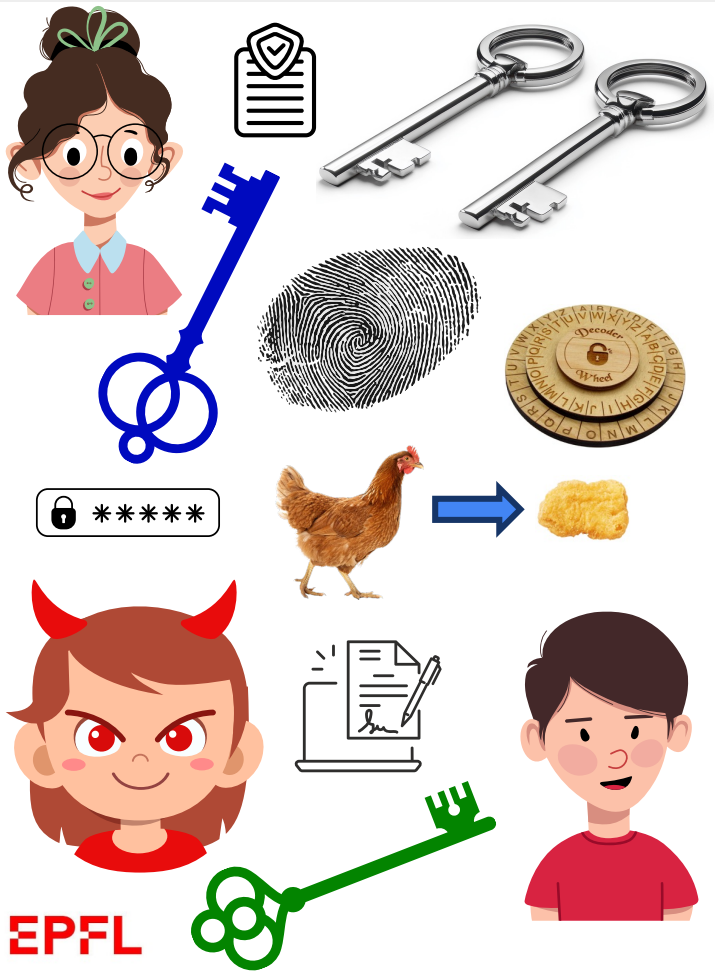
Information, Calcul et Communication

CS-119(k) ICC – Théorie
Semaine 13

Rafael Pires
rafael.pires@epfl.ch

Lausanne, 22.05.2026

Précédemment, dans ICC-T 13



- Les besoins en sécurité : confidentialité, authenticité, intégrité, non-répudiation
- Les outils cryptographiques :
  - Chiffrement symétrique (AES) : rapide, clé partagée
  - Diffie-Hellman : permet un échange de clé sans la transmettre
  - Fonctions de hachage : empreinte numérique, vérification d'intégrité
  - Chiffrement asymétrique (RSA) : deux clés, mais plus lent
  - Signature numérique : prouve l'identité et l'intégrité d'un message
- Certains cryptosystèmes (ex. : RSA, Diffie–Hellman) reposent sur des problèmes mathématiques difficiles (ex. : factorisation, logarithme discret) ; d'autres (AES, OTP) s'appuient sur des constructions algorithmiques ou de l'aléa parfait.

ICC

Information

- Représentation de l'information
  - Nombres entiers et réels
  - Circuits logiques et transistors
- Echantillonnage et reconstruction de signaux
- Entropie et compression de données

Calcul

- Algorithmes
- Complexité
- Récursivité
- Algorithmes gloutons / prog. dynamique
- Théorie de la calculabilité

Communication

- **Cryptographie**
- **Réseaux**

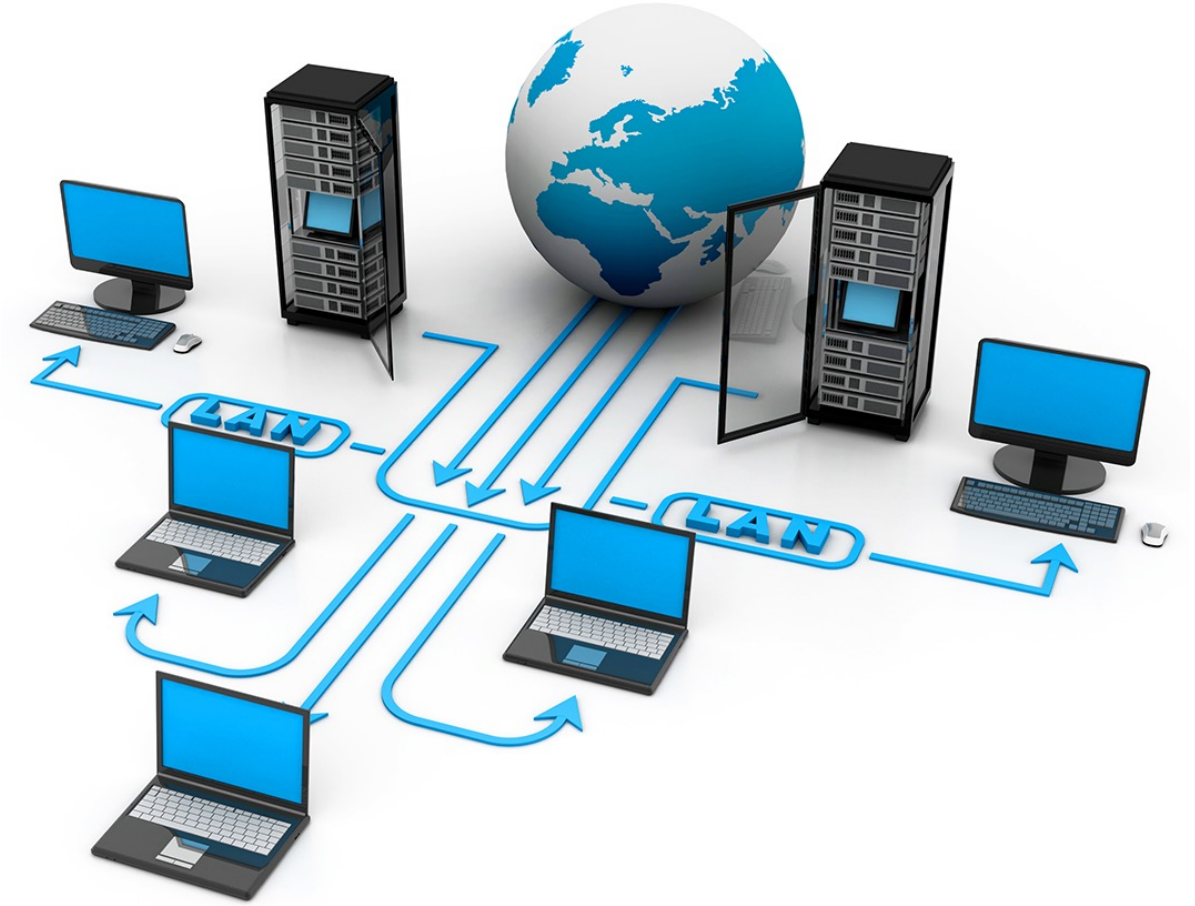
Aujourd'hui

- **Introduction aux réseaux informatiques : paquets et couches de protocoles**
  - Comment les données circulent (paquets & couches)
- **Routage (Protocole IP)**
  - Comment les données trouvent leur chemin (routage/IP)
- **Congestion (Protocole TCP)**
  - Comment on évite l'engorgement (congestion/TCP)

