



DE LA RECHERCHE À L'INDUSTRIE

# Sécurité réseau

Pascal MALTERRE

14 novembre 2022

Ce cours n'est pas exhaustif.

L'objectif principal est ici de rappeler quelques caractéristiques importantes sur la sécurité des réseaux basés sur IPv4 dans leurs implémentations les plus courantes.

## Première partie I

### Rappels et généralités

- ▶ Réseau = Interconnexion d'équipements qui échangent des informations
  - ▶ Les « équipements » sont les ordinateurs (au sens large), les périphériques (par ex : imprimantes), les objets connectés, etc.
  - ▶ Le support de communication peut être le fil électrique, le réseau téléphonique, la fibre optique, les ondes radio, etc.
- ▶ Quelques exemples de problèmes à résoudre : attribution et partage du support, traitement des erreurs de transmission, optimisation de la bande passante, acheminement des informations, etc.

### Definition (Intéropérabilité)

Capacité que possède un produit ou un système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs et ce sans restriction d'accès ou de mise en œuvre.

*« Be liberal in what you accept, and conservative in what you send »*  
– Jon Postel

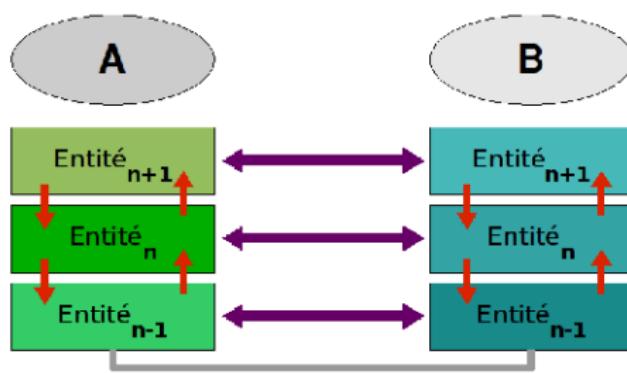
- ▶ Normes de fait vs. normes de droit
- ▶ L'ISO (International Organization for Standards) regroupe les organisations de normalisation nationaux : ANSI (us), AFNOR (France), etc.
- ▶ Pour les télécommunications : UIT/T (ex CCITT), ETSI (European Telecommunications Standards Institute)
- ▶ Les normes de l'Internet sont appelées RFC (Request For Comments) et sont élaborées par l'IETF (Internet Engineering Task Force)
  - ▶ Les RFC sont à la fois des normes de fait (les géants de l'informatique participent à leurs élaborations et c'est un choix collectif largement respecté) et des normes de droit (normalisés par l'IETF)
  - ▶ Consignes pour les implémentations
- ▶ Autres organismes : IEEE (réseaux locaux)

## Définition

Le « modèle en couche » est une manière de **décomposer** un mécanisme de communication en différents niveaux de détails.

Le modèle OSI est une norme **générale et abstraite** définissant un modèle basé sur 7 couches distinctes :

- ▶ (7) Application
- ▶ (6) Presentation
- ▶ (5) Session
- ▶ (4) Transport
- ▶ (3) Network
- ▶ (2) Data link
- ▶ (1) Physical



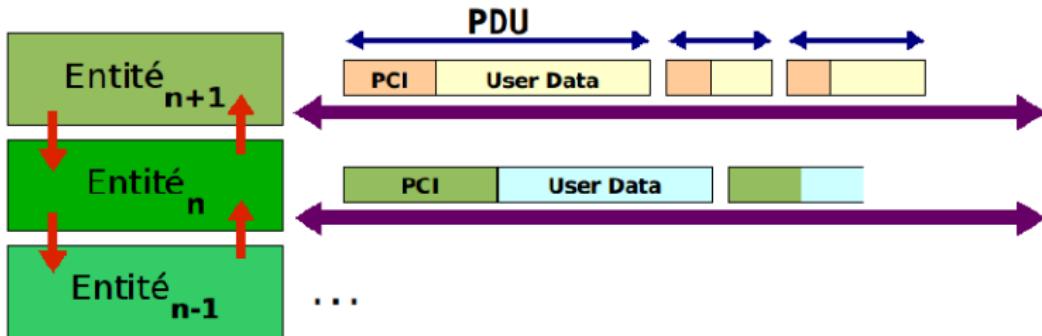
**PDU (Protocol Data Unit)**

Unité de données spécifique à un protocole de communication.

Le PDU( $n$ ) désigne le PDU caractéristique de la couche  $n$

**PCI (Protocol Control Information)****UD (User Data)**

Et on a :  $\text{PDU} = \text{PCI} + \text{UD}$

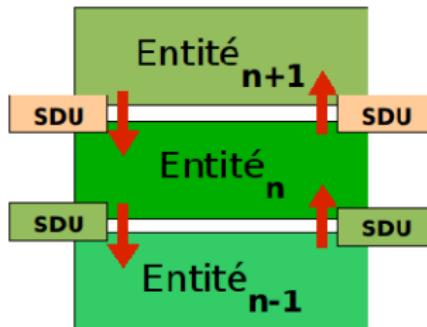


*fonctions<sub>n</sub>* Ensemble des traitements accomplis par l'entité de la couche  $n$

*service<sub>n</sub>* Ensemble des tâches que peut fournir la couche  $n$  à la couche au dessus :  $\{fonctions_n \cup service_{n-1}\}$

**SDU** (*Service Data Unit*)

Unité de données spécifique à un service. Le  $SDU_n$  désigne le SDU caractéristique de la couche  $n$



**Bourrage** Rajout d'octets arbitraires (« *padding* ») quand le champ *user data* du PDU( $n$ ) est plus long que la PDU( $n+1$ )

**Segmentation** Si la capacité du champ *user data* de le PDU( $n$ ) est plus courte que la taille de PDU( $n+1$ ), on doit segmenter les données de la PDU( $n+1$ ) en plusieurs PDU( $n$ )

**Concaténation** Le champ *user data* de le PDU( $n$ ) peut contenir plusieurs PDU( $n+1$ )

La **dissection des protocoles** est généralement une source très importante de vulnérabilités logicielles dans les applicatifs réseaux.

## Le support

Fils électriques, fibres optiques, ondes radio, etc.

## La couche 1 : physique

La SDU de la couche 1 est le **bit**. Sa fonction est de transmettre des bits en les encodant sous forme de signaux de données. Les informations de contrôle PCI peuvent être hors-bande ou non

## La couche 2 : lien

Le PDU de la couche 2 est l'**octet** ou la **trame** (bloc structuré de bits). Les fonctions de cette couche sont :

- ▶ La gestion du support (adressage, attribution, etc.)
- ▶ Le contrôle de flux et la détection d'erreurs

### La couche 3 : réseau

- ▶ Le PDU de la couche 3 est appelé **paquet** ou **datagramme**
- ▶ La fonction principale de cette couche est l'acheminement des paquets à travers un réseau de commutateurs

### La couche 4 : transport

Au sein d'un réseau, la couche 4 fournit le même genre de services que la couche 2 : contrôle de flux, détection d'erreurs, etc.

## La couche 5 : session

La couche 5 est responsable des échanges et de la gestion des connexions (ouverture, fermeture, problèmes de déconnexion, persistence des données, etc.). Par exemple : RPC

- ▶ La notion de session est souvent gérée au niveau applicatif (par exemple les sessions HTTP au travers des *cookies*)

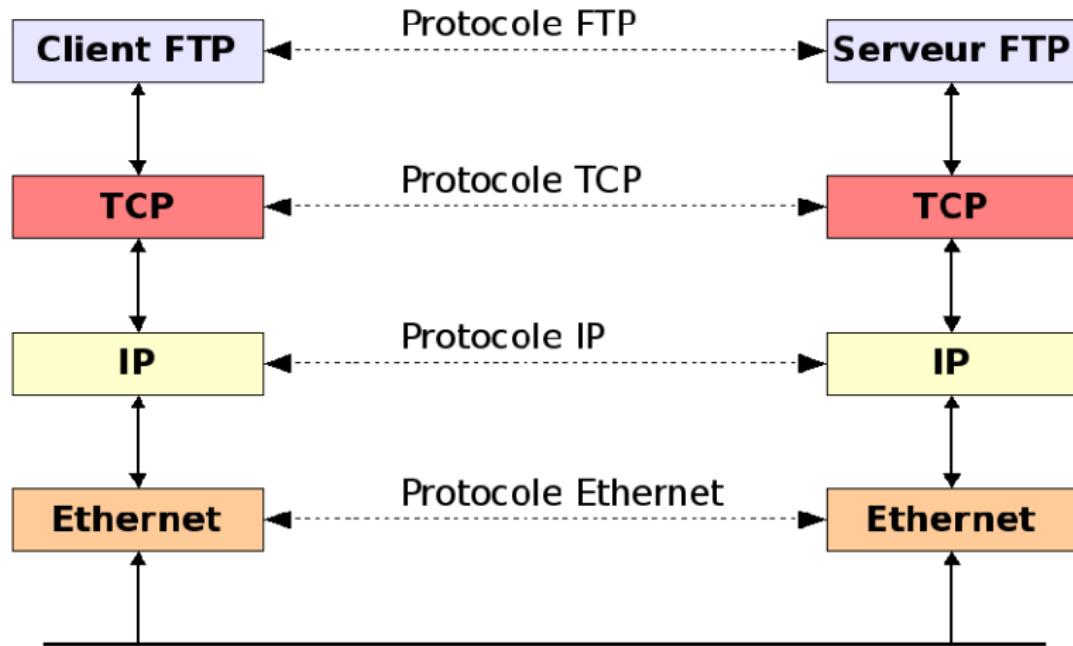
## La couche 6 : présentation

La couche 6 gère les problèmes de syntaxe ou de codage

- ▶ Par la conversion des données reçues de la couche supérieure, la couche présentation permet à deux applications de dialoguer même si elles utilisent des règles d'encodage différentes

## La couche 7 : application

SMTP, FTP, HTTP, etc.



