Sys 1- Amphi 7

Introduction aux réseaux - utiliser les socket TCP

Guillaume DIDIER - 17/10/2024

Sys 1 - Amphi 7

Programme du jour

- Introduction du cours
- Introduction aux Réseaux
- Utilisation de socket TCP, premier aperçu de la couche applicative
- Votre première requête Réseau

Introduction du cours

Qui suis-je ?

DIRECTION GÉNÉRALE DE L'ARMEMENT
DGA Maîtrise de l'information

Ingénieur Principal de l'Armement
Évaluateur logiciel

Inria UMR IRISA

Collaborateur Scientifique

Équipe TARAN - 1j / semaine
Sécurité des Micro-architectures

UMR IRISA
ENS | PSL

Thèse (2019 - 2022)

Carnegie Mellon University
Electrical & Computer ENGINEERING

École d'application (2018-2019)

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

2014 - 2017

Introduction du cours - Motivation

Pourquoi étudier les réseaux

- Sujet intéressant pour coder en C
- Les réseaux sont partout:
  - Débat politiques
  - Grilles de calcul
  - Systèmes distribués
- C’est au programme de l’Agrégation
- Domaine de recherche toujours actif
  - Beaucoup de questions théoriques non-résolues

Introduction du cours

Objectifs - Contenu

- Fondamentaux pour comprendre les réseaux
- Concepts
  - Modèle en couche
  - Notion de protocoles
  - Hiérarchie
- Quelques protocoles à titre d'*exemple*
  - HTTP, DNS, TCP, UDP, IP, routage, quelques idées sur les liens
- Quelques problèmes de sécurité

Introduction du cours

Organisation

- 7 CM - 1h30 - Jeudi soir avec moi
  - Les slides ne sont pas un support de cours
  - Prenez des notes
  - Support de références: Kurose, CS:APP, Notes de Martin
- 6 Séances pratiques - 1h30 - Mardi après-midi avec Matteo et Mathieu
  - 2 TP non notés, mais relus
  - 1 TP noté, individuel
  - 2 TD

- Note:
  - TP noté 40%
  - Partiel 60%
  - Bonus 5%
- Pour les questions:
  - Je serai sur en vocal discord les jeudi de 18h à 19h, profitez-en
  - Message sur le discord ENS
  - Mail guillaume.didier@inria.fr, matteo.chancerel@irisa.fr, mathieu.laurent@ens-rennes.fr
  - On peut prendre RDV

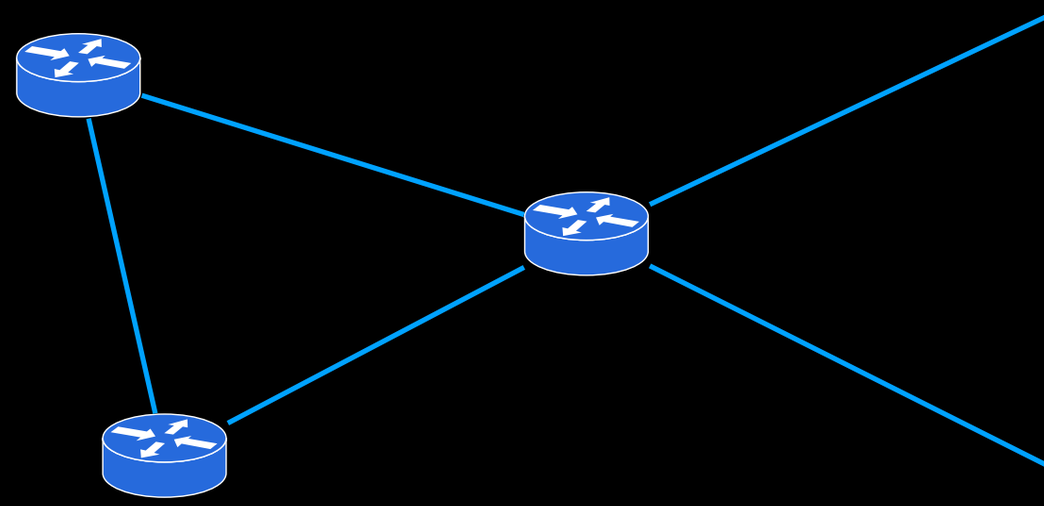
Introduction du cours

Règles du jeu

- Your own work !
  - Discuter OUI - Partager du code NON
  - Prendre du code d’internet ?
    - Pas de fraction significative de la solution
    - Citation de la source
  - Mathieu, Matteo et moi somme là pour vous aider, même sur votre code.
- Ne restez pas bloqués
- Prenez soin de vous, si soucis perso / santé etc, on peut s’arranger
- Lisez le syllabus
- ChatGPT et autres LLM sont à proscrire.

Introduction aux réseaux

Let’s get started

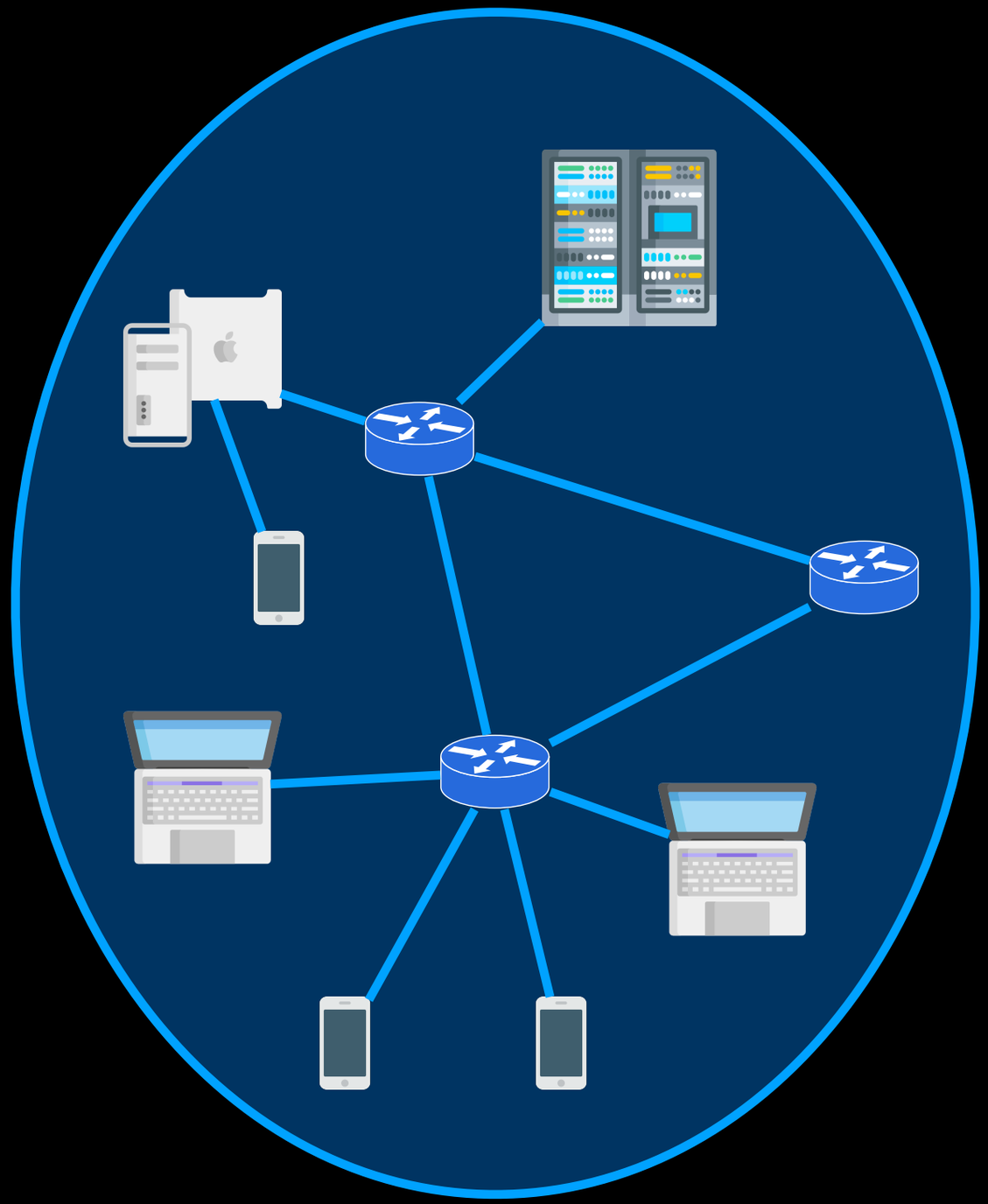


Introduction aux réseaux

Qu’est-ce qu’Internet ?

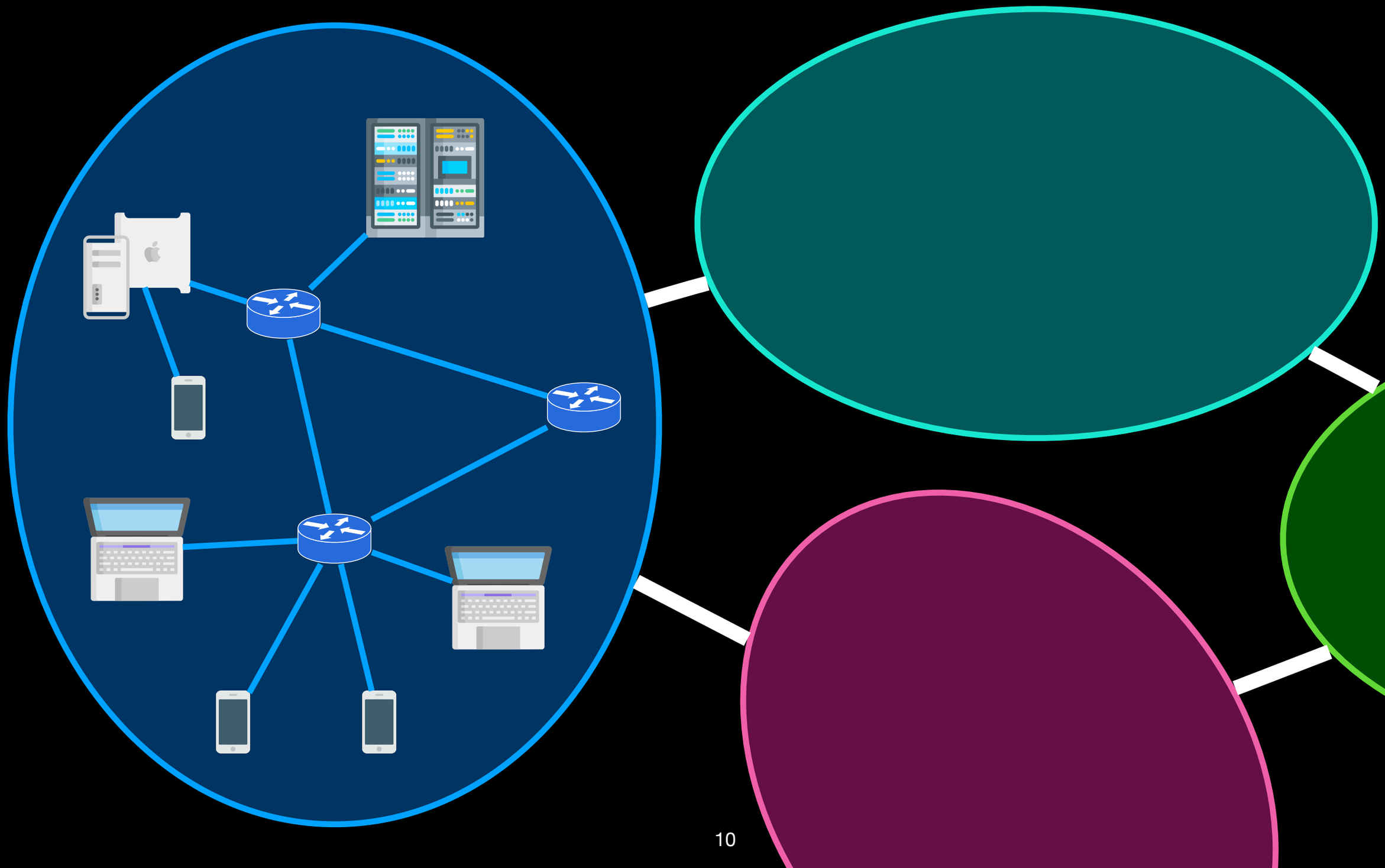
Introduction aux réseaux

Qu’est-ce qu’internet ? Qu’est qu’un Réseau ?



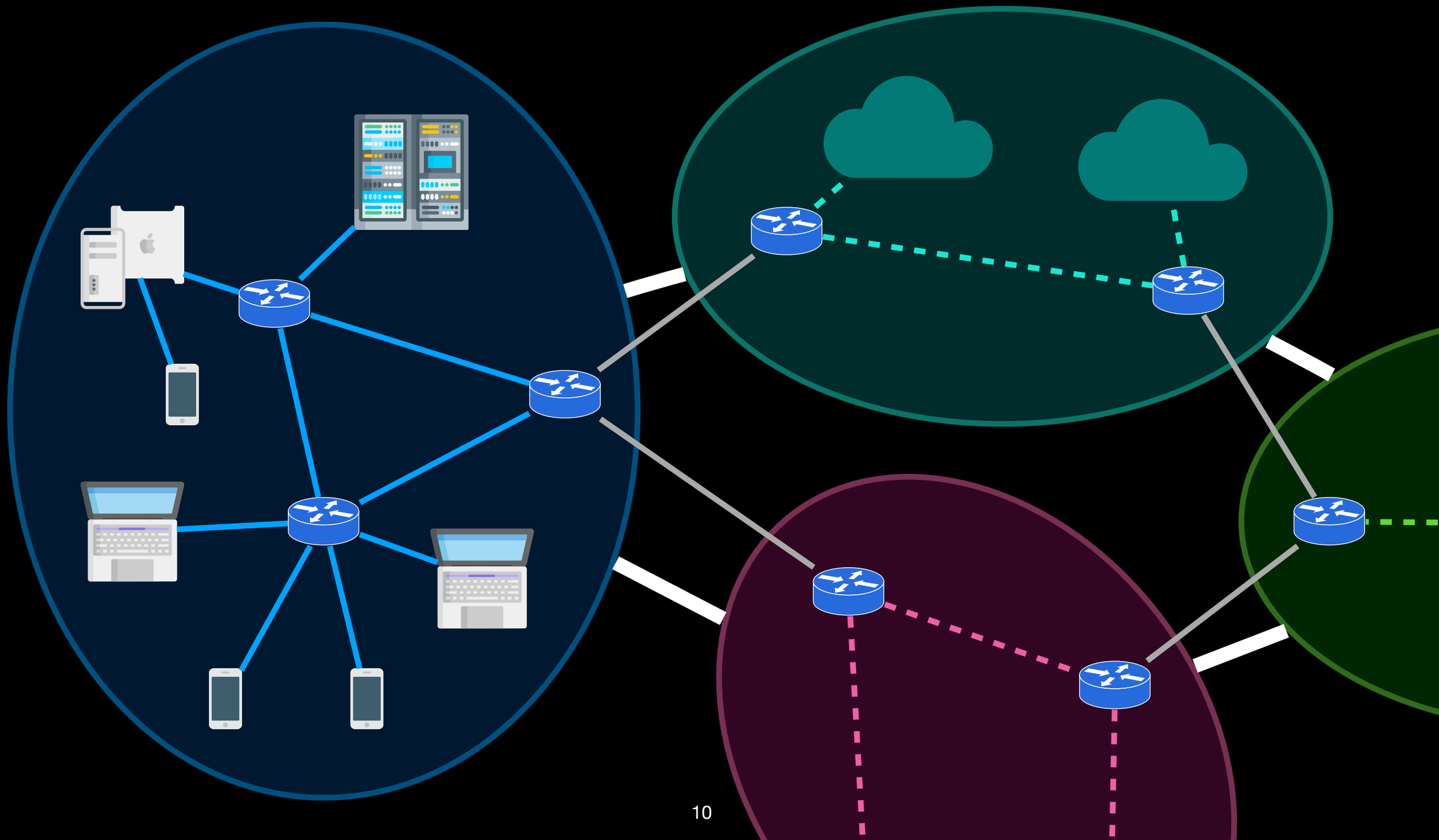
Introduction aux réseaux

Qu’est-ce qu’internet ? Qu’est qu’un Réseau ?



