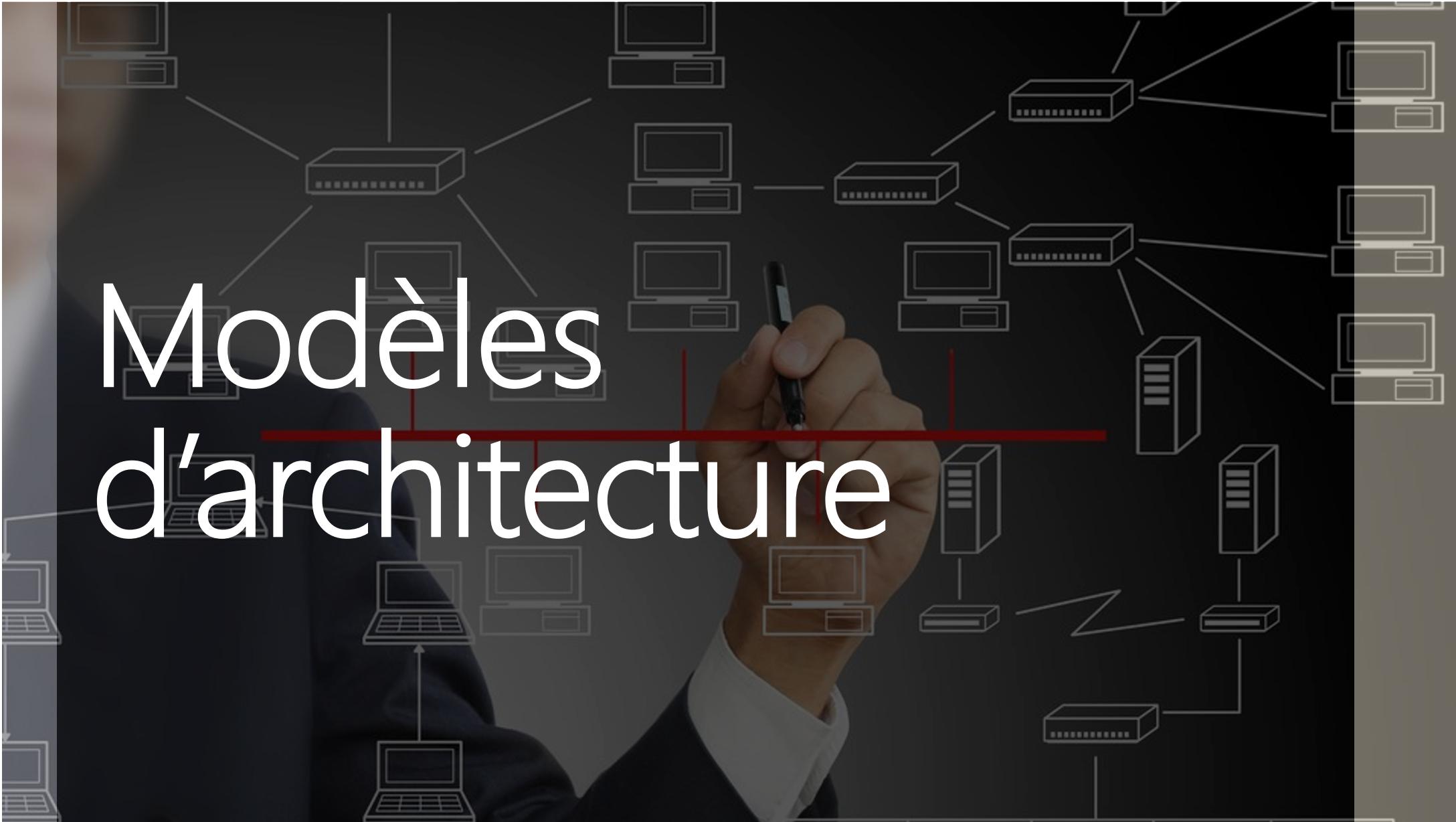


Les fondamentaux du réseau

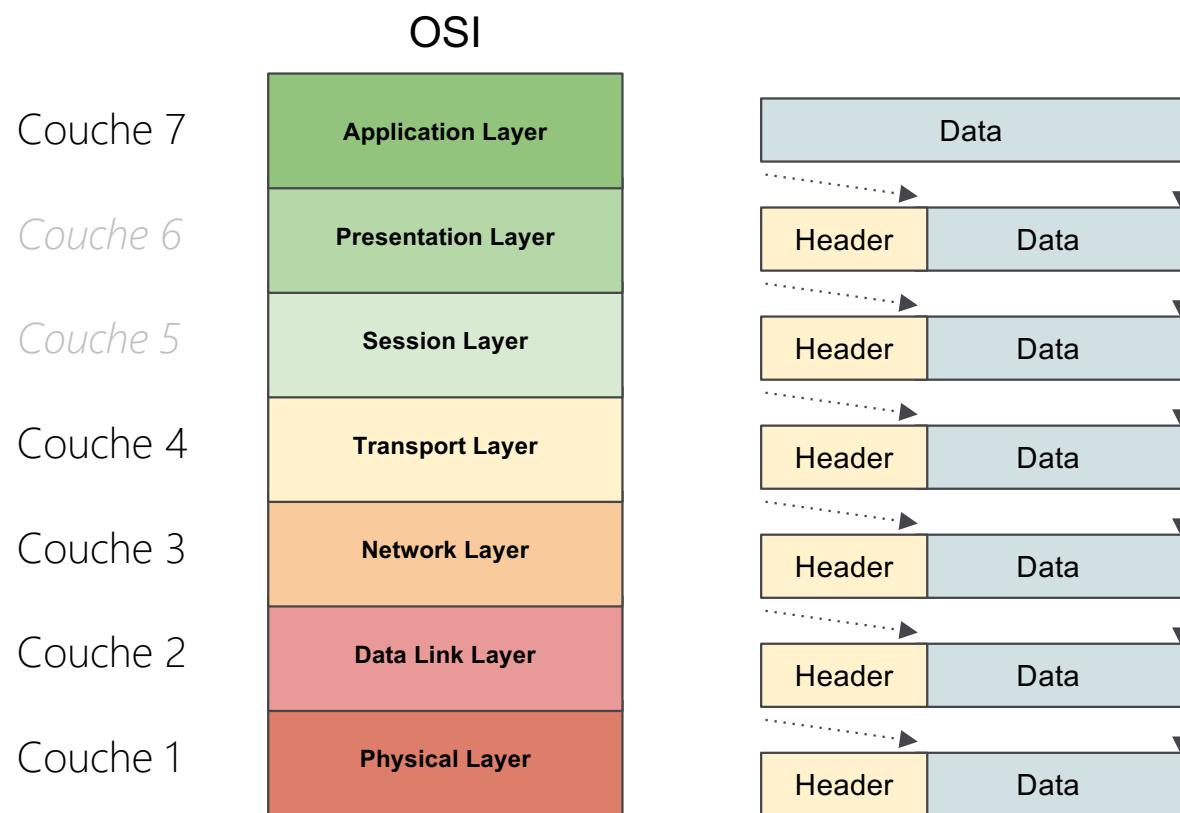
Networking basics



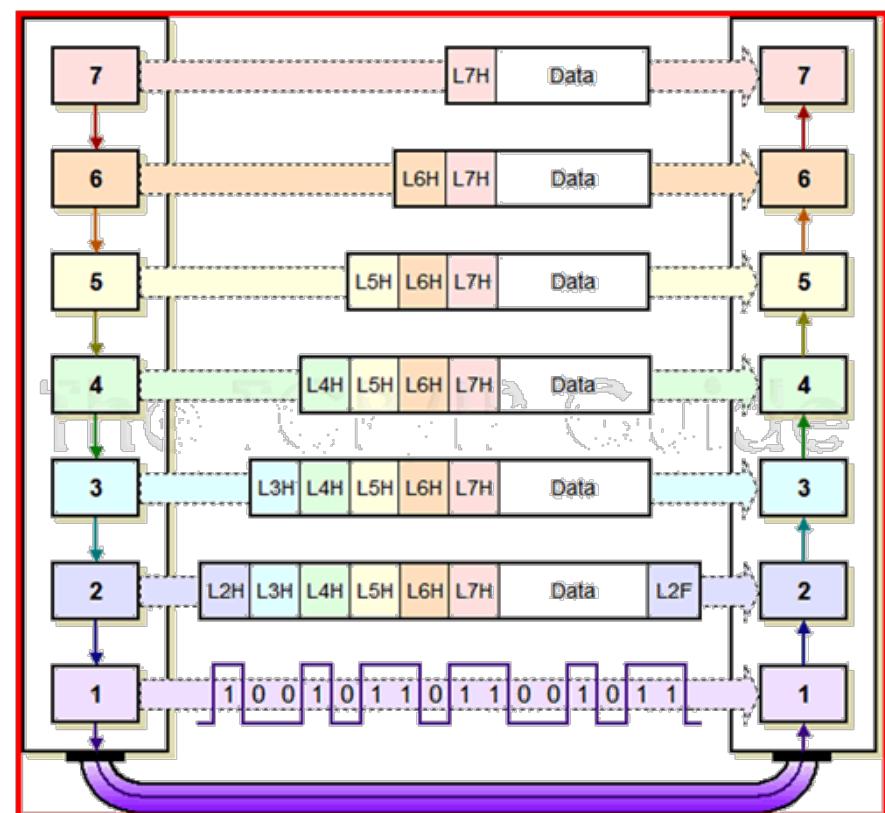
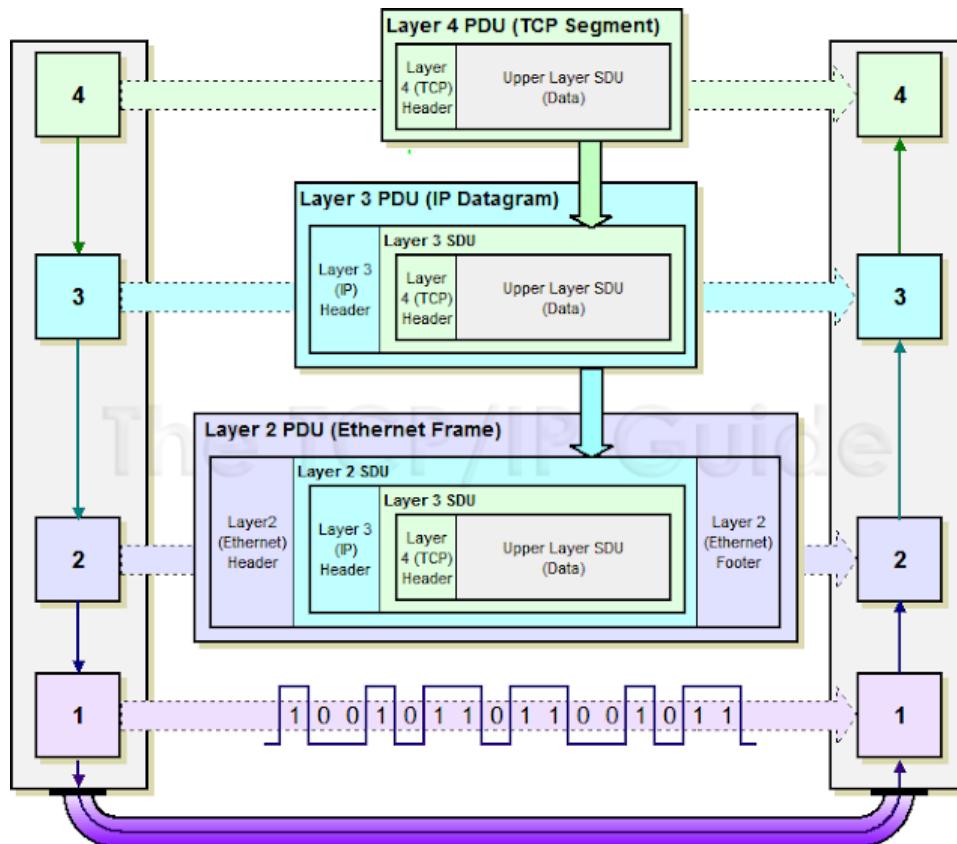
Modèles d'architecture