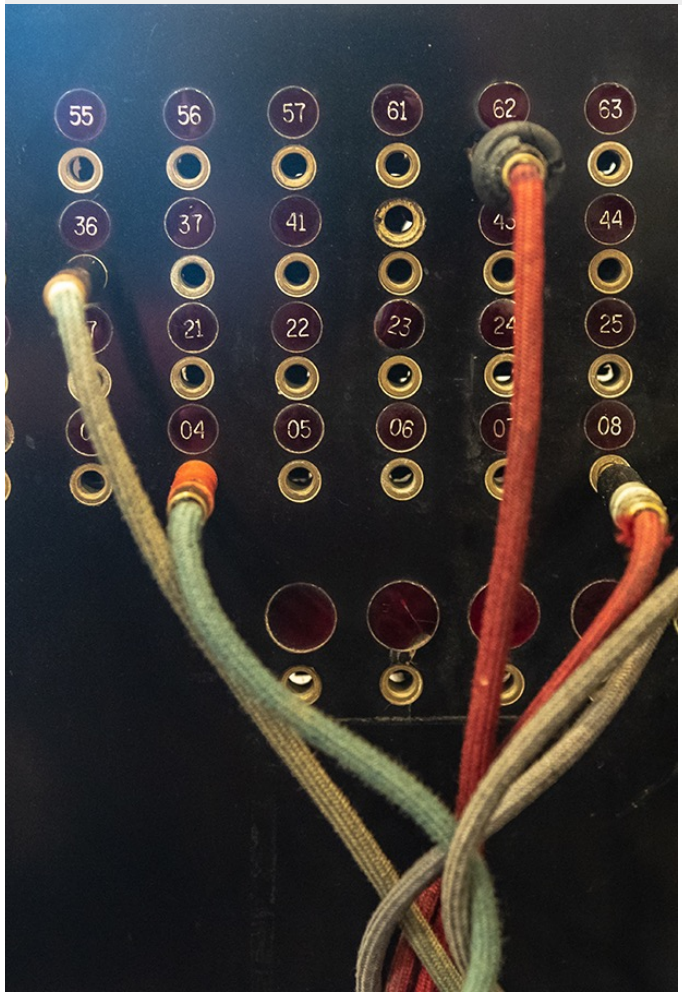
Introduction aux réseaux informatiques : paquets et couches de protocoles

- Explosion des réseaux : PC, smartphones, objets connectés
- Internet ≈ interconnexion de réseaux
- Internet ≠ Web !
- Objectifs du cours : paquets, protocoles, couches, routage, congestion

Réseaux : Internet

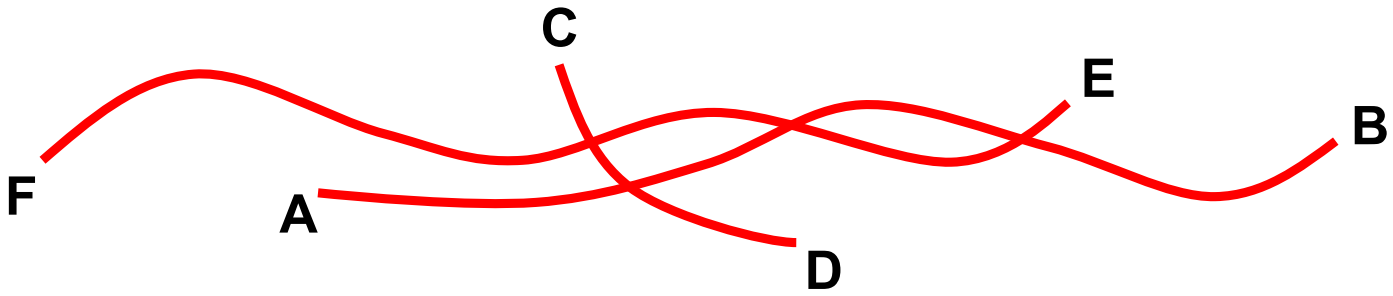


Réseau : Le téléphone



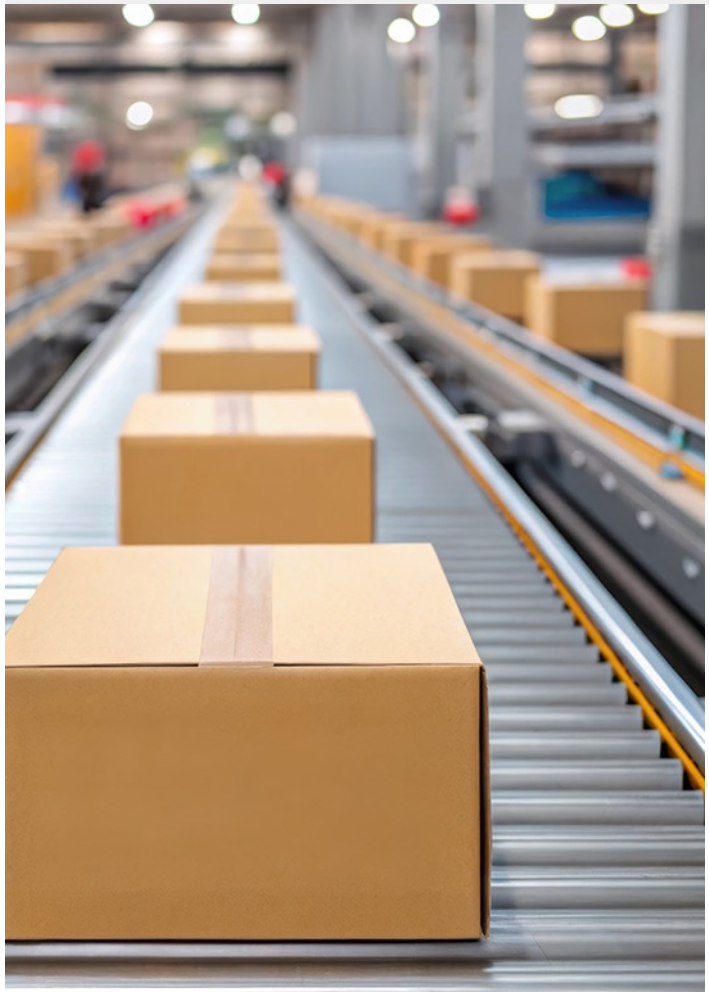
Principe :

- Établir une ligne de communication pour chaque paire d'utilisateurs désirant se parler
- Fermer celle-ci lorsque la communication est finie
  (et en ouvrir d'autres au fil des demandes)



Avec l'avènement des communications entre ordinateurs (**beaucoup plus de données** à échanger), ce système ne pouvait plus tenir...

