Administration des réseaux

Alexandre Kervadec

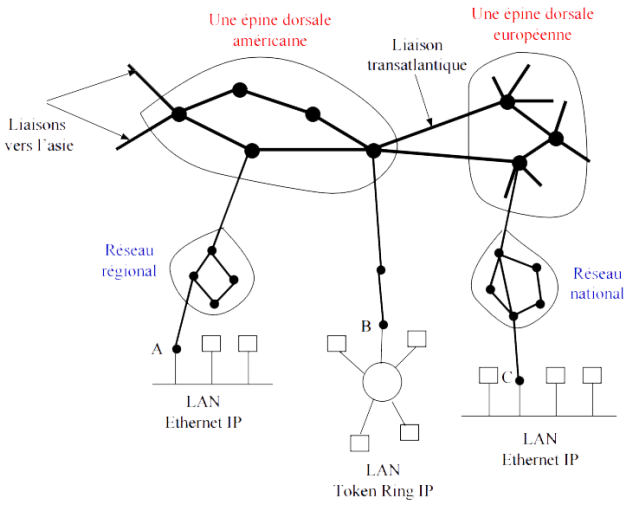
***Résumé*—Notes du cours d'administration des réseaux de A.Guermouche**

I. Modèle TCP/IP

A. Le visage d'internet

- Une construction à partir du "bas"
  - réseau local (laboratoire, département)
  - réseau local (campus, entreprise)
  - réseau régional
  - réseau national
  - réseau mondial
- 3 niveaux d'interconnexion
  - postes de travail (ordinateur, terminal, ...)
  - liaisons physiques (câble, fibre, RTC, ...)
  - routeurs (équipement spécialisé, ordinateur, ...)

C'est un ensemble de sous-réseaux indépendants (Autonomous System) et hétérogènes qui sont inter-connectés (organisation hiérarchique)



B. Architecture TCP/IP

L'architecture TCP/IP est une version simplifiée du modèle OSI.

- **Application** : FTP, WWW, Telnet, SMTP, ...
- **Transport** : TCP, UDP (entre 2 processus aux extrémités)
  - TCP : transfert fiable de données en mode connecté
  - UDP : transfert non garanti de données en mode non-connecté
- **Réseau** : IP (routage)
- **Physique** : transmission entre 2 sites
  - TCP : Transport Control Protocol
  - UDP : User Datagram Protocol
  - IP : Internet Protocol

