Paramètres réseau

Un réseau privé est accessible par connexion filaire (connecter un câble RJ45 au *switch*) ou sans fil :

SSID : **`hex`**
Mot de passe : **`JohnNash`**

Une fois connecté, un site web est accessible à l'adresse **`http://hexbox.lan/`**

Programmation **réseau** en Python

formation **ISN** — Orléans, 8 juin 2016

Qui sommes-nous ?

Deux intervenants :

- ➡ Mathieu Liedloff, MdC section 27
- ➡ Nicolas Ollinger, PU section 27

Nous enseignons les réseaux aux étudiants de L3 et M1 informatique à l'université d'Orléans.

Contact : **`prénom.nom@univ-orleans.fr`**

BO spécial n°8 du 13 octobre 2011

Savoirs	Capacités
Transmission point à point Principes de base d'une transmission d'informations numériques entre un émetteur et un récepteur.	Établir une communication sérielle entre deux machines.
Adressage sur un réseau Mécanismes d'adressage pour identifier des machines distantes.	**Décrire** une situation d'adressage sur un type de réseau particulier. **Analyser** le trafic (trames) sur un réseau et mettre ainsi en évidence la notion de protocole.
Routage Mécanismes induits par la communication sur un réseau dont la structure est de type graphe. Notion de paquets, de chemins, de routage.	**Analyser** les entêtes de messages électroniques, pour décrire le chemin suivi par l'information.
Supranationalité des réseaux	Prendre conscience du caractère supranational des réseaux et des conséquences sociales, économiques et politiques qui en découlent.

Ce matin

« Comment faire communiquer nos programmes à travers un réseau d'ordinateurs ? »

1. **observation d'un réseau**
2. **un peu de théorie**
3. **programmation réseau en pratique**

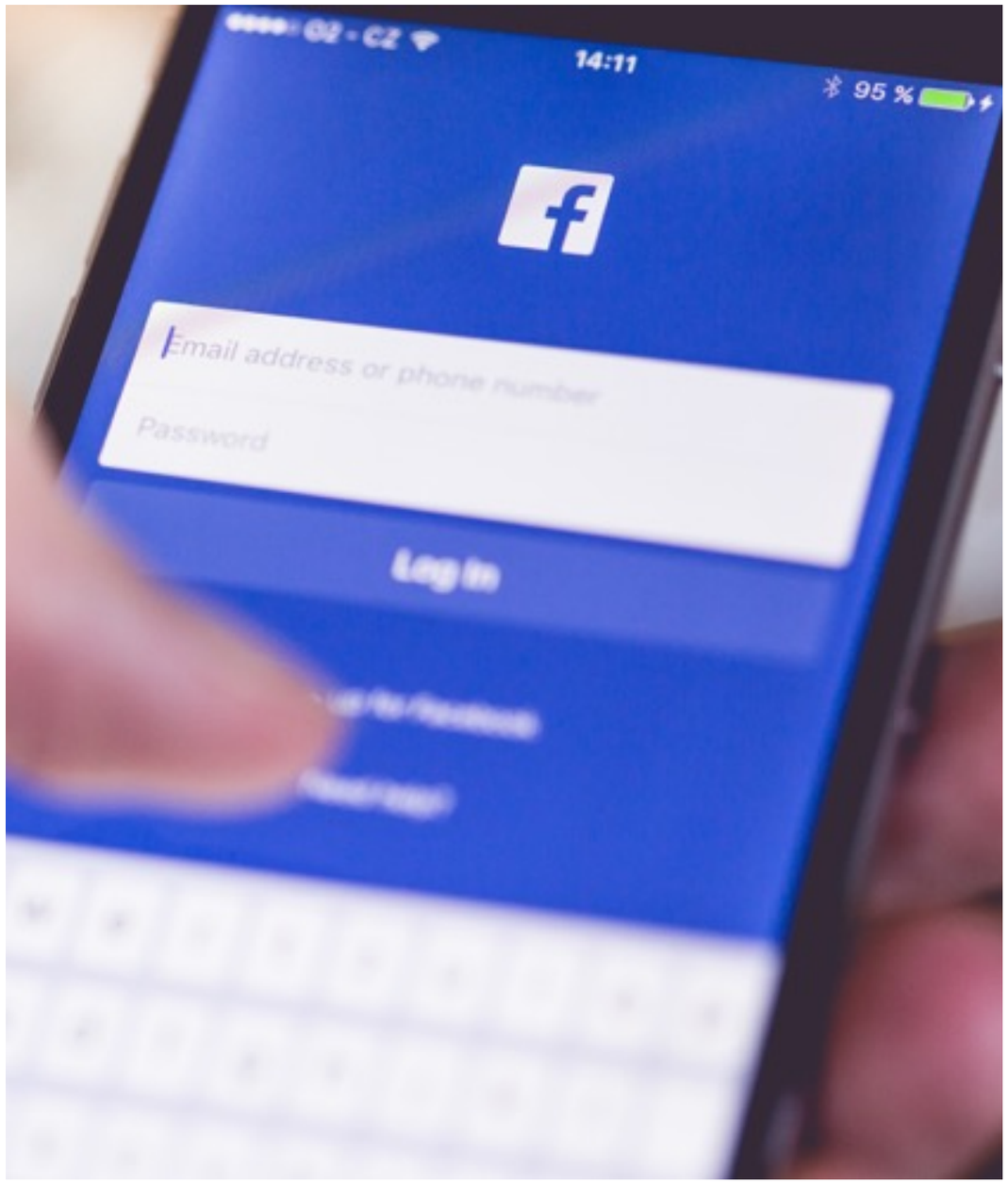
On traite ici uniquement de réseaux TCP/IPv4.

1. observation

Alice veut consulter une page web

- ➡ connecter son terminal au réseau (Ethernet/Wifi)
- ➡ ouvrir un navigateur web
- ➡ saisir l'**URL** de la page

Le navigateur web se **connecte** au serveur web correspondant à l'URL et récupère la page web puis l'affiche.



Vous Etes Perdu ?

http://www.perdu.com/

Rechercher

Perdu sur l'Internet ?

Pas de panique, on va vous aider

```
* <----- vous êtes ici
```

Vous Etes Perdu ?

http://www.perdu.com/

Rechercher

Perdu sur l'Internet ?

Pas de panique, on va vous aider

```
* <----- vous êtes ici
```

Inspecteur | Console | Débogueur | { } Éditeur de st... | Performances | Réseau

html > body > h1

```
<html>
  <head>
    <title>Vous Etes Perdu ?</title>
  </head>
  <body>
    <h1>Perdu sur l'Internet ?</h1>
    <h2>Pas de panique, on va vous aider</h2>
    <strong>
      <pre>* <----- vous êtes ici</pre>
    </strong>
  </body>
</html>
```

Vous Etes Perdu ?

http://www.perdu.com/

Rechercher

Perdu sur l'Internet ?

Pas de panique, on va vous aider

```
* <----- vous êtes ici
```

Inspecteur | Console | Débogueur | { } Éditeur de st... | Performances | Réseau

En-têtes | Cookies | Paramètres | Réponse | Délais | Aperçu

200

URL de la requête : http://www.perdu.com/
Méthode de la requête : GET
Adresse distante : 208.97.177.124:80
Code d'état : 200 OK
Version : HTTP/1.1

Modifier et renvoyer | En-têtes bruts

En-têtes de requête :

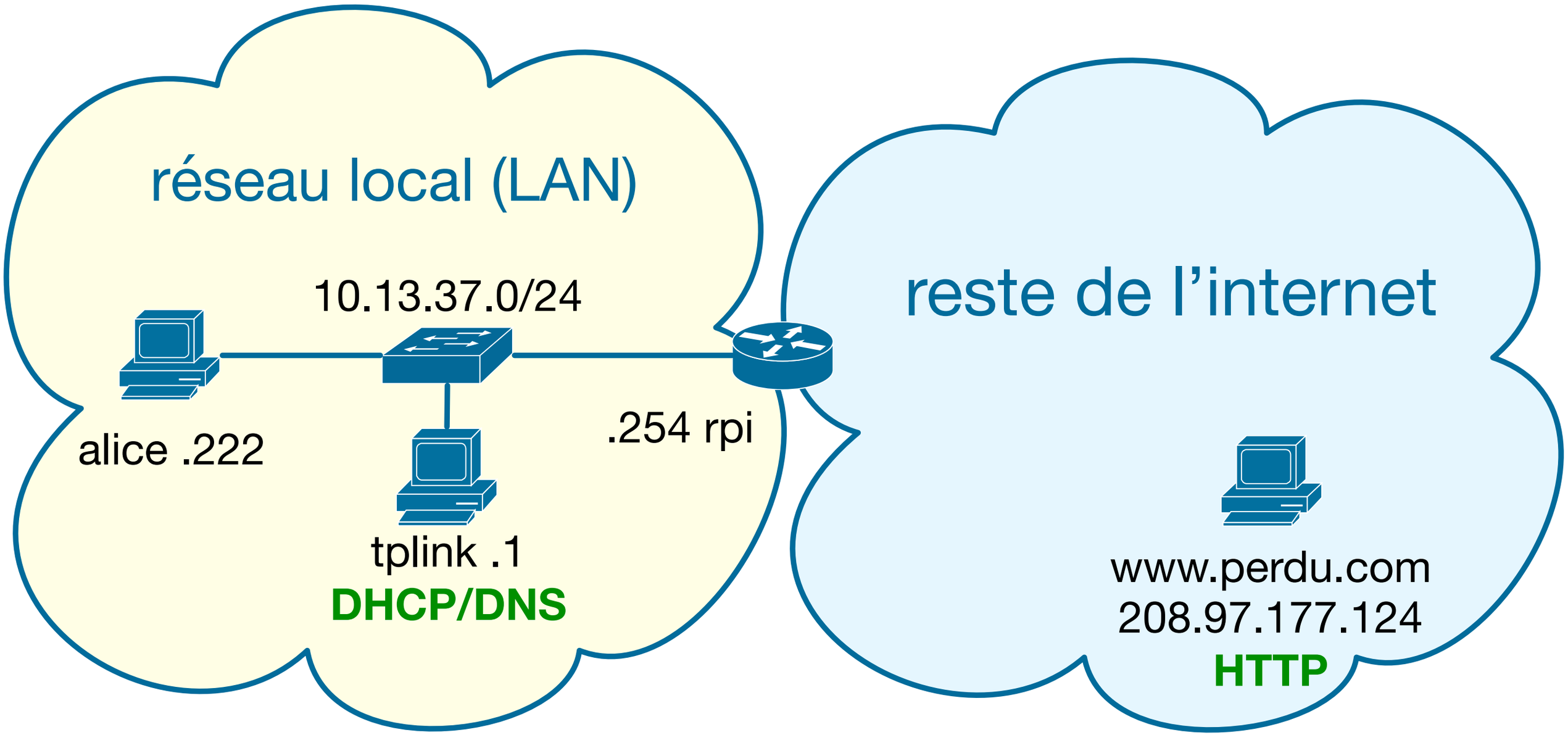
Host: www.perdu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:46.0)
Gecko/20100101 Firefox/46.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

En-têtes de réponse :

