Coupes-feux

- Coupe Feu: généralités, problèmatique
- Filtre de paquet: exemples d'utilisation, limitations
- coupefeu à états: notion d'état, exemples
- Limitation des pares-feux
- limitation des pares feux à état
- bibliographie

Coupe Feu: généralités

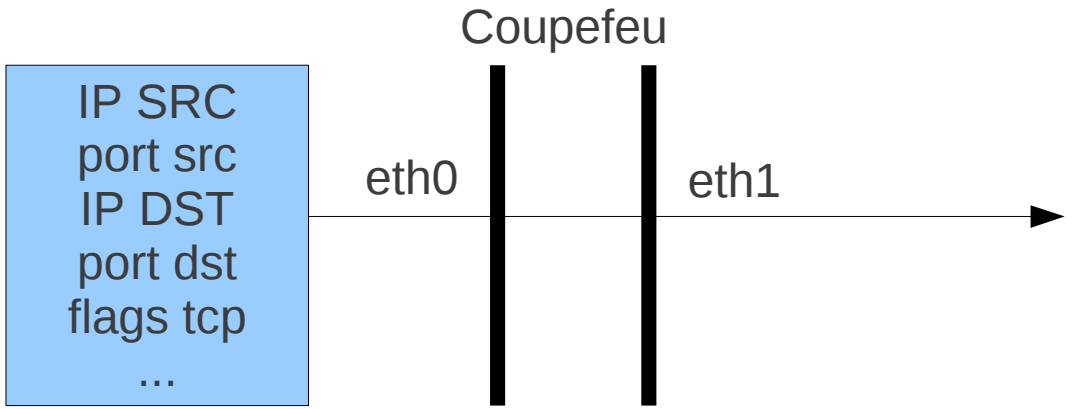
- termes équivalents : parefeu, coupefeu, garde barrière (US: firewall)
- élément d'une politique de sécurité :
  - Buts possibles:
    - protéger les postes internes des attaques
    - interdire la fuite des données de l'entreprise (cas d'un espion en interne)
    - contrôler les accès réseau des programmes présents sur un poste de travail
  - Moyens:
    - filtrer/interdire le trafic non autorisé/dangereux,
    - laisser passer le trafic légitime
    - modifier les paquets (NAT, REDIRECT, mandataire transparent, ...)

Divers types de coupes-feux

- terme recouvrant des réalités variées :
  - filtre de paquet
  - coupe feu à état
  - mandataire (proxy applicatif)
  - coupe feu personnel
- agissant à des niveaux variés:
  - couche liaison
  - couche réseau/transport
  - couche application

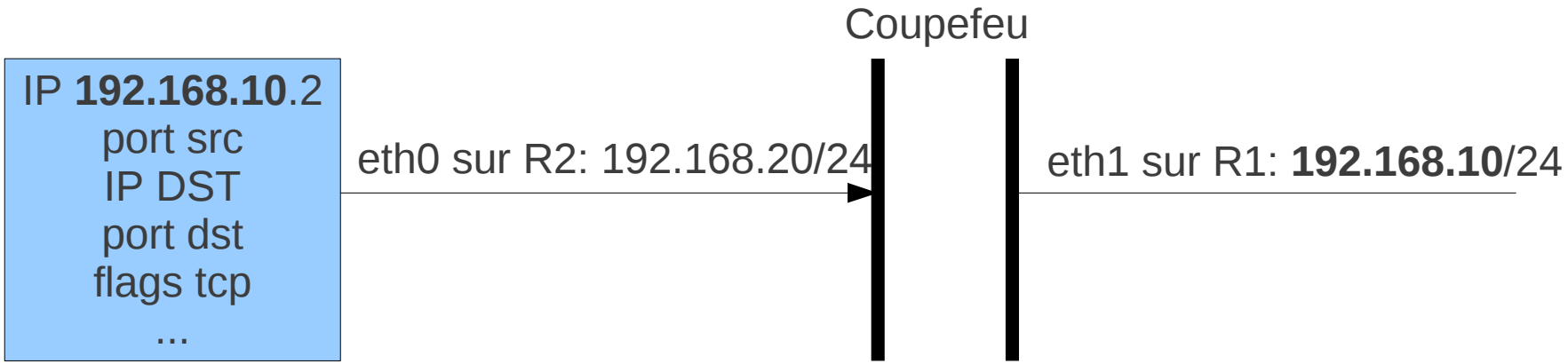
Filtre de paquet

- analyse les paquets indépendamment les uns des autres
- critères de filtrage:
  - paquet IP: IP src, IP destination, ports sources et destination
  - interface réseau sur laquelle se présente le paquet



Filtre de paquet: exemples typiques (1)

- filtrage de paquet avec une source sur un sous-réseau incorrect:
  - le coupe feu ne doit pas accepter sur eth0 des paquets ayant une IP source sur R1 (eth1)



Filtre de paquet: exemples typiques (2)

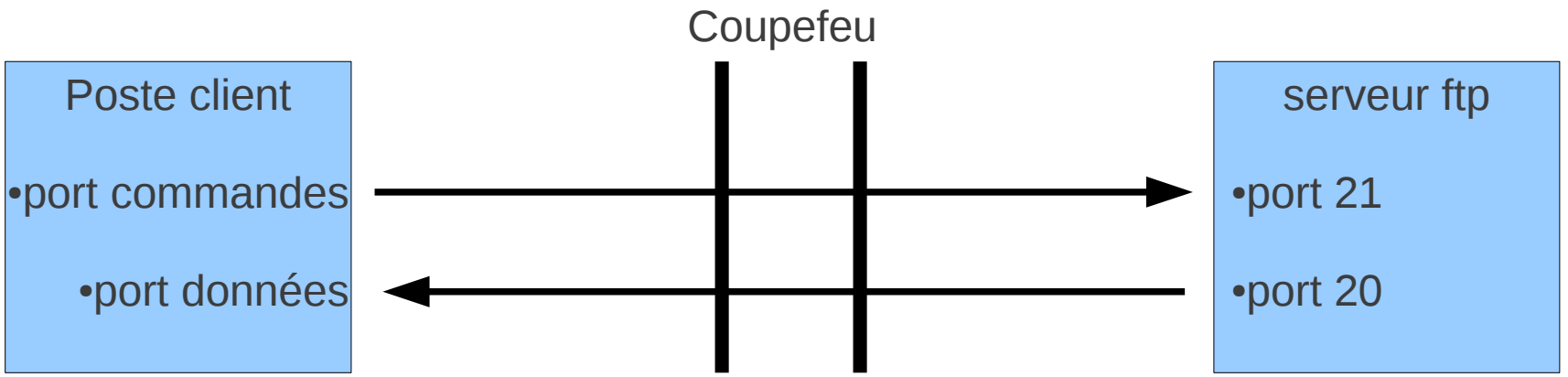
- autorisation des accès au WeB (http: tcp/80, https: tcp/443)
- en sortie: paquet vers le port 80 de toute machine externe
- paquet retour: paquet depuis le port 80 de toute machine externe
- Problème: tout paquet venant de l'extérieur et ayant le port 80 comme port source sera autorisé.
- dans la vraie vie, on utilise un mandataire WeB (proxy WeB) qui est la seule machine visible de l'extérieur

Filtre de paquet: exemples typiques (3)

- connexion tcp: l'ouverture de session est déterminée par les flags des segments
- exemple: autoriser uniquement les connexions tcp sortantes:
  - paquets tcp sortant avec flag SYN: OK
  - paquet tcp entrant avec flags Syn+Ack: OK
  - paquets tcp entrant ou sortant sans flag SYN: OK
- 2 Questions liées :
  - comment réagit une machine qui reçoit un paquet syn/ack comme premier paquet d'une connexion tcp ?
  - est-il pertinent de faire confiance aux drapeaux des segments ?