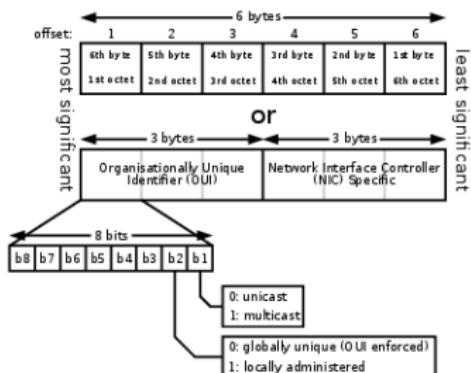
## Deuxième partie II

### Le réseau local

- ▶ Cette couche fournit une **liaison de données** entre des systèmes connectés à un même support physique (notion de « réseau local »)
- ▶ Les principales fonctions fournies au niveau de cette couche sont :
  - ▶ le partage du support (→ gestion des collisions);
  - ▶ l'acheminement des données (adressage local);
  - ▶ la délimitation des données.
- ▶ Le PDU peut être un octet (caractère) ou un ensemble structuré d'octets (trame ou *frame*)
- ▶ La couche de lien étant généralement complexe, elle peut être découpée en deux sous-couches distinctes :
  - LLC** (*Logical Link Control*)
  - MAC** (*Media Access Control*)

- ▶ Il existe différents types de liaisons :
  - ▶ point à point;
  - ▶ multipoints (configurations en étoile);
  - ▶ bus (plusieurs stations partagent le même support);
  - ▶ en anneau.
- ▶ La complexité augmente en fonction du nombre de stations
- ▶ Si à un instant donné le protocole ne peut être utilisé que par une seule station, il faut :
  - ▶ Gérer un contrôle d'attribution pour éviter les collisions
  - ▶ Gérer les collisions dans le protocole

- ▶ La sous-couche MAC est chargée de prendre en compte l'**adressage** au niveau de la couche liaison de données :
  - ▶ insertion des adresses source et destination lors de l'émission d'une trame;
  - ▶ filtrage en réception (on ne garde que les trames qui nous concernent).
- ▶ Utilisation d'une adresse **matérielle** (MAC – *Media Access Control Address*)



Source : Wikipedia

### Aspects sécurité :

- ▶ tracking
- ▶ filtrage basé sur l'adresse MAC

L'adresse MAC est codée en dur dans la carte réseau **mais** peut néanmoins être changée de manière logicielle.

- ▶ La délimitation est indispensable au contrôle d'intégrité
- ▶ Repérage du début et de la fin d'une PDU
  - ▶ Des caractères spéciaux ajoutés au début et/ou à la fin d'un bloc peuvent jouer le rôle de marqueurs ou indiquer la longueur du bloc
- ▶ La délimitation peut aussi être héritées des couches 0 ou 1
- ▶ Relation temporelle entre l'émetteur et le récepteur
- ▶ Délimitation des champs à l'intérieur d'une PDU

- ▶ Utilisation d'une somme de contrôle d'intégrité (*checksum*)
- ▶ Le *checksum* est calculé par l'émetteur et transmis avec la PDU. Cette somme est ensuite recalculée par le récepteur qui peut la comparer ainsi à la valeur transmise
- ▶ Une erreur de transmission peut néanmoins engendrer :
  - ▶ Un *checksum* erroné entraînant la retransmission
  - ▶ Une PDU jugée intègre à la réception
- ▶ Séquencements et acquittements

HDLC est une norme (initialement ISO-3309) spécifiant différents services de la couche de lien :

1. La **délimitation des données** au moyen d'un drapeau inséré au début et à la fin de chaque trame (0x7E ou 01111110b)
2. La gestion de trames de différentes types : informations, supervision, etc.
3. Détection des erreurs de transmission au moyen d'un contrôle d'intégrité sur la trame (FCS)
4. Contrôle de flux (mécanisme d'acquittement)
5. Adressage

Protocole de niveau 2 spécifié par l'IETF (RFC 1661)

- ▶ Connexion directe ("point à point") entre 2 noeuds du réseau
- ▶ Largement déployé par les ISP : RTC, PPPoE, VPN, etc.
- ▶ Protocole en 4 sous-couches dont les 3 premières sont issues de HDLC (délimitation, intégrité, contrôle logique)
  - ▶ La couche supérieure de PPP gère le multiplexage des protocoles
  - ▶ Le protocole LCP (*Link Control Protocol*) permet d'établir, configurer et tester la connexion de lien de données
  - ▶ Le protocole NCP (*Network Control Protocol*) est spécifique à chaque couche réseau encapsulée

Le terme « Ethernet » fait référence à un standard initialement publié en 1982 (DEC, Intel et Xerox) puis normalisé par le comité 802 de l'IEEE quelques années plus tard.

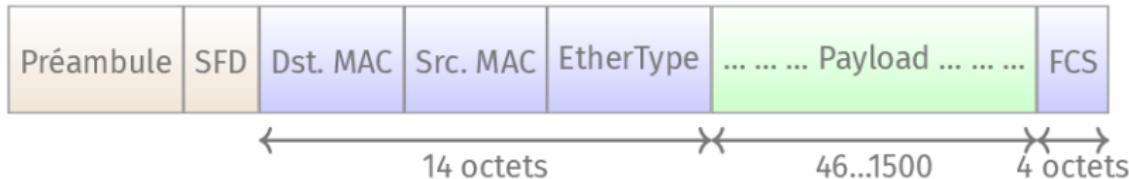
### Bus à méthode d'accès CSMA/CD

*Carrier Sense Multiple Access/Collision Detection*

#### **Principe de fonctionnement :**

- ▶ Chaque station peut émettre sur le bus si la voie est libre
- ▶ Une station détecte une collision quand ce qu'elle entend n'est pas identique à ce qu'elle a émis
- ▶ En cas de collision, la trame est réémise après un délai tiré au hasard dans un intervalle de plus en plus grand

- ▶ La taille minimale d'une trame (PDU) est de **64 octets**.
- ▶ La taille de la PCI est généralement de **18 octets** :
  - ▶ Les adresses matérielles (MAC) sont codées sur 6 octets (48 bits);
  - ▶ Un champ appelé « EtherType » (2 octets) identifie le protocole de la couche supérieure (par exemple 0x800 pour un paquet IP);
  - ▶ Somme de contrôle (FCS) sur 4 octets.
- ▶ La taille minimale des données transportées (UD) est de **46**
- ▶ « *EtherLeak* » est le nom générique des vulnérabilités (anciennes) causées par l'ajout d'octets de *padding* non initialisés<sup>1</sup>.



1. <https://www.cvedetails.com/cve/CVE-2003-0001/>

Le MTU est d'abord<sup>1</sup> une caractéristique de la **couche de lien** spécifiant la taille maximale des données que l'on peut envoyer dans une trame.

- ▶ Selon le contexte, le MTU peut être relatif à la longueur de la PDU ou aux données transportées (par exemple, pour Ethernet il est courant de dire que le MTU est 1500 octets).
- ▶ Pour les liaisons point à point type PPP, c'est une limite logique fixée en fonction de différents paramètres (temps de réponse souhaité, etc.).
- ▶ Augmenter la valeur du MTU offre des avantages et des inconvénients :
  - ▶ (+) optimise la bande passante (ratio PCI/UD)
  - ▶ (+) diminue le coût de traitement par paquet
  - ▶ (-) augmente la latence
  - ▶ (-) augmente les risques d'erreurs

---

1. On peut également considérer le MTU « de chemin », qui représente le plus petit MTU pour l'ensemble des tronçons de niveau 2 empruntés par les paquets réseaux dans un sens donné.

Chaque système d'exploitation offre des mécanismes pour envoyer et recevoir des paquets directement au niveau de la couche 2 :

- ▶ La réception de paquets est couramment mise en œuvre par les outils de capture passive (« *network sniffing* ») dans un contexte de diagnostic de problèmes réseaux et/ou de sécurité.
- ▶ L'envoi de paquets est notamment utilisée par les outils de *scan réseau*.

Le terme « Raw socket » est l'appellation générique de cette fonctionnalité.