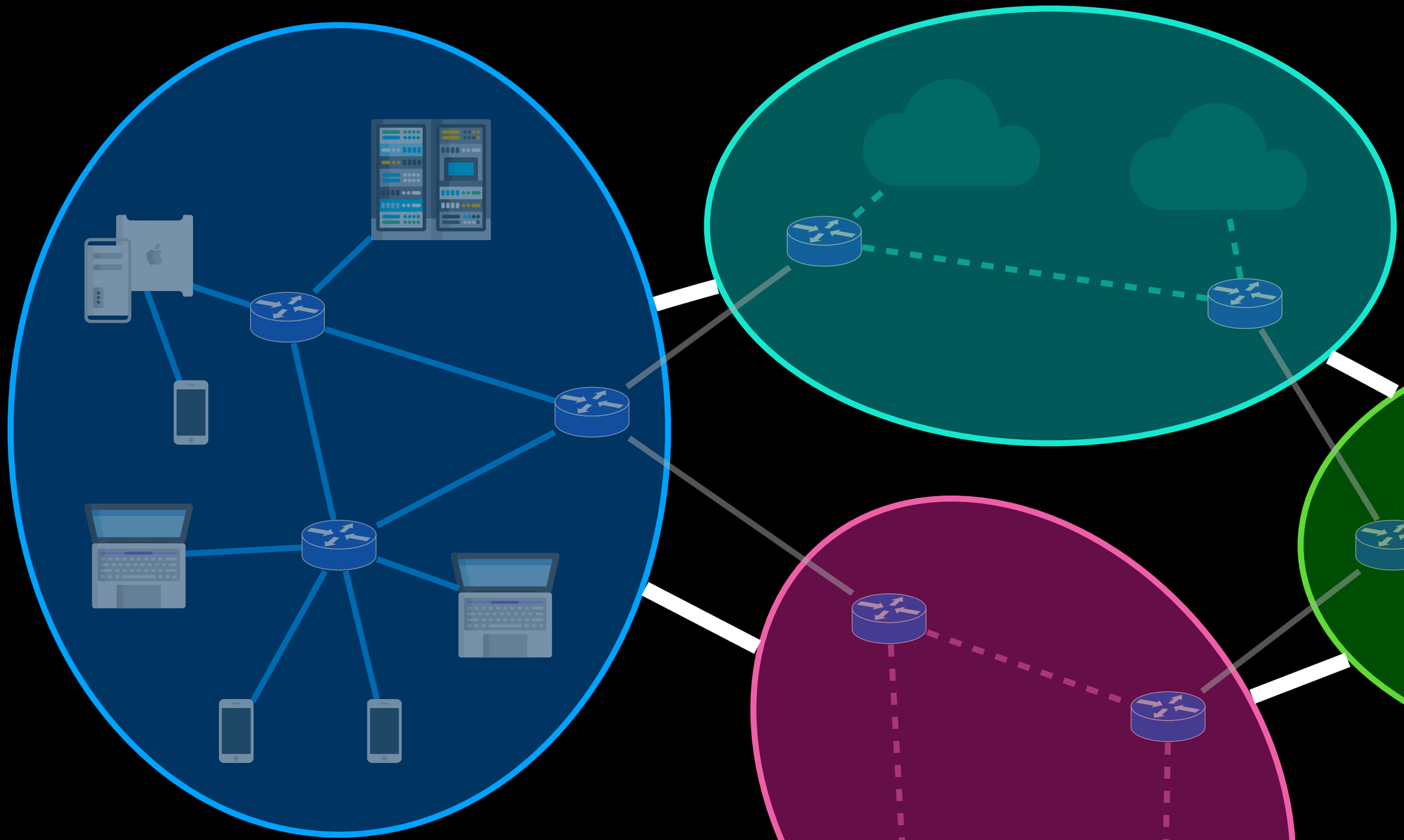
Introduction aux réseaux

Qu’est-ce qu’internet ? Qu’est qu’un Réseau ?



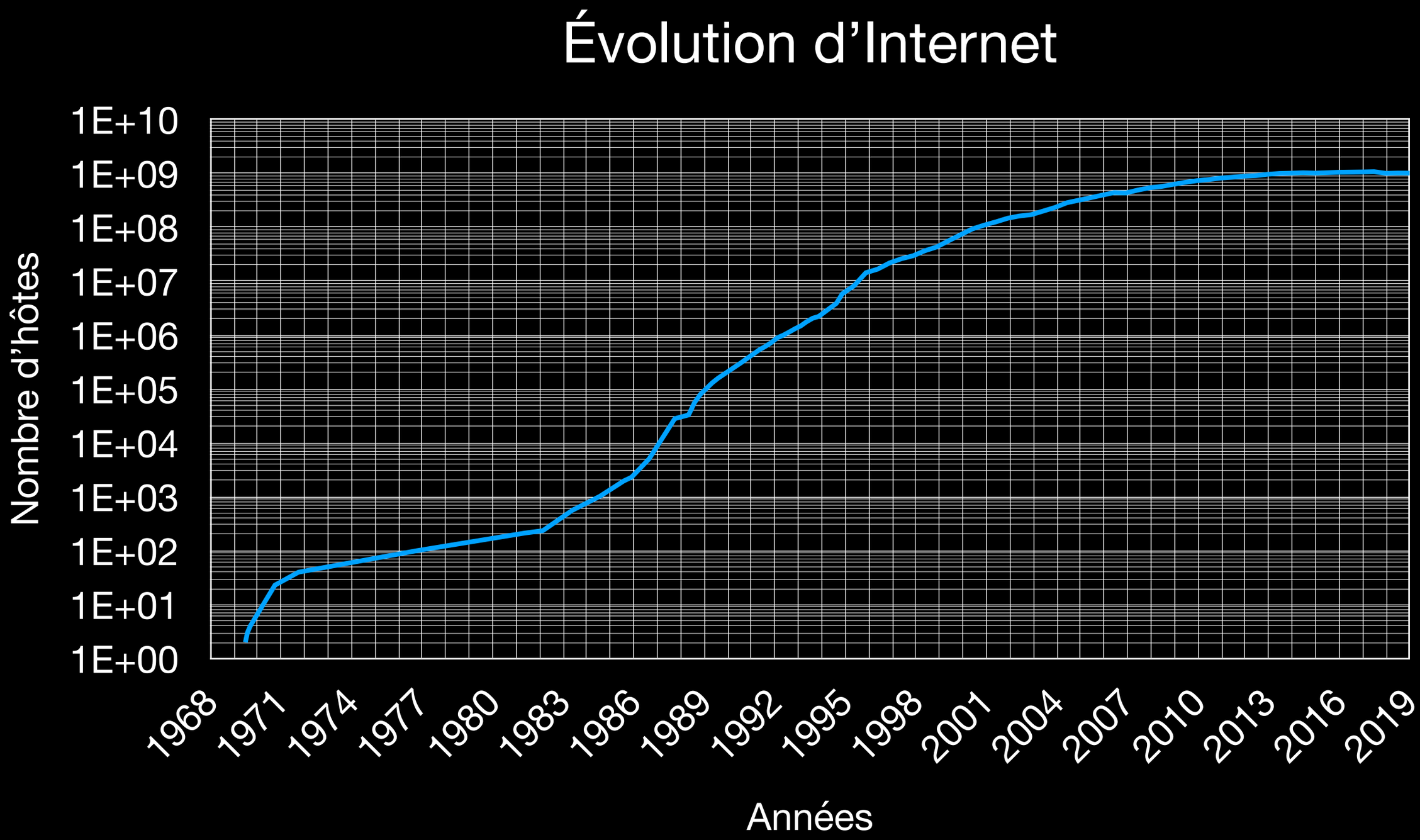
Introduction aux réseaux

Qu’est-ce qu’internet ? Qu’est qu’un Réseau ?



Repères historiques

L’internet évolue !



La diversité d’internet

Matériels, logiciels, connections, applications…

How can that even hold together ?!!

RFCs: les standards d'internet

Ou comment faire inter-opérer tout ça

L'IETF définit des **protocoles**:

« Request for Comments »

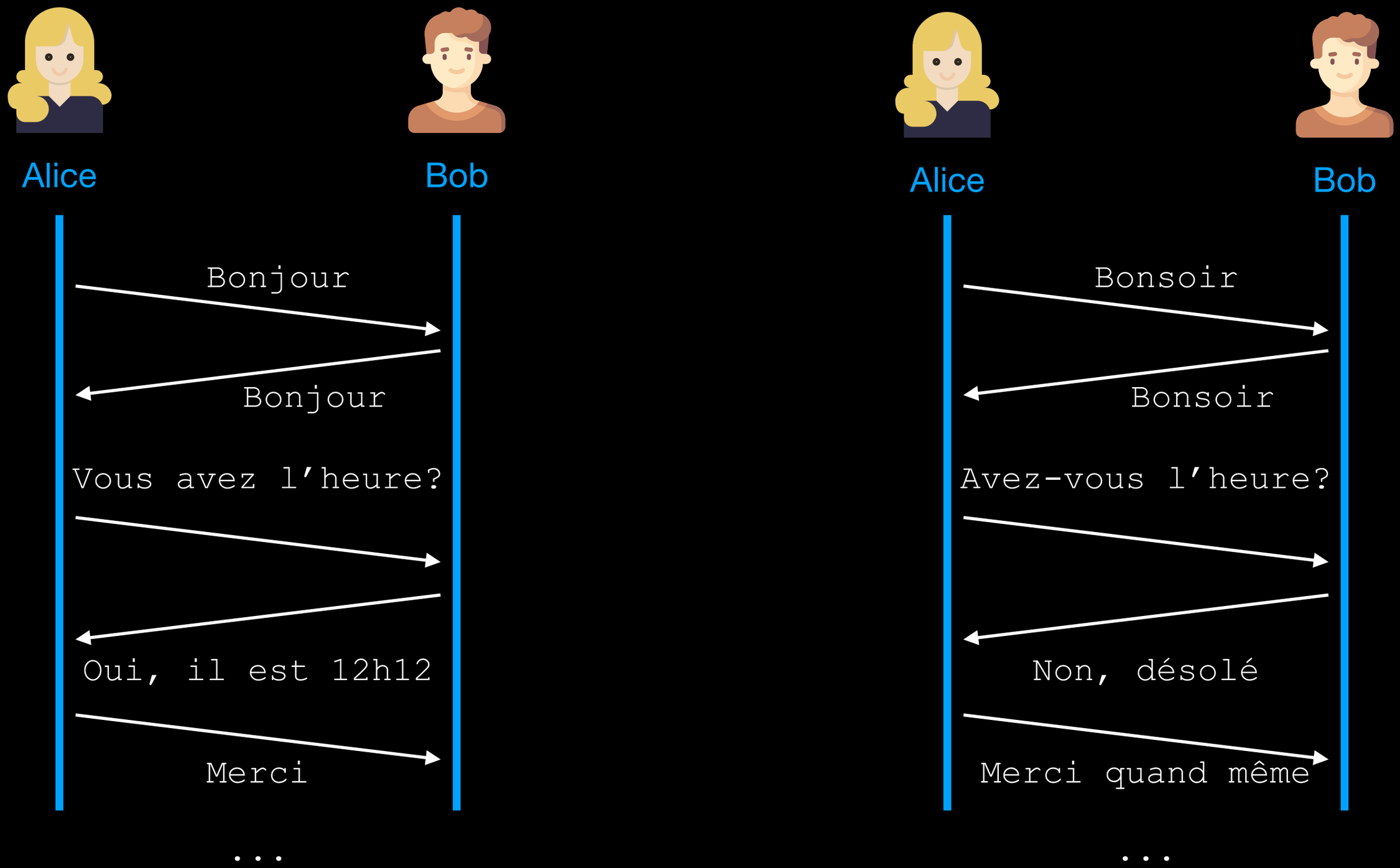
Robustness principle (Postel):

Be conservative in what you send, liberal in what you accept

IETF: *Rough Consensus and Running Code*

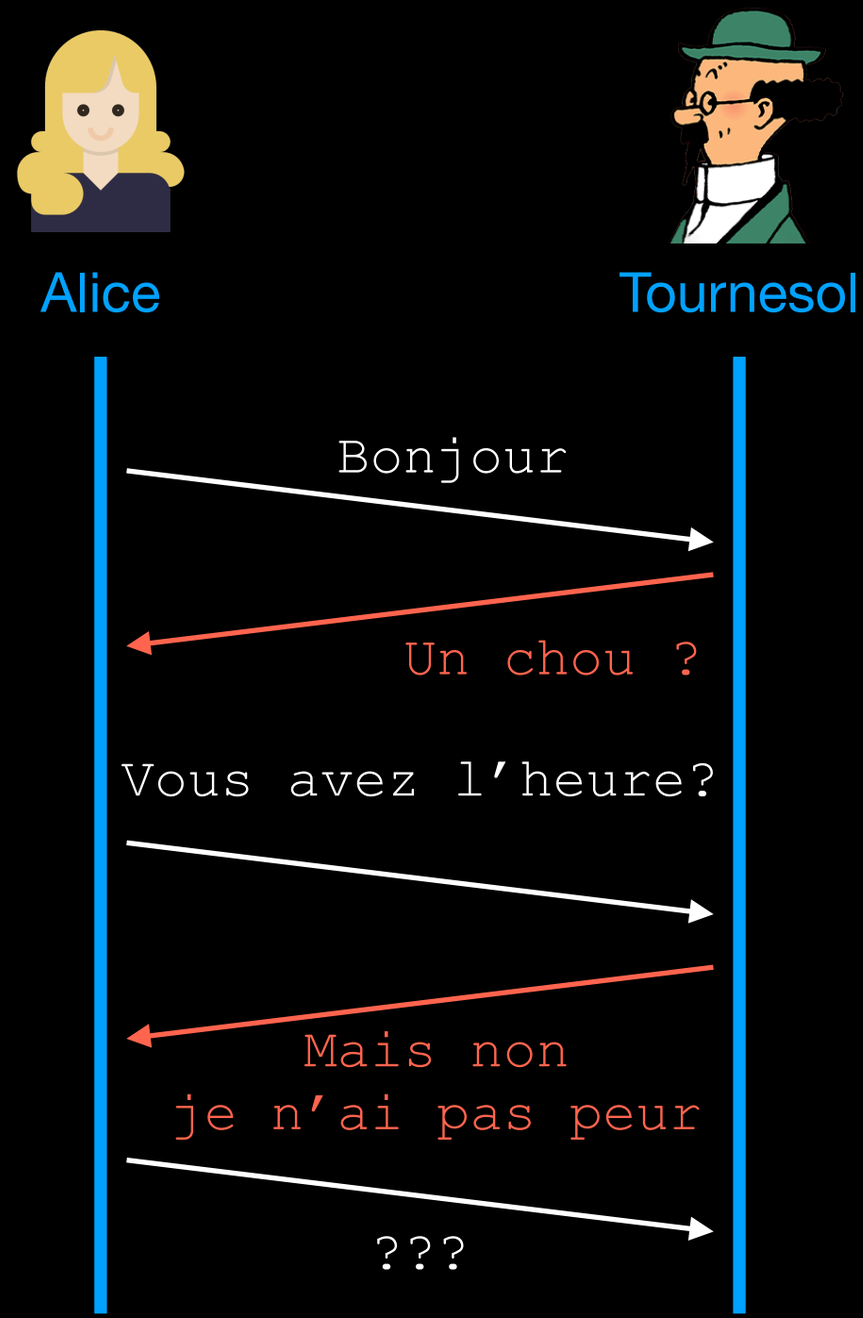
Protocole ?

