Open Source **performante**
- Mécanisme de réplication (**CARP**) sans perte d'état (**pfsync**)
- Pare-feu **stateful** sans inspection applicative
- Configuration dans `/etc/pf.conf`
- Activation des modifications avec `pfctl`
- Debug avec `pftop`

# Principe

- Liste de règles avec action et conditions
- Pour chaque paquet on parcourt toutes les règles
- La dernière dont on vérifie les critères est appliquée… sauf si le mot-clé `quick` apparaît.

# Exemple

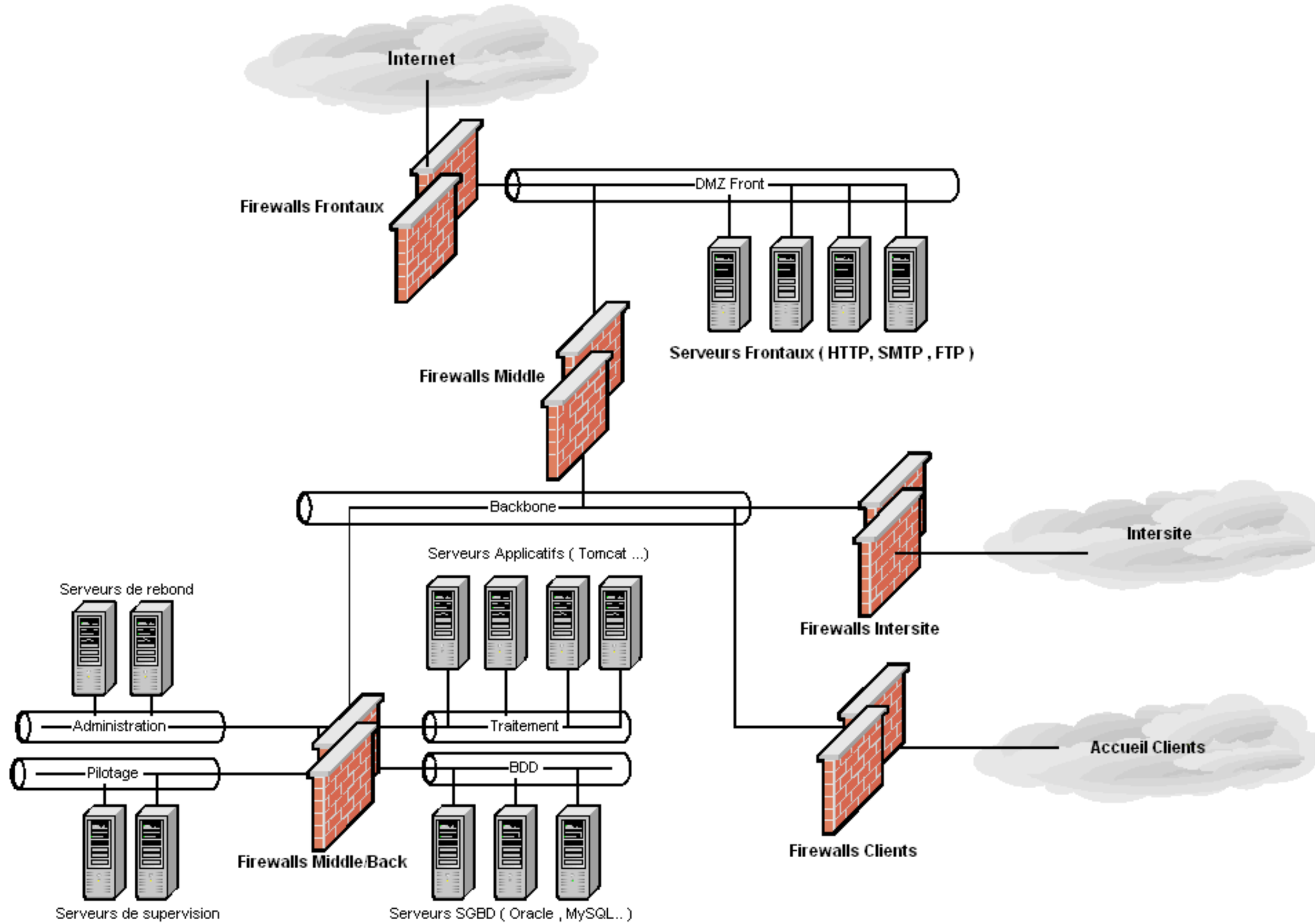
```
# règle par défaut
block log all

# autorise tout trafic sortant
pass out quick all keep state

# autorise ICMP entrant
pass in quick proto icmp

# autorise HTTP entrant
pass in quick on en0 proto tcp to 13.37.0.1 port http
```

