Matériel de réseau

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Chapitre 4

Dispositifs de réseau

Contenu

Les fonctionnalités de réseau de Linux

1. Aperçu
2. Dispositifs et appareils commerciaux
3. Applications spécifiques
   1. Pontage LAN
   2. Routage dynamique (RIP, OSPF, ...)
   3. Firewall
   4. NAT

Motivation

- Linux comme boîte à outils pour les études en télécoms
  - Pratiquement tous les protocoles et toutes les fonctionnalités de réseau sont disponibles sous Linux
  - Plusieurs ordinateurs ou des machines virtuelles suffissent pour construire un réseau de test
    - Expérimentation pratique avec les protocoles de réseau
- Réalisation de dispositifs commerciaux
  - Beaucoup de AP WLAN, routeurs d'accès, firewalls, ... sont basés sur Linux embarqué

Aperçu des fonctionnalités de réseau

Technologies d'accès
- Ethernet
- Wireless LAN
- ATM
- Interfaces série RS-232
- USB
- Bluetooth
- Infrarouge

Protocoles couche liaison
- Ethernet, Wireless LAN, Token
- PPP / SLIP
- HDLC
- X.25 sur LAPB
- Frame Relay

Fonctionnalités couche réseau

Protocoles
- IPv4
- IPv6
- IPSec

Méthodes de routage
- Multicast PIM-SM
- Routage multipath
- Tunneling (IP-in-IP, IPv4-IPv6, …)

Protocoles de routage
- RIP, OSPF, BGP avec Zebra, Quagga, XORP ou gated

Fonctions LAN
- Pontage Ethernet
- Spanning Tree Protocol
- VLAN
- Bonding
- Point d'accès WLAN

Aperçu des fonctionnalités de réseau

Sécurité

VPN
- IPSec
- PPTP
- L2TP
- SSL

Firewall
- Filtrage de paquets sans et avec mémoire
- NAT
- Redirection de connexions (port forwarding)

Qualité de service
- Classification de paquets
- Différentes types de files d'attente
- Architecture DiffServ
- IntServ avec RSVP

Applications
- **Serveur de fichiers** avec NFS et Samba
- Proxy **VoIP** avec Asterisk et SER
- **Détection d'intrusions** avec Snort
- **Mesure de trafic** avec MRTG et RRD
- **Streaming multimédia** avec VLC ou autres

Exemples de dispositifs commerciaux

Equipement	Photo	Description
Routeur WLAN Linksys WRT-54GL		• WLAN 802.11b/g • Routage, NAT, Firewall, Port forwarding • 4 MB flash, 16 MB RAM • Switch Ethernet intégré • Prix : $70
Routeur WLAN Linksys WRV54G		• Routeur WLAN 802.11b/g • Authentification 802.1x / RADIUS • Tunnels VPN IPSec • Processeur de réseau IXP4xx • 64 MB RAM • $150
Serveur de fichiers (NAS) Linksys NSLU2		• Serveur de fichiers Windows / Linux avec Samba • Accessible via LAN ou Internet • Fonction de backup • Connexion d'un disque dur externe USB • Processeur de réseau IXP4xx • Prix : $75

Exemples de dispositifs commerciaux

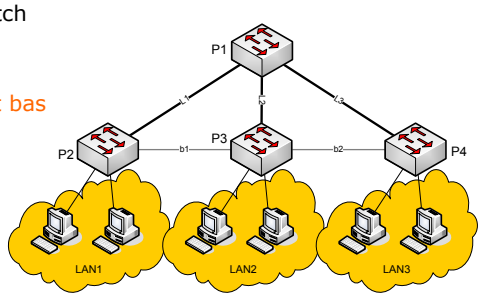
Equipement	Photo	Description
Routeur/Firewall CyberGuard SG300		• Firewall jusqu'à 30 Mb/s • NAT • Détection d'intrusions • VPN IPSec ou PPTP/L2TP • Qualité de Service • $250
Firewall / Client VPN IPSec Innominate mGuard		• Dimensions 5cm x 10cm • Firewall stateful • Client VPN IPSec à 40 Mb/s • Protection contre des virus • Processeur IXP425
Gateway VoIP i3 micro VRG 321		• Permet de connecter des téléphones analogiques. • Gateway VoIP avec SIP / H.323
Set-Top Box Vidéo Comtrend CT-702		• Client « vidéo à la demande » • Sortie TV • Routeur WLAN intégré • Télécommande infrarouge • Processeur IXP425

Exemples de dispositifs commerciaux

Equipement	Photo	Description
Téléphone VoWiFi Siemens OptiPoint WL2 Professional		• Téléphone VoIP avec WiFi • $500
Radio Internet / WiFi AE WiFi radio		• Se connecte sur un routeur d'accès WiFi • Reçoit 2500 stations de radio Internet
Téléphone portable GSM Motorola A732		• GSM • Lecteur MP3