Accept-Ranges: bytes
Connection: keep-alive
Content-Length: 204
Content-Type: text/html
Date: Sat, 04 Jun 2016 18:18:01 GMT
Etag: "5752d6dd-cc"
Last-Modified: Sat, 04 Jun 2016 13:25:49 GMT
Server: nginx/1.6.2

```
$ curl -v http://www.perdu.com/
*   Trying 208.97.177.124...
* Connected to www.perdu.com (208.97.177.124) port 80
>>> GET / HTTP/1.1
>>> Host: www.perdu.com
>>> User-Agent: curl/7.47.1
>>> Accept: */*
>>>
<<< HTTP/1.1 200 OK
<<< Server: nginx/1.6.2
<<< Date: Sat, 04 Jun 2016 18:17:25 GMT
<<< Content-Type: text/html
<<< Content-Length: 204
<<< Last-Modified: Sat, 04 Jun 2016 13:25:49 GMT
<<< Connection: keep-alive
<<< ETag: "5752d6dd-cc"
<<< Accept-Ranges: bytes
<<< <html><head><title>Vous Etes Perdu ?</title></
head><body><h1>Perdu sur l'Internet ?</h1><h2>Pas de panique,
on va vous aider</h2><strong><pre>    * <----- vous &ecirc;tes
ici</pre></strong></body></html>
```

GET / HTTP/1.1
Host: www.perdu.com
@IP 208.97.177.124
port TCP 80
2625 E Ojai Dr
HTTP/1.1 200 OK
(...)
Content-Type: text/html
Content-Length: 204
<html><head><title>Vous Etes Perdu ?</title></hea
Lycée Général et Technologique Voltaire
Canada
États-Unis
Mexique
Groenland
Islande
Norvège
Suède
Finlande
Royaume-Uni
Irlande
Danemark
Allemagne
Pologne
Biélorussie
Ukraine
Italie
Grèce
Turquie
Syrie
Irak
Maroc
Algérie
Libye
Égypte
Tunisie
Mauritanie
Mali
Niger
Tchad
Soudan
Nigeria
Éthiopie
Somalie
Kenya
Tanzanie
Angola
Zambie
Mozambique
Namibie
Zimbabwe
Botswana
Afrique du Sud
Madagascar
Brésil
Pérou
Bolivie
Paraguay
Chili
Uruguay
Argentine
Colombie
Venezuela
Équateur
Guatemala
Nicaragua
Cuba
Porto Rico
Mer de Norvège
Northwestern Passages
Océan Atlantique Nord
Océan Atlantique Sud
Océan Pacifique Sud
Mer Méditerranée
Golfe du Mexique
Mer des Caraïbes
Golfe de Guinée

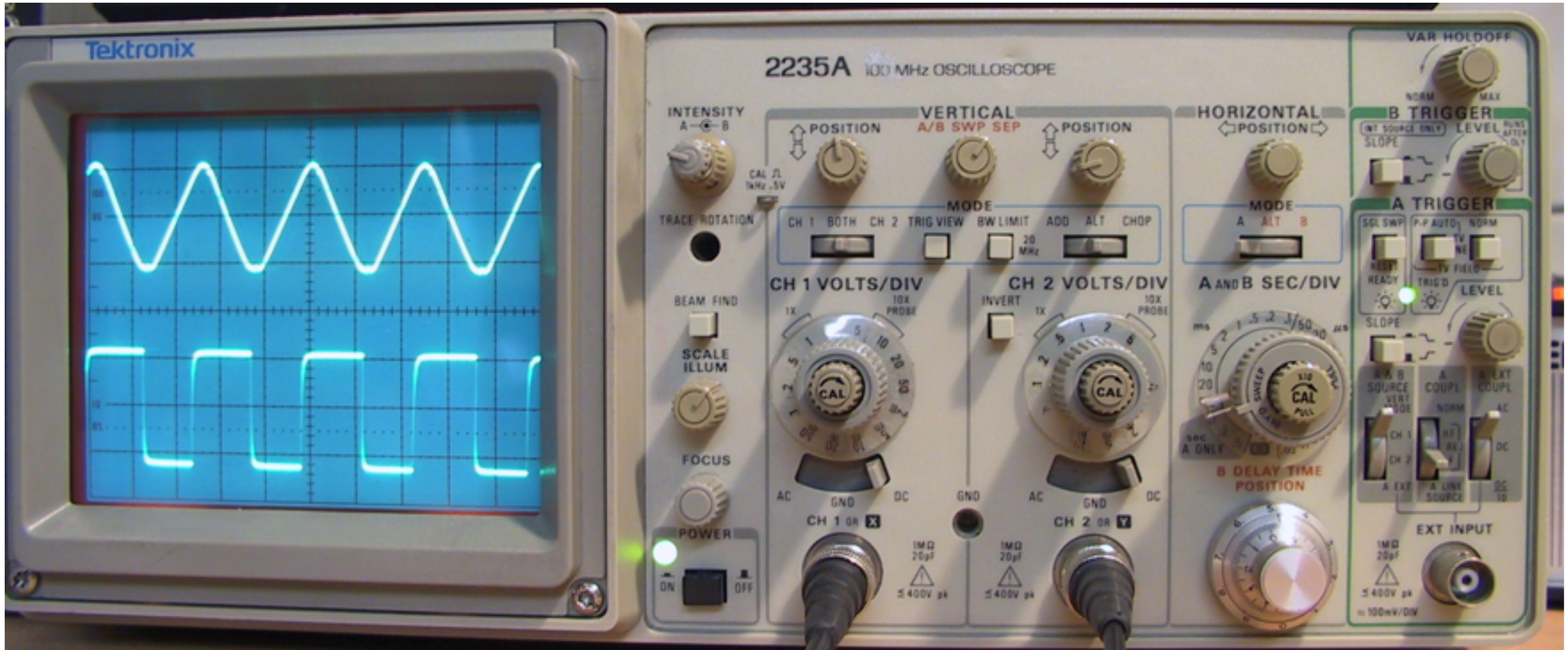


Observons ce qui se passe localement !

Analyser le trafic réseau

À l'aide d'un analyseur de trames, enregistrer et disséquer les échanges réseau visibles depuis une carte réseau.

Par exemple wire**shark** : **https://www.wireshark.org/**



trace.pcap

Appliquer un filtre d'affichage … <⌘/>

Expression… +

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xab4b6e1a
2	0.075880	10.13.37.1	10.13.37.222	DHCP	342	DHCP Offer - Transaction ID 0xab4b6e1a
3	1.106004	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0xab4b6e1a
4	1.212867	10.13.37.1	10.13.37.222	DHCP	342	DHCP ACK - Transaction ID 0xab4b6e1a
5	1.219279	Apple_98:33:f4	Broadcast	ARP	42	Who has 10.13.37.222? Tell 0.0.0.0
6	4.017568	Apple_98:33:f4	Broadcast	ARP	42	Who has 10.13.37.254? Tell 10.13.37.222
7	4.023757	Raspberr_bd:43:c4	Apple_98:33:f4	ARP	60	10.13.37.254 is at b8:27:eb:bd:43:c4
8	4.083233	Apple_98:33:f4	Broadcast	ARP	42	Who has 10.13.37.1? Tell 10.13.37.222
9	4.085006	Tp-LinkT_81:90:5b	Apple_98:33:f4	ARP	42	10.13.37.1 is at f4:f2:6d:81:90:5b
10	8.592285	10.13.37.222	10.13.37.1	DNS	73	Standard query 0x61b3 A www.perdu.com
11	8.600033	10.13.37.1	10.13.37.222	DNS	89	Standard query response 0x61b3 A www.perdu.com A…
12	9.548265	Raspberr_bd:43:c4	Apple_98:33:f4	ARP	60	Who has 10.13.37.222? Tell 10.13.37.254
13	9.548300	Apple_98:33:f4	Raspberr_bd:43:c4	ARP	42	10.13.37.222 is at 64:76:ba:98:33:f4
14	9.573067	Tp-LinkT_81:90:5b	Apple_98:33:f4	ARP	42	Who has 10.13.37.222? Tell 10.13.37.1
15	9.573139	Apple_98:33:f4	Tp-LinkT_81:90:5b	ARP	42	10.13.37.222 is at 64:76:ba:98:33:f4
16	9.668292	10.13.37.222	208.97.177.124	TCP	78	60087 → 80 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0…
17	9.670190	208.97.177.124	10.13.37.222	TCP	74	80 → 60087 [SYN, ACK, ECN] Seq=0 Ack=1 Win=28960…
18	9.670254	10.13.37.222	208.97.177.124	TCP	66	60087 → 80 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TS…
19	9.670467	10.13.37.222	208.97.177.124	HTTP	384	GET / HTTP/1.1
20	9.671843	208.97.177.124	10.13.37.222	TCP	66	80 → 60087 [ACK] Seq=1 Ack=319 Win=30080 Len=0 T…
21	9.680899	208.97.177.124	10.13.37.222	HTTP	506	HTTP/1.1 200 OK (text/html)
22	9.680937	10.13.37.222	208.97.177.124	TCP	66	60087 → 80 [ACK] Seq=319 Ack=441 Win=131328 Len=…
23	9.719810	10.13.37.222	208.97.177.124	HTTP	395	GET /favicon.ico HTTP/1.1
24	9.723680	208.97.177.124	10.13.37.222	HTTP	388	HTTP/1.1 404 Not Found (text/html)
25	9.723746	10.13.37.222	208.97.177.124	TCP	66	60087 → 80 [ACK] Seq=648 Ack=763 Win=130976 Len=…

```
▶ Frame 1: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits)
▶ Ethernet II, Src: Apple_98:33:f4 (64:76:ba:98:33:f4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▶ Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
▶ User Datagram Protocol, Src Port: 68 (68), Dst Port: 67 (67)
▶ Bootstrap Protocol (Discover)
```

```
0000  ff ff ff ff ff ff 64 76  ba 98 33 f4 08 00 45 00   ......dv ..3...E.
0010  01 48 8c ec 00 00 ff 11  2d b9 00 00 00 00 ff ff   .H...... -.......
0020  ff ff 00 44 00 43 01 34  48 ab 01 01 06 00 ab 4b   ...D.C.4 H......K
0030  6e 1a 00 02 00 00 00 00  00 00 00 00 00 00 00 00   n....... ........
0040  00 00 00 00 00 00 64 76  ba 98 33 f4 00 00 00 00   ......dv ..3.....
```

trace Paquets: 25 · Affichés: 25 (100.0%)· Temps de chargement: 0:0.1 Profil: Default

Trames Ethernet

- Toute interface réseau possède une adresse MAC
- **Encapsulation** de protocoles de plus haut niveau à l'intérieur des trames.
- Échange de trames en **unicast** d'un émetteur vers un récepteur.
- Échange de trames en **broadcast** d'un émetteur vers toutes les machines du même réseau local.