Communication par paquets

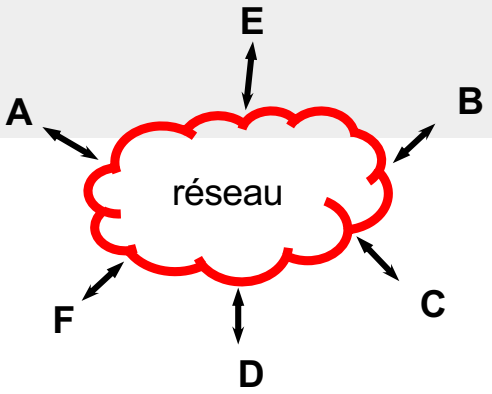


- Messages découpés en paquets
- Chaque paquet contient :
  - Données (payload)
  - En-tête (destination, numéro, etc.)
- Acheminement indépendant
- Besoin de fiabilité : erreurs, pertes, doublons

Structure d'un paquet

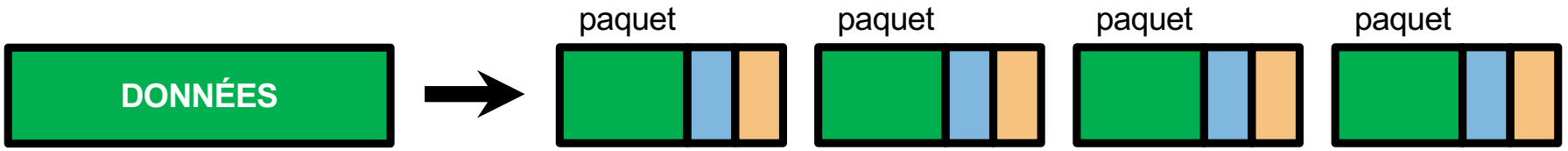
- En-tête + données (payload)
- Champs classiques :
  - Source, destination
  - Type, numéro de séquence
  - Contrôle d'erreur (checksum)

Réseau : Internet



Principe (schématique) de la communication par paquets :

Les données transmises par un utilisateur sont découpées
en **paquets**, qui sont ensuite envoyés dans le réseau :



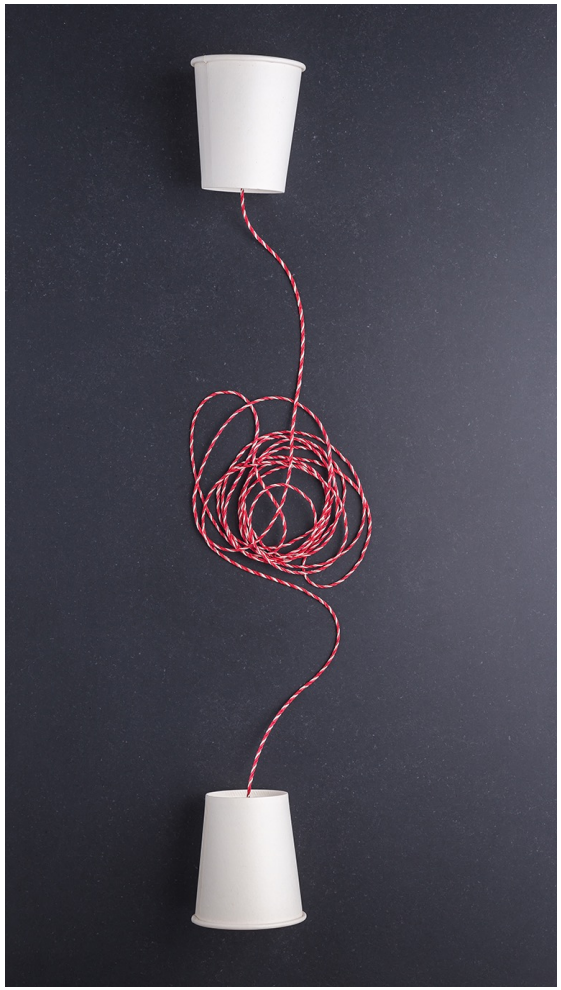
À chaque paquet on ajoute des *informations*, par exemple :

- Sa **destination**
- Son **identifiant**

(pour que le destinataire puisse remettre les paquets dans l'ordre à l'arrivée)