Pontage LAN

- Scénario d'application
  - Interconnexion de réseaux Ethernet
  - Normalement fait avec des switch Ethernet
- Ponts Linux
  - Plus cher qu'un switch Ethernet bas de gamme
  - Moins performant qu'un switch
  - Fonctionnalités sophistiquées
    - VLAN
    - Filtrage de paquets
    - Mesure de trafic, Qualité de Service
  - Mise à jour possible
  - Meilleur marché qu'un switch avec les mêmes fonctionnalités



Fonctions de base à réaliser

- Fonctions d'un bridge/switch transparent (norme IEEE 802.1D)
  1. Apprentissage dynamique de la topologie du LAN ('mémoire CAM')
  2. Acheminement des trames sur la base de la table de filtrage
  3. Évitement de boucles à l'aide du protocole STP

Pontage dans Linux

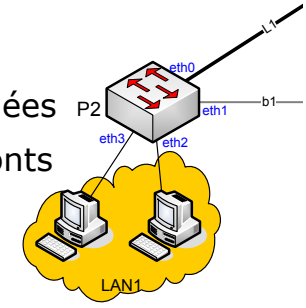
- Nécessite deux éléments:
  1. Fonctionnalité au niveau du noyau: module « bridge »
  2. Utilitaire bridge-utils
     - Commande brctl
     - Permet la configuration de la fonctionnalité de pontage

```
> brctl
commands:
      addbr           <bridge>                add bridge
      delbr           <bridge>                delete bridge
      addif           <bridge> <device>       add interface to bridge
      delif           <bridge> <device>       delete interface from bridge
      setbridgeprio   <bridge> <prio>         set bridge priority
      setpathcost     <bridge> <port> <cost>  set path cost
      show                                    show a list of bridges
      showmacs         <bridge>               show a list of mac addrs
      showstp         <bridge>                show bridge stp info
      stp             <bridge> <state>        turn stp on/off
```


Configuration d'un pont

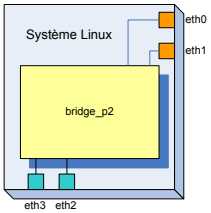
- Création d'un pont logiciel
  - Initialise les structures de données
  - Possibilité de créer plusieurs ponts sur le même dispositif

```
P2> brctl addbr bridge_p2
P2> ifconfig bridge_p2 up
```



- Ajout des interfaces au pont

```
P2> brctl addif bridge_p2 eth0
P2> brctl addif bridge_p2 eth1
P2> brctl addif bridge_p2 eth2
P2> brctl addif bridge_p2 eth3
```



Diagnostique

Affichage de la configuration

```
P2> brctl show bridge_p2
bridge name     bridge id               STP enabled     interfaces
bridge_p28000.0002b3010101              no              eth0
                                                        eth1
                                                        eth2
                                                        eth3
```

Contenu de la table de filtrage

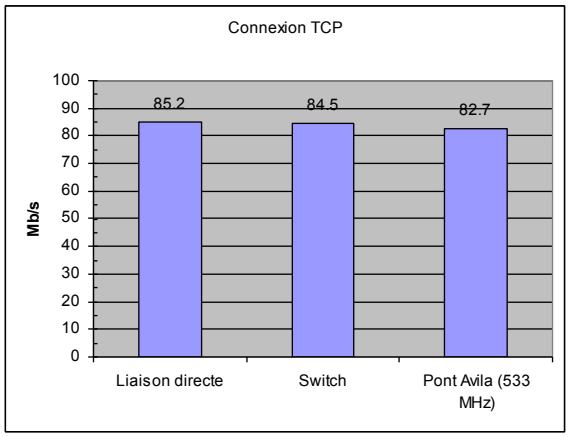
```
P2> brctl showmacs bridge_p2
 port no   mac addr        is local?       ageing timer
   1       00:00:4c:9f:0b:ae     no            17.84
   1       00:00:4c:9f:0b:d2     yes            0.00
   2       00:00:4c:9f:0b:d3     yes            0.00
   1       00:02:55:1a:35:09      no           53.84
   1       00:02:55:1a:82:87      no           11.53
```


Performances du pont Linux

- Comparaison
  - Liaison directe
  - Switch Ethernet (matériel)
  - Pont Linux (logiciel)
- Résultats pour une seule connexion TCP

MAIS :

- Un (bon) switch pourra fournir le débit maximum sur tous les ports
- La vitesse du pont Linux est limitée par la puissance de calcul



Protocole STP

- Requis lorsque plusieurs ponts sont interconnectés

Configuration

- Activer STP sur un pont

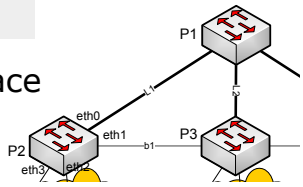
  p2> ***brctl stp bridge_p2 on***

- Modifier la priorité d'un pont de devenir racine

  p2> ***brctl setbridgeprio bridge_p2 4000***

- Modifier le coût local d'une interface

  p2> ***brctl setpathcost p2 eth1 200***



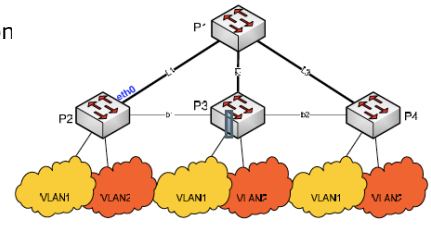
Réseaux locaux virtuels: VLAN

Objectif

- Création de réseaux locaux séparés sur une infrastructure physique partagé
  - Augmentation de la sécurité
  - Réduction du trafic de diffusion
  - Facilité d'administration

Types de VLAN

- VLAN par port
- VLAN par adresse MAC
- VLAN par IP, ...