# (4) Conclusion

# Pare-feux en entreprise

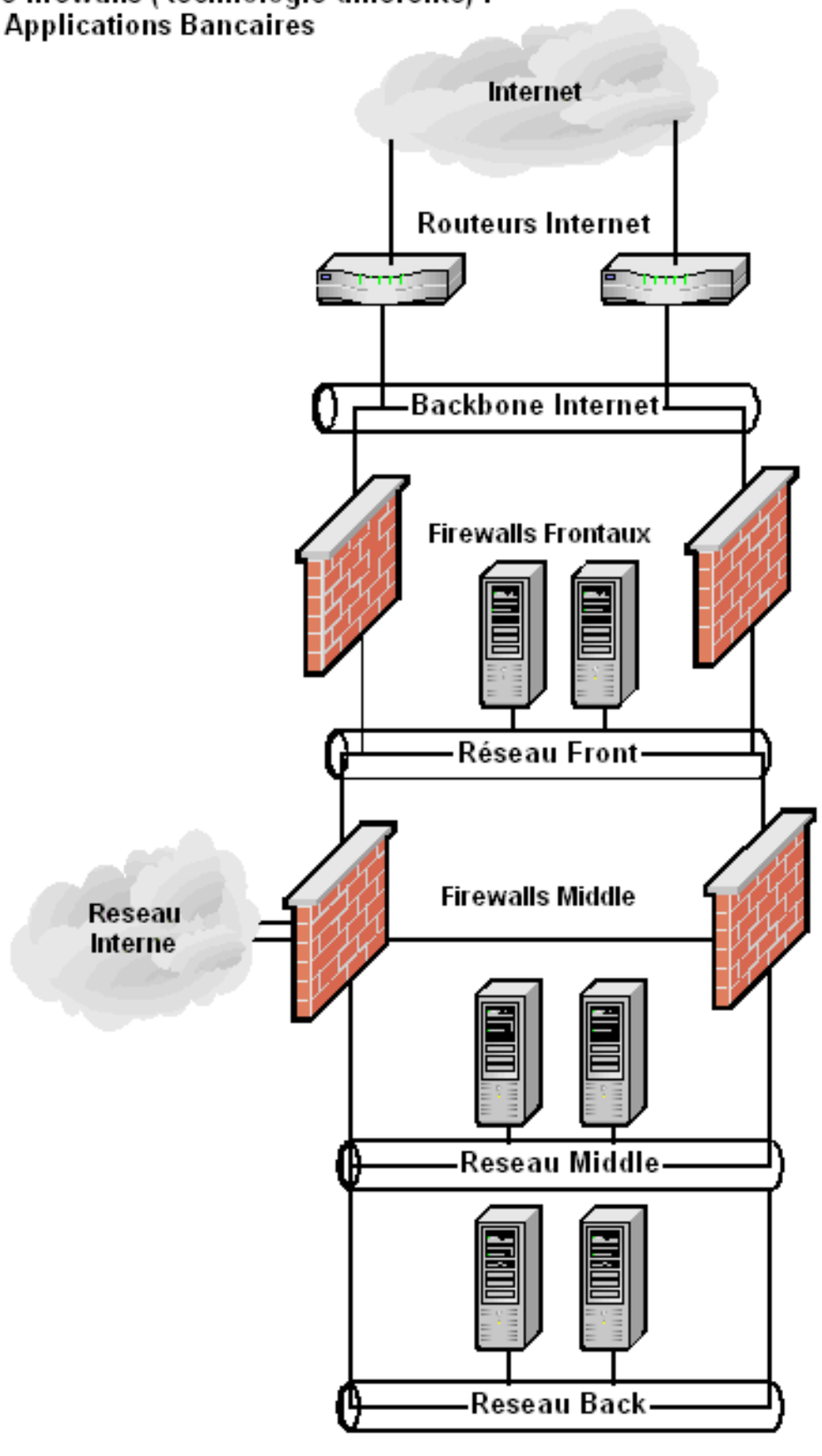
- Un outil essentiel pour **segmenter** un réseau IP
- **Filtrer** le trafic provenant de réseaux peu sûrs : internet, intersite, interconnexions.
- **Isoler** des ressources dédiées à une application ou à un client.