L’abstraction IP

- Toute interface réseau possède une **adresse IP** **`x.y.z.t`** (éventuellement locale)
- Statique ou dynamique (DHCP)
- routage vers le reste du réseau via une passerelle

```
$ ifconfig en0 10.13.37.222/24
$ route add default gw 10.13.37.254
```

- le protocole ARP fait la correspondance entre adresses MAC et IP

L’abstraction DNS

- Pour éviter de retenir des adresses numériques et permettre de migrer d’une machine à une autre, les machines sont nommées via le DNS

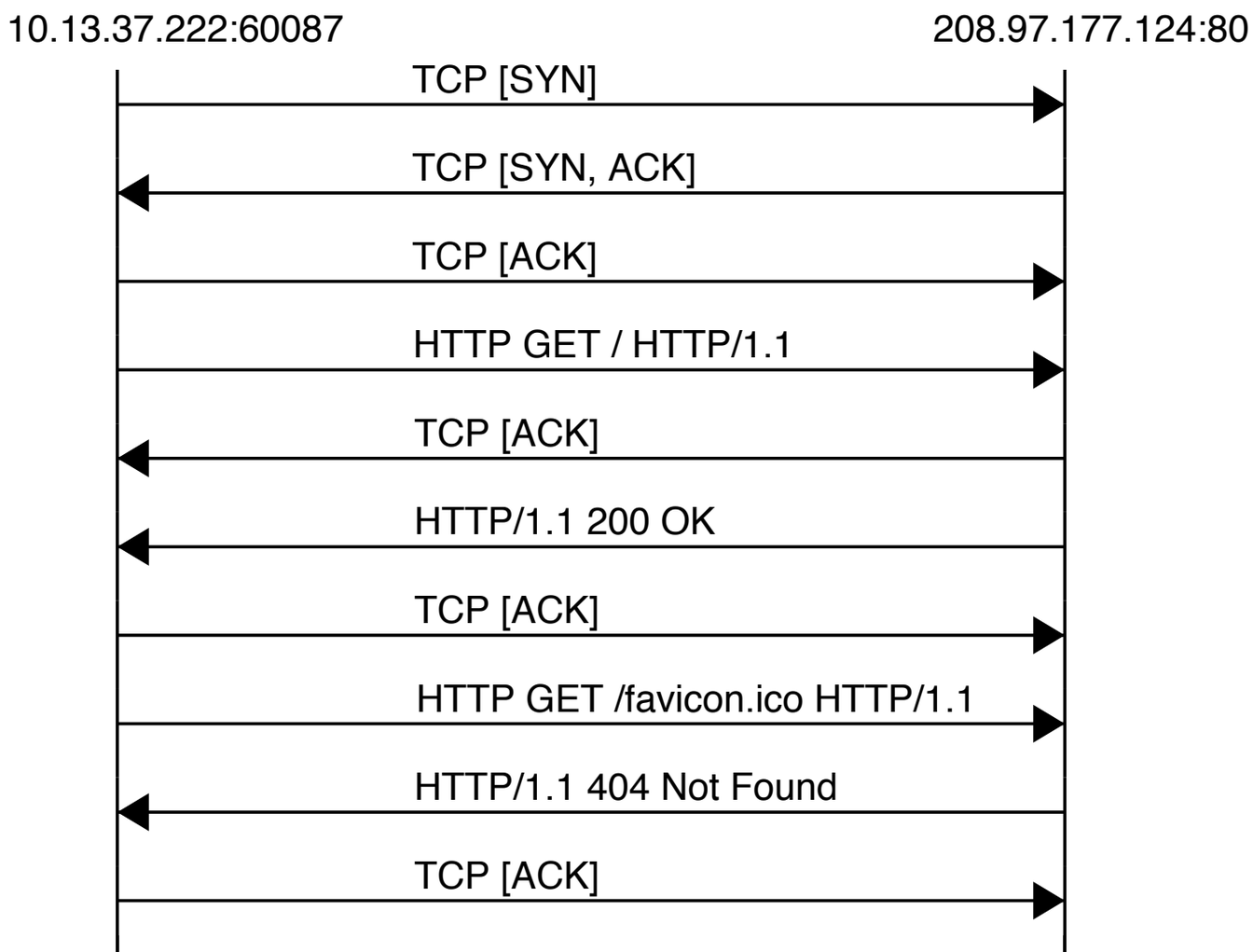
- On cache les adresses IP

```
$ cat > /etc/resolv.conf <<EOF
nameserver 10.13.37.1
EOF

$ dig +short www.perdu.com
208.97.177.124
```

L’abstraction TCP

- Modèle client/serveur
- Le serveur écoute sur un **port**
- Flux de données abstrait.



Le protocole DHCP

Paramètres pour configurer sa machine :

- **adresse IP** + **masque** de réseau
- adresse de la **passerelle**
- adresse du **serveur DNS**

Le protocole **DHCP** permet d'obtenir ces paramètres automatiquement à la connexion.

2. théorie

Qu'est-ce qu'un réseau ?

Connexion de **plusieurs entités** (machines) entre elles, afin qu'elles puissent **échanger des informations** selon un **protocole** convenu à l'avance.

Le terme réseau peut désigner selon le contexte :

- l'ensemble des machines ;
- l'infrastructure (media utilisé, protocole utilisé, etc)

Modèles à couches

La plupart des réseaux reposent sur un modèle à couches :

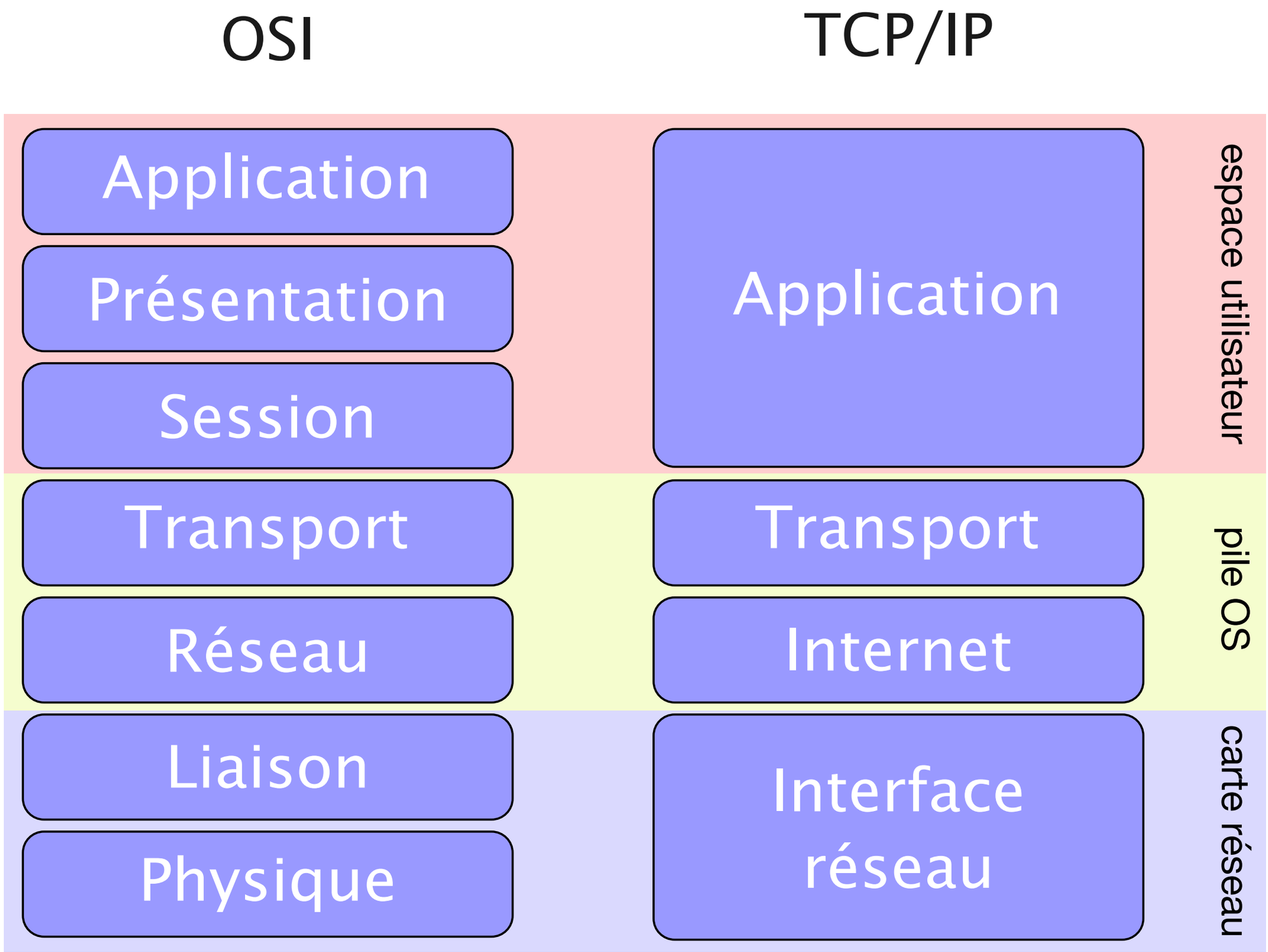
- Chaque couche ne peut parler qu'avec la couche immédiatement inférieure ;
- La couche de niveau *n* d'une machine dialogue avec la couche de niveau *n* d'une autre machine, par l'intermédiaire d'un **protocole** ;
- Seule la couche de niveau 1 d'une machine peut parler directement avec la couche de niveau 1 d'une autre machine ;
- Un protocole de niveau *n* sera donc **encapsulé** dans un protocole de niveau *n-1*.

Modèles de référence

Le modèle de référence **OSI** (Open Systems Interconnection, ISO 7498, 1984) est un modèle théorique. Ce n'est **pas** une norme.

Le modèle **TCP/IP** de l'internet (**RFC 1122**, 1989, utilisé dès les années 1970). C'est un modèle réel et le standard.

Modèles de référence



Couche physique (physical layer)

Transmission d'une suite de bits **d'un point A à un point B** reliés par un medium (câble, atmosphère)

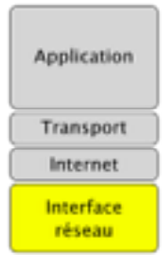
Liaison **point-à-point** entre deux extrémités :

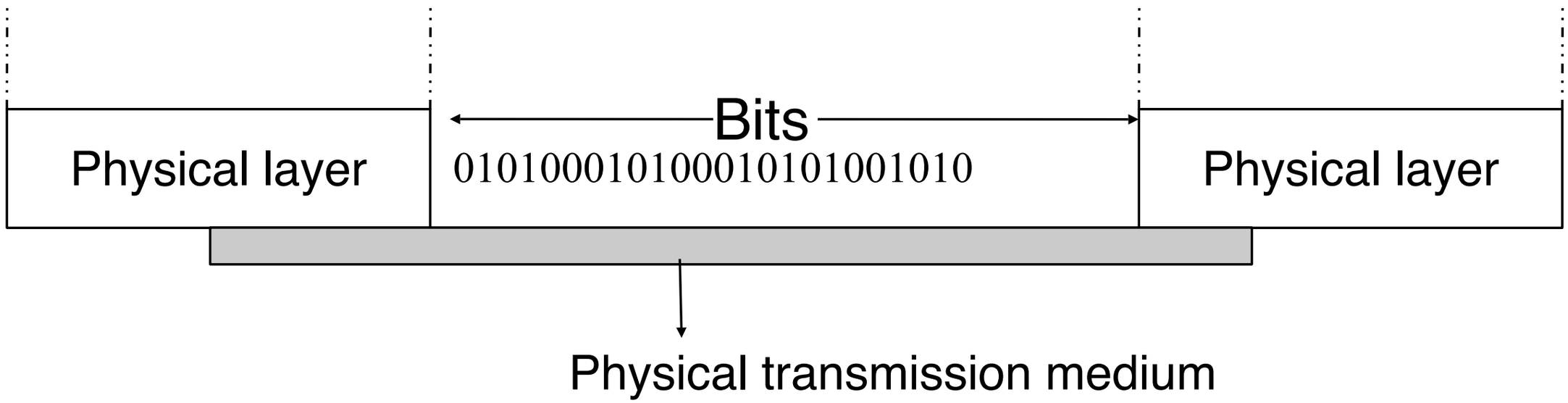
- quand un équipement émet des données, elles sont destinées à l'autre extrémité ;
- quand il en reçoit, elles viennent de l'autre extrémité.