### Example (man 7 packet)

*Packet sockets are used to receive or send raw packets at the device driver (OSI Layer 2) level. They allow the user to implement protocol modules in user space on top of the physical layer.*

- ▶ Nécessite des priviléges
- ▶ Pas vraiment standardisé (→ libpcap pour recevoir des paquets)

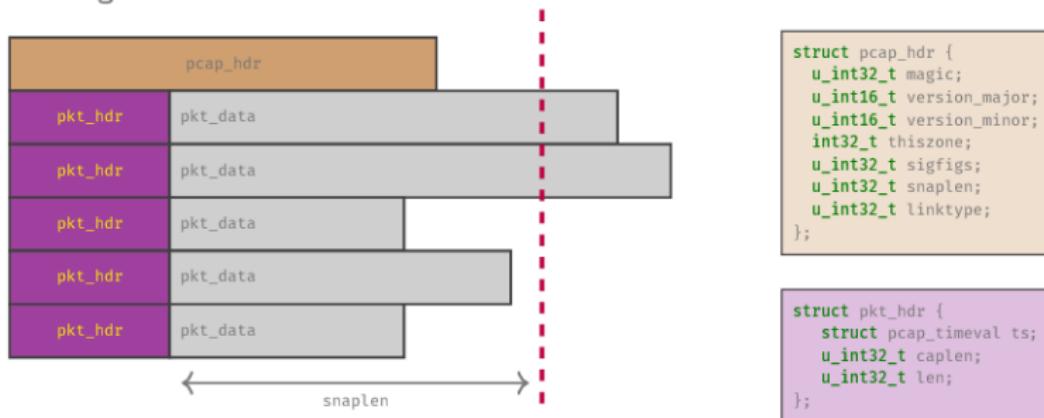
C'est l'association de deux actions :

1. Écouter ("renifler") le trafic transitant par une interface réseau
2. Enregistrer les données reçues de manière permanente

### Example (tcpdump)

```
$ sudo tcpdump -n -i eth0 -s 500 -C 20 -w capture.pcap
...
tcpdump: verbose output suppressed, use -v or -vv for full protocol
listening on wlp1s0 , link-type EN10MB (Ethernet)
^C
$ ls -alrt
-rw-r--r-- 1 root root 20000035 2015-04-04 23:34 capture.pcap
-rw-r--r-- 1 root root 20000084 2015-04-04 23:35 capture.pcap1
-rw-r--r-- 1 root root 20000278 2015-04-04 23:36 capture.pcap2
-rw-r--r-- 1 root root 7284159 2015-04-04 23:36 capture.pcap3
```

Le format PCAP est un format de fichier (popularisé par la *libpcap*) permettant d'enregistrer le trafic réseau.



## Remarques

- ▶ format simple et universel;
- ▶ pas de métainformations sur les données capturées.

- ▶ En fonctionnement « normal », une interface réseau ne remonte dans les couches supérieures *que* les trames dont l'adresse (MAC) destination est la machine elle-même, une adresse de *broadcast* ou de *multicast*.
- ▶ Une carte réseau peut être paramétrée en mode « *promiscuous* » pour faire remonter **toutes** les trames qu'elle reçoit.
- ▶ La quantité de trafic visible en mode « *promiscuous* » dépend du support physique et du mode de raccordement au niveau de la couche de lien.
- ▶ Une interface réseau en écoute passive n'est pas forcément silencieuse.

L'utilisation massive de commutateurs (*switches*) dans les réseaux Ethernet complique la capture du trafic réseau à grande échelle et nécessite l'utilisation de technologies spécifiques.

**Port Mirroring** nom générique de la fonctionnalité permettant à un équipement réseau de dupliquer le trafic d'un port donné vers un autre port.

- ▶ SPAN ou RITE;
- ▶ uniquement sur les équipements relativement évolués;
- ▶ traitement coûteux pouvant entraîner des problèmes de performances ou de stabilité;
- ▶ pertes de paquets, rejet des trames incorrectes, etc.

**Boîtiers TAP** (« *Test Access Port* »)

Équipement passif conçu spécifiquement pour la capture réseau.

- ▶ Déploiement en **coupe** (mode *fail-open*) du lien réseau que l'on veut écouter/capturer
- ▶ Les flux réseaux sont séparés et recopiés au niveau physique :
  - ▶ recopie des paquets sans la moindre altération;
  - ▶ peut nécessiter de re-fusionner les flux (« *channel bonding* »)
- ▶ Les boitier TAP sont similaires à des diodes et ne laissent passer le flux que dans le sens de la duplication.



La sécurité de la couche 2 est (surtout) une question de **protection physique**.

Si la protection physique est faible, l'attaquant peut accéder au support de communication.

- ▶ Cet accès peut être rendu **discret et permanent** grâce à la connexion d'un équipement spécifique (par ex : Raspberry Pi) [1] [2] [3].



En fonction des capacités de l'attaquant vis-à-vis du média :

- ▶ Écoute passive → **confidentialité**
- ▶ Utilisation active → +(intégrité/disponibilité/imputabilité)

Un réseau à accès contrôlé<sup>1</sup> repose sur le protocole 802.1x [4].

### Un peu de terminologie

- ▶ Les **supplicants** désignent les équipements d'extrême (par exemple les PC portables) qui cherchent à se connecter au réseau.
- ▶ Les **clients** (commutateurs, points d'accès, etc.) offrent des ports de connexion au réseau local aux supplicants.
- ▶ Les **serveurs AAA<sup>2</sup>** communiquent avec les clients pour autoriser (ou refuser) l'ouverture des ports après authentification des supplicants.
  - ▶ Le protocole RADIUS est généralement mis en œuvre pour la communication entre les serveurs et les clients.
  - ▶ L'authentification des supplicants se base sur le protocole EAP.

---

1. On laisse de côté le contrôle d'adresses MAC (permet juste de se protéger de l'erreur humaine).  
2. *Authentication, Authorization, Accounting*

-  **DarkVishnya : Banks attacked through direct connection to local network**  
<https://securelist.com/darkvishnya/89169/>
-  **Aaron Swartz : sur les traces d'une étoile filante du Net**  
<https://www.telerama.fr/medias/aaron-swartz-sur-les-traces-d-une-etoile-filante-1000011111.html>
-  **Notes about hacking with drop tools**  
<https://blog.erratasec.com/2018/12/notes-about-hacking-with-drop-tools.html>
-  **Recommandations de déploiement du protocole 802.1x pour le contrôle d'accès à des réseaux locaux**  
[https://www.ssi.gouv.fr/uploads/2018/08/guide\\_802.1x\\_anssi\\_pa\\_043\\_v1.pdf](https://www.ssi.gouv.fr/uploads/2018/08/guide_802.1x_anssi_pa_043_v1.pdf)

La fonction principale de la couche 3 est **l'acheminement des paquets** (ou datagrammes) au travers d'un réseau.

## Caractéristiques

- ▶ Différents types d'éléments : routeurs vs. équipement terminaux
- ▶ Adressage des éléments au sein du réseau
- ▶ Commutation de circuits vs. commutation de paquets
- ▶ Mode connecté et mode non-connecté

Une connexion réalisée par la couche 3 dans le mode commutation de circuit commence par la mise en place d'un **circuit virtuel**

- ▶ Les deux points du réseau sont **logiquement** connectés
- ▶ Cette connexion peut être permanente ou établie à la demande
- ▶ La couche 3 peut aussi prendre en charge le contrôle de flux et le séquencement
- ▶ Obligation de moyens ou de résultats
- ▶ Différents niveaux de PCI (connexion / transmission)
- ▶ Routage
- ▶ Vitesse de commutation
- ▶ Contrôle de flux
- ▶ Séquencement / duplication
- ▶ Facturation

En raison de la qualité de service exigée par les nouvelles applications (VoIP, etc.), les deux modes sont utilisés  
→ par exemple, au sein du coeur de réseau d'un opérateur, les datagrammes sont souvent encapsulés dans des circuits virtuels

IP (défini dans la RFC 791) fournit un service de transport de datagrammes (**commutation de paquets** donc sans connexion) et **non fiable**. La version 4 (IPv4) est la plus répandue.

- ▶ Les données transportées peuvent être altérées
- ▶ Les datagrammes peuvent être perdus, dupliqués ou arriver dans le désordre (pas de contrôle de flux, ni de séquencement)
- ▶ Les informations nécessaires à l'acheminement des paquets sont contenues dans le paquet lui-même

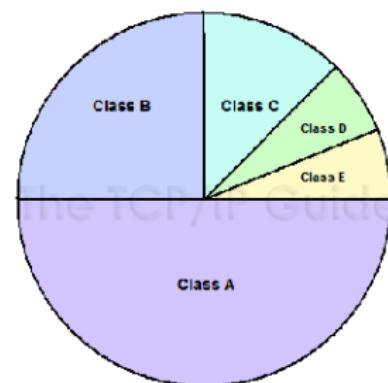
## Aspects sécurité

La couche IP n'apporte **aucune garantie de sécurité**, que cela soit en termes de confidentialité, d'intégrité, de disponibilité ou d'imputabilité!

- ▶ Notation décimale pointée : A.B.C.D
- ▶ Une adresse peut être découpée en deux parties (identifiant réseau et identifiant machine), historiquement en cinq grandes classes.

#### Découpage initial (1980-1993)

- ▶ A : 0.0.0.0 à 127.255.255.255
- ▶ B : 128.0.0.0 à 191.255.255.255
- ▶ C : 192.0.0.0 à 223.255.255.255
- ▶ D : 224.0.0.0 à 239.255.255.255
- ▶ E : 240.0.0.0 à 247.255.255.255



CIDR est un nouveau découpage (proposé en 1993) de l'espace d'adressage IP afin de palier aux déficiences du découpage historique en classes A, B, etc.

- ▶ Utilisation de préfixes (masques de réseau) **de longueur variable**, permettant à la fois :
  - ▶ une gestion décentralisée (découpage et délégation)
  - ▶ une diminution de la taille des tables de routage (aggrégation)
- ▶ Format d'un bloc CIDR : **A.B.C.D/n**

La RFC 3927 formalise<sup>1</sup> la méthode à utiliser pour qu'une interface réseau puisse s'auto-attribuer une adresse IP quand il n'y a pas de serveur DHCP et qu'elle ne fait pas l'objet d'une configuration statique.

- ▶ La plage réseau 169.254.0.0/16 est réservée<sup>2</sup> pour l'auto-attribution des adresses IP type « *Link-Local* »
- ▶ Ces adresses IP doivent être choisies **de manière aléatoire mais néanmoins reproductible** (l'algorithme peut se baser sur des éléments statiques<sup>3</sup> pour dériver *la même* adresse IP à chaque reboot de la machine)
- ▶ Les conflits d'adresse IP doivent être gérés (ARP *probe* → stratégie de type « *claim-and-defend* »)

1. L'auto-configuration des adresses IP locales existait déjà pour la plupart des OS...

2. À l'exception des 256 premières/dernières adresses IP, réservées à un usage futur.