EPFL

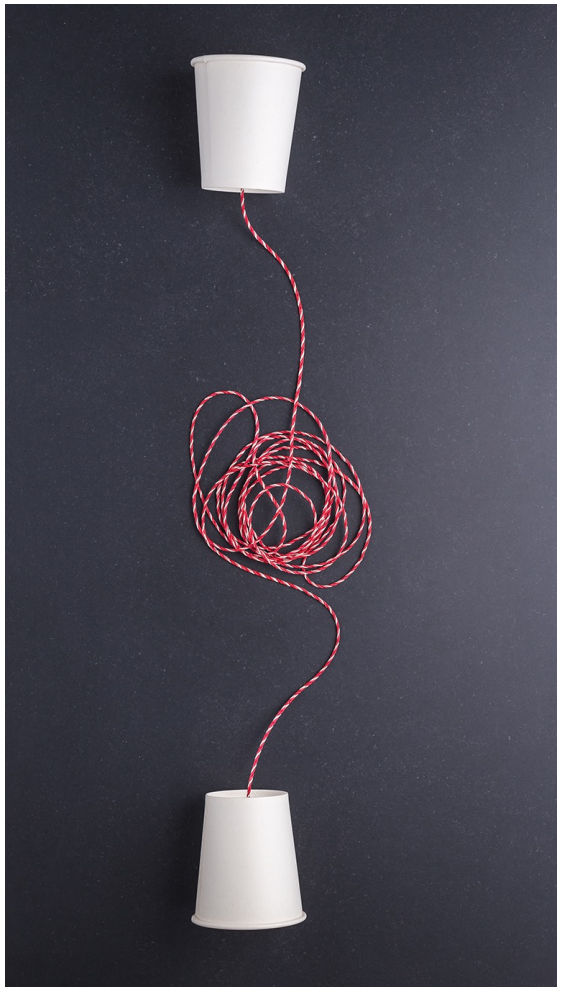
Notion de protocole



- Ensemble de règles pour communiquer

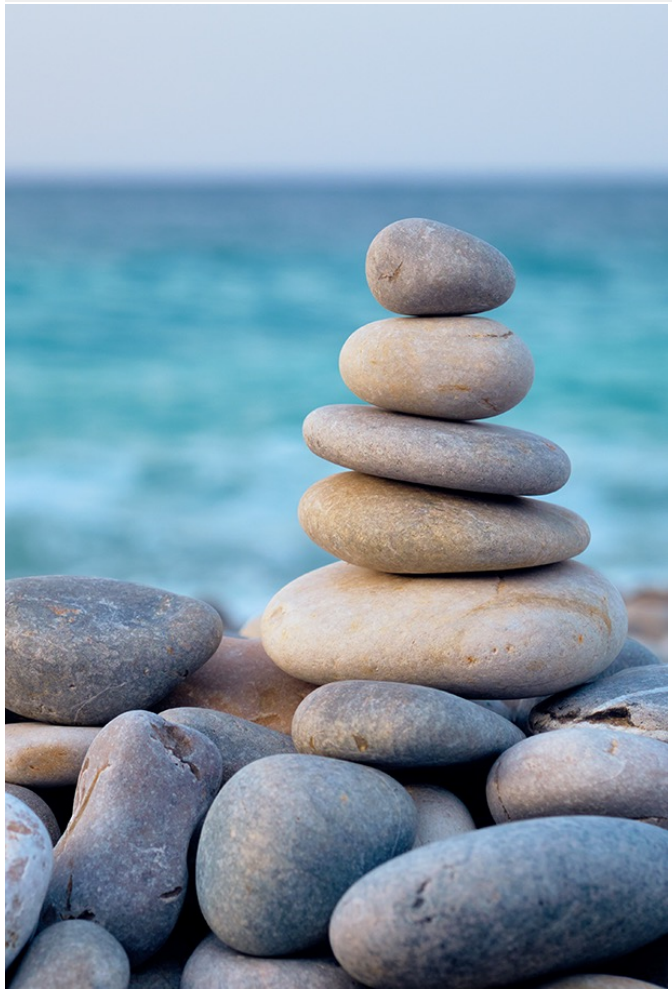
- Exemple humain : lever la main en classe
- Exemple informatique : ACK, retransmission, checksum

Téléphone à ficelle : un réseau… avec protocole !



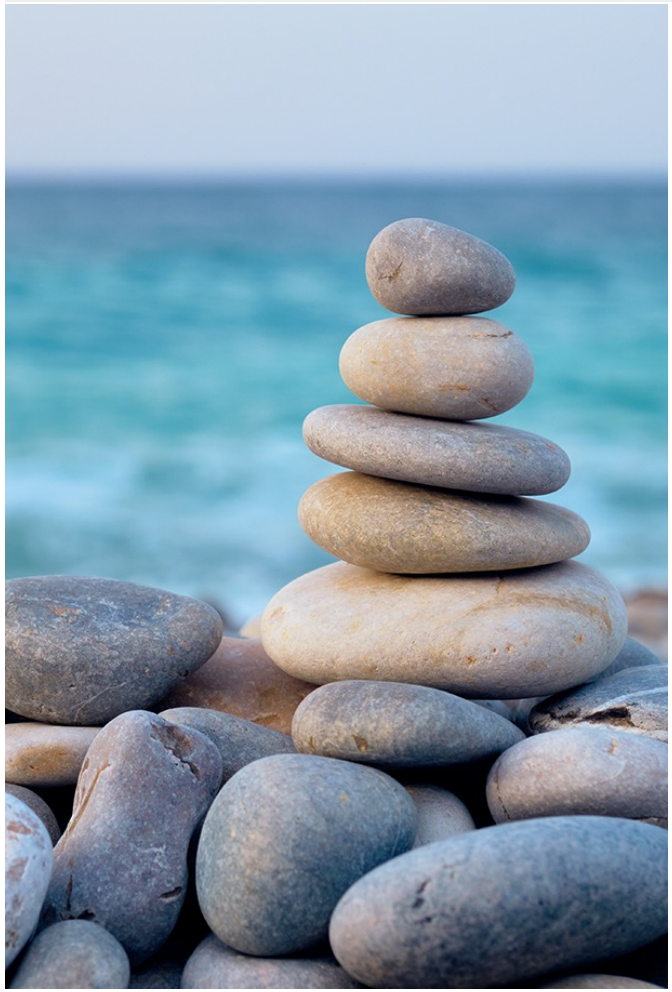
- Même avec deux gobelets reliés par une ficelle, on a besoin de règles pour que la communication fonctionne :
  - Accès au canal (couche liaison) : On ne peut pas parler tous les deux en même temps
    « *Tu parles,* je *t'écoute. Puis je réponds.* »
  - Détection d'erreur / demande de répétition si le son est étouffé
    « *Hein ? Tu peux répéter ?* »
  - Démarrage/fin de communication
    « *Allô ? T'es là ?* », « *J'ai fini !* »

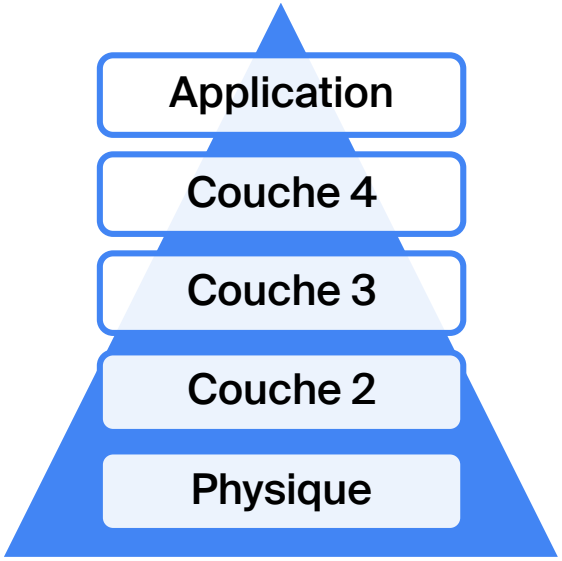
Pile de protocoles

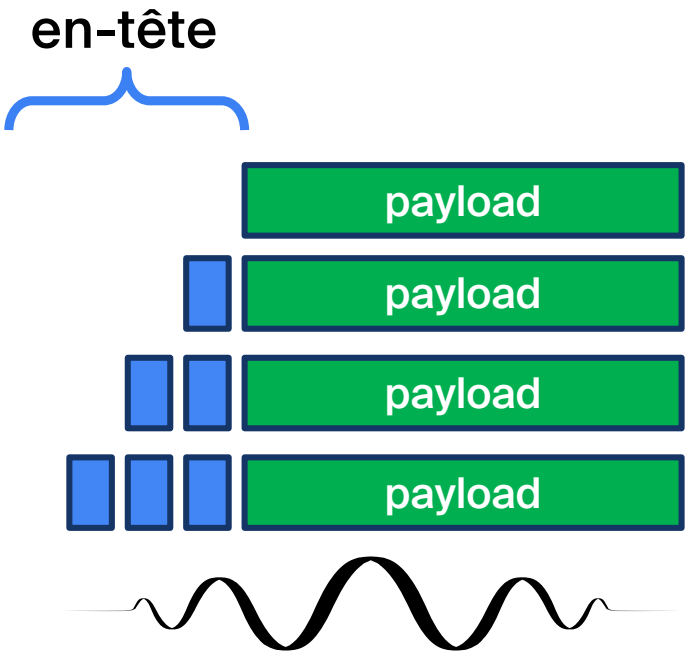


- Complexité → décomposition en couches
- Modèle TCP/IP (5 couches)
- Chaque couche ajoute un en-tête (encapsulation)
- Décapsulation à la réception