# Protection d'un centre de traitement informatique ( Data-Center )

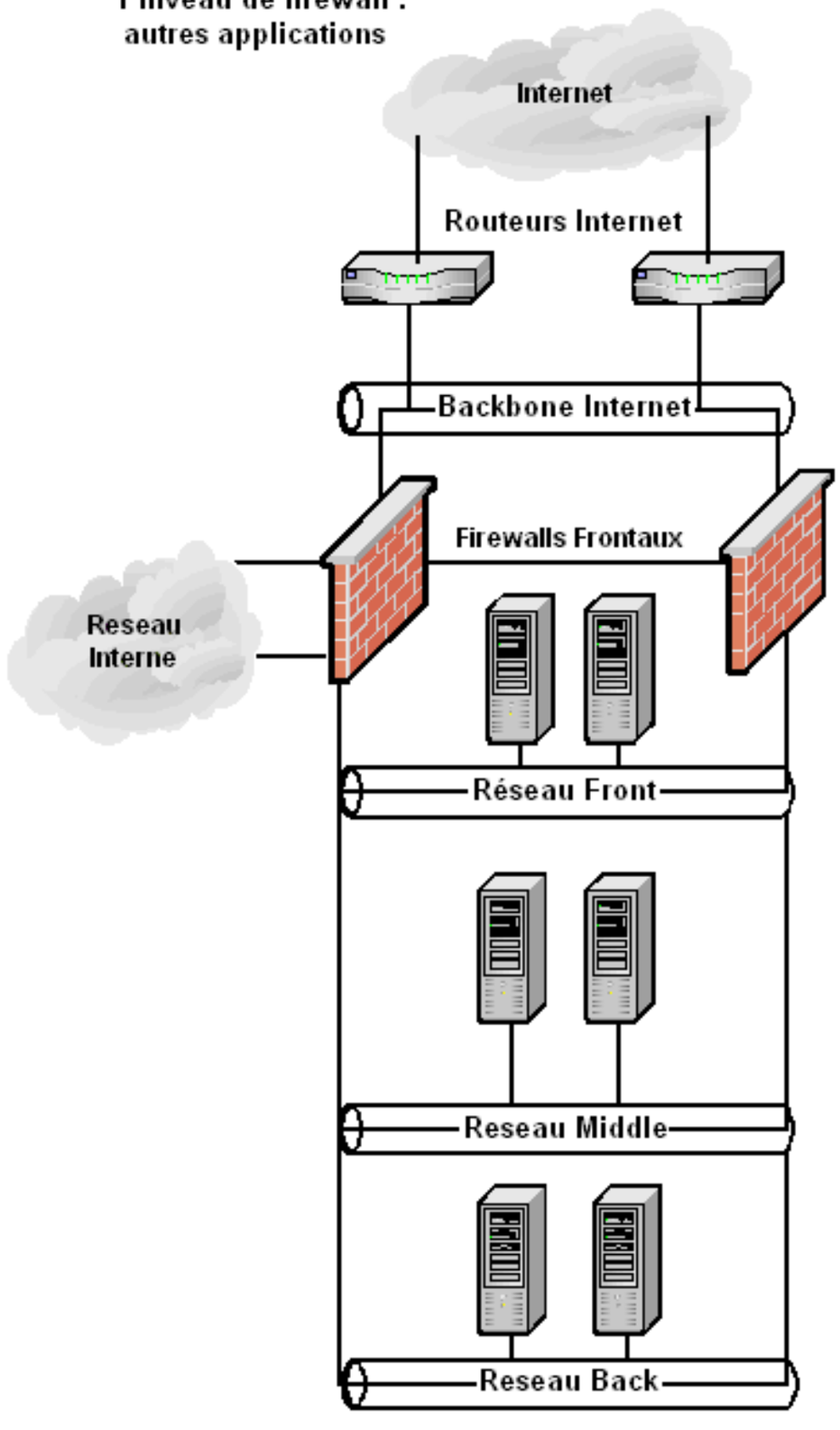


# Architectures d'une bulle réseau dedié

2 niveaux de firewalls ( technologie différente) :
Applications Bancaires




1 niveau de firewall :
autres applications



# Filtrage IP

- sur les **switches** : filtrage **stateless**, très basique ;
- sur les **routeurs** : filtrage **state{less/ful}** simple ;
- sur les **pare-feux** : filtrage **stateful** et **applicatif**.

**Remarque** La frontière entre switch de niveau 3, routeur et pare-feu a tendance à s'amincir quand on fait du routage statique.