3. Par exemple l'adresse MAC de l'interface.

La RFC 1918 spécifie plusieurs blocs d'adresses IP réservées aux **réseaux privés** (non routés sur Internet – « *Martian Address* ») :

- ▶ 10.0.0.0-10.255.255.255 (10.0.0.0/8 - bloc 24 bits)
- ▶ 172.16.0.0-172.31.255.255 (172.16.0.0/12 - bloc 20 bits)
- ▶ 192.168.0.0-192.168.255.255 (192.168.0.0/16 - bloc 16 bits)

### Aspects sécurité

Aujourd'hui, il est fortement déconseillé d'utiliser des adresses IP publiques pour des machines qui n'exposent pas de services sur Internet.

D'autres blocs d'adresses IP réservées à des usages particuliers ont été définies, par exemple dans la RFC 5737 :

- ▶ 192.0.2.0/24 (TEST-NET-1)
- ▶ 198.51.100.0/24 (TEST-NET-2)
- ▶ 203.0.113.0/24 (TEST-NET-3)

Broadcast et multicast sont mis en oeuvre lorsqu'une application souhaite envoyer un seul message à plusieurs récepteurs en même temps

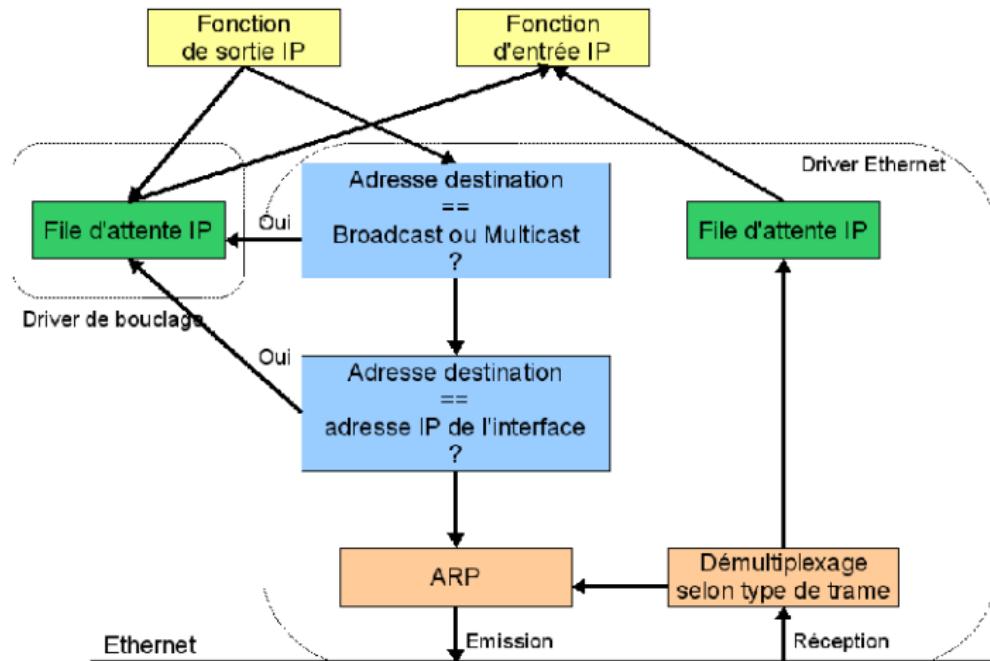
- ▶ Rappel : un filtrage est fait à tous les niveaux de la pile protocolaire
- ▶ Broadcast limité (255.255.255.255) et broadcast de réseau
- ▶ Multicast
  - ▶ Adresses de groupes : 224.0.0.0 à 239.255.255.255

Les implémentations IP supportent généralement une **interface de bouclage** (appelée « *loopback* ») permettant à deux processus s'exécutant sur une *même machine* de communiquer via un lien réseau IP.

- ▶ L'identificateur de réseau de classe A 127 est réservé au *loopback*
- ▶ Par convention, la plupart des systèmes utilisent l'adresse IP 127.0.0.1 et lui assignent le nom *localhost*
- ▶ Un datagramme envoyé à cette adresse IP ne doit apparaître dans aucun réseau

### Aspects sécurité

Cette interface est souvent utilisée pour **restreindre** l'accès à des services réseaux **qui n'ont pas besoin** d'être accessibles à distance (principe du moindre privilège).



1. <https://github.com/torvalds/linux/blob/master/drivers/net/loopback.c>

- ▶ Le périphérique enregistre les trames reçues dans la RAM (via un transfert DMA) en utilisant éventuellement plusieurs files différentes
  - ▶ au niveau matériel → *Receive Side Scaling* (RSS);
  - ▶ au niveau logiciel → *Receive Packet Steering* (RPS) / *Receive Flow Steering* (RFS)
- ▶ Une interruption matérielle (IRQ) est déclenchée (+ *polling NAPI*)
- ▶ *Large Receive Offload* (LRO) / *Generic Receive Offload* (GRO)
- ▶ Point d'entrée de la couche IP : ip\_rcv() / ip\_rcv\_core()<sup>1</sup>

```
/* Main IP Receive routine. */
static struct sk_buff *ip_rcv_core(struct sk_buff *skb, struct net *net)
{
    const struct iphdr *iph;
    u32 len;

    /* When the interface is in promisc. mode, drop all the crap
     * that it receives, do not try to analyse it. */
    if (skb->pkt_type == PACKET_OTHERHOST)
        goto drop;
```

1. [https://github.com/torvalds/linux/blob/master/net/ipv4/ip\\_input.c#L516](https://github.com/torvalds/linux/blob/master/net/ipv4/ip_input.c#L516)

- ▶ Les plus gros blocs (préfixes courts ou /8) sont directement gérés par l'IANA (*Internet Assigned Numbers Authority*)
- ▶ L'IANA délègue la gestion de chaque bloc aux RIRs (*Regional Internet Registries*) : RIPE, APNIC, ARIN, etc.
- ▶ Ces derniers délèguent ensuite les sous-blocs aux LIRs (*Local Internet Registries*) : ISP, universités, très grandes entreprises, etc.
- ▶ Généralement, les ISPs délèguent ensuite des blocs plus petits aux entreprises (de /24 à /29)

### Exercice n°1

Un hébergeur de services Internet loue un emplacement dans une baie ainsi que le bloc d'adresses 201.96.10.224/28

- ▶ Quel est le nombre d'adresses IP utilisables ?
- ▶ Donner l'adresse de broadcast et le masque de sous-réseau

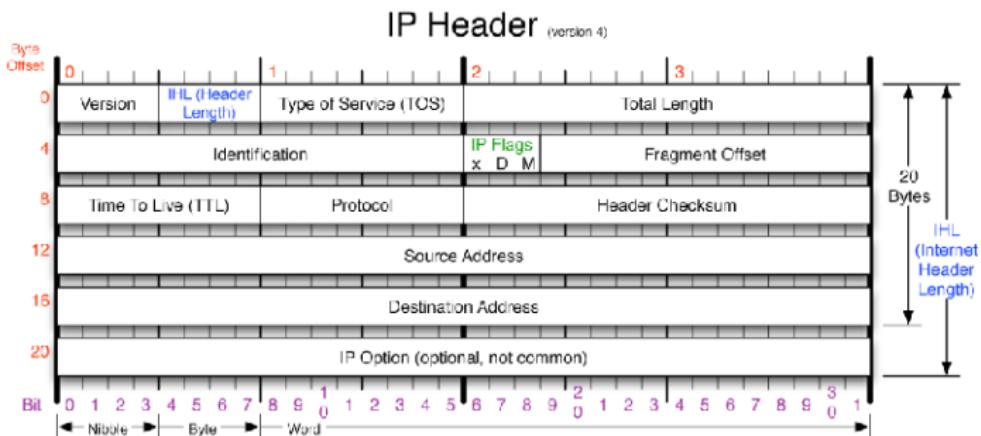
### Exercice n°2

Découper le réseau 10.0.8.0/24 en trois sous-réseaux et donner pour chacun d'entre-eux l'adresse de réseau, le broadcast et le masque

### Travaux dirigés

La copie d'écran ci-dessous est la sortie standard obtenue en exécutant la commande ifconfig sous Unix. Essayez d'interpréter le maximum d'informations sur la configuration réseau de la machine.

```
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
      groups: lo
      inet 127.0.0.1 netmask 0xffffffff
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      lladdr 00:04:23:bd:44:d2
      media: Ethernet autoselect (10baseTX full-duplex)
      status: active
      inet 192.168.10.254 netmask 0xfffffff0 broadcast 192.168.10.255
em1: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
      lladdr 00:04:23:bd:51:31
      media: Ethernet autoselect (none)
      status: no carrier
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      lladdr 00:14:22:0b:d5:e0
      media: Ethernet autoselect (10baseT half-duplex)
      status: active
pppoe0: flags=8851<UP,POINTOPOINT,RUNNING,SIMPLEX,MULTICAST> mtu 1492
      dev: bge0 state: session
      sid: 0x1a45 PADI retries: 0 PADR retries: 0 time: 03:15:29
      groups: pppoe egress
      inet 213.41.244.114 --> 0.0.0.1 netmask 0xffffffff
```

**Version**

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

**Header Length**

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

**Protocol**

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGMP	51 AH	115 L2TP

**Fragment Offset**

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

**IP Flags**

x D M

- x 0x80 reserved (evil bit)
- D 0x40 Do Not Fragment
- M 0x20 More Fragments follow

**RFC 791**

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

**Total Length**

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

**Header Checksum**

Checksum of entire IP header

La fragmentation IP intervient lorsqu'un datagramme IP est plus grand que le MTU de l'interface par laquelle il doit être envoyé.