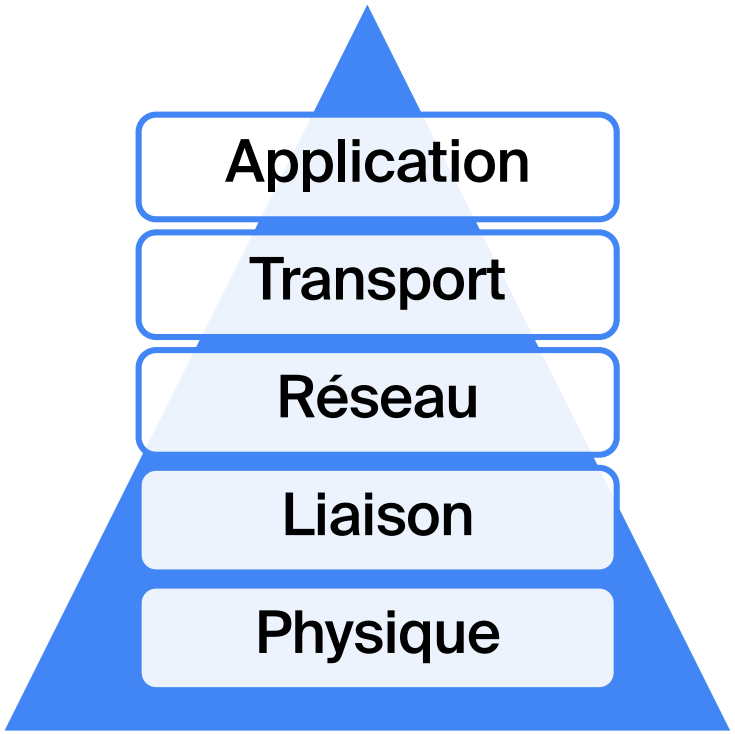
Pile de protocoles





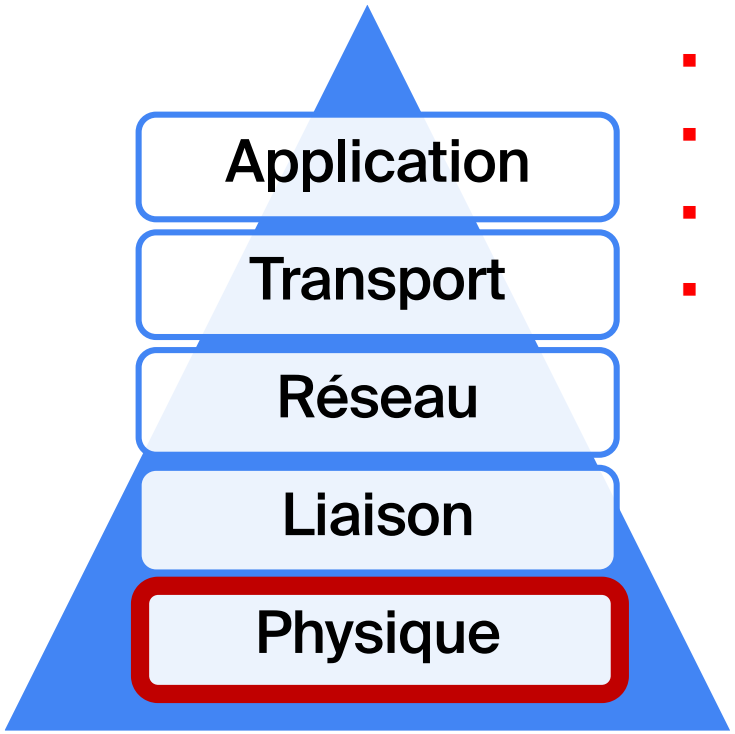


Les 5 couches TCP/IP

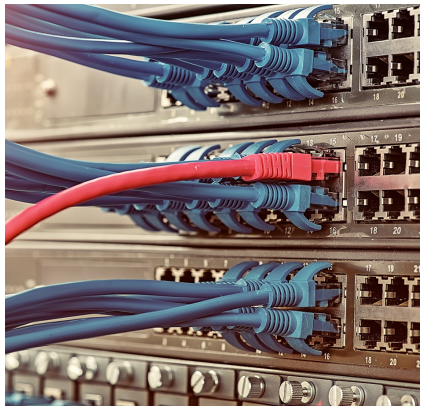


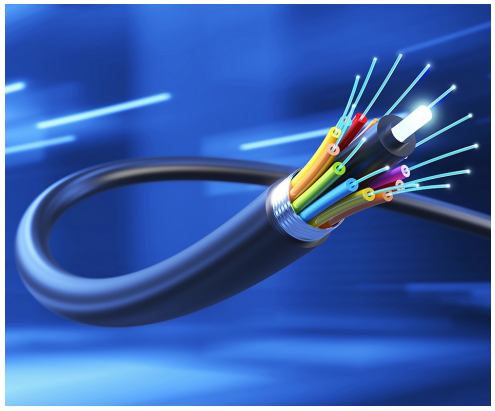
- Application (HTTP, mail, etc.)
- Transport (TCP, UDP)
- Réseau (IP)
- Liaison de données (Ethernet, Wi-Fi)
- Physique (câble, ondes)

Les 5 couches TCP/IP : Physique (couche 1)