VLAN à travers plusieurs switches

- Nécessite la technique du « VLAN trunking »
  - Encapsulation IEEE 802.1Q

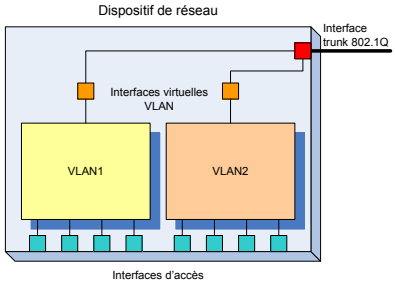
VLAN dans Linux

- Nécessite plusieurs éléments:
  - Fonctionnalité de pontage (noyau, utilitaire de configuration)
  - Module du noyau: « 8021q »
  - Utilitaire **vconfig**
    - Permet la configuration de l'encapsulation 802.1Q
- Problème possible
  - Taille maximum des cartes réseau: Les anciennes cartes réseau ont des problèmes avec la taille maximum de 1518 octets

Idée de réalisation de VLAN par port

VLAN
- Configurer un pont logiciel par VLAN
- Ajout de interfaces d'accès au pont du VLAN correspondant
- ➢ Les ponts n'ont aucun lien et sont donc séparés

Trunking
- Création de plusieurs interfaces virtuelles par interface de trunk
- Chaque interface virtuelle est assignée à un pont VLAN
- Les interfaces virtuelles ajoutent et enlèvent l'encapsulation 802.1Q



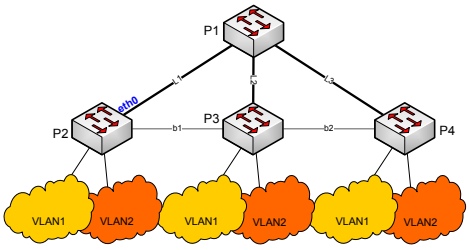
Exemple de configuration

- Création de deux ponts

```
brctl addbr vlan1
brctl addbr vlan2
```

- Ajout des interfaces

```
brctl addif vlan1 ethv
brctl addif vlan1 ethx
brctl addif vlan2 ethy
brctl addif vlan2 ethz
```

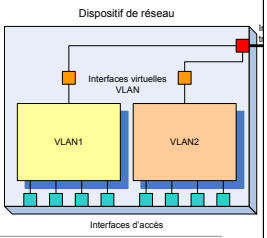


- Création des interfaces virtuelles VLAN

```
vconfig add eth0 1
vconfig add eth0 2
```

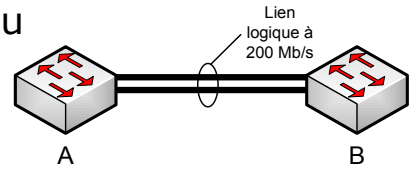
- Ajout aux ponts VLAN

```
brctl addif vlan1 eth0.1
brctl addif vlan2 eth0.2
```



Fonctions LAN avancées
Agrégation de liens

- Appelée « bonding » ou « port trunking »



Idée

  - Combiner plusieurs interfaces Ethernet physiques en une seule interface logique
    - Multiplication du débit maximum entre deux équipements
    - Meilleure protection contre des pannes

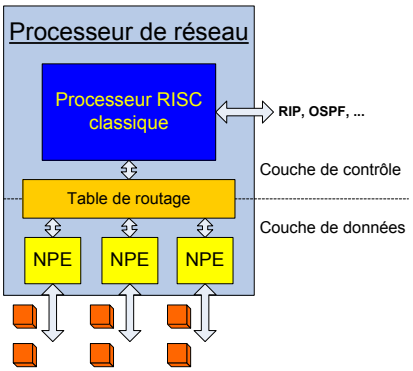
```
modprobe bonding mode=802.3ad  miimon=100
modprobe e100
ifconfig bond0 192.168.1.1 up
ifenslave bond0 eth0
ifenslave bond0 eth1
```

Routage dynamique

- Pourquoi Linux ?
  - Meilleur marché qu'un routeur Cisco
  - Plus flexible qu'un routeur normal
    - Tunnels
    - Firewall, NAT
    - VPN
    - Contrôle de trafic / QoS
  - ➢ Linux est idéal comme routeur d'accès peu coûteux qui intègre une multitude de fonctions