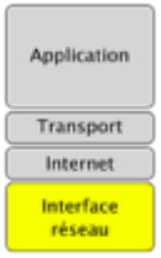
Potentiellement des **erreurs** de transmission (parasite) ou de décodage.

paire de cuivre

Couche physique (physical layer)







Couche liaison (data link layer)

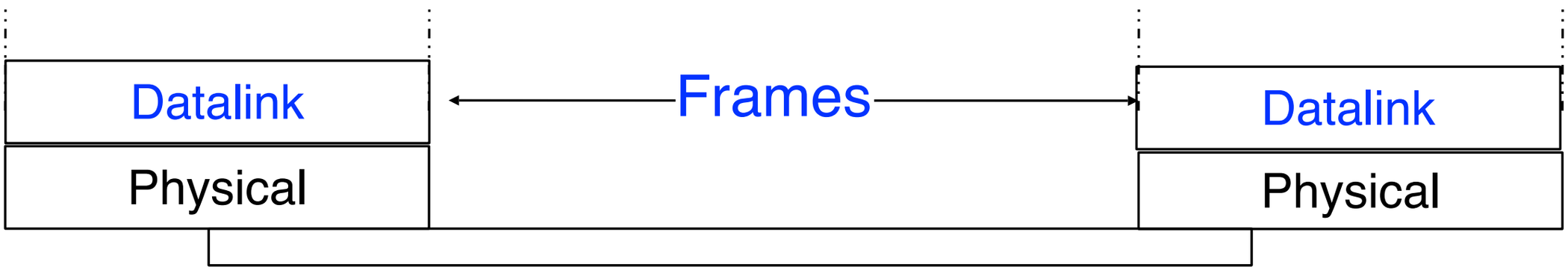
Transmission de **trames** d'un point A à un point B reliés par un medium :

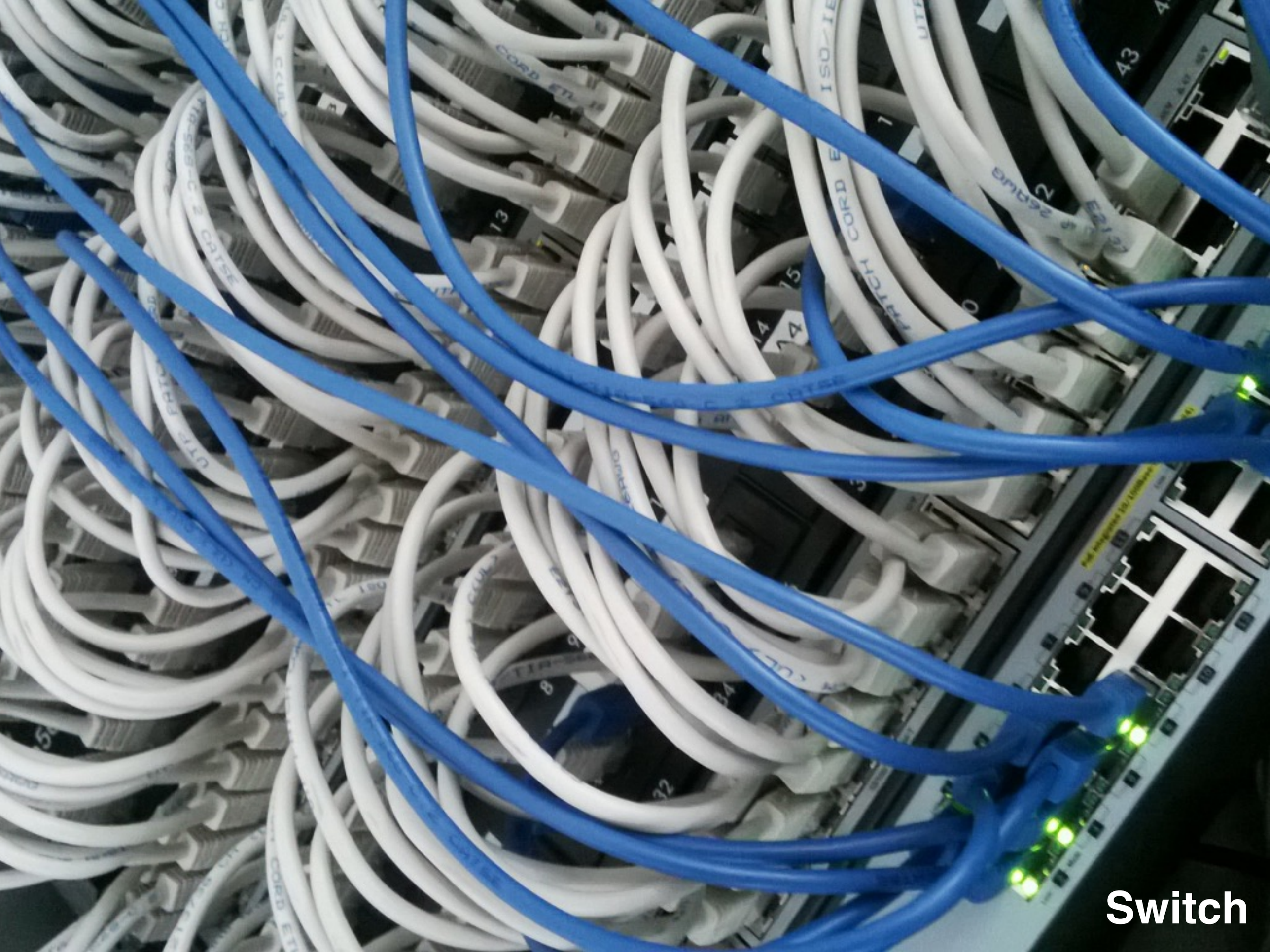
- sans erreurs ;
- sans duplication.

Toujours **point-à-point** ou **réseau de diffusion** avec gestion d'**accès au canal**.

Ethernet

Couche liaison (data link layer)





Switch

Trame Ethernet

Flag	Destination	Source	Type	Données	CRC
8	6	6	2	46-1500	4

Une trame Ethernet transporte au maximum 1500 octets de charge utile (Maximum Transfer Unit, **MTU**)

Au minimum 46 octets de charge utile !

Taille maximum de la trame : 1518 octets (+ flag)

CRC **`0x04C11DB7`**

Adresse MAC

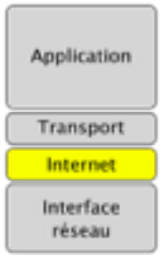
- Adresse codée sur 48 bits
- Par exemple **`64:76:ba:98:33:f4`**
- Seulement 2^{46} adresses potentielles car un bit est réservé pour local/global et un bit pour le multicast.
- Certaines adresses ont une signification précise : ainsi **`ff:ff:ff:ff:ff:ff`** est l'adresse de broadcast.

```
0000  ff ff ff ff ff ff f2 1b 6e ba 74 06 08 06 00 01
0010  08 00 06 04 00 01 f2 1b 6e ba 74 06 01 02 03 04
0020  00 00 00 00 00 00 01 02 03 01
```

```
0000  f2 1b 6e ba 74 06 8e a3 73 23 61 35 08 06 00 01
0010  08 00 06 04 00 02 8e a3 73 23 61 35 01 02 03 01
0020  f2 1b 6e ba 74 06 01 02 03 04
```

```
0000  8e a3 73 23 61 35 f2 1b 6e ba 74 06 08 00 45 00
0010  00 3c 48 0a 40 00 40 06 ea a9 01 02 03 04 01 02
0020  03 01 93 fb 00 16 a3 68 da bb 00 00 00 00 a0 02
0030  16 d0 84 46 00 00 02 04 05 b4 04 02 08 0a ff ff
0040  92 b1 00 00 00 00 01 03 03 01
```

```
0000  f2 1b 6e ba 74 06 8e a3 73 23 61 35 08 00 45 00
0010  00 3c 00 00 40 00 40 06 32 b4 01 02 03 01 01 02
0020  03 04 00 16 93 fb a3 6c 28 0c a3 68 da bc a0 12
0030  16 a0 1b 2a 00 00 02 04 05 b4 04 02 08 0a ff ff
0040  9d c2 ff ff 92 b1 01 03 03 01
```



Couche réseau (network layer)

Transmission de données d'un point A à un point B en utilisant des points **intermédiaires** comme relais.

Notion d'**adresse**, comprise par tous les points intermédiaires.

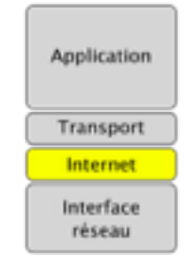
Notion de **routage**, indépendant ou non des données.

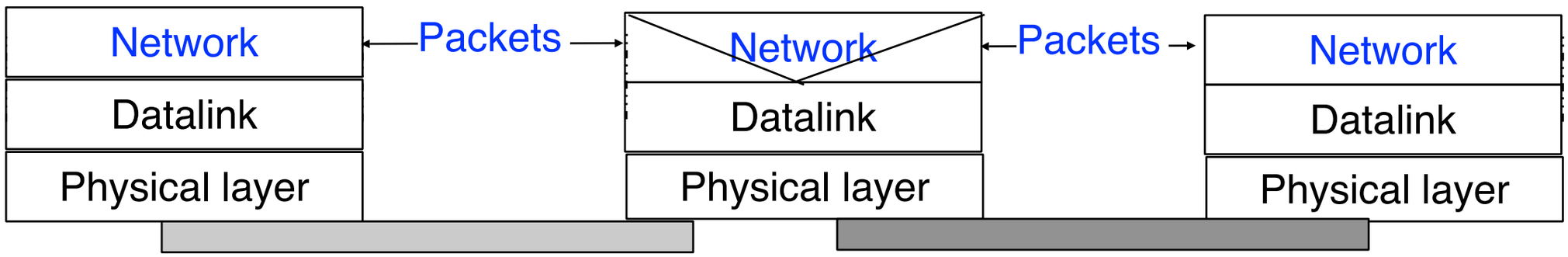
Le message peut être perdu.