Modèle OSI (Open Systems Interconnection)

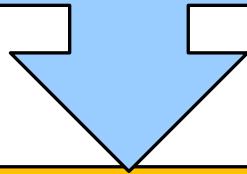


Modèle OSI



Couche 1 : physique

La couche physique s'occupe du transfert effectif de données à travers le réseau.



Types de problématiques : encodage, fréquences, tailles des câbles...

Couche 2 : liaison de données

La couche 2 est responsable de l'établissement d'un réseau local.

Différents types de protocoles, plus ou moins vivants, tous standardisés par le groupe de travail de l'IEEE 802 :

- Ethernet (IEEE 802.3)
- Wi-fi (IEEE 802.11)
- ~~Token Ring (IEEE 802.5)~~
- ...

Tous ces protocoles définissent une adresse physique (aka adresse MAC ou adresse de couche 2) pour identifier une machine au sein du réseau.

Couche 3 : réseau

la couche 3 adresse l'interconnexion des différents réseaux, et donc l'envoi de messages à une machine distante, c'est-à-dire en dehors du réseau local.

Parmi les rôles remplis par la couche 3, on trouve notamment :

- L'adressage logique : une adresse, qui contient en elle l'identifiant du réseau local à laquelle elle appartient.
- Le routage
- La fragmentation des paquets

Couche 4 : transport

La couche 4 permet en particulier de cibler un process sur la machine cible avec lequel on veut parler grâce aux ports TCP et UDP

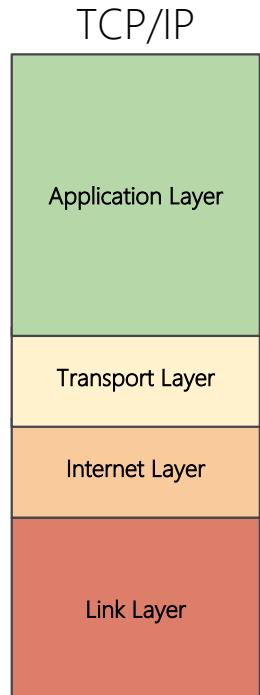
Les autres responsabilités sont quelque peu optionnelles, mais se trouvent dans la couche 4 si on désire les implémenter :

- La couche de transport peut offrir une garantie de bonne réception des messages envoyés à travers un mécanisme d'accusés de réception, et proposer une retransmission des paquets le cas échéant.
- Elle permet aussi l'établissement et la clôture d'une connexion, c'est-à-dire plutôt que de paquets indépendants et autonomes (qu'on appelle alors datagrammes), des paquets faisant partie d'une série de communication plus large, garantissant également leur réception dans le bon ordre.

Couche 7 : application

Dans la couche 7, on retrouve les protocoles tels que HTTP, FTP, DNS qui sont utilisés par les applications pour communiquer sur le réseau.

Modèle TCP/IP



Modèle TCP/IP

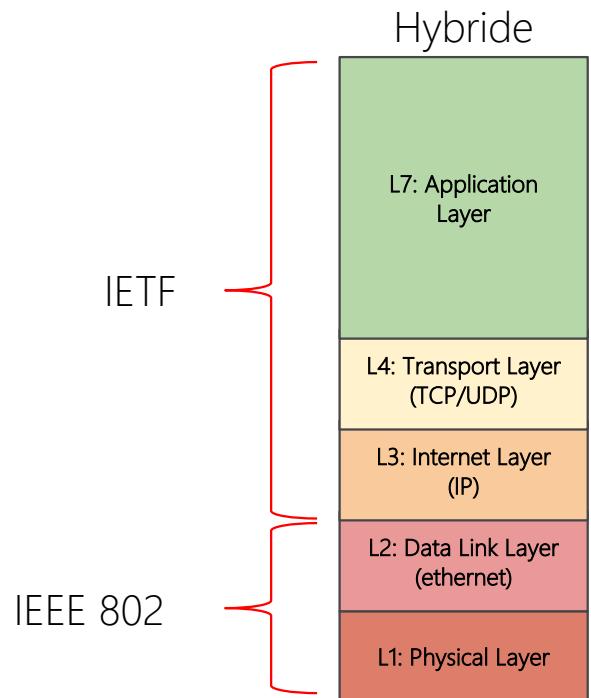
Défini par l'IETF à travers des RFC

Ses couches intermédiaires ont globalement le même rôle que celles d'OSI

Sa problématique : que de l'interconnexion de réseaux locaux déjà définis

À part de quoi faire la glue, aucune spécification des protocoles de couche 2 !

Modèle Hybride



Trame réseau simplifiée

DATA	GET / HTTP/1.1
DEST PORT	80
SRC PORT	37234
DEST IP	192.0.2.42
SRC IP	203.0.113.51
DEST MAC	03:B3:AB:34:52:F5
SRC MAC	A7:4B:D6:B3:4A:6D

Capture Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: http Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
272	13.873755534	192.168.240.206	195.154.165.95	HTTP	150	GET / HTTP/1.1
274	13.876101987	195.154.165.95	192.168.240.206	HTTP	242	HTTP/1.1 404 Not Found (text/plain)

► Frame 272: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0

▼ Ethernet II, Src: LfcHefe_35:3c:6f (e8:6a:64:35:3c:6f), Dst: HewlettP_ff:01:1f (80:c1:6e:ff:01:1f)

 ► Destination: HewlettP_ff:01:1f (80:c1:6e:ff:01:1f)

 ► Source: LfcHefe_35:3c:6f (e8:6a:64:35:3c:6f)

 Type: IPv4 (0x0800)

 ► Internet Protocol Version 4, Src: 192.168.240.206, Dst: 195.154.165.95

 ► Transmission Control Protocol, Src Port: 55162, Dst Port: 80, Seq: 1, Ack: 1, Len: 84

▼ Hypertext Transfer Protocol

 ► GET / HTTP/1.1\r\n

 Host: deliciousmuffins.net\r\n

 User-Agent: curl/7.61.0\r\n

 Accept: */*\r\n

\r\n

[Full request URI: <http://deliciousmuffins.net/>]

[HTTP request 1/1]

[Response in frame: 274]

Hex	Dec	ASCII
0000	80 c1 6e ff 01 1f e8 6a	..n....j d5<o...E.
0010	64 35 3c 6f 08 00 45 02@. 5.....
0020	00 88 ea 2c 40 00 40 06	..z.P..!..
0030	35 d0 c0 a8 f0 ce c3 9aB..#f
0040	a5 5f d7 7a 00 50 c9 0a	TGGET / HTTP/1.1
0050	f6 9a 13 ce e1 21 80 18	.Host: deliciou
0060	00 e5 1a ec 00 00 01 01	smuffins .net..Us
0070	08 0a 03 42 b7 f3 23 66	er-Agent : curl/7
0080	48 54 54 50 2f 31 2e 31	.61.0..A ccept: *
0090	6d 72 2d 41 67 65 6e 74	/*....
00a0	2e 36 31 2e 30 0d 0a 41	
00b0	63 63 65 70 74 3a 20 2a	
00c0	2f 2a 0d 0a 0d 0a	

Une topologie simple



Le switch

- Le switch (ou bridge) est l'appareil dominant pour l'établissement d'un réseau local en Ethernet
- Fonctionnement simple : tous les appareils connectés au switch appartiennent au même réseau local !
- Deux switches connectés entre eux forment un super réseau local