Routage et processeurs de réseau

- Combinent les performances des routeurs matériels avec la flexibilité des routeurs logiciels
- Couche contrôle
  - Mise à jour des tables de routage avec RIP, OSPF
  - Lent
- Couche des données
  - Acheminement des paquets
  - Recherche dans la table de routage
    - Réalisée sur processeurs spécialisés en 'microcode'



Routage dynamique dans Linux

Implémentations 'classiques' du monde Unix

- routed :
  - Démon de routage RIPv1 et RIPv2
- GateD :
  - Démon de routage RIP, OSPF, BGP, EGP
  - Maintenant sous licence propriétaire

Systèmes modèrnes

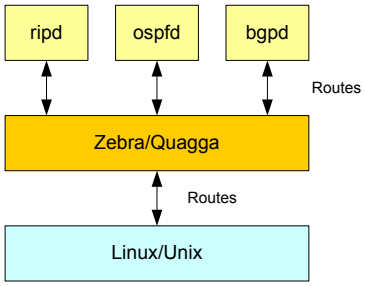
- Zebra / Quagga
  - RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4, BGP-4+
  - IPv4 et IPv6
  - Actuellement les systèmes de routage les plus complets
- XORP
  - RIPv2, RIPng, BGP-4
  - IPv4 et IPv6
  - Ré-implémentation complète en C++, orientée recherche/extensibilité
  - Licence BSD

Zebra et Quagga

- Quagga est le successeur de Zebra
  - Architecture et langage de commande presque identiques

Architecture

- Modulaire
- Les différents démons de routage communiquent avec Zebra/Quagga pour importer/exporter des routes
- Zebra/Quagga communique avec l'OS pour mettre à jour la table de routage



Configuration de Zebra/Quagga et RIP

- Fichiers de configuration
  - /etc/zebra/zebra.conf

```
hostname zebra_r1
password sesame
```

  - /etc/zebra/ripd.conf

```
hostname rip_r1
password ouvre-toi
```

- Démarrer les processus

```
> zebra -d
> ripd -d
```

- Configuration via une connexion Telnet sur la machine locale

Port	Démon
2601	Zebra
2602	RIP
2603	RIPng
2604	OSPF
2605	BGP

Configuration de Zebra

- Connexion :

```
> telnet 127.0.0.1 2601
Password: ******
zebra> show ip route
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
    Network          Next Hop       Metric From          Tag Time
C(i) 100.0.0.0/8      0.0.0.0            1 self            0
R(n) 110.0.0.0/8      100.0.0.2          2 100.0.0.2       0 02:39
C(i) 120.0.0.0/8      0.0.0.0            1 self            0
C(r) 192.168.128.0/24  0.0.0.0           1 self            0
C(r) 200.1.1.0/24     0.0.0.0            1 self            0
R(n) 200.1.2.0/24     100.0.0.2          2 100.0.0.2       0 02:39
R(n) 200.1.3.0/24     120.0.0.1          2 120.0.0.1       0 02:48
```

- Langage de commande très similaire à celui de Cisco
- Le démon Zebra permet
  - la configuration des interfaces réseau
  - d'afficher les routes

Configuration des interfaces

- Configuration

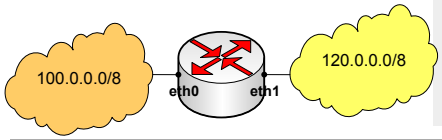
```
zebra> enable
zebra# configure terminal
zebra(config)# interface eth0
zebra(config-if)# ip address 10.192.74.100/25
zebra(config-if)# exit
zebra(config)# exit

zebra# show interface
Interface eth0
  index 2 metric 1 mtu 1500 <UP,BROADCAST,RUNNING,MULTICAST>
  HWaddr: 00:02:b3:01:01:01
  inet 10.192.74.100/25 broadcast 10.192.74.127
  ...
```

- Sauvegarde

```
zebra# write file
Configuration saved to /etc/zebra/zebra.conf
```


Configuration de RIPd

- A configurer
  - Version de RIP (1 ou 2)
  - Quelles routes annoncer
    - Un routeur peut parler RIP sur une interface et OSPF sur une autre
    - On peut décider si l'on veut p.ex. annoncer les routes OSPF via RIP
  - Interfaces/réseaux sur lesquels on utilise RIP

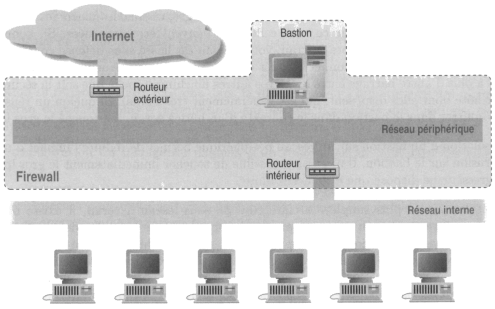



```
rip> enable
rip# configure terminal
rip(config)# router rip
rip(config-router)# version 2
rip(config-router)# redistribute connected
rip(config-router)# network 100.0.0.0/8
rip(config-router)# network 120.0.0.0/8

rip(config-router)# show running-config
Current configuration:
!
hostname rip
password rip
!
router rip
 version 2
 redistribute connected
 network 100.0.0.0/8
 network 120.0.0.0/8
!
line vty
!
End
rip(config-router)# write file
```