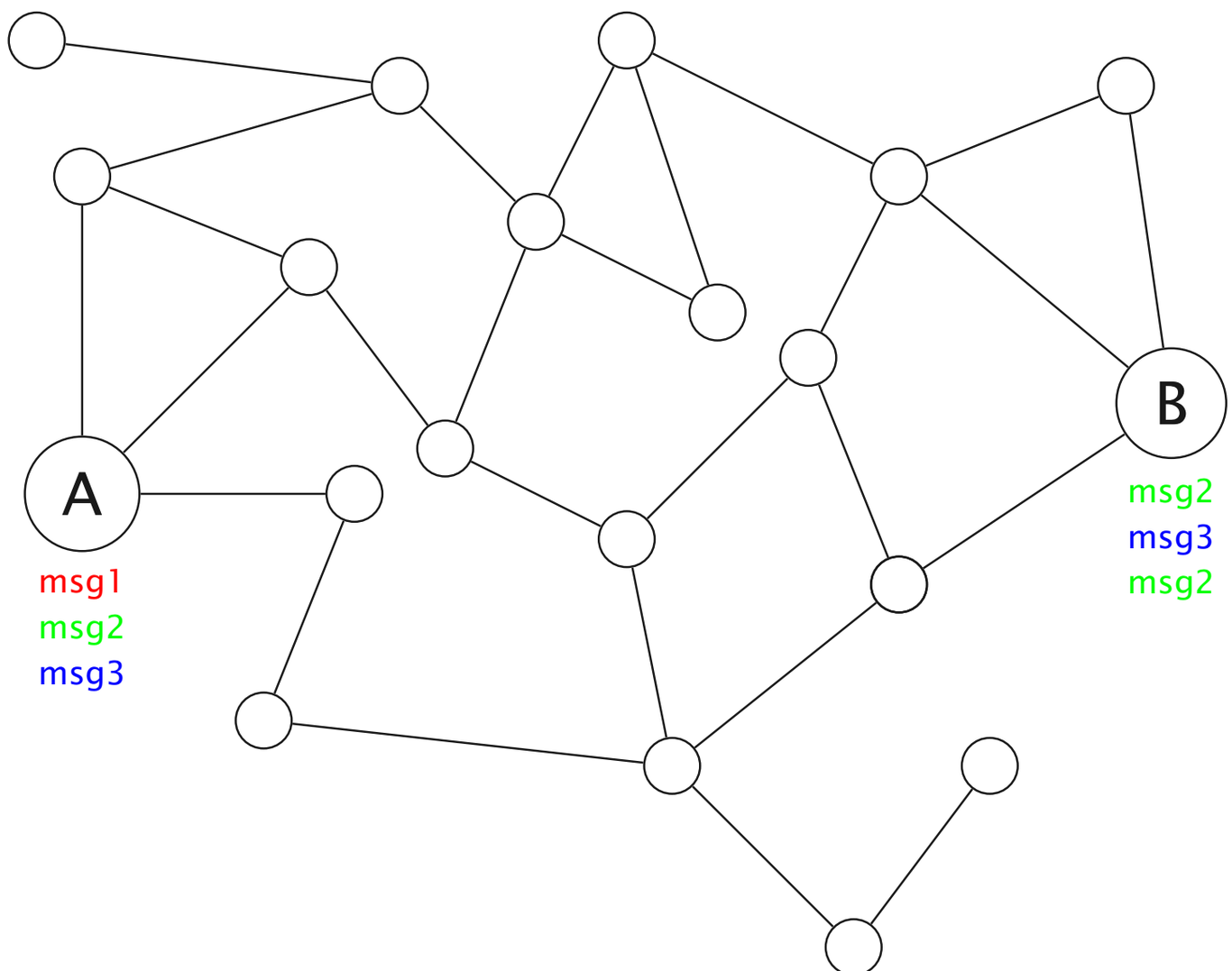
Les morceaux d'un message peuvent arriver dans le désordre.

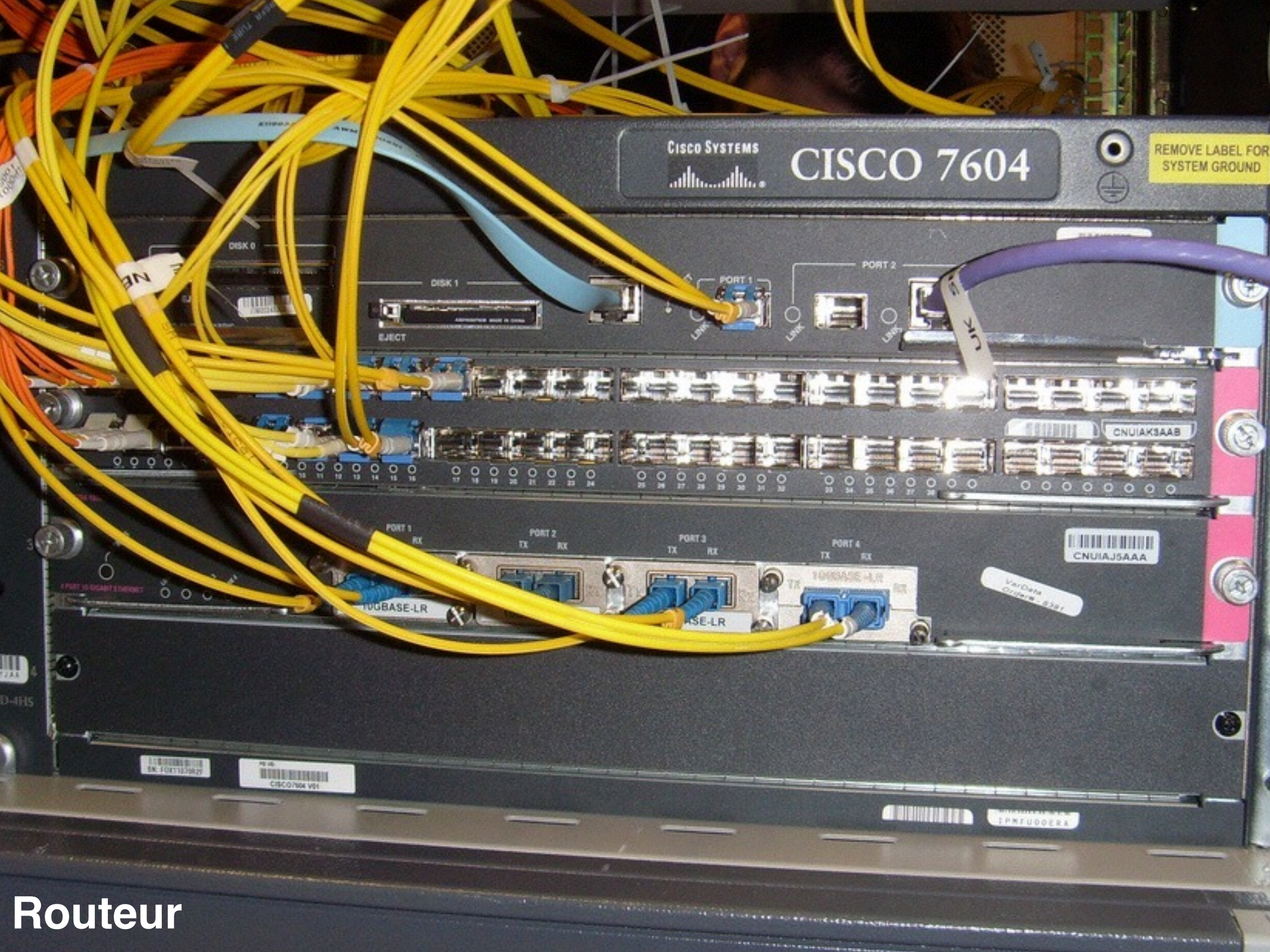
IP

Couche réseau (network layer)



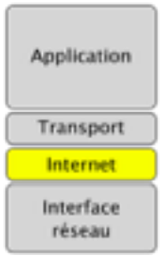





CISCO SYSTEMS
CISCO 7604
REMOVE LABEL FOR SYSTEM GROUND
DISK 0
DISK 1
EJECT
PORT 1
PORT 2
LINK
CNUIAK3AAB
CNUIAJ5AAA
PORT 1 RX
PORT 2 TX RX
PORT 3 TX RX
PORT 4 TX RX
10GBASE-LR
10GBASE-LR

Routeur

Problématique de la couche réseau



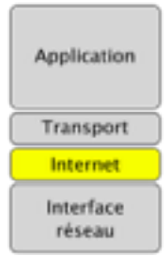
interconnexion de réseaux locaux sans hypothèse sur la topologie du réseau.

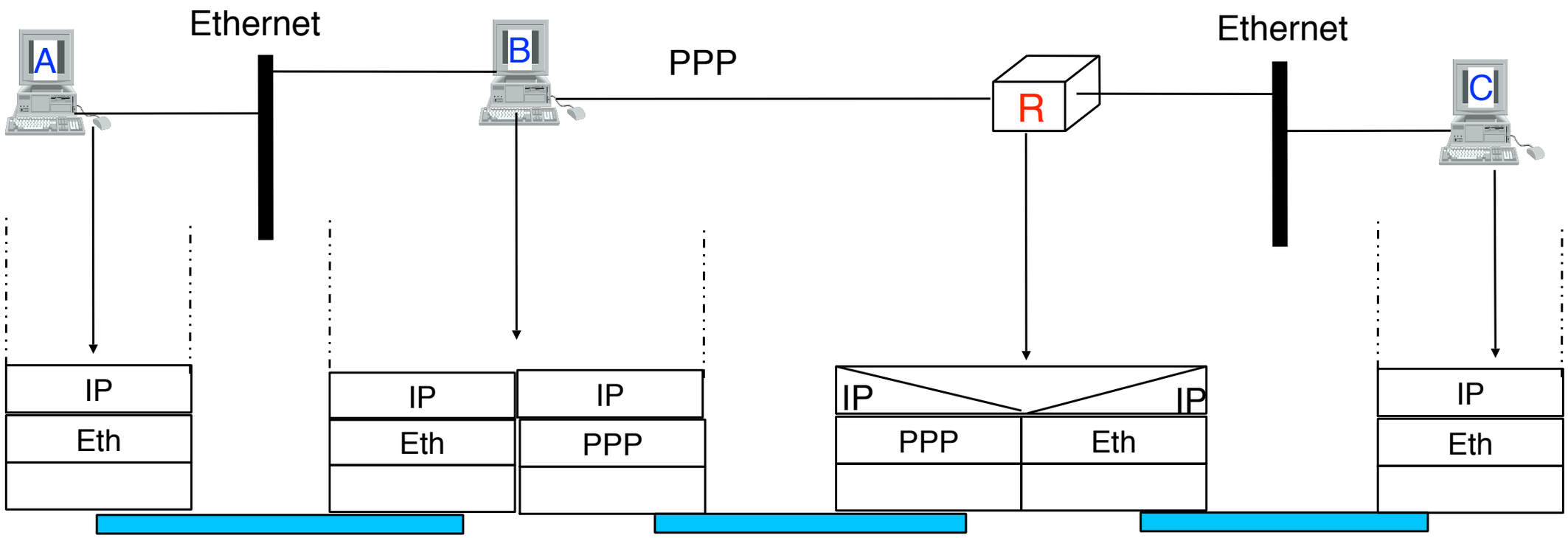
Si une machine veut communiquer avec une autre machine, elle doit pouvoir :

- la nommer ⟶ **adressage** ;
- la contacter ⟶ **routage**.

La couche réseau spécifie le **format des paquets** qui permet de réaliser ces objectifs.

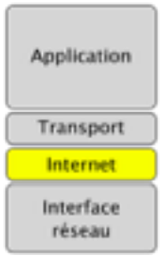
Internet Protocol





Un **hôte** peut **émettre** et **recevoir** des paquets dont il est expéditeur ou destinataire, il connaît au moins un **routeur** sur son réseau local.

Un **routeur** peut **émettre** et **recevoir** mais aussi **relayer** des paquets, il sait joindre les autres routeurs.



Adressage

Attribuer une **adresse** à chaque **interface réseau**.

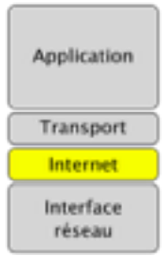
Comment envoyer un paquet à une machine en connaissant uniquement son adresse ?

Identifier, à partir de l'adresse, le **réseau** auquel appartient la machine.

Principe : pour envoyer un paquet à une machine

ou bien je m'aperçois qu'elle est dans mon réseau local et je lui envoie directement ;

ou bien ce n'est pas le cas, et je trouve quelqu'un (un **routeur**) pour le faire à ma place.



En pratique

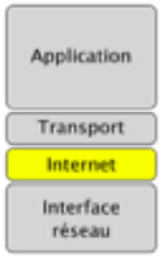
Une **adresse IPv4** est codée sur **32 bits** (4 octets) et représentée en notation pointée.

Exemple :

```
10001100      01011110      00010000      01000000
   140     .     94     .     16     .     64
```

Comment identifier l'ensemble des adresses des machines appartenant à un même réseau ?

Le **masque réseau** permet de de découper l'adresse en un **identifiant réseau** + un **identifiant d'hôte**.



