

TCP/IP, les protocoles qui font fonctionner l'Internet

Jean-Yves Le Boudec

Février 2023

Objectif

Comprendre les principes de TCP/IP

Connaître les outils simples d'exploration et de débogage

Manuels

Kurose and Ross, Computer Networking, a top down approach (8th edition)

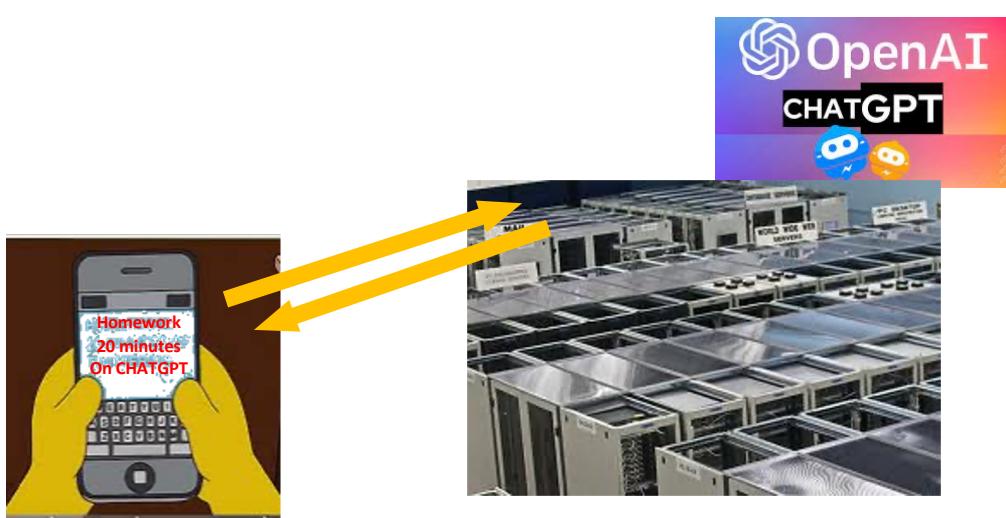
Olivier Bonaventure, Computer Networking : Principles,Protocols and Practice, open book, <https://www.computer-networking.info/>

Table des matières

1. Le modèle en couches
2. La couche réseau et la couche MAC
3. Noms et adresses
4. IPv6
5. Les NATs
6. La couche physique
7. La couche transport
8. Le transport sécurisé
9. La couche application

1. Le Modèle en couches

Qu'est ce que l'Internet ?



Des Clients (smartphones, PCs) et des Serveurs (data centers)

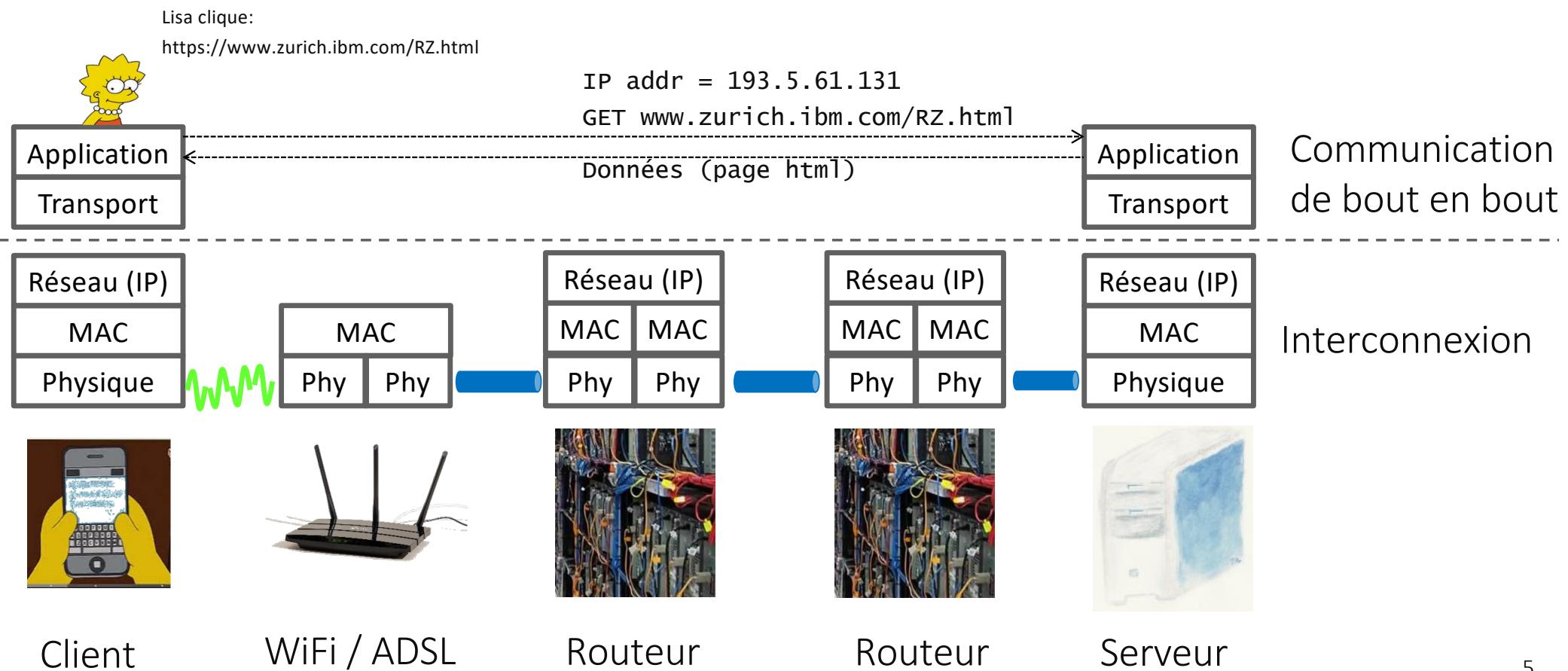
Serveur: un ordinateur qui attend des requêtes



Des cables, des liaisons radios, des routeurs, des switches.

Les fonctions de l'Internet sont organisées en couches (*layers*):

- Chaque couche communique avec une entité paire de la même couche
- En utilisant les services des couches inférieures



La couche Transport

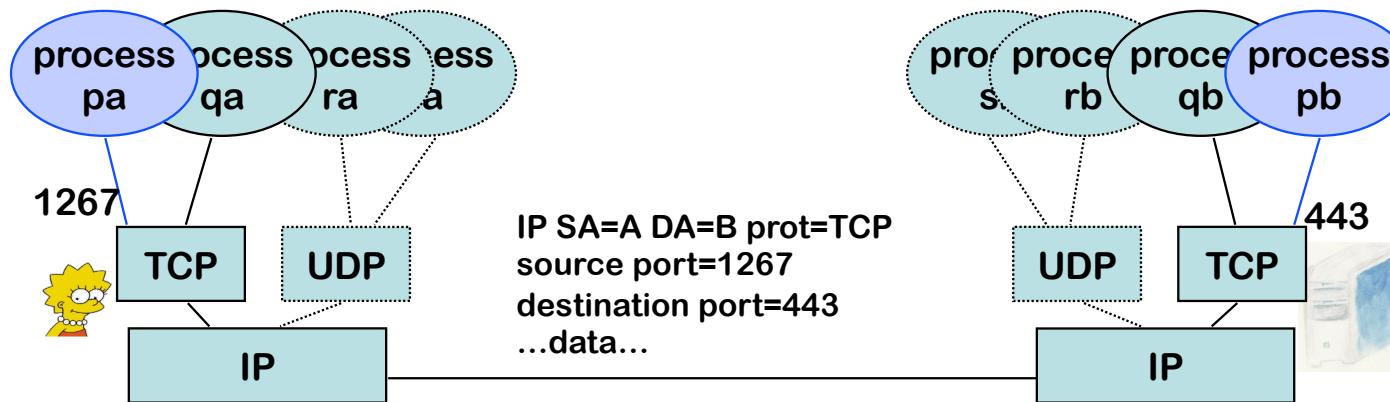
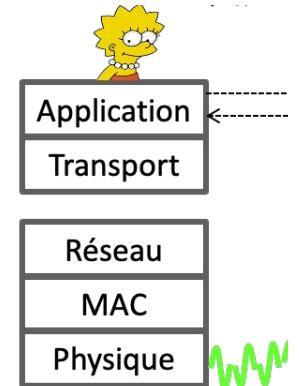
La couche application est le logiciel des applications en réseau.

La couche application utilise la couche Transport, qui fournit une **interface de programmation** à la couche application.

La couche Transport existe en 2 variantes:

- UDP: simpliste, non fiable.
- TCP: répare toutes les pertes de paquets réparables, plus complexe.

La couche transport utilise des **numéros de port** pour distinguer les processus à l'intérieur d'une machine.



2. La couche Réseau et la couche MAC

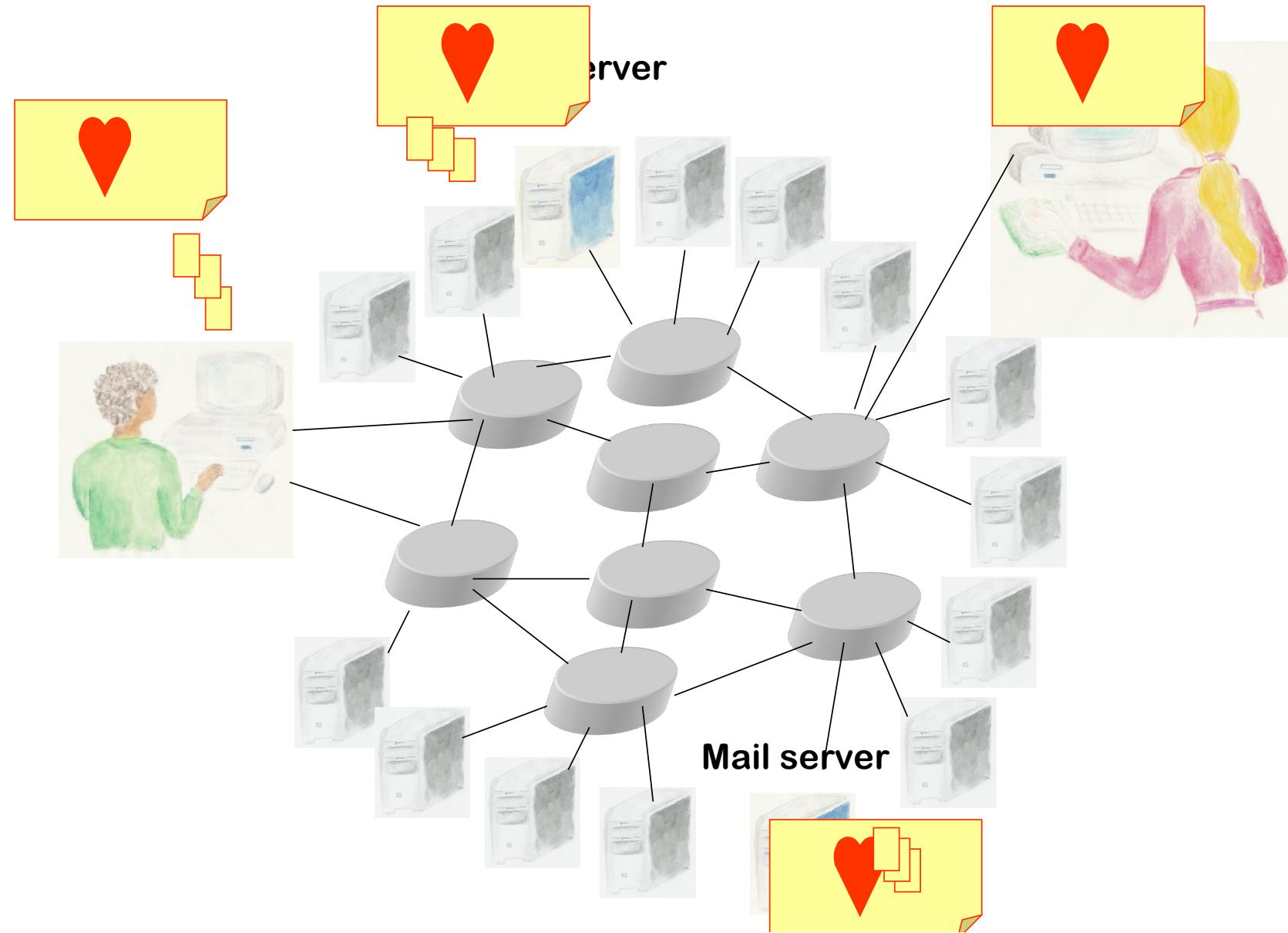
La couche réseau permet d'interconnecter tous les systèmes dans un univers donné (Internet, Intranet). Elle utilise le protocole **IP** (Internet Protocol).

Principe No1 du protocole IP:

- Chaque interface de communication reçoit un numéro appelé « adresse IP », qui est unique dans un univers donné.
- Les données à transmettre de la couche transport sont mises en **paquets** de longueur ≤ 1500 Octets (en principe).
- Chaque «paquet IP» contient l'adresse de destination (et de source). Des systèmes intermédiaires, appelés **routeurs** trient et transmettent les paquets d'après l'adresse de destination (comme des cartes postales).

Louis Pouzin 1973, premier réseau à commutation de paquets, Cyclades, France
Vint Cerf et Bob Kahn, TCP/ IP, 1974





Format des Adresses: IPv4

L'ancien plan de numérotation, encore très répandu, est appelé **IPv4**

Chaque adresse comporte 32 bits, et est souvent écrite en notation décimale pointée, où un entier dans {0, 1,..., 254, 255} représente 8 bits.

Une adresse à l'EPFL: 128.178.151.1

Adresses privées: 192.168.1.23, 172.16.0.0 à 172.31.255.255, 10.201.121.98.

Cette machine (localhost): 127.0.0.1

Système	Alphabet	Exemple
Binaire	{0,1}	1000 0000 1011 0010 1001 0111 0000 0001
Decimal pointé	{0,1,2,3,4,5,6,7,8,9,.}	128.178.151.1
Hexa-décimal	{0,1,2,3,4,5,6,7,8,9, a, b, c, d, e, f}	80b2 9701

$$1011\ 0010_{bin} = b2_{hex} = 128 + 32 + 16 + 2 = 178$$

$$ff_{hex} = 1111\ 1111_{bin} = 255$$

Principe No 2 du Protocole IP: «longest prefix match»

- Chaque système (terminal ou intermédiaire) possède une **table de routage** qui est une liste de couples (préfixes, où le paquet doit être envoyé)
- Quand un paquet IP est reçu, si l'adresse de destination n'est pas une des adresses de ce système, on cherche le plus long match dans la liste des prefixes

<i>Destination</i>	<i>Next-Hop / Interface</i>
128.178.29/24	128.178.100.2 / south
128.178/16	128.178.100.3 / south
0/0	128.178.47.3 / north