Firewall

- Un firewall empêche les dangers provenant d'Internet de se répandre à l'intérieur d'un réseau
  - Firewall interne entre VLAN

Firewall au niveau réseau
- Filtre de paquets (« routeur écran »)
- Fait partie d'une structure de défense à plusieurs couches



Firewall logiciel ou matériel

Firewall logiciel
- PC normal
- Linux, OpenBSD comme OS
- Effectue l'analyse et le filtrage des paquets en logiciel

- ➢ Flexible, peut effecteur d'autres fonctions
- ➢ Bon marché
- ➢ Peu performant
- ➢ Surface d'attaque plus grande (OS, services, performances)

Firewall matériel
- Matériel spécialisé (processeurs, mémoire, ...) pour analyser les paquets à haute vitesse
- OS réduit (p.ex. VxWorks ou propriétaire)

- ➢ Très bonnes performances, moins susceptibles à des attaques de DoS
- ➢ Niveau de sécurité plus élevé
- ➢ Cher

Processeurs de réseau
- Combinent les avantages des deux approches
- Filtrage de paquets est programmable en microcode
- Utilise des fonctions réalisées en matériel

Firewall « stateful » ou « stateless »

Firewall stateless

- Ne garde pas de trace des paquets déjà analysés
- N'a pas de mémoire des connexions établies, requêtes-réponses, ...

- Rapide
- Simple à configurer
- Susceptible à des attaques qui violent la sémantique des échanges de paquets

Firewall stateful

- Mémorise l'état des connexions
- Permet d'utiliser l'état d'une connexion comme critère de filtrage

- Empêche certains types d'attaques
- Plus lent
- Règles de filtrage plus complexes

Filtrage de paquets

- Les règles de filtrage définissent quels paquets doivent être bloqués
- Deux éléments de configuration:

1. Politique par défaut
   - Définit le comportement lorsque aucune règle spécifique n'est applicable
   - Politique « Interdiction par défaut »
2. Règles spécifiques
   - Indiquent les actions à prendre pour un certain type de paquets
   - Critères:
     - IP src/dst,
     - Port src/dst,
     - Protocole de transport,
     - État de la connexion

Firewall Linux

- Système « Netfilter » du noyau
- Utilitaire iptables pour la configuration des règles

Règles de filtrage

- Les règles sont organisées en tables:
  - Table « filter » : filtrage de paquets
  - Table « nat » : Translation d'adresses
- Les règles de chaque table sont organisées en chaînes
  - Chaîne « INPUT » : paquets qui ont le FW comme destinataire
  - Chaîne « OUTPUT » : paquets qui ont le FW comme source
  - Chaîne « FORWARD » : paquets acheminés par le FW
- La cible indique l'action à prendre si une règle est applicable:
  - Cible « ACCEPT » : le paquet peut passer
  - Cible « DROP » : le paquet est supprimé
  - Cible « REJECT » : comme DROP, mais un message ICMP est envoyé

Exemple de configuration Firewall « stateless »

```
# Etablir la politique "Interdiction par défaut" pour toutes les chaînes
> iptables -P INPUT DROP
> iptables -P OUTPUT DROP
> iptables -P FORWARD DROP
> iptables -F # "Flush" : efface toutes les règles spécifiques

# Autoriser le connexions SSH entre l'extérieur et le serveur 100.1.2.3
> iptables -t filter -A FORWARD --destination 100.1.2.3 --protocol tcp
        --sport 1024: --dport 22 -j ACCEPT
> iptables -t filter -A FORWARD --source 100.1.2.3 --protocol tcp
        --dport 1024: --sport 22 -j ACCEPT

# Autoriser les paquets ping (ICMP echo-request et echo-reply)
> iptables -t filter -A FORWARD --protocol icmp --icmp-type 8 -j ACCEPT
> iptables -t filter -A FORWARD --protocol icmp --icmp-type 0 -j ACCEPT

# Autoriser les connexions HTTP pour les machines internes (100.80.0.0/16)
> iptables -t filter -A FORWARD --source 100.80.0.0/16 --protocol tcp
        --dport 80 --sport 1024: -j ACCEPT
> iptables -t filter -A FORWARD --destination 100.80.0.0/16 --protocol tcp
        --sport 80 --dport 1024: -j ACCEPT
```

Firewall « stateful »

- Permet de tenir compte de l'état d'une connexion

États les plus importants

- NEW : établissement d'une nouvelle connexion (paquet TCP SYN)
- ESTABLISHED : connexion déjà établie
- RELATED : paquets en rapport avec une connexion déjà établie, p.ex. message d'erreur ICMP
- INVALID : correspond aux paquets qui ne font pas partie d'une connexion déjà établie

Exemple de configuration Firewall « stateful »