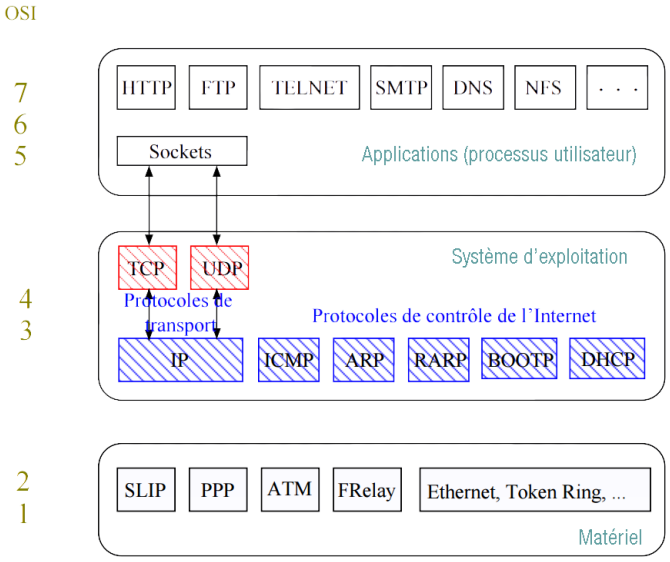


FIGURE 1. Architecture TCP/IP

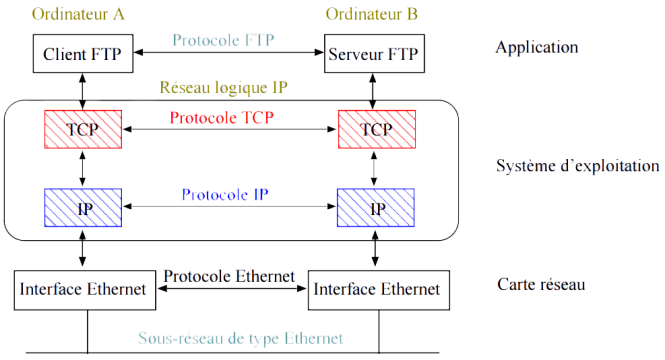


FIGURE 2. Sous-réseau IP

Professeur : A.Guermouche

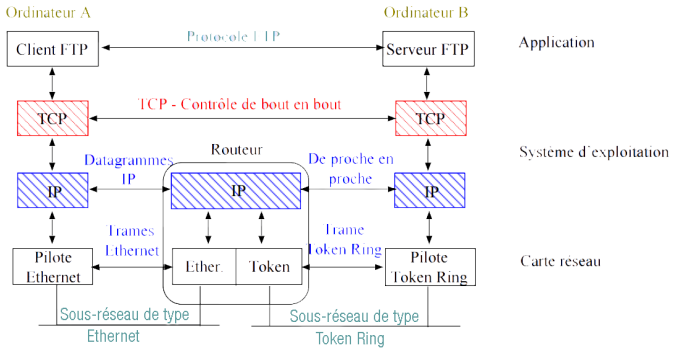


FIGURE 3. Prise en compte de l'hétérogénéité

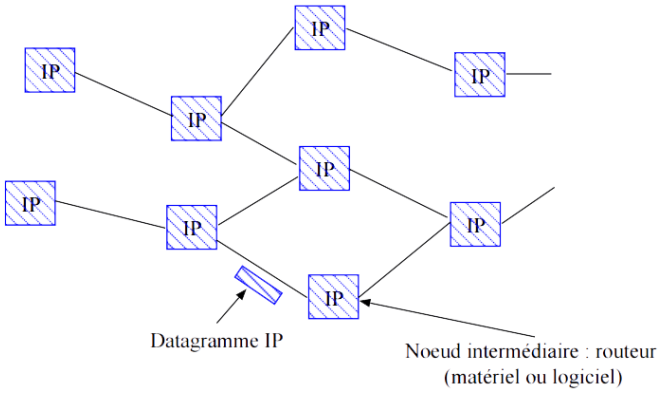


FIGURE 4. Couche réseau : communication entre machines

C. Protocole IP

IP - Protocole d'interconnexion, best-effort
- acheminement de *datagrammes* (mode *non-connecté*
- peu de fonctionnalités
- pas de garanties, simple mais robuste (défaillance d'un noeud intermédiaire)

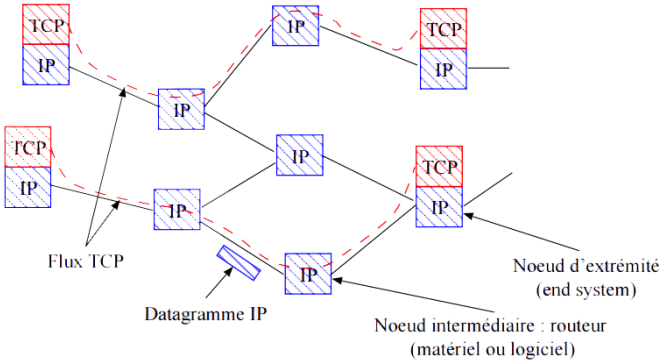


FIGURE 5. Couche réseau : communication entre applications

D. Protocole TCP

TCP - Protocole de transport *de bout en bout*
- uniquement présent aux *extrémités*
- transport *fiable* de *segments* (mode *connecté*)
- protocole complexe (retransmission, gestion des erreurs, séquencement, ...)