Table de routage du routeur ed0-swi

= routeur

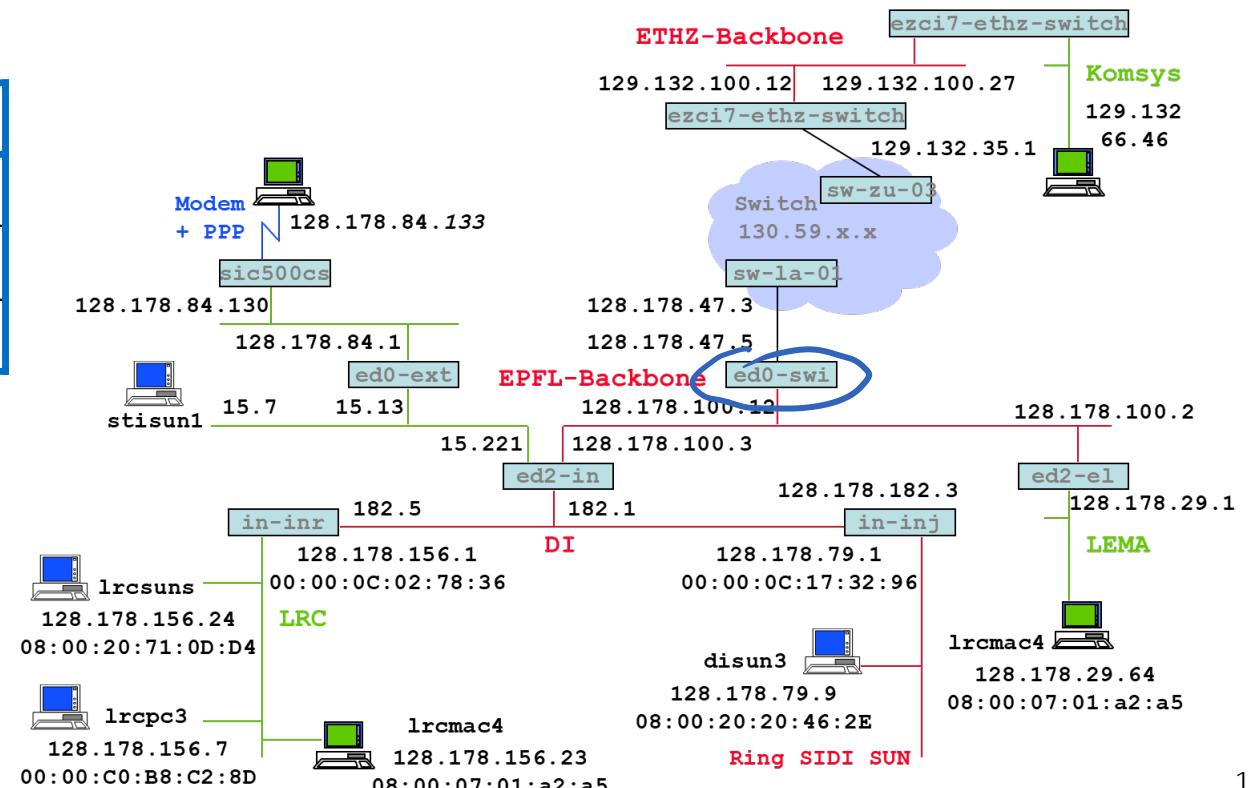
/24 signifie un préfixe de 24 bits

0/0 (vide): défaut

Ex: pour 128.178.* → ed2-in

pour 128.178.29.* → ed2-el

sinon → sw-la-01

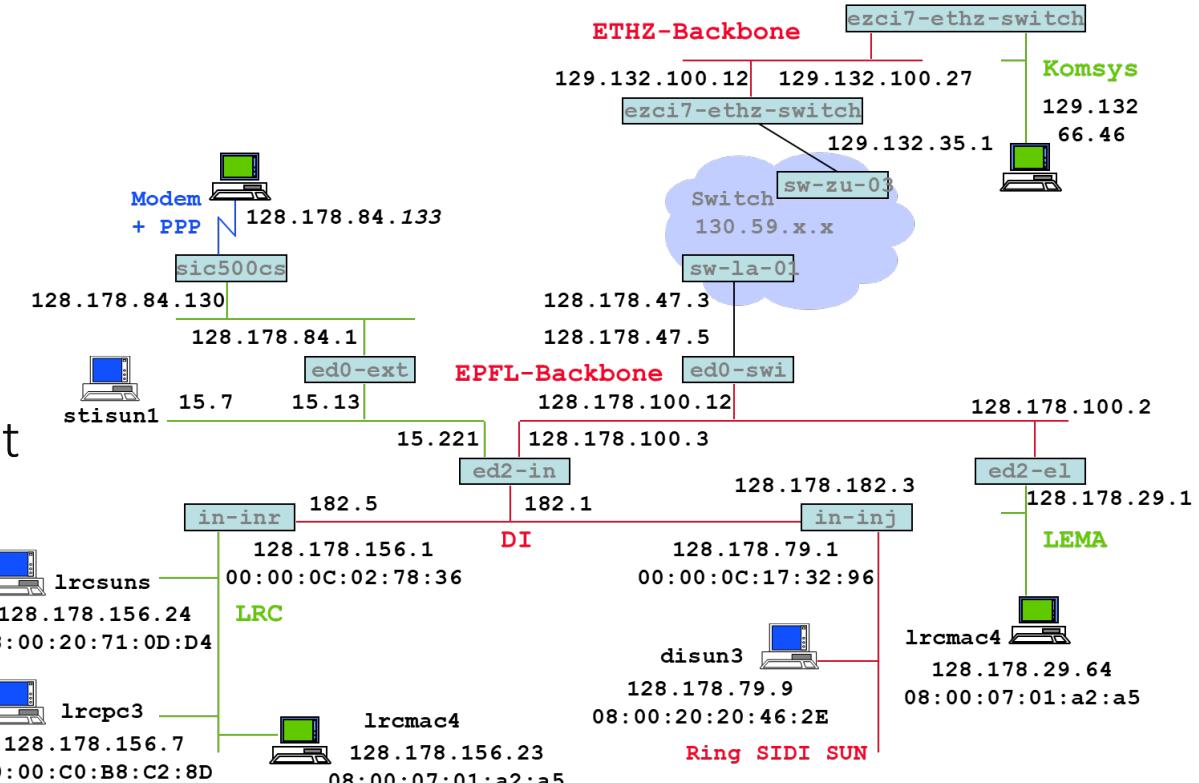


Corollaire: Les adresses IP sont structurées

L'adresse IP de mon smartphone n'est pas la même quand je suis à l'EPFL ou à la maison car elle reflète à quel routeur je suis connecté.

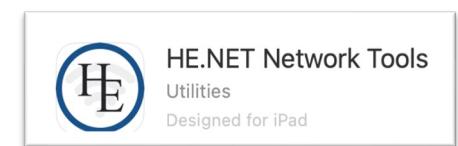
Elle doit être **configurée**, manuellement ou automatiquement (DHCP)

Dans les routeurs, les tables de routage doivent aussi être configurées, manuellement ou par un protocole de routage (OSPF, BGP).

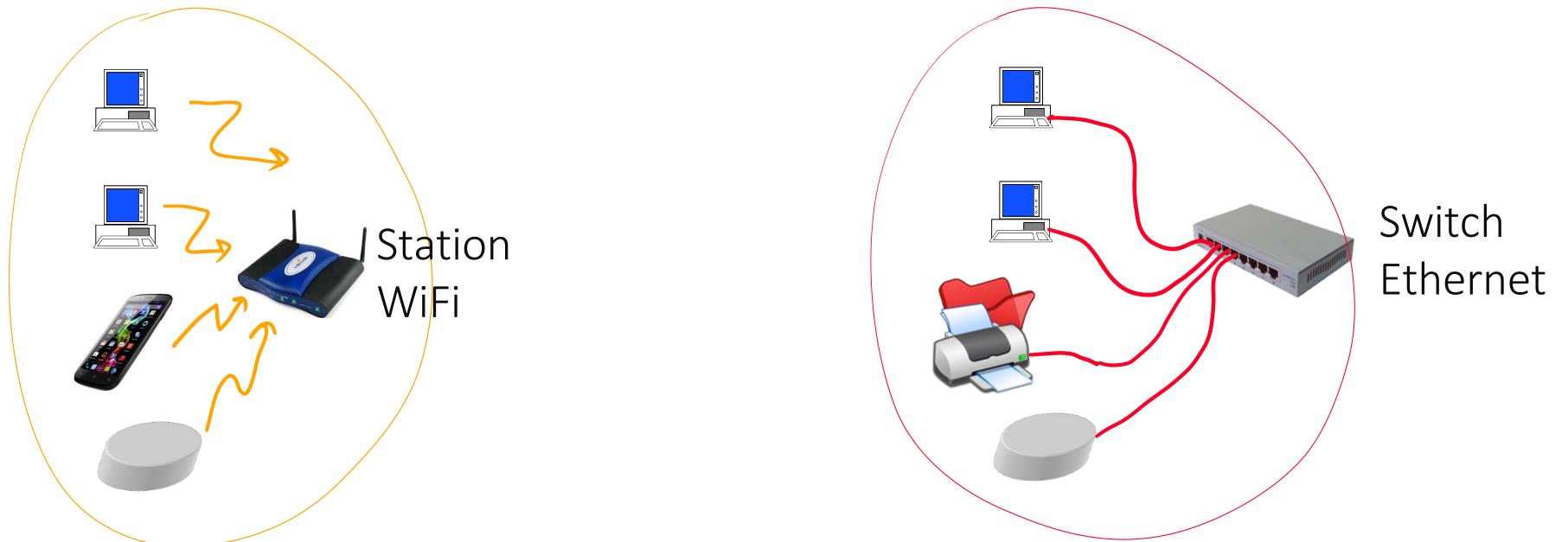


Regardez votre adresse IP:

- Windows: ipconfig
- Mac, Linux: ifconfig
- Smartphone: HE.NET app



On pourrait faire un réseau en interconnectant les systèmes avec des routeurs, puis en connectant les routeurs à des routeurs. Mais ce n'est pas exactement comme cela qu'on fait d'habitude: on interconnecte les systèmes d'abord via des systèmes intermédiaires qui n'utilisent pas les adresses IP (et ne sont donc pas des routeurs).



Pourquoi ? Comment ça marche ?

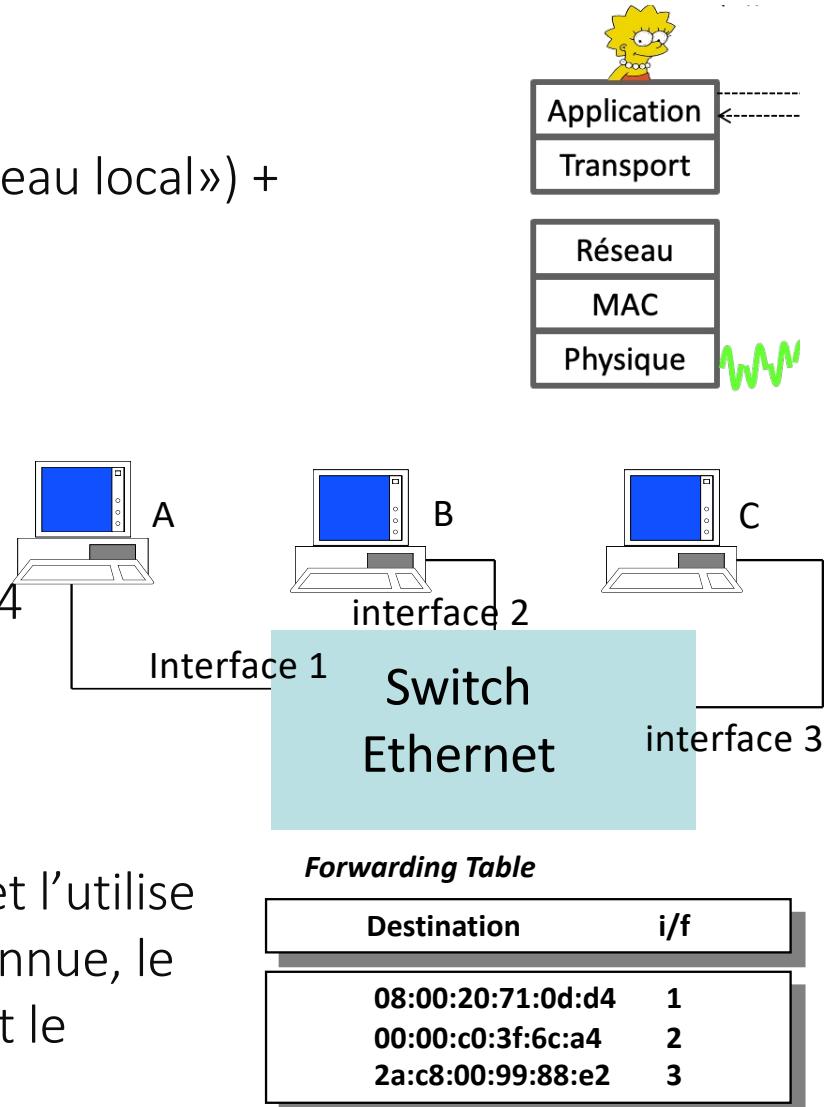
La couche MAC

But = interconnecter des systèmes en petit nombre («réseau local») + sans configuration («plug and play»)

Principe:

- Chaque interface de communication possède une «**adresse MAC**» (48 bits), aussi appelée adresse physique, qui est (en principe) son numéro de série; écrite en hexa, par exemple 08:00:20:71:0d:d4
- Les paquets MAC contiennent adresses MAC de destination et de source.
- Un switch Ethernet construit, en observant le trafic, une table des adresses MAC des systèmes connectés, et l'utilise pour transférer les paquets. Si une destination est inconnue, le paquet est broadcasté (envoyé sur tous les ports). C'est le fonctionnement en mode «bridge».

Aucune configuration, mais ne passe pas à l'échelle.



Comment IP et MAC interagissent

On peut interconnecter des réseaux locaux au moyen de routeurs

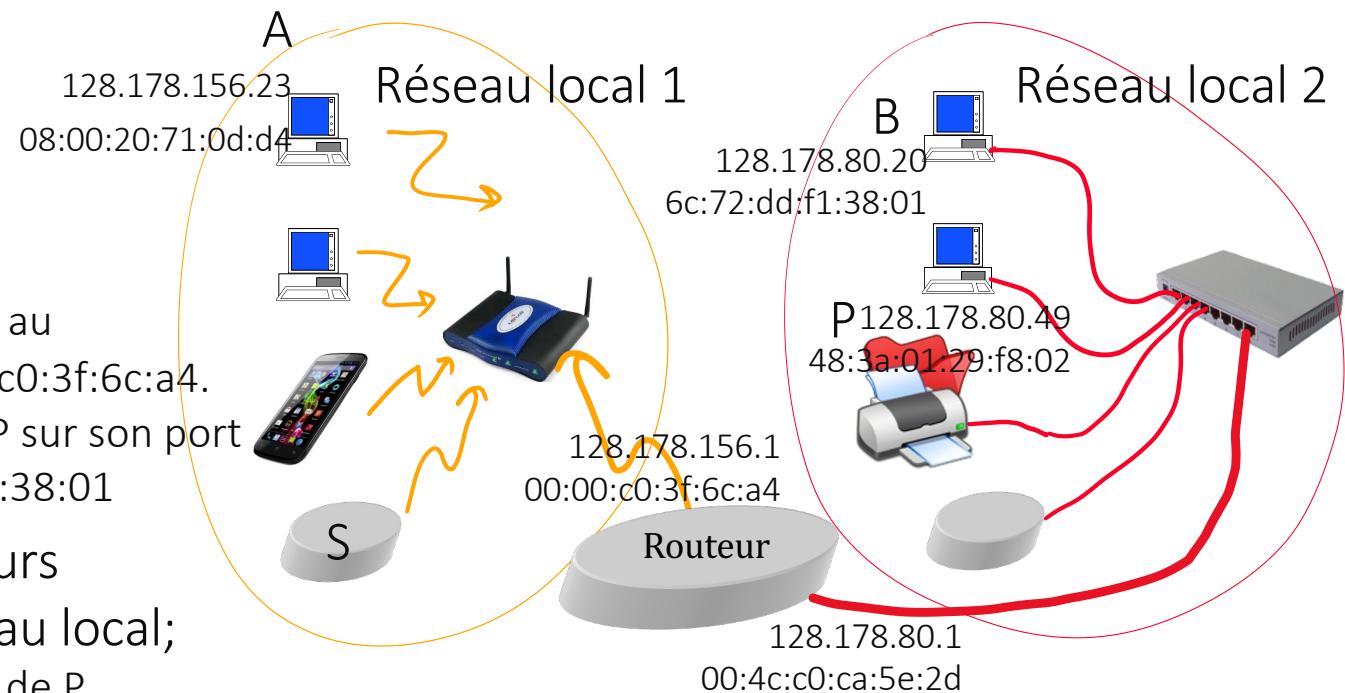
Chaque hôte doit connaître l'adresse IP de son routeur par défaut («default gateway»).