- Autoriser les réponses HTTP uniquement si une connexion a été établie préalablement

```
# Etablir la politique "Interdiction par défaut"
> iptables -P INPUT DROP
> iptables -P OUTPUT DROP
> iptables -P FORWARD DROP
> iptables -F

# Autoriser les connexions HTTP depuis les machines internes
> iptables -t filter -A FORWARD -m state --state NEW,ESTABLISHED
        --source 100.80.0.0/16 --protocol tcp
        --sport 1024: --dport 80 -j ACCEPT
> iptables -t filter -A FORWARD -m state --state ESTABLISHED
        --destination 100.80.0.0/16 --protocol tcp
        --sport 80 --dport 1024: -j ACCEPT
```

Translation d'adresses NAT

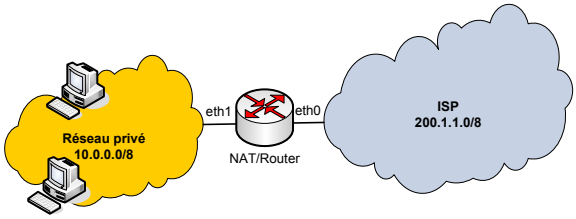
- Types de NAT
  - NAT statique
    - Une adresse publique est assignée à chaque adresse privée de manière fixe
    - Traduction à l'aide d'une table fixe
  - NAT dynamique
    - L'équipement NAT dispose d'un pool d'adresse publiques
    - Les adresses publiques sont allouées temporairement à une machine dès qu'elle établit une connexion avec l'extérieur
    - Le nombre de connexions simultanées est limité par le nombre d'adresses publiques disponibles
  - NAT/PT
    - Toutes les machines internes utilisent la même adresse publique pour les connexions sortantes
    - Les machines internes sont distinguées à l'aide du port TCP/UDP
    - Plus de 60'000 connexions simultanées avec une seule adresse publique

NAT dans Linux

- Utilise la fonction « Connexion tracking » du noyau
- Configuration avec iptables
- Linux permet les variantes suivantes de NAT
  - Source NAT
    - Variante de NAT/PT
    - Permet d'utiliser une ou plusieurs adresses publiques
    - Méthode principale de NAT, si les adresses publiques sont fixes
  - Masquerading
    - Similaire à NAT/PT
    - Utilise une seule adresse publique
    - Utilisable lorsque l'adresse publique est assignée dynamiquement (DHCP)
    - Indique uniquement l'interface de sortie

Exemple
Configuration de « Source NAT »

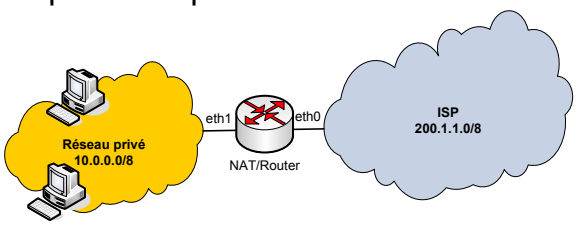
- L'entreprise dispose de trois adresses publiques: 200.1.1.1, 200.1.1.2, 200.1.1.3




```
iptables -t nat -F      # Effacer les anciennes règles NAT
iptables -t nat -A POSTROUTING -o eth0 -j SNAT -to 200.1.1.1-200.1.1.3
```


Exemple
Configuration de « Masquerading »

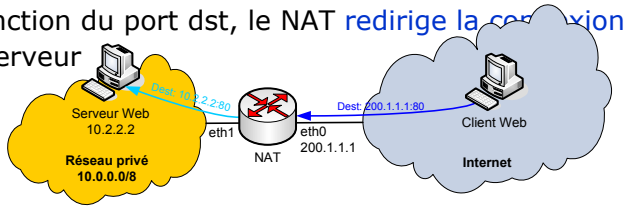
- Une seule adresse IP publique, assignée dynamiquement par DHCP




```
iptables -t nat -F      # Effacer les anciennes règles NAT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

« Destination NAT » dans Linux

- Permet à une machine externe de contacter un serveur interne ayant une adresse privée
- La machine externe envoie la requête sur l'adresse publique du NAT, mais sur le bon port TCP/UDP de destination
- En fonction du port dst, le NAT redirige la connexion vers le bon serveur