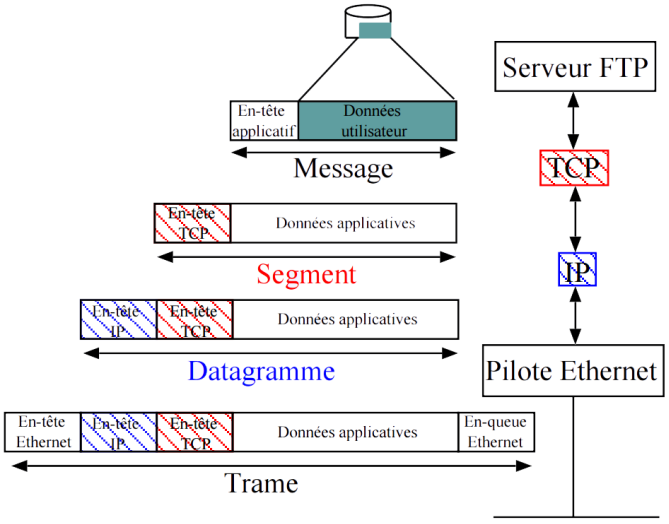


FIGURE 6. Couche réseau : communication entre applications

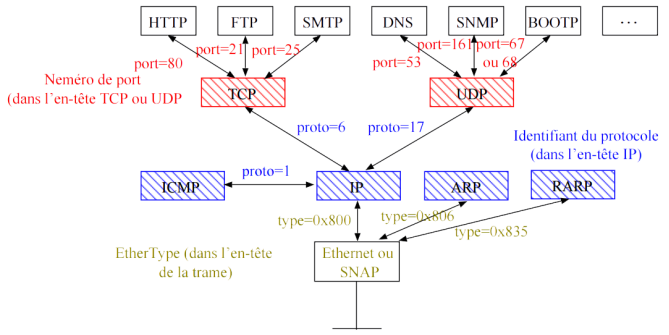


FIGURE 7. Identification des protocoles

E. Identification des protocoles

- Une adresse de transport = une adresse IP + un numéro de port (16 bits) → adresse de socket
- Une connexion s'établit entre une socket source et une socket destinataire → une connexion = un quintuplé (proto, src, port src, dest, port dest)
- Deux connexions peuvent aboutir à la même socket
- Les ports permettent un multiplexage ou démultiplexage de connexions au niveau transport
- Les ports inférieurs à 1024 sont appelés *ports réservés*

F. Protocole UDP

UDP (RFC 768) - User Datagram Protocol :
- protocole de transport le plus simple
- service de type best-effort (comme IP)
  - les segments UDP peuvent être perdus
  - les segments UDP peuvent arriver dans le désordre

— mode non connecté : chaque segment UDP est traité indépendamment des autres

Pourquoi un service non fiable sans connexion ?
— simple donc rapide (pas de délai de connexion, pas d'état entre émetteur/récepteur)
— petit en-tête donc économie de bande passante
— sans contrôle de congestion donc UDP peut émettre aussi rapidement qu'il le souhaite

Les utilisations de l'UDP :
— Performance sans garantie de délivrance
— Souvent utilisé pour les applications multimédias
  — tolérantes aux pertes
  — sensibles au débit
— Autres utilisations d'UDP
  — applications qui envoient peu de données et qui ne nécessitent pas un service fiable
  — exemples : DNS, SNMP, BOOTP/DHCP
— Transfert fiable sur UDP
  — ajouter des mécanismes de compensation de pertes (reprise sur erreur) au niveau applicatif
  — mécanismes adaptés à l'application

G. Protocole TCP

Transport Control Protocol (RFC 793, 1122, 1323, 2018, 2581) Transport fiable en mode connecté :
— point à point, bidirectionnel : entre deux adresses de transport (@IP src, port src) → (@IP dest, port dest)
— transporte un flot d'octets (ou flux)
  — l'application lit/écrit des octets dans un tampon
— assure la délivrance des données en séquence
— contrôle la validité des données reçues
— organise les reprises sur erreur ou sur temporisation
— réalise le contrôle de flux et le contrôle de congestion (à l'aide d'une fenêtre d'émission)

H. Exemples de protocole applicatif

— HTTTP - HyperText Transport Protocol
  — protocole du web
  — échange de requête/réponse entre un client et u serveur web
— FTP - File Tranfer Protocol
  — protocole de manipulation de fichiers distants
  — tranfert, suppression, création, ...
— TELNET - TELetypewriter Network Protocol
  — système de terminal virtuel
  — permet l'ouverture d'une session distante
— DNS - Domain Name System
  — assure la correspondance entre un nom symbolique et une adresse Internet (adresse IP)
  — bases de données réparties sur le globe

II. Routage dans TCP/IP

A. Routage dans IP

1) Sous-réseaux:

Un sous-réseau est un sous-ensemble d'un réseau de classe. Les intérêts d'un sous-réseau sont :
— diviser un réseau de grande taille en plusieurs réseaux physiques connectés par des routeurs (locaux ou distants)
— possibilité de faire coexister des technologies de réseaux différents
— diminution de la congestion du réseau par redirection du traffic et réduction des diffusions

Pour créer ce sous-réseau, il faut un ID de sous-réseau en séparant les bits d'ID d'hôtes en plusieurs sections.

2) Linux : Positionner/Modifier une adresse IP:

La manipulation des adresses IP se fait à l'aide de l'utilitaire *ifconfig*.