Pour A c'est 128.178.156.1

Il faut le configurer

A→P: A crée un paquet IP, adresse destination 128.178.80.49 et l'envoie au routeur, donc à l'adresse MAC 00:00:c0:3f:6c:a4. Le routeur renvoie le même paquet IP sur son port Ethernet à l'adresse MAC 6c:72:dd:f1:38:01

La communication utilise toujours IP même à l'intérieur d'un réseau local; ex: B→P, B doit connaître l'adresse IP de P



Principe No 3 du Protocole IP

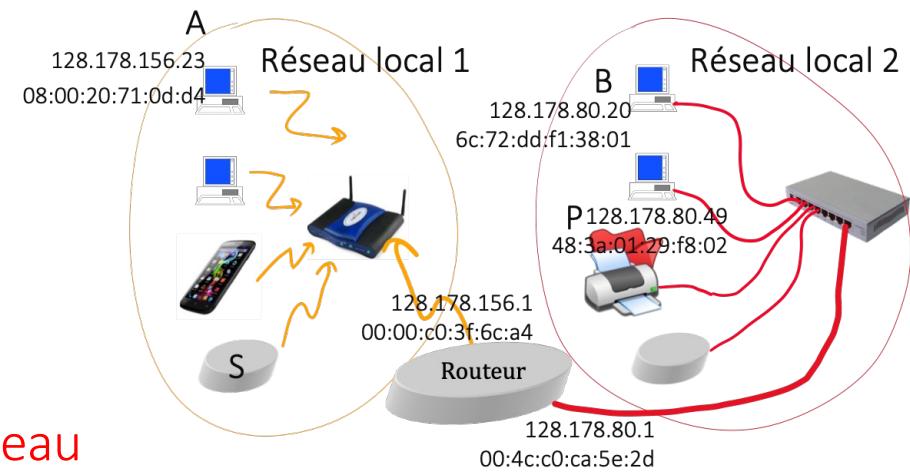
IP est défini comme une méthode d'interconnexion de réseaux locaux

IP = Internet protocol

«subnet» = sous-réseau = réseau local

Principe No 3:

- Communication entre **sous-réseaux différents** passe par un ou plusieurs routeurs
- Communication entre hôtes du **même sous-réseau** ne passe pas par un routeur



Corollaire: un hôte doit savoir reconnaître si une adresse IP de destination est dans le même sous-réseau.

Masque de sous-réseau

A chaque sous-réseau (= réseau local) doit être associé un préfixe de sous-réseau («subnet prefix»).

Toutes les adresses IP d'un même sous-réseau doivent être configurées avec le même préfixe.

La taille (en bits) du préfixe peut-être quelconque. Il faut la spécifier dans la configuration de l'interface

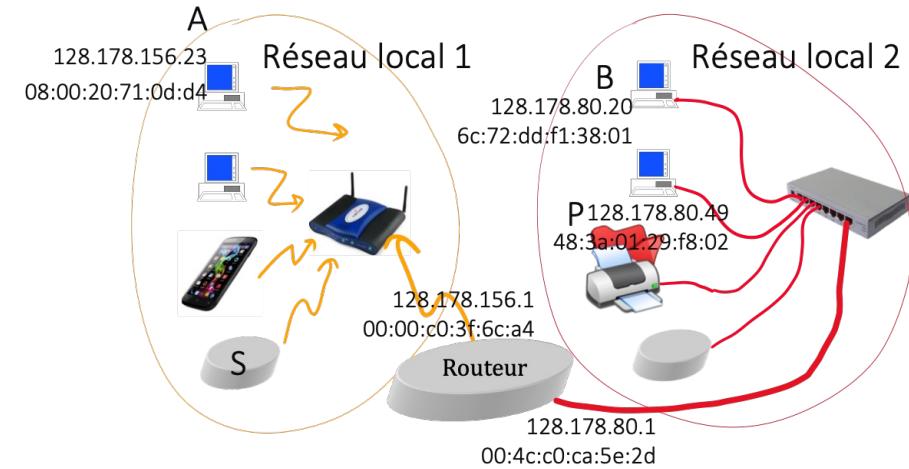
EPFL-IPv4: 24 bits: Exemple: **128.178.156.34** taille de préfixe = 24

ETHZ IPv4: taille de préfixe = 26

La taille du préfixe est usuellement spécifiée en utilisant un **masque** = suite de bits où les bits 1 indiquent les positions du préfixe, souvent écrite en notation décimale pointée.

Ex: EPFL: 1111 1111 1111 1111 1111 1111 0000 0000 → 255.255.255.0

ETHZ: 1111 1111 1111 1111 1111 1111 1100 0000 → 255.255.255.192



Traduction pratique du principe No 3

<i>Destination</i>	<i>Next-Hop</i>	Table de routage de B
128.178.80/24	onlink	
0/0	128.178.80.1	

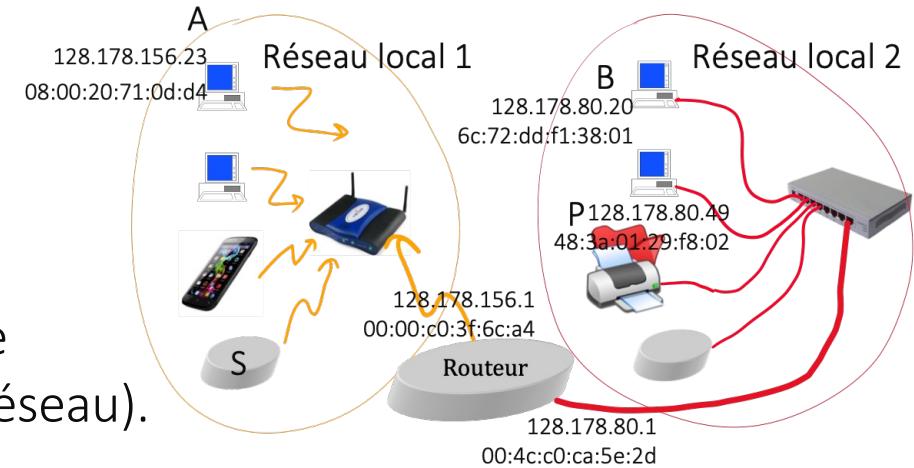
La table de routage d'un hôte tel que B est dérivée automatiquement de son adresse, son masque et le routeur par défaut («onlink»= dans le même sous-réseau).

B→A

B reçoit de la couche transport la tâche d'envoyer un paquet à 128.178.156.23. Longest prefix match ⇒ B envoie le paquet à son routeur par défaut, càd B met le paquet IP dans un paquet MAC dont l'adresse de destination est 00:4c:c0:ca:5e:2d (interface Ethernet du routeur).

B→P

Longest prefix match ⇒ B envoie le paquet directement à P càd B met le paquet IP dans un paquet MAC dont l'adresse de destination est 48:3a:01:29:f8:02 (adresse MAC de P)

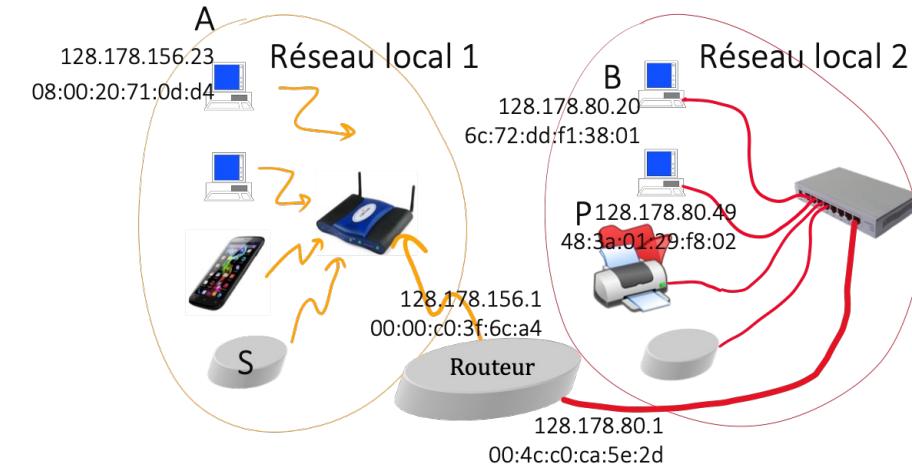


ARP

Comment A connaît-il l'adresse MAC de l'interface WiFi du routeur ?

Comment B connaît-elle l'adresse MAC de P ?

Par le protocole ARP (Address Resolution Protocol)



B envoie un paquet ARP diffusé («broadcast») à tous le sous-réseau Ethernet, qui demande: qui a l'adresse IP 128.178.80.49. P répond et donne son adresse MAC. On espère que la réponse vient bien de P et pas d'un système malveillant. La correspondance adresse IP ↔ adresse MAC est conservée (dans un «cache ARP») tant que le trafic reçu la confirme.

Regardez votre cache ARP
Ligne de commande Windows, Mac, Linux: arp -a

Configuration d'un hôte IP

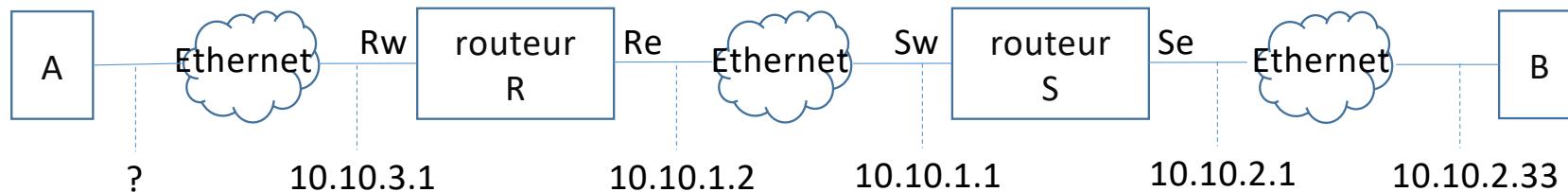
Un hôte IP doit être configuré avec:

- Pour chaque interface (WiFi, Ethernet, etc): adresse IP + masque
- Adresse IP de routeur par défaut
- Adresse IP de serveur DNS

Peut-être entrée manuellement, ou automatisée par DHCP («Dynamic Host Configuration Protocol»): au démarrage d'une interface, le système contacte un serveur de configuration (par exemple dans la borne WiFi).

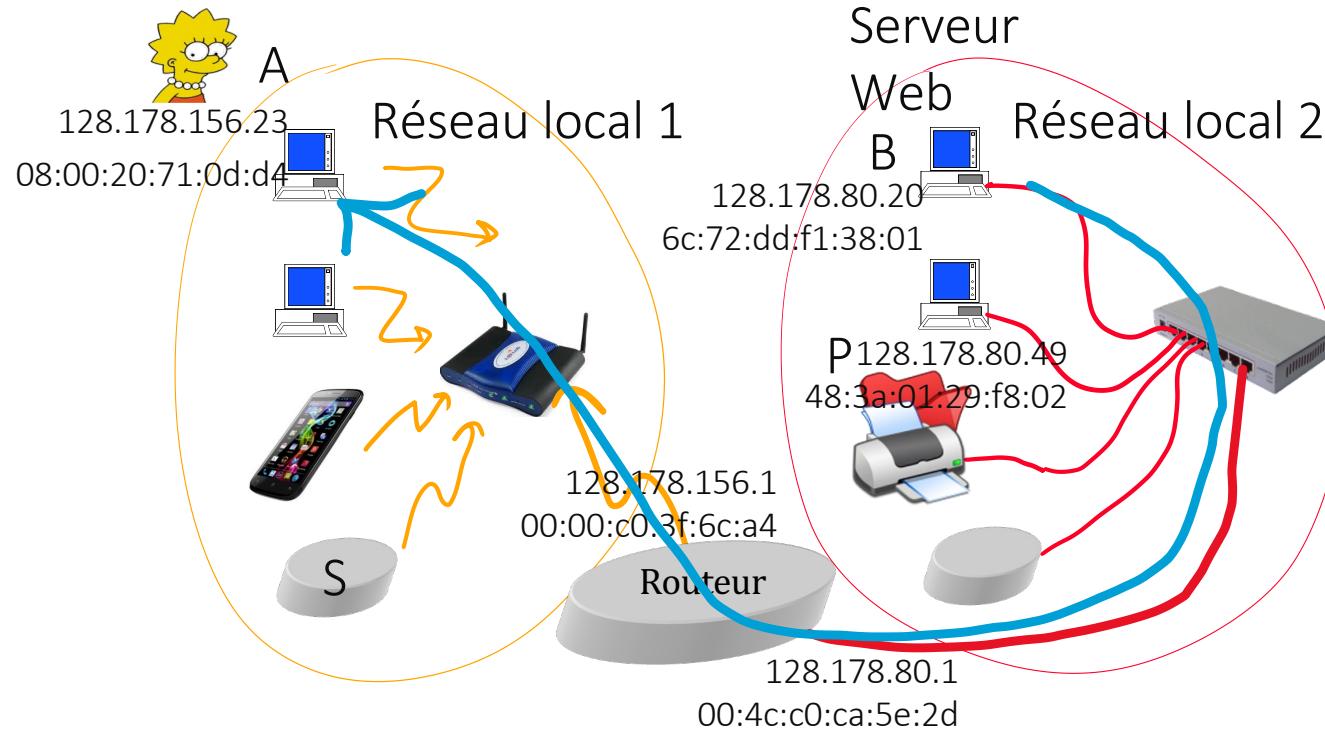
Dans tous les cas il faut un plan de numérotation.

Les masques de sous-réseaux sont tous égaux à 255.255.255.0. Quelle adresse est valide pour A ?



- A. 10.10.3.33
- B. 10.10.2.22
- C. A et B
- D. Aucun des deux
- E. Je ne sais pas

Rassemblons tous ces éléments...



Un serveur Web en B envoie des données à A



A



Application

read(s1, dataBlock)

Transport (TCP)

1 2 3 4 5

Network (IP)

2 3 4 5

MAC (WLAN)

2 3 4 5

Physical

Router

Serveur Web



Application

send(s2, dataBlock)

Transport (TCP)

1 2 3 4 5

Network (IP)

1 2 3 4 5

MAC (Ethernet)

1 2 3 4 5

Physical

Switch Ethernet

Network (IP)

1 2 3

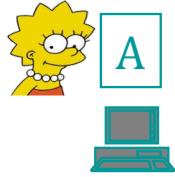
MAC

1 2 3

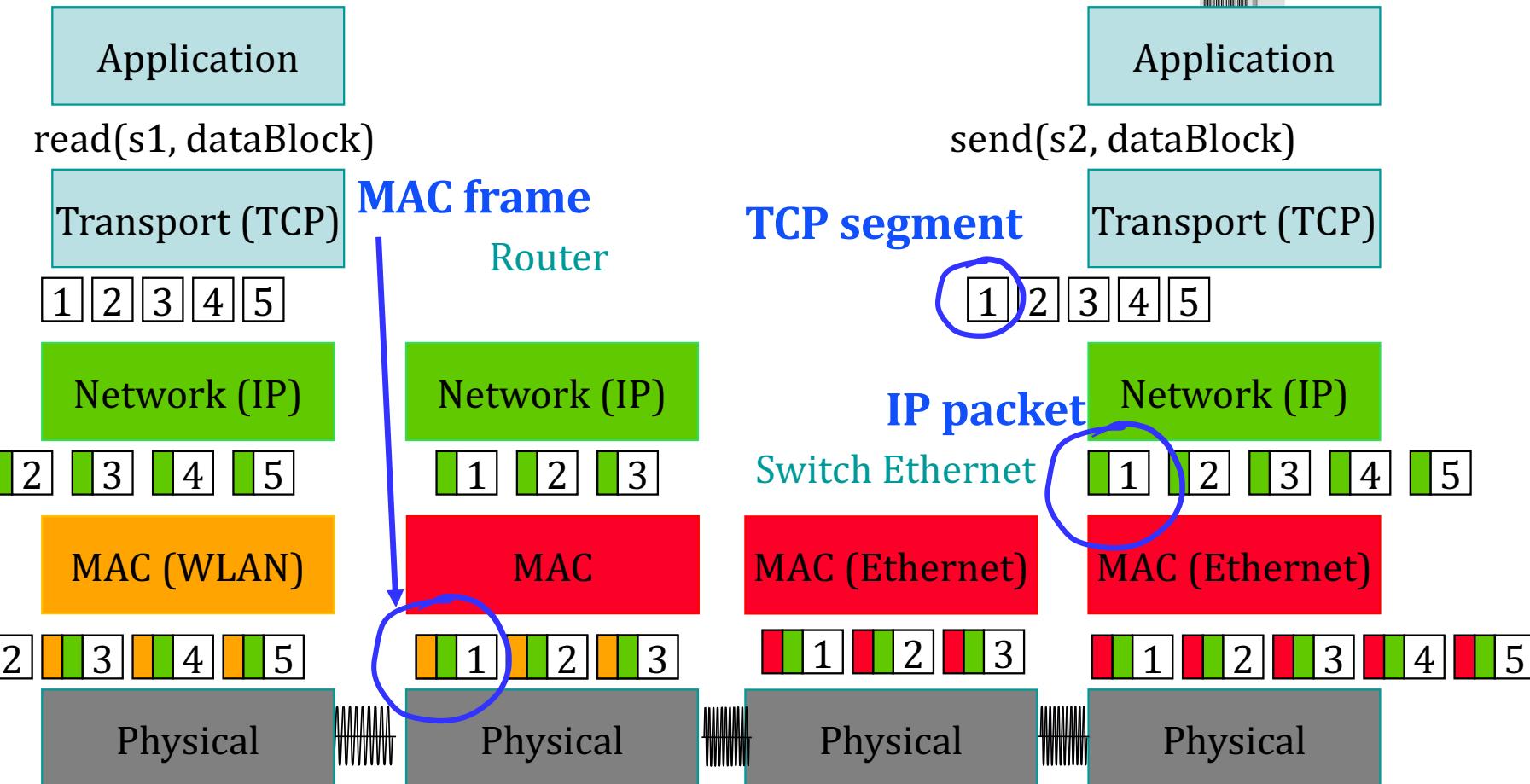
MAC (Ethernet)

1 2 3

Physical



Terminologie



3. Noms et adresses

L'internet utilise des noms, qui sont (entre autres) des synonymes pour des adresses IP.