```
> iptables -t nat –F   # Effacer les anciennes règles
> iptables -t nat -A PREROUTING -p tcp --destination-port 80 -i eth0
        -j DNAT --to 10.2.2.2:8080
```

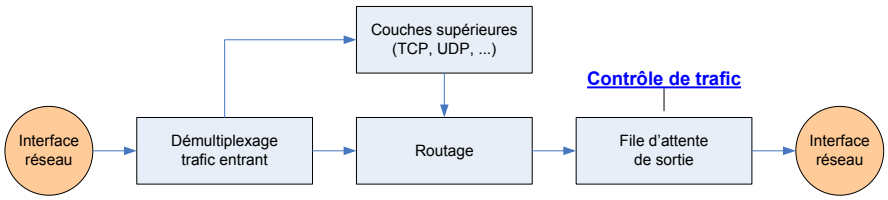

Contenu

Contrôle de trafic et QoS

- Mécanismes qui visent à influencer les performances des flux de trafic
  - Optimiser l'utilisation du réseau en évitant des congestions
  - Modifier la répartition de la bande passante entre les différents flux
- Exemples d'applications
  - Limiter la bande passante pour certains ordinateurs
  - Limiter la bande passante vers certains ordinateurs
  - Partager équitablement la bande passante
  - Réduire la congestion du réseaux

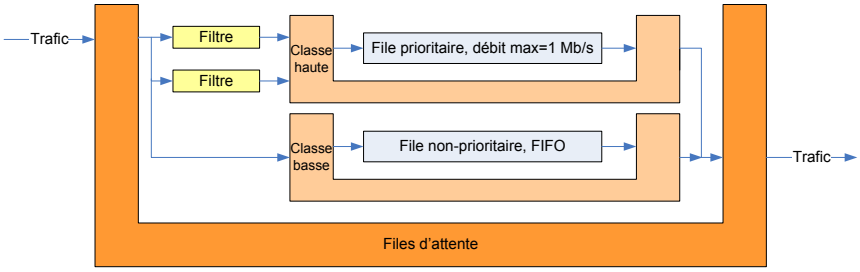
Contrôle de trafic dans Linux

- Utilitaire de configuration: tc
  - Fait partie de la suite d'utilitaires iproute2
  - Syntaxe complexe
- Compilateur tcc
  - Traduit des descriptions de configuration en commandes tc
- Ne s'applique qu'au trafic de sortie



Exemple d'une configuration

- Un paquet entrant est examiné par plusieurs filtres
- S'il correspond à un des filtres, le paquet est assigné à la classe correspondante
- Chaque classe contient alors une partie du trafic
- Elle gère les paquets à l'aide d'une file d'attente



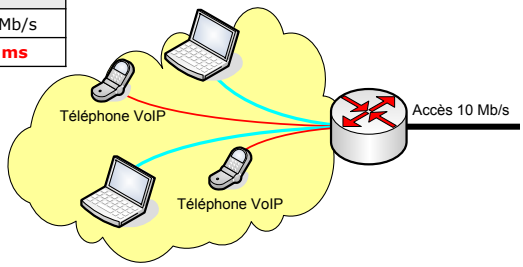
Types de files d'attente

- Appelés « qdisc » (*Queueing Discipline*)
- **PRIO** : file d'attente prioritaire
- **CBQ** (*Class Based Queueing*) : Permet de repartir la bande passante totale sur différentes classes
- **HTB** (*Hierarchical Token Bucket*) : Similaire à CBQ, mais avec un algorithme plus efficace
- **RED** (*Random Early Discard*) : Implémente l'algorithme de contrôle active de files d'attente RED qui élimine des paquets de manière probabilistique afin d'éviter une congestion du réseau
- **SFQ** (*Stochastic Fair Queuing*) : File d'attente qui distribue équitablement la bande passante disponible entre différentes connexions. Elle évite qu'une connexion agressive monopolise un lien
- et plusieurs autres

Exemple de configuration: Scénario VoIP

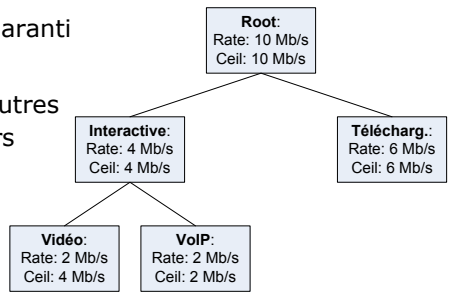
- Multiplexage de flux TCP et VoIP (< 1 Mb/s)
- Mesures de performances sans QoS (FIFO)

Mesures avec flux séparés	
TCP seul : débit	9.44 Mb/s
VoIP seul : délai de transit	1.3 ms
Mesures avec flux multiplexés	
TCP : débit	8.44 Mb/s
VoIP : délai de transit	**30.5 ms**



File d'attente HTB

- HTB (*Hierarchical Token Bucket*)
  - Permet de subdiviser la bande passante de manière hiérarchique
  - Paramètres pour chaque classe :
    - « Rate » : débit (moyen) garanti
    - « Ceil » : débit maximum, emprunté au parent, si d'autres enfants n'épuisent pas leurs débits



Exemple de configuration VoIP

- Réserver 2 Mb/s pour les flux VoIP
- Fichier voip.tcc

```
#include "fields.tc"
#include "ports.tc"
dev eth0 {
   egress {
      class ( <$voip> ) if ip_proto == IPPROTO_UDP ;
      class ( <$other> ) if 1 ;
      htb {
         class (rate 10Mbps, ceil 10Mbps) {
            $voip = class (rate 2Mbps, ceil 2Mbps);
            $other = class (rate 8Mbps, ceil 8Mbps);}}}}
```

- Génère les commandes tc