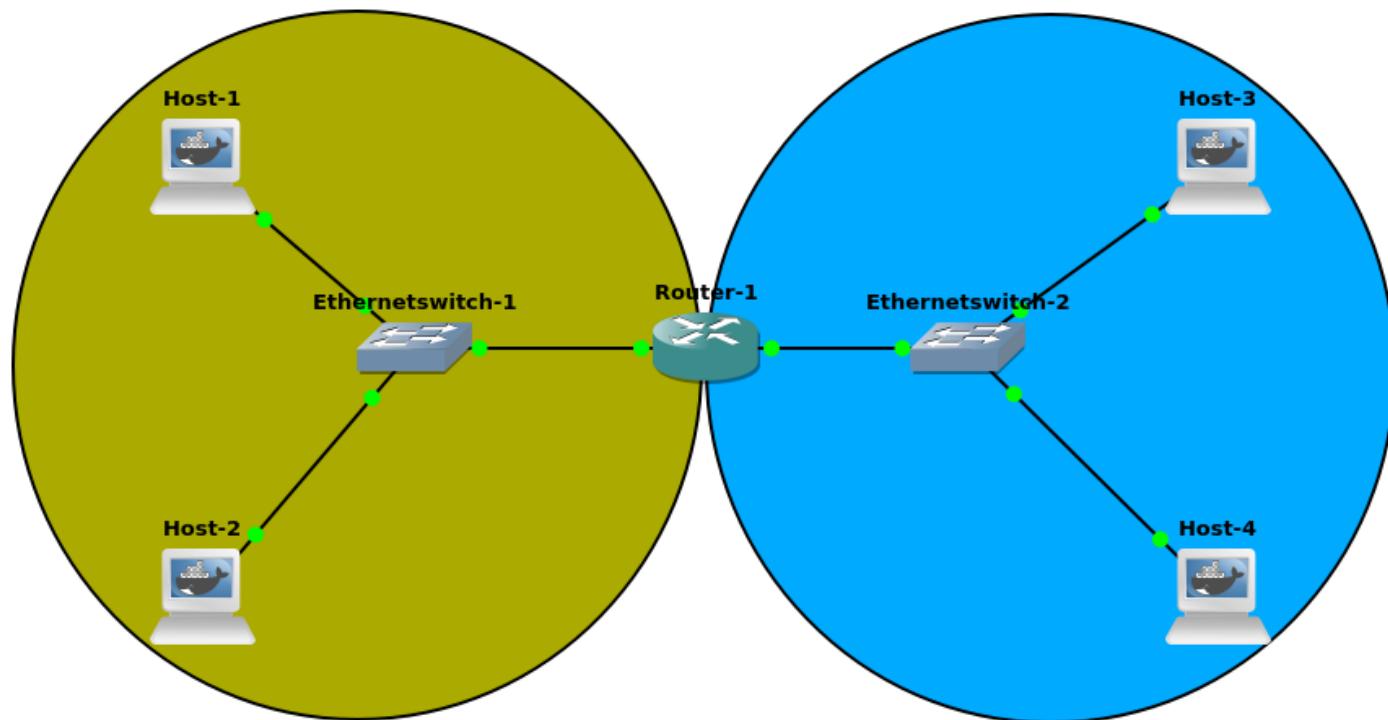


Le routeur

- Le router un appareil de couche 3, servant à interconnecter plusieurs réseaux locaux
- Une patte dans chaque réseau
- Les autres machines **doivent** passer par lui pour sortir du réseau !



Exemple de topologie triviale



Jargon moderne

De nos jours, une entreprise n'a pas qu'un seul réseau local, mais une multitude

La séparation entre couche 2 et couche 3 est de plus en plus floue en terme d'équipements réseau

Aujourd'hui, on appelle "switch" un appareil qui fonctionne au sein des réseaux d'une entité (qui fait du switching + du routage)

On appelle "routeur" un appareil qui permet de s'interconnecter avec un réseau distant géré par une autre entité

Internet Protocol



Le secret d'une adresse IPv4

Une adresse IPv4 est codée sur **32 bits** et est composée de **2 parties** de taille variable

- La première partie est le **préfixe**, qui détermine le **réseau** dans lequel vit la machine
- La deuxième partie est l'**identifiant de la machine** au sein de ce réseau, et est **unique**.
- Le **masque de sous réseau** permet d'identifier la partie réseau et la partie machine de l'adresse IP (les bits à 1 dans le masque **représente la partie réseau**)

Ex : adresse 192.168.1.42 avec le masque de sous réseau 255.255.255.0 donne :

192.	168.	1.	42
11000000.	10101000.	00000001.	00101010 /
11111111.	11111111.	11111111.	00000000
255.	255.	255.	0

CIDR - Classless Inter-Domain Routing

Pour faciliter l'écriture des masques de sous-réseaux, on peut également utiliser la notation CIDR. Les bits à 1 du masque sont ainsi utilisés pour le représenter.

192.168.1.42/255.255.255.0 peut donc s'écrire 192.168.1.42/24

(/24 correspond aux 24 bits à 1 dans le masque de sous réseau 11111111.11111111.11111111.00000000)

CIDR - Classless Inter-Domain Routing

Grâce à la notation CIDR, on peut également calculer très rapidement le **nombre d'adresses machines disponibles** dans un réseau avec la formule suivante :

$$\text{IP dispo} = 2^{\text{(nombre de bits à 0)}} - 2$$

Ex pour un /26 -> IP dispo = $2^6 - 2$

Dans la formule précédente, le "-2" correspond aux **adresses du réseau et de broadcast**, qui elles sont réservées.

Question : Combien a-t-on d'adresses machines avec le masque de sous-réseau /24 ?

Adresse du réseau, adresse de broadcast et adresse machine

Considérons le réseau 192.168.1.0/24

- La première adresse du réseau est l'**adresse du réseau**. Elle permet de le désigner. 192.168.1.0
- La dernière adresse du réseau est l'**adresse de broadcast**. Un message envoyé à l'adresse de broadcast sera reçu par toutes les machines du réseau. 192.168.1.255
- Le reste des adresses sont des **adresses machines** disponibles. Ex: 192.168.1.42

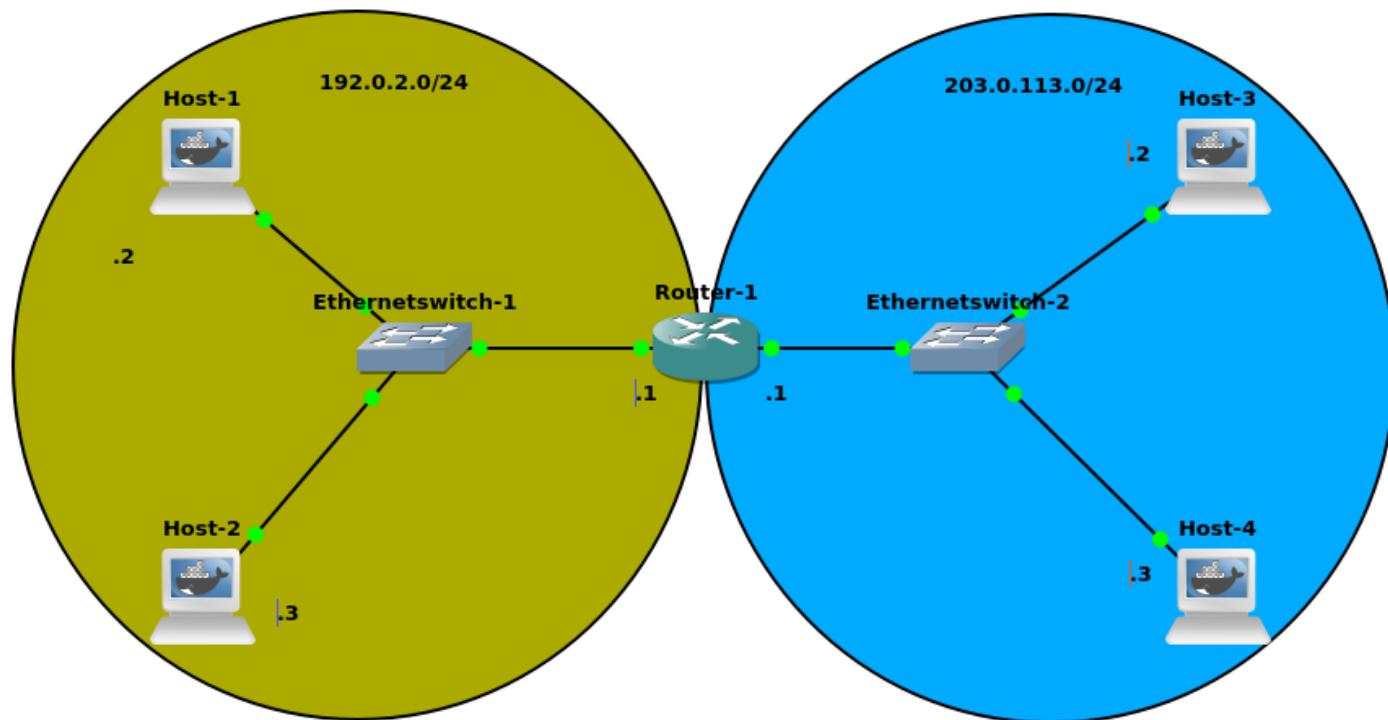
Question 1 : Quelles sont les adresses du réseau et de broadcast de 10.1.0.0/16 ?

Question 2 : Quelle type d'adresse (réseau, broadcast ou machine) correspond à 192.168.1.0/23?

IPCALC

```
$ ipcalc 192.168.1.0/23
Address: 192.168.1.0          11000000.10101000.0000000 1.00000000
Netmask: 255.255.254.0 = 23  11111111.11111111.1111111 0.00000000
Wildcard: 0.0.1.255          00000000.00000000.0000000 1.11111111
=>
Network: 192.168.0.0/23      11000000.10101000.0000000 0.00000000
HostMin: 192.168.0.1          11000000.10101000.0000000 0.00000001
HostMax: 192.168.1.254        11000000.10101000.0000000 1.11111110
Broadcast: 192.168.1.255     11000000.10101000.0000000 1.11111111
Hosts/Net: 510                Class C, Private Internet
```

Le réseau trivial, numéroté



La table de routage

- Lorsqu'une machine envoie (ou repropage si c'est un routeur) un paquet, elle inspecte sa table de routage pour savoir à quelle interface l'envoyer
- On choisit l'entrée de la table la plus spécifique (*longest prefix match*)
- Deux types de routes:
 - Route directe (ou route locale) : le destinataire existe sur un des LANs auxquels la machine est connectée
 - Route distante : le destinataire se situe sur un réseau distant, il faut alors passer par un routeur

La table de routage

```
$ ip route  
default via 192.168.240.254 dev enp0s31f6 proto dhcp  
172.17.0.0/16      dev docker0 proto kernel      scope link      src 172.17.0.1  
192.168.122.0/24    dev virbr0 proto kernel      scope link      src 192.168.122.1  
192.168.185.0/24    dev virbr1 proto kernel      scope link      src 192.168.185.1  
192.168.240.0/24    dev enp0s31f6 proto kernel      scope link      src 192.168.240.206
```

↑
Préfixe

↑
Interface

↑
Route directe

↑
IP source

IP privées

Une IP publique doit être attribuée globalement par l'IANA, interdiction de s'auto-attribuer une IP sans demander la permission !

Que faire pour un réseau privé ?

La RFC 1918 définit trois préfixes réservées à un usage privé:

- 10.0.0.0/8
- 192.168.0.0/16
- 172.16.0.0/12

Ces adresses n'ont pas le droit d'exister sur Internet, aucun risque de conflit !