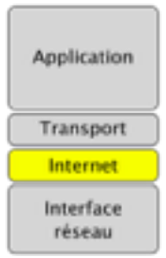
Notation CIDR

La notation **CIDR** spécifie la longueur du **préfixe** correspondant à l'**identifiant réseau**.

`10.13.37.0/24` : une machine appartient au réseau si ses 24 premiers bits sont égaux à **`10.13.37`**

Deux adresses particulières :

- **adresse réseau** : identifiant hôte à **`0`**
- **adresse broadcast** : tous les bits hôte à **`1`**



Blocs réservés

Machine locale (ne se voit jamais sur le réseau) :

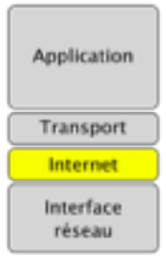
127.0.0.0/8

Réseaux privés (ne se voient jamais sur l'internet) :

10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
169.254.0.0/16

There is no place like 127.0.0.1

Attribution des adresses publiques

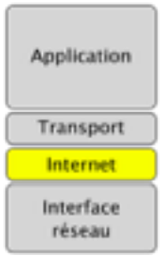


Allocations de **blocs d'adresses**.

Nécessité d'un organisme centralisateur :

IANA (Internet Assigned Numbers Authority)

```
$ whois 208.97.177.124
…
NetRange:       208.97.128.0 - 208.97.191.255
CIDR:           208.97.128.0/18
NetName:        DREAMHOST-BLK5
NetHandle:      NET-208-97-128-0-1
Parent:         NET208 (NET-208-0-0-0-0)
NetType:        Direct Allocation
…
```



Routage

Chaque machine possède une **table de routage**.

Les routeurs **relaie** les paquets en utilisant uniquement cette table.

Toute la subtilité est dans le remplissage (et la mise à jour !) de la table.

Entrée **par défaut** dans la table pour les destinations que le machine ne connaît pas explicitement.

Network	Gateway	Interface
10.13.37.0/24	*	en0
default	10.13.37.254	en0

Couche transport (transport layer)

Transmission **fiable** de données d'un point A vers un point B en utilisant des points intermédiaires comme relais.

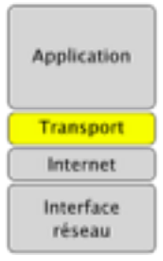
Notion de **port** pour distinguer les applications sur un même hôte.

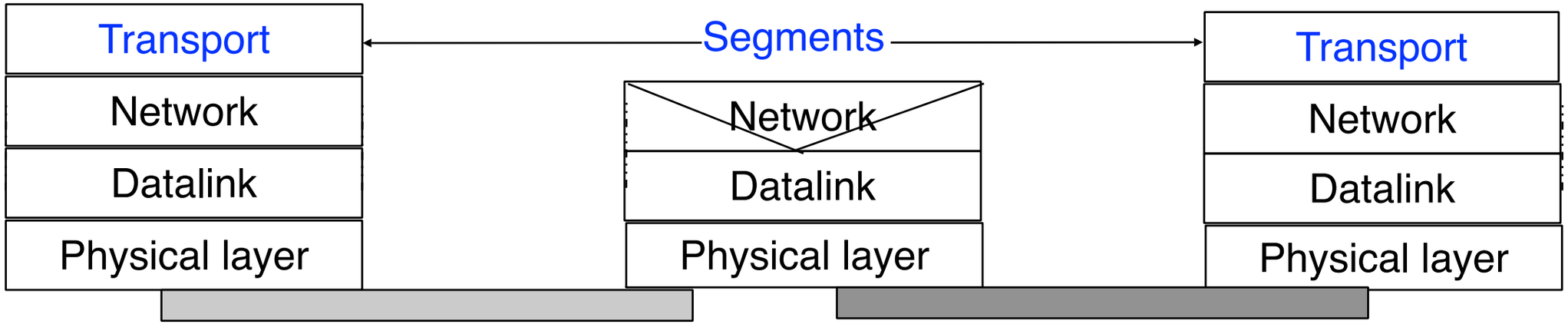
Transmission de **datagrammes**, en mode non-connecté.

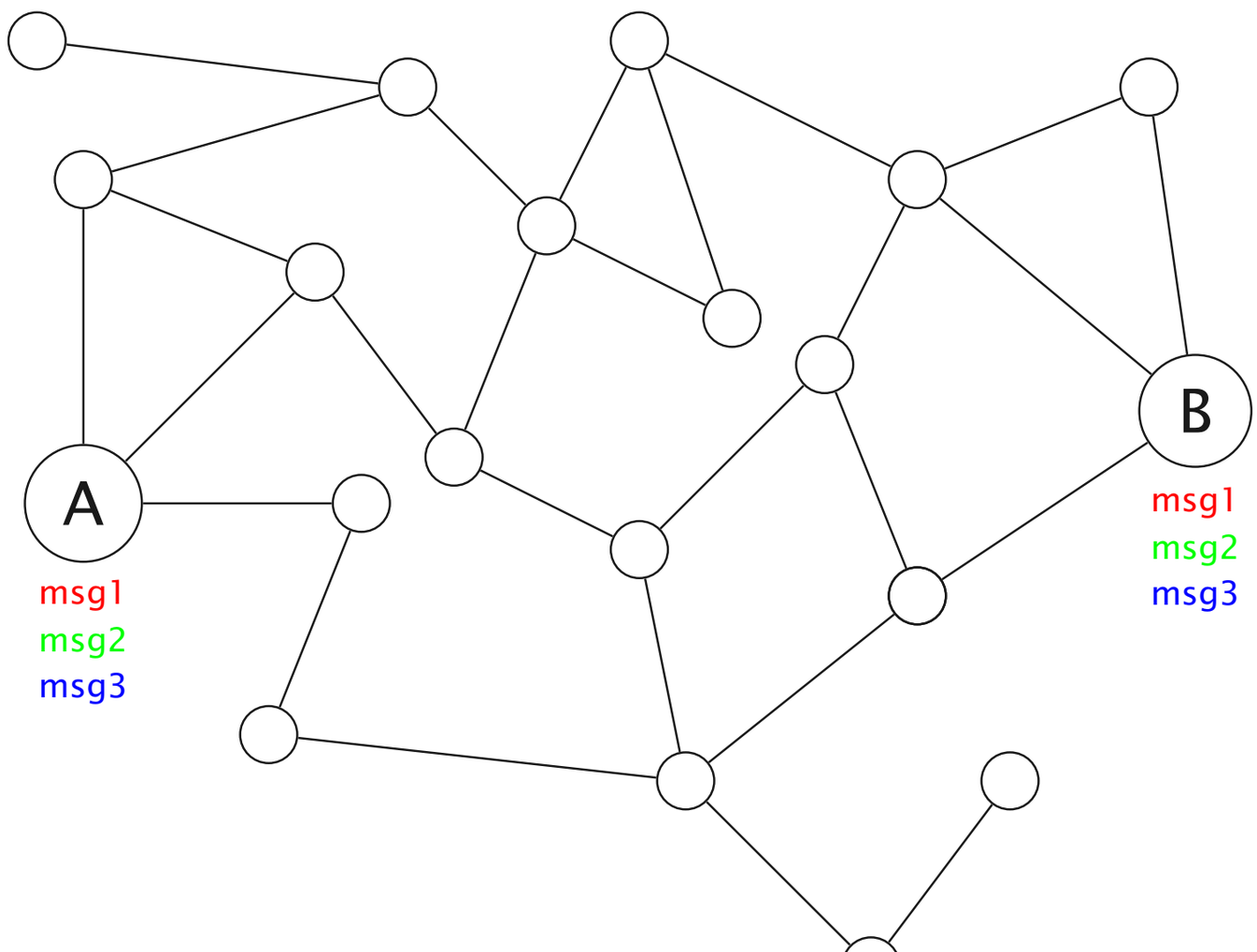
Transmission de **flux de données**, en mode connecté.

TCP/UDP

Couche transport (transport layer)







UDP

Permet le multiplexage entre applications en ajoutant des **ports** aux paquets IP… et c'est tout !

```
 0       7 8       15 16      23 24      31
+--------+---------+---------+---------+
|     Source        |   Destination   |
|      Port         |      Port       |
+--------+---------+---------+---------+
|     Length        |    Checksum     |
+--------+---------+---------+---------+
```

Service **non connecté** et **non fiable**, l'application doit continuer à fonctionner si certains datagrammes sont perdus.

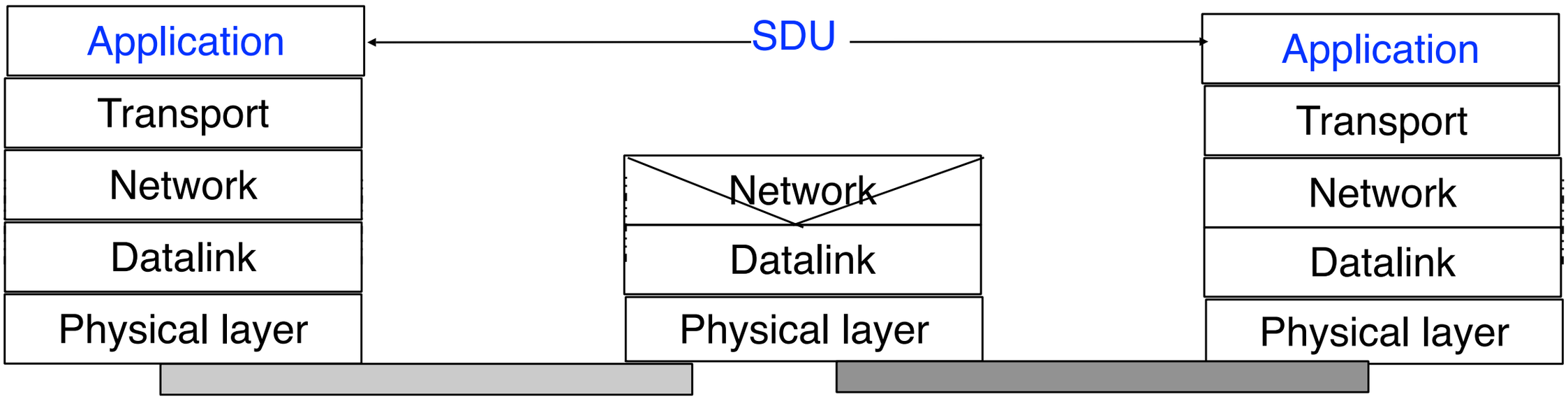
TCP

Service **connecté** et **fiable** avec contrôle de **congestion**.

Phase de **connexion**, de **communication** puis de **fermeture** de connexion.

Flux de données bidirectionnel : sans perte de données, sans duplication, dans le même ordre.

Couche application



Protocoles applicatifs, aussi nombreux que variés.

SMTP, **HTTP**, **FTP**, **XMPP**, ...

Couche application

Dernière couche des modèles **OSI** et **TCP/IP**

Échange de **messages** entre processus, généralement utilisée en **bordure de réseau**.

Un **protocole** de niveau application doit spécifier :

- le type et le contenu des messages échangés ;
- les règles déterminant quand ces messages doivent être envoyés/reçus.

Agent utilisateur (**user agent**) : interface entre l'utilisateur et le protocole (ex : navigateur web)

Modèle client/serveur

En général un **protocole** applicatif utilisera le **modèle client/serveur** :

- Le serveur attend des connexions, fournit un service
- Le client initie la connexion, demande un service

Exemples :

- **HTTP** : navigateur/serveur Web
- **SMTP** : MTA qui envoie/MTA qui reçoit

Sockets

Une des extrémités d'une communication sur l'internet. Abstraction de l'API de référence.