Syntaxe :

```
ifconfig interface @IP netmask masque ...
```

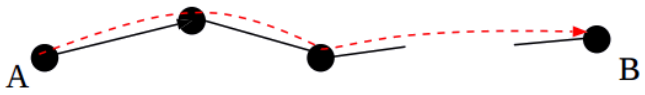
Exemple (configuration de l'interface eth0) :

```
ifconfig eth0 192.168.0.1 netmask 255.255.255.0
```

3) Problématique du routage:

Objectif : Acheminer des datagrammes IP d'une machine source A vers une machine destination B.

Problématique : Commet atteindre la machine B en connaissant son adresse IP ?



→ Nécessité d'identifier toutes les machines intermédiaires.

4) Routage IP : principe de base:

Définition :
— Processus de choix des chemins par lesquels les paquets sont transmis à la machine destinataire
— Processus basé sur une table de routage IP *IP routing table* contenant les informations relatives aux différentes destinations possibles et à la façon de les atteindre

Principe de base :
— L'émetteur ne connapit pas la route complète mais l'adresse du prochain site IP qui le rapprochera de la destination (prochain saut)
— Changements dynamiques possibles (en cas de panne)
— Extraire du datagramme l'adresse IP de destination (*IPDest*)
— Calculer l'adresse du réseau de destination (*IPRes*)
— Si *IPRes* = *IPLocal* alors
  — *IPDest* est directement accessible sur le réseau élémentaire commun

- La couche IP locale tente la translation d'adresse logique *IPDest* en adresse physique à travers la table maintenue en cache
- Si le réseau est de type Ethernet, le protocole utilisé est ARP
- Sinon les adresses physiques destinataires X21 auront dû être configurées à la main au préalable

— Sinon (ce n'est pas une adresse accessible, il faut alors consulter la table de routage IP locale) :
- Si *IPRes* est dans la table alors :
  - Router le datagramme selon les indications de la table (vers un autre nœud du réseau local, avec résolution adresse IP → adresse physique, ou vers un autre coupleur connecté à un réseau externe)
- SInon *IPRes* n'est pas dans la table alors :
  - Prendre la route par défaut indiquée dans la table
  - Router le datagramme selon les indications de l'entrée par défaut de la table (vers un autre nœud du réseau local, avec résolution d'adresse IP → adresse physique, ou vers un autre coupleur connecté à un réseau externe)

5) Tables de routage IP dans Linux:

Afficher les routes :

˜/# route

Destination	Gateway	Genmask	Iface
default	10.3.255.254	0.0.0.0	wlan2
10.3.0.0	*	255.255.0.0	wlan2

Ajouter une route par défaut :

˜/# route add defaut gw @passerelle

Ajouter une route utilisatn l'interface réseau *iface* vers un hôte particulier) :

˜/# route add −host @host gw @passerelle dev iface

Ajouter une route utilisant l'interface *iface* vers un réseau particulier :

˜/# route add −net @reseau netmask mask dev iface gw @gw

Pour les suppression de route, il suffit de remplacer add par del.

6) Mise en place d'un réseau:

Concentrateur (hub) : partage de bande passante entre les hôtes raccordés. **Commutateur (switch)** : pas d'interférences entre les connexions simultanées. **Routeur** : Pas d'interférences entre des connexions simultanées, possibilité de communication entre 2 réseaux logiques différents.

III. Réseaux privés

A. Introduction

Les adresse privées ont été crées pour :
- gérer la pénurie d'adresses au sein d'un réseau
- masquer l'intérieur d'un réseau par rapport à l'extérieur
- améliorer la sécurité pour le réseau interne

Il existe 2 mécanismes de translation d'adresses (NAT - Network Address Translation) :
- **statique** : association entre *n* adresses publiques et *n* adresses privées
- **dynamique** : association entre *1* adresse publique et *n* adresses privées

B. NAT statique

Intérêt :
- Uniformité de l'adressage dans la partie privée du réseau (modif. de la correspondance @pu/@pri facile)
- Sécurité accrue (tout passe par la passerelle NAT)

L'inconvénient majeur est que la pénurie d'@pu n'est pas résolue.