NAT et firewalling

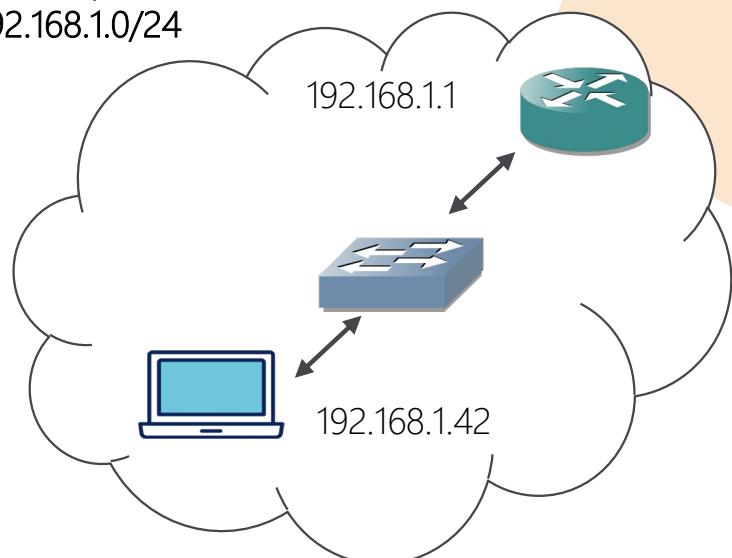
RFC 1918 : le paradoxe

Si ces adresses n'ont pas
le droit d'exister sur
Internet... Comment aller
sur Internet avec ?

Si deux réseaux
choisissent chacun le
même préfixe, comment
peuvent-ils communiquer
sans conflit ?

RFC 1918: le paradoxe

Réseau privé d'Alice
192.168.1.0/24

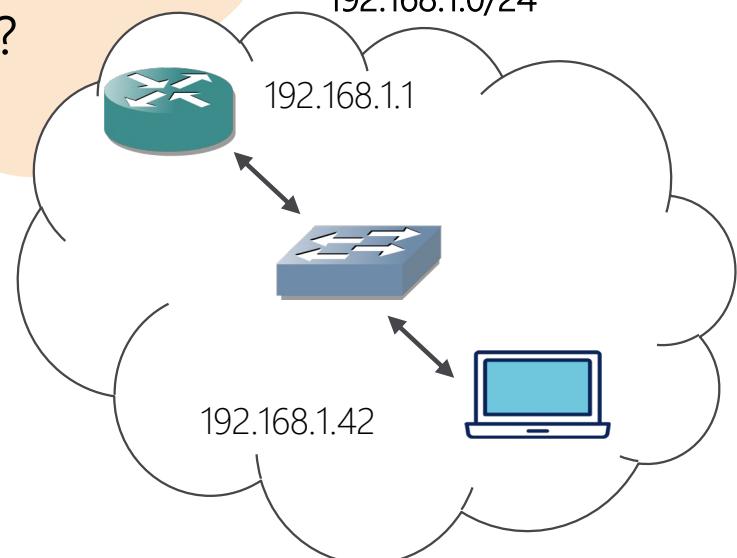


Internet

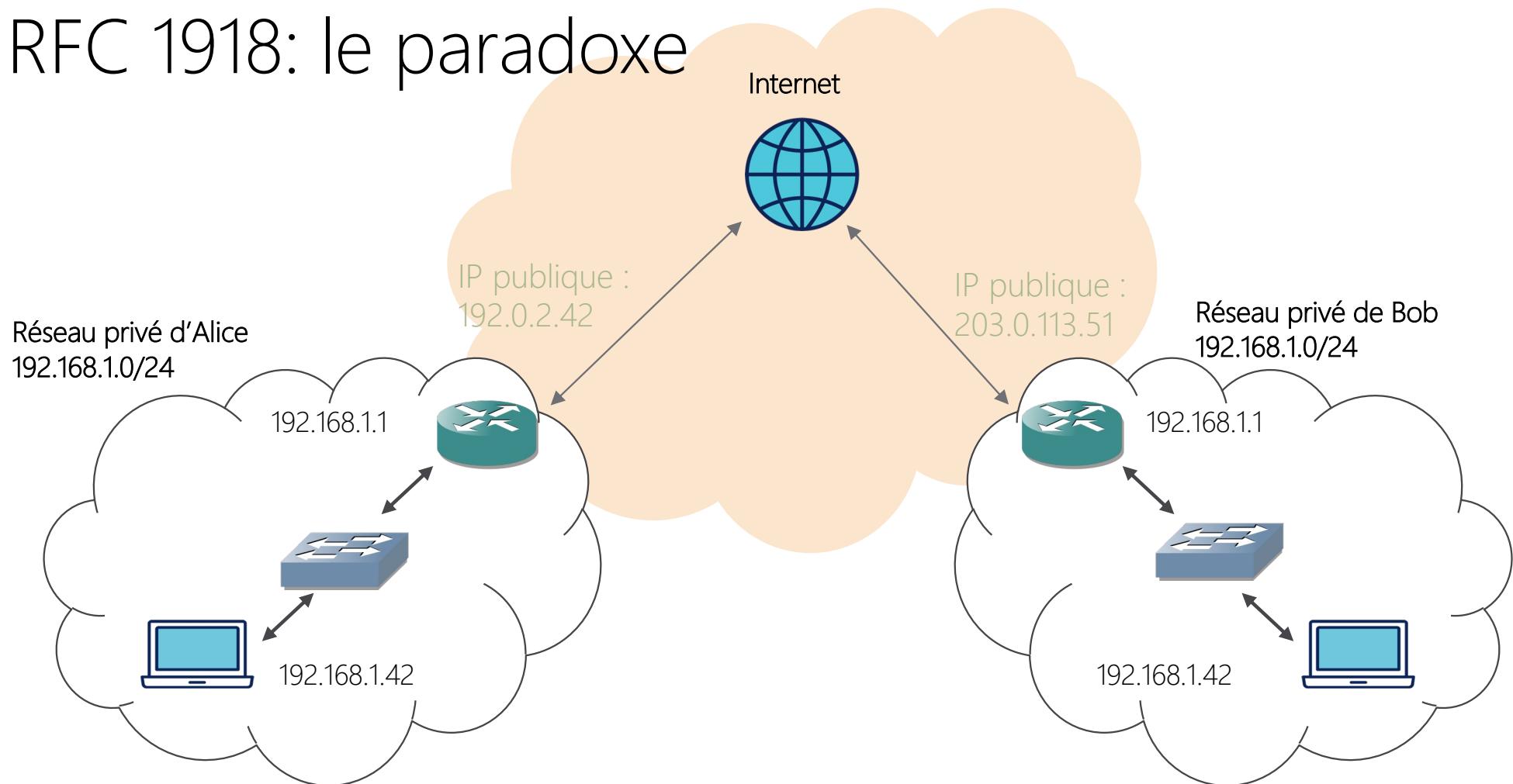


Comment les réseaux
privés d'Alice et Bob
communiquent-ils ?