Une socket est donnée par :

- l'adresse de transport locale (**IP** + **port**) ;
- l'adresse de transport du destinataire ;
- le protocole de transport (**UDP**, **TCP**, ...)

Une socket connectée est déterminée par le quadruplet : `(adresse réseau source, port source, adresse réseau destination, port destination)`

Types de sockets

Plusieurs types possibles. Les plus fréquents :

- Socket **stream** (connectée, par ex. TCP) ;
- Socket **datagram** (non connectée, par ex. UDP) ;
- Socket **raw** : accès à la couche réseau.

Sockets en Python 3

Le module **socket** permet d'utiliser les socket BSD.

Portable sous Linux, Windows, Mac OS, …

Le module propose un objet **`socket`** à instancier.

Les données lues ou écrites à travers une socket sont de type **`bytes`**, à convertir depuis/vers des chaînes de caractères UTF-8 à l'aide des méthodes **decode** et **encode** :

```
>>> s="un éléphant qui se balançait"
>>> s.encode('utf-8')
b'un \xc3\xa9l\xc3\xa9phant qui se balan\xc3\xa7ait'
>>> m = b"12345"
>>> m.decode('utf-8')
'12345'
```

Socket TCP en Python 3

côté serveur

```
from socket import socket

s=socket()
s.bind(('0.0.0.0',5555))
s.listen(3)
while True:
    c,addr=s.accept()
    print(">>> %s:%d" % addr)
    msg=c.recv(1024)
    print(msg.decode('utf-8'))
    c.send(msg)
    c.close()
```

côté client

```
from socket import socket

s=socket()

s.connect(('127.0.0.1',5555))
msg=input("Message : ")
s.send(msg.encode('utf-8'))
msg=s.recv(1024)
print(repr(msg.decode('utf-8')))
s.close()
```

Méthodes

`bind((addr,port))` : réserve le port **`port`** et filtre selon **`addr`** (utiliser `'0.0.0.0'` pour ne pas filtrer).

`listen(n)` : crée une file d'attente pour **`n`** connexions en attente d'acceptation.

`accept()` : accepte une connexion entrante, retourne une paire **`(socket,addr)`**.

`connect((addr,port))` : connexion en tant que client.

`send(msg)` : envoie **`msg`** à travers la socket.

`recv(max)` : reçoit jusqu'à **`max`** octets à travers la socket.

Socket UDP en Python 3

côté serveur

côté client

```
from socket import socket, AF_INET, SOCK_DGRAM
```

```
s=socket(AF_INET,SOCK_DGRAM)
s.bind(('0.0.0.0',5555))
while True:
    msg,addr=s.recvfrom(1024)
    print(">>> %s:%d" % addr)
    print(repr(msg.decode('utf-8')))
    s.sendto(msg,addr)
```

```
s=socket(AF_INET,SOCK_DGRAM)
msg=input("Message : ")
s.sendto(msg.encode('utf-8'),
('127.0.0.1',5555))
msg,addr=s.recvfrom(1024)
print(repr(msg.decode('utf-8')))
s.close()
```

Multiplexage

Un serveur peut recevoir **plusieurs connexions** en même temps.

Chaque connexion est accessible à travers une **socket** distincte.

Multiplexage dans un programme séquentiel avec **select**...

...dans un programme parallèle à l'aide de threads.

netcat en Python 3

```
from sys import argv
from socket import socket, gethostbyname
from _thread import start_new_thread,
allocate_lock

def sockread(lock):
    try:
      while True:
        msg=s.recv(1024)
        if len(msg)==0:
            lock.release()
        print(msg.decode('utf-8'),end='')
    except:
      lock.release()

def sockwrite(lock):
    try:
      while True:
        msg=input("")
        s.send((msg+'\n').encode('utf-8'))
    except:
      lock.release()

if len(argv)!=3:
    print("usage: nc -l port | nc addr
port")
    exit(1)
```

```
s=socket()
if argv[1]=='-l':
    port=int(argv[2])
    s.bind(('0.0.0.0',port))
    s.listen(1)
    c,addr=s.accept()
    s=c
else:
    addr=gethostbyname(argv[1])
    port=int(argv[2])
    s.connect((addr,port))

lock=allocate_lock()
lock.acquire()
start_new_thread(sockread,(lock,))
start_new_thread(sockwrite,(lock,))
try:
  lock.acquire()
except:
  pass
```

Bibliographie

(a) O. Bonaventure. Computer Networking: Principles, Protocols and Practice. **http://cnp3book.info.ucl.ac.be/**

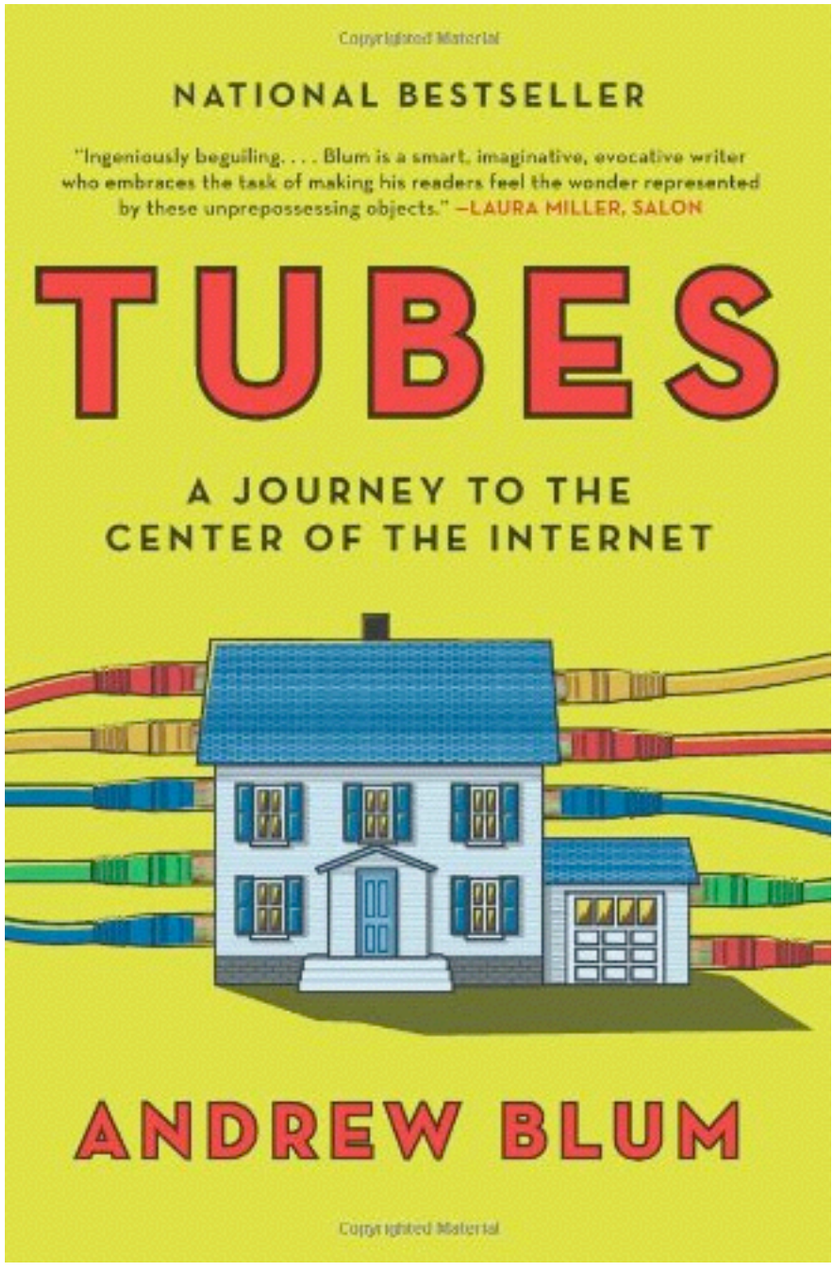
(b) D. Comer. Internetworking with TCP/IP. Pearson.

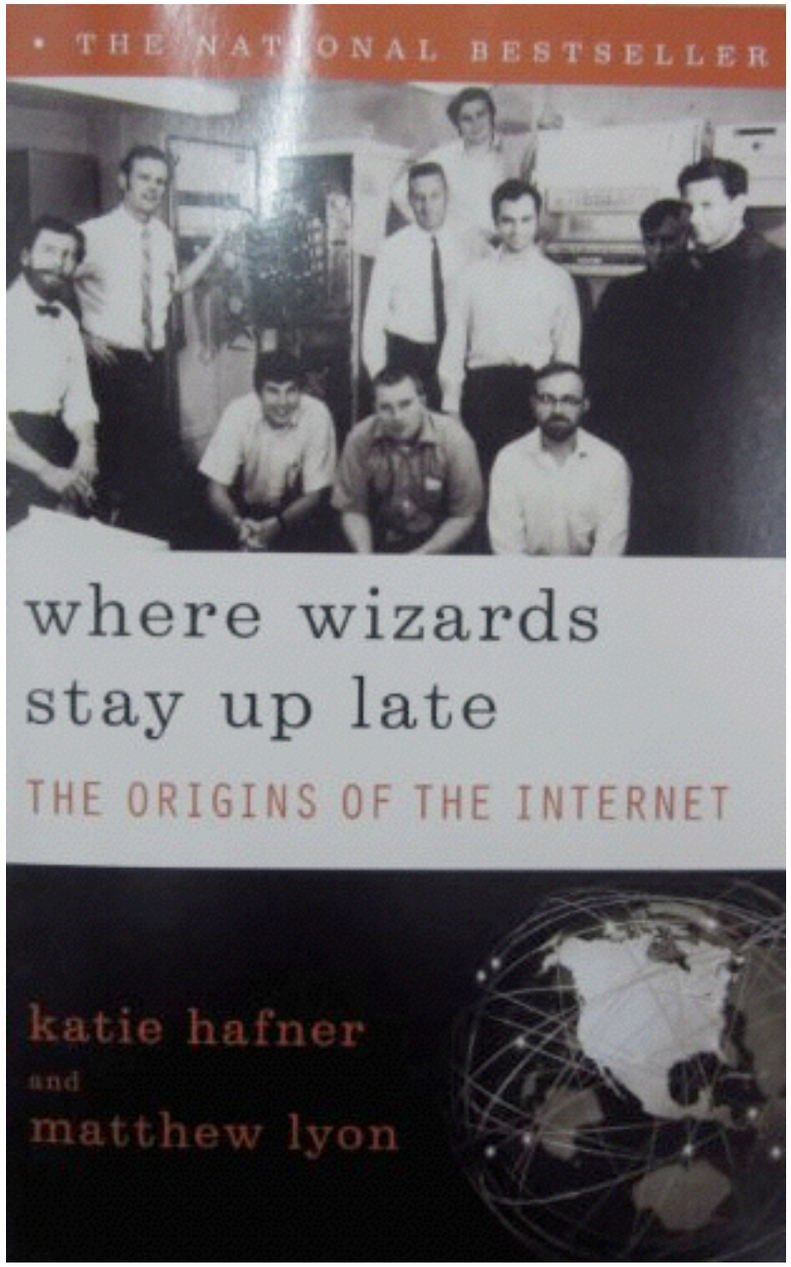
(c) J. Kurose and K. Ross. Computer Networking: a top-down approach. Pearson.

(d) G. Pujolle. Les réseaux. Eyrolles.

(e) A. Tanenbaum. Réseaux. Dunod.