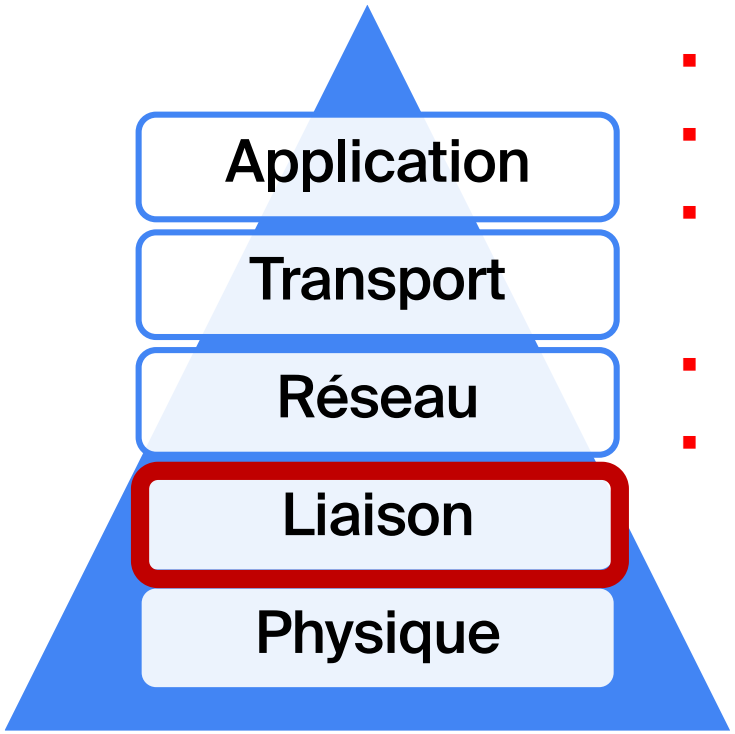


- Transmission brute des bits
- Support : ondes, fils, fibres optiques, radio...
- Pas d'en-tête : juste des signaux physiques (tension, lumière, ondes)
- Exemples : Ethernet à 1 Gbps, Wi-Fi, LTE, 5G, satellite, fibre optique

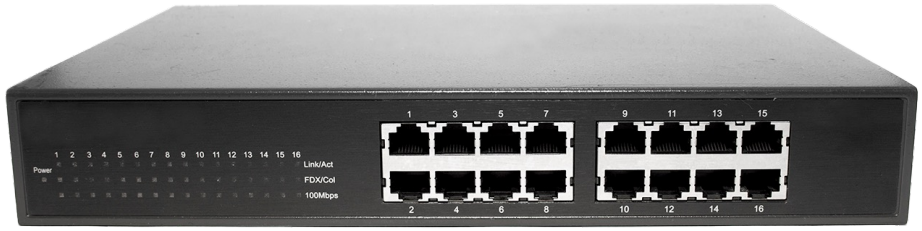




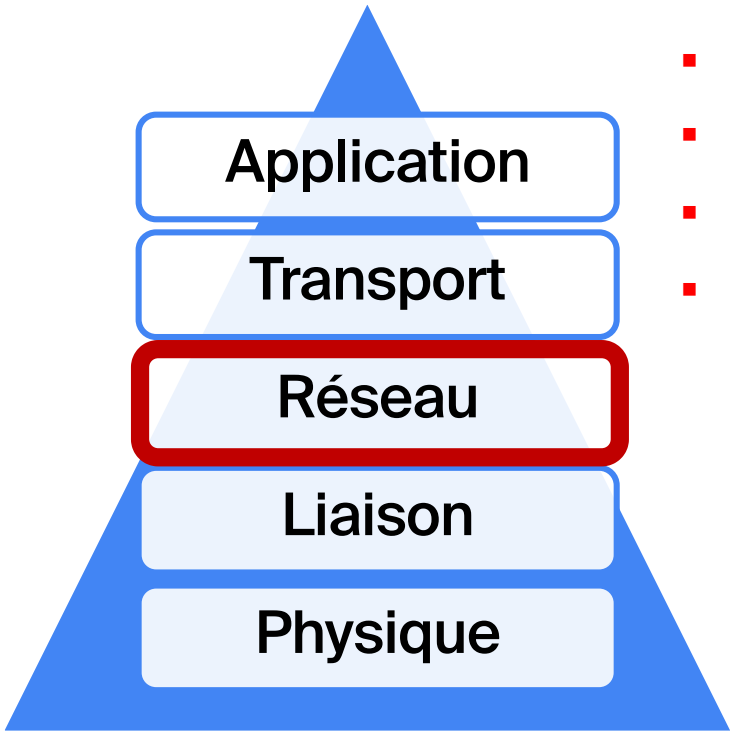
Les 5 couches TCP/IP : Liaison (couche 2)



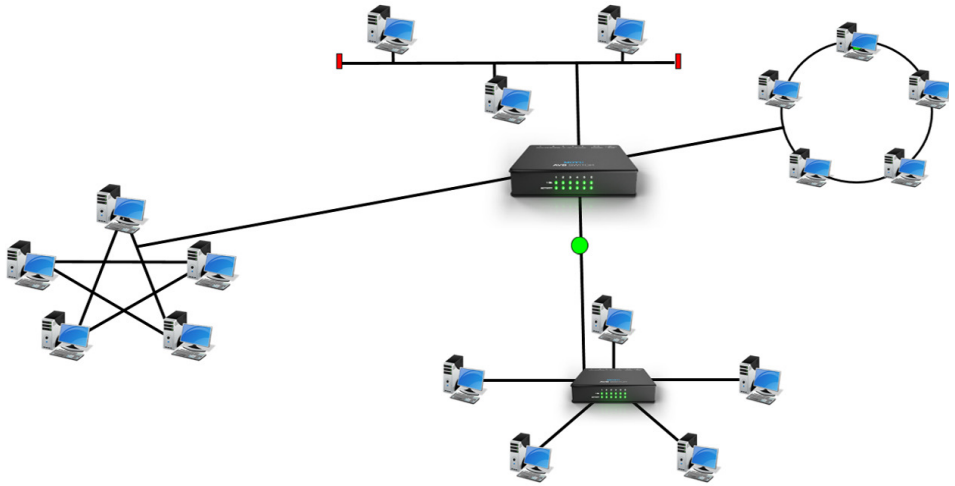
- Gère l'accès au canal (MAC), collisions, retransmissions locales
- Adresse locale : adresse MAC (ex.: `f8:f2:1e:aa:cd:c0`)
- Portée : concernée par la communication
  au sein d'un réseau local (LAN)
- Encapsulation : ajout d'un en-tête
- Exemples de protocoles : Ethernet, Wi-Fi (IEEE 802.11), TDMA, FDMA, CSMA/CD, PPP