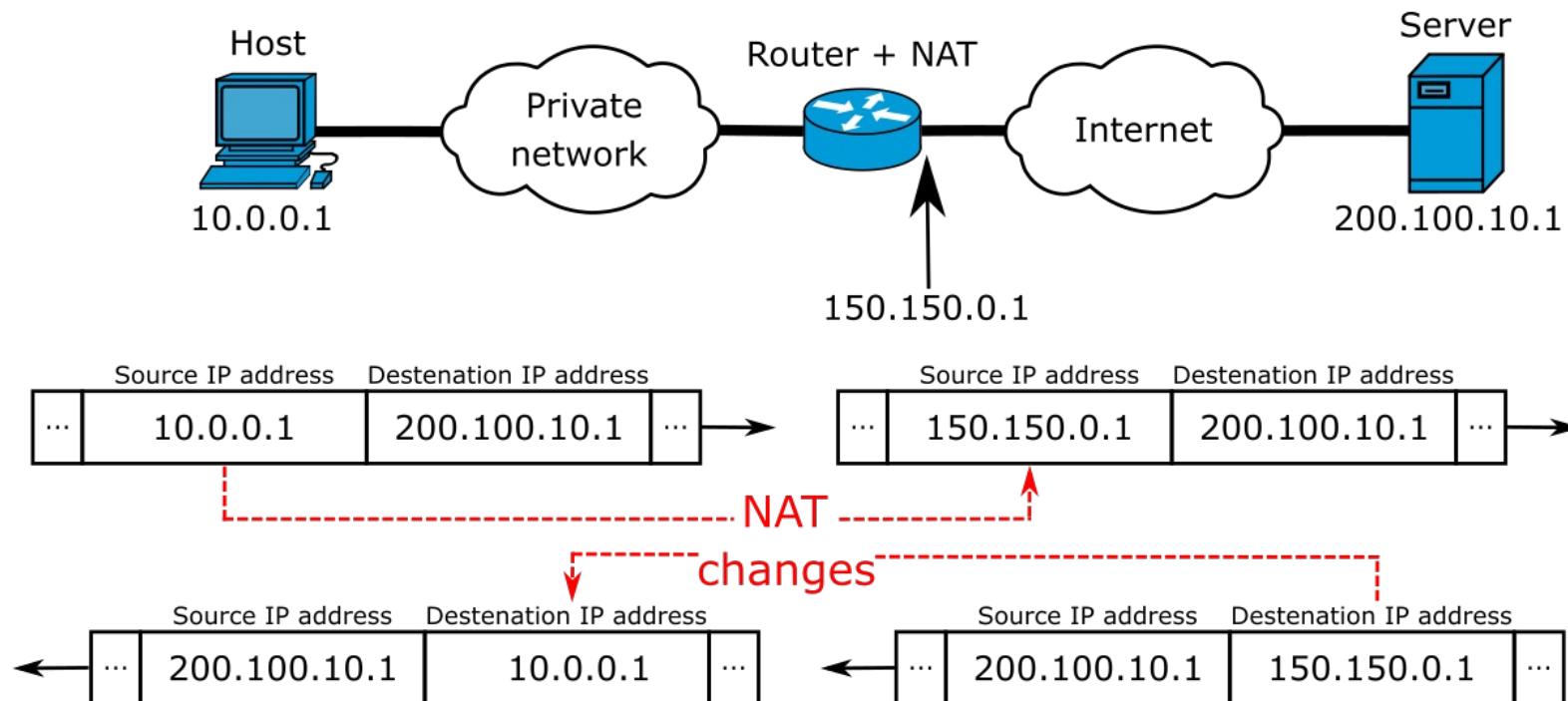
Réseau privé de Bob
192.168.1.0/24



RFC 1918: le paradoxe



Network Address Translation



Network Address Translation

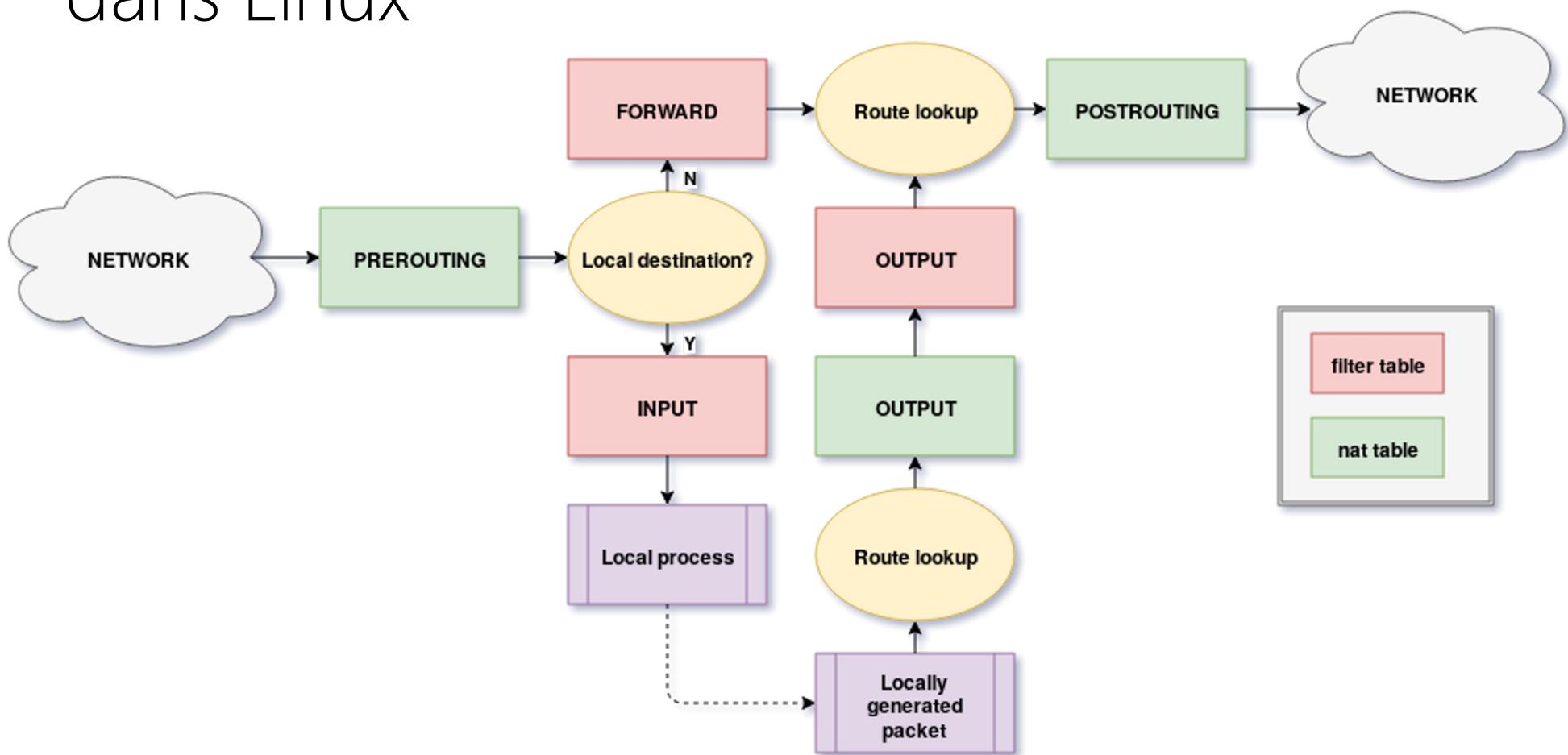
- Lorsqu'un routeur traite un paquet, il peut le modifier avant de l'envoyer, on parle alors de NAT
- On distingue deux types de NAT:
 - SNAT (Source NAT), parfois appelé masquerading
 - DNAT (Destination NAT)
- Les règles de NAT peuvent être modifiées par l'outil `iptables`, en sélectionnant la table `nat`.

Jouer avec les réseaux et les paquets sous Linux, avec iptables

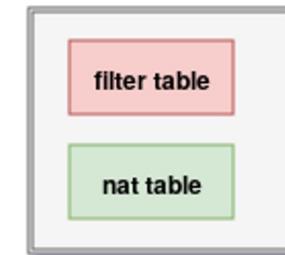
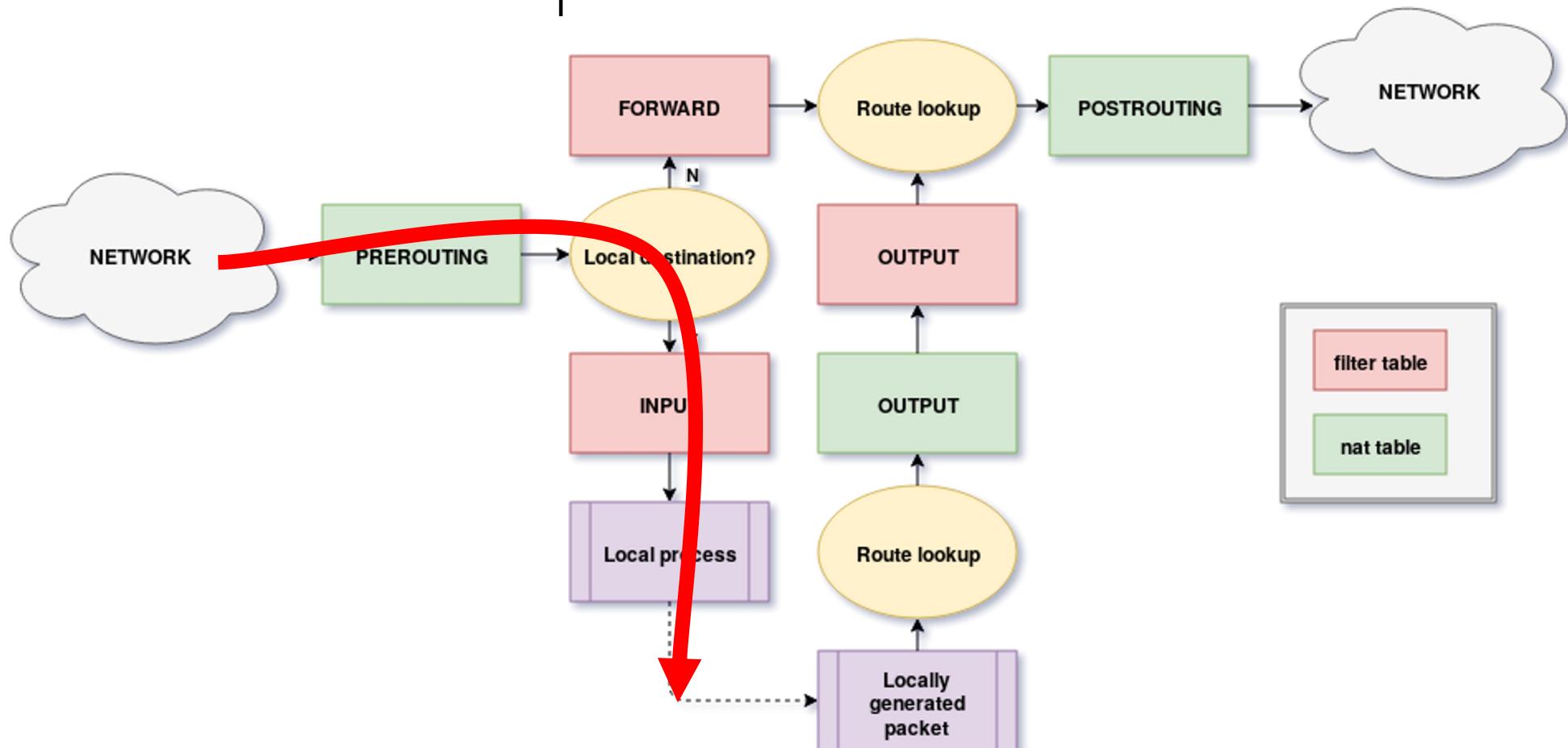
- Filtrer les flux (firewall)
- Modifier les paquets réseau



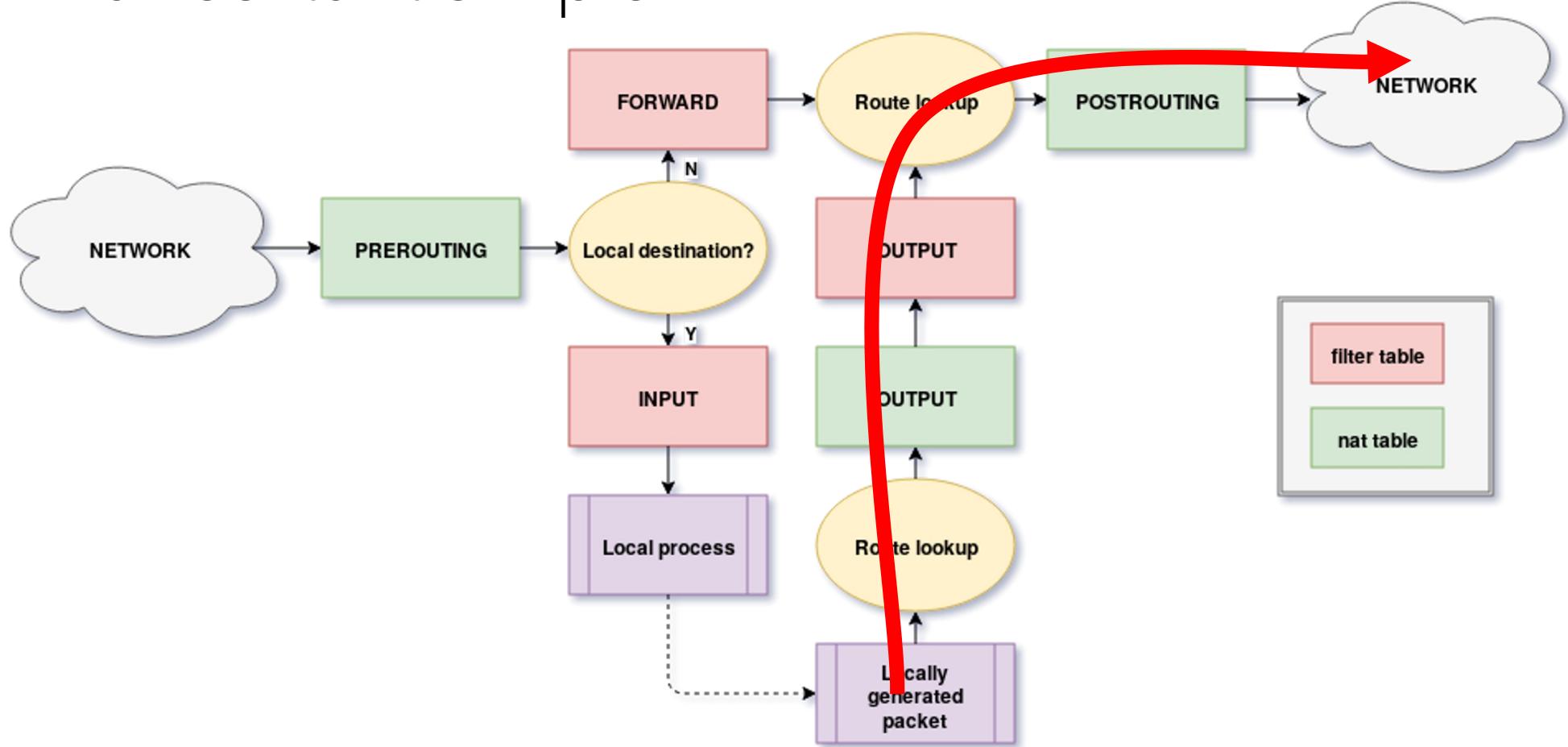
Le cheminement d'un paquet réseau dans Linux