Filtre de paquet: exemples typiques (4): ftp

- le port «données» est négocié dans la session
- on peut juste le supposer >= 1024



Filtre de paquet: exemples typiques (5): ftp

- autoriser:
  - paquets syn sortant vers le port 21 du serveur ftp
  - paquets syn/ack entrant du port 21 du serveur ftp
  - paquets sans syn de/vers le port 21 du serveur ftp
  - paquets syn entrant du port 20 du serveur ftp (p20/s.ftp)vers un port >= 1024 du poste client (p >= 1024/c.)
  - paquets syn/ack sortant vers p20/s.ftp depuis p >= 1024/c.
  - paquets sans syn entrants et sortant entre le p20/s.ftp et un p >= 1024/c.
- Ces règles
  - forment une ensemble complexe
  - permettent néanmoins à une machine distante de scanner les ports tcp >= 1024 si elle prend le port

Filtre de paquets: bilan

- analyse paquet par paquet
- simple à implémenter
- syntaxe simple s'appuyant sur les propriétés du paquet (interface réseau entrante comprise)
- pas de suivi de l'historique des paquets
  - => manque de souplesse pour les autorisations
  - complexité et taille des jeux de règles: il faut plusieurs règles pour gérer des cas classiques
  - choix entre trop fermer (ne pas rendre le service) ou trop ouvrir (ne plus protéger)
  - cf exemple accès WeB sortant

coupefeu à états

- termes équivalents: coupefeu dynamique, à états, par suivi de connexion, « Statefull Packet Inspection»
- enrichit le filtrage des paquets par la mémorisation de l'état des sessions,d'échanges de données en fonction des paquets déjà vus
- analyse s'appuyant sur l'historique des sessions
- session
  - naturel avec tcp
  - la connaissance des couches réseau, transport, voire application permet d'en gérer avec udp et icmp

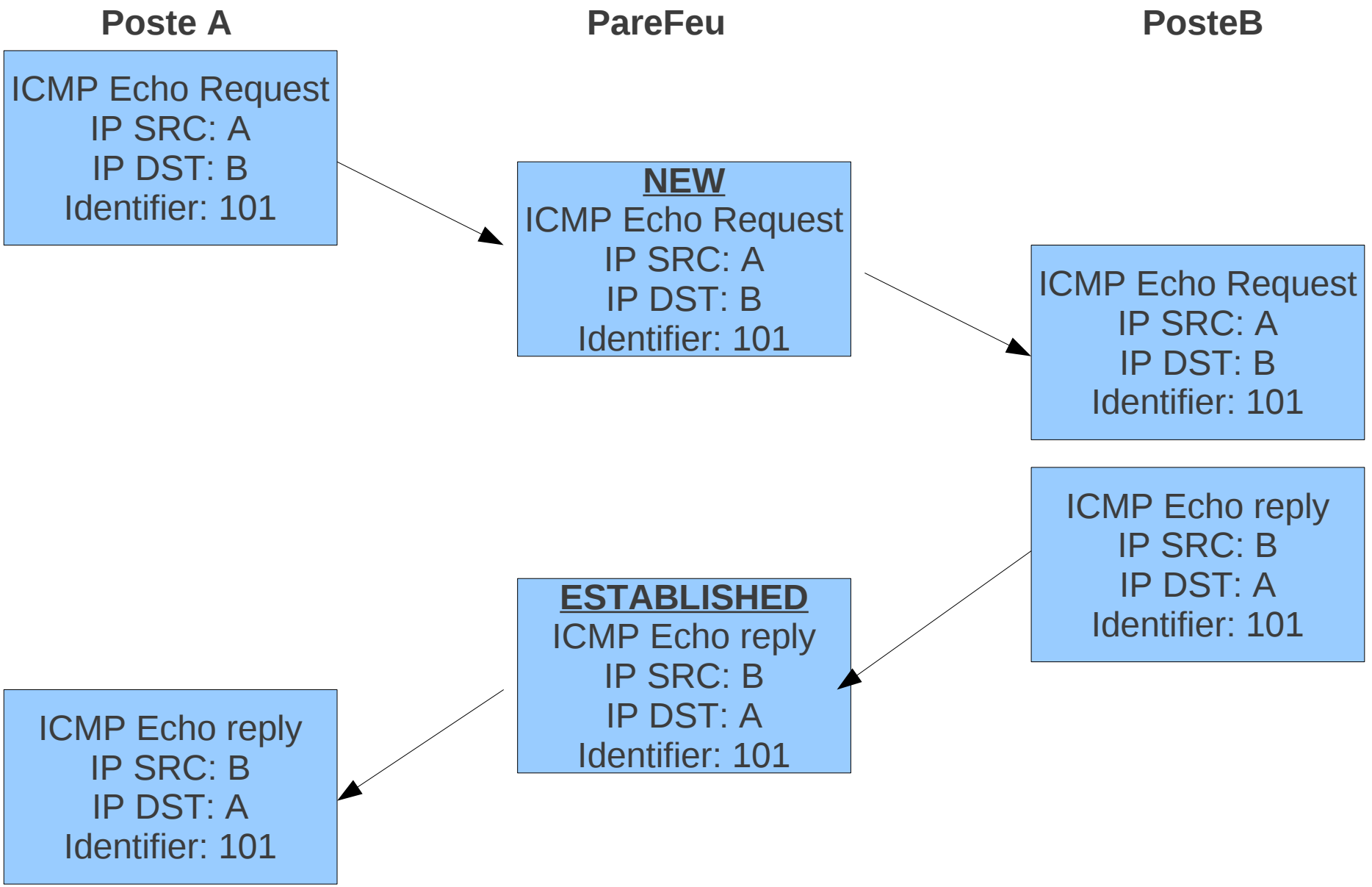
parefeu à état: état d'une session

- avec le parefeu NetFilter (Linux 2.4+), un paquet faisant partie d'une session peut être l'un des 4 états suivants :
  - New: ne correspond à aucune entrée de la table des états. Création d'une nouvelle entrée
  - Established: le paquet fait partie d'une connexion existante (entrée existante dans la table des états)
  - Related: le paquet fait partie d'une nouvelle connexion faisant partie d'une session existante.
  - Invalid: paquet dont l'état n'a pu être déterminé
- il y a des états internes plus détaillés accessibles par « cat /proc/net/ip_conntrack »