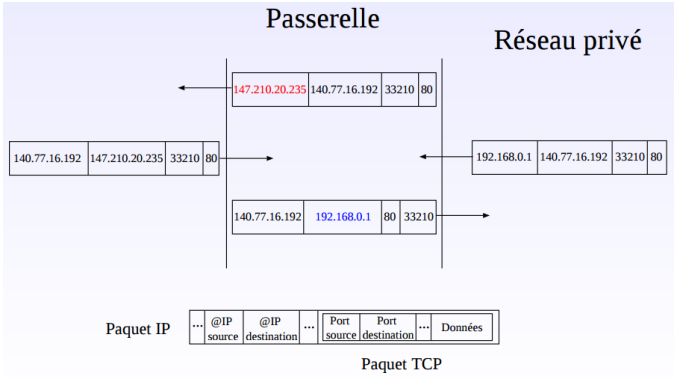


FIGURE 8. NAT statique : principe de fonctionnement

C. NAT dynamique : Masquerading

Intérêt :
- Plusieurs machines utilisent la même @pu pour sortir du réseau privé
- Sécurité accrue (tout passe par la passerelle NAT)

L'inconvénient majeur est que les machines du réseau interne ne sont pas accessibles de l'extérieur (impossible d'initier une connexion de l'extérieur).

1) Fonctionnement:

Comment le routeur fait-il la diférence entre les paquets qui lui sont destinés et les paquet à relayer :

```
A chaque nouvelle connexion :
  Modif. @src et port source:
  (@src_pri, port_src)->(@pu, port_src)
  Sauvegarder l'asso ds table NAT
Pour chaque paquet entrant :
  Chrcher une asso. (@dest., port_dest)
  Si exist une asso. ds table NAT Alors :
    Modif. (@dest., port_dest)
    Relayer le paquet
  Sinon
    /*Erreur de routage*/
  FinSi
```

Le routeur gère toutes les associations, il y a donc unicité de l'association (donc du port source après translation).

2) Problèmes liés à NAT dynamique:

— Nécessité d'implémenter une méthode spécifique aux protocoles n'utilisant pas de port
— Protocoles utilisant @IP, nécessité de mettre en place un "*proxy*" (FTP en mode actif par exemple)
— Nécessité de faire de la redirection de port (port mapping/forwarding)

3) Proxy:

Un proxy est un intermédiaire dans une connexion entre le client et le serveur.
Le client s'adresse toujours au proxy.
Le proxy est spécifique à une application donnée (HTTP, FTP, ...).

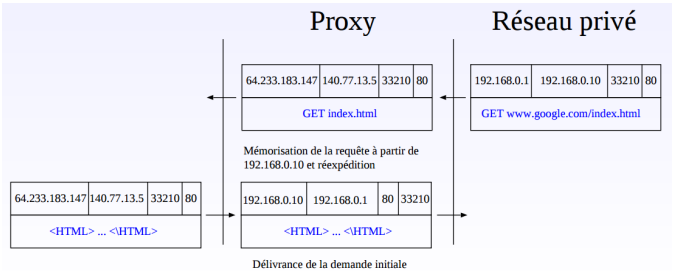


FIGURE 9. NAT statique : principe de fonctionnement

IV. IPTABLES ET NAT

La fonctionnalité de *firewall* est implémentée dans le noyau de Linux. Il y a 3 type de firewall filtrant :

— *ipfwadm* : jusqu'à la version 2.1.102 du noyau linux
— *ipchains* : entre les versions 2.2.0 et 2.4 du noyau linux
— *iptables* : à partir des noyaux 2.4

Nous nous intéresserons seulement à *iptables* dans ces notes de cours, car c'est la version que nous avons utilisé en TP. Se référer au cours complet pour avoir des informations sur les autres firewalls.

3 types de règles :

— **INPUT** : sont appliquées lors de l'arrivée d'un paquet
— **FORWARD** : sont appliquées lorsque la destination du paquet n'est pas le routeur
— **OUTPUT** : sont appliquées dès qu'un paquet doit sortir du routeur

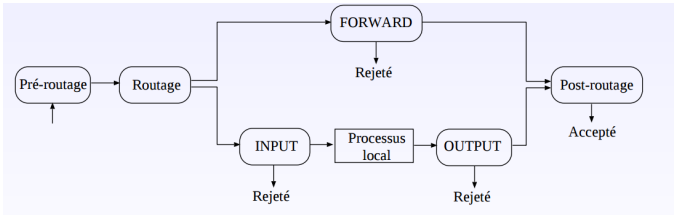


FIGURE 10. *iptables* : principe de fonctionnement

A chaque table une fonctionnalité :

— *filter* : filtrage des paquets
— *nat* : NAT
— *mangle* : marquage des paquets

FILTER		NAT	
INPUT	paquet entrant sur le routeur	PREROUTING	NAT de destination
OUTPUT	paquet émis par le routeur	POSTROUTING	NAT de source
FORWARD	paquet traversant le routeur	OUTPUT	NAT sur les paquets émis localement