Exemples: www.epfl.ch, smtp.sunrise.ch



Les noms sont traduits en adresses par les serveurs DNS

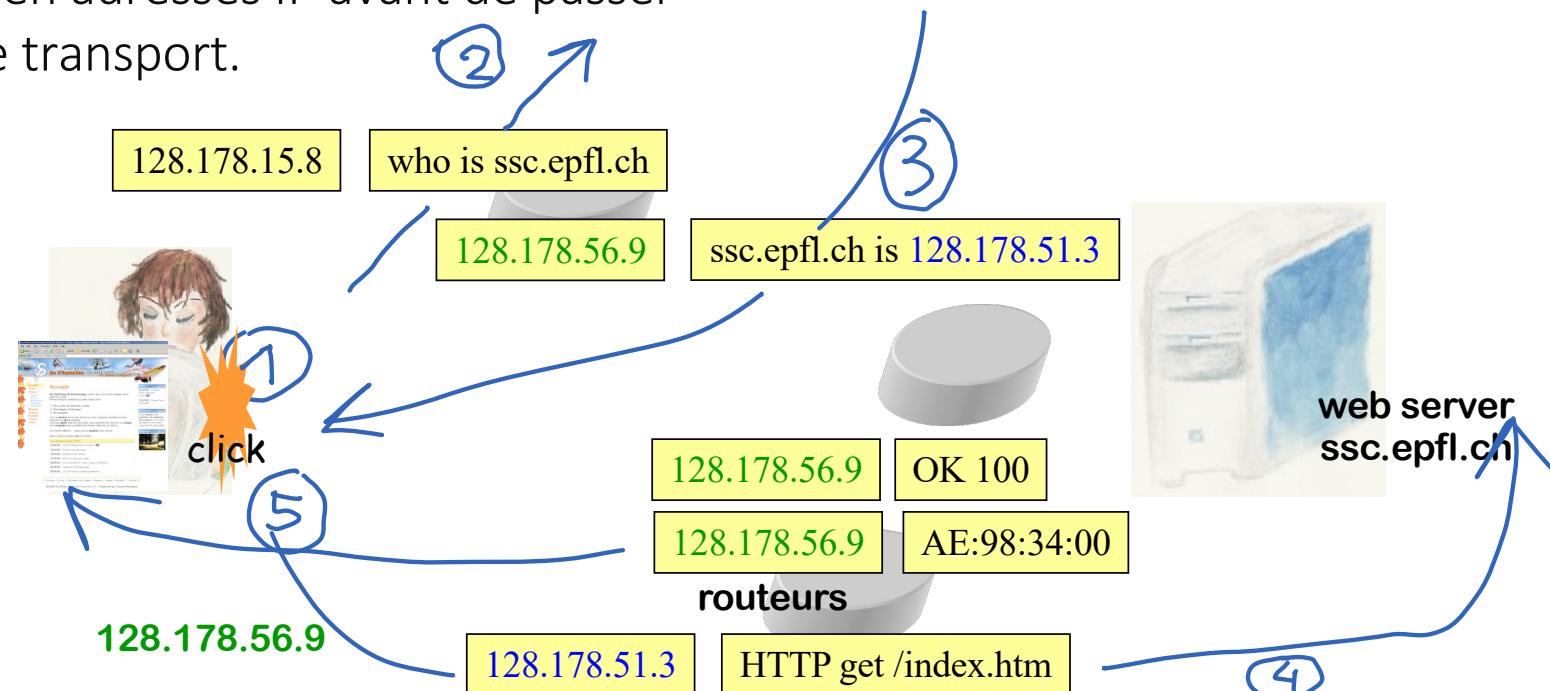
DNS = domain name system

Chaque fournisseur de service internet doit offrir des serveurs DNS à ses clients

La couche application utilise les serveurs DNS et traduit les noms en adresses IP avant de passer des données à la couche transport.



Dans le fichier du serveur DNS
ssc.epfl.ch is 128.178.51.13



Essayez:
Windows, Mac, Linux:
nslookup
Smartphone: HE.NET app

ping

La commande [ping](#) (windows, Mac, Linux, HE.NET app) envoie un message de test à une destination donnée sous forme de nom ou d'adresse IP

```
leboudec@icsil1noteb240 ~ % ping -c 3 www.canterbury.ac.nz
PING www.canterbury.ac.nz (132.181.106.9): 56 data bytes
64 bytes from 132.181.106.9: icmp_seq=0 ttl=223 time=302.115 ms
64 bytes from 132.181.106.9: icmp_seq=1 ttl=223 time=301.977 ms
64 bytes from 132.181.106.9: icmp_seq=2 ttl=223 time=302.241 ms
--- www.canterbury.ac.nz ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 301.977/302.111/302.241/0.108 ms
```

```
leboudec@icsil1noteb240 ~ % ping -c 3 www.ethz.ch
PING www.ethz.ch (129.132.19.216): 56 data bytes
64 bytes from 129.132.19.216: icmp_seq=0 ttl=48 time=4.517 ms
64 bytes from 129.132.19.216: icmp_seq=1 ttl=48 time=4.356 ms
64 bytes from 129.132.19.216: icmp_seq=2 ttl=48 time=4.311 ms
--- www.ethz.ch ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 4.311/4.395/4.517/0.088 ms
```

traceroute

```
leboudec@icsil1noteb240 ~ % traceroute www.canterbury.ac.nz
traceroute to www.canterbury.ac.nz (132.181.106.9), 64 hops max, 52 byte packets
 1  cv-dist-b-v151-ro (128.178.151.1)  0.598 ms  0.349 ms  0.241 ms
 2  cv-core-dist-b-168 (10.0.2.168)  0.335 ms  0.352 ms  0.307 ms
 3  fortி-core-epfl-6 (10.0.2.6)  0.266 ms  0.204 ms  0.255 ms
 4  cv-core-intranet-181 (10.0.2.181)  0.449 ms  0.531 ms  0.397 ms
 5  cv-internet-intranet-127 (10.0.2.127)  0.814 ms  0.478 ms  0.400 ms
 6  fortி-internet-intranet-178 (10.0.2.178)  0.254 ms  0.264 ms  0.263 ms
 7  cv-internet-internet-133 (10.0.2.133)  0.799 ms  0.740 ms  0.767 ms
 8  swiel3 (192.33.209.17)  0.954 ms  1.075 ms  1.057 ms
 9  swige3-100ge-0-0-1-1.switch.ch (130.59.36.82)  2.103 ms  2.017 ms  2.038 ms
10  swice2-100ge-0-0-0-10.switch.ch (130.59.38.193)  2.241 ms  3.127 ms  4.041 ms
11  switch.mx1.gen.ch.geant.net (62.40.124.21)  3.247 ms  1.872 ms  34.086 ms
12  ae7.mx1.par.fr.geant.net (62.40.98.239)  9.890 ms  9.817 ms  9.745 ms
13  hundredrdege-0-0-0-22.102.core1.newy32aoa.net.internet2.edu (198.71.45.236)  83.302 ms  86.952 ms  83.599 ms
14  fourhundredrdege-0-0-0-2.4079.core1.ashb.net.internet2.edu (163.253.1.116)  158.890 ms  158.650 ms  157.387 ms
15  fourhundredrdege-0-0-0-18.4079.core2.ashb.net.internet2.edu (163.253.1.107)  158.884 ms  158.413 ms  158.692 ms
16  fourhundredrdege-0-0-0-1.4079.core2.clev.net.internet2.edu (163.253.1.139)  158.654 ms  158.798 ms  158.848 ms
17  fourhundredrdege-0-0-0-2.4079.core2.eqch.net.internet2.edu (163.253.2.17)  182.013 ms  173.136 ms  198.211 ms
18  fourhundredrdege-0-0-0-2.4079.core2.chic.net.internet2.edu (163.253.2.18)  160.076 ms  158.350 ms  158.850 ms
19  fourhundredrdege-0-0-0-1.4079.core1.kans.net.internet2.edu (163.253.1.245)  158.972 ms  160.021 ms  160.103 ms
20  fourhundredrdege-0-0-0-1.4079.core1.denv.net.internet2.edu (163.253.1.242)  160.490 ms  158.259 ms  158.155 ms
21  fourhundredrdege-0-0-0-3.4079.core1.salt.net.internet2.edu (163.253.1.171)  159.987 ms  157.754 ms  159.299 ms
22  fourhundredrdege-0-0-0-1.4079.core1.seat.net.internet2.edu (163.253.1.157)  159.423 ms  158.744 ms  158.431 ms
23  fourhundredrdege-0-0-0-16.4079.core2.seat.net.internet2.edu (163.253.2.55)  159.511 ms  158.893 ms  158.472 ms
24  reannz-1-lo-jmb-706.sttlwa.pacificwave.net (207.231.240.33)  156.902 ms  157.074 ms  156.801 ms
25  210.7.37.209 (210.7.37.209)  302.161 ms  302.010 ms  302.132 ms
26  210.7.37.210 (210.7.37.210)  384.072 ms  319.099 ms  320.738 ms
27  202.36.179.65 (202.36.179.65)  302.168 ms  302.182 ms  302.007 ms
28  132.181.3.236 (132.181.3.236)  303.059 ms  302.994 ms  304.032 ms
29  132.181.106.9 (132.181.106.9)  302.111 ms  302.115 ms  302.263 ms
```

La commande traceroute
(Mac, Linux, HE.NET) ou
tracert (Windows) détecte
les routeurs entre ce
système et la destination

Essayez :



www.wireshark.org

```
leboudec@icsil1noteb240 ~ %
traceroute www.ethz.ch
traceroute to www.ethz.ch (129.132.19.216),
64 hops max, 52 byte packets
```

puis wireshark avec display filter:

ip.addr == 129.132.19.216

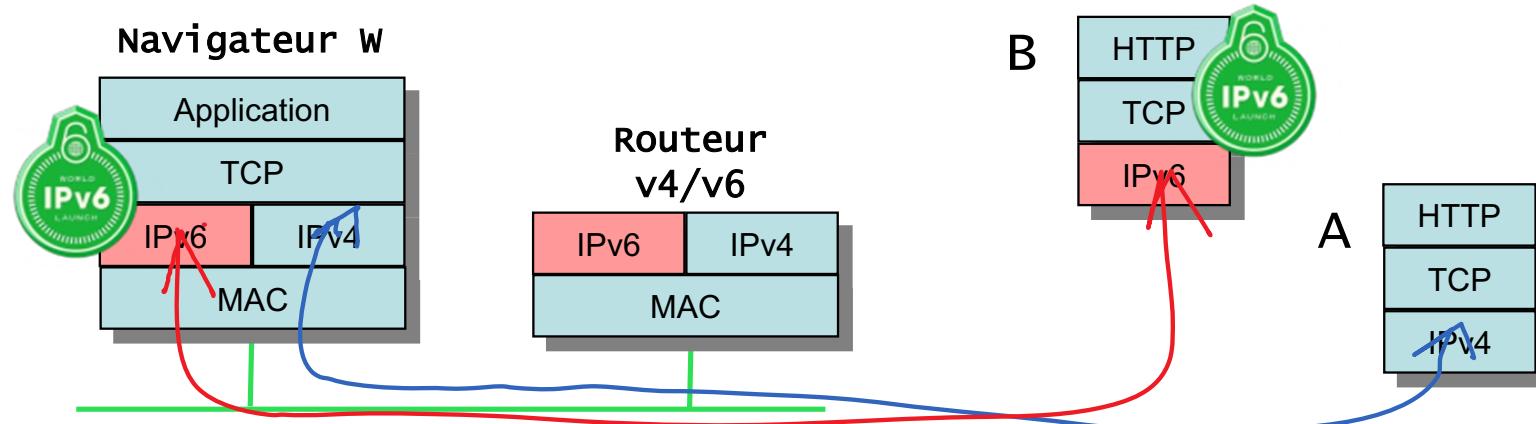
4. IPv6

La version la plus répandue de IP est IPv4 mais la plus récente est IPv6.

Pourquoi une nouvelle version ? IPv4 a $2^{32} \approx 4 \cdot 10^9$ adresses et les adresses IPv4 sont épuisées.

Comment ? IPv6 redéfinit le format des paquets et étend l'adresse à 128 bits ($\approx 3 \cdot 10^{38}$ adresses). Sinon, mêmes principes que IPv4.

Mais IPv6 est incompatible avec IPv4: W peut communiquer avec A et B mais A et B ne peuvent pas communiquer entre eux directement. Les systèmes à-jour supportent à la fois IPv4 et IPv6, mais certains réseaux ne supportent que IPv6 (ou que IPv4).