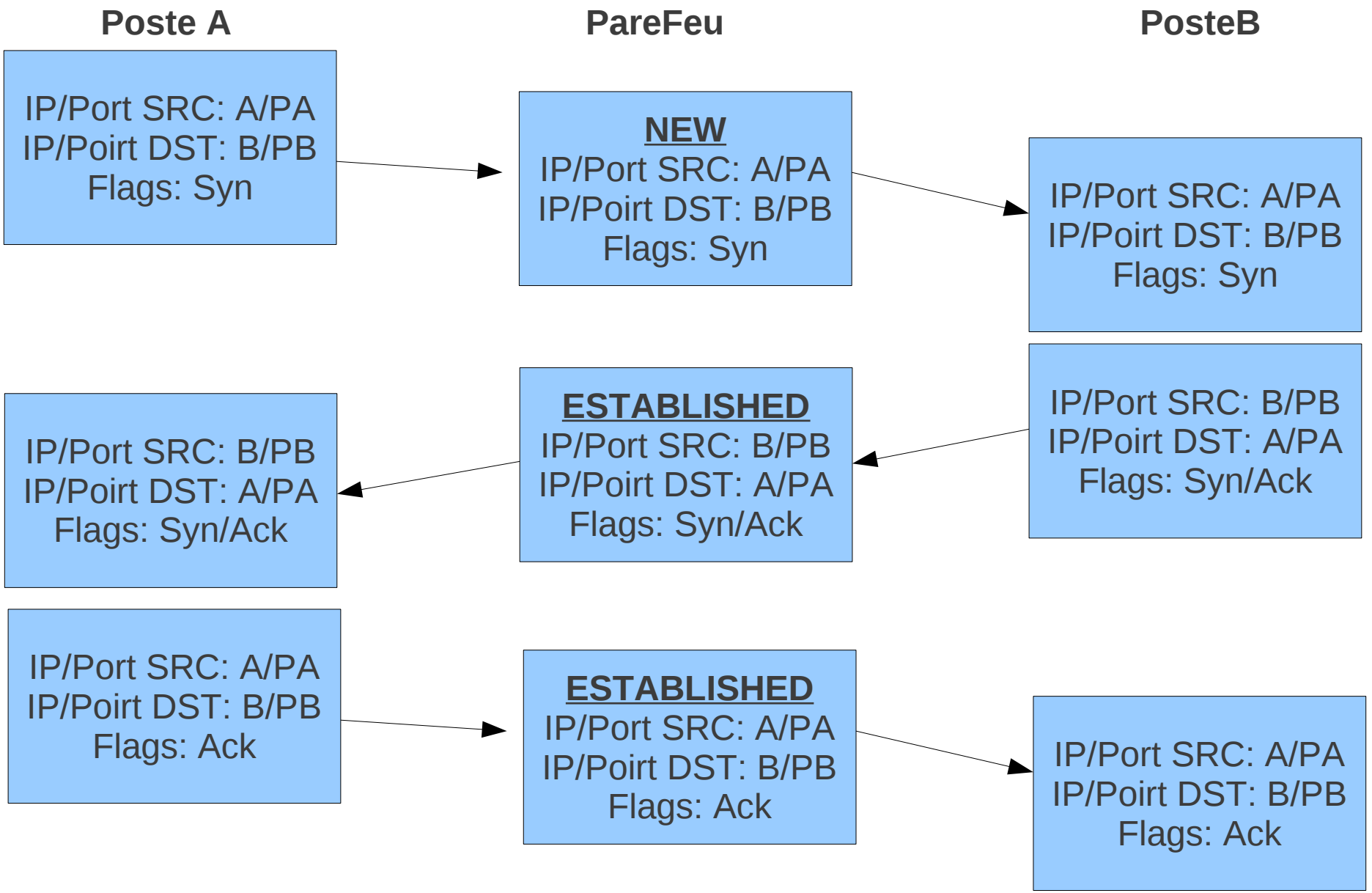
pare feu à état: états d'une session

- Attention: c'est l'étude de l'historique des paquets qui permet de déterminer l'état, pas les FLAGS TCP
  - les états fournissent « seulement » des critères supplémentaires pour le filtrage:
- l'utilisation dépend du logiciel firewall:
  - NETFILTER (linux 2.4+):
    - autoriser les paquets TCP SYN sortant
    - autoriser les paquets TCP et ICMP entrants dont l'état est RELATED ou ESTABLISED
    - interdire les paquets TCP NEW sans flag SYN
  - IPFilter (FreeBSD, Solaris 10, ...), pf (OpenBSD, FreeBSD, ...):
    - autoriser les paquets TCP SYN sortant et tous les paquets suivants de la session seront automatiquement acceptés

exemple de sessions: icmp echo



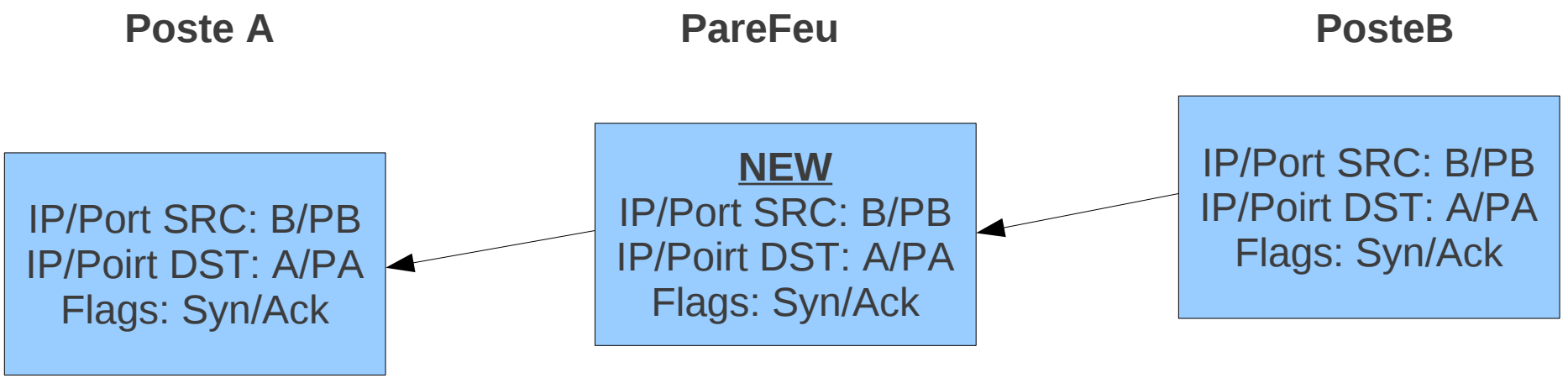
exemple de sessions: tcp



exemple de session: tcp

- Netfilter: règles associées pour autoriser un accès sortant au WeB
  - autoriser les paquets TCP sortant NEW vers le port http ou https avec un flag syn seul
  - autoriser les paquets TCP entrant/sortant ESTABLISHED
  - autoriser les paquets icmp RELATED entrant
  - refuser le reste
- ràf: animation pour illustrer une connexion sortante et une connexion entrante venant du port 80 d'une machine inconnue