La fragmentation peut se produire sur l'émetteur ou sur un routeur intermédiaire :

- ▶ chaque fragment devient un paquet IP indépendant;
- ▶ le réassemblage est effectué par la pile IP de la destination.

## Principe

- ▶ Le champ **identification** (ID) permet (avec les IP source et destination et le champ **protocol**) d'identifier les fragments d'un même paquet.
- ▶ Le bit **More Fragments** (MF) vaut 1 excepté pour le dernier fragment.
- ▶ Le champ **Fragment Offset** contient l'offset de ce fragment depuis le début du datagramme original (par unité de 8 octets).
- ▶ Le champ **Total Length** est modifié pour contenir la taille du fragment.

La fragmentation IP est transparente pour les couches supérieures mais, en pratique, induit des dysfonctionnement et des baisses de performances.

- ▶ Pour réassembler un paquet, il faut attendre que **tous** les fragments soient reçus (→ la perte d'un seul d'entre-eux empêche le réassemblage!).
- ▶ Le dernier fragment a rarement la taille optimale.
- ▶ La pile IP doit conserver en mémoire l'ensemble des fragments en cours de réassemblage (→ attaques possibles sur la disponibilité).
- ▶ Seul le premier fragment contient les entêtes des couches supérieures (par ex : TCP ou UDP) → impossibilité de filtrer sur des éléments spécifiques de ces entêtes, par exemple les numéros de port.

La couche IP fournit nativement un moyen d'éviter la fragmentation grâce au bit **Don't Fragment** (DF) présent dans l'entête.

## Aspects sécurité

- ▶ Si elle n'est pas soigneusement prise en compte dans les équipements de sécurité (pare-feu, passerelles VPN, etc.), sa gestion peut poser des problèmes de disponibilité.
- ▶ La fragmentation IP peut être utilisée de manière volontaire pour « camoufler » un trafic réseau (→ évasion IDS).

Au sein d'un même réseau local, l'adresse MAC (niveau 2) est utilisée pour la communication entre deux machines.

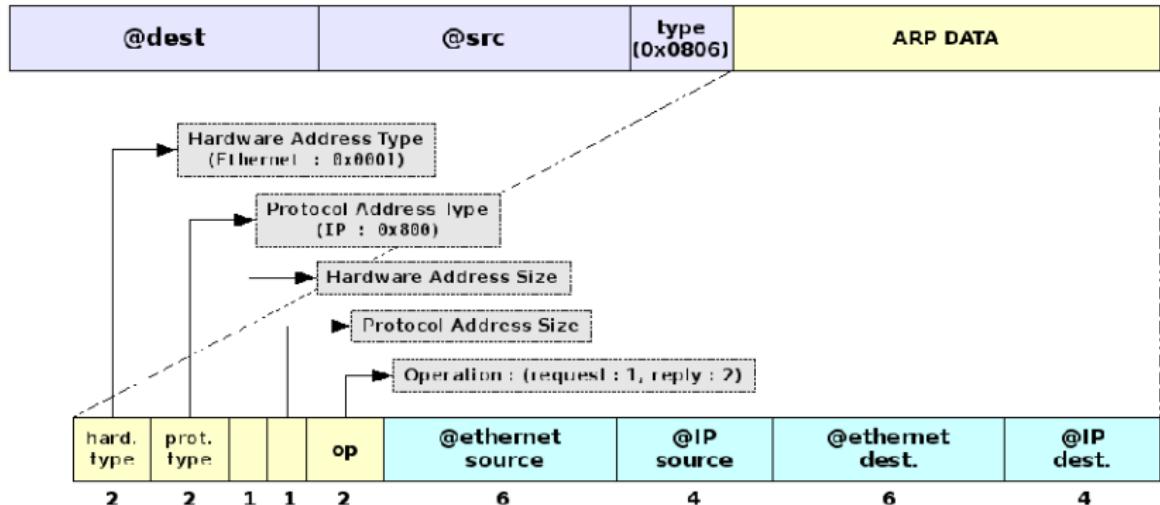
ARP est un protocole de niveau 2 (défini dans la RFC 826) fournissant une correspondance **dynamique** entre les adresses de la couche 3 (réseau) et les adresses de la couche 2 (lien).

Le problème auquel s'attaque ARP est : comment trouver l'adresse Ethernet (48 bits) d'une station à partir de son adresse IP (32 bits)

On suppose que la station 192.168.1.10 souhaite envoyer un datagramme IP à la machine 192.168.1.254; elle doit pour cela connaître l'adresse matérielle de la destination

- ▶ Elle envoie une requête ARP à **toutes** les machines connectées au support (*broadcast*)
- ▶ La requête contient la question : "que celui (ou celle) qui a l'adresse IP 192.168.1.254 se dénonce immédiatement!"
- ▶ La machine 192.168.1.254 reçoit la requête, et envoie une réponse (destinée uniquement à l'émetteur) contenant son adresse IP et son adresse matérielle

- ▶ ARP est un protocole général et peut fonctionner avec tout type de réseau supportant le *broadcast* : Token Ring, FDDI, etc.
- ▶ Pour des raisons d'efficacité, chaque station dispose d'une mémoire cache contenant les correspondances adresses IP/MAC récemment utilisées
  - ▶ Les caches des stations destinations sont mis à jour lorsqu'elles reçoivent la requête (@ADR\_IP\_SRC -> @ADR\_MAC\_SRC)
  - ▶ Le cache de la machine source est mis à jour lorsqu'elle reçoit la réponse (@ADR\_IP\_DST -> @ADR\_MAC\_DST)
- ▶ ARP est un protocole sans état car il n'y a pas de notion de transaction entre les requêtes et les réponses
- ▶ ARP Gratuit (*Gratuitous ARP*)



**Exercice 1**

Peut-on configurer plusieurs adresses IP attachées à une seule interface réseau ?

**Exercice 2**

Pour choisir une adresse IP sur un réseau où il n'y a pas de serveur DHCP, il est préférable de tester *au préalable* si cette adresse n'est pas déjà utilisée.

- Expliquer comment la couche ARP peut aider à résoudre ce problème
- Avec Scapy, créer et envoyer une trame ARP (« ARP probe ») permettant de tester la présence d'une IP particulière sur le réseau local
- Y a-t-il un autre logiciel installé sur votre distribution Linux permettant de faire cette vérification ?

**Les trames ARP ne sont pas authentifiées.** Les attaques sur le protocole ARP représentent une vulnérabilité classique et très connue de la couche de lien.

- ▶ À partir du moment où l'attaquant dispose d'un accès au média, il peut **forger** des fausses trames ARP (requêtes et/ou réponses) de manière à induire une mise à jour des caches avec de fausses informations.
  - ▶ « ARP *cache poisoning* »
- ▶ Le prérequis pour l'attaquant est de pouvoir « parler » sur le média de communication : **c'est une attaque active.**
- ▶ L'attaquant peut usurper l'adresse IP d'une station, voire même de la passerelle par défaut (→ *Man-in-the-Middle*)
  - ▶ « ARP *poison routing* »

Le protocole VRRP a pour objectif d'améliorer la disponibilité des routeurs pour les équipements paramétrés avec une table de routage statique.

- ▶ VRRP est un protocole de niveau IP : les messages sont encapsulés dans des datagrammes ayant pour destination l'adresse *multicast* 224.0.0.18 .
- ▶ Un **routeur virtuel** est défini par un identifiant numérique (VRID) et un ensemble d'adresses IP ( $VRIP_n$ ).
- ▶ Au sein d'un groupe de redondance VRRP, le rôle du routeur virtuel est attribué à un seul équipement (*master*) à un instant  $t$ .

Ce dernier :

- ▶ répond aux requêtes ARP des adresses IP
- ▶ envoie des messages à intervalle régulier (*Router Advertisements*)

La mise en place d'équipements redondants permet d'améliorer la disponibilité d'un système, que ce soit au niveau de la reprise sur panne (*fail-over*) ou bien de l'équilibrage de charge (*load-balancing*)

- ▶ Exemples de protocoles :
  - VRRP** (*Virtual Router Redundancy Protocol*)
  - HSRP** (*Hot Swap Redundancy Protocol*)
  - CARP** (*Common ARP Redundancy Protocol*)
- ▶ La redondance au niveau réseau est une notion différente de la redondance applicative
  - ▶ Pas de gestion des états (*stateless*)
  - ▶ Peu d'intelligence donc peu de complexité