Les adresses IPv6

128 bits = 8 groupes de 16 bits = 8 groupes de 4 chiffres hexa

Une adresse EPFL: 2001:0620:0618:01a6:0a00:20ff:fe78:30f9

Une adresse privée EPFL: fd24:ec43:12ca:01a6:0a00:20ff:fe78:30f9

Une adresse «link-local»: fe80::cdb:b44a:3a8c:3c7e

Cette machine (localhost): 0000:0000:0000:0000:0000:0000:0000:0001

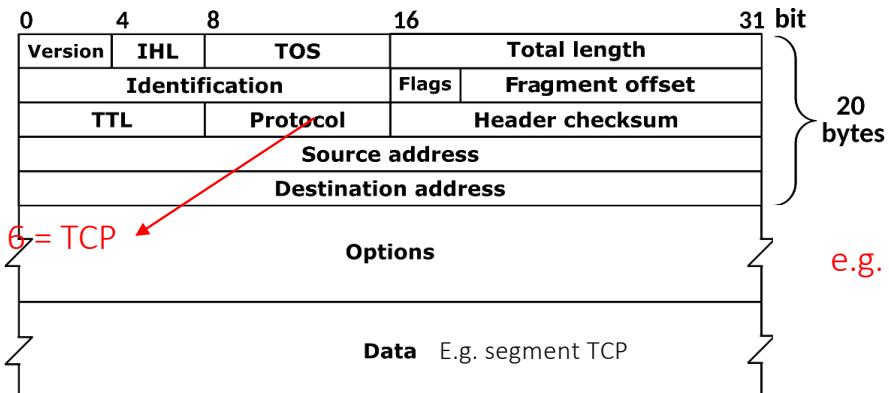
Règles de compression pour lecture et écriture:

- Les groupes de 16 bits (= 4 chiffres hexa) sont séparés par :
- Les zéros au début d'un groupe peuvent être omis
- :: remplace un nombre quelconque de groupes égaux à 0; ne peut apparaître qu'au plus une seule fois dans une adresse

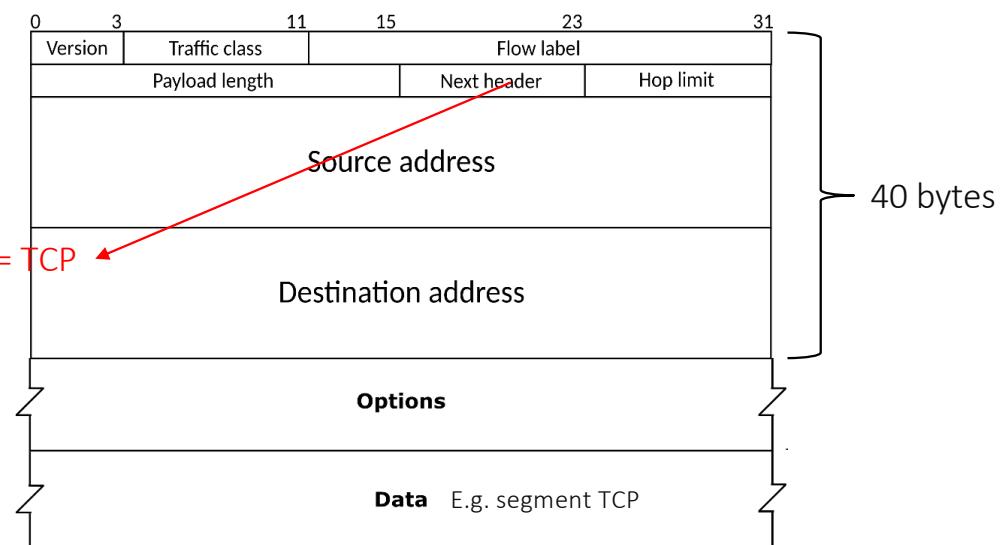
<i>Non comprimée</i>	<i>Comprimée</i>
2002:0000:0000:0000:0000:ffff:80b2:0c26	2002::ffff:80b2:c26
2001:0620:0618:01a0:0000:0000:fe78:30f9	2001:620:618:1a0::fe78:30f9
0000:0000:0000:0000:0000:0000:0000:0001	::1

Formats des paquets IP

IPv4

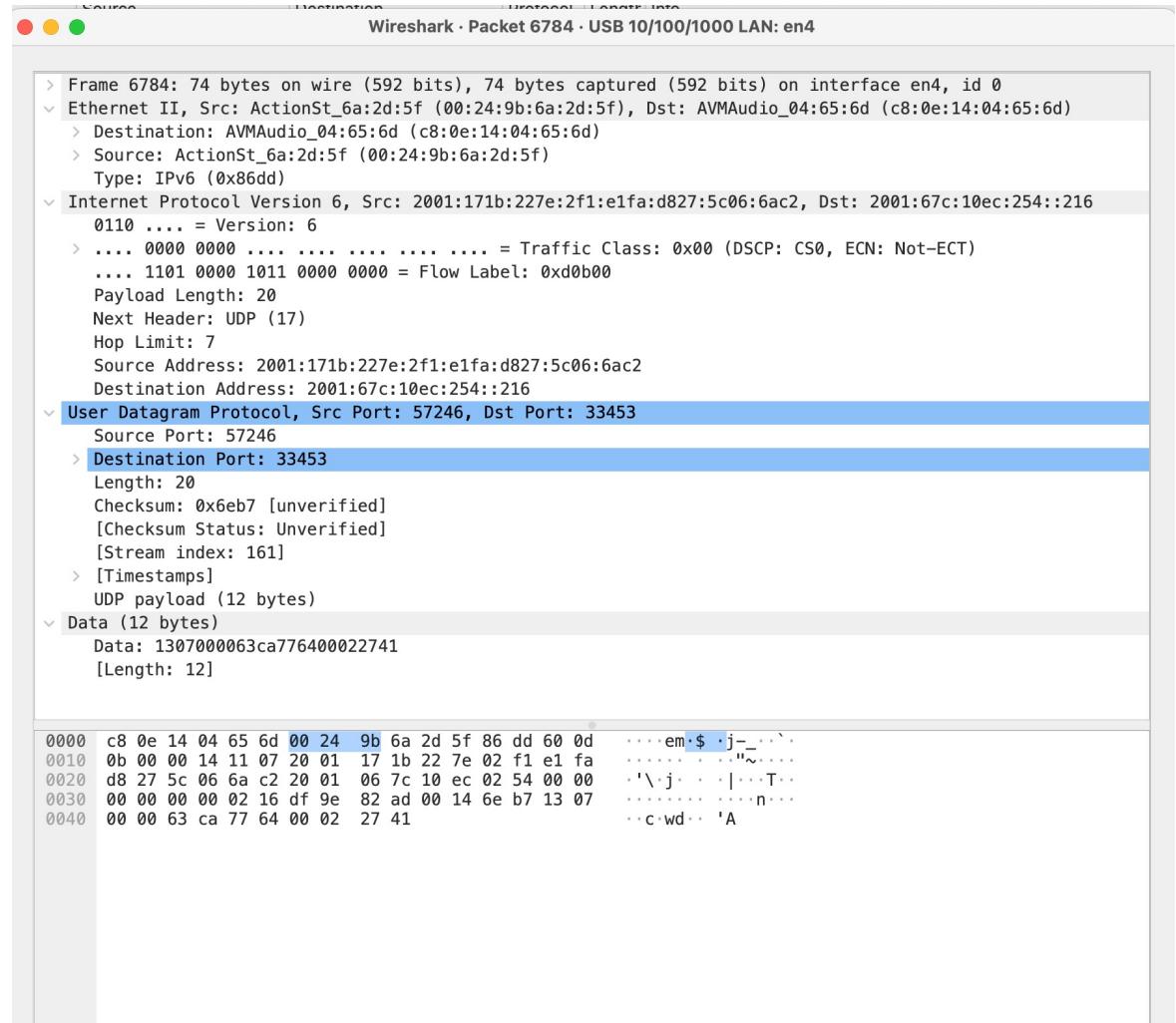


IPv6



L'en-tête IP comprend 20 octets (IPv4) / 40 octets (IPv6) + options

Le champ TTL/HL est initialisé à 64 à la source (par défaut) et décrémenté d'1 par chaque routeur; s'il atteint 0, le paquet est détruit – utilisé pour détruire les paquets pris dans des boucles, aussi utilisé par traceroute.



```

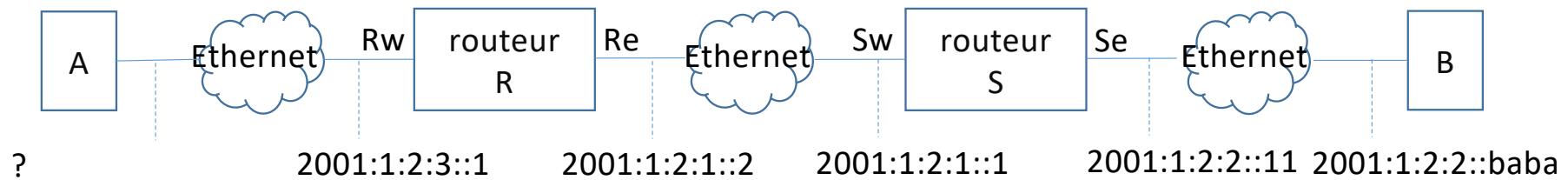
leboudec@icsil1noteb240 ~ % traceroute6 www.ethz.ch
traceroute6 to www.ethz.ch (2001:67c:10ec:254::216)
from 2001:171b:227e:2f1:e1fa:d827:5c06:6ac2, 64 hops
max, 12 byte packets

```

puis wireshark avec display filter:

ipv6.addr == 2001:67c:10ec:254::216

Les masques de sous-réseaux ont tous pour longueur 64.
Quelles sont les adresses valides pour A ?

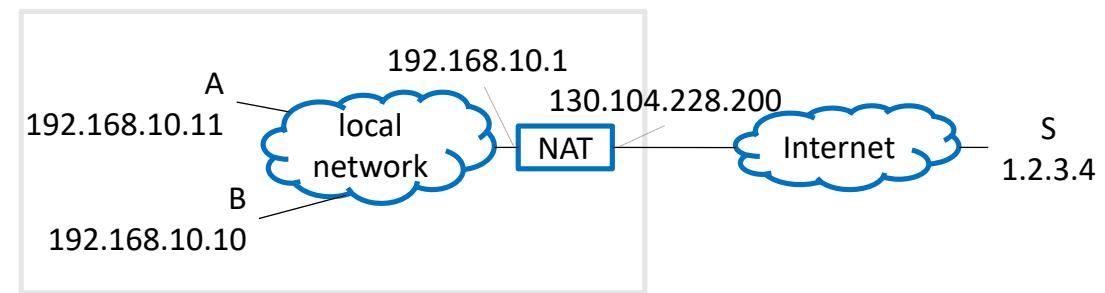


- A. 2001:1:2:2::2
- B. 2001:1:2:3::2
- C. A et B
- D. Aucun des deux
- E. Je ne sais pas

5. Les NATs («Network Address Translation»)

Pourquoi ? Utiliser *n* appareils sur internet à partir d'*une seule* adresse IPv4.

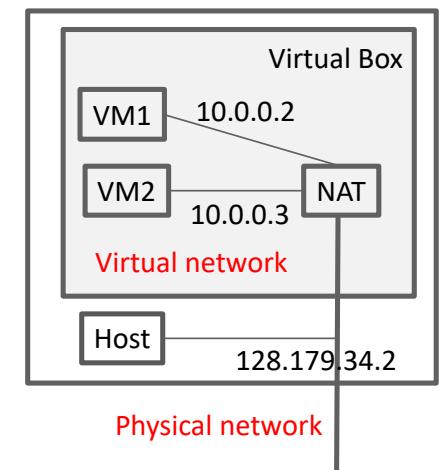
Une ou plusieurs machines virtuelles accèdent l'interface physique de la machine hôte.



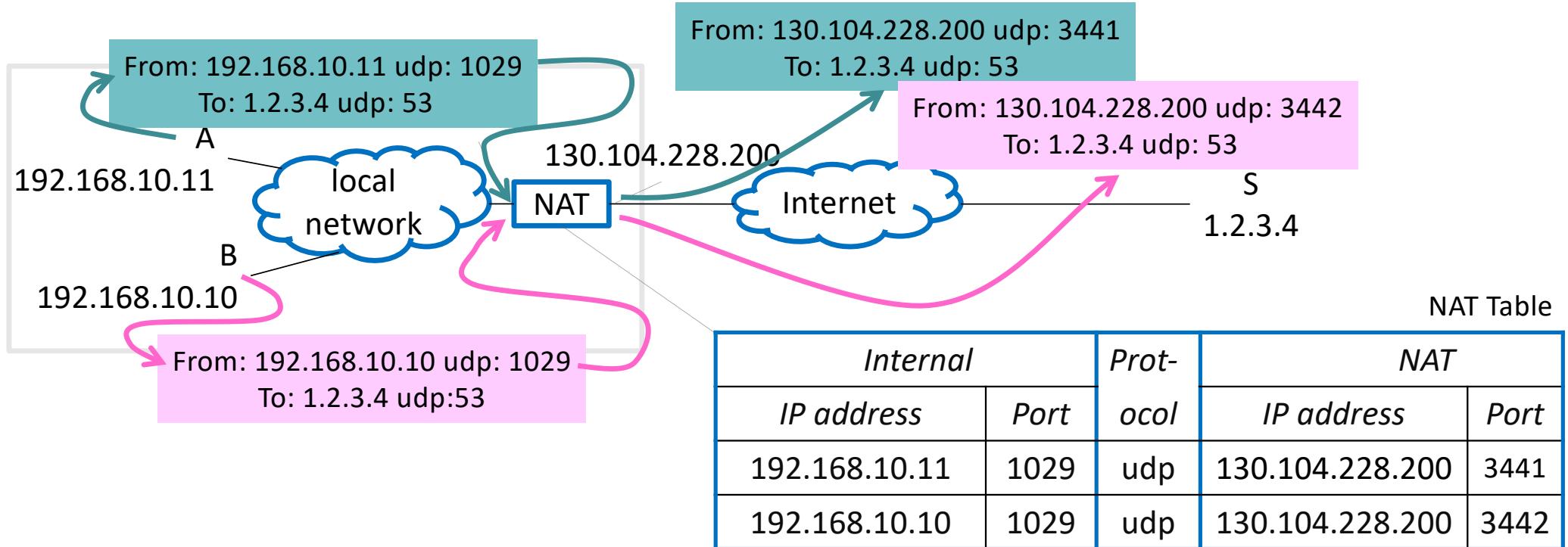
Le NAT permet de

- partager une adresse IP entre plusieurs machines,
- utiliser des adresses privées pour se connecter à l'Internet public.

C'est une déviation des principes du protocole IP.



Le NAT traduit les adresses IP et triche avec les ports

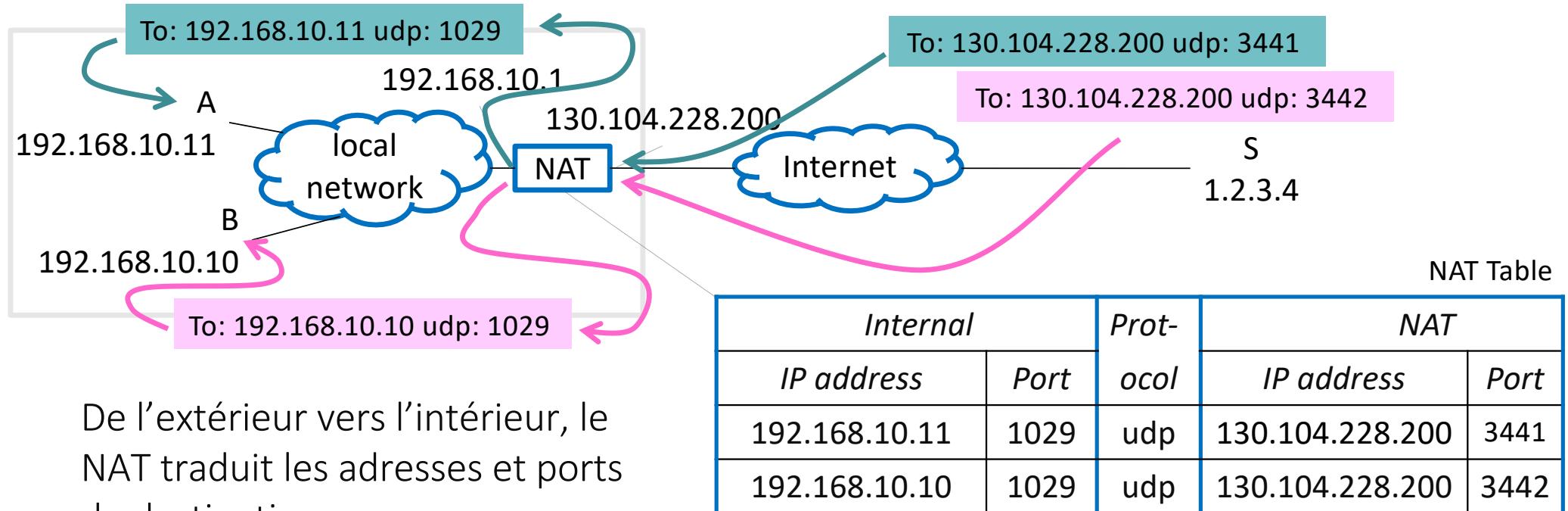


Le NAT remplace l'adresse IP source de A par sa propre adresse IP.

Problème: comment différencier A et B ?

Solution: en manipulant les numéros de port (de la couche transport !)

Le NAT traduit les adresses IP (suite)



De l'extérieur vers l'intérieur, le NAT traduit les adresses et ports de destination.