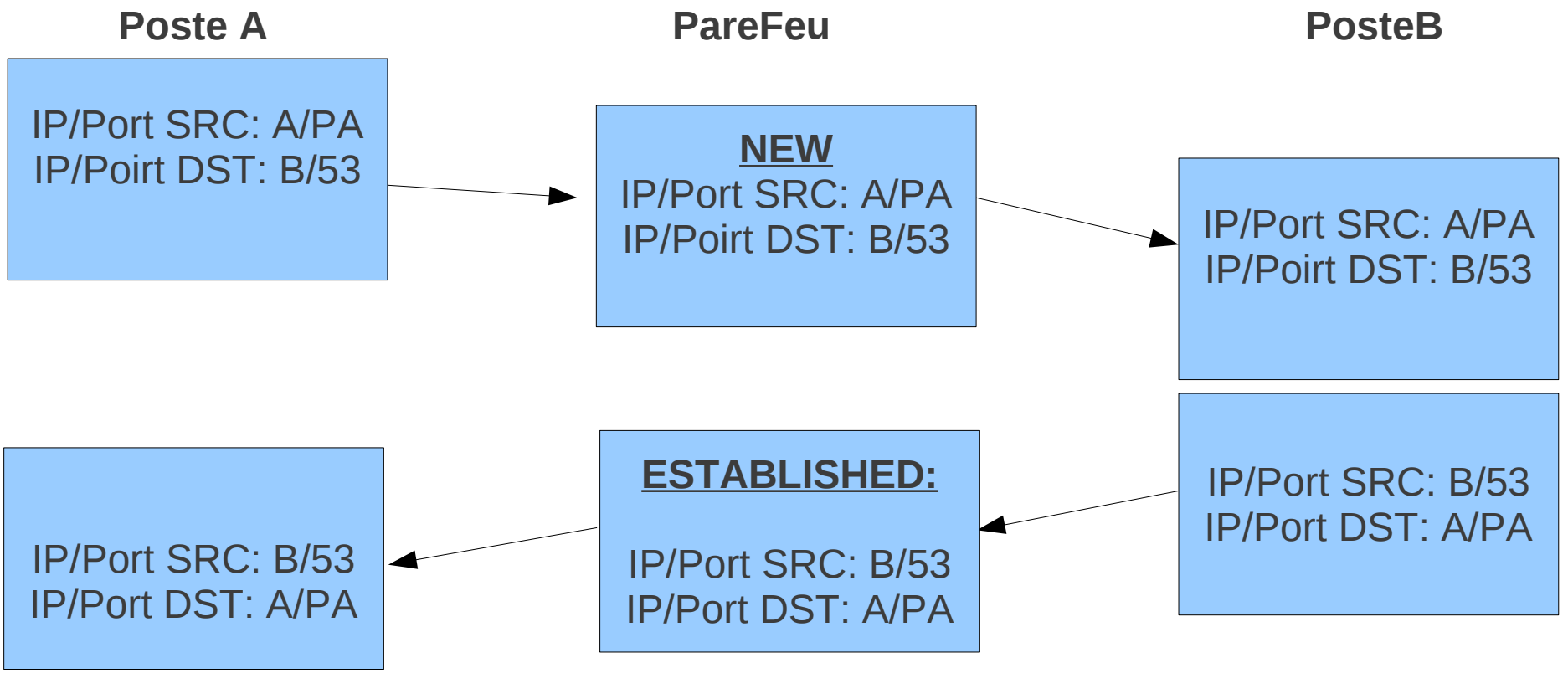
exemple de sessions: tcp particularité de netfilter



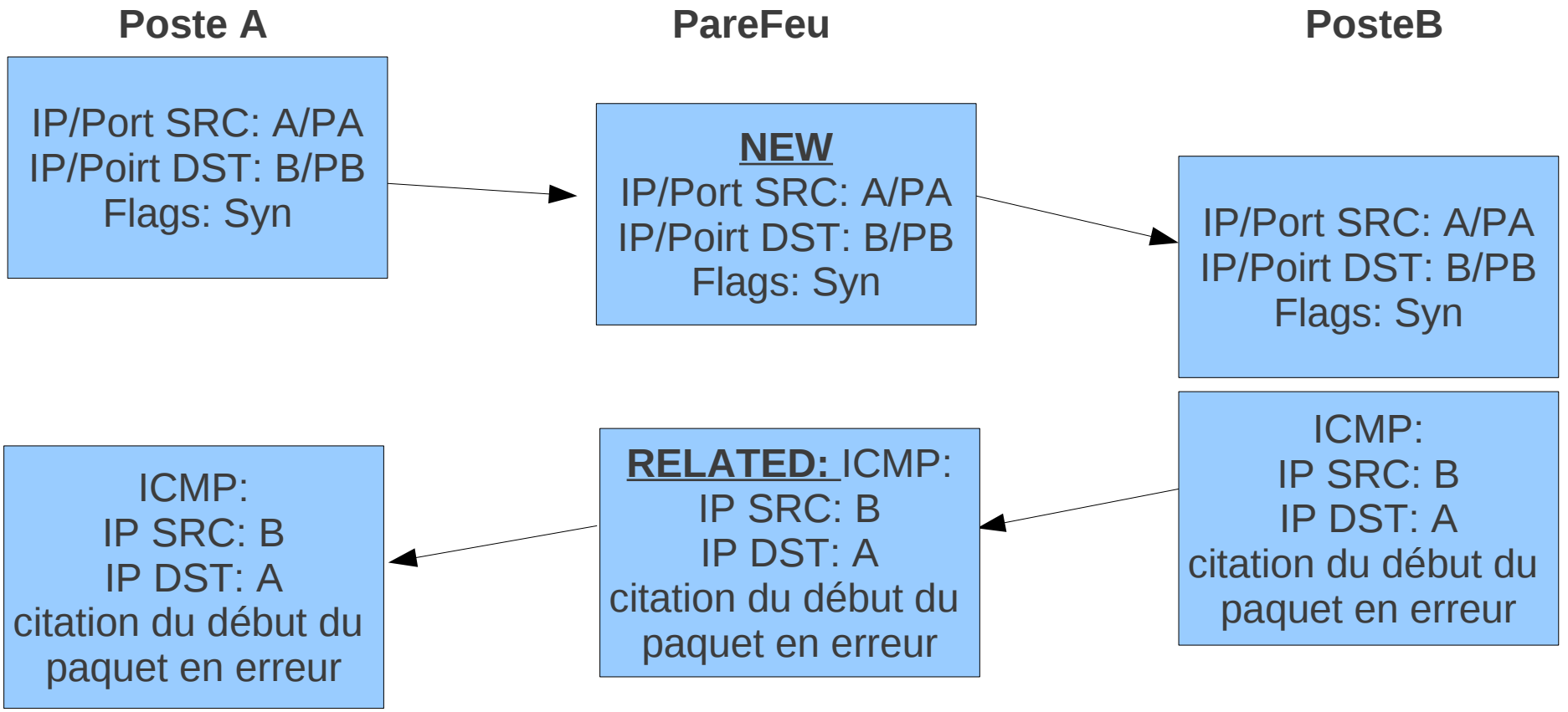
le premier paquet vu sera considéré comme NEW même s'il est incorrect comme premier paquet du point de vue tcp.
Dans l'exemple, ce premier paquet est un segment d'acquittement (alors qu'un premier paquet devrait être un SYN)
cet exemple illustre deux points :

- avec NetFilter, les états sont un critères utilisable supplémentaire qu'il faut croiser avec les autres critères pour en faire ce que l'on veut
- Application : l'une des règles usuelles utilisées avec netfilter consiste à filtrer les paquets TCP NEW sans flag SYN seul.

exemple de session: udp (dns)



exemple de session: tcp/icmp(host unreachable)