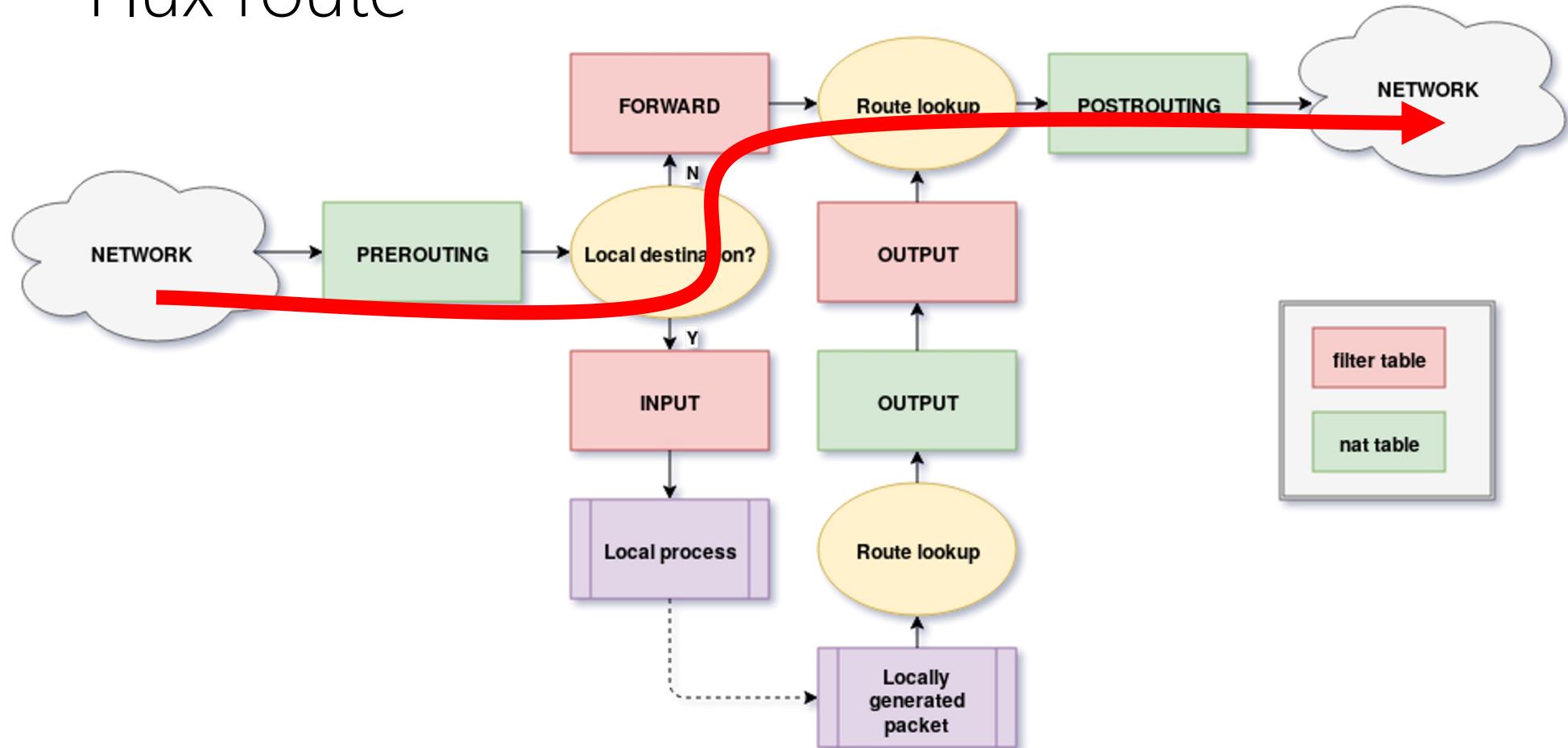
Flux entrant simple



Flux sortant simple



Flux routé



HTTP, DNS, DHCP

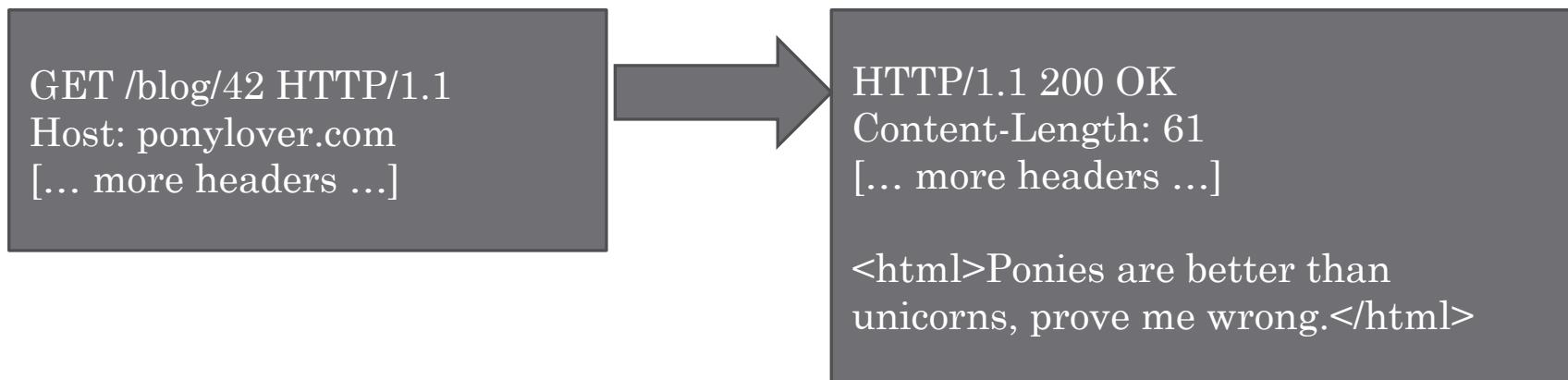
Trois protocoles applicatifs à connaître



HTTP

- La *Lingua Franca* de l'Internet
- Protocole de requête/réponse sur TCP
- Peut renvoyer du HTML, mais pas uniquement !
- Également très utilisé de manière programmatique : API over HTTP
- Il existe plusieurs styles de design d'API sur HTTP, mais le dominant aujourd'hui est REST (REpresentational State Transfer)
- On se sert des **verbes** HTTP (GET, POST, PUT, DELETE...) pour décrire une action sur une **ressource**

Exemple de requête (1)



Exemple de requête (2)

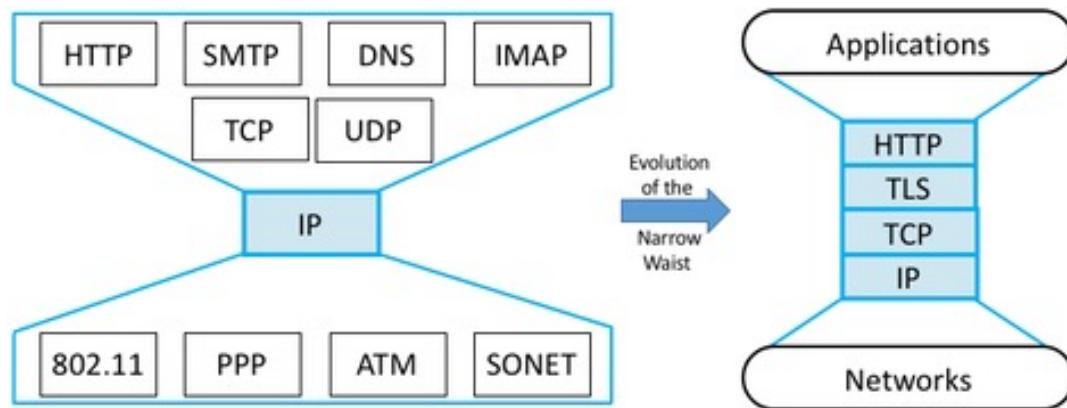
```
POST /blog/ HTTP/1.1  
Host: ponylover.com  
Content-Length: 45  
[... more headers ...]
```

```
<html>Why unicorns are indeed  
superior</html>
```



```
HTTP/1.1 201 Created  
Location http://ponylover.com/blog/51  
[... more headers ...]
```