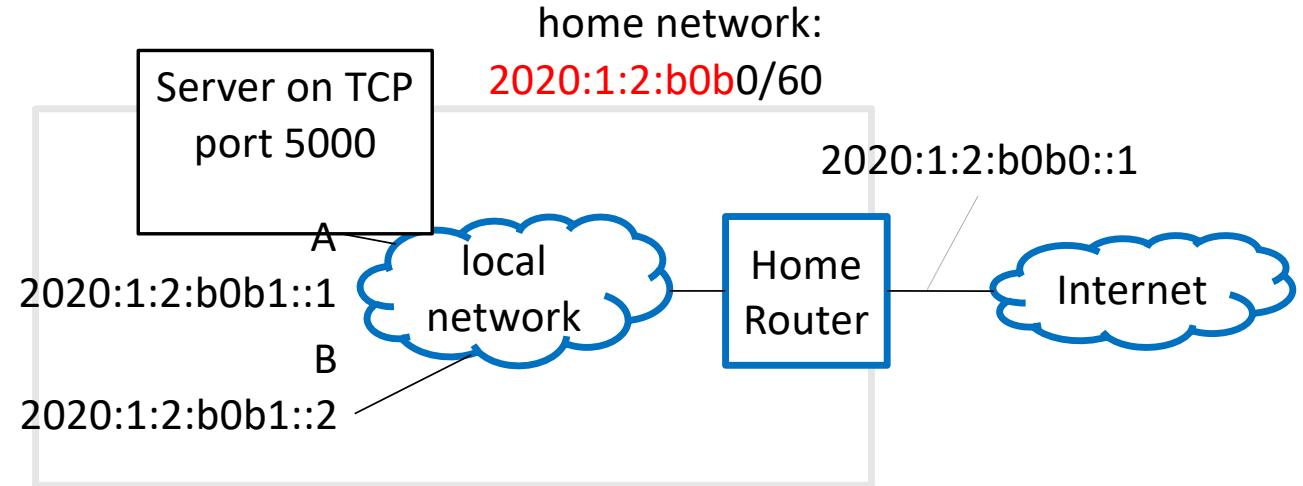
Pour S, A et B apparaissent comme s'ils étaient deux processus dans une même machine.

Le NAT est souvent appelé «routeur» mais ce n'est pas formellement correct (un routeur est un système intermédiaire de la couche IP) (fonction ≠ objet).

Les NATs en IPv6

Le NAT a été développé à cause de la rareté des adresses IPv4.

Les «home routers» n'ont pas besoin de fonctionner en mode NAT car, avec IPv6, on reçoit typiquement un (énorme) bloc d'adresses.

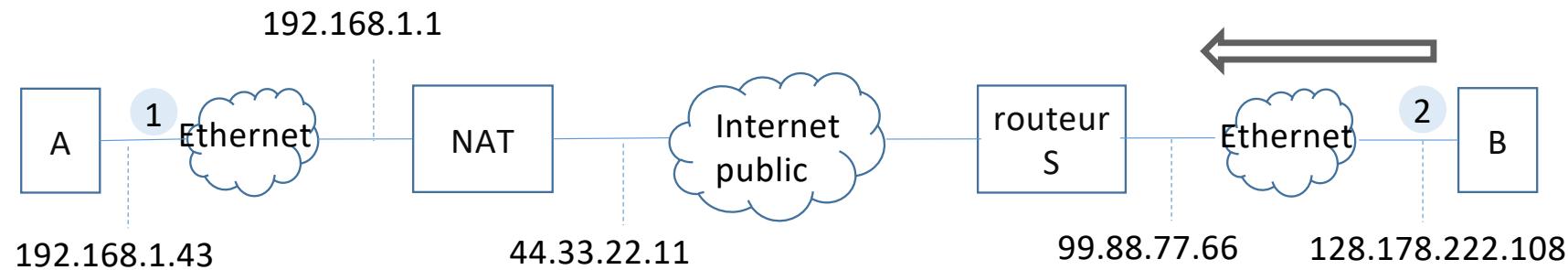


Le «home router» fonctionne alors comme un vrai routeur.

Il implémente aussi une fonction de **filtrage des ports**: la communication depuis l'extérieur n'est possible que si elle est autorisée par la configuration.

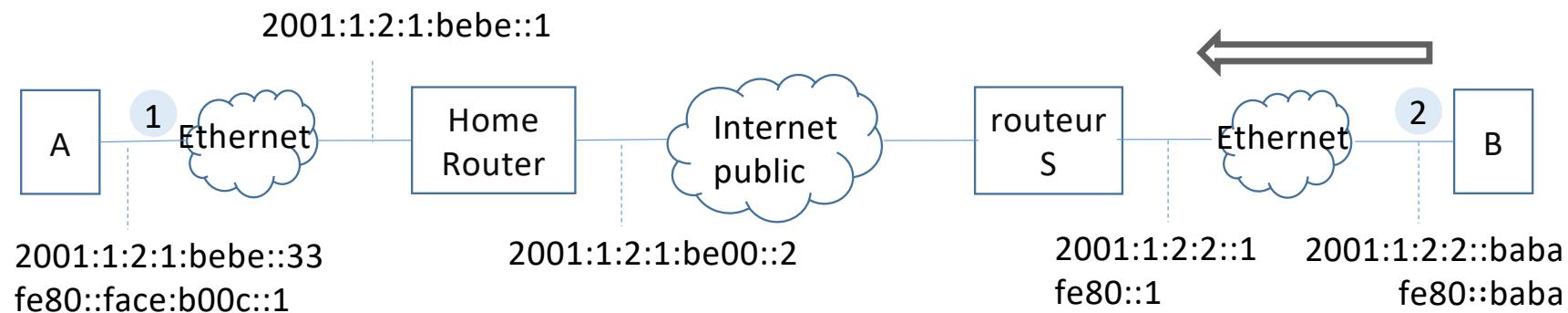
Dans les machines virtuelles, on utilise souvent un NAT IPv6.

B envoie des paquets IP à A. Quelle est l'adresse IP destination observée en (2) ?



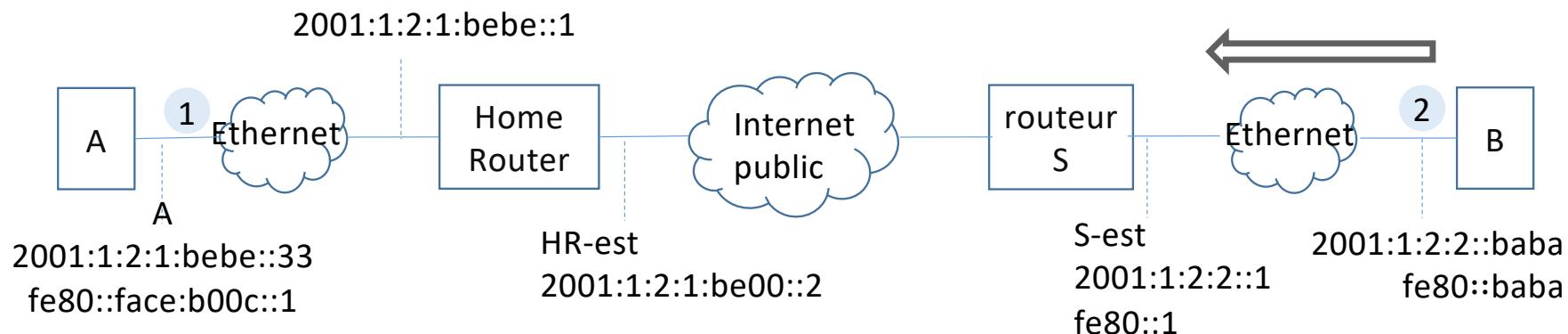
- A. 99.88.77.6
- B. 44.33.22.11
- C. 192.168.1.43
- D. Aucune de celles-ci
- E. Je ne sais pas

B envoie des paquets IP à A. Quelle est l'adresse IP destination observée en (2) ?



- A. 2001:1:2:2::1
- B. fe80::1
- C. 2001:1:2:1:be00::2
- D. 2001:1:2:1:bebe::33
- E. fe80::face:b00c::1
- F. Aucune de celles-ci
- G. Je ne sais pas

B envoie des paquets IP à A. Quelle est l'adresse MAC destination observée en (2) ?

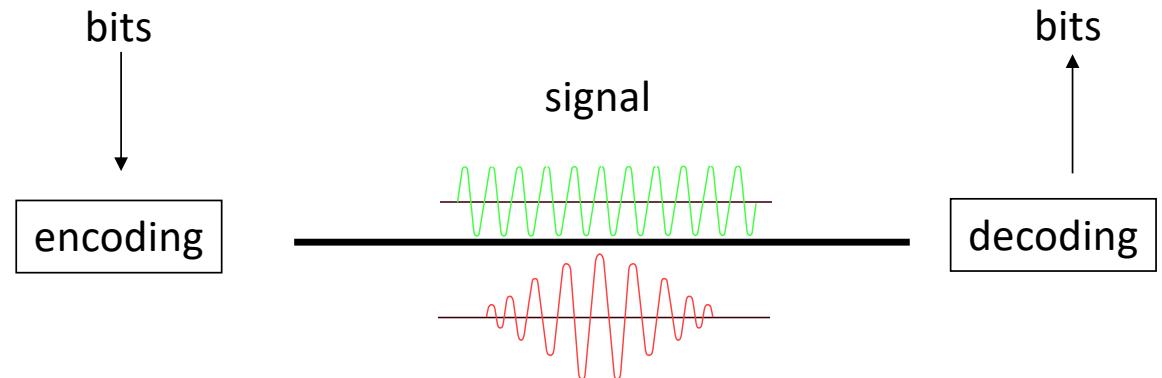


- A. l'adresse MAC (S-est) du routeur S
- B. l'adresse MAC (HR-est) du home-routeur
- C. l'adresse MAC de A
- D. Aucune de celles-ci
- E. Je ne sais pas

6. La couche physique

La couche physique transforme des blocs de bits (de la couche MAC) en signaux physiques, typiquement électromagnétiques.

Évolue avec la technologie: ex: Ethernet 10Mb/s à 1.6Tb/s.

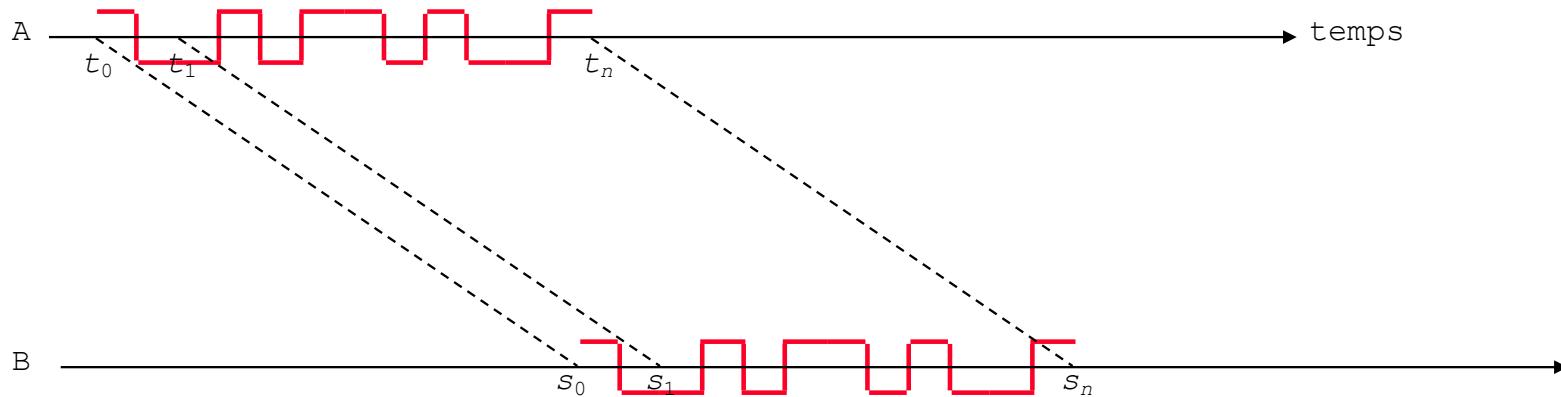


Le **débit binaire** est le nombre de bits transmis par seconde \neq **bande passante**. Pour un canal simple où les seules distorsions sont le bruit gaussien, la théorie de l'information dit que $C_{\max} = B \log_2(1 + \text{SNR})$, où B est la bande passante (en Hz), C_{\max} la borne supérieure de débit binaire atteignable en b/s et SNR le rapport signal-bruit.

Exemple: ligne ADSL de faible qualité: $B= 2 \text{ MHz}$, $\text{SNR}=50 \text{ dB}$, $C_{\max} = 33.2 \text{ Mb/s}$.

Essayez: <https://www.cnlab.ch/speedtest>

Propagation



Propagation: $s_n - t_n = s_0 - t_0$. Est donnée par la vitesse de la lumière dans le canal: dans le vide $c = 3E+08\text{m/s}$, dans le cuivre $c = 2.3E+08\text{ m/s}$, dans les fibres optiques en verre $c = 2eE08\text{ m/s}$ (tour de la terre en 200ms).

Transmission: $t_n - t_0$. Est égal au nombre de bits à transmettre divisé par le débit binaire du canal. Dépend de la technologie.

Délai total (unidirectionnel): $t_n - t_0 = \text{propagation} + \text{transmission}$

La commande **ping** mesure le délai bidirectionnel, y compris le traitement par la destination + files d'attentes dans le réseau

Quel temps faut-il pour transmettre 1kB (8000bits) ?

	<i>data center</i>	<i>ADSL</i>	<i>modem</i>	<i>Internet</i>
distance	20 m	2 km	20 km	20'000 km
débit binaire	1Tb/s	10Mb/s	10kb/s	1Mb/s
propagation	0.1 μ s	0.01ms	0.1ms	100ms
transmission	0.008 μ s	0.8ms	800ms	8ms
total	0.108 μ s	0.81ms	800.1ms	108ms

```
leboudec@icsil1noteb240 ~ % ping -c 3  
www.canterbury.ac.nz  
64 bytes from 132.181.106.9: icmp_seq=0 ttl=223 time=302.035 ms  
64 bytes from 132.181.106.9: icmp_seq=1 ttl=223 time=301.972 ms  
64 bytes from 132.181.106.9: icmp_seq=2 ttl=223 time=301.911 ms  
  
leboudec@icsil1noteb240 ~ % ping -c 3 www.nzherald.co.nz  
64 bytes from 195.176.255.75: icmp_seq=0 ttl=49 time=4.619 ms  
64 bytes from 195.176.255.75: icmp_seq=1 ttl=49 time=4.418 ms  
64 bytes from 195.176.255.75: icmp_seq=2 ttl=49 time=4.460 ms
```

7. La couche Transport

Rappel: la couche transport offre une interface de programmation aux applications. UDP offre le service de base, TCP ajoute la récupération des pertes.

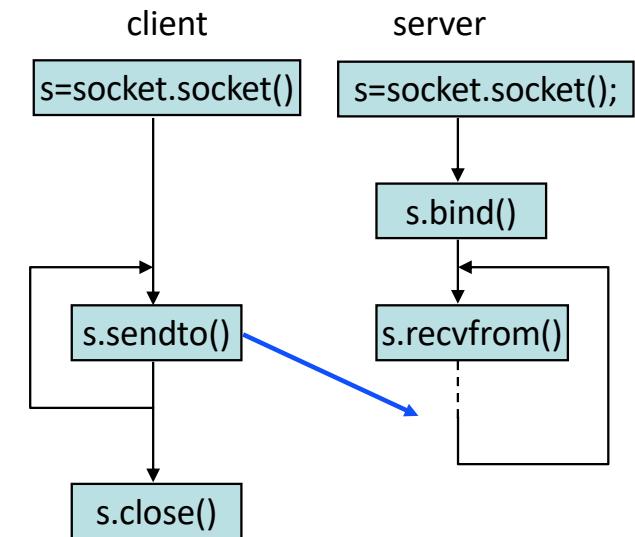
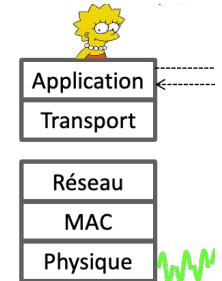
UDP et TCP sont accessibles à travers l'API «socket» en C, Python etc.