Limitation des pares-feux

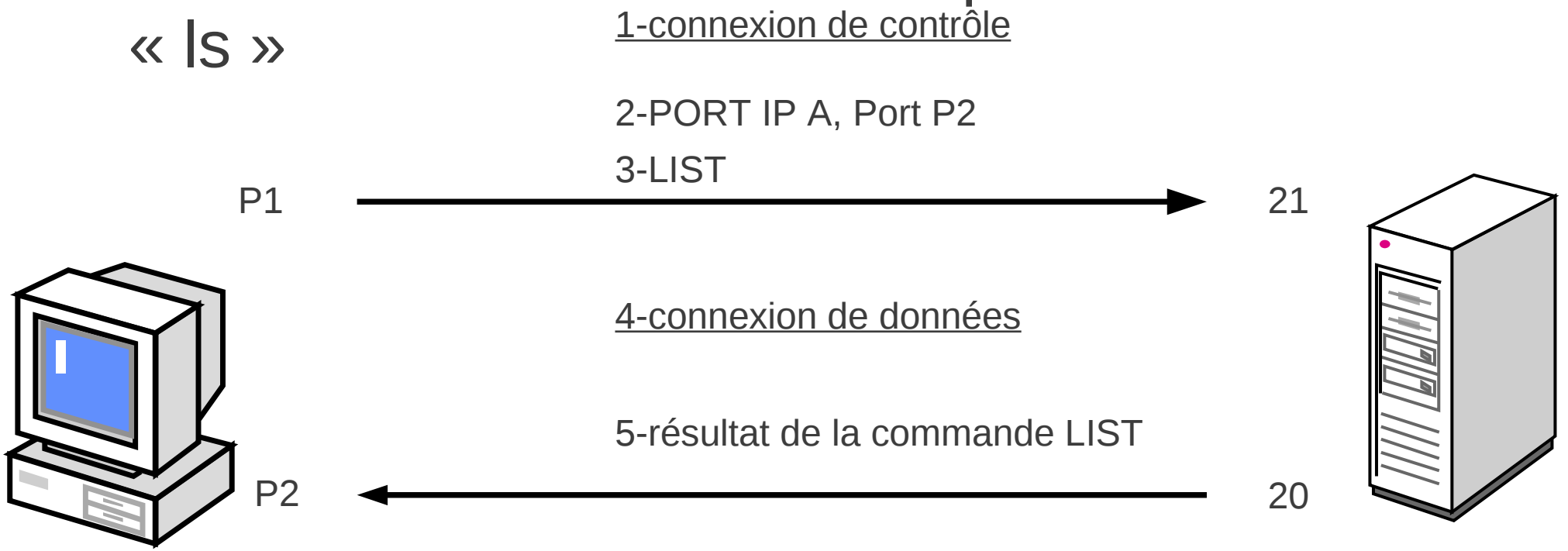
- but d'un pare feu:
  - protéger des machines internes
  - interdire les sorties/entrées d'information (plus dur)
- pare feu sans état:
  - soit on ouvre trop peu, soit on ouvre trop (ex.: connexion WeB qui ouvre tout en entrée depuis un port 80 distant)
- gestion de la fragmentation en particulier et de la normalisation de paquets en général:
  - attaque: fragmenter pour diminuer les possibilités d'identification de charge malicieuse
  - attaque: mécanisme de recouvrement de fragment

limitation des pares feux à état

- qualité du suivi de session: icmp, fenêtre tcp, ...
- analyse du niveau application souvent nécesaire (ftp, H323, ...)=> module d'analyse spécifique au protocole (ALG de la RFC 2663 ou 2993)
- insuffisant si l'information ne transite pas dans la connexion (exemple irc, sip, skype, ...)
- des applications utilisent les ports http/https
  - => vérifier que ce qui y passe est http/https
- des applications s'encapsulent dans http ou https.

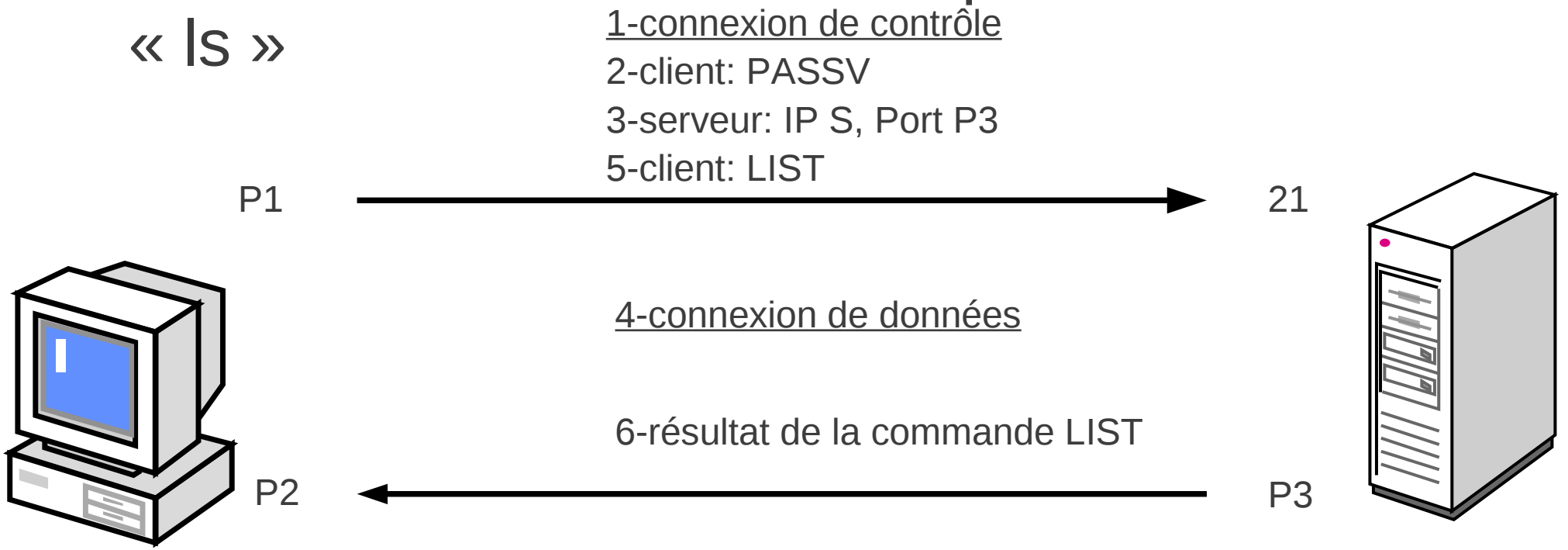
ftp : mode actif

- l'utilisateur se connecte et tape la commande « ls »