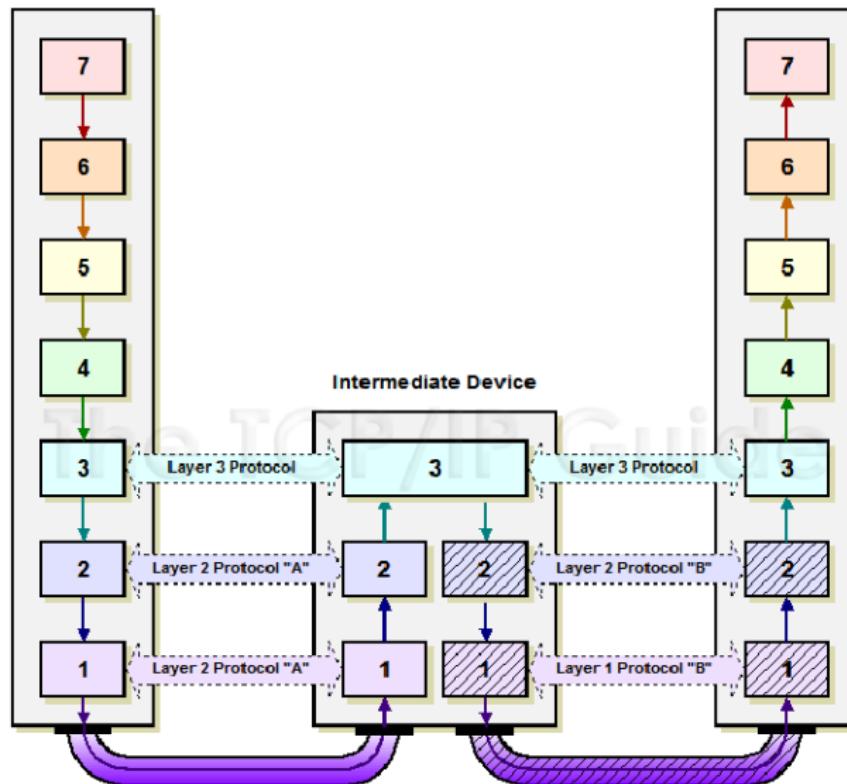
## Principe

Si la machine « connaît » la destination (*i.e.* elles sont connectées via un lien de niveau 2), elle lui envoie directement le datagramme, sinon elle l'envoie à une autre machine (routeur).

- ▶ Une station ne réémet jamais un datagramme
- ▶ IP ne connaît pas la route complète des destinations non connectées directement à la machine

## Aspects sécurité

**Rappel :** IP n'assure pas la confidentialité des données → un routeur peut « voir » (et même modifier) le contenu de tous les paquets qui transitent par son intermédiaire.



# Table de routage

- ▶ Chaque ligne de la table de routage contient les informations suivantes :
  - ▶ Destination (adresse de station ou de réseau)
  - ▶ Routeur de saut suivant (*next hop router*)
  - ▶ Spécification de l'interface de sortie
- ▶ Importance de la qualité de la table de routage

Destination	Gateway	Flags	Refs
Use	Mtu	Prio	Iface
default	88.171.200.254	UGS	5
1067787	-	8	rlo
10.0.20.0/25	10.0.20.193	UGS	4
2214988	-	8	vr0
10.0.20.128/26	10.0.20.193	UGS	0
0	-	8	vr0
10.0.20.192/26	link#2	UC	1
0	-	4	vr0
10.0.20.193	00:15:f2:3d:7c:55	UHLc	4
5056	-	4	vr0
88.171.200/24	link#1	UC	1

## Routage fixe

- ▶ La table de routage est fixe (réseau local)
- ▶ Support mal la panne ou la congestion

### Routage adaptatif centralisé

- ▶ Un superviseur centralise l'information sur l'état des commutateurs et des lignes, il décide du routage et informe les équipements en temps-réel
- ▶ La centralisation évolue vers une hiérarchie pour les gros réseaux
- ▶ Les informations transitent par le réseau supervisé lui-même

### Routage adaptatif décentralisé

- ▶ Chaque noeud maintient une table d'état des liens avec ses voisins
- ▶ Méthodes complexes, ce routage peut entraîner des boucles (RIP)

D'un point de vue macroscopique, Internet est partitionné en un ensemble de gros blocs appelés « systèmes autonomes » (AS). Chaque AS représente un ensemble de réseaux IP (préfixes CIDR distincts) **gérés par une même entité**.

- ▶ Le rattachement d'un préfixe CIDR à un AS (et son enregistrement dans un registre public) est fait par le RIR à qui appartient le bloc réseau.
- ▶ À l'intérieur de chaque AS, le routage IP repose généralement sur les protocoles classiques : OSPF, RIP, etc.
- ▶ Les routeurs situés aux frontières de chaque AS s'appuient sur le protocole BGP pour échanger avec leurs voisins des **politiques de routage**, c'est-à-dire les préfixes CIDR qu'ils permettent d'atteindre, directement ou indirectement.

En termes de sécurité, les risques de détournement de trafic IP dépendent de la confiance entre les opérateurs d'AS.

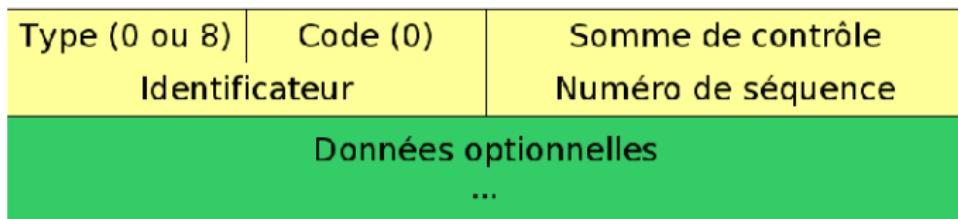
- ▶ ICMP communique les messages d'erreur et d'informations au niveau IP (spécifié dans la RFC 792)
- ▶ Un paquet ICMP peut encapsuler une requête ou un message d'erreur et on lui associe un type et un code, ainsi qu'un contenu de (dépendant de ce type et de ce code)
- ▶ Une erreur ICMP contient toujours l'entête IP et les 8 premiers octets du datagramme qui a causé l'erreur

Le programme « ping » est un outil de diagnostic permettant de vérifier si une machine est accessible.

- ▶ Envoi d'une requête ICMP de type **echo-request**
- ▶ Attente d'une réponse ICMP de type **echo-reply**

### Aspects sécurité

Les requêtes ICMP sont souvent filtrées pour des raisons de sécurité. Si une machine ne répond pas aux requêtes ping, cela ne signifie pas que les applications réseaux s'exécutant sur cette machine ne sont pas accessibles.



- ▶ Association requêtes/réponses → identifiant et num. séquence
- ▶ Enregistrement d'un *timestamp* les données optionnelles → calcul du RTT (*round-trip time*)

### Aspects sécurité

Les différences d'implémentations des protocoles (par exemple ici le choix des données optionnelles) peuvent donner des informations sur le système d'exploitation (→ *Passive OS Fingerprinting*).

## Définition

*La cartographie réseau consiste à agréger, pour un ensemble d'adresses IP, des informations sur les services et/ou les données qu'elles exposent, sur les acteurs qui les utilisent ainsi que sur les liens qu'elles peuvent avoir entre elles ou avec d'autres ensembles d'adresses IP.*

- ▶ La collecte des informations peut se faire de manière **active** (au moyen de scans réseau) ou **passive** (par la capture et l'analyse du trafic)
- ▶ Par nature, les résultats de la cartographie sont statiques et peuvent être généralement interprétés comme une « photo » du réseau à un instant  $t$
- ▶ La collecte des informations peut prendre un temps non négligeable
- ▶ La conservation de l'historique permet de suivre les évolutions dans le temps
- ▶ Les informations récoltées peuvent se révéler différentes selon le point d'observation (c'est-à-dire l'emplacement réseau).

Lorsqu'un routeur reçoit un datagramme dont le champ TTL vaut 0 ou 1, il ne doit pas le retransmettre. Au lieu de cela, le routeur élimine le datagramme et envoie à l'émetteur un message ICMP de type time-exceeded

Envoyer une série de paquets vers la même adresse IP en incrémentant le TTL permet d'énumérer les routeurs rencontrés grâce aux messages ICMP reçus.

## Remarques

- ▶ ICMP vs. UDP
- ▶ IP option obsolète « *Record Route* »
- ▶ Attention aux fausses interprétations<sup>1</sup> (notamment des temps de latence via le passage dans les *backbones* des opérateurs)

1. <http://movingpackets.net/2017/10/06/misinterpreting-traceroute/>

## TP

Après avoir consulté la documentation (man), faire quelques tests avec le programme traceroute

- ▶ comment interpréter les caractères « \* » ?
- ▶ comment énumérer les interfaces d'un routeur ?

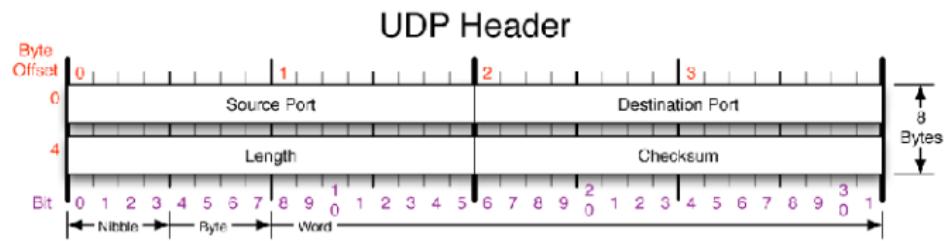
Lors d'une attaque informatique, la découverte de la topologie du réseau IP est généralement une des premières étapes effectuée par les pirates (ou par les consultants lors d'un audit de sécurité de type « test d'intrusion »).

1. Les outils de base (ping, traceroute, tcpdump, etc.) permettent d'avoir un premier aperçu de la structure du réseau
2. D'autres outils plus spécialisés (scapy, nmap, etc.) peuvent ensuite être utilisés
3. La découverte des couches supérieures (scans applicatifs) fournit également des informations intéressantes/complémentaires

UDP est un protocole de couche de transport simple, orienté datagramme

- ▶ Exactement un datagramme est généré pour chaque opération de sortie effectuée par un processus (contrairement à un protocole orienté flux)
- ▶ Aucune garantie de fiabilité
- ▶ La taille des datagrammes émis est de la responsabilité de l'application
- ▶ La somme de contrôle inclut l'entête UDP et les données

# Format du datagramme



Les spécifications originales sont données par la RFC 793 (1981) mais d'autres spécifications apportent des informations complémentaires et des extensions.