HTTP is the new narrow waist

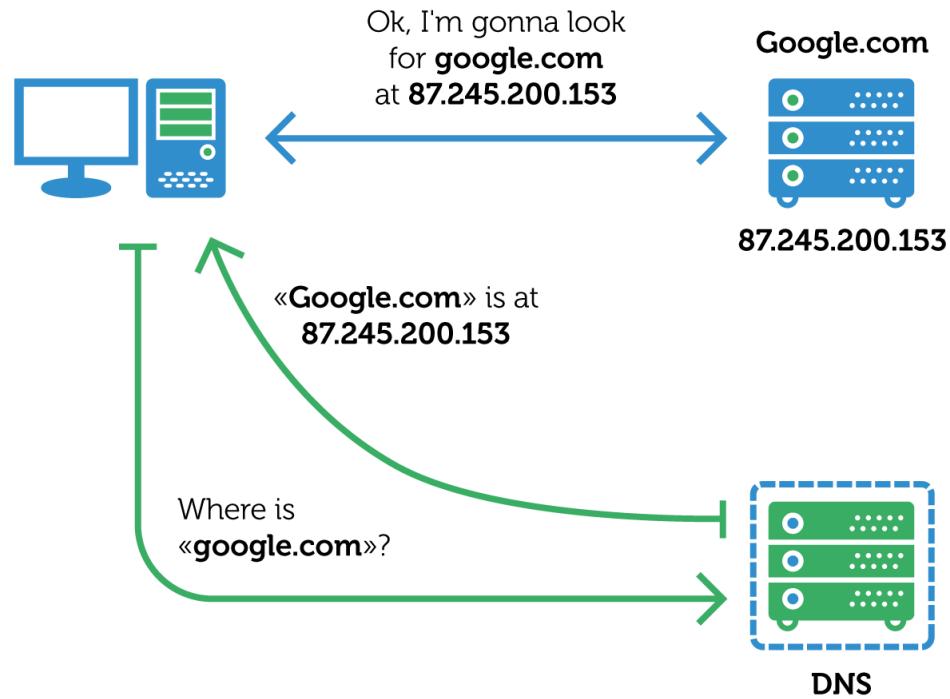


Source: <https://www.systemsapproach.org/blog/http-is-the-new-narrow-waist>

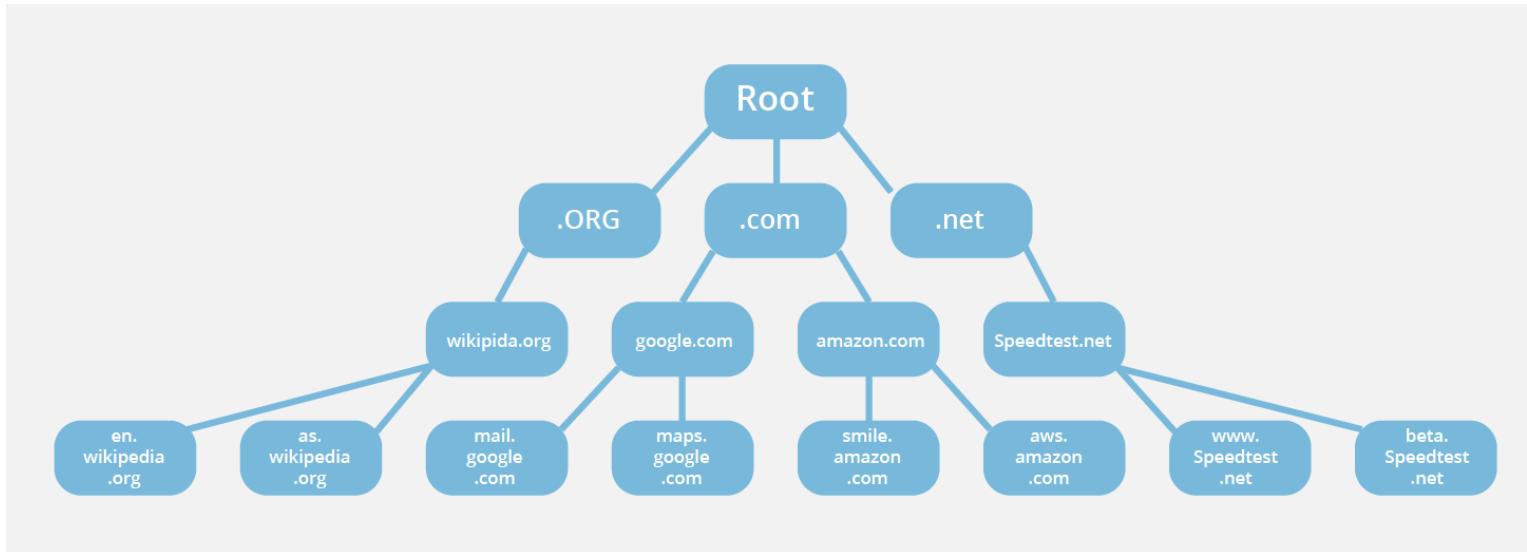
DNS

- Domain Name System
- Traduction des noms de domaines (e.g. www.example.com) en adresses IP
- Fonctionnement décentralisé :
 - Les domaines sont hiérarchiques : "www" appartient à "example" qui appartient à "com". Chaque sous domaine peut être délégué à une autre entité
 - Un serveur DNS peut avoir autorité sur un, plusieurs, ou aucun domaine
 - En haut de la hiérarchie, les "root servers" ont délégué les TLD (top level domains) comme "com" et "fr", et ainsi de suite.
- Tous les noms d'un domaine sont recensés au sein d'un fichier de zone, contrôlé par le serveur ayant autorité sur ce domaine

DNS



Exemple honteusement volé de cloudflare



DNS : Les resource records

- Un nom de domaine peut être associé à une adresse IP, mais pas forcément !
- Une liste non exhaustive des RR communs :
 - "A" : une adresse IPv4
 - "AAAA" : une adresse IPv6
 - "CNAME" : un alias vers un autre nom (canonical name)
 - "NS" : délégation d'un nom à une autre entité
 - "SOA" : Start of Authority, RR de metadata obligatoire au début d'un fichier de zone

Exemple de fichier de zone

```
1 $ORIGIN deliciousmuffins.net.-
2 @ IN SOA ns1.deliciousmuffins.net. root.deliciousmuffins.net. (~
3 2015060904 ; Serial-
4 3600 ; Refresh-
5 900 ; Retry-
6 86400 ; Expire-
7 600 ) ; Negative Cache-
8 -
9 @ IN NS ns1-
10 @ IN NS ns2-
11 -
12 ; Mail stuff via gandi-
13 pop 10800 IN CNAME access.mail.gandi.net.-
14 webmail 10800 IN CNAME agent.mail.gandi.net.-
15 smtp 10800 IN CNAME relay.mail.gandi.net.-
16 imap 10800 IN CNAME access.mail.gandi.net.-
17 @ 10800 IN MX 10 spool.mail.gandi.net.-
18 @ 10800 IN MX 50 fb.mail.gandi.net.-
19 -
20 @ IN A 195.154.165.95-
21 IN AAAA 2001:bc8:3101:101::3-
22 -
23 * IN A 195.154.165.95-
24 IN AAAA 2001:bc8:3101:101::3-
~
```

It's not DNS



There's no way it's DNS

It was DNS



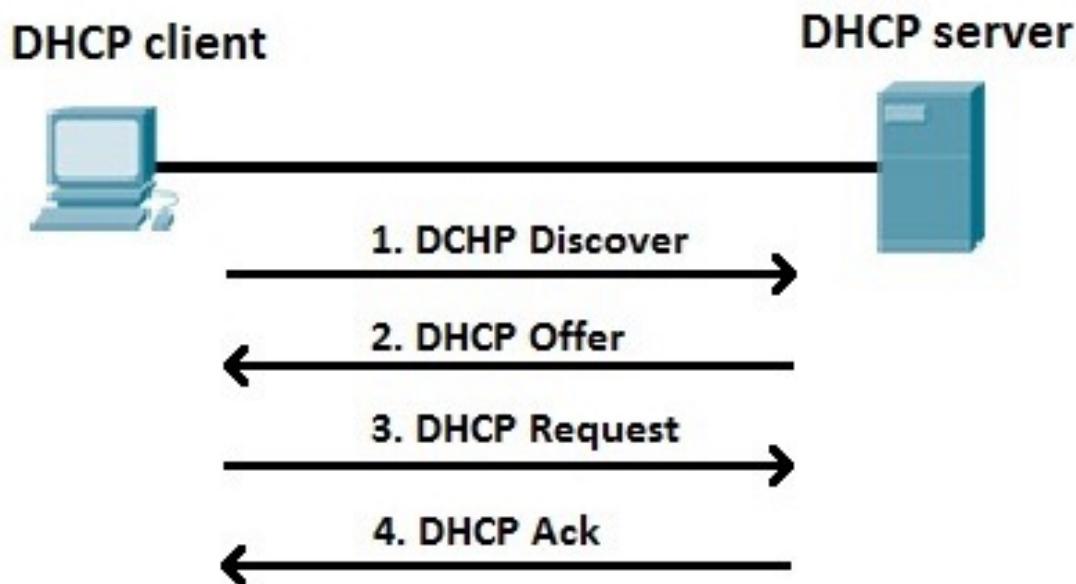
山家
花をあわせ
とあわよ
う



DHCP

- Protocole d'autoconfiguration des machines, pour éviter les dépressions nerveuses
- Fournit à la machine son adresse IP, mais pas que !
 - Taille du préfixe
 - Adresse du routeur
 - Routes statiques supplémentaires
 - Serveur DNS
 - Serveur NTP
 - ...
- Fonctionnement très simple : on crie très fort sur le réseau qu'on a besoin d'être configuré, et le serveur DHCP répond avec ce qu'il faut

DHCP



TLS et PKI

Un peu de sécurité dans ce monde de brutes

Le CIA de la sécurité

Confidentiality : un message ne doit pas être compris par un tiers

Integrity : un message ne peut pas être modifié/invalidé par un tiers

Authenticity : Je veux être sûr d'être en train de parler au bon destinataire

Chiffrement symétrique

Les participants se mettent d'accord sur une clé de chiffrement partagée au préalable (PSK : Pre-Shared Key)

Exemples d'algorithmes : César, Vigenère... et surtout AES

Problème : pour établir une communication sécurisée, je dois communiquer une clé... de manière sécurisée !