Exemple avec UDP: le client envoie un message, le serveur l'affiche.
`socket(socket.AF_INET,...)` crée une socket IPv4 socket et retourne un descripteur en cas de succès. `socket(socket.AF_INET6,...)` crée une socket IPv6.

`bind()` associe le port et l'adresse locale avec la socket – peut être omis pour un client;

`sendto()` envoie un message à une address IP et un numéro de port passés en argument;

`recvfrom()` bloque jusqu'à ce qu'un message arrive, et retourne alors le message, l'adresse IP de la source et le port source.



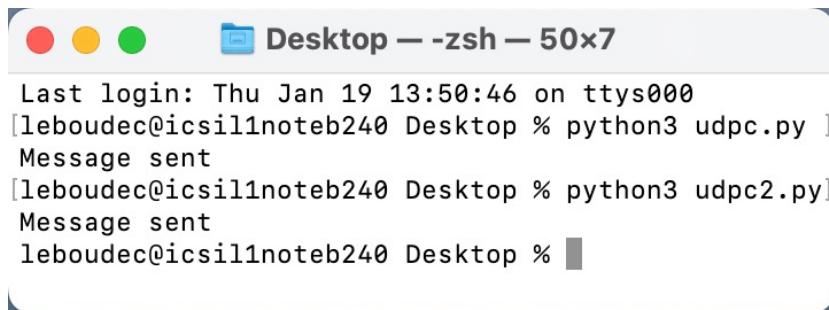
Essayez !

udpc.py

```
import socket
HOST = 'localhost' # The remote host
PORT = 50007 # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(b'Bonjour les amis', (HOST,PORT))
s.close()
print('Message sent')
```

udpc2.py

```
import socket
HOST = 'localhost' # The remote host
PORT = 50007 # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(b'Hoi zamme', (HOST,PORT))
s.close()
print('Message sent')
```

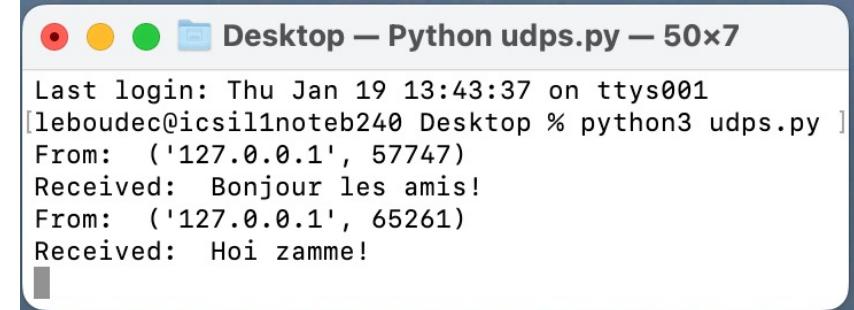


Desktop — zsh — 50x7

```
Last login: Thu Jan 19 13:50:46 on ttys000
[leboudec@icsil1noteb240 Desktop % python3 udpc.py ]
Message sent
[leboudec@icsil1noteb240 Desktop % python3 udpc2.py]
Message sent
leboudec@icsil1noteb240 Desktop %
```

udps.py

```
import socket
HOST = "" # Symbolic name meaning all available interfaces
PORT = 50007 # Arbitrary non-privileged port
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind((HOST, PORT))
while True:
    data, addr = s.recvfrom(1024)
    print('From:', addr)
    print('Received:', data.decode('utf-8'))
```



Desktop — Python udps.py — 50x7

```
Last login: Thu Jan 19 13:43:37 on ttys001
[leboudec@icsil1noteb240 Desktop % python3 udps.py ]
From: ('127.0.0.1', 57747)
Received: Bonjour les amis!
From: ('127.0.0.1', 65261)
Received: Hoi zamme!
```

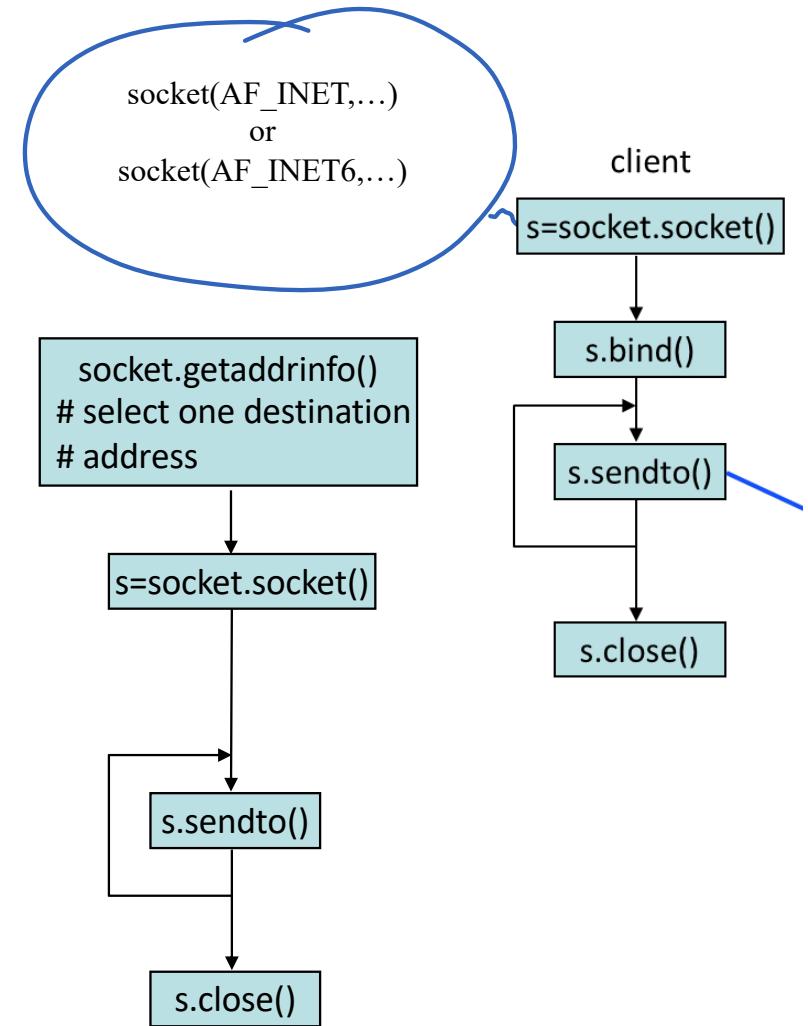
Y-a t'il un UDPv6 ?

Non, La couche transport est la même pour IPv4 et IPv6. Mais une socket doit savoir si elle manipule des adresses IPv4 ou IPv6.

C'est l'application qui choisit IPv4 ou IPv6 en utilisant DNS.

Essayez:

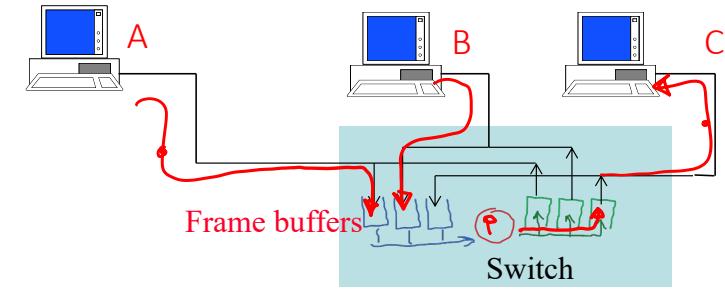
```
leboudec@lca2-2111021 ~ % python3
>>> import socket
>>> socket.getaddrinfo("lca.epfl.ch",None)
[(<AddressFamily.AF_INET6: 23>, 0, 0, '',
('2001:620:618:521:1:80b3:2127:1', 0, 0, 0)),
(<AddressFamily.AF_INET: 2>, 0, 0, ('128.179.33.39', 0))]
```



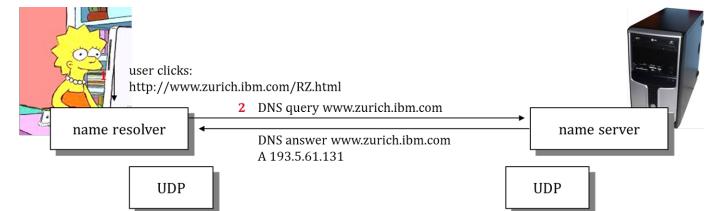
TCP

Les paquets IP peuvent être perdus en chemin:

- **erreurs de transmission** (rares sur câbles, fréquentes en mobile)
- **congestion**: débordement de buffer dans les switches et les routeurs (normal)



Les applications UDP doivent gérer les pertes.

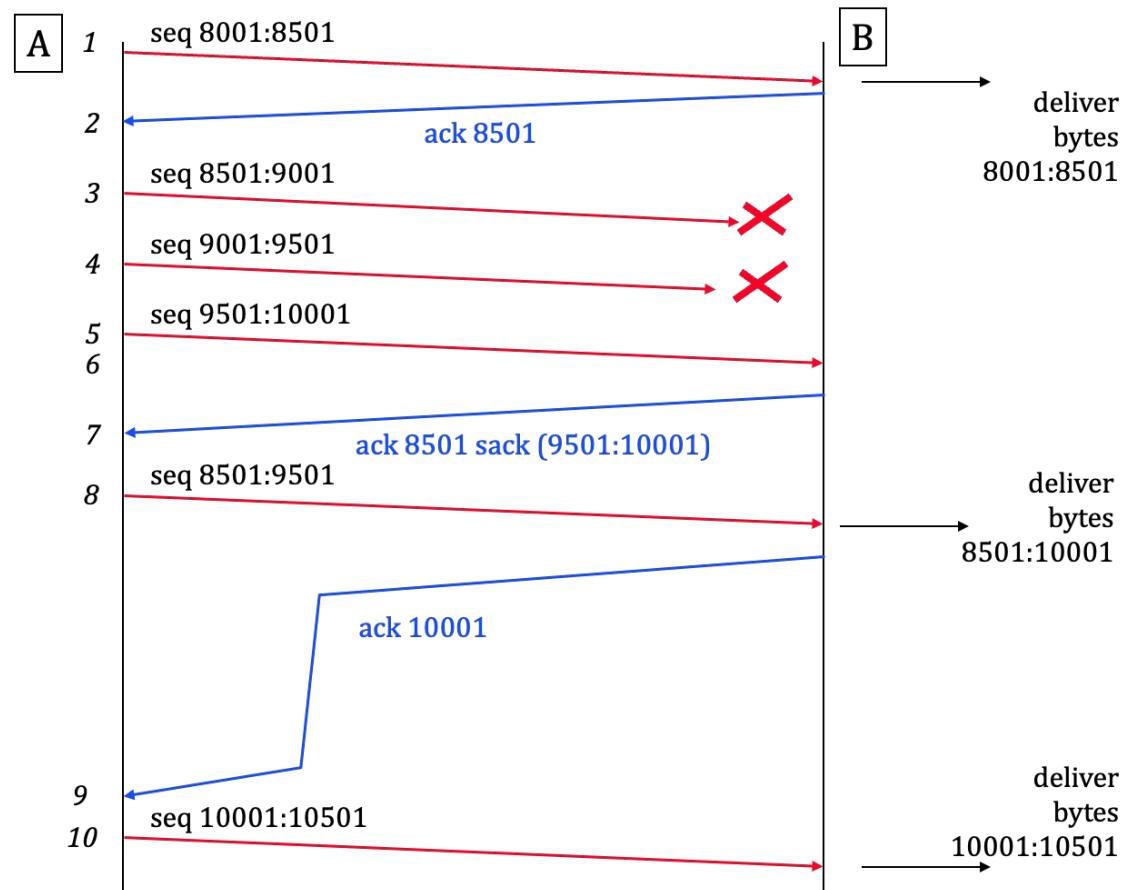


L'application UDP «name resolver» attend une réponse et essaie de nouveau si aucune réponse n'arrive avant un certain délai.

TCP gère les pertes une fois pour toutes.

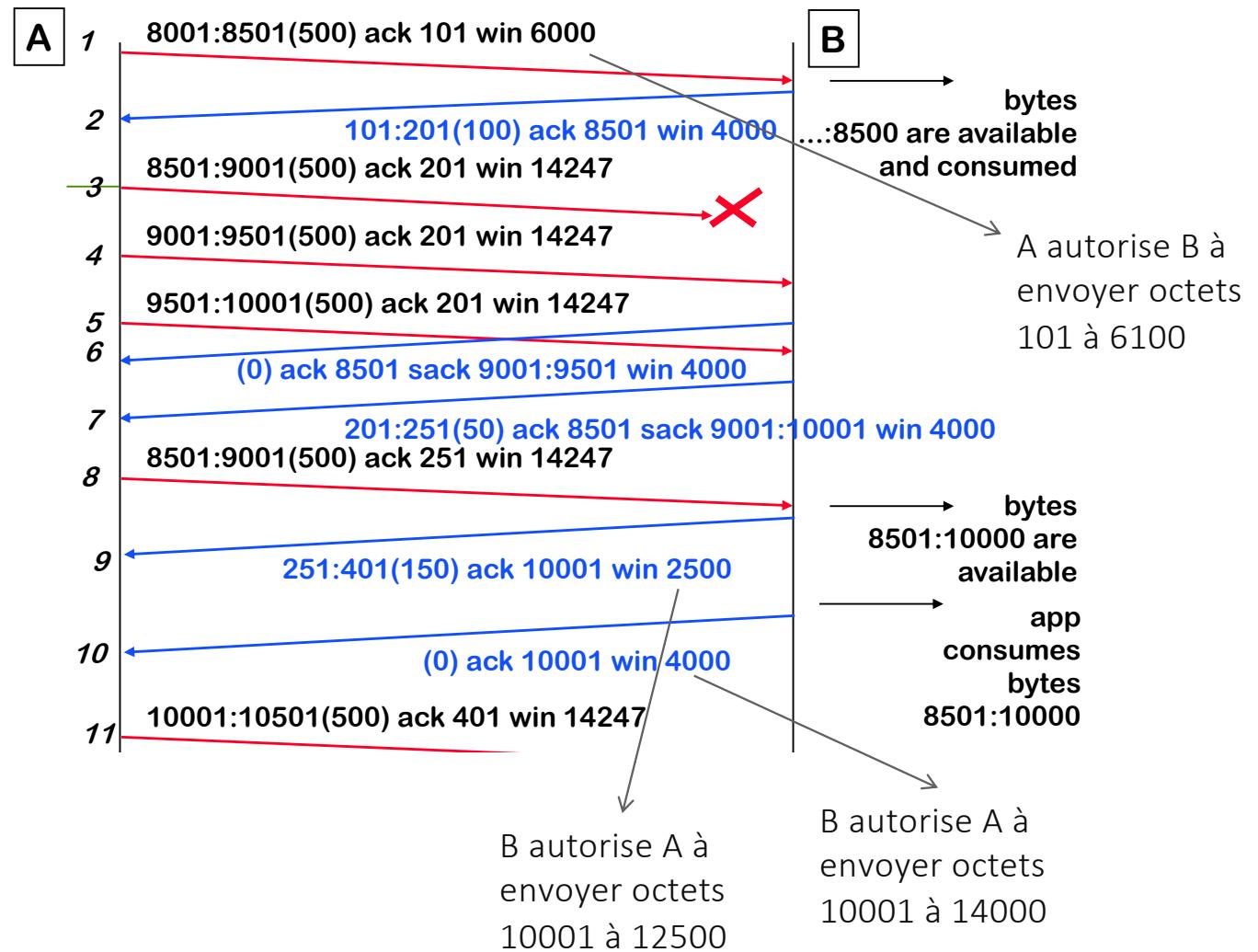
Opération de TCP

- les données sont numérotées (par octet)
- TCP-source attend un **acquittement** de chaque octet envoyé, détecte les pertes et retransmet si nécessaire.
- TCP-destination stocke les données reçues dans un buffer et ne les délivre que s'il n'y a pas de trou dans la séquence.