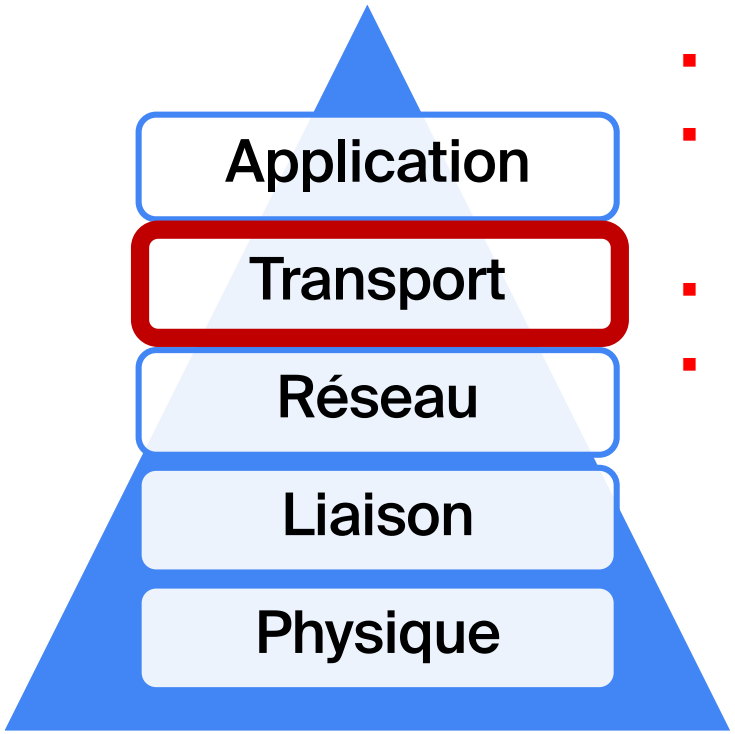
Les 5 couches TCP/IP : Réseau (couche 3)



- Routage des paquets de bout en bout (protocole IP)
- Adresse logique : adresse IP (ex. : `104.20.229.42`)
- En-tête IP avec adresses, TTL, etc.
- Rôle du routeur = lire l'en-tête et relayer

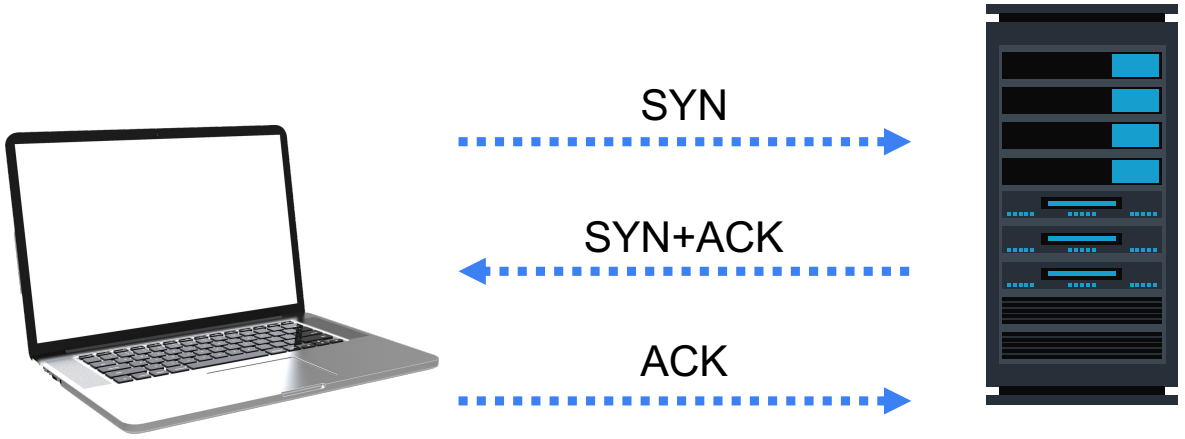


Les 5 couches TCP/IP : Transport (couche 4)

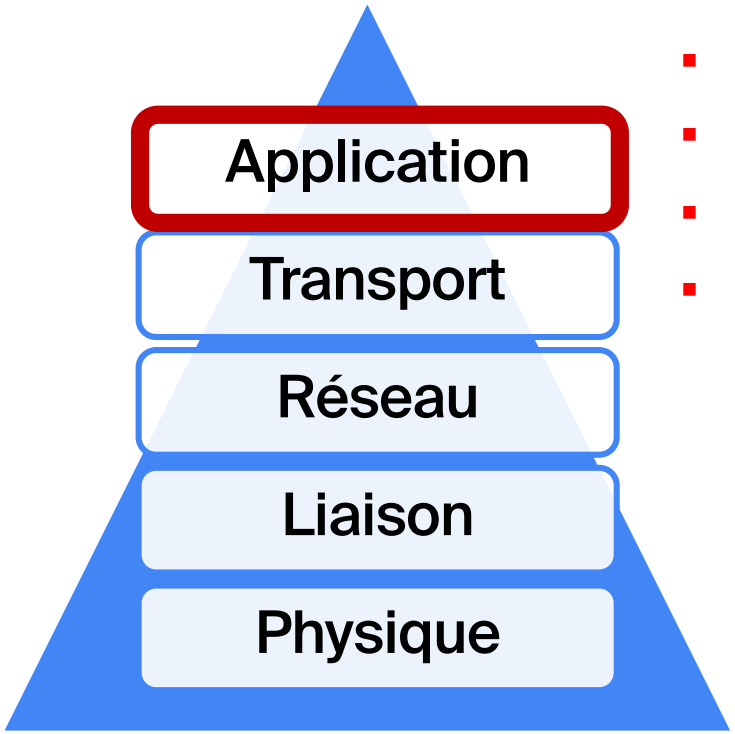


- Communication de processus à processus
- Adresse utilisée : numéro de port

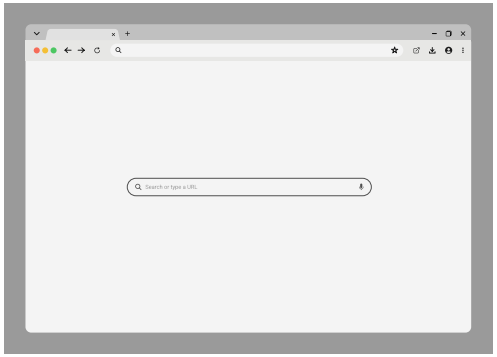
  identifie l'application cible (HTTP = 80, etc.)
- TCP : fiable, avec ACK, numérotation
- UDP : plus simple, sans vérification

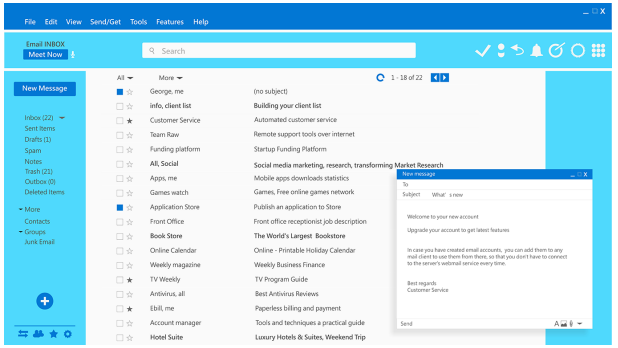


Les 5 couches TCP/IP : Application (couche 5)