Exemple : Demander l’heure



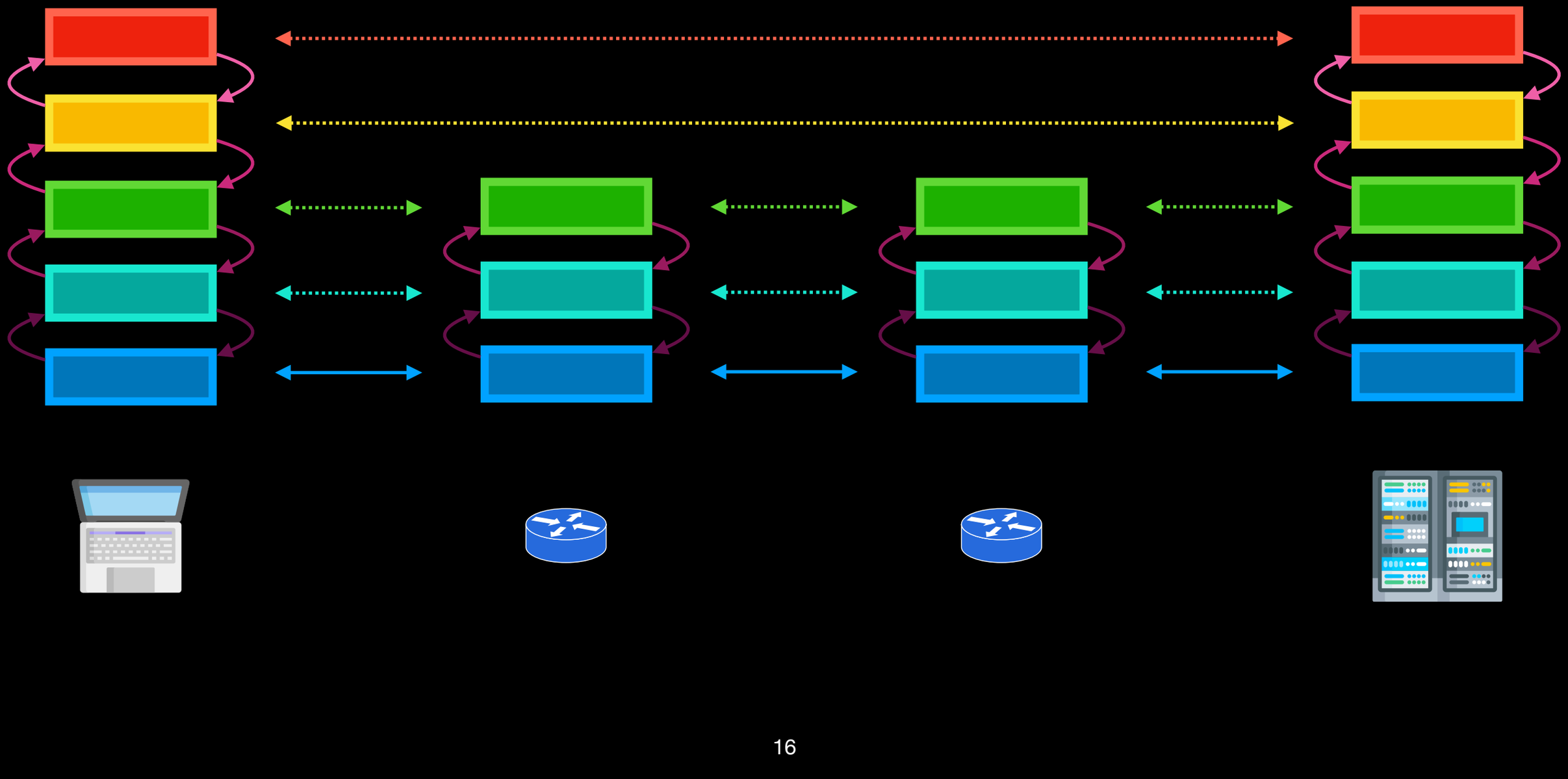
Protocole ?

Problème de communication



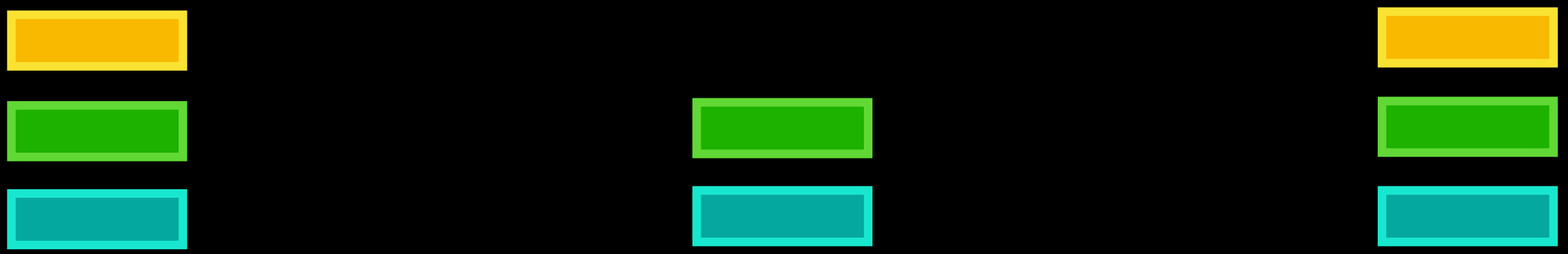
Un modèle par couche

Abstraction to the rescue !



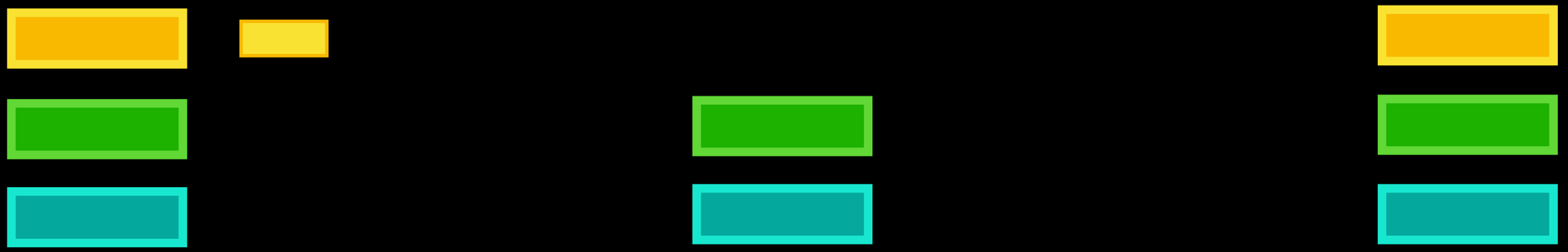
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



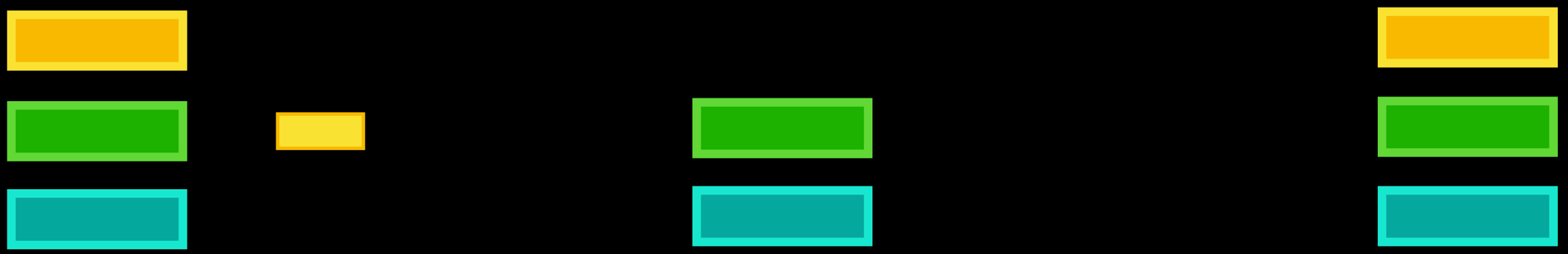
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



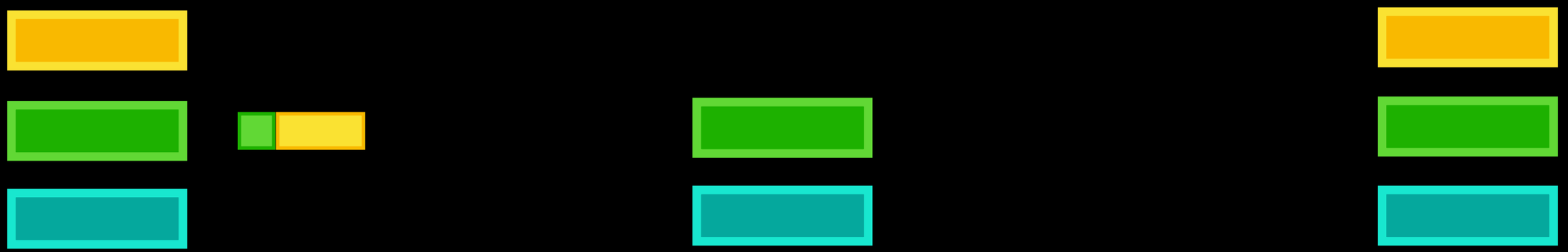
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



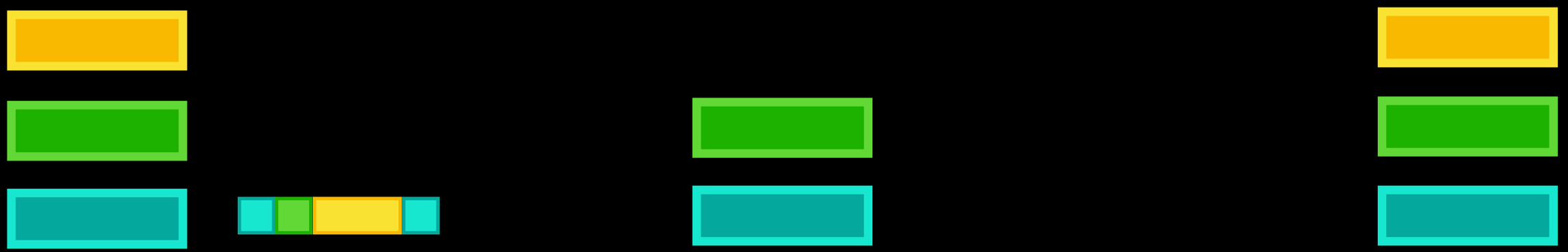
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



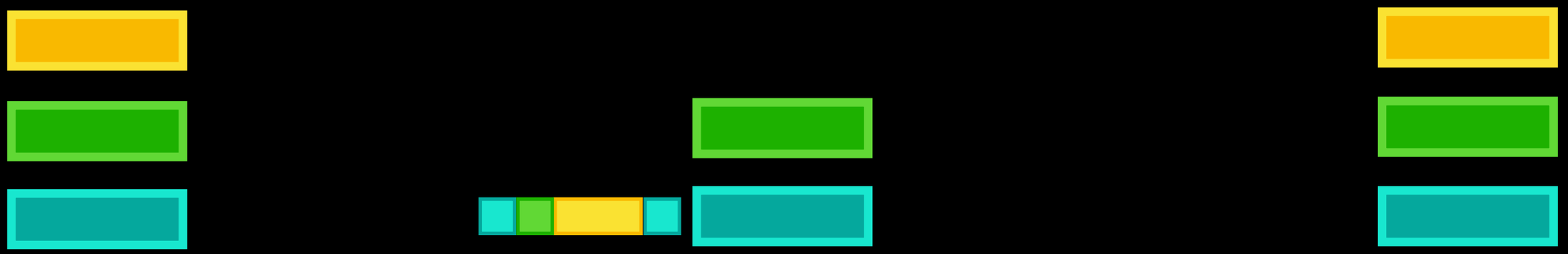
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



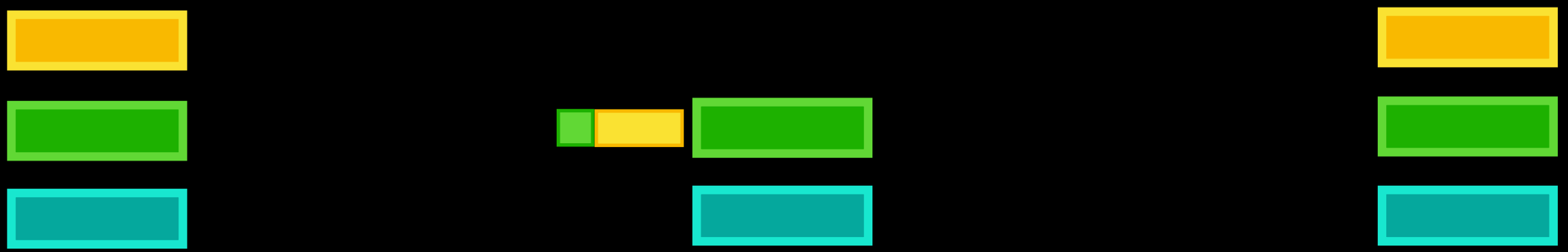
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



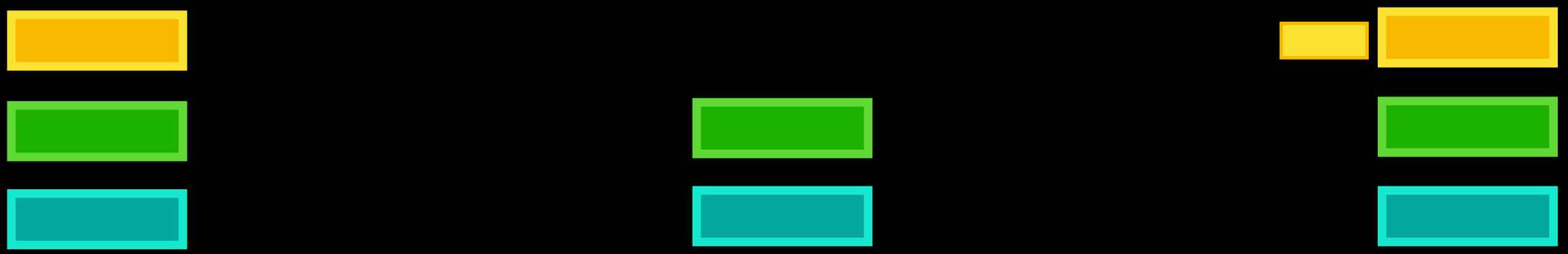
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



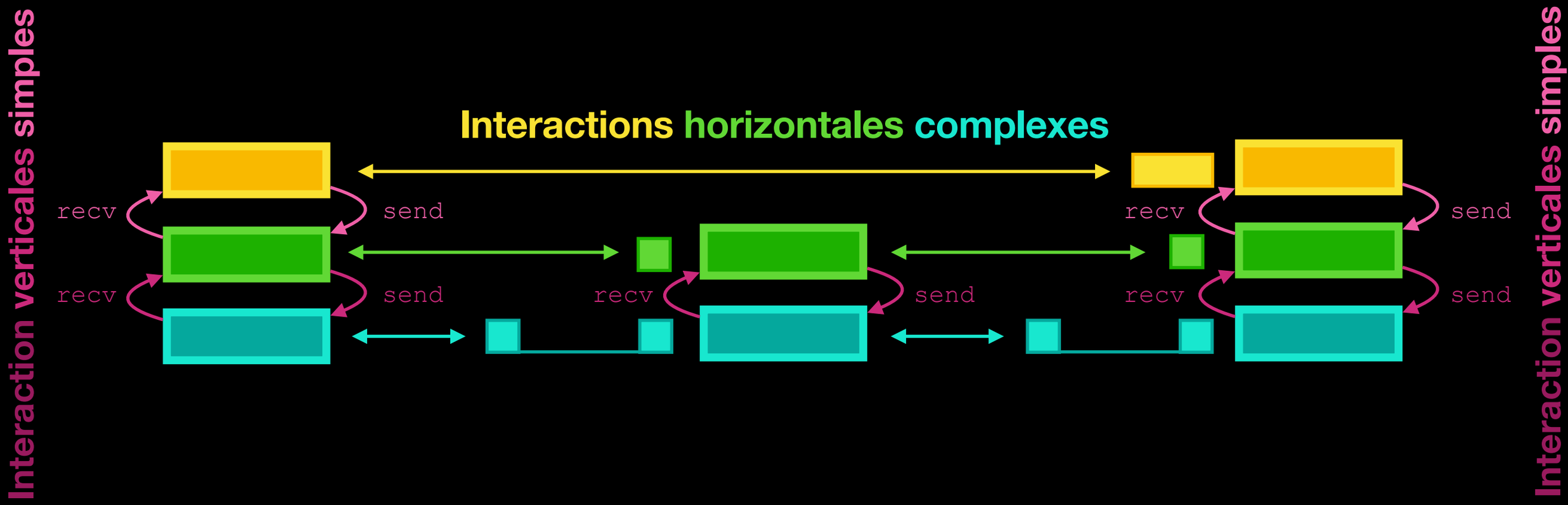
Fonctionnement d’une couche

Interactions verticales simples et horizontale complexe !



Fonctionnement d'une couche

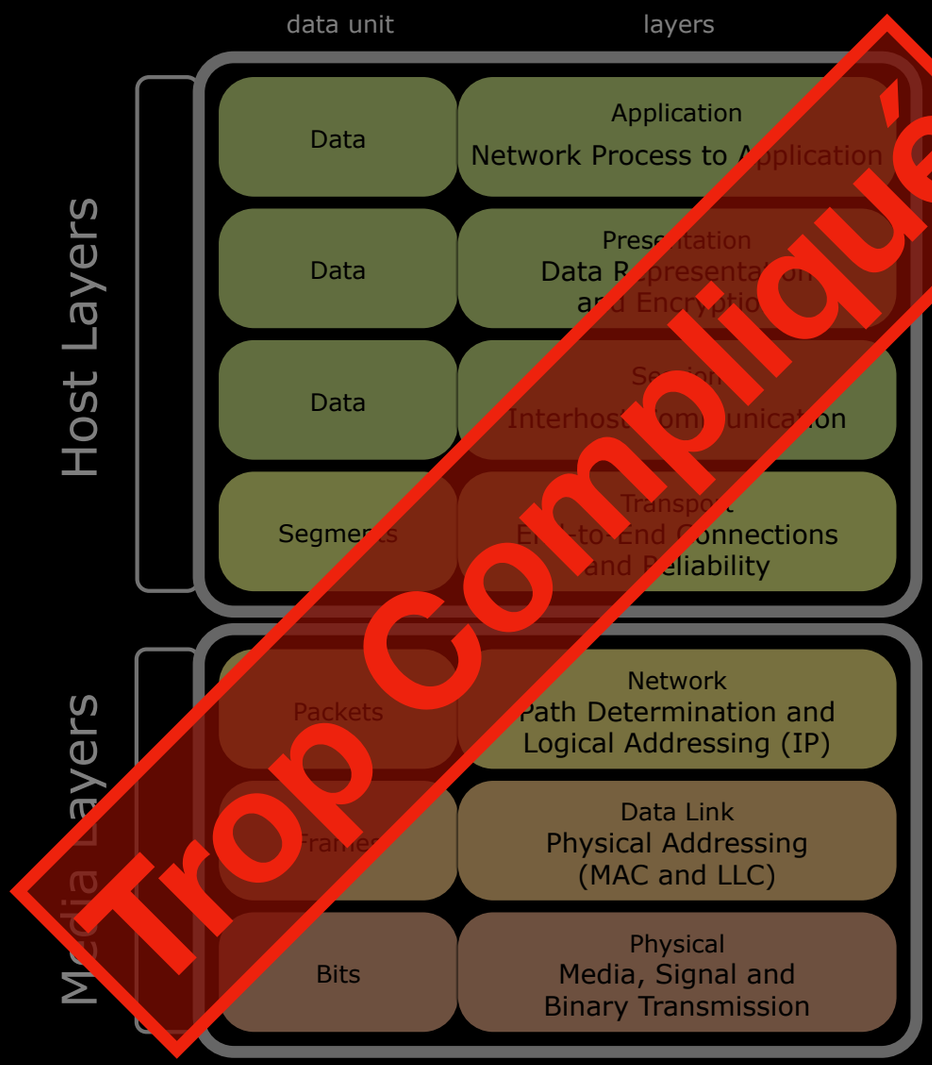
Interactions verticales simples et horizontale complexe !



Aperçu des couches (TCP / IP)

Et de la suite du cours !

Modèle OSI



Modèle TCP/IP (IETF)

Application

Transport

Network

Link

Physical

On en parlera que peu
Normes IEEE
Compliquées

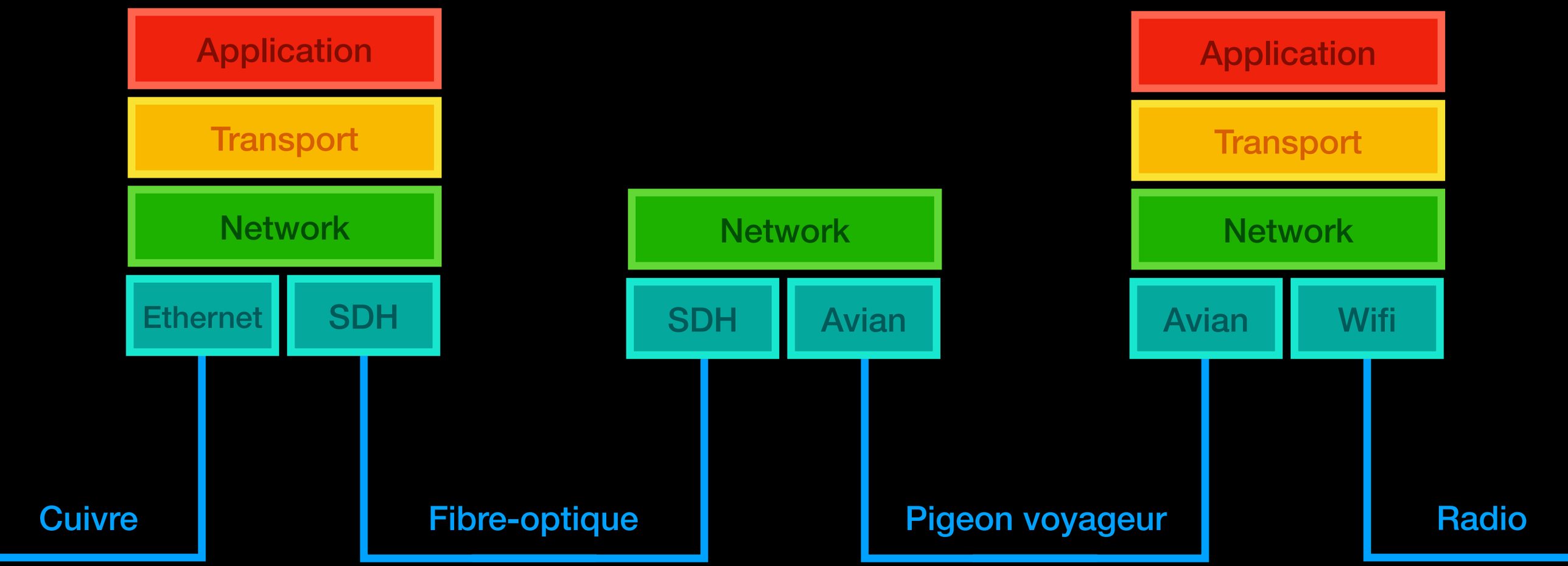
Aperçu des couches (TCP / IP)

Et de la suite du cours !

Date		Amphi	Références
17/10/2024 15h-16h30	**CM1**	**Introduction, Socket TCP**	**Kurose 1.1, 1.5, 1.7, 2.2, 2.7.2 CS:APP 11.1, 11.2, 11.4**
05/11/2024 15h-16h30	**TP1**	Initiation au Réseau, Socket TCP	
12/11/2024 15h-16h30	**CM2**	Protocoles Applicatifs over TCP, UDP, e.g HTTP, DNS	Kurose 2.1 à 2.4, 2.7 CS:APP 11.3, 11.5, 11.6, 11.7
12/11/2024 15h-16h30	**TP2**	Socket TCP et UDP	
14/11/2024 15h-16h30	**CM3**	Protocoles de Transport	Kurose Ch 3, CS:APP 11.3
19/11/2024 15h-16h30	**TP3**	TP (noté): Reliable Data Transport	
21/11/2024 15h-16h30	**CM4**	TCP suite et fin, protocole IP, notion de Lien	Kurose Ch 3, 4.1, 4.3, 6.1
26/11/2024 15h-16h30	**TP4**	TP (noté): Reliable Data Transport (2)	
28/11/2024 15h-16h30	**CM5**	Forwarding + Routage	Kurose 4.1, 4.3, 5.1, 5.2
03/12/2024 15h-16h30	**TP5**	TD : IP + Routage	
05/12/2024 15h-16h30	**CM6**	Routage + (un peu plus de lien)	Kurose 5.1, 5.2, 5.4 (6)
10/12/2024 15h-16h30	**TP6**	TD : IP + Routage	
12/12/2024 15h-16h30	**CM7**	Conclusion : Les défis d'internet	Divers sections du Kurose
18/12/2024 9h-11h(40)	**Ex.**	Examen Final	Tout ce qu'il y a au dessus

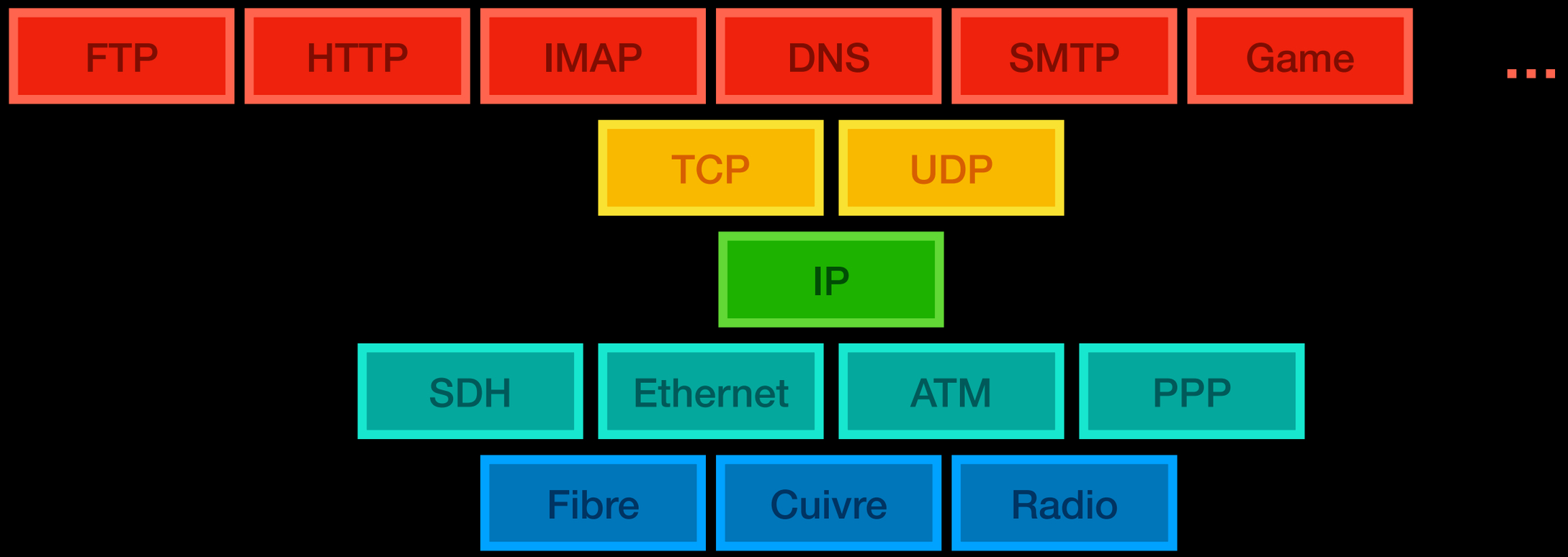
Modèle en sablier

Masquer la diversité matérielle



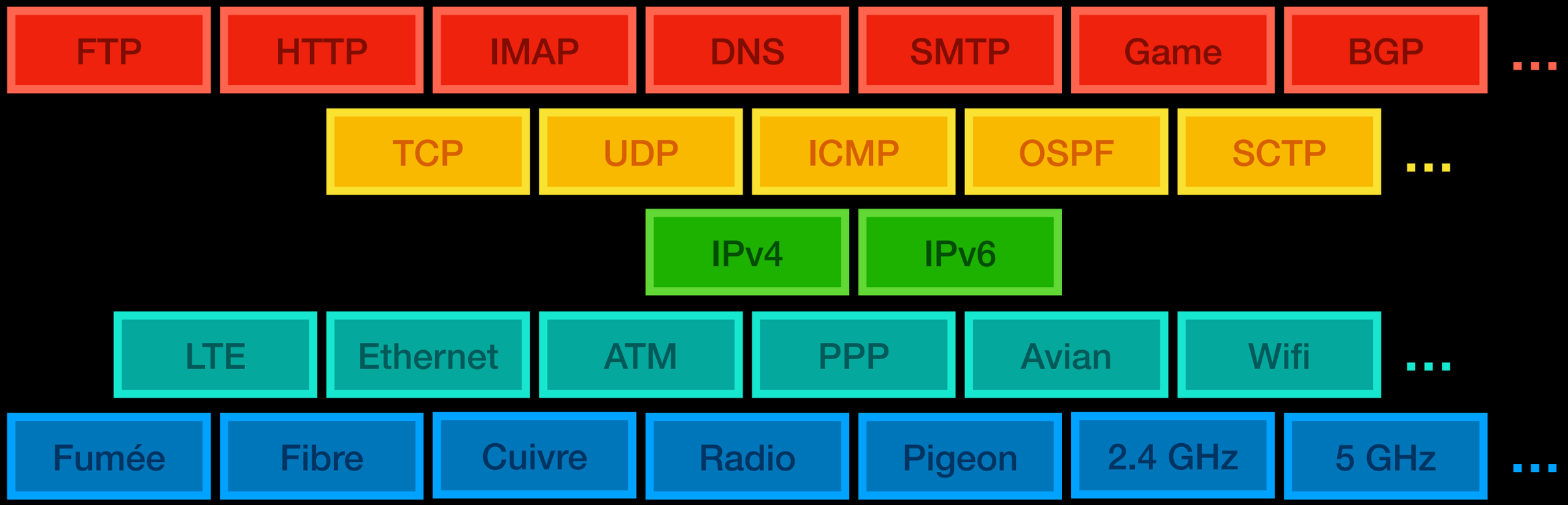
Le Modèle en Sablier

Le protocole IP est au cœur d'internet



Le Modèle en Sablier

Enfin, les protocoles IP

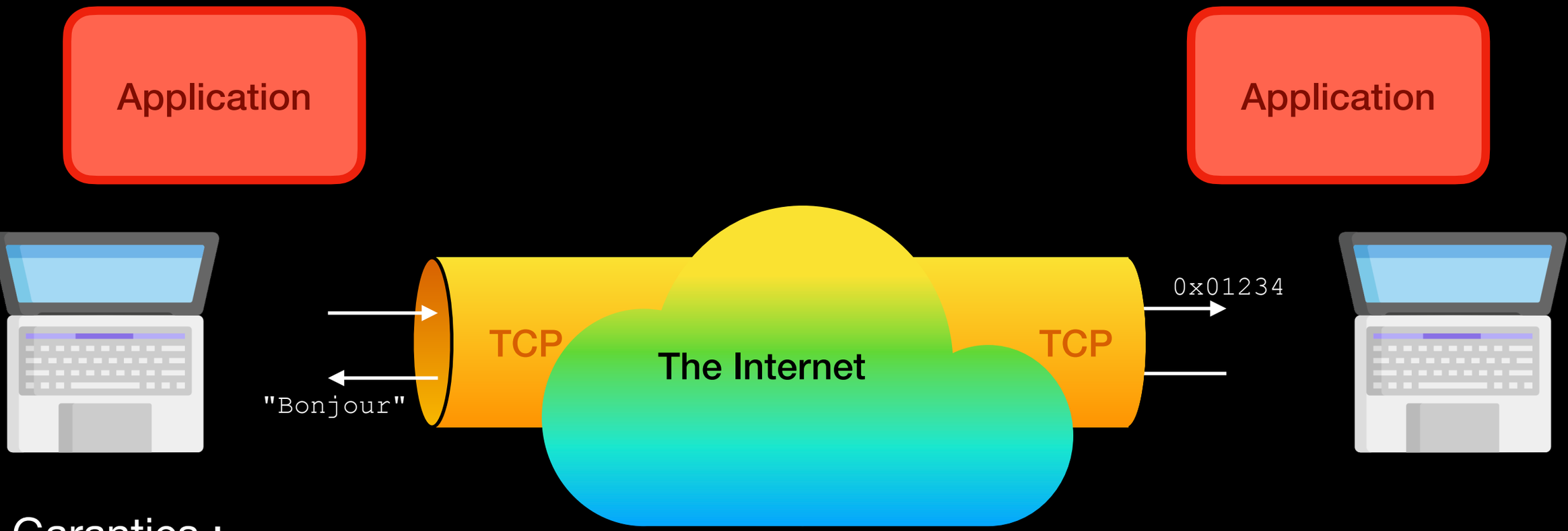


Utiliser les socket TCP

Votre premier programme réseau

Socket, quésaco ?

Un pipeline à octet



Garanties :

- Tous les octets arrivent dans l’ordre, sans duplication ni perte
- Pas de garantie de délais, s’il y a un bouchon, ça peut arriver lentement, et après longtemps
- Détection de corruption accidentelle

D’où ça vient

Quelques éléments d’historiques

1983
BSD4.2

Aujourd’hui :
Interface Omniprésente
(natif ou sur-couche)

Utiliser une socket

Envoyer et recevoir

Une socket est représenté par un file descriptor,
i.e. un index dans une table du système, d'ou `int socket_fd`

```
#include <unistd.h>
ssize_t read(int socket_fd, void *buf, size_t nbyte);          man 2 read

                                                               man 2 recv
#include <sys/socket.h>
ssize_t recv(int socket_fd, void *buffer, size_t length, int flags);
```

Attention aux Short Counts et erreurs !

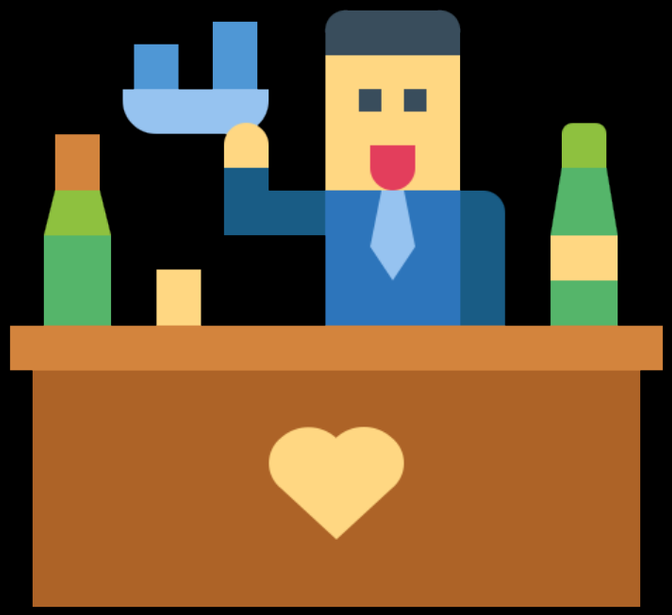
```
#include <unistd.h>
ssize_t write(int socket_fd, const void *buf, size_t nbyte);   man 2 write

                                                               man 2 send
#include <sys/socket.h>
ssize_t send(int socket_fd, const void *buffer, size_t length, int flags);
```


CS:APP a une API alternative, rio_ pour éviter ce problème

Ouvrir une socket

Client et Serveur

1 Ouverture du Bar



2 Bonjour, je voudrais un cidre

3 Bonjour, voici votre cidre

Serveur
Attend des clients

Client
Arrive (et demande du cidre)

Ouvrir une socket - Client

Easy mode with csapp.h / csapp.c

Thanks Dave !

e.g. "ens-rennes.fr"

e.g. "80" pour http

```
int open_clientfd(const char *hostname, const char *port); // in csapp.h
```

```
int clientfd = open_clientfd(hostname, port);
if (clientfd < 0) {
  // Handle errors (-2 address info error, -1 the socket itself failed)
}

// talk to the server using clientfd, a now connected socket

close(clientfd); // Say good-bye
```

Ouvrir une socket - Serveur

Easy mode with csapp.h / csapp.c

Thanks Dave !

```
// in csapp.h
int open_listenfd(const char *port);
// in sys/socket.h
int accept(int socket,
           struct sockaddr *restrict addr,
           socklen_t *restrict addr_len
           );
```

e.g. "80" pour http

Adresse du client

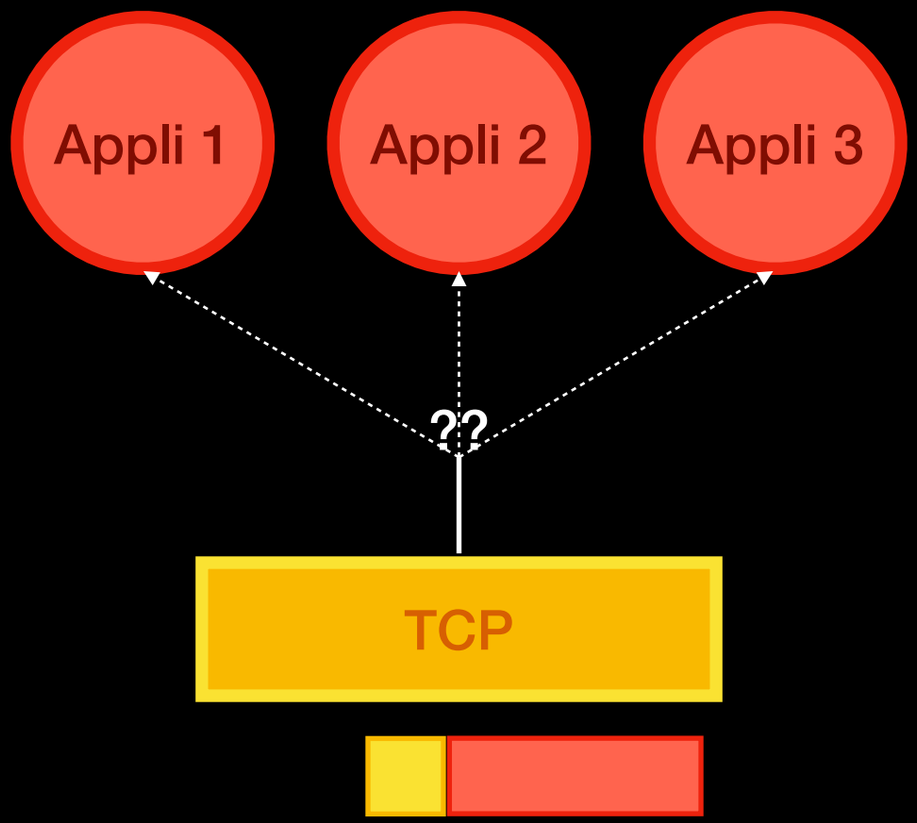
Longueur de l'adresse

man 2 accept

```
int listenfd = -1, clientfd = -1;
listenfd = open_listenfd(port); // "Open the bar"
// error handling not shown

socklen_t clientlen;
struct sockaddr_storage clientaddr; // Enough space for any address

while ((int clientfd = accept(listenfd, &clientaddr, &clientlen)) > 0) {
                                    // pointers can also be both NULL
  // clientfd is now a connected TCP socket
  // Talk with the client using read/write on clientfd

  // You could spawn a separate thread
  close(clientfd); // Don't forget to say good-bye
}
close(listenfd); // Stop taking new clients.
```

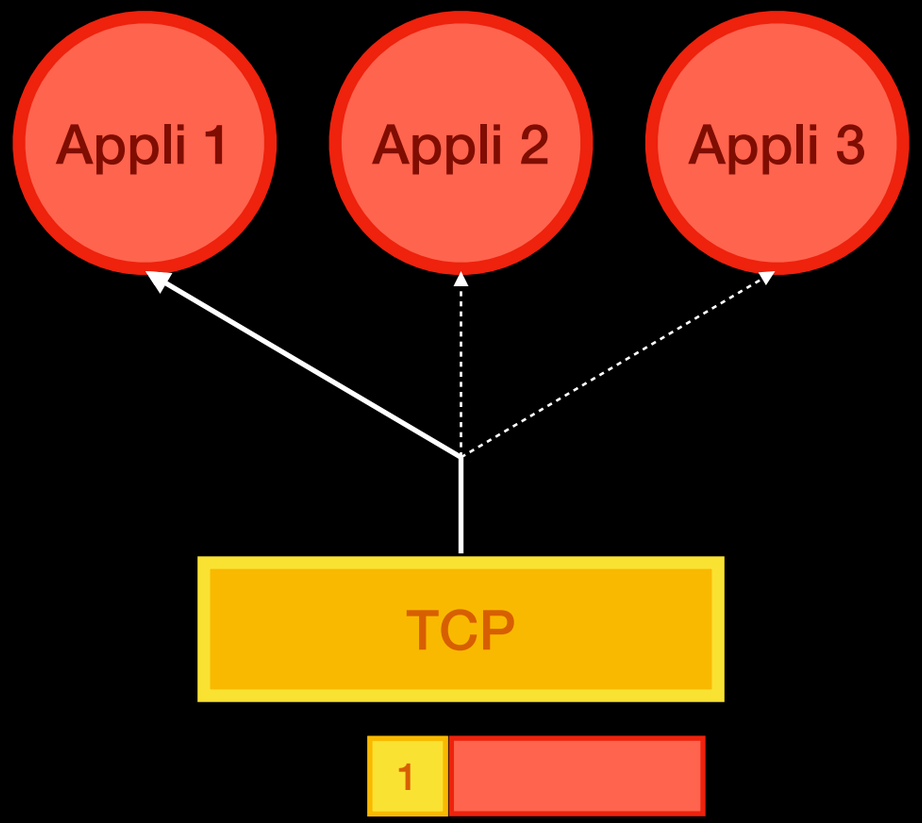
Pourquoi un numéro de port

Le dé-multiplexage



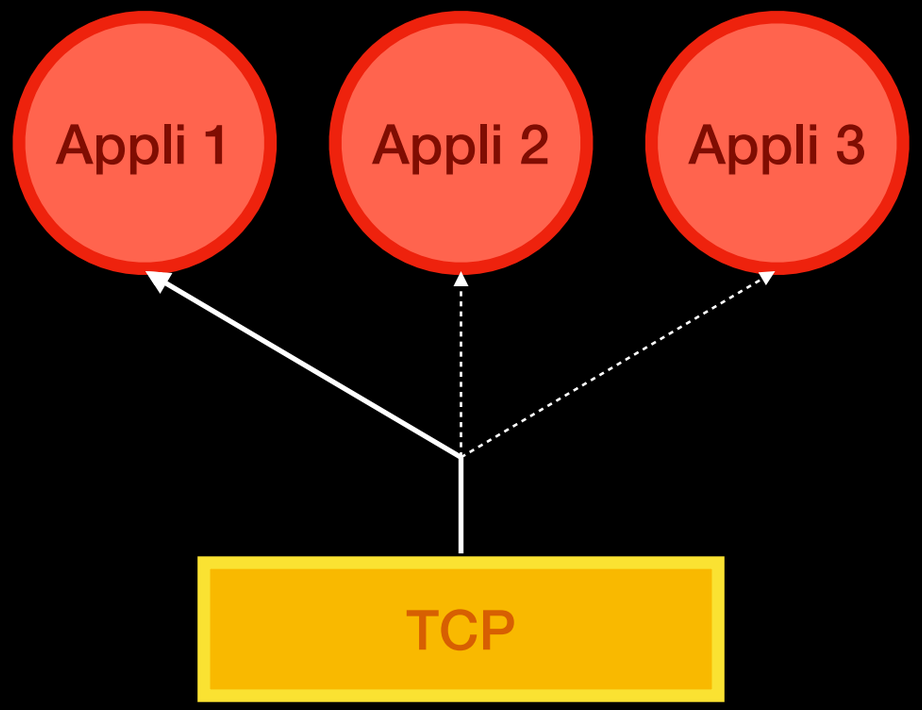
Pourquoi un numéro de port

Le dé-multiplexage



Pourquoi un numéro de port

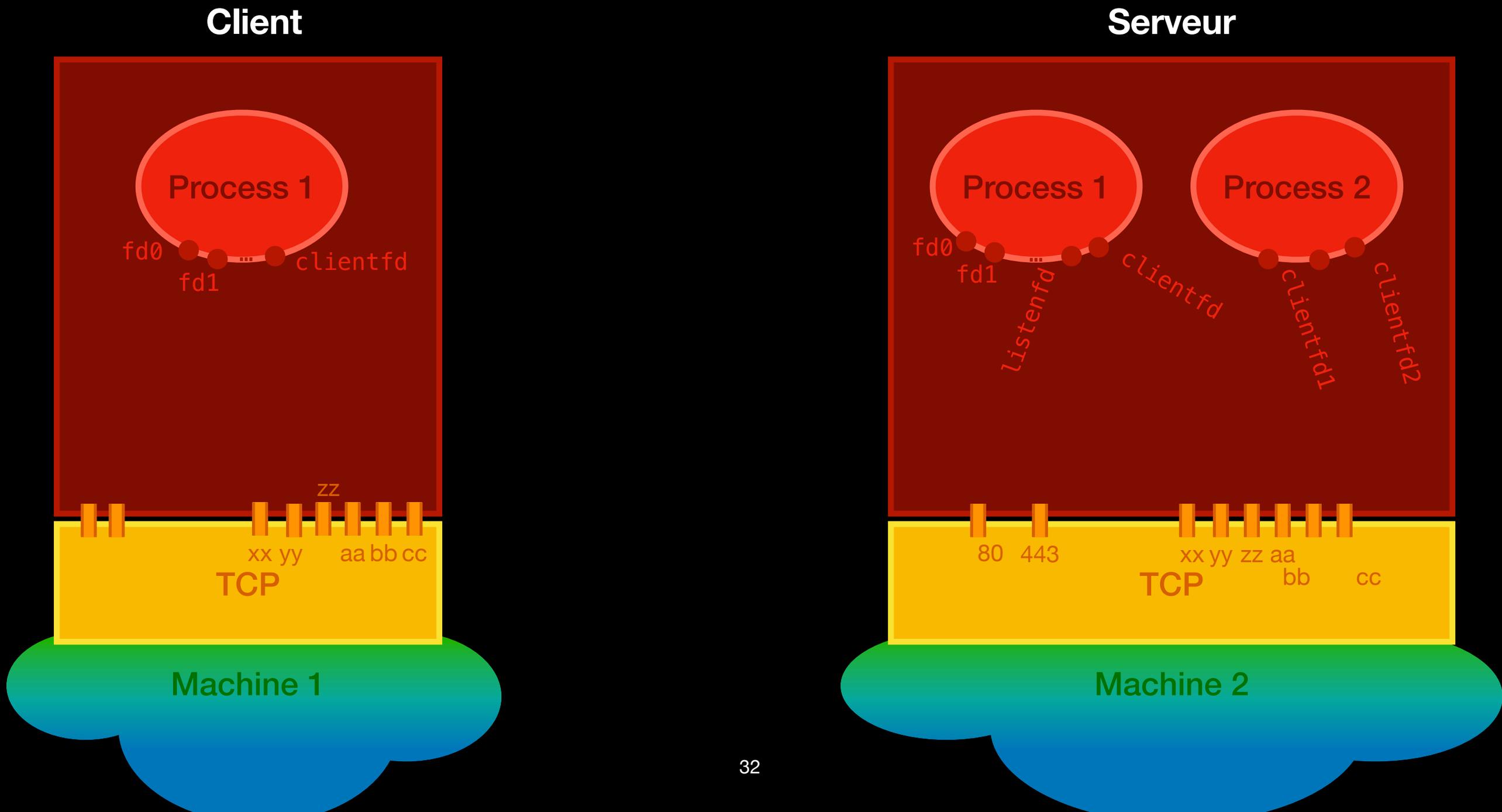
Le dé-multiplexage



Le numéro de port permet le dé-multiplexage à l'arrivée

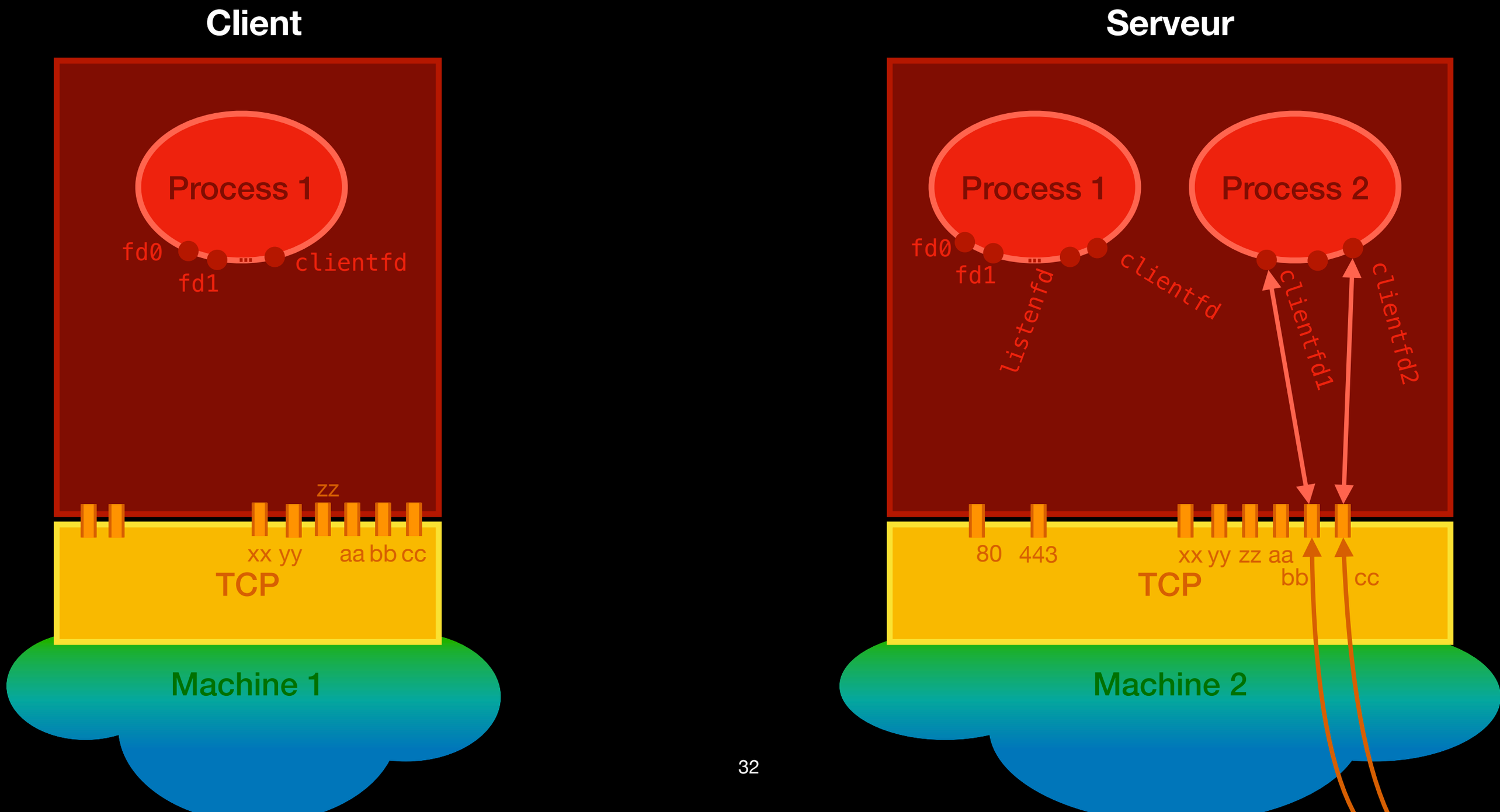
Des serveurs et des ports

Que se passe-t-il quand je fais toc-toc



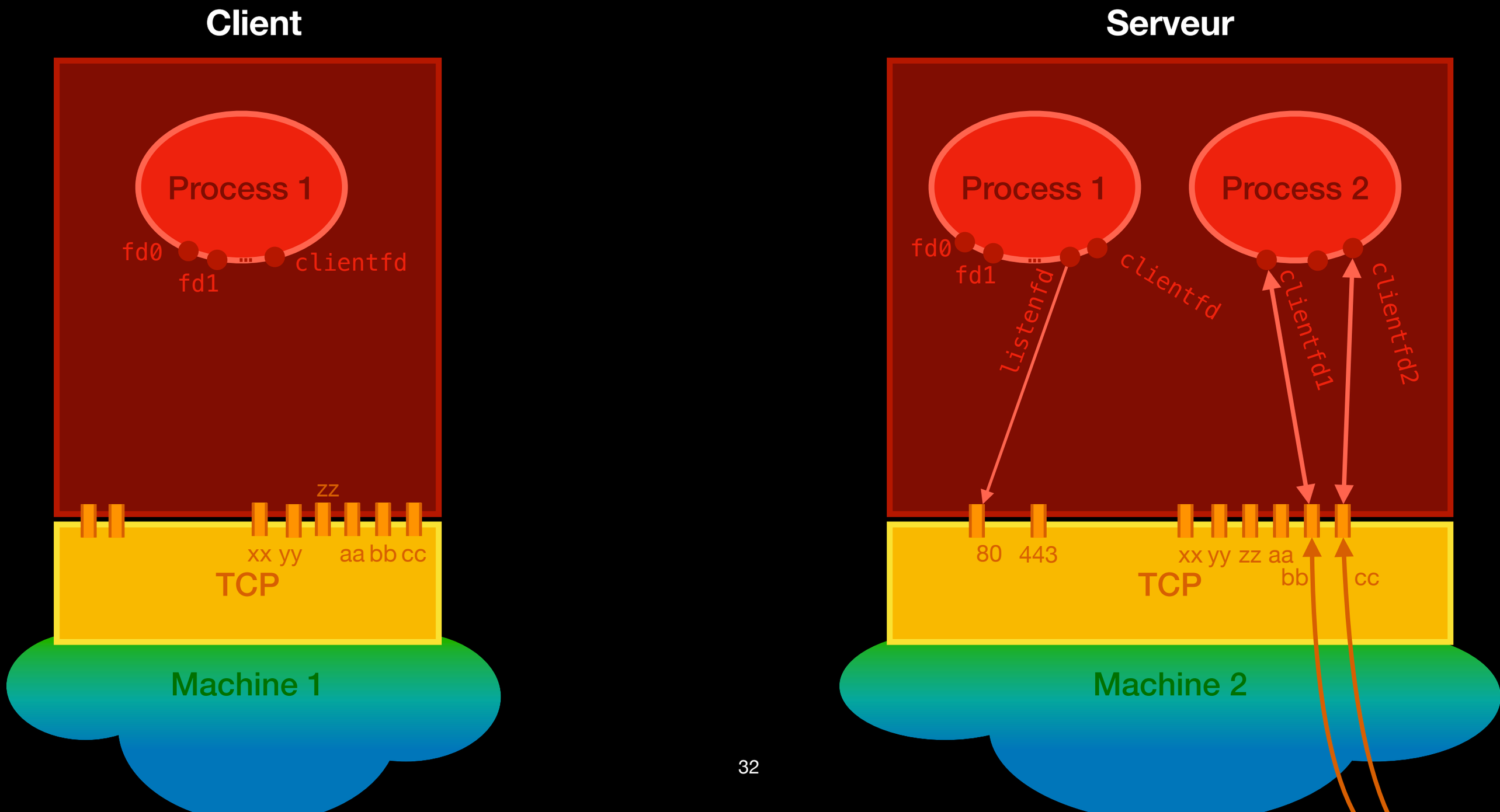
Des serveurs et des ports

Que se passe-t-il quand je fais toc-toc



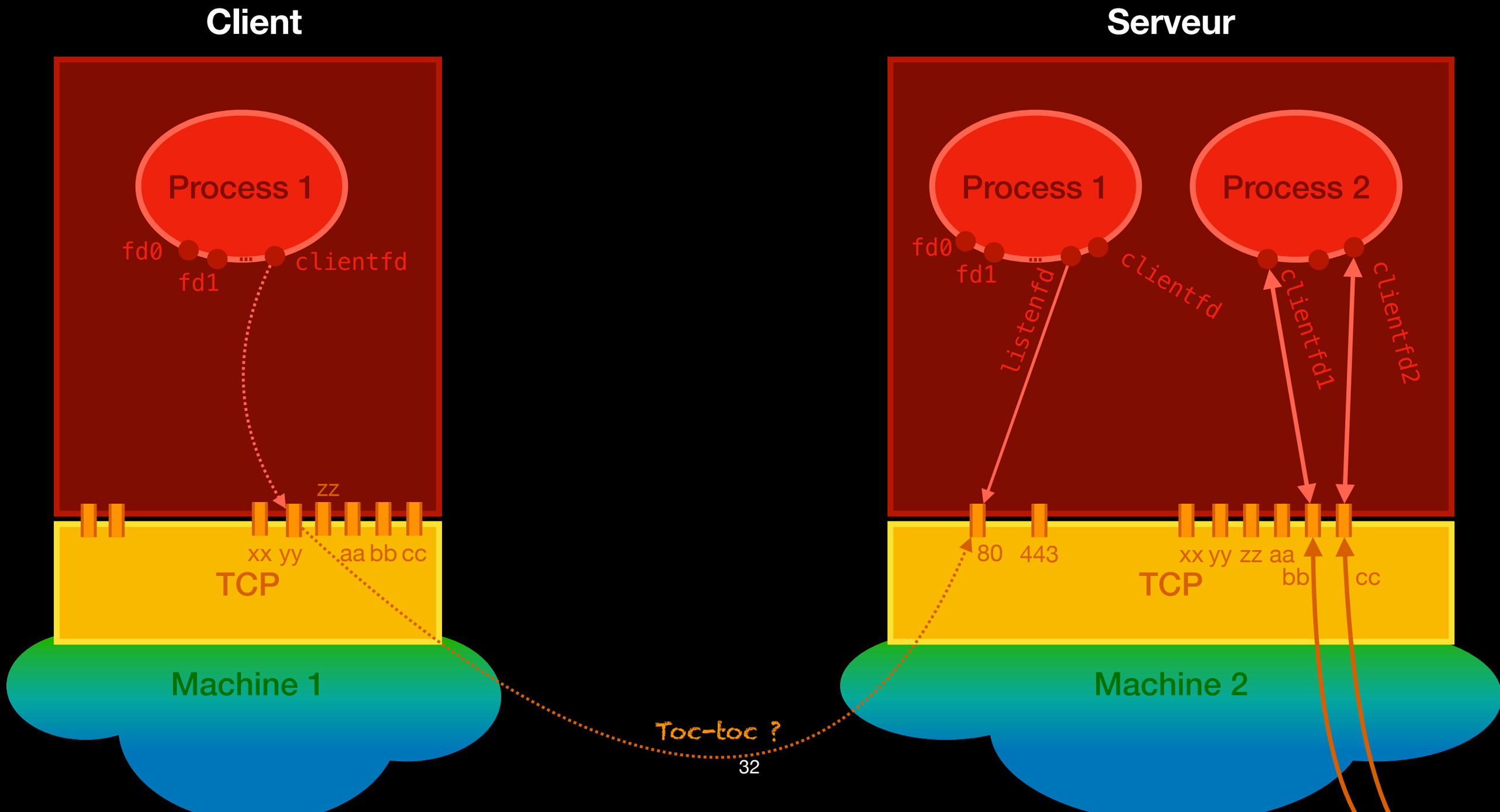
Des serveurs et des ports

Que se passe-t-il quand je fais toc-toc



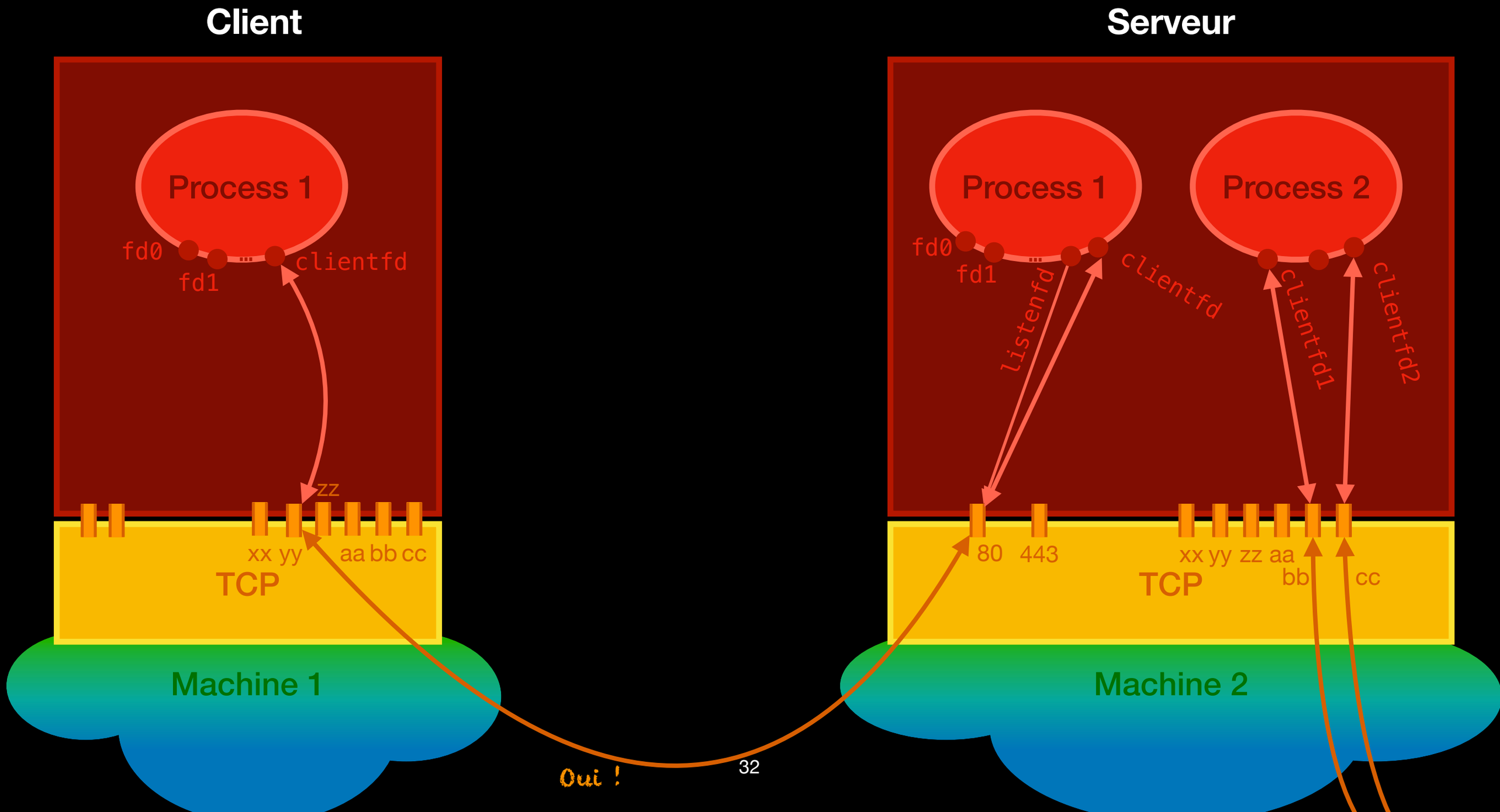
Des serveurs et des ports

Que se passe-t-il quand je fais toc-toc



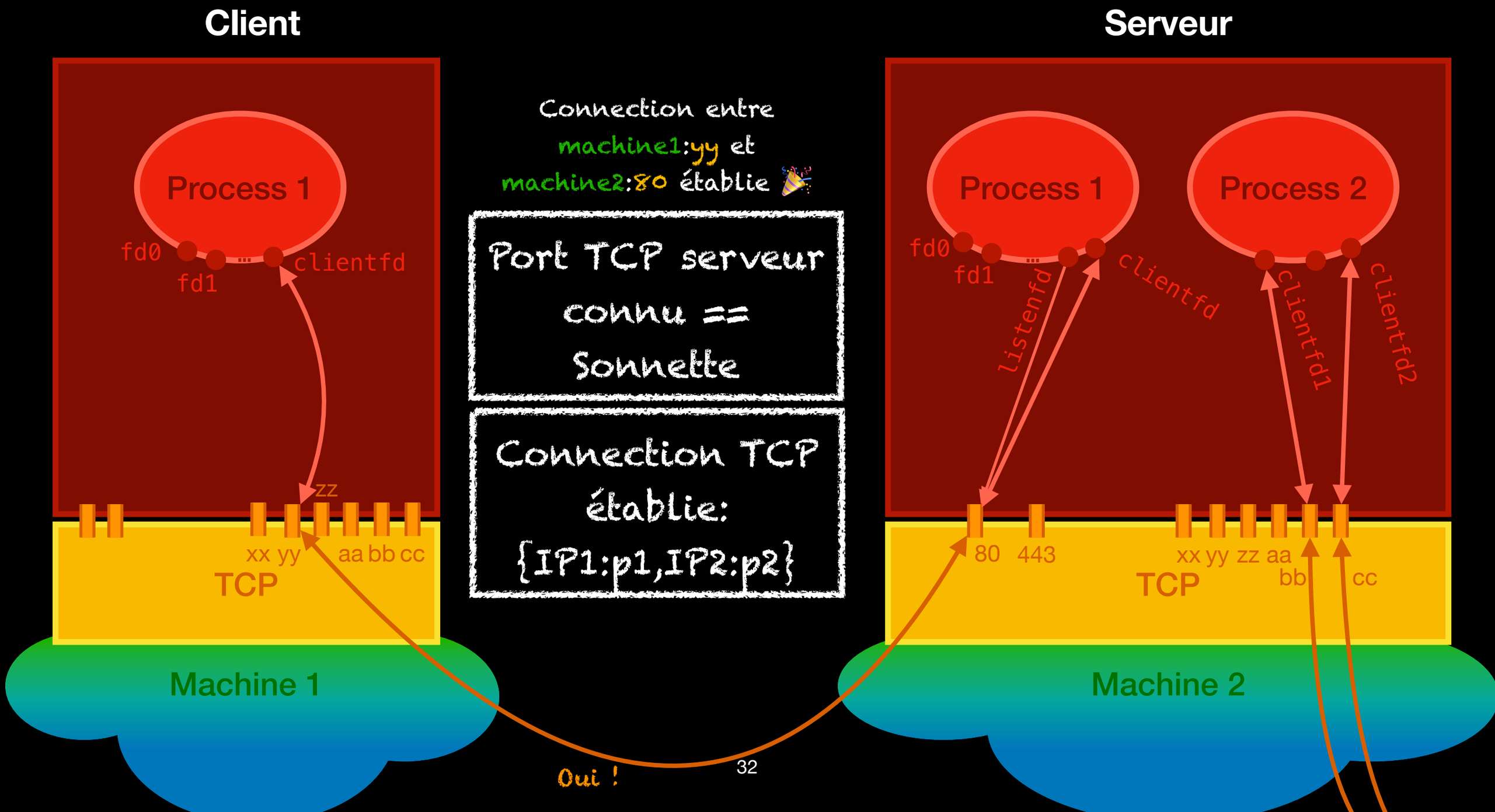
Des serveurs et des ports

Que se passe-t-il quand je fais toc-toc



Des serveurs et des ports

Que se passe-t-il quand je fais toc-toc



HTTP

Comment demander un site internet

```
GET / HTTP/1.0\r\n
Host: ens-rennes.fr\r\n
\r\n
```


```
HTTP/1.1 200 OK\r\n
…\r\n
\r\n
```

```
<!DOCTYPE html>
<!--[if lte IE 7]> <html class="ie7 oldie no-js" xmlns="http://www.w3.org/
1999/xhtml" lang="fr" xml:lang="fr"> <![endif]-->
<!--[if IE 8]> <html class="ie8 oldie no-js" xmlns="http://www.w3.org/1999/
xhtml" lang="fr" xml:lang="fr"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" xmlns="http://www.w3.org/1999/xhtml" lang="fr"
      xml:lang="fr"> <!--<![endif]-->
<head>
…
```

Anatomie d'une requête simple

HTTP / 1.0

```
GET /SYS1-Reseau-Syllabus.pdf HTTP/1.0\r\n
Host: sys1.guillaumedidier.fr\r\n
Connection: Close\r\n
\r\n
```

Fins de ligne CRLF

METHOD: GET, POST, ... (c.f. RFC)

```
METHOD /path HTTP/1.0\r\n
Header: Value\r\n
...
Header: Value\r\n
\r\n
<Data if applicable>
```

Path débute toujours par /

En-tête Host obligatoire,
"Connection: Close" peut être utile

Une ligne vide signale la fin des entêtes

e.g. pour PUT / POST

Anatomie d'une réponse

HTTP / 1.1 Le serveur vous répondra en 1.1 s'il peut

```
HTTP/1.1 308 Permanent Redirect\r\n
Content-Type: text/html\r\n
Content-Length: 171\r\n
Connection: close\r\n
Location: https://sys1.guillaumedidier.fr/SYS1-Reseau-Syllabus.pdf\r\n
\r\n
<html data …>
```

```
HTTP/1.1 200 OK\r\n
Server: nginx/1.26.2\r\n
Date: Tue, 08 Oct 2024 08:21:14 GMT\r\n
Content-Type: application/pdf\r\n
Content-Length: 121895\r\n
Last-Modified: Sat, 05 Oct 2024 17:00:15 GMT\r\n
Connection: close\r\n
ETag: "6701709f-1dc27"\r\n
Accept-Ranges: bytes\r\n
\r\n
<pdf data …>
```

Réponse sur le port 8192
de sys1.guillaumedidier.fr

```
HTTP/x.y STATUS_CODE Message\r\n
Header: Value\r\n
...
Header: Value\r\n
\r\n
<Data if applicable>
```

Questions ?

S’il vous en vient plus tard:
E-mail à l’équipe (3 personnes),
Office Hours les jeudi 18h-19h sur discord

Résumé

What did we see today?

- Qu’est-ce qu’un réseau, qu’est-ce qu’Internet ?
- La notion de protocole, l’approche en couche
- Comment utiliser une socket TCP
- Comment demander une page web en HTTP
- Vous avez ce qu’il faut pour le TP - sujet dispo d’ici les vacances

Est-ce que vos notes ont tout ça ?

La prochaine fois

Teaser

- Demander une page web à un serveur - HTTP en détails
- TCP vs UDP
- Comment résoudre des nom sur internet - le protocole DNS
- Quelques histoires de Mails

There are 2 hard problems in computer science:
Naming things,
cache invalidation,
and off-by one errors.

DNS faces both of them

Bonnes Vacances !

Et bon courage pour la semaine de projet

I LIED

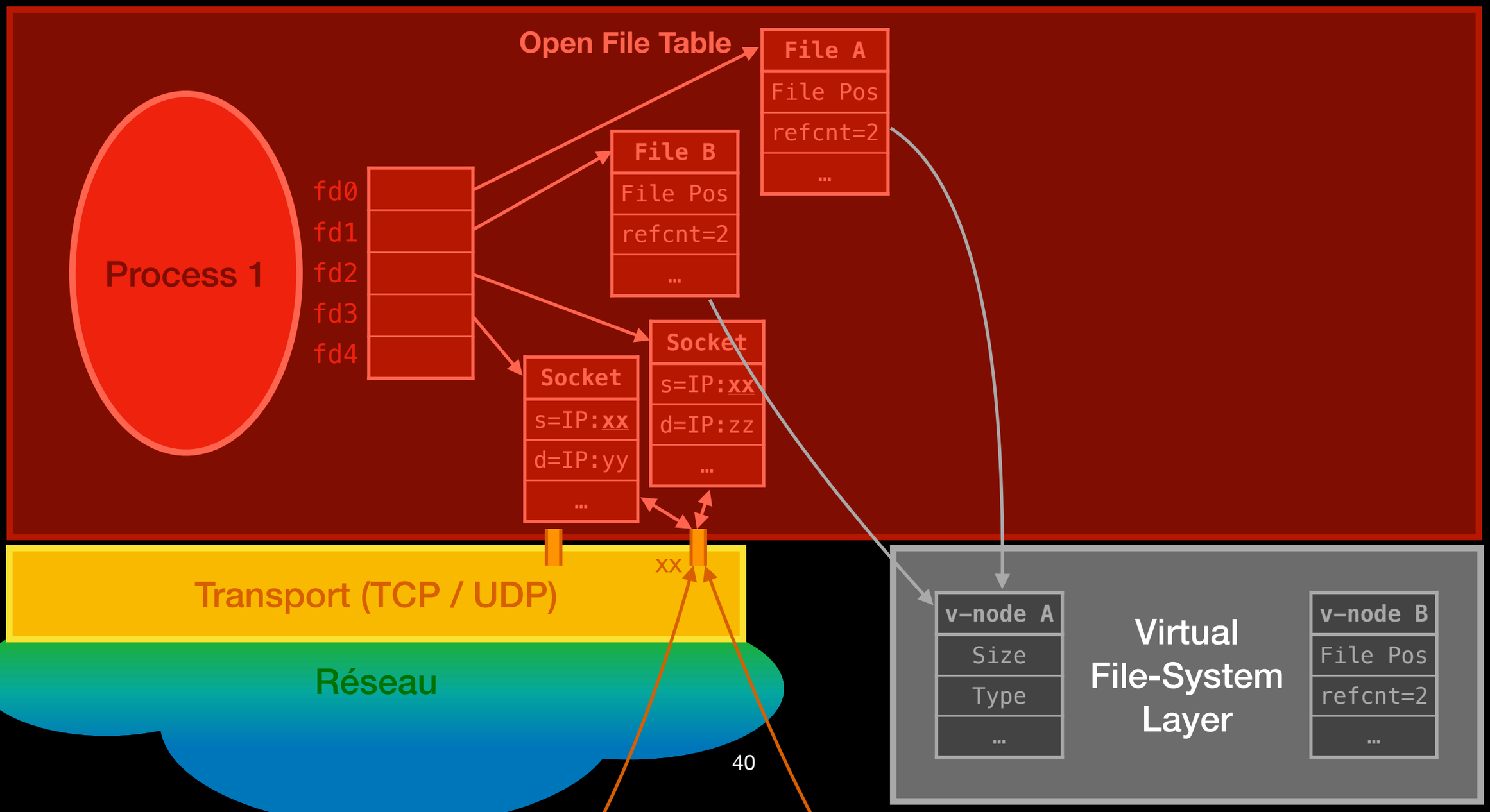
Ma représentation des ports et sockets est fausse