TCP utilise une fenêtre coulissante (“sliding window”)

- la fenêtre coulissante limite le nombre d'octets “en vol”, pour éviter que le buffer de destination déborde
- TCP est bidirectionnel, A et B sont à la fois sources et destinations



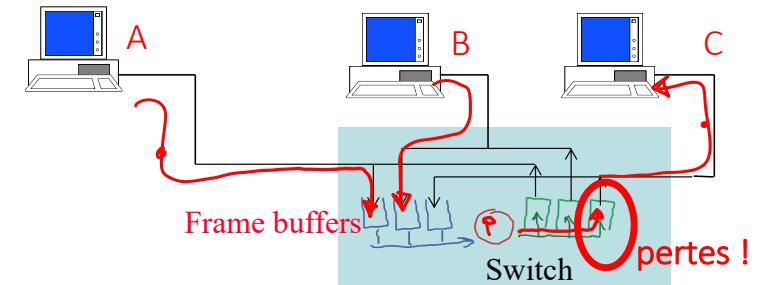
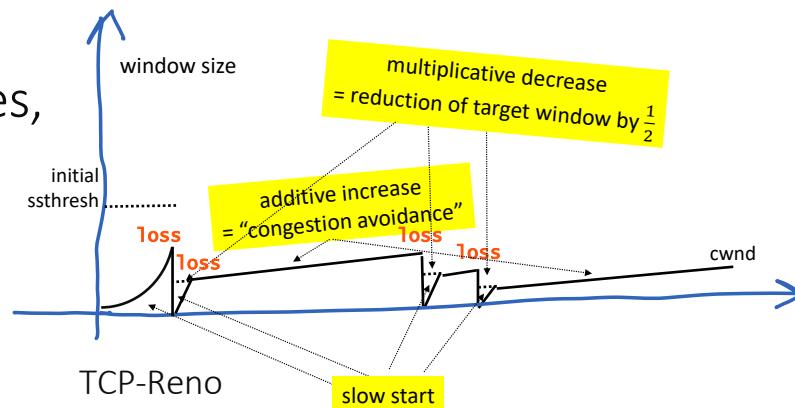
Contrôle de Congestion

TCP est aussi utilisé pour régler le trafic dans le réseau (contrôle de congestion). La taille de la fenêtre utilisée par TCP est $\min(\text{win}, \text{cwnd})$ où win est la fenêtre annoncée par TCP-destination et cwnd est calculée par l'algorithme de contrôle de congestion de TCP-source.

Principes:

- quand TCP détecte une perte, cwnd diminue
- quand TCP reçoit des acquittements, cwnd augmente

Il y a différents algorithmes, les plus répandus sont *Reno* et *Cubic*.



A et B envoient des données à C à 100 Mb/s chacun. Sans contrôle de congestion (e.g. avec UDP), il y a 50% de pertes de paquets au buffer de sortie du switch (200 Mb/s essaient de sortir à 100 Mb/s). Si A et B utilisent TCP, le contrôle de congestion réagit aux pertes dans le buffer de sortie du switch et limite A et B à environ 50 Mb/s. Il y a des pertes de temps en temps car le control de congestion de TCP a besoin d'un signal de pertes.

Sockets TCP

Avant d'envoyer des données par TCP, il faut synchroniser les compteurs en envoyant des paquets SYN, SYN-ACK et ACK qui ne contiennent pas de données.

L'échange de données commence en 1/2

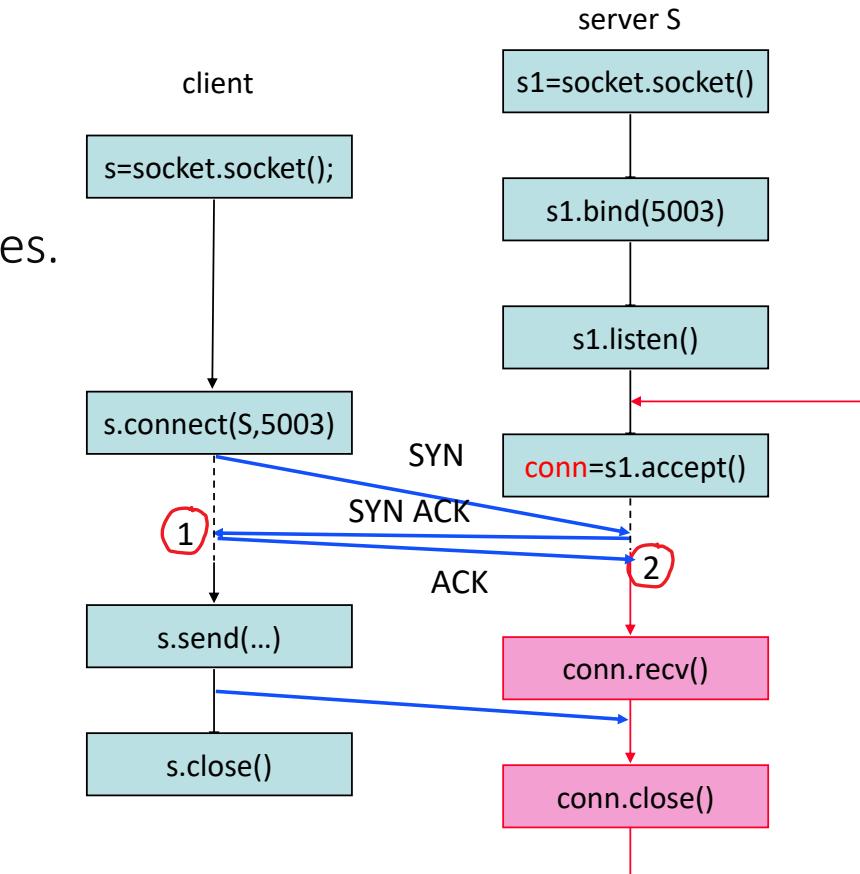
Exemple: le client envoie un message, le serveur l'affiche.

`connect()` associe la socket à l'adresse IP et au numéro de port du serveur, envoie le paquet SYN, et bloque jusqu'à réception de SYN-ACK

`send()` envoie un bloc de données (les données seront effectivement envoyées quand la fenêtre coulissante le permet)
`listen()` prépare le buffer de réception de paquets SYN

`accept()` bloque jusqu'à ce qu'un paquet SYN arrive, et crée une nouvelle socket conn sur laquelle les données seront échangées. La socket conn comprend l'adresse IP et le numéro de port du client (elle est *connectée*).

`read()` bloque jusqu'à ce que des données soient disponibles en réception, i.e. ont été reçues en séquence, et alors les délivre.



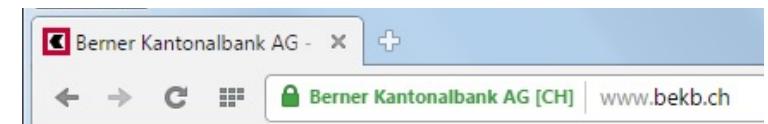
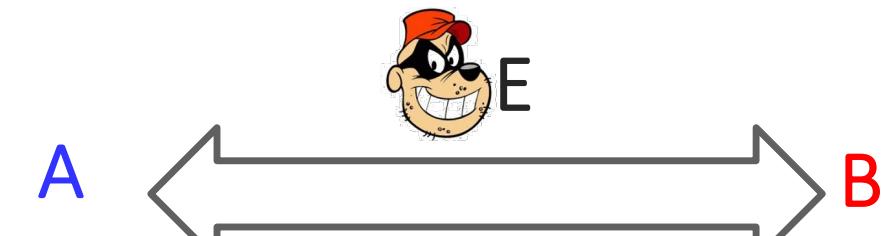
8. Le transport sécurisé

Toute communication dans un réseau doit être sécurisée. En particulier, il doit être impossible pour un ennemi de

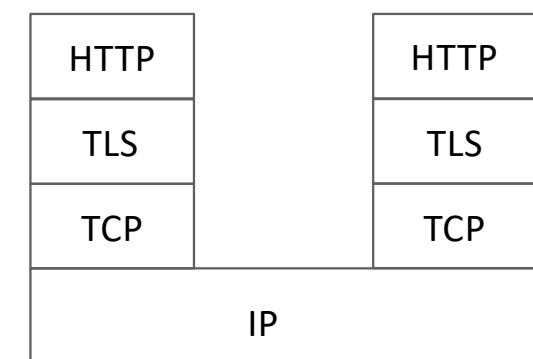
- lire le message envoyé
- modifier le message

TCP (comme UDP) ne fournit aucune sécurité et doit être complémenté par e.g. TLS.

Il existe une version alternative (QUIC) qui utilise UDP et réimplémente les fonctions de TCP dans le transport sécurisé.



http used over TLS and port 443 = https



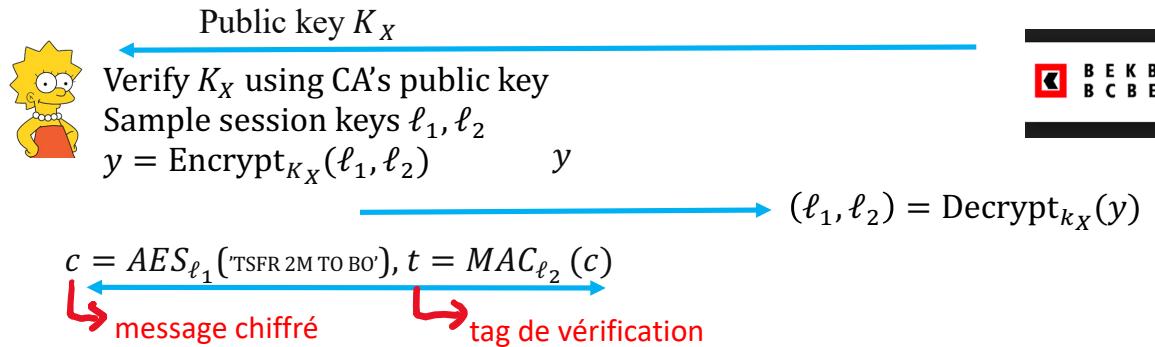
Cryptographie

La cryptographie **symétrique** (e.g. AES) utilise une clé secrète connue seulement de l'émetteur et du récepteur, et il est pratiquement impossible de décrypter le message sans connaître la clé. Peut aussi permettre de vérifier l'intégrité d'un message en ajoutant un tag de vérification calculé avec la clé secrète. Est très rapide mais la distribution des clés est un problème.

La cryptographie **asymétrique** (e.g RSA) utilise deux clés, une pour le chiffrement, l'autre pour le déchiffrement. Une des clés est secrète (privée), l'autre est publique. Il est pratiquement impossible de calculer la clé privée à partir de la clé publique. Le chiffrement et le déchiffrement sont assez lents.

Les clés doivent être **assez grandes** pour que des attaques par recherche systématique soient impossibles (e.g. 256 bits pour AES, 2048 bits pour RSA).

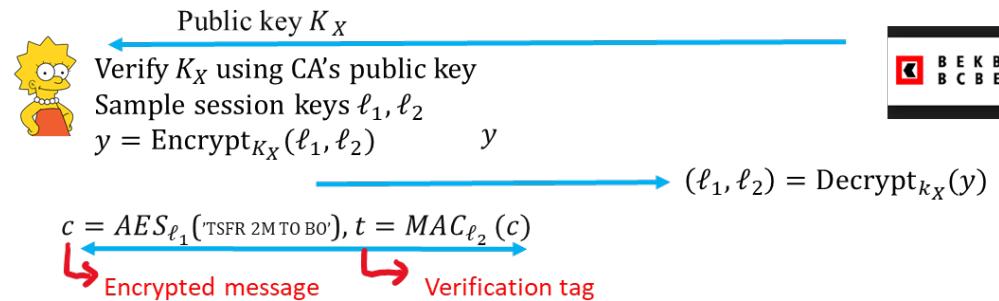
TLS combine cryptographies symétrique et asymétrique



1. Lisa vérifie que K_X est bien la clé publique de la banque en utilisant un **certificat**, qui contient la clé publique de la banque signée par une Autorité de Certification (CA). Lisa vérifie la signature du certificat en utilisant la **clé publique de CA**, préinstallée dans son navigateur web.
2. Lisa crée des clés de session secrètes qui permettront de chiffrer et authentifier la communication.

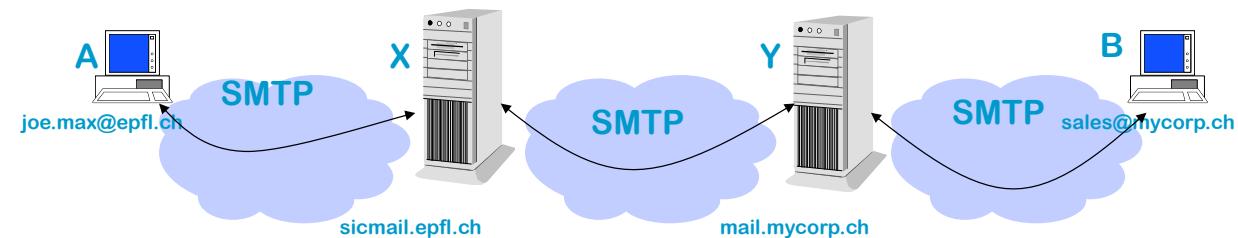
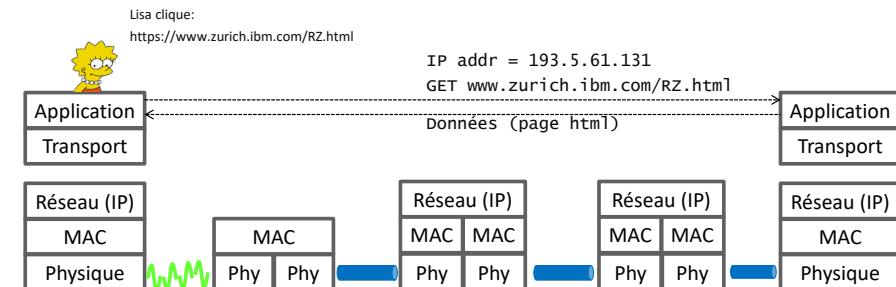
Le navigateur de Lisa utilise le certificat de la BCBE pour obtenir...

- A. La clé publique de la BCBE
- B. Le nom official de la BCBE tel qu'il apparaît au Registre du Commerce
- C. La clé publique de l'autorité de certification(CA) qui a émis le certificat de la BCBE
- D. A et B
- E. A et C
- F. B et C
- G. Tous
- H. Aucun
- I. Je ne sais pas



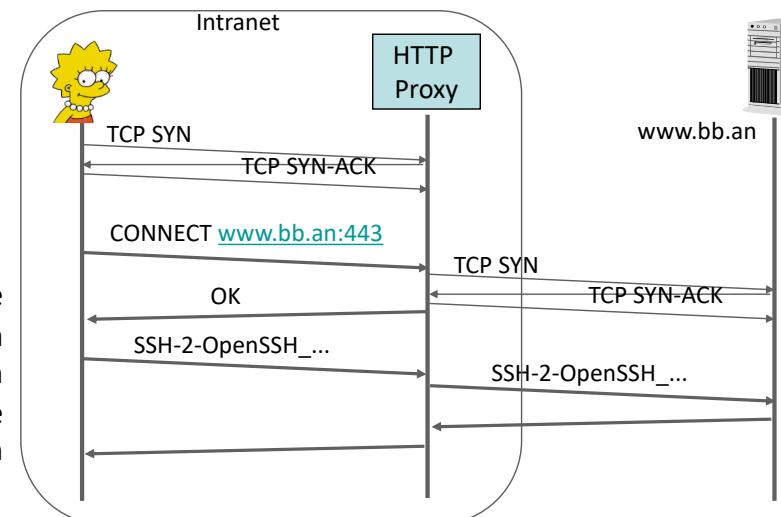
9. La couche application

Principe de bout-en-bout («**end-to-end principle**»): les applications sont dans les systèmes d'extrême pas dans les systèmes intermédiaires.



Exceptions pour mobilité, sécurité, performance, capture des clients, surveillance de la population.

Le “tunnel HTTP” oblige les clients de l’Intranet à établir une connexion TCP avec le proxy. Le proxy décide ou non d’autoriser la connexion.



Conclusion

L'internet est basé sur le modèle en couche et quelques principes simples

Les mêmes technologies sont aussi utilisées dans des réseaux industriels (réseau électrique, usine, avion, voiture, ...)

Des outils simples et puissants permettent d'explorer ce monde fascinant !