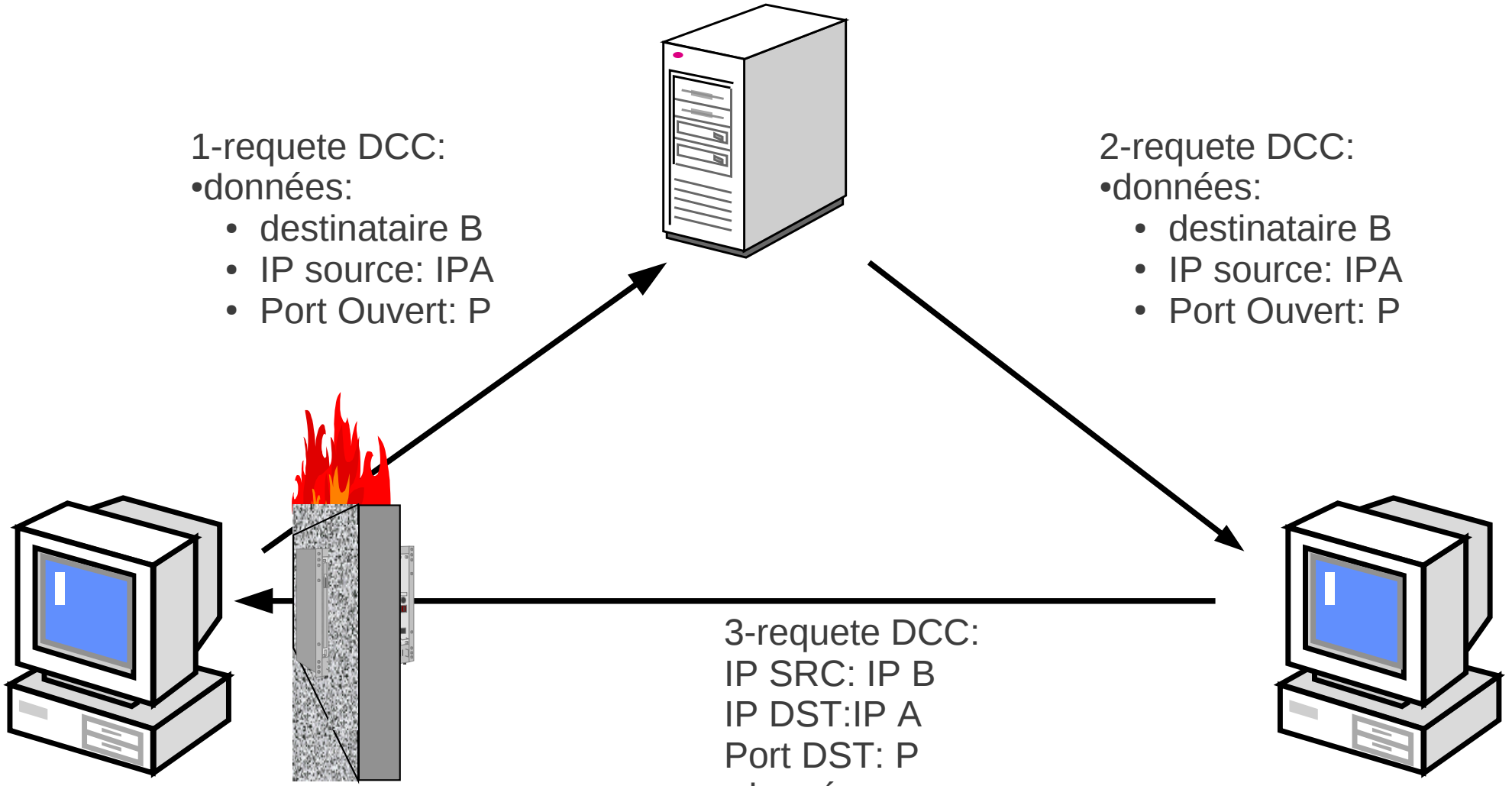


ftp : mode passif

- l'utilisateur se connecte et tape la commande « ls »



limitation des pares-feux à etat: irc



1-requete DCC:
•données:
- destinataire B
- IP source: IPA
- Port Ouvert: P

2-requete DCC:
•données:
- destinataire B
- IP source: IPA
- Port Ouvert: P

3-requete DCC:
IP SRC: IP B
IP DST:IP A
Port DST: P
•données:
- destinataire B
- IP source: IPA
- Port Ouvert: P

Netfilter : le firewall de linux 2.4 et 2.6

- Netfilter: le logiciel, IPTABLES: commande de config.
- netfilter (noyaux 2.4 et premiers noyaux 2.6):
  - filtre à état pour ipv4 et ipv6
  - filtre pour decnet, arp et (via des rustines) pour IPX
- Netfilter est un gros progrès par rapport au coupe feu des noyaux 2.2 (ipchain)
  - architecture modulaire
  - filtre à état sur ipv4/ipv6
  - traduction d'adresses,
  - altération d'entêtes de paquets (mangle)
- configuration/sauver/restaurer les tables

Netfilter

- présent dans les sources du noyau
- la version de l'outil iptables doit être compatible avec celle de netfilter
  - sinon toutes les fonctionnalités ne seront pas accessibles
- patch-o-matic: rustines apportant des fonctionnalités supplémentaires
  - submitted: rustines soumises pour la prochaine version du noyau
  - pending: en attente de soumission
  - base: rustines variées sans conflits entre eux
  - extra: le reste (conflits possibles)

Netfilter

- Thème de cette présentation
  - filtrage à état ipv4 avec netfilter
- 2 bonnes documentations (en français) :
  - « netfilter/iptables: le fonctionnement interne du parefeu selon linux »: linux mag France HS 12, octobre 2002
  - « didacticiel sur iptables » par Oskar Andreasson
    http://www.linux-france.org/prj/inetdoc/guides/iptables-tutorial/

Netfilter: tables et chaînes

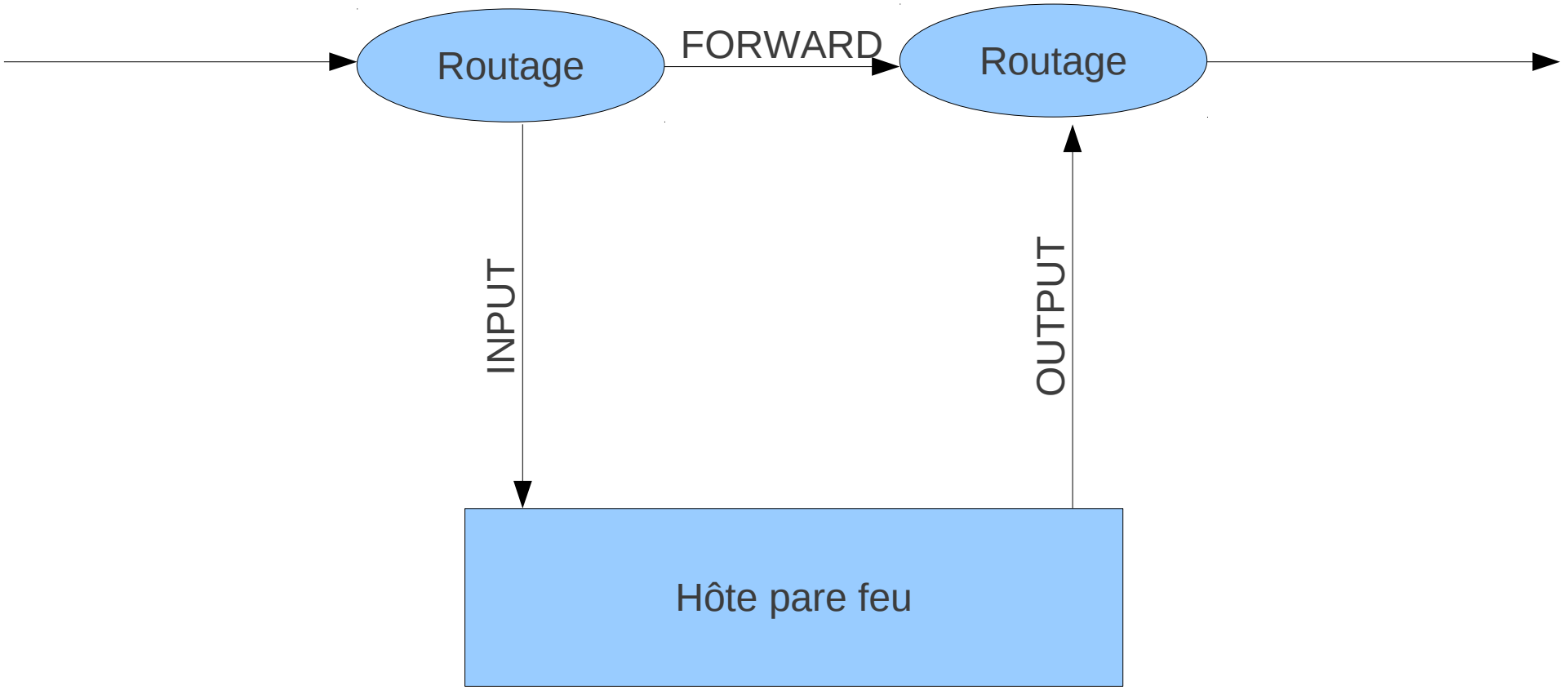
- tables: ensemble de chaînes.
- chaîne: suite linéaires de règles
- règle: constituée
  - d'un motif permettant de reconnaître des paquets selon certaines critère
  - d'un cible indiquant l'action à effectuer sur les paquets reconnus
- un paquet
  - sera traité par certaines chaînes des tables
  - dans ces chaînes, il sera traité consécutivement par toutes les règles jusqu'à en trouver une dont il valide les critères
  - la cible de cette règle sera alors appliquée