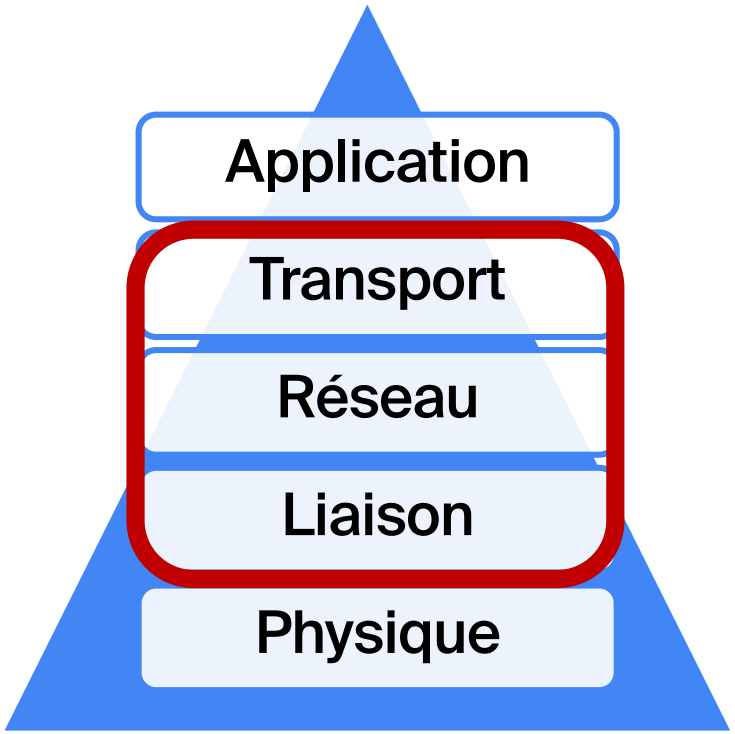


- Interface avec l'utilisateur
- Protocoles : HTTP, SMTP, FTP, DNS...
- Requêtes GET/POST, messages email, etc.
- Vue logique des échanges



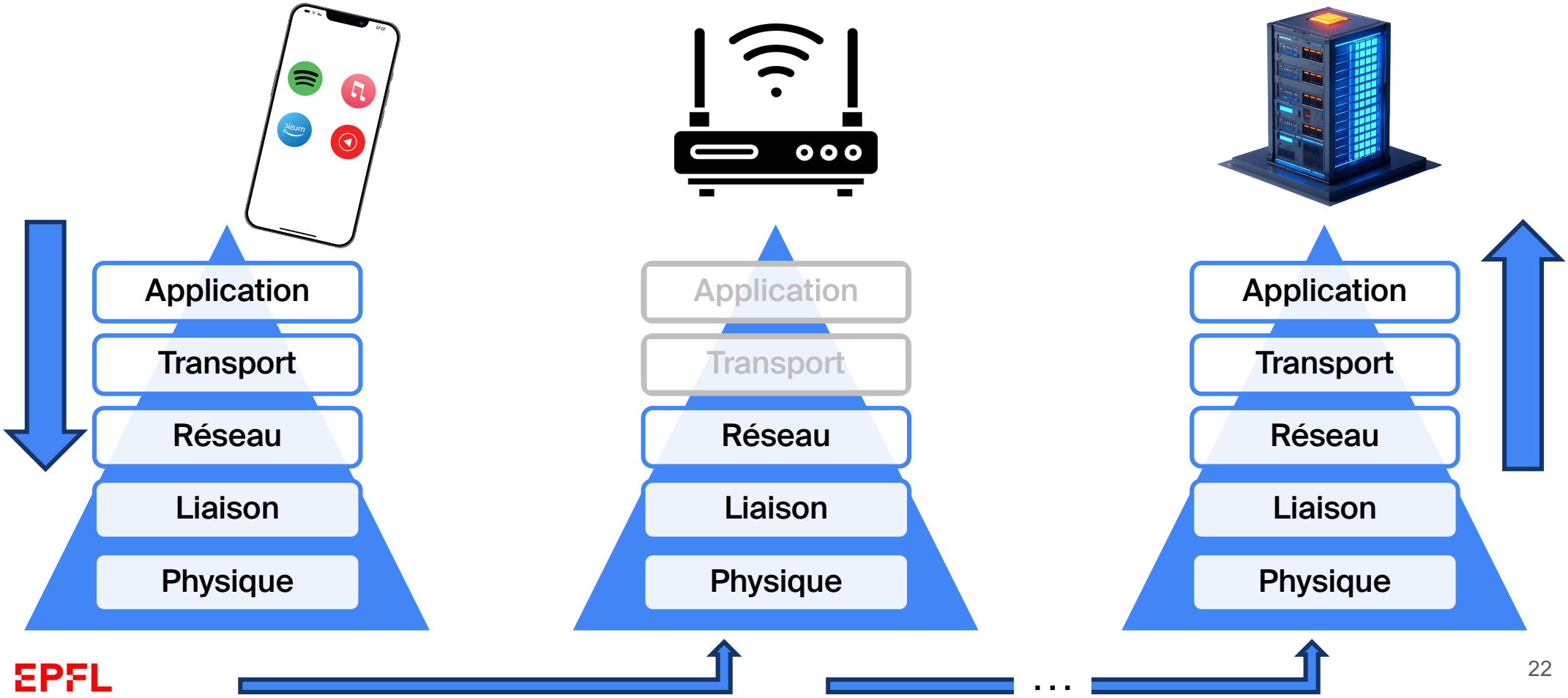


Les couches TCP/IP



Couche	Nom	Adresse	Portée
4	Transport	Port	Machine (processus)
3	Réseau	IP	Global (bout en bout)
2	Liaison	MAC	Local (réseau local)

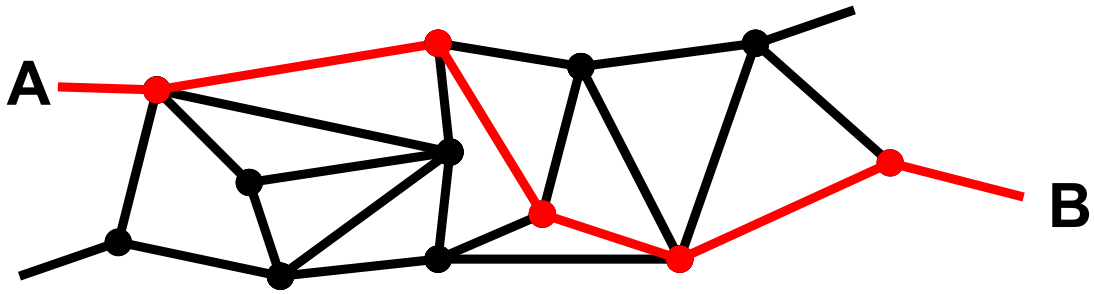
Pile de protocoles



Deux protocoles

Nous allons maintenant étudier plus attentivement :

1. La question de **l'acheminement** du paquet à travers le réseau par un algorithme de **routage** :



Protocole **IP**