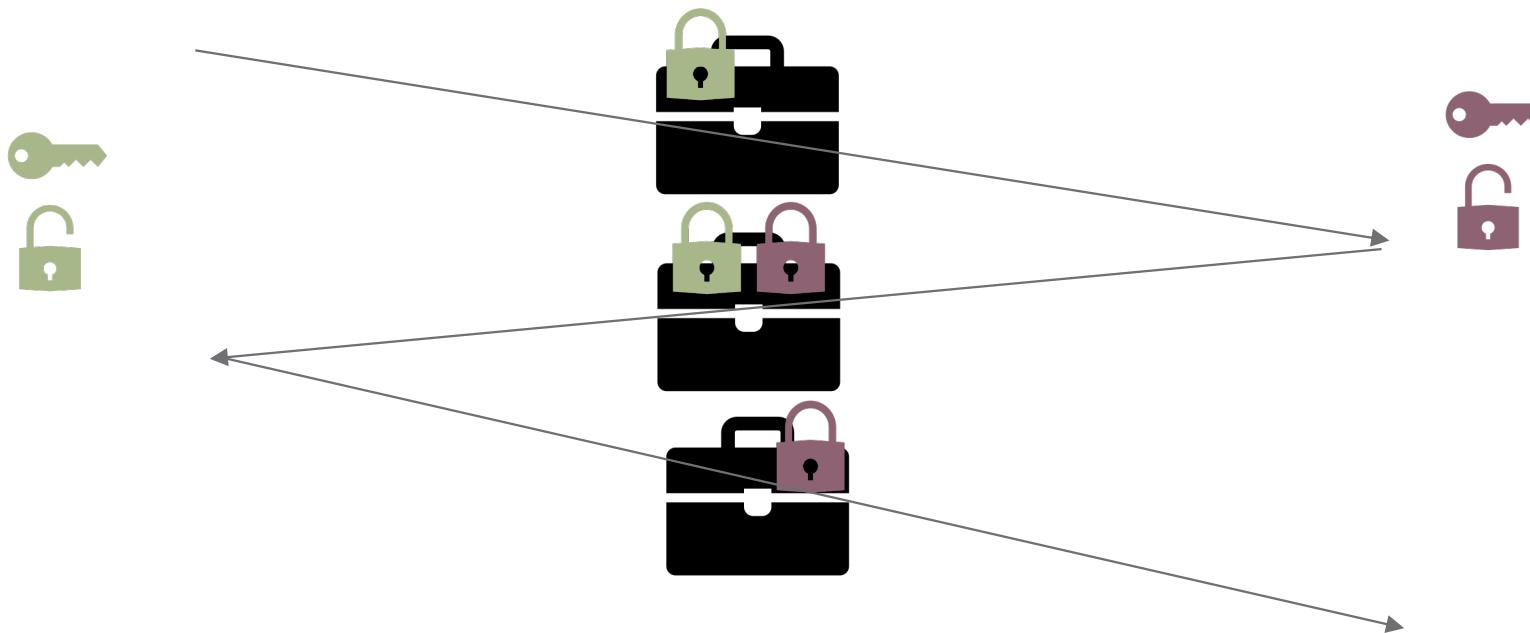
Chiffrement asymétrique

Chaque participant possède deux clés,
qui sont complémentaires

Si on chiffre avec l'une, seule l'autre
peut déchiffrer, et inversement !

Comment ce type de chiffrement peut
résoudre notre problème ?

Diffie-Hellman



À qui appartient la clé ?

Aucune garantie que la clé n'appartient à la bonne personne

Si ça se trouve, j'ai établi une communication sécurisée avec mon attaquant depuis le début

Il faut pouvoir lier une clé à une entité, et avoir la preuve de cette liaison

Le certificat x509

Savant mélange d'une clé publique et
d'une identité (par exemple, un site web)

Le tout signé cryptographiquement et
inforgeable

... mais qui signe le certificat ?

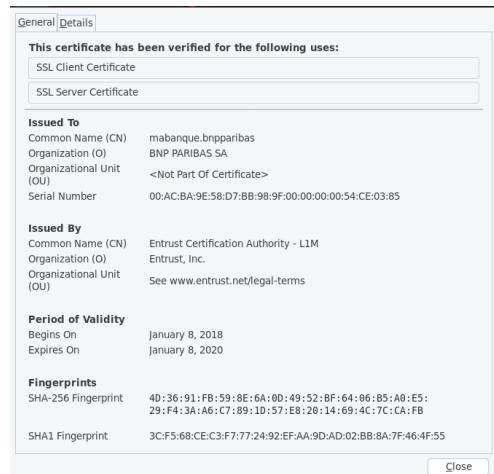
L'autorité de certification

Une entité tierce considérée de confiance par tous les participants

Je fais confiance à mon destinataire parce que mon CA l'a validé

Les CAs de confiance sont déjà installés sur votre poste !

Exemple de certificat x509



Le monde des serveurs web

Nginx : votre nouveau compagnon de jeu

Nginx est le serveur HTTP le plus populaire aujourd'hui, devant le vénérable Apache2

Capable de faire de l'hébergement statique, du load balancing (LB) L7 et du reverse proxying (RP)

Ce n'est **pas** un serveur applicatif !

Exercise 1 : installation

- Pour commencer simplement, installons nginx
- => **sudo apt-get install nginx**
- Le serveur doit ensuite être démarré via **systemd**
- => **sudo systemctl start nginx (ou sudo service nginx start)**
- Les fichiers de configurations se trouvent dans **/etc/nginx**
- Vous pouvez redémarrer le serveur à tout moment :
- => **sudo systemctl restart nginx (ou sudo service nginx restart)**
- Vérifiez la bonne installation sur **http://localhost:80**

Les serveurs statiques

- Un serveur statique a un fonctionnement très simple : il pointe sur un dossier et sert le contenu de ce dossier
- Par défaut, nginx pointe vers **/var/www/html/**
- Modifiez le contenu à l'intérieur pour confirmer
- Nginx peut faire tourner plusieurs "serveurs" virtuels
- Copiez le contenu de **/etc/nginx/sites-enabled/default** et modifiez le port et la racine (retirez l'option **default_server**)
- Vous pouvez désormais avoir plusieurs sites statiques !

HTTP, HTTPS

HTTPS, c'est quand même mieux que HTTP, mais obtenir un certificat est un processus long...

Plutôt que de générer soi-même un certificat auto-signé, on peut en trouver un tout fait dans le paquet **ssl-cert**

On parle communément de "snake oil certificates"

À ne surtout pas utiliser en production !

Exercice 2 : HTTPS

- Installez le paquet **ssl-cert**
- Configurez nginx pour HTTPS en vous inspirant de la configuration commentée par défaut
- Que se passe-t-il lorsque vous accédez à <https://localhost> ?

Les vhosts

- Que faire si l'on souhaite avoir des sites différents derrière le même port ?
- Le nom de domaine a été résolu en adresse IP par le client avant d'envoyer la requête...
- ... Mais HTTP 1.1 impose de préserver le nom de domaine original dans la requête !
- On peut se servir de cette information pour distinguer les différents sites derrière une même IP.
- ... Mais configurer un serveur DNS pour le dev local, c'est quand même lourd...

nip.io à la rescouisse !

nip.io est un faux DNS qui renvoie une réponse à la demande pour simuler un wildcard DNS

Toto.127.0.0.1.nip.io et tata.127.0.0.1.nip.io résolvent tous les deux vers 127.0.0.1.

Extrêmement utile pour le développement local !

Exercice 3 : utilisation des vhosts

- À l'aide de xip.io, configurez deux sites différents résolvant sur votre adresse locale
- Cette fois-ci, au lieu de modifier le port d'écoute, il faut modifier le champ **server_name**.
- N'oubliez pas de redémarrer le serveur après modification !

Les serveurs applicatifs

Un serveur peut executer du code arbitraire et retourner ce qu'il souhaite, tant qu'il parle HTTP.

Le langage de programmation n'est pas important, tant qu'il parle HTTP !@

On ne parle plus vraiment de serveur, mais d'application web.

Exemple en Python

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route("/")
6 def index():
7     return "Meow-lo world!"
8
9 @app.route("/<int:n>")
10 def kittycounter(n):
11     return str(n * 2)
```

Le reverse proxying

- Un serveur applicatif n'écoute généralement pas directement sur l'extérieur, mais se cache derrière un premier serveur : le reverse proxy
- Le RP s'occupe de la paperasse administrative telle que la SSL termination et la centralisation des logs d'accès
- Comme par hasard, nginx est un excellent reverse proxy !
- Plutôt que de renvoyer du contenu statique, nous allons demander à nginx de passer la main à un serveur derrière lui.

Exemple de configuration

```
1 server {  
2     listen 80 default_server;  
3     listen [::]:80 default_server;  
4  
5     server_name _;  
6  
7     location / {  
8         proxy_pass http://127.0.0.1:8000;  
9     }  
10 }
```

Exercice 4 : Reverse Proxy

- Simulez un serveur applicatif à l'aide de **python -m SimpleHTTPServer**
- Sur un nouveau vhost, passez la main sur ce serveur
- Que se passe-t-il si le serveur applicatif s'arrête ?

Load balancing

Plutôt que de prendre directement une URL, **proxy_pass** peut également pointer vers un bloc **upstream**

Les différentes URLs présentes dans le bloc **upstream** seront load balancées entre elles par nginx

D'autres configurations sont possibles :
http://nginx.org/en/docs/http/ngx_http_upstream_module.html

Exemple d'upstream

```
1 upstream backend {  
2   http://127.0.0.1:8000;  
3   http://127.0.0.1:4242;  
4 }  
5 server {  
6   listen 80 default_server;  
7   listen [::]:80 default_server;  
8  
9   server_name _;  
10  
11  location / {  
12    proxy_pass http://backend;  
13  }  
14 }
```

Exercice 5 : jouer sur les locations

- Pour l'instant, chaque serveur ne fait qu'une seule chose, concevons un scénario plus réaliste !
- Créez un serveur qui renvoie un contenu statique lorsque **/static** est appelé
- Pour tous les autres chemins, renvoyez vers le serveur applicatif.
- Indice : il vous faudra plusieurs blocs **location**

It's dangerous to go alone! Take this.

- **Iproute2** : administration réseau quotidienne, remplace tous les outils oldschool comme `ifconfig` et `route`. Cf `man 8 ip`.
-> `ip a / ip r`
- **Netstat/ss** : affiche pas mal de choses, mais en pratique est utilisé pour lister les ports actuellement en écoute, et par quels processus
- **tcpdump/wireshark**: permet de capturer les paquets réseaux qui transitent sur votre machine, et analyser en live le fonctionnement d'un protocole
- **Ipcalc** : petit utilitaire pour calculer rapidement les adresses disponibles au sein d'un bloc
- **dig/nslookup** : pour faire des requêtes DNS
- **Netcat** : ouvrir et manipuler des sockets TCP/UDP à la main