TCP est un protocole **orienté connexion, fiable**, et fournissant un **flux d'octets** aux couches supérieures

### end-to-end argument

Le design de TCP est principalement basé sur l'idée que l'intelligence du système doit reposer sur les deux extrémités et non sur les couches de liens et les équipements intermédiaires

## Flux d'octets

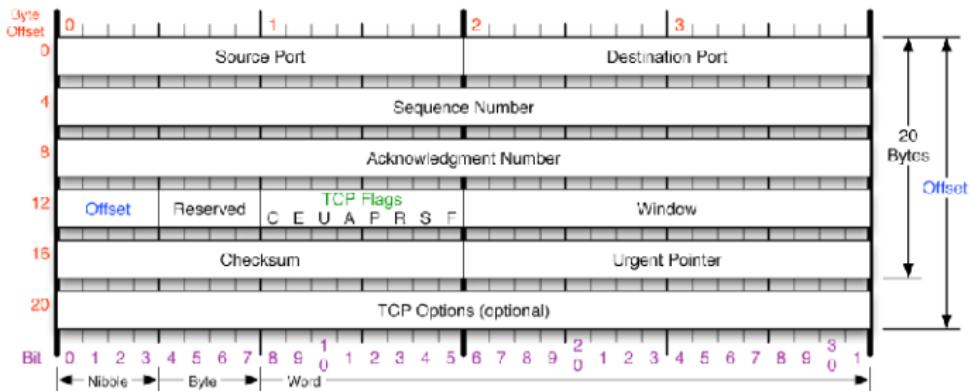
TCP est un service à flux d'octets (*byte stream service*) : si l'application écrit 30 octets en 3 opérations de 10 octets, l'application destinataire ne peut pas connaître la taille des écritures individuelles

## Orienté connexion

Avant que des données puissent être échangées en TCP, les deux extrémités doivent se mettre d'accord (se synchroniser)

- ▶ Les données applicatives sont fractionnées par TCP en fragments de taille adéquate (l'unité de base est le **segment**)
- ▶ Après émission d'un segment, TCP déclenche un *timer* de retransmission et attend un acquittement
- ▶ TCP maintient une somme de contrôle de son entête et de ses données
- ▶ Les fragments TCP sont réordonnés à l'arrivée
- ▶ TCP gère également le contrôle de flux et les paquets dupliqués

## TCP Header

**TCP Flags**

C	E	U	A	P	R	S	F
C 0x80 Reduced (CWR)							
E 0x40 ECN Echo (ECE)							
U 0x20 Urgent							
A 0x10 Ack							
P 0x08 Push							
R 0x04 Reset							
S 0x02 Syn							
F 0x01 Fin							

**Congestion Notification**

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Byn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0

No Congestion	0	0
No Congestion	0	0
Receiver Response	1	0
Sender Response	1	1

**TCP Options**

- 0 End of Options List
- 1 No Operation (NOP, Pad)
- 2 Maximum segment size
- 3 Window Scale
- 4 Selective ACK ok
- 8 Timestamp

**Offset**

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

**RFC 793**

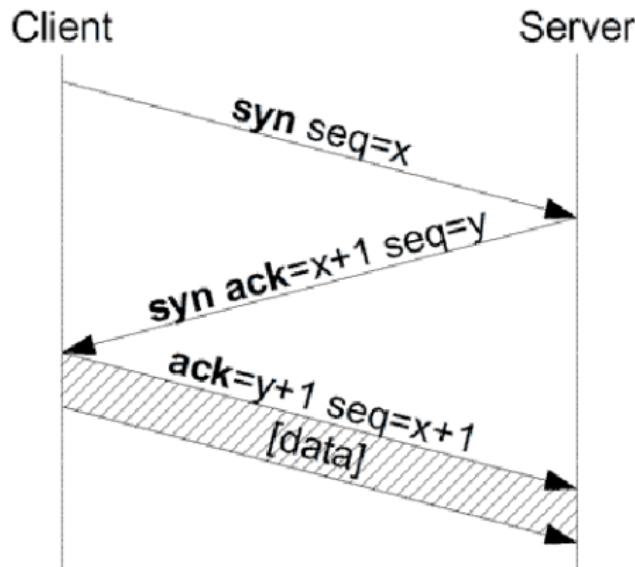
Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

**Checksum**

Checksum of entire TCP segment and pseudo header (parts of IP header)

### Poignée de main en trois étapes

- ▶ Le client émet un segment SYN spécifiant le numéro de port du serveur vers lequel il veut se connecter ainsi que le numéro de séquence initial (NSI)
- ▶ Le serveur répond avec son propre segment SYN contenant également son numéro de séquence initial. Dans le même segment, le serveur acquitte le SYN du client : ACK ( NSI(client) + 1 )
- ▶ Le client acquitte le SYN du serveur : ACK ( NSI(serveur) + 1 )



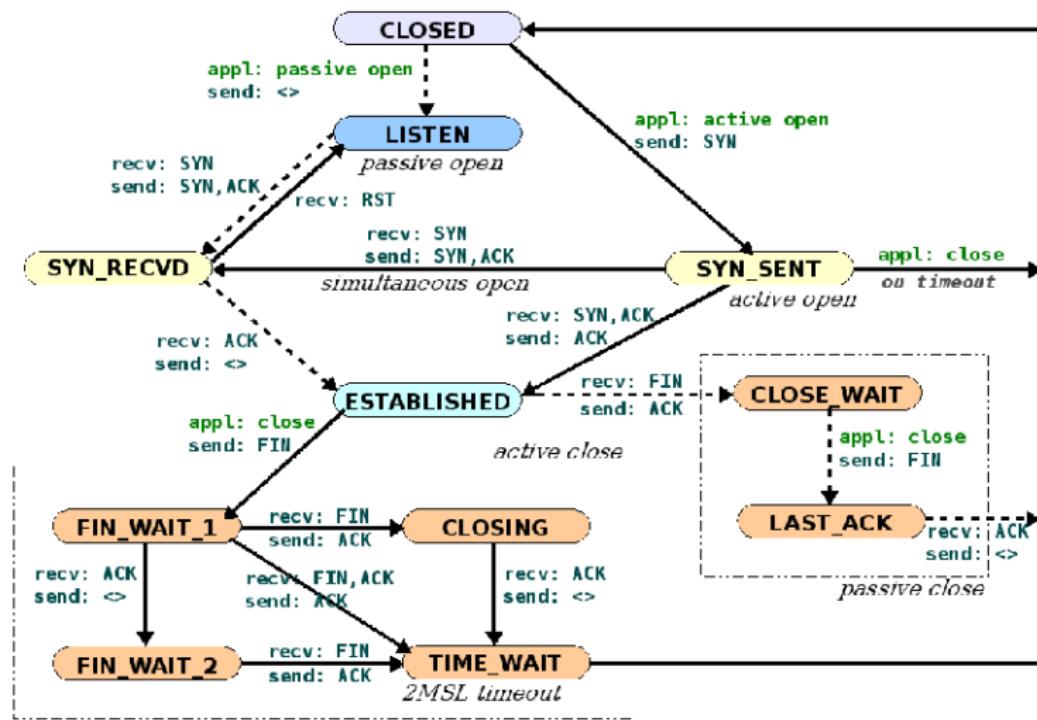
Il faut 4 segments pour fermer une connexion TCP

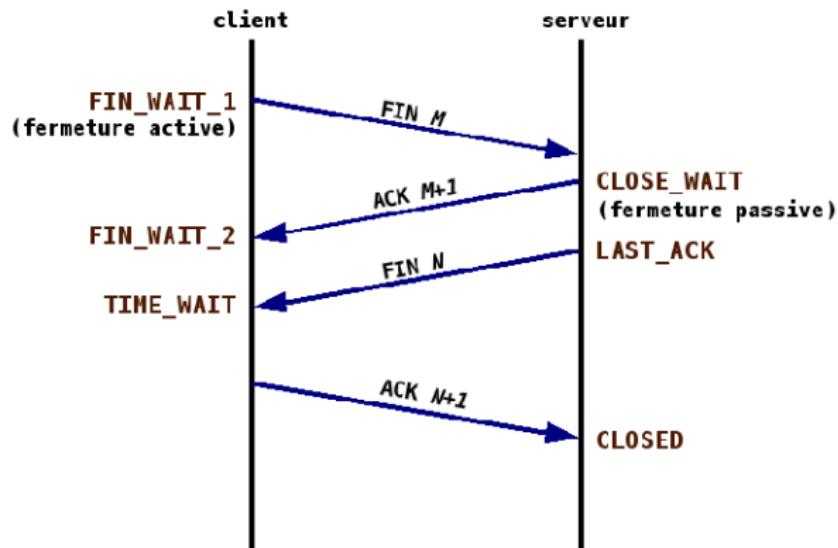
### Rappel

TCP est un protocole *full-duplex* : les données peuvent s'écouler dans les deux sens de façon indépendant

- ▶ N'importe quelle extrémité peut envoyer un FIN lorsqu'elle est en train d'émettre des données. A la réception du FIN, TCP notifie à l'application au dessus que l'autre extrémité a fini d'envoyer des données
- ▶ La réception du FIN signifie **seulement** qu'il n'y a plus de données dans cette direction
- ▶ Notions de fermeture active et fermeture passive

## Diagramme d'états TCP





L'état TIME\_WAIT est aussi appelé **état d'attente 2MSL**

La durée de vie maximale pendant laquelle un segment peut exister sur le réseau est appelée MSL (*Maximum Segment Lifetime*)

- ▶ Limitation supplémentaire par rapport au TTL de la couche IP
- ▶ Valeurs inférieures à 2mns pour la plupart des implémentations

Lorsque qu'il ne reste plus de données à envoyer (après une fermeture active) et après l'envoi du ACK final, TCP doit entrer dans l'état TIME\_WAIT et y rester pendant **2 x MSL**

- ▶ Si le ACK final est perdu, il pourra alors être réémis
- ▶ Durant ce laps de temps, la socket ne peut pas être utilisée

L'algorithme de Nagle (proposé dans la RFC 896) propose une solution simple pour optimiser le trafic réseau résultant de l'émission de paquets de petites tailles (*tinygrams*).