Qu’est-ce que l’internet ?

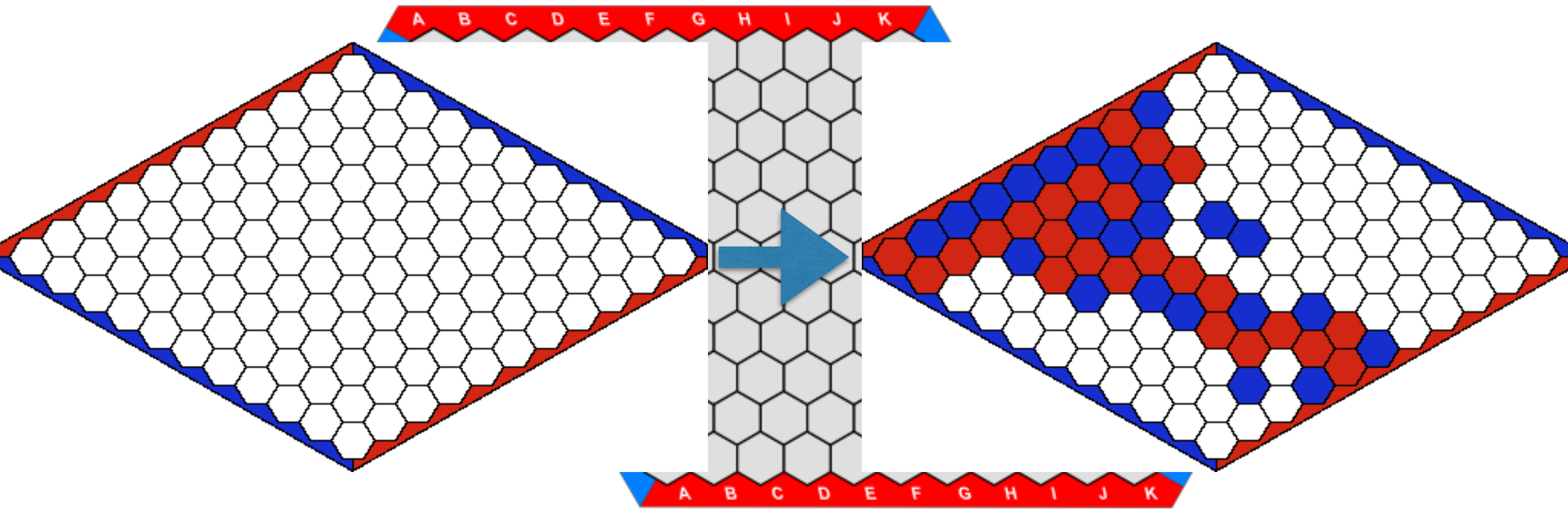




3. **hex**ercice pratique

hexplication

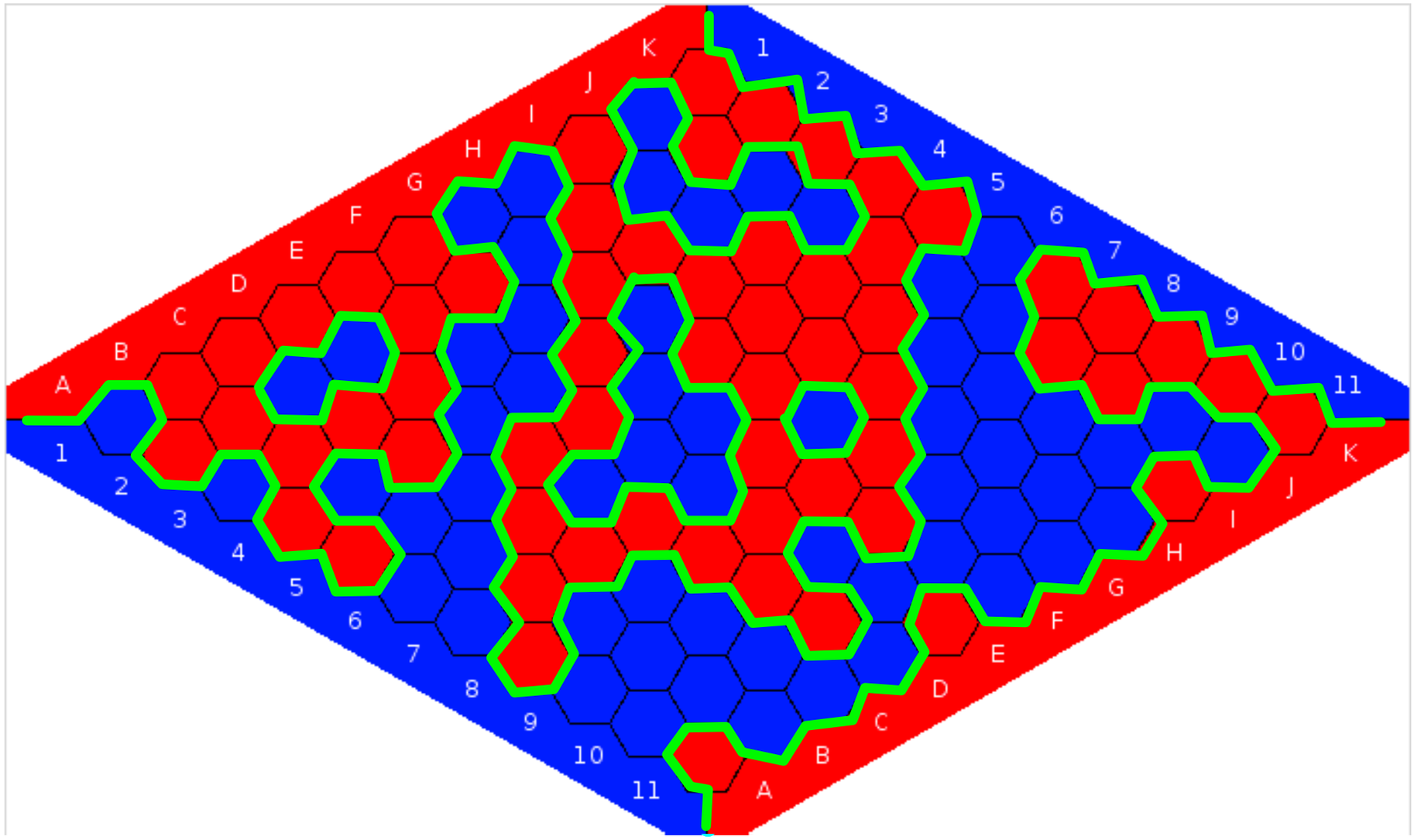
Jeu stratégique à 2 joueurs, sur une grille hexagonale, habituellement 11 x 11.



Le jeu de Hex

Inventeurs : Piet Hein (1942) et John Nash (1947)

Jamais de match nul



Le premier joueur peut gagner

Si les deux joueurs jouent parfaitement, c'est le premier qui gagnera.

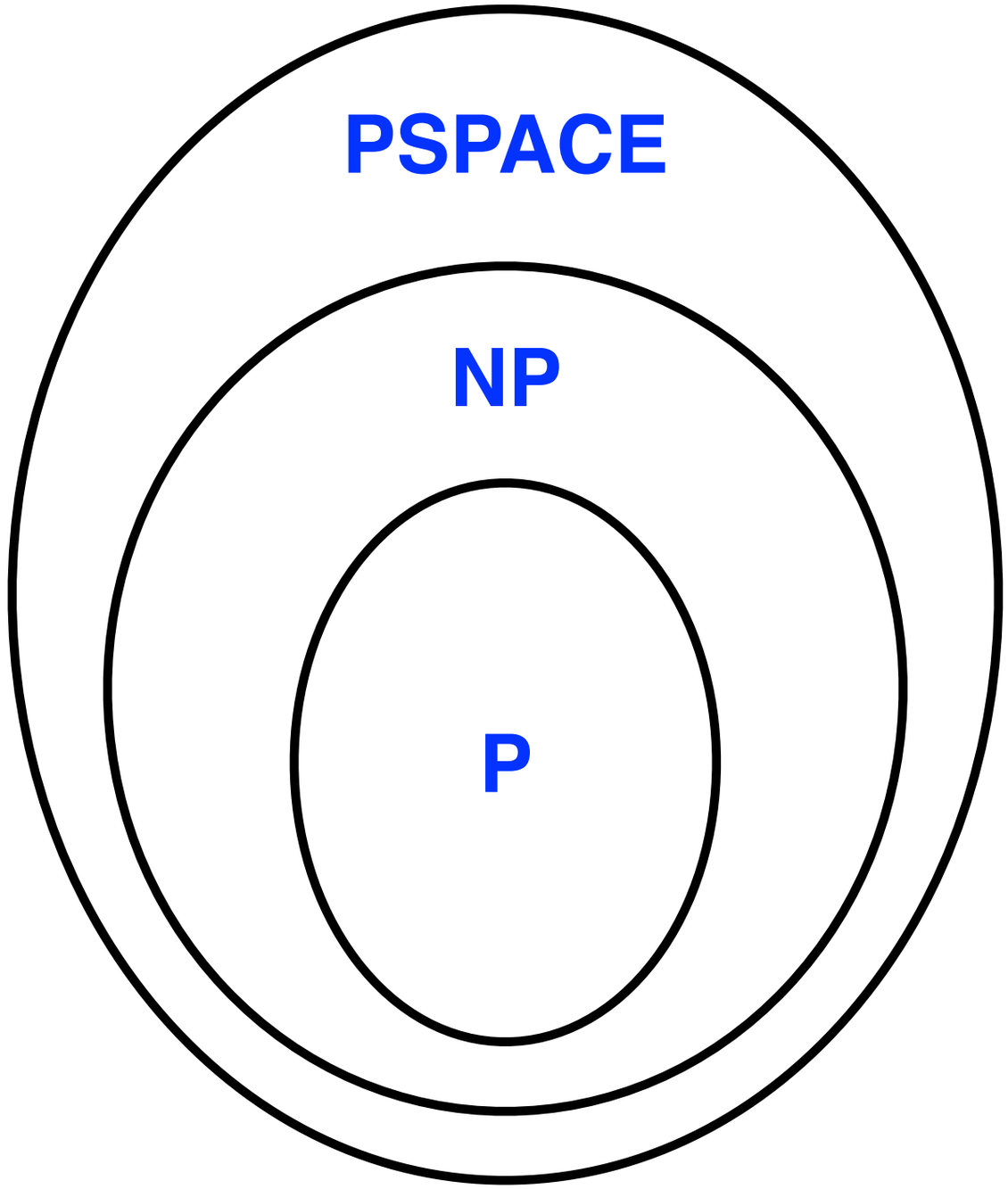
Preuve par «vol de stratégie».

Observation: Avoir une pièce en plus de sa couleur sur le tablier ne peut pas géner de gagner

Par contradiction, supposons que le joueur 2 ait une stratégie gagnante. Alors le joueur 1 peut commencer en plaçant une pièce au hasard. Le joueur 2 devient le premier à vraiment jouer, et le joueur 1 devient le second joueur. Il peut alors jouer la stratégie gagnante. Puisqu'il n'y a pas de match nul à la fin, le joueur 1 gagne.

Cette preuve n'est pas constructive.

hextrêmement difficile ?



Etant donnée une position sur un tablier n x n , décider si elle est gagnante est PSPACE-complet.

hexercice

Jeu de hex en réseau !

0. écrire un client minimaliste (socket, ...)

1. développer un client (qui joue contre une IA)

2. développer un serveur (pour jouer à 2)

4. gérer des extensions (pile ou face, ...)