Tables NetFilter

- Filter:
  - pour les opérations de filtrage IP.
  - les paquets n'y sont jamais modifiés
  - cibles: ACCEPT, DROP, LOG, REJECT, RETURN, ...
- NAT:
  - pour les opération de traduction d'adresses
  - cibles: SNAT, SAME, DNAT, MASQUERADE, REDIRECT, RETURN, ...
- Mangle:
  - pour modifier les paquets (TTL, TOS, ...)
  - cibles: TTL, TOS, TCPMSS, RETURN, ...

traversée des tables

- cf http://www.linux-france.org/prj/inetdoc/guides/iptables-tutorial/traversingoftables.html



Les 3 chaînes de la table filter

paquet entrant

Étape	Table	Chaîne	Commentaire
1			Sur le câble (ex. Internet)
2			Arrive sur l'interface (ex. eth0)
3	mangle	PREROUTING	Cette chaîne sert normalement à modifier les paquets, i.e. changer les bits de TOS, etc.
4	nat	PREROUTING	Cette chaîne sert principalement au DNAT. Évitez de filtrer dans cette chaîne puisqu'elle est court-circuitée dans certains cas.
5			Décision de routage, i.e. le paquet est-il destiné à notre hôte local, doit-il être réexpédié et où ?
6	mangle	INPUT	Ici, il atteint la chaîne INPUT de la table mangle. Cette chaîne permet de modifier les paquets, après leur routage, mais avant qu'ils soient réellement envoyés au processus de la machine.
7	filter	INPUT	C'est l'endroit où est effectué le filtrage du trafic entrant à destination de la machine locale. Notez bien que tous les paquets entrants et destinés à votre hôte passent par cette chaîne, et ceci quelle que soit leur interface ou leur provenance d'origine.
8			Processus/application local (i.e. programme client/serveur)

tableau tiré de
http://www.linux-france.org/prj/inetdoc/guides/iptables-tutorial/traversingoftables.html

paquet sortant

Étape	Table	Chaîne	Commentaire
1			Processus/application local (i.e. programme client/serveur)
2			Décision de routage. Quelle adresse source doit être utilisée, quelle interface de sortie, et d'autres informations nécessaires qui doivent être réunies.
3	mangle	OUTPUT	C'est là où les paquets sont modifiés. Il est conseillé de ne pas filtrer dans cette chaîne, à cause de certains effets de bord. C'est aussi où le traçage de connexion généré localement prend place, nous verrons celà dans le chapitre La machine d'état.
4	nat	OUTPUT	Cette chaîne permet de faire du NAT sur des paquets sortant du pare-feu.
5			Décision de routage, comment les modifications des mangle et nat précédents peuvent avoir changé la façon dont les paquets seront routés.
6	filter	OUTPUT	C'est de là que les paquets sortent de l'hôte local.
7	mangle	POSTROUTING	La chaîne POSTROUTING de la table mangle est principalement utilisée lorsqu'on souhaite modifier des paquets avant qu'ils quittent la machine mais après les décisions de routage. Cette chaîne est rencontrée d'une part par les paquets qui ne font que transiter par le pare-feu, d'autre part par les paquets créés par le pare-feu lui-même.
8	nat	POSTROUTING	C'est ici qu'est effectué le SNAT. Il est conseillé de ne pas filtrer à cet endroit à cause des effets de bord, certains paquets peuvent se faufiler même si un comportement par défaut a été défini pour la cible DROP.
9			Sort par une certaine interface (ex. eth0)
10			Sur le câble (ex. Internet)

tableau tiré de
http://www.linux-france.org/prj/inetdoc/guides/iptables-tutorial/traversingoftables.html

paquet routé

Etape	Table	Chaîne	Commentaire
1			Sur le câble (ex. Internet)
2			Arrive sur l'interface (ex. eth0)
3	mangle	PREROUTING	Cette chaîne est typiquement utilisée pour modifier les paquets, i.e. changer les bits de TOS, etc. C'est ici aussi que le traçage de connexion généré non-localement prend place, nous verrons celà dans le chapitre La machine d'état.
4	nat	PREROUTING	Cette chaîne sert principalement à réaliser du DNAT. Le SNAT est effectué plus loin. Evitez de filtrer dans cette chaîne car elle peut être court-circuitée dans certains cas.
5			Décision de routage, c-à-d. le paquet est-il destiné à votre hôte local, doit-il être redirigé et où ?
6	mangle	FORWARD	Le paquet est alors envoyé à la chaîne FORWARD de la table mangle. C'est utile pour des besoins très spécifiques, lorsque l'on souhaite modifier des paquets après la décision de routage initiale, mais avant la décision de routage finale effectuée juste avant l'envoi du paquet.
7	filter	FORWARD	Le paquet est routé vers la chaîne FORWARD. Seuls les paquets réexpédiés arrivent ici, et c'est ici également que tout le filtrage est effectué. Notez bien que tout trafic redirigé passe par ici (et pas seulement dans un sens), donc vous devez y réfléchir en rédigeant vos règles.
8	mangle	POSTROUTING	Cette chaîne est employé pour des formes particulières de modification de paquets, que l'on veut appliquer postérieurement à toutes les décisions de routage, mais toujours sur cette machine.
9	nat	POSTROUTING	Cette chaîne est employé pour des formes particulières de modification de paquets, que l'on veut appliquer postérieurement à toutes les décisions de routage, mais toujours sur cette machine

tableau tiré de
http://www.linux-france.org/prj/inetdoc/guides/iptables-tutorial/traversingoftables.html

Exemple:

- on souhaite filtrer le trafic :
  - des paquets routés
  - sortant d'un réseau où le FW fait de la traduction d'adresse (SNAT)
- A quel niveau agir ?
- peut-on filtrer sur les adresses sources des postes internes ?