```
tc qdisc add dev eth0 handle 1:0 root dsmark indices 4 default_index 0
tc qdisc add dev eth0 handle 2:0 parent 1:0 htb
tc class add dev eth0 parent 2:0 classid 2:1 htb rate 1250000bps ceil 1250000bps
tc class add dev eth0 parent 2:1 classid 2:2 htb rate 250000bps ceil 250000bps
tc class add dev eth0 parent 2:1 classid 2:3 htb rate 1000000bps ceil 1000000bps
tc filter add dev eth0 parent 2:0 protocol all prio 1 tcindex mask 0x3 shift 0
tc filter add dev eth0 parent 2:0 protocol all prio 1 handle 2 tcindex classid 2:3
tc filter add dev eth0 parent 2:0 protocol all prio 1 handle 1 tcindex classid 2:2
tc filter add dev eth0 parent 1:0 protocol all prio 1 u32 match u8 0x11 0xff at 9 classid 1:1
tc filter add dev eth0 parent 1:0 protocol all prio 1 u32 match u32 0x0 0x0 at 0 classid 1:2
```


Mesures VoIP

Sans QoS

Mesures avec flux séparés		
	TCP seul : débit	9.44 Mb/s
	VoIP seul : délai de transit	1.3 ms
Mesures avec flux multiplexés		
	TCP : débit	8.44 Mb/s
	VoIP : délai de transit	**30.5 ms**

Avec QoS

Mesures avec flux séparés		
	TCP seul : débit	**7.69 Mb/s**
	VoIP seul : délai de transit	1.3 ms
Mesures avec flux multiplexés		
	TCP : débit	7.69 Mb/s
	VoIP : délai de transit	**1.7 ms**

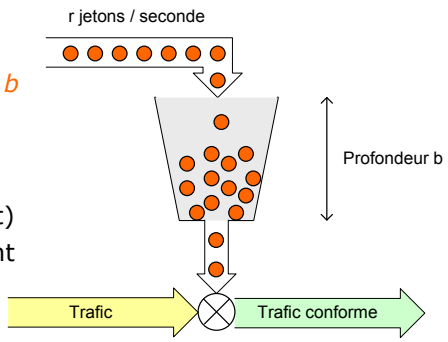
Exemple de configuration: Scénario ISP

- SLA : « *Service Level Agreement* »
  - Définit le service à fournir par l'ISP à un client professionnel
  - Tient souvent compte de rafales de trafic
  - Exemple
    - Débit moyen garanti : 2 Mb/s
    - Le client peut envoyer des rafales d'au maximum 500 kB à un débit maximum de 4 Mb/s
- Paramètres du SLA :
  - CIR (*Committed Interface Rate*)
  - CBS (*Committed Burst Size*)
  - PIR (*Peak Information Rate*)
- Comment mesurer si le client respecte ces paramètres ?

Le Seau à Jetons (*Token Bucket*)

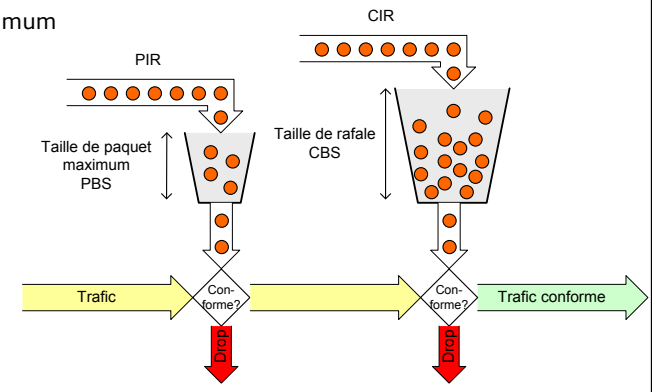
Algorithme de contrôle de trafic

- Le seau est rempli à une vitesse de *r* jetons/s
- Le seau peut contenir jusqu'à *b* jetons
- Un paquet à transmettre consomme un nombre de jetons (p.ex. 1 jeton par octet)
- Si suffisamment de jetons sont disponibles, le paquet est conforme au profile de trafic
- Les paquets non conformes sont retardés ou écartés



Contrôle du SLA avec des seau à jetons

- SLA:
  - CIR : débit moyen garanti
  - CBS : taille de rafale
  - PIR : débit maximum
- Contrôle avec 2 seau à jetons



Réalisation avec Linux

- Définition d'un double seau à jetons par client
  - Commande « DLB »
- Les paquets non conformes sont supprimés (ou marqués)
  - Commande « DLB_else_drop »

```
#include "fields.tc"
#include "meters.tc"
$client1_sla = DLB(cir 2Mbps, cbs 500kB, pir 4Mbps, pbs 1500B);
$client2_sla = ...
dev eth0 {
   egress {
     class (<$client1>) if ip_src == 1.2.3.4 && DLB_else_drop($client1_sla);
     class (<$client2>) if ...
     htb {
         class (rate 100Mbps, ceil 100Mbps) {
              $client1 = class (rate 4Mbps, ceil 4Mbps);
              $client2 = ...
} }   }    }
```