## Une connexion TCP ne peut avoir qu'un seul petit segment non acquitté

- ▶ On considère un segment comme "petit" si sa longueur est inférieure au MSS
- ▶ Cet algorithme peut être dévalidé avec l'option TCP\_NODELAY

- ▶ Timeout d'ouverture de connexion : 6 secs, 24 secs, etc.
- ▶ Taille maximum de segment (MSS)
- ▶ Semi-fermeture de la connexion (*close()* vs. *shutdown()*)
- ▶ Acquittements retardés
- ▶ Taille de fenêtre

Chaque système d'exploitation offre des mécanismes pour envoyer/recevoir des paquets directement au niveau de la couche 2.

- ▶ On utilise généralement le terme « RAW socket » (AF\_PACKET sous Linux).
- ▶ Nécessite des privilèges
- ▶ Pas vraiment standardisé (→ libpcap pour recevoir des paquets)

# Que fait le programme suivant?

```
#!/usr/bin/env python
#
# Source: https://pastebin.com/Pb0qE4jN

import sys
import socket

def main():
    try:
        (port, msg) = (int(sys.argv[1]), ' '.join(sys.argv[2:]))
    except (IndexError, ValueError):
        print("Usage: ./client.py <port> message...")
        sys.exit(1)

    s = socket.socket()
    while True:
        try:
            s.connect(('127.0.0.1', port))
            s.send(msg.encode())
            print(s.recv(256).decode())
            break
        except ConnectionRefusedError:
            pass

    if __name__ == '__main__':
        main()
```

Historiquement, l'**injection de trames** dans une connexion TCP existante par un attaquant *externe* (injection dite « en aveugle ») a toujours été considérée comme difficile car il faut connaître :

- ▶ les adresses IP et les ports source et destination;
- ▶ les numéros de séquence.

Les évolutions des applications modernes basées sur TCP augmentent les risques :

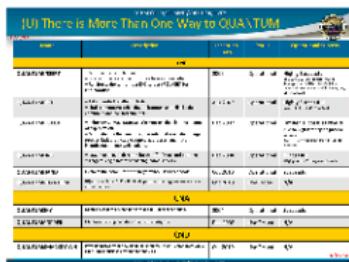
- ▶ connexions persistantes de longue durée;
- ▶ augmentation significative des fenêtres de réception;

L'injection de segments SYN, RST ou de données arbitraires *valides dans la fenêtre d'émission* peut mener à des dénis de services (fermeture de connexions, corruption de données, etc.)

→ Mitigations : « Challenge ACK » (RFC 5961), TCP MD5 (RFC 2385)

L'injection de trafic TCP est réalisable pour un attaquant disposant à la fois de capacités d'écoute passive et d'un moyen d'injecter des paquets.

- ▶ Le succès de cette attaque dépend de la capacité à répondre *plus rapidement* (→ le paquet injecté doit parvenir à la victime *avant* le paquet original)
  - ▶ Les révélations d'E. Snowden en 2013 ont mis en évidence l'utilisation de cette technique par la NSA (→ attaque « QUANTUMINSERT<sup>1</sup> »)
  - ▶ Cette attaque est à la portée de n'importe quel attaquant de **niveau étatique** (mais peut aussi être mise en oeuvre localement : opérateurs, réseaux WiFi, etc.)



1. <https://blog.fox-it.com/2015/04/20/deep-dive-into-quantum-insert/>

Un attaquant peut-il **se faire passer pour quelqu'un d'autre** (*i.e.* prendre une fausse adresse IP) pour établir une connexion TCP?

Pour cela, il doit être capable :

1. d'envoyer des paquets sur le réseau en prenant comme adresse IP source celle du protagoniste pour qui il veut se faire passer (*IP Spoofing*)
2. de deviner le **numéro de séquence initial** qui sera transmis dans la réponse (SYN/ACK) du destinataire (*et qu'il ne recevra pas* → attaque aveugle)
3. de répondre (dernier ACK de la poignée de main à trois temps) *avant* que le destinataire ne reçoive une réponse (paquet RST) de la victime

Vérifier si l'adresse IP source d'un paquet est légitime est un problème complexe :

- ▶ réseaux massivement interconnectés (*multihoming*);
- ▶ règles/politiques de routage dynamiques;
- ▶ présence de routage asymétrique.

La protection contre l'IP spoofing ne peut se faire efficacement que par un filtrage au niveau du *dernier kilomètre*, ou plus généralement au niveau des équipements réseaux situés aux extrémités.

Un ISP peut bloquer le trafic arrivant sur son réseau (par exemple depuis les plages attribuées à ses clients) si ce dernier présente des adresses IP source (ou des prefixes réseaux) inattendus :

- ▶ filtrage statique;
- ▶ Reverse Path Forwarding (RPF) (*Strict/Feasible/Loose*);

Il existe aujourd'hui des préfixes CIDR « non annoncés » sur Internet, provenant le plus souvent d'**assignations historiques**<sup>1</sup>, et pour lesquelles les RIR ne disposent plus de moyens légaux pour les récupérer et les réaffecter.

---

#### 1. Early Registration Transfer (ERX)

## Principe

Les attaques de type « SYN flood » paralysent la pile TCP/IP de la victime en saturant les files d'attentes des connexions TCP à moitié établies.

- ▶ Sous Linux : « SYN Queue » vs. « Accept Queue »
- ▶ Retransmission SYN|ACK : net.ipv4.tcp\_synack\_retries
- ▶ Au niveau applicatif, la taille des files d'attente peut être paramétrée par la valeur du « backlog » (au maximum : net.core.somaxconn)

```
int listen(int sockfd, int backlog);
```

```
/* The backlog argument defines the maximum length to which the queue of
pending connections for sockfd may grow. If a connection request arrives
when the queue is full, the client may receive an error with an indication
of ECONNREFUSED or, if the underlying protocol supports retransmission,
the request may be ignored so that a later reattempt at connection succeeds */
```

Cette attaque est rendue plus efficace :

- ▶ quand elle est distribuée (*DDoS*)  
    (→ difficile à bloquer);
- ▶ en utilisant des adresses IP  
    sources forgées (injoignables).



Mécanismes de protection :

- ▶ augmenter la taille du *backlog* / diminuer net.ipv4.tcp\_synack\_retries;
- ▶ activation des *syncookies*;
- ▶ filtrage (→ ignorer les paquets<sup>1</sup>).

1. <https://blog.cloudflare.com/how-to-drop-10-million-packets/>

Lors de l'établissement d'une connexion TCP, l'extrémité qui répond avec un SYN|ACK doit garder les paramètres de la session dans une file d'attente pour pouvoir valider *plus tard* le ACK final, notamment :

- ▶ le numéro de séquence initial qu'il a envoyé;
- ▶ les options TCP qu'il a reçues avec le SYN.

## Principe

Encoder les paramètres dans le numéro de séquence initial envoyé au client de manière à pouvoir les retrouver à la réception du ACK final.

Le NSI envoyé avec le SYN|ACK est construit avec :

- ▶ un *timestamp* (5 bits)
- ▶ une valeur pour le MSS (3 bits)
- ▶ un *hash* de la socket ( $IP_{src}, Port_{src}, IP_{dst}, Port_{dst}$ ) et du *timestamp* (24 bits)

## Travaux pratiques

- ▶ Retrouver les informations sur la gestion des syncookies dans le noyau Linux (par exemple dans les pages de *man* de *listen(2)* et *tcp(7)*).
  - ▶ Dans la documentation, on trouve notamment la phrase suivante : "*syncokies seriously violate TCP protocol, do not allow to use TCP extensions, can result in serious degradation of some services (f.e. SMTP relaying), visible not by you, but your clients and relays, contacting you*". Quelles explications pouvez-vous donner?
- 
- ▶ Les syncookies ne s'activent que lorsque la file d'attente est saturée
  - ▶ La gestion du MSS envoyé par le client est limitée à 8 valeurs
  - ▶ Pas d'autres options TCP possibles
  - ▶ La perte du ACK final envoyé par le client peut poser des problèmes car il n'y a pas de *timer* de retransmission du SYN | ACK

---

1. <https://lwn.net/Articles/277146/>

Les attaques de type déni de services peuvent être :

- ▶ **directes** : le trafic réseau est envoyé directement depuis l'attaquant (ou *les* attaquants en cas de *DDoS*) vers la cible ;
- ▶ **amplifiées** : le trafic réseau est démultiplié à l'aide d'un ensemble de serveurs vulnérables

*Sockstress* est le nom générique donné à une famille d'attaques ciblant spécifiquement sur les différents composants de la couche TCP.

Il existe peu de mécanismes de protection réellement efficaces (à l'exception du filtrage réseau strict en mode « *white-list* ») car ces attaques se basent sur des caractéristiques inhérentes de la couche TCP.