Chaînes

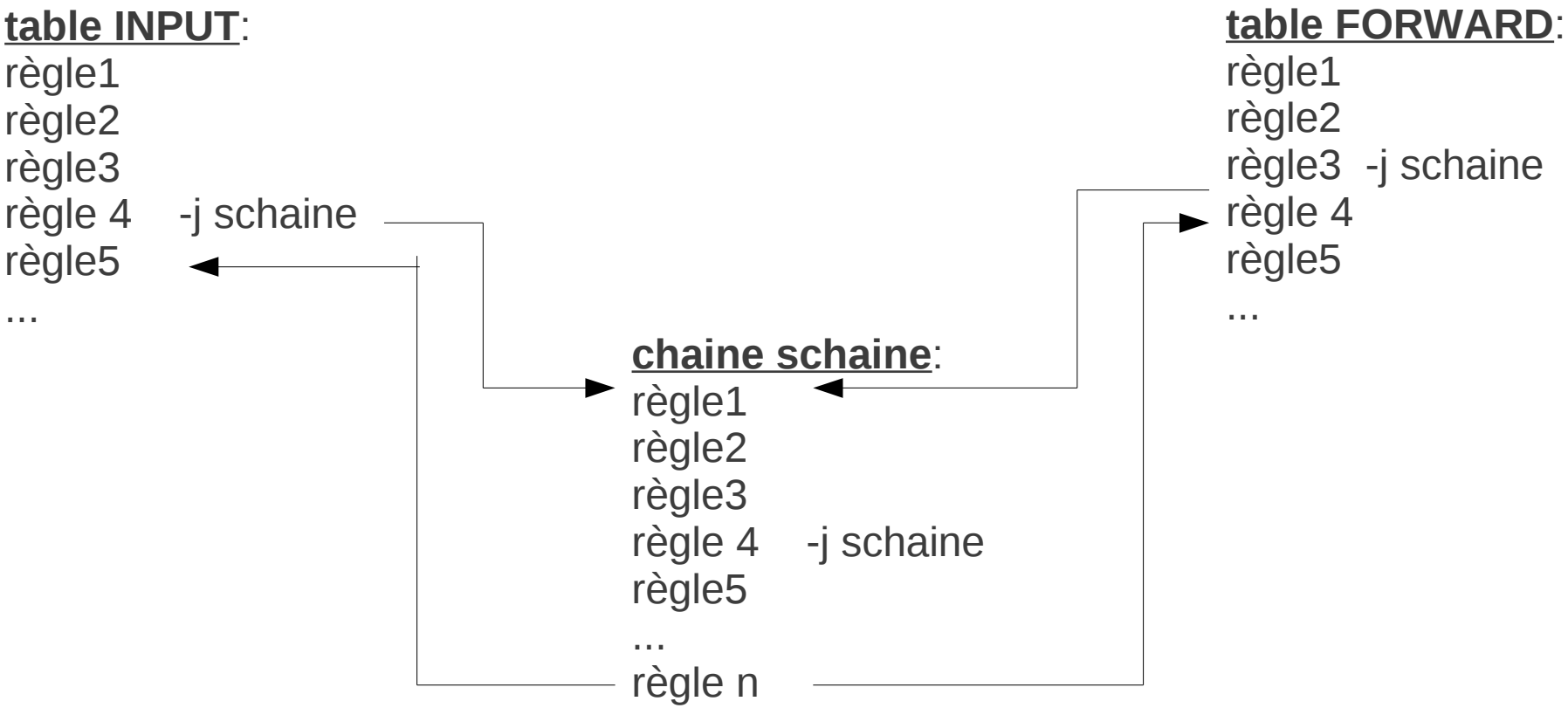
- 2 types de chaînes: par défaut (builtin) et utilisateurs
- chaînes par défaut:
  - propres à certaines tables
    - table Filter: INPUT, OUTPUT et FORWARD
    - table NAT: PREROUTING et POSTROUTING
    - table MANGLE: INPUT, OUTPUT, FORWARD, PREROUTING et POSTROUTING
  - politique par défaut:
    - politique à appliquer en fin de chaîne par défaut: ACCEPT ou DROP
    - commande -P d'iptables: « iptables -P INPUT DROP »

chaînes utilisateurs

- les appels aux chaînes utilisateurs peuvent être inclus à une ou plusieurs chaîne par défaut (on utilise le nom de la chaîne utilisateur comme cible)
- à la fin de la chaîne utilisateur, le flot d'exécution reprend à la ligne suivante de la chaîne appelante
- compteurs associés aux règles des chaînes
  - consultation avec l'option -v d'iptables

chaînes utilisateurs

- intérêt :
  - factoriser des règles
  - éviter le passage dans certaines règles à certains paquets



Netfilter: syntaxe

- iptables [-t table] commande [correspondance] [cible/saut]
  - table: table concernée. Par défaut, c'est la table filter qui est utilisée
  - commande: commande iptable (ajout de règle, suppression de règle, ...)
  - correspondance: critères du filtre de sélection de paquets.
  - cible/saut: action à effectuer sur le paquet
- cf « iptables -m correspondance --help» pour plus de détails sur une correspondance
- cf chapitres 9, 10 et 11 du didacticiel d'IPTABLES: http://www.linux-france.org/prj/inetdoc/guides/iptables-tutorial/

Netfilter: correspondance (matches)

- Les critères de base peuvent être enrichis par des modules externes qu'il convient de préciser avec l'option -m
- un protocole sans module spécifique devra se contenter des critères de base
- exemples de modules:
  - -m mac: utiliser l'adresse mac source comme critère
  - -m multiport: pour spécifier plusieurs ports d'un seul coup séparés par une virgule
  - -m state : pour utiliser le suivi de connexion

Netfilter: exemples

- placer une politique par défaut à DROP sur la table INPUT:
  - iptables -P INPUT DROP
- détruire les paquets tcp entrants avec un flag SYN seul. Deux solutions produisant les mêmes effets :
  - iptables -A INPUT -p tcp --tcp-flags SYN,ACK,RST,FIN SYN -j DROP
  - iptables -A INPUT -p tcp --syn -j DROP

Netfilter: exemples (2)

- accepter les paquets routés venant d'une source donnée:
  - venant d'un hôte: iptables -A FORWARD -s 192.168.196.246 -j ACCEPT
  - venant d'un sous-réseau: iptables -A FORWARD -s 192.168.196.0/24 -j ACCEPT
- accepter les paquets routés venant d'une adresse MAC source données:
  - iptables -A FORWARD -m mac --mac-source 00-50-56-C0-00-01 -j ACCEPT
  - noter « -m mac » qui active le module mac

Netfilter: exemples (3)

- accepter les paquets entrants appartenant à des connexions déjà établies (ESTABLISHED ou RELATED):
  - iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
  - noter le « -m state » qui active le module state
- accepter les paquets tcp routés à destination d'un port donné d'une machine données et venant d'un sous-réseau donné
  - iptables -A FORWARD -p TCP -d 192.168.196.246 --dport 22 -s 192.168.195.0/24 -j ACCEPT