



Programmes coopérants



Introduction réseaux

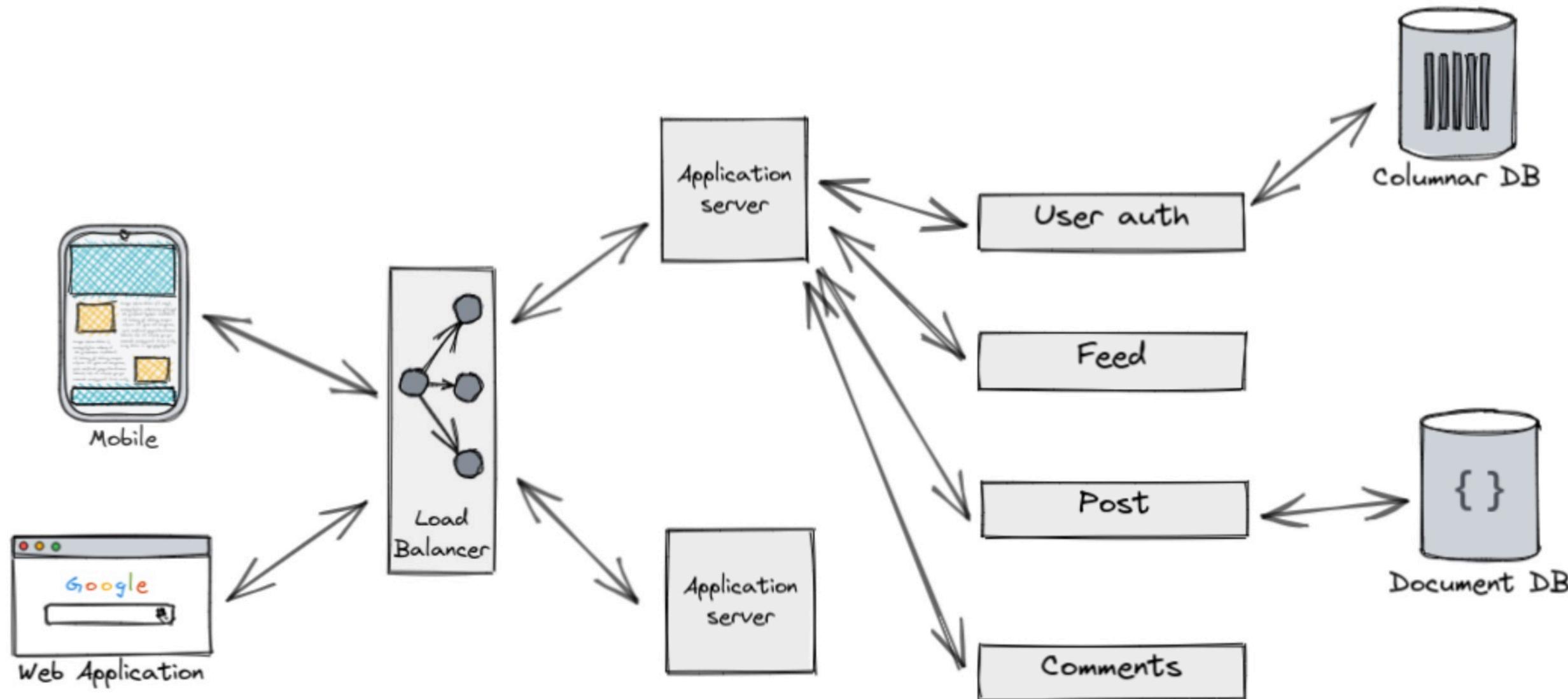


Basile Marchand¹



1 - Plateforme SISDev, Centre des Matériaux, MINES Paris - CNRS - Université PSL

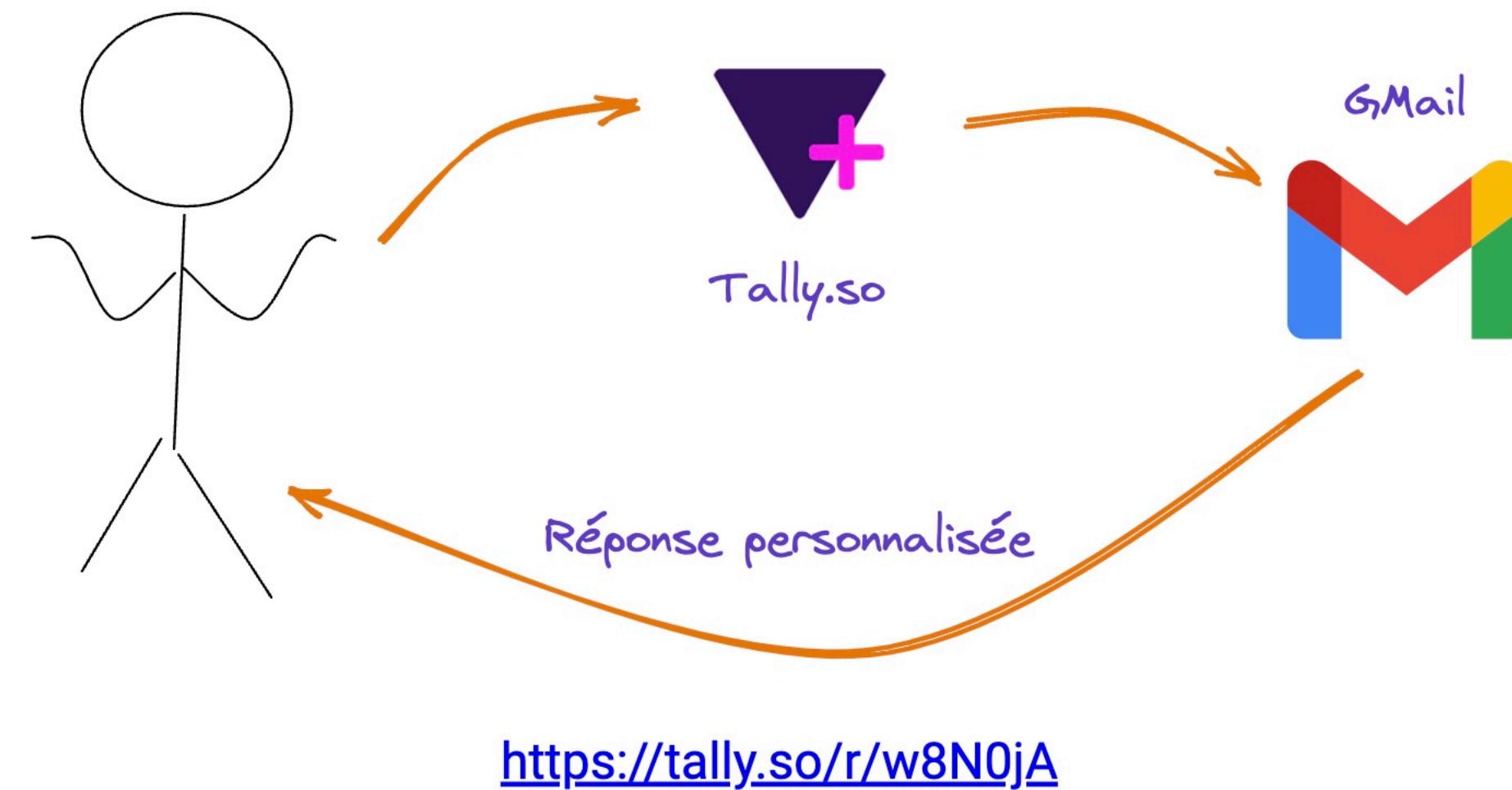
Le monde d'aujourd'hui - ultra connecté



La plupart des systèmes informatiques/services web que vous pouvez utiliser quotidiennement ne sont pas **une** application mais un **ensemble** d'application qui interagissent entre elles.

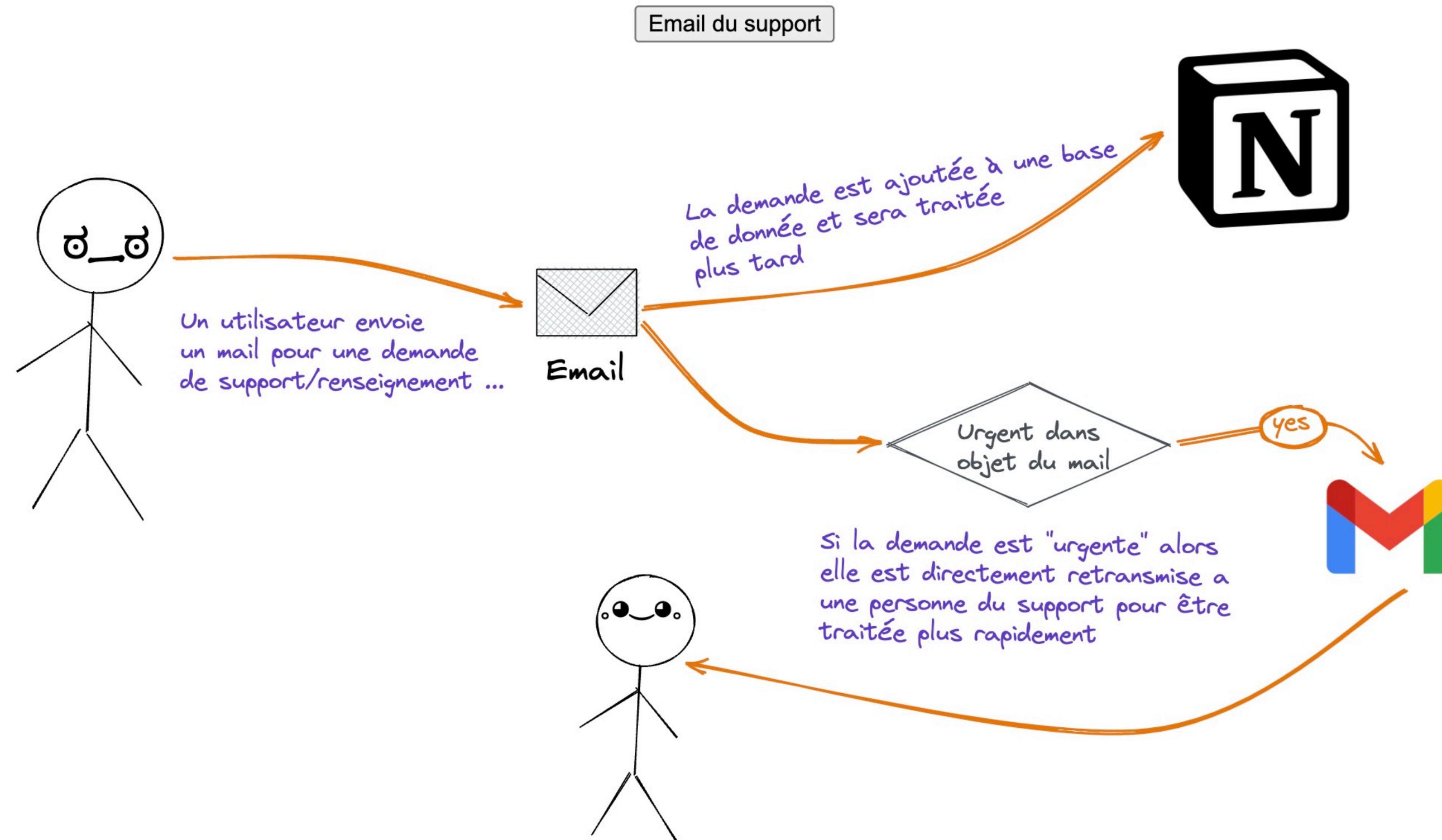
Premier Use-case

Un élève répond à un questionnaire de satisfaction en ligne et après la soumission de son questionnaire reçoit automatiquement une réponse personnalisée et spécifique selon ses réponses.



? Quels ingrédients, outils doivent être mis en oeuvre ?

Un second use case

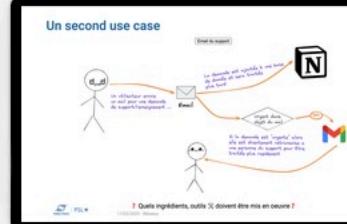




Les ingrédients nécessaires



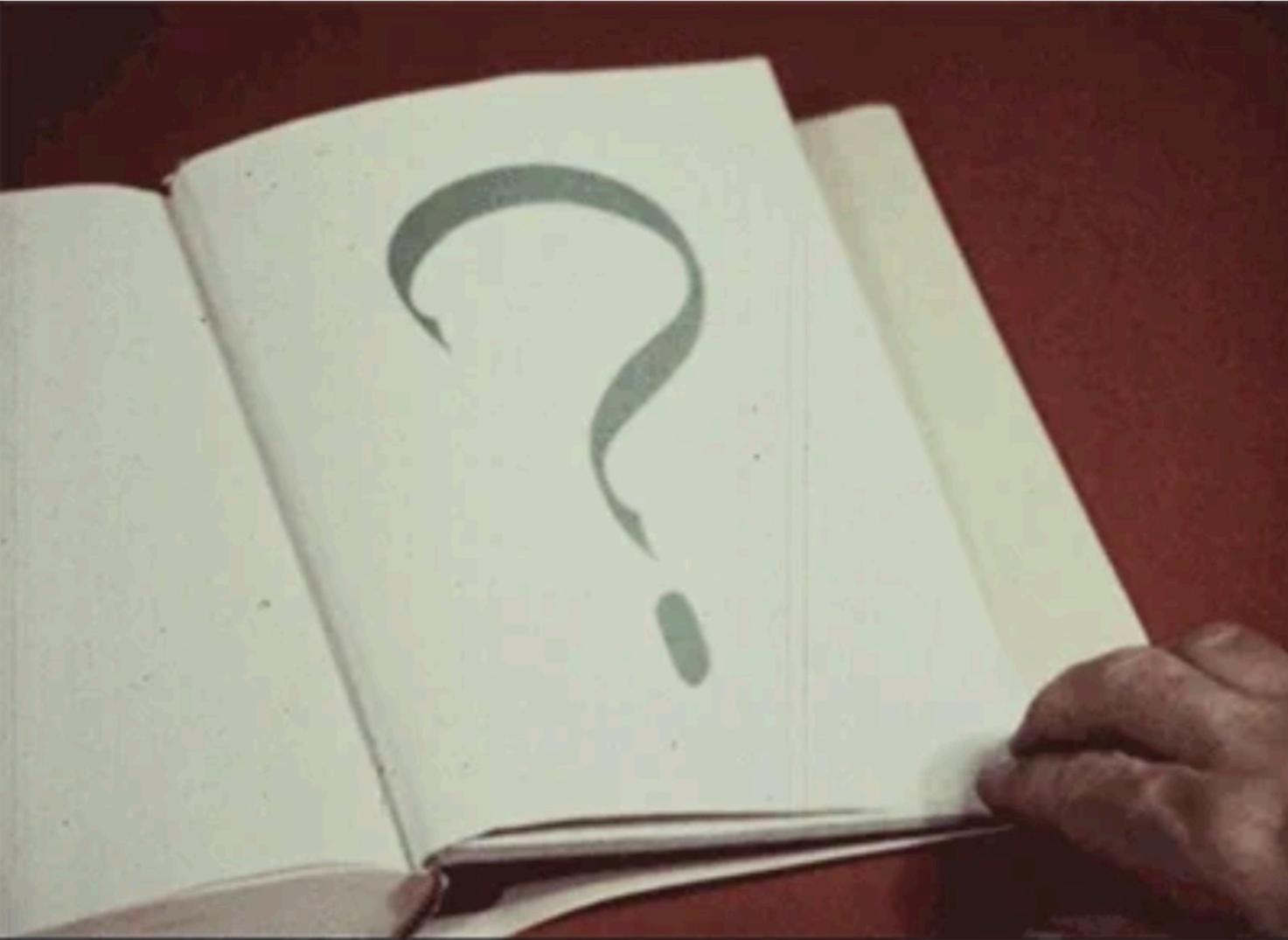
Des **applications** qui peuvent se **contacter**,
échanger des données
avec des règles clairement établie permettant de **déclencher des actions**



Dans ce cours

On va essayer de répondre aux questions suivantes

- comment communiquer entre deux applications sur un réseau ?
- Comment envoyer un message d'une application vers une autre via le réseau ?
- Sous quel format envoyer ce message ?
- Comment fait-on une application Python capable d'écouter sur le réseau ?



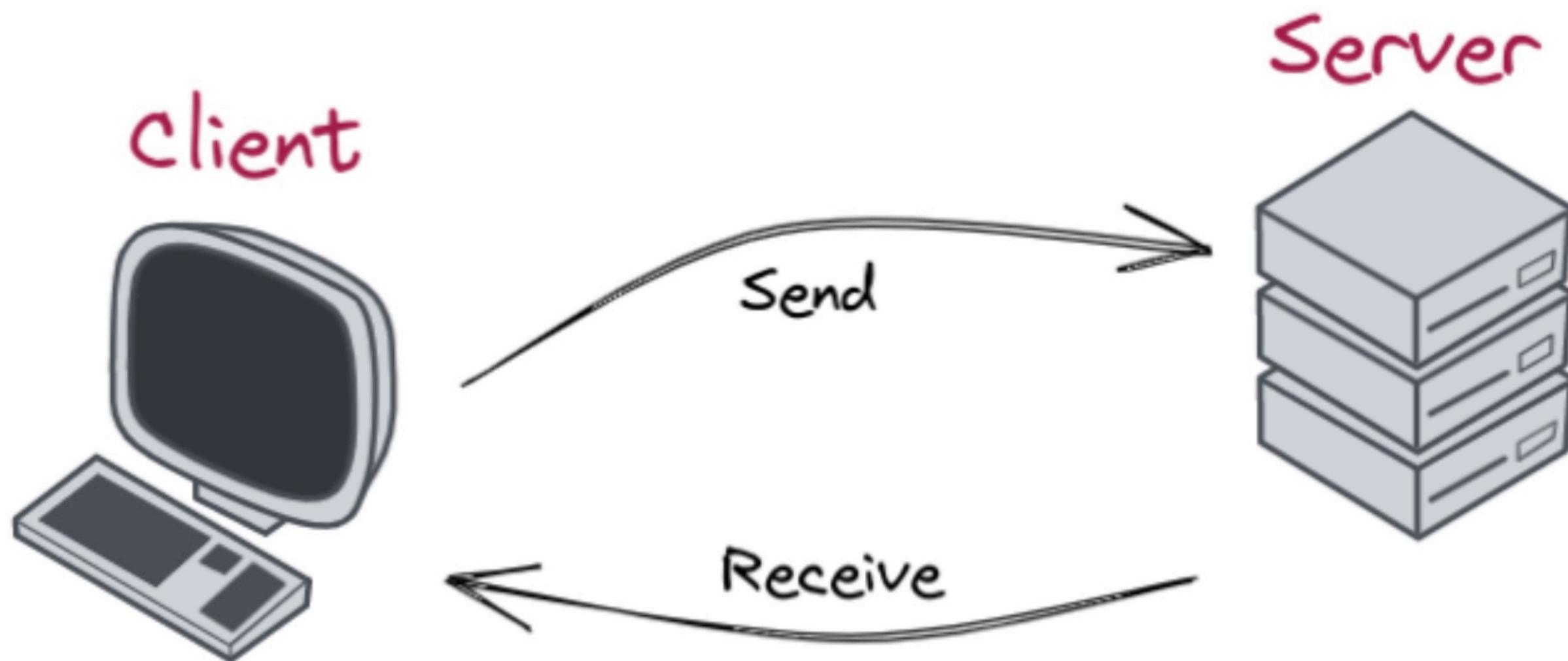
Architecture

Pour faire collaborer des applications ensemble il existe plein de modèles, d'architectures différentes

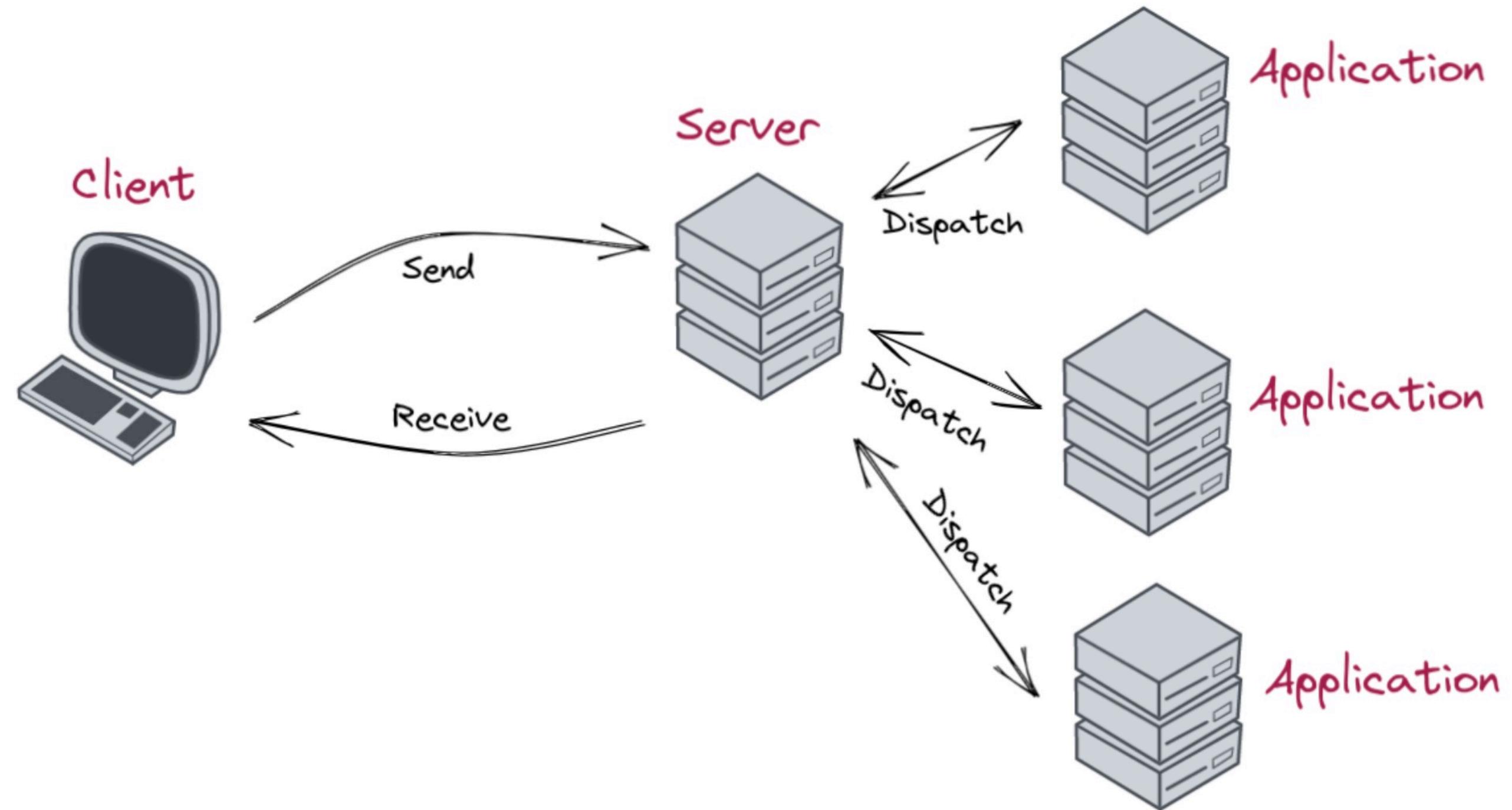


On va regarder les plus classiques

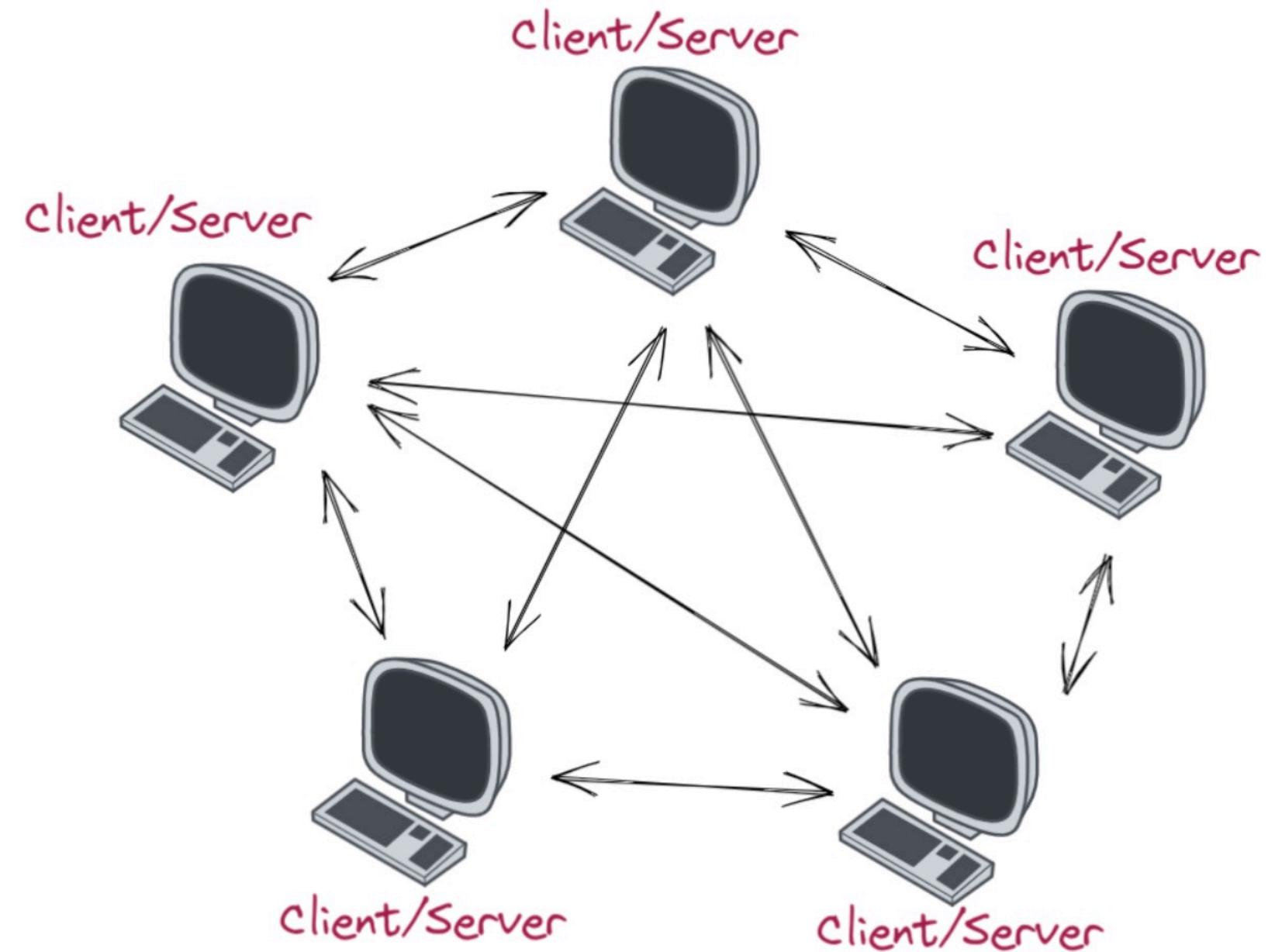
Client-serveur



Architecture trois-tiers

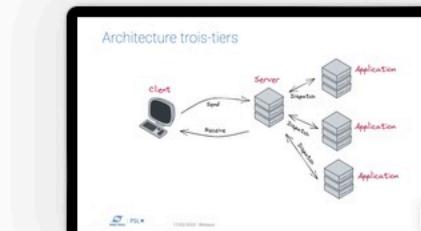


Architecture pair à pair



Très à la mode une époque où Netflix/Amazon Prime/... n'existaient pas (oui oui cette période est réelle 😬)

 Projet [folding@home](#)



Le Web



Juste un gros réseau

Le cloud



Le réseau : principe



Réseau Infrastructure

Tout d'abord un réseau c'est quoi ?

Et bien c'est une **infrastructure** que l'on utilise pour faire transiter des données.

Dans sa version la plus élémentaire qui soit un réseau est composé de deux appareils reliés entre eux par un câble réseau par exemple.

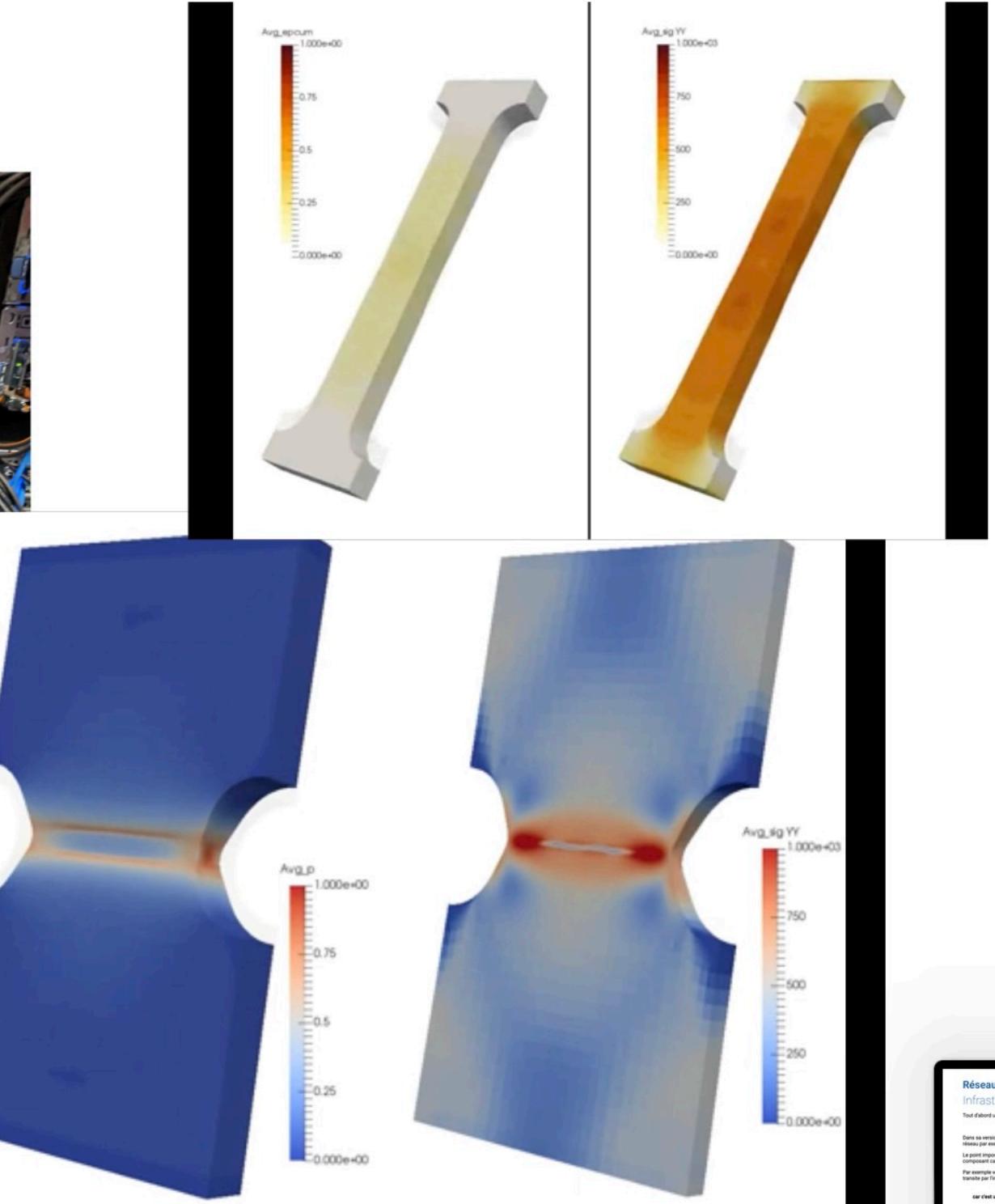
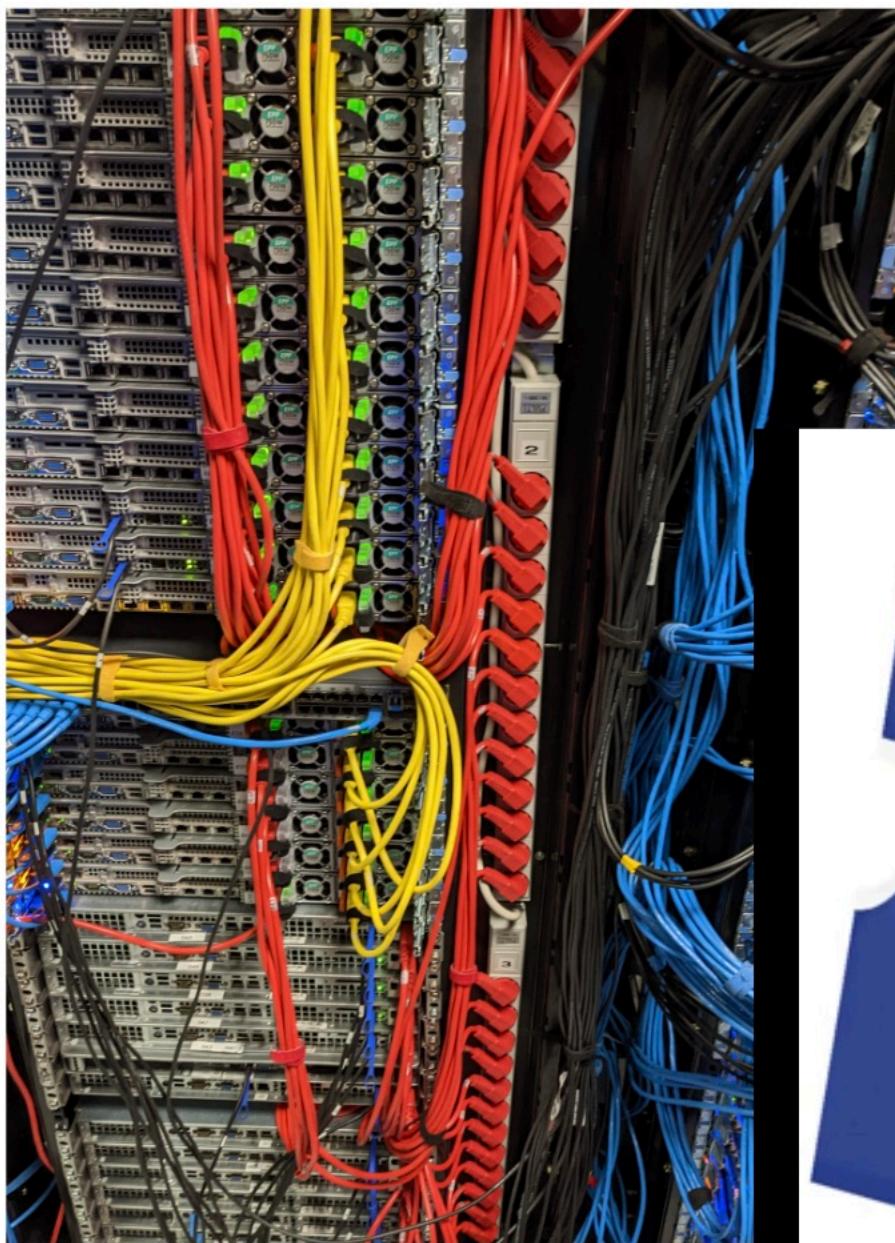
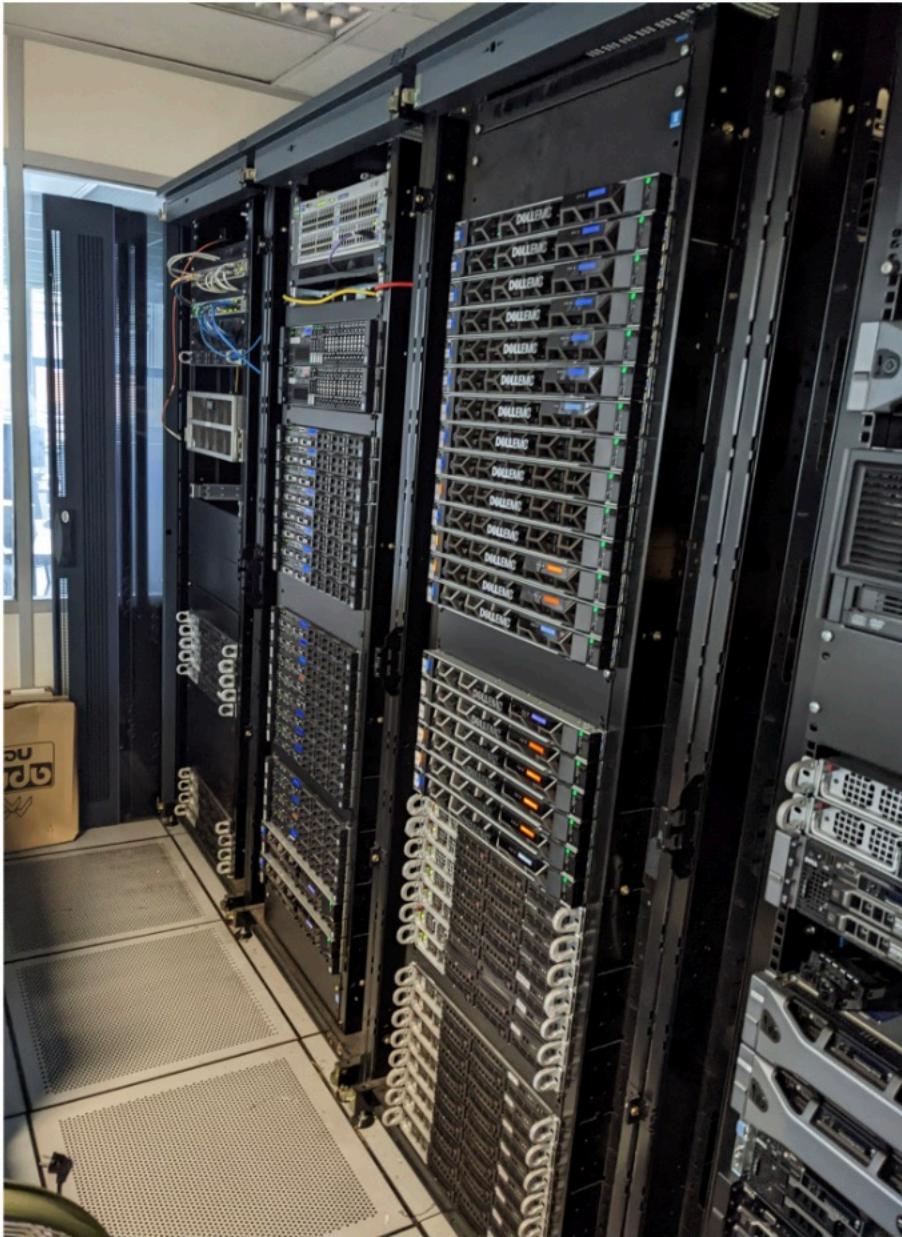
Le point important là-dedans c'est qu'un appareil connecté au réseau doit posséder une **interface réseau**, un composant capable de communiquer c'est-à-dire d'envoyer et recevoir un signal.

Par exemple votre ordinateur portable possède deux interfaces réseau : la prise RJ45 et la carte wifi. Le signal qui transite par l'interface réseau est un signal binaire.

**⚠ L'appareil en lui-même n'a pas besoin de connaître la signification de ce signal,
car c'est un programme tournant derrière l'interface réseau qui se chargera de traiter le signal en question ⚠**

Petite parenthèse

Supercalculateur : un modèle peer-to-peer



High Performance Computing

Diviser pour mieux régner

Décomposition en sous-domaines



17/02/2023 - Réseaux



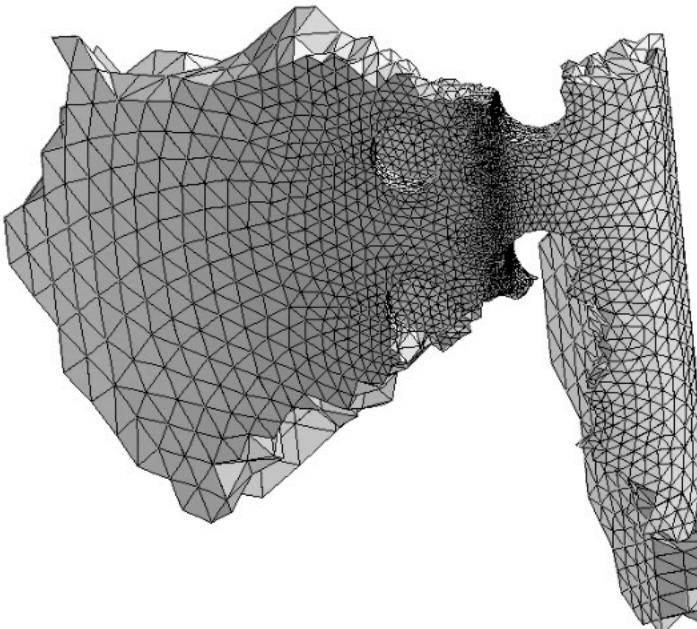
High Performance Computing

Diviser pour mieux régner

Décomposition en sous-domaines



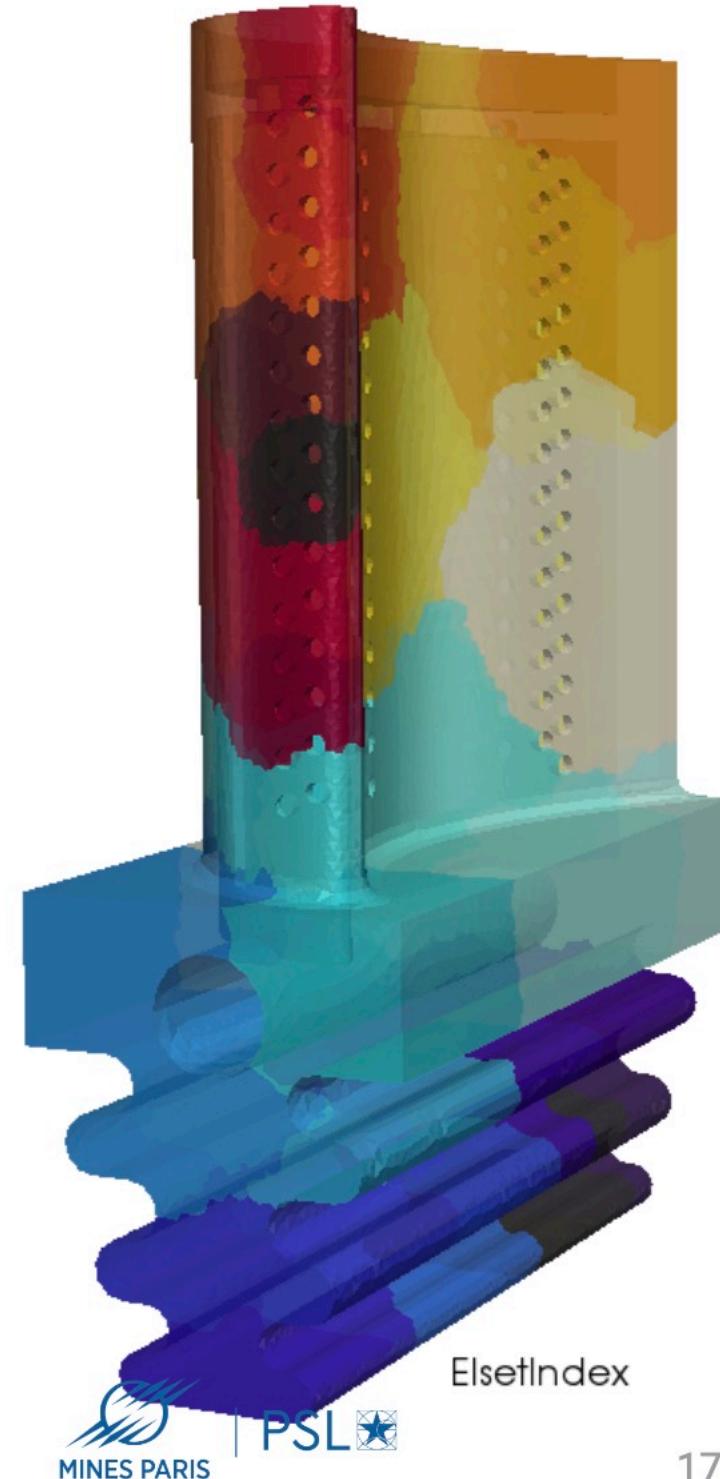
Chaque sous-domaine "envoyé"
sur une machine de calcul



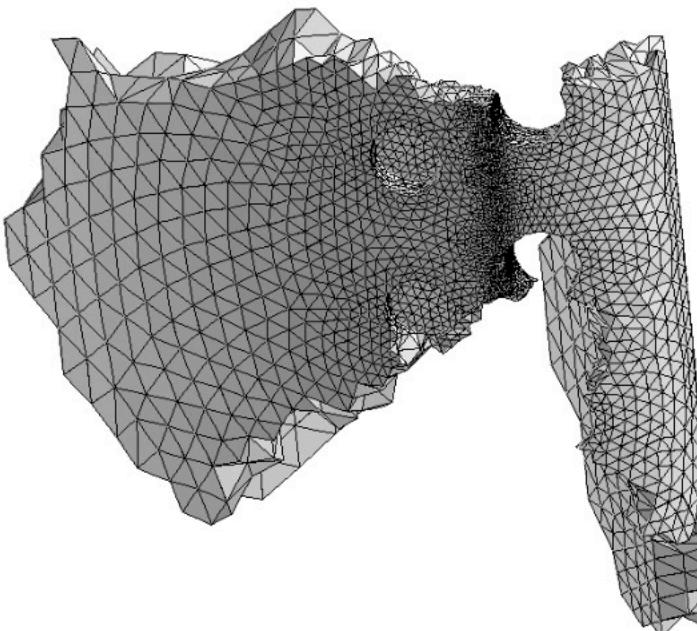
High Performance Computing

Diviser pour mieux régner

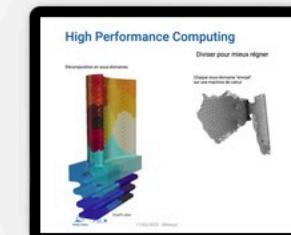
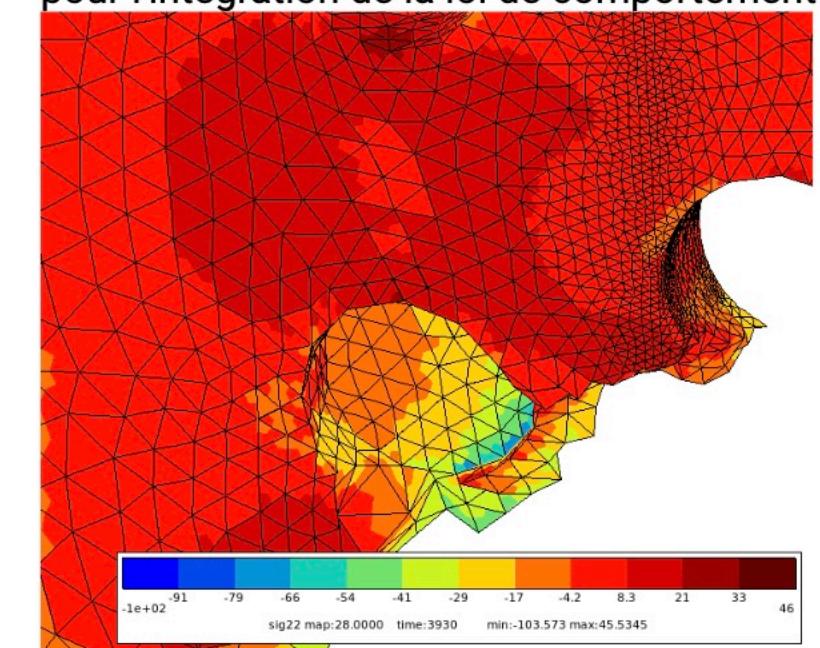
Décomposition en sous-domaines



Chaque sous-domaine "envoyé" sur une machine de calcul



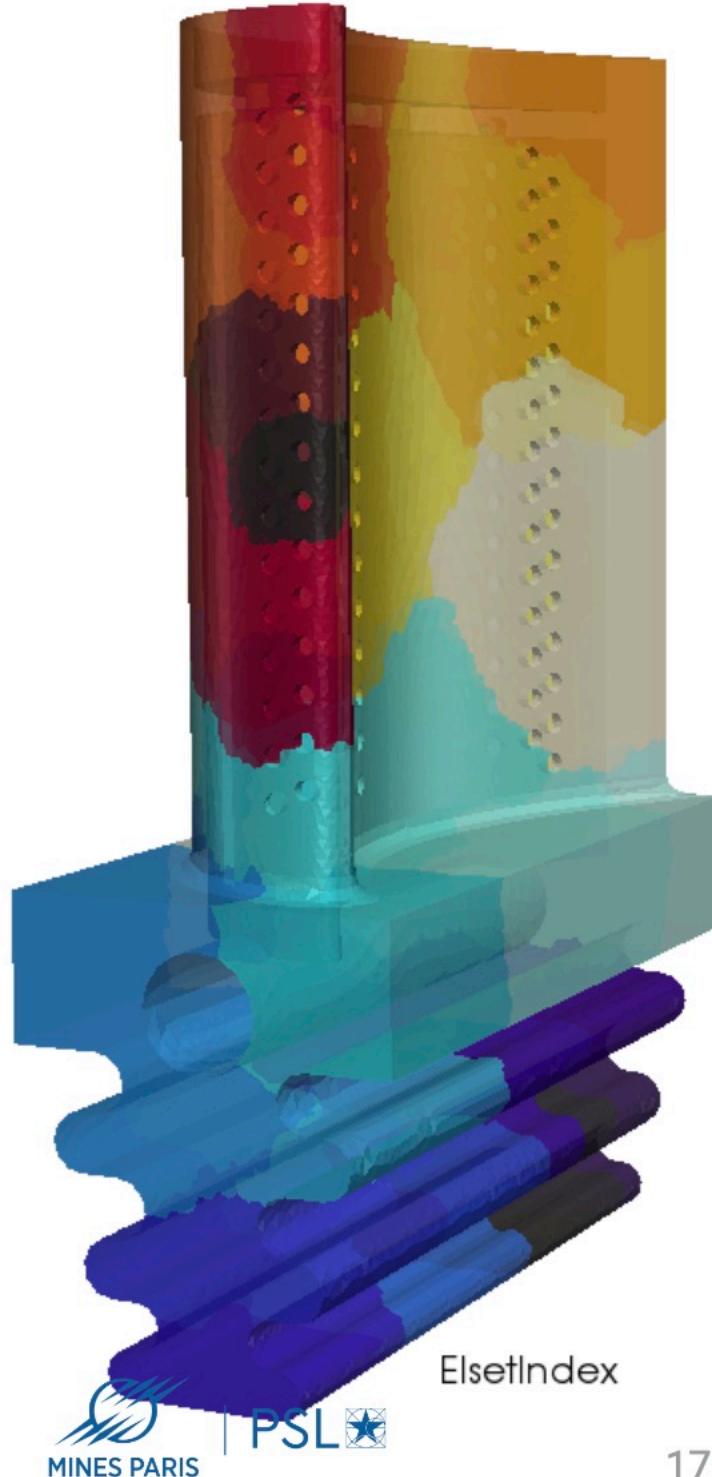
Chaque sous-domaine "éclaté" par paquet d'éléments pour l'intégration de la loi de comportement



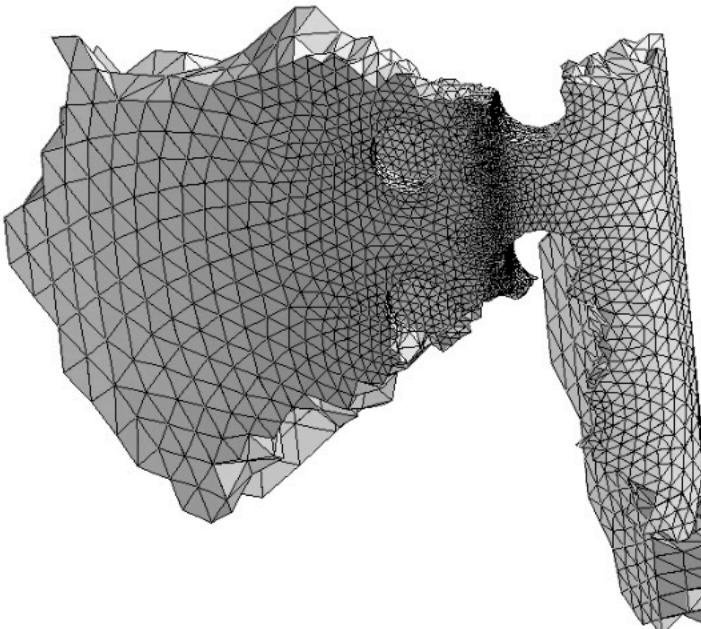
High Performance Computing

Diviser pour mieux régner

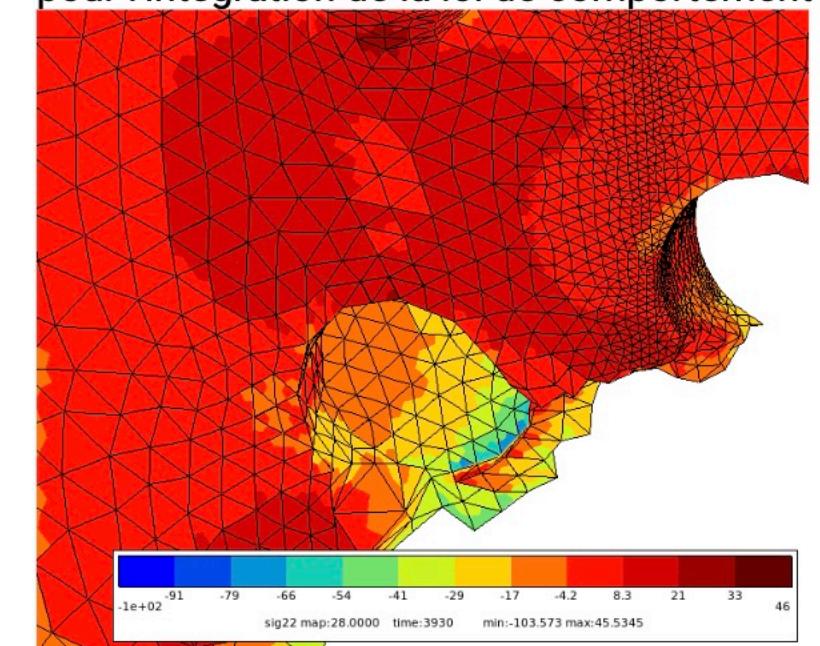
Décomposition en sous-domaines



Chaque sous-domaine "envoyé" sur une machine de calcul

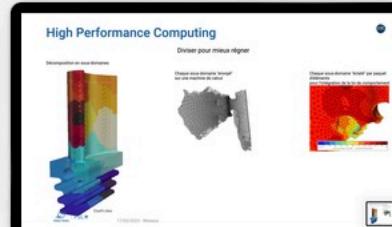


Chaque sous-domaine "éclaté" par paquet d'éléments pour l'intégration de la loi de comportement



Au niveau de chaque sous-domaines :

- Opérations algébriques distribuées
- Résolution de problèmes locaux (solveurs DD)

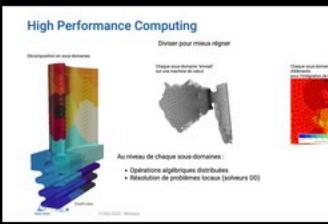


Réseau

Différentes qualités

La qualité du réseau, un petit truc qui a son importance suivant l'application 🚀📈

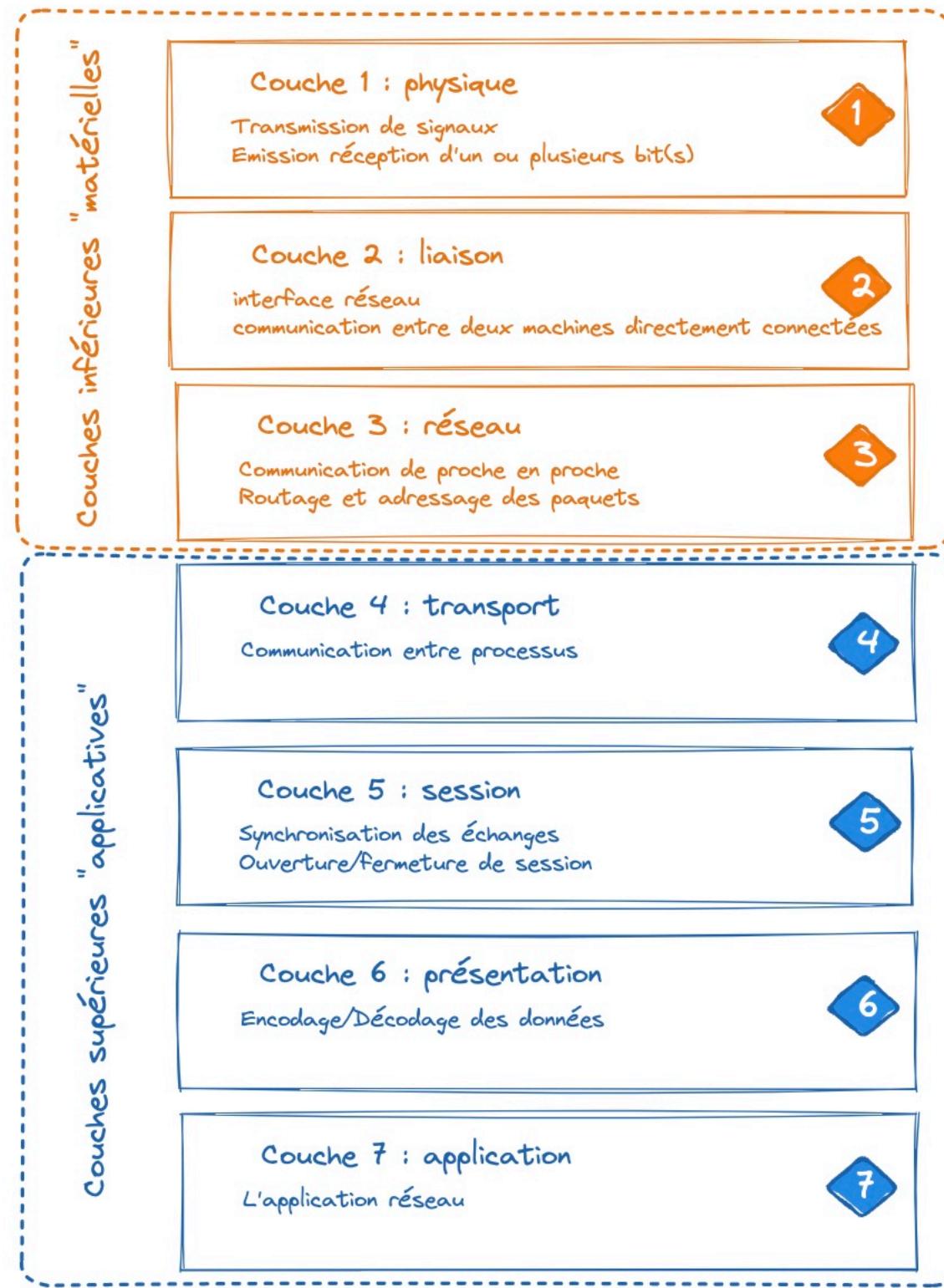
⌚ Sur des grosses simulation le temps des échanges peut représenter 20% du temps de calcul 💣



Un réseau et c'est tout ?



Modèle OSI



Open System Interconnexion

norme mise en place par le comité ISO en 1984

Objectifs :

standardiser les communications
entre appareils sur un réseau

Adressage

Associer à chaque interface de chaque machine sur un réseau une adresse unique

Cette adresse peut être *temporaire* ou bien *fixe*.

C'est ce qu'on appelle l'adresse IP, pour *Internet Protocol*. L'adresse IP d'une interface réseau s'écrit comme une combinaison de quatre nombres compris entre 0 et 255.

172.16.254.42

Masque réseau Machine sur le réseau



Adressage

Associer à chaque interface de chaque machine sur un réseau une adresse unique

Cette adresse peut être *temporaire* ou bien *fixe*.

C'est ce qu'on appelle l'adresse IP, pour *Internet Protocol*. L'adresse IP d'une interface réseau s'écrit comme une combinaison de quatre nombres compris entre 0 et 255.



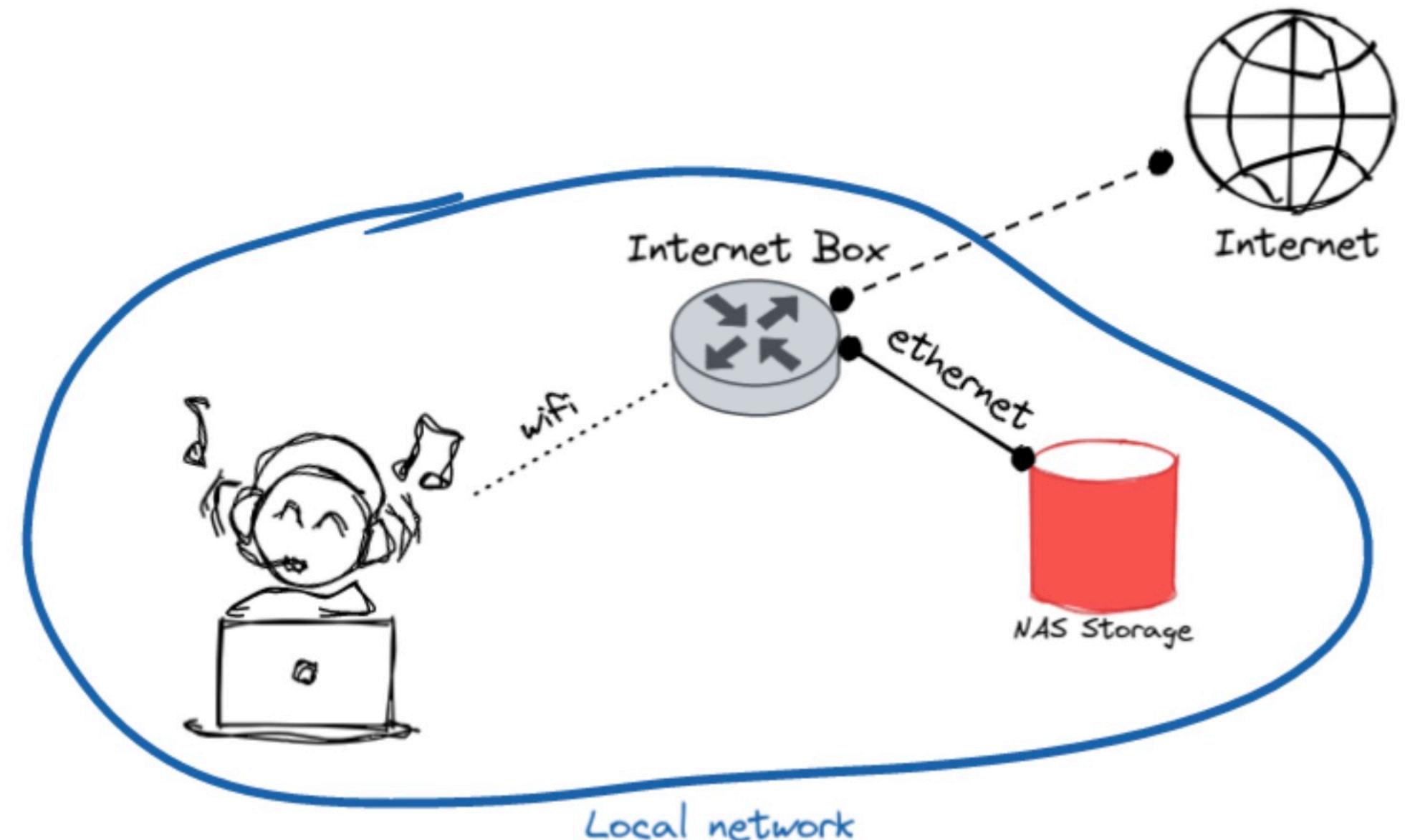
Remarque : en 2011 💣 l'épuisement des adresses IP disponibles...

Il a donc été mis en place le protocole IP v6 (l'ancien protocole était le v4)

Le principe est simple passer d'une adresse définie sur 32 bits à une adresse sur 128 bits (érites en hexadecimale) → .
exemple 2001:0db8:0000:85a3:0000:0000:ac1f:8001

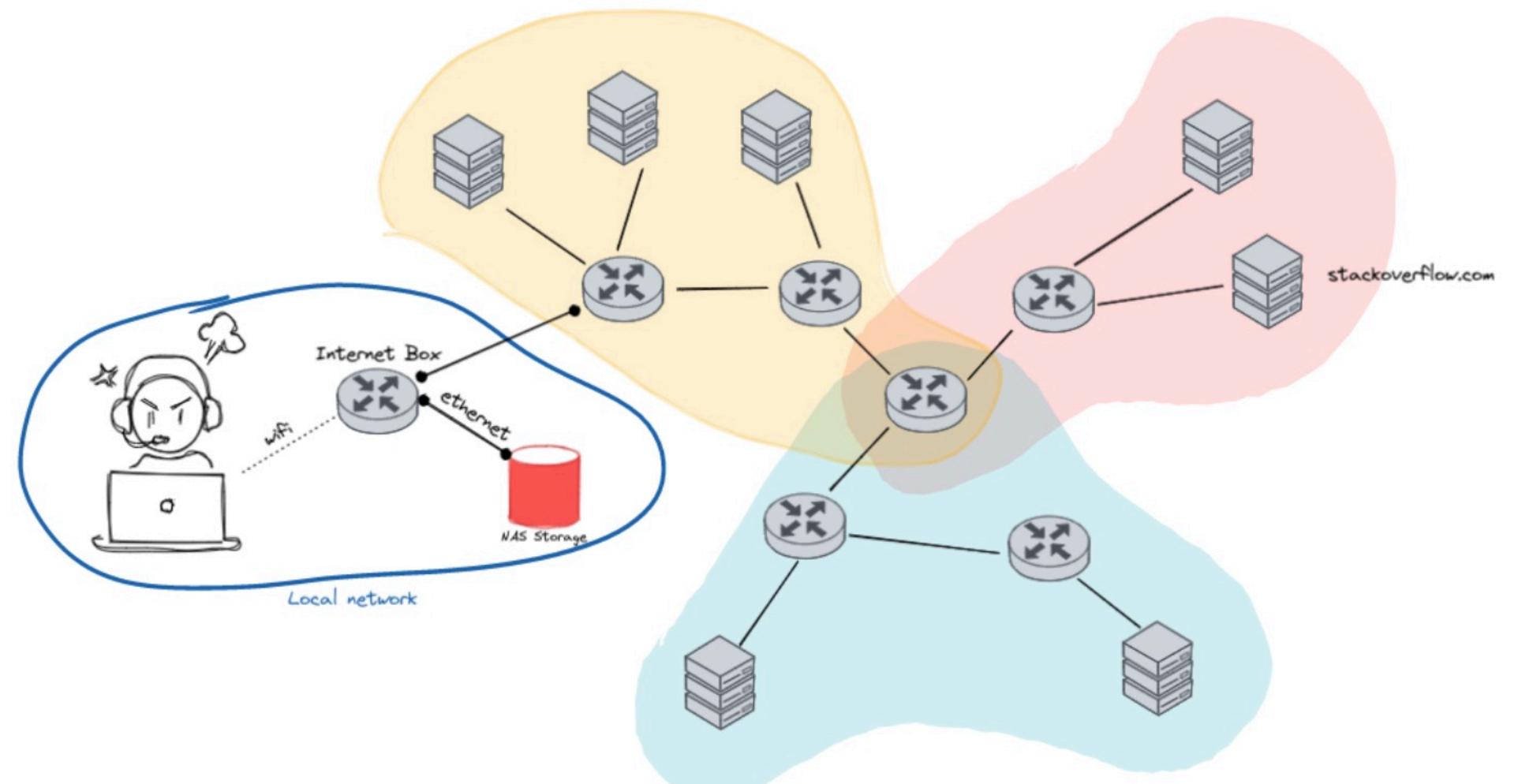
Interconnexion

Réseau local



Interconnexion

Réseau distant



Interconnexion

Pour résumer :

interconnexion qui constitue en fait la troisième couche du modèle OSI

gère trois éléments :

- Routage

chemin entre deux machines dans des réseaux différents,
chemin passant par les passerelles : ces fameuses machines ayant des interfaces dans deux réseaux distincts.

- Relayage

s'occupe, une fois la route déterminée,
de faire transiter l'information de la machine A à la machine B

- Contrôle de flux

une fonctionnalité optionnelle mais néanmoins essentielle
qui permet de décongestionner l'ensemble du réseau (au sens large).
Un peu le Waze du transit de données

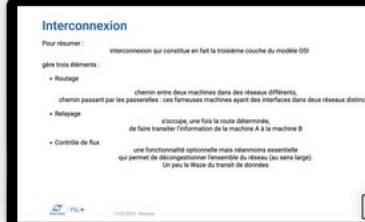


Les noms de domaines là-dedans !

Retenir les adresses IP c'est quand même pas super 😱 !

Par exemple imaginez que vous deviez retenir 77.158.173.120 pour savoir les salles de cours ~~on ne vous verrait pas souvent~~

* c'est l'adresse IP du serveur qui héberge OASIS



Les noms de domaines là-dedans !

Retenir les adresses IP c'est quand même pas super 😱 !

Par exemple imaginez que vous deviez retenir 77.158.173.120 pour savoir les salles de cours ~~on ne vous verrait pas souvent~~

Un truc magique le :

Domain Name System

En gros c'est le service qui fait l'association entre un nom de domaine et un adresse IP.

* c'est l'adresse IP du serveur qui héberge OASIS



Les noms de domaines là-dedans !

Retenir les adresses IP c'est quand même pas super 😱 !

Par exemple imaginez que vous deviez retenir 77.158.173.120 pour savoir les salles de cours ~~on ne vous verrait pas souvent~~

Un truc magique le :

Domain Name System

En gros c'est le service qui fait l'association entre un nom de domaine et un adresse IP.

```
$ nslookup www.minesparis.psl.eu
```

Non-authoritative answer:

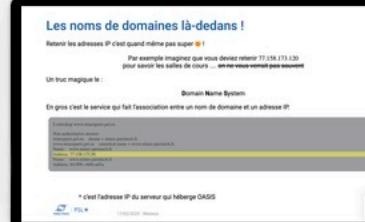
```
minesparis.psl.eu  dname = mines-paristech.fr.  
www.minesparis.psl.eu  canonical name = www.mines-paristech.fr.  
Name: www.mines-paristech.fr  
Address: 77.158.173.58  
Name: www.mines-paristech.fr  
Address: 64:ff9b::4d9e:ad3a
```

* c'est l'adresse IP du serveur qui héberge OASIS



On sait s'orienter comment on cause maintenant

➡ On a besoin de la 4ème couche du modèle OSI



La couche transport



La quatrième couche du modèle

spécification de comment on fait pour envoyer des données d'un serveur A vers un client B et inversement.

Différents protocole établis :

- TCP
- UDP
- ...

⚠ Attention ⚠

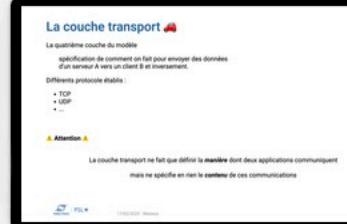
La couche transport ne fait que définir la ***manière*** dont deux applications communiquent

mais ne spécifie en rien le ***contenu*** de ces communications

Un serveur == une application ?

Connaitre l'IP du serveur ne vous permet pas encore de communiquer avec l'application qui se trouve sur ce serveur

? D'ailleurs sur un serveur il ne peut y avoir qu'une application réseau ou peut-on en mettre plusieurs ?



Un serveur == une application ?

Connaitre l'IP du serveur ne vous permet pas encore de communiquer avec l'application qui se trouve sur ce serveur

? D'ailleurs sur un serveur il ne peut y avoir qu'une application réseau ou peut-on en mettre plusieurs ?

On peut avoir plusieurs applications sur un même serveur et heureusement 😊

Le choix de l'application avec laquelle on va discuter implique la notion de **port**

port = porte d'entrée du service 🚪

Sur une machine on a $2^{16} = 65\,536$ \$

mais on ne fait pas tourner autant d'application



Un serveur == une application ?

Connaitre l'IP du serveur ne vous permet pas encore de communiquer avec l'application qui se trouve sur ce serveur

? D'ailleurs sur un serveur il ne peut y avoir qu'une application réseau ou peut-on en mettre plusieurs ?

On peut avoir plusieurs applications sur un même serveur et heureusement 😊

Le choix de l'application avec laquelle on va discuter implique la notion de **port**

port = porte d'entrée du service 🚪

Sur une machine on a $2^{16} = 65\,536$

mais on ne fait pas tourner autant d'application

Quelques port normalisés :

22 : SSH, 25 : SMTP, 80 : HTTP, 443 : HTTPS



Bas niveau

TCP/IP

Transmission Control Protocol

est **le** protocole historique (Bob Kahn et Vinton Cerf, Septembre 1973), qui doit sa longévité par sa robustesse et sa fiabilité.

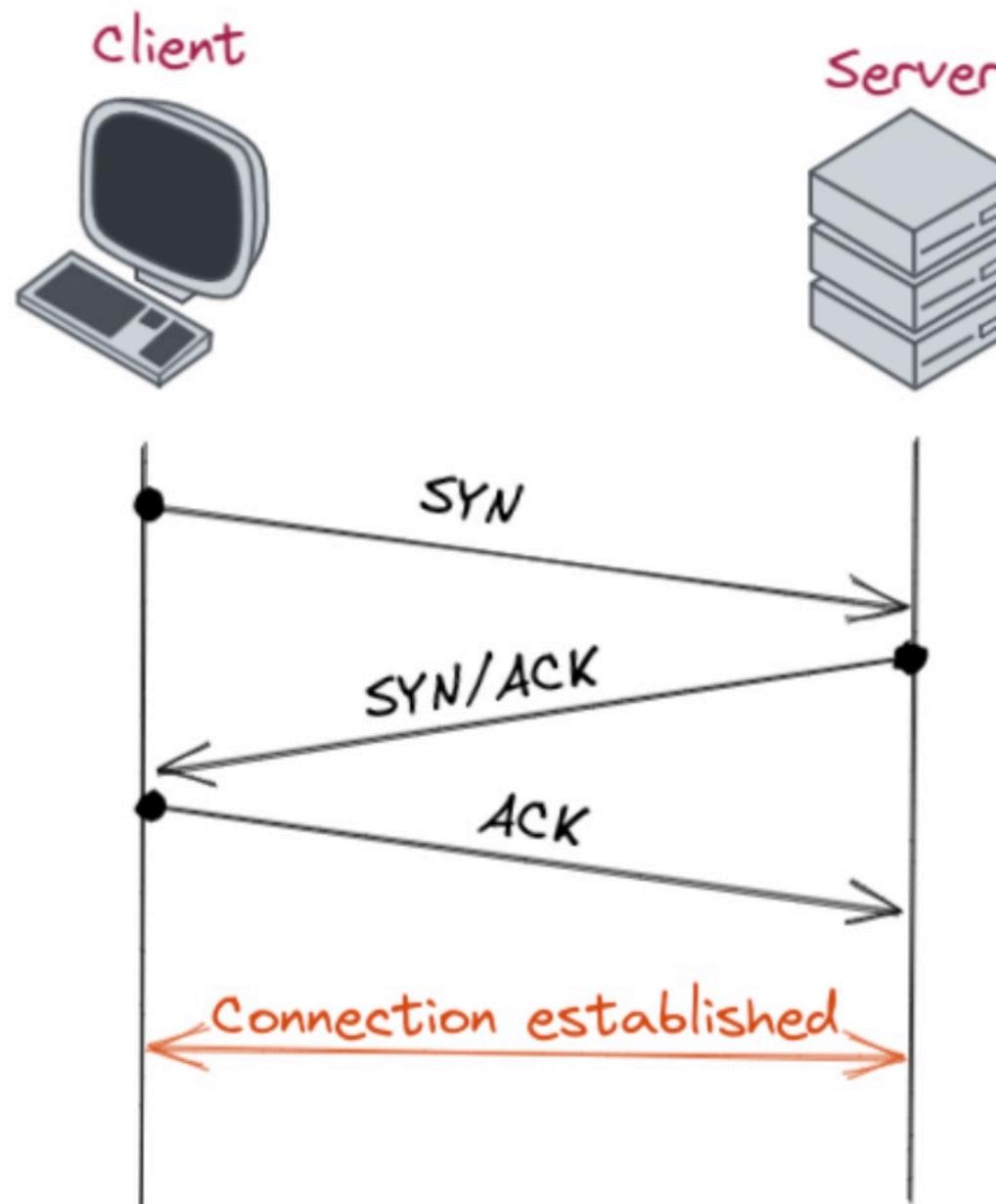
Aujourd'hui lorsque vous naviguez sur le web
la plupart des échanges qui ont lieu entre votre navigateur et les sites web sont basés sur du TCP

Le principe du TCP est très simple et se décompose en trois étapes:

- établissement de la connexion
- transfert de données
- fin de la connexion

Bas niveau

TCP/IP : open



La connexion d'un client à un serveur TCP se décompose en trois étapes

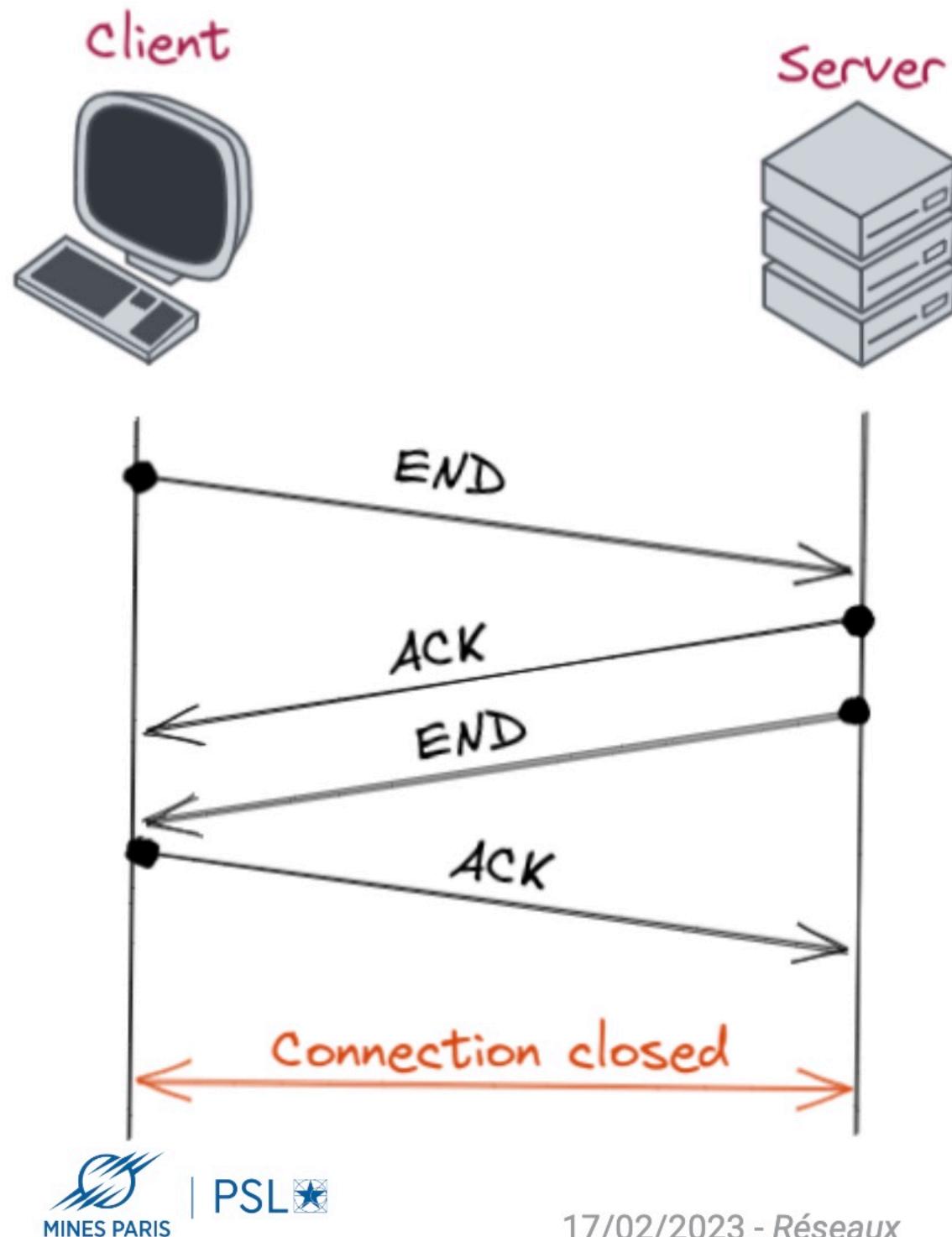
three way handshake

de la manière suivante :

- 1 Client : Hello le serveur tu m'entends ?
- 2 Serveur : Oui je t'entends et toi ?
- 3 Client : Oui c'est bon je t'entends

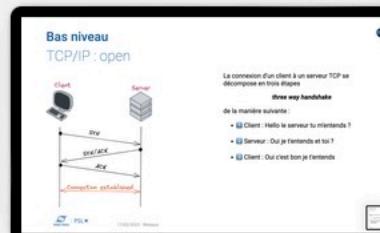
Bas niveau

TCP/IP : close



Clotûre en 4 étapes

- ① Client : j'ai fini
- ② Serveur : Ok c'est noté
- ③ Serveur : moi aussi je n'ai plus rien à te dire
- ④ Client : Ok à la prochaine



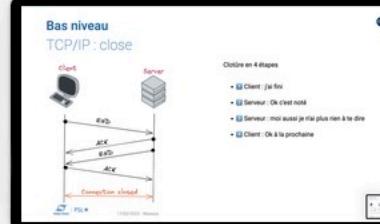
Regardons un peu en vrai comment ca marche

<https://replit.com/@BasileMarchand/TcpExample?v=1>

ou

<http://bit.ly/3HQ49i>

ou



Bas niveau

TCP un truc de riche 

Vous pouvez donc voir qu'avec cette approche

 la connexion est extrêmement fiable et il y a peu de chances d'avoir des loups

En revanche cette fiabilité n'est pas gratuite 

 elle s'accompagne d'un coût en terme d'échanges relativement élevé

C'est pour cela qu'il existe une alternative au TCP 

Bas niveau

UDP

Le protocole UDP (User Datagram Protocol) est complémentaire au protocole TCP. Crée par David Reed en 1980.

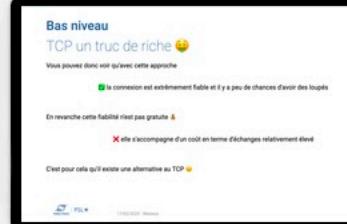
Cas d'usage :

Transmission rapide de données et réception de l'intégralité **pas impérative**

TCP = très fiable mais lent

vs

UDP = rapide mais peu fiable



Bas niveau

UDP

Le protocole UDP (User Datagram Protocol) est complémentaire au protocole TCP. Crée par David Reed en 1980.

Cas d'usage :

Transmission rapide de données et réception de l'intégralité **pas impérative**

TCP = très fiable mais lent

vs

UDP = rapide mais peu fiable

Les applications :



La couche 4 suffisante ou besoin de plus ?

Avec tcp ou udp on peut faire nos transfert de données entre application

A votre avis c'est tout bon du coup ou on a besoin d'un truc en plus ?



La couche 4 suffisante ou besoin de plus ?

Avec tcp ou udp on peut faire nos transferts de données entre application

A votre avis c'est tout bon du coup ou on a besoin d'un truc en plus ?

🔍 Regardons sur un exemple concret 🔎

<https://replit.com/@BasileMarchand/tcpexample?v=1>

ou

<http://bit.ly/3YpoKDR>

ou



Un verrou



Rien de standard dans mes échange de données 😊

J'ai créé ma propre logique

mais elle ne l'est ~~peut-être~~ certainement pas aux yeux des autres.

Un verrou



Rien de standard dans mes échange de données 😬

J'ai créé ma propre logique

mais elle ne l'est ~~peut-être~~ certainement pas aux yeux des autres.

Un peu de standardisation ne ferait pas de mal ...



Au passage : transfert de données ...

La grande question qui peut se poser est

sous quel format est-il pertinent d'échanger des données ?

Le modèle OSI ne spécifie pas vraiment de format de données autre que dire c'est du binaire 😐

La couche 6 spécifie un peu les choses en réalité mais ça donne un spectre assez large en fait

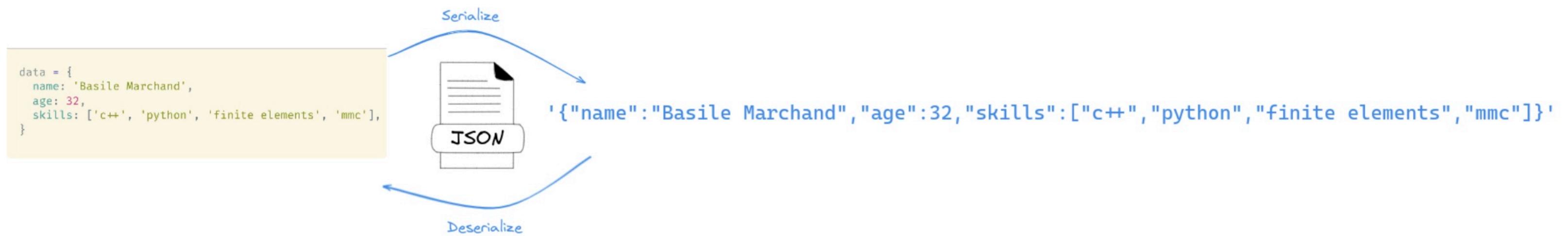
😩 Comment on fait si on veut faire transiter

un paquet de donnée structurée mais hétérogène ?

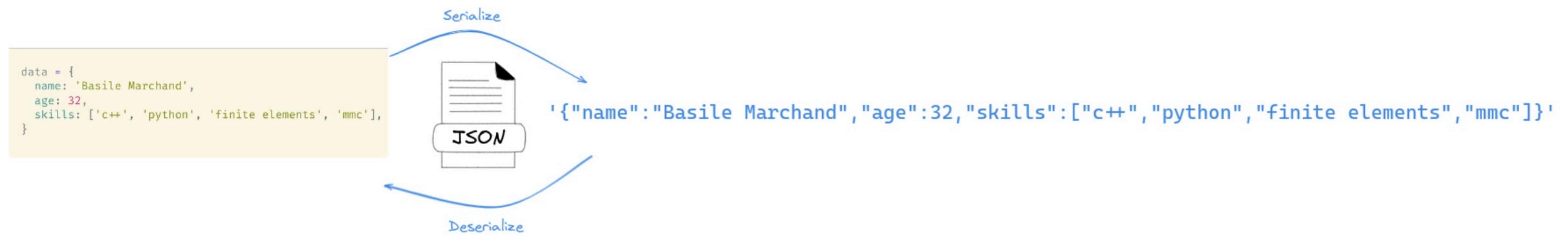
Par exemple les informations d'une personne :

Nom, Prénom, Date de naissance, nombre d'enfants, ...

Sérialisation JSON



Sérialisation JSON



Via Python 🐍 c'est facile !

```
import json
data = {}
serialized = json.dump(data)
```

```
import json
serialized = '{name: "basile marchand", ...}'
data = json.load(serialized)
```



Haut niveau : la couche 7 du modèle OSI

C'est là que les choses concrète commencent 😊

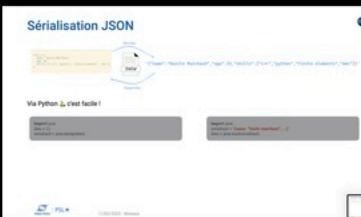
Couche 7 = couche Application

Chaque "catégorie" d'application spécifie alors :

- Comment se font les communication entre le client et l'application
- format des message, contenu attendu, ...

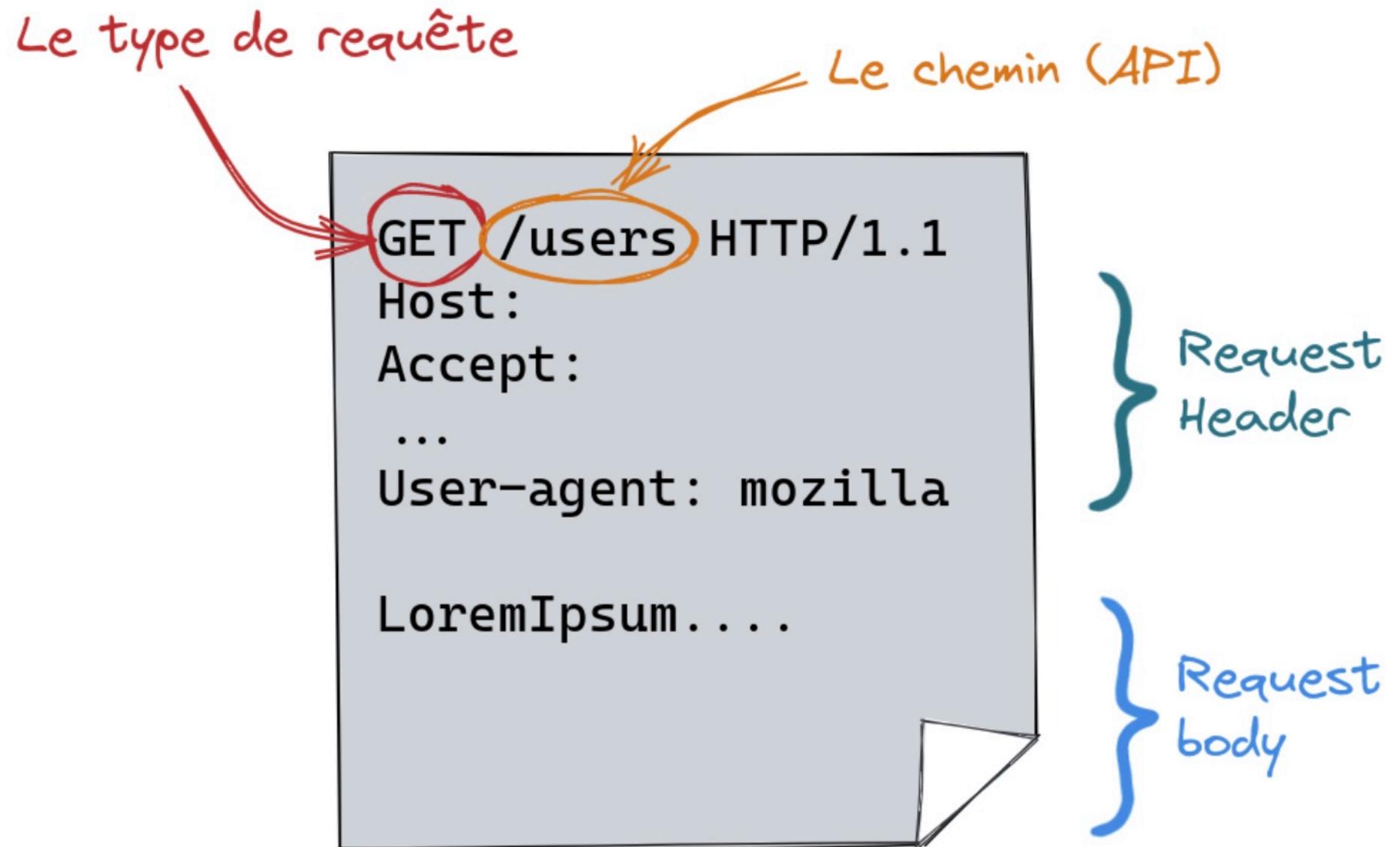
On parle de protocole :

- Transfert de fichiers 📁 : (S)FTP
- Messagerie 📩 : SMTP, POP, IMAP
- Sessions distantes : telnet, SSH



Protocol HTTP

Format d'une requête



Types de requêtes

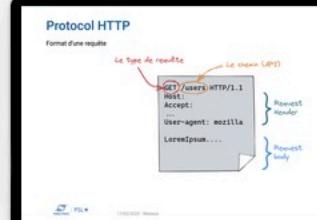
Vous avez peut être remarqué le GET dans la requête précédente.

En gros c'est pour dire que l'on veut faire un requête de type GET. Sous-entendu il existe d'autre type de requête ... dans le monde HTTP(S) il existe

- GET : requêtes pour **obtenir** du serveur une ressource (fichier html/css/js, image, video, données, ...)
- POST : requêtes pour **envoyer** des données au serveur en vu d'un traitement (ajout d'un utilisateur dans une base de donnée, ...)
- PATCH : requêtes pour **modifier partiellement** une ressource du serveur (mettre à jour l'adresse mail d'un utilisateur dans la base de donnée)
- DELETE : requêtes pour **supprimer** une ressource du serveur (supprimer un commentaire sur un article, ...)

Il s'agit là des principaux types de requêtes mais il en existe d'autre, pour la liste complète vous pouvez faire un tour https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

⚠ Il arrive souvent que POST soit utilisée, à la place de PATCH, pour mettre à jour une donnée déjà présente côté serveur ... 😞



Expérimetons

Dans Python 🐍 vous vous en doutez il existe tout ce qu'il faut !!

```
import requests
```

Nous allons utiliser le site <http://httpbin.org> qui met à disposition un serveur de test relativement utile.

<http://bit.ly/3XmaLNE>
or



Les codes de retour

Lorsque l'on fait une requête à un serveur via http/https ce dernier nous renvoie en premier lieu un code de retour.

Ces codes sont normalisés

Voici un extrait non complet des codes possibles :

- 200 : ok tout s'est bien passé ✓
- 301/302 : redirection de la page ↗
- 401 : il faut s'authentifier 🔒
- 403 : minute papillon tu n'as pas le droit d'accéder à ça ! 🚫
- 404 : ce que tu me demande n'existe pas !?
- 5XX : la c'est un problème de serveur 💣

Et donc la première chose à faire lorsque vous faites une requête à un serveur c'est de vérifier que le code de retour est bien 200 car sinon pas la peine de continuer !

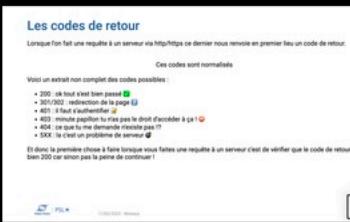


La notion d'API

Application Programming Interface

Permet de définir comment un programme **consommateur** va pouvoir exploiter les **fonctionnalités** données d'un programme **fournisseur**

Dans le domaine particulier du Web l'API se définit en fait à partir d'une URL. En effet l'accès à la ressource se fait en effectuant une requête GET sur un url particulière.



La notion d'API

Application Programming Interface

Permet de définir comment un programme **consommateur** va pouvoir exploiter les **fonctionnalités** données d'un programme **fournisseur**

Dans le domaine particulier du Web l'API se définit en fait à partir d'une URL. En effet l'accès à la ressource se fait en effectuant une requête GET sur un url particulière.

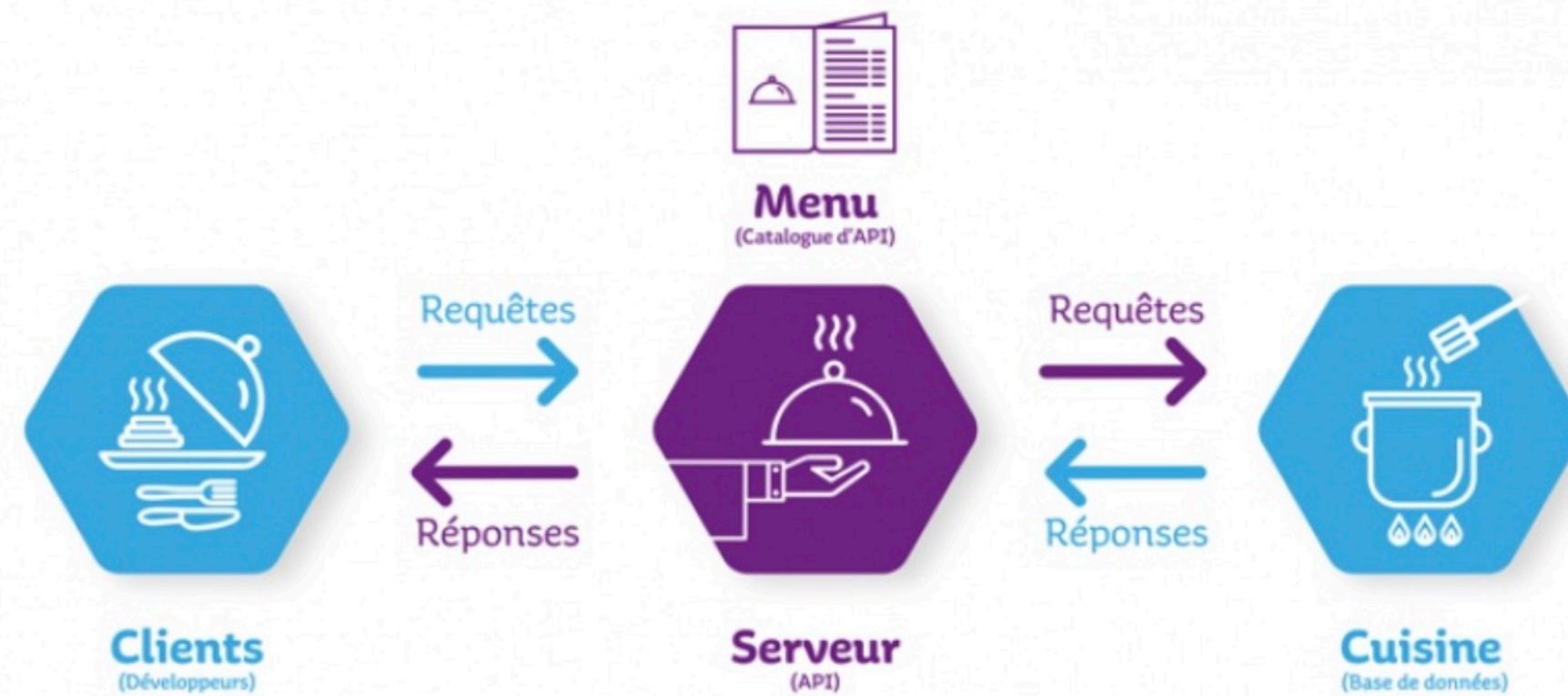


Image from Jérémy Mézière, Architecte Middleware chez Manutan

Faisons notre propre API

Considérons par exemple le cas d'un serveur générant des listes de nombres aléatoires à la demande. L'api d'un tel serveur pourrait être

- /api/integer renvoie un nombre aléatoire entier
- /api/float renvoie un nombre aléatoire flottant
- /api/integer?n=100 renvoie 100 nombres aléatoires entiers
- ...

<http://bit.ly/3HONIFN>

ou



Une API utilisable est une API documentée

Donc pour conclure sur les API il s'agit d'un moyen très simple pour offrir une interface vers des ressources et données distantes. La seule difficulté dans ce domaine c'est la définition et surtout la documentation des API. Donc si vous mettez en place un service Web disposant d'une API et que vous souhaitez ouvrir votre service vers l'extérieur merci de prendre le temps de documenter votre API.

On trouve en ligne plein d'API ouverte un lien pour avoir une liste non exhaustive

<https://github.com/public-apis/public-apis>

ou

<http://bit.ly/3YHC1qX>

ou

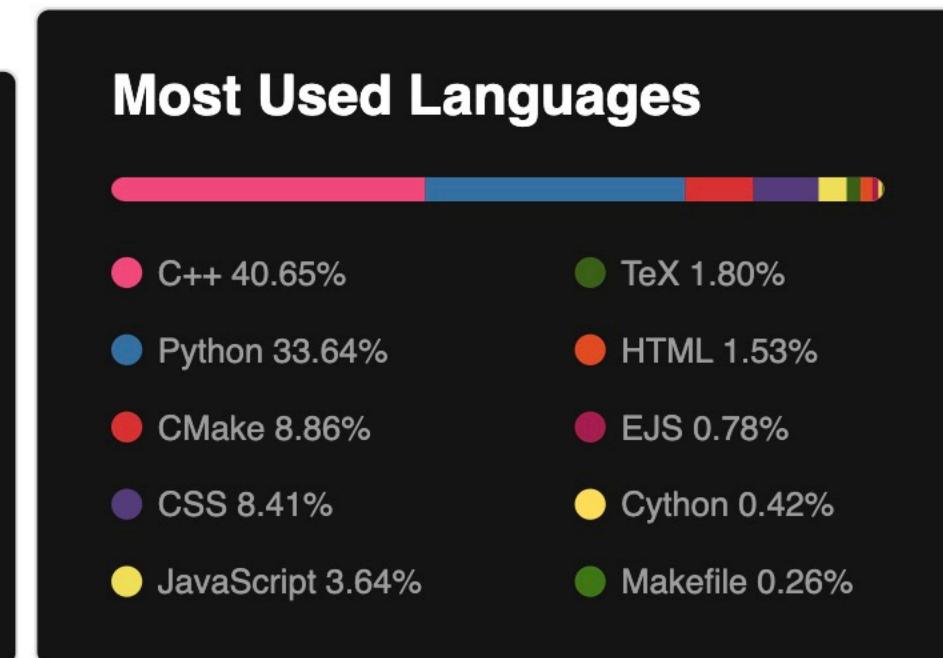
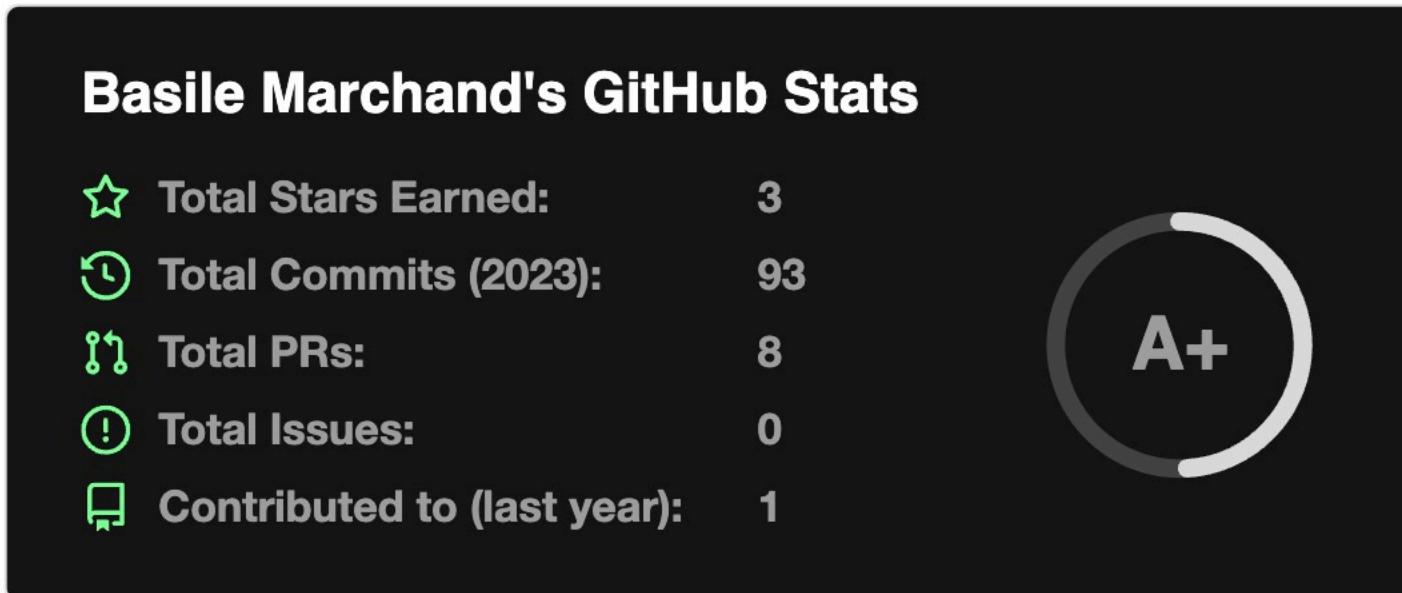


Par exemple

Générer quelques statistique sur Github

![Basile's GitHub stats](https://github-readme-stats.vercel.app/api?username=basileMarchand&count_private=true&show_icons=true&theme=dark)

![Basile's top languages](https://github-readme-stats.vercel.app/api/top-langs/?username=basileMarchand&hide=jupyter%20notebook&langs_count=10&theme=dark&layout=compact)

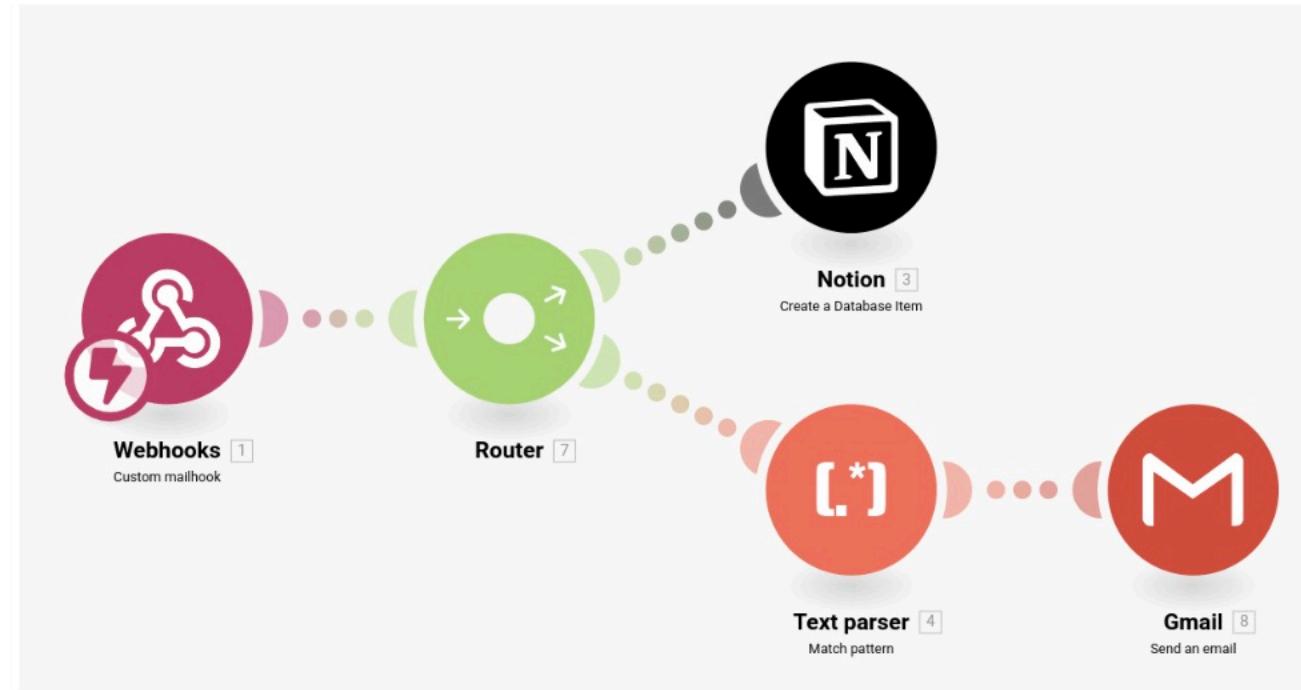


Un mot sur le "No Code"

Depuis quelques années de plus en plus à la mode

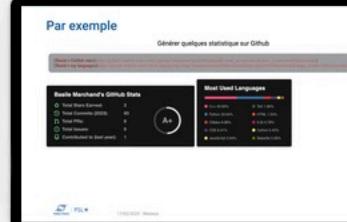
No Code, Low Code

Par exemple :



C'est toute la complexité qui se cache derrière mon Use-Case :

demande de support par mail qui provoque une nouvelle entrée dans une base de donnée
et une notification par mail si "urgent" dans le sujet du mail 😱



Application

Je vous ai mis en place un serveur minimaliste offrant une API permettant :

1. Lister l'ensemble des utilisateurs de la base de donnée
2. Mettre à jour votre status
3. Envoyer un message à un utilisateur
4. Récupérer les messages qui m'ont été envoyés.

🚀 <https://mines.bmarchand.fr/api/doc> 🚀



Application

Je vous ai mis en place un serveur minimaliste offrant une API permettant :

1. Lister l'ensemble des utilisateurs de la base de donnée
2. Mettre à jour votre status
3. Envoyer un message à un utilisateur
4. Récupérer les messages qui m'ont été envoyés.

🚀 <https://mines.bmarchand.fr/api/doc> 🚀

L'idée est que vous réalisez les actions suivantes :

1. A l'aide d'un programme Python 🐍 :
 1. faire une requête GET permettant de trouver quel est votre ID d'utilisateur
 2. faire une requête PATCH pour mettre à jour votre status
 3. faire des requêtes GET/POST pour vous envoyer des messages entre vous
2. Pour les plus joueurs, à l'aide du combo HTML/CSS/JS
 1. Faire le client web de ce serveur 😊 !



La semaine prochaine !

On passe du côté obscur
et on voit comment définir nos API