There's an extra level of indirection

I LIED

Ma représentation des ports et sockets est fausse

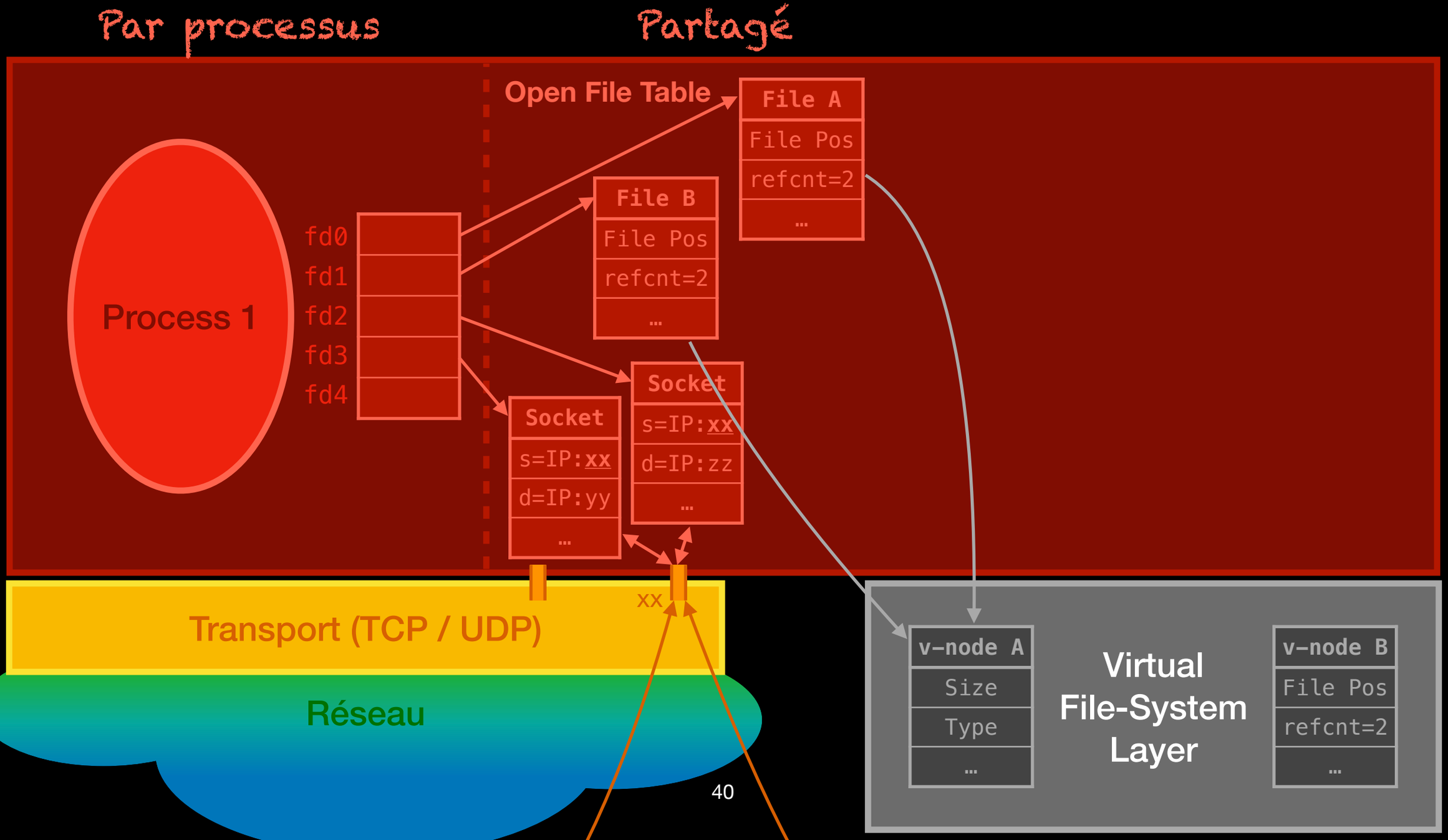
There's an extra level of indirection



I LIED

Ma représentation des ports et sockets est fausse

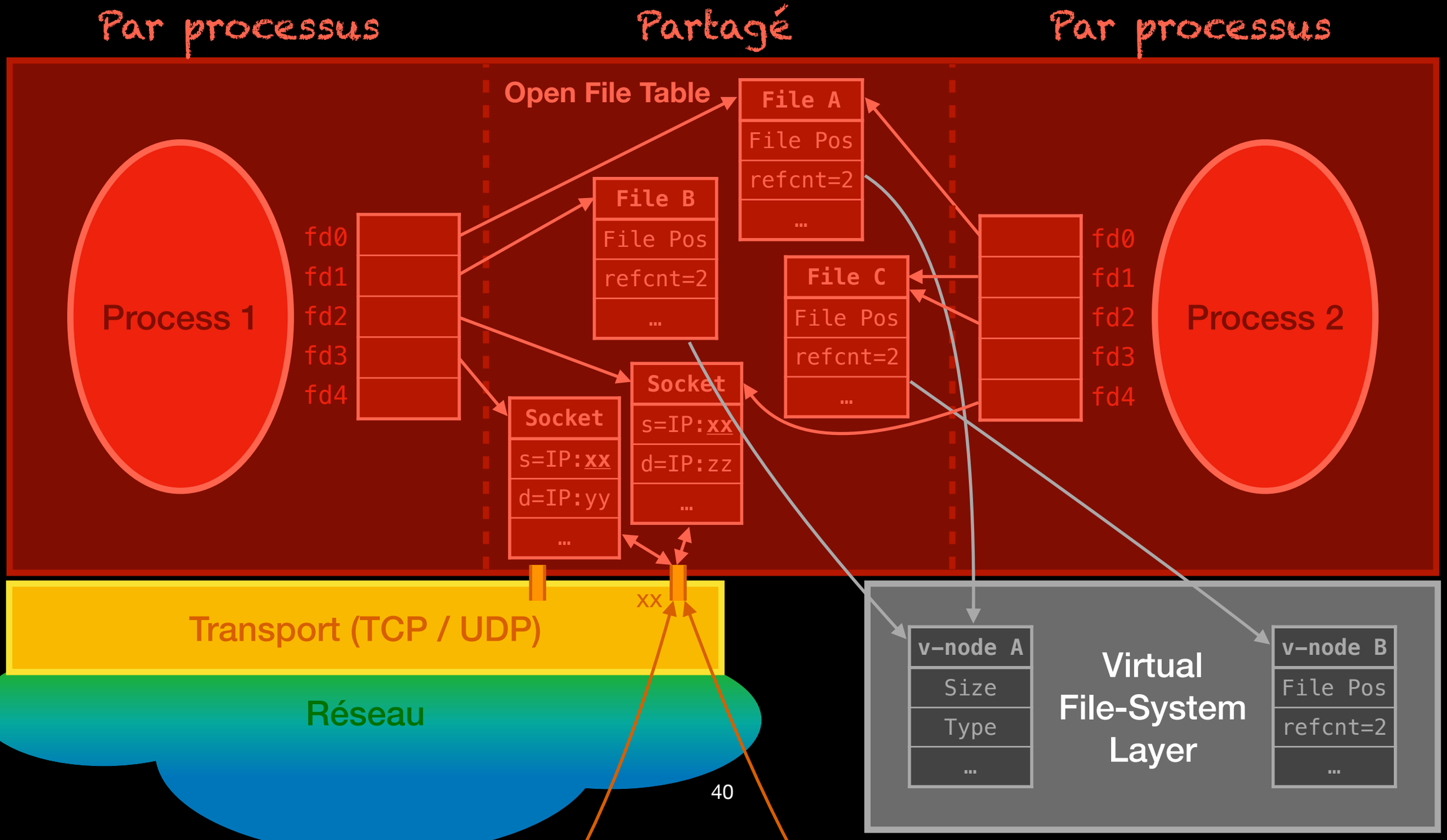
There's an extra level of indirection



I LIED

Ma représentation des ports et sockets est fausse

There's an extra level of indirection



Ouvrir une socket

Les morceaux sous le capot - 1. getaddrinfo
Comment je contact https://ens-rennes.fr/

Résoudre des noms

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int getaddrinfo(const char *hostname, const char *servname, const struct addrinfo *hints,
                struct addrinfo **res);

void freeaddrinfo(struct addrinfo *ai);
```

e.g. "ens-rennes.fr"

e.g. "http" ou "80"

Contraintes (TCP, options, etc)

Liste chaînée, à dé-allouer après

Ouvrir une socket

Les morceaux sous le capot - 1. getaddrinfo
Comment je contact https://ens-rennes.fr/

Résoudre des noms

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int getaddrinfo(const char *hostname, const char *servname, const struct addrinfo *hints,
                struct addrinfo **res);

void freeaddrinfo(struct addrinfo *ai);
```

e.g. "ens-rennes.fr"

e.g. "http" ou "80"

Contraintes (TCP, options, etc)

Liste chaînée, à dé-allouer après

```c
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Ouvrir une socket

Les morceaux sous le capot - 1. getaddrinfo
Comment je contact https://ens-rennes.fr/

Résoudre des noms

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int getaddrinfo(const char *hostname, const char *servname, const struct addrinfo *hints,
                struct addrinfo **res);

void freeaddrinfo(struct addrinfo *ai);
```

e.g. "ens-rennes.fr"

e.g. "http" ou "80"

Contraintes (TCP, options, etc)

Liste chaînée, à dé-allouer après

```c
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

man 3 getaddrinfo

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

(Serveur) L'attacher à une adresse et écouter

```
#include <sys/socket.h>

int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);
```

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

(Serveur) L'attacher à une adresse et écouter

```
#include <sys/socket.h>

int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);
```

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

(Serveur) L'attacher à une adresse et écouter

```
#include <sys/socket.h>

int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);
```

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

(Serveur) L'attacher à une adresse et écouter

```
#include <sys/socket.h>

int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);

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

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

(Serveur) L'attacher à une adresse et écouter

```
#include <sys/socket.h>

int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);
```

man 2 bind

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

man 2 listen

Taille de la file d'attente

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

(Serveur) L'attacher à une adresse et écouter

```
#include <sys/socket.h>

int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);
```

man 2 bind

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

man 2 listen

Taille de la file d'attente

(Client) Contacter un serveur

```
#include <sys/socket.h>

int connect(int sockfd,
            const struct sockaddr *addr,
            socklen_t addrlen);
```

Ouvrir une socket

Les morceaux sous le capot 2. Ouvrir les sockets

getaddrinfo

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

Créer une socket (sans adresse)

man 2 socket

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

(Serveur) L'attacher à une adresse et écouter

```
#include <sys/socket.h>

int bind(int sockfd,
         const struct sockaddr *addr,
         socklen_t addrlen);

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

man 2 bind

man 2 listen

Taille de la file d'attente

(Client) Contacter un serveur

```
#include <sys/socket.h>

int connect(int sockfd,
            const struct sockaddr *addr,
            socklen_t addrlen);
```

man 2 connect

man 2 setsockopt

Ouvrir une socket

Le coté client - open_clientfd(const char *hostname, const char *port)

```
int clientfd = -1, rc;  struct addrinfo hints = {0}, *listp, *p;

hints.ai_socktype = SOCK_STREAM; // Open a connection
hints.ai_flags = AI_NUMERICSERV; // ... using a numeric port arg.
hints.ai_flags |= AI_ADDRCONFIG; // Recommended for connections

// Get a list of potential server addresses
if ((rc = getaddrinfo(hostname, port, &hints, &listp)) != 0){/* Handle error */ … }

// Walk the list for one that we can successfully connect to
for (p = listp; p; p = p->ai_next) {
    // Create a socket fd
    clientfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol);
    if (clientfd < 0){ /* Socket failed, try the next */ continue; }

    // Connect to the server
    if (connect(clientfd, p->ai_addr, p->ai_addrlen) != -1){ /* Success */ break; }

    // Connect failed, try another
    if (close(clientfd) < 0){ /* close failed ARGH */ … return -1; }
}

freeaddrinfo(listp); // Clean up

if (!p) { /* All connects failed */ return -1; }

return clientfd; // The last connect succeeded - we can ask for cidre and get it !
```

Ouvrir une socket

Le coté serveur - open_listenfd(const char *port)

```
struct addrinfo hints = {0}, *listp, *p;  int listenfd = -1, rc, optval = 1;

hints.ai_socktype = SOCK_STREAM; // Accept connections
hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG; // … on any IP address
hints.ai_flags |= AI_NUMERICSERV; // if the port number is already a number

// Get a list of potential server addresses
if (( rc = getaddrinfo(NULL, port, &hints, &listp)) != 0){ /* Handle error */ … }

for (p = listp; p; p = p->ai_next){ // Walk the list for one that we can bind to
    listenfd = socket(p->ai_family, p->ai_socktype, p->ai_protocol); // Create a socket fd
    if (listenfd < 0){ /* Socket failed, try the next */ continue; }

    /* Eliminates "Address already in use" error from bind */
    setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, (const void *)&optval, sizeof(int));

    // Bind the descriptor to the address
    if (bind(listenfd, p->ai_addr, p->ai_addrlen) == 0){ /* Success */ break; }
    // not success, close and try next
    if (close(listenfd) < 0){ /* close failed */ return -1; }
}
freeaddrinfo(listp); // Clean up

if (!p) { return -1; /* No address worked */ }

// Make it a listening socket ready to accept connection requests
if (listen(listenfd, LISTENQ) < 0) { /* Error */ close(listenfd); return -1; }

return listenfd; // Call accept on listenfd to serve your clients their cidre
```

Ouvrir une socket TCP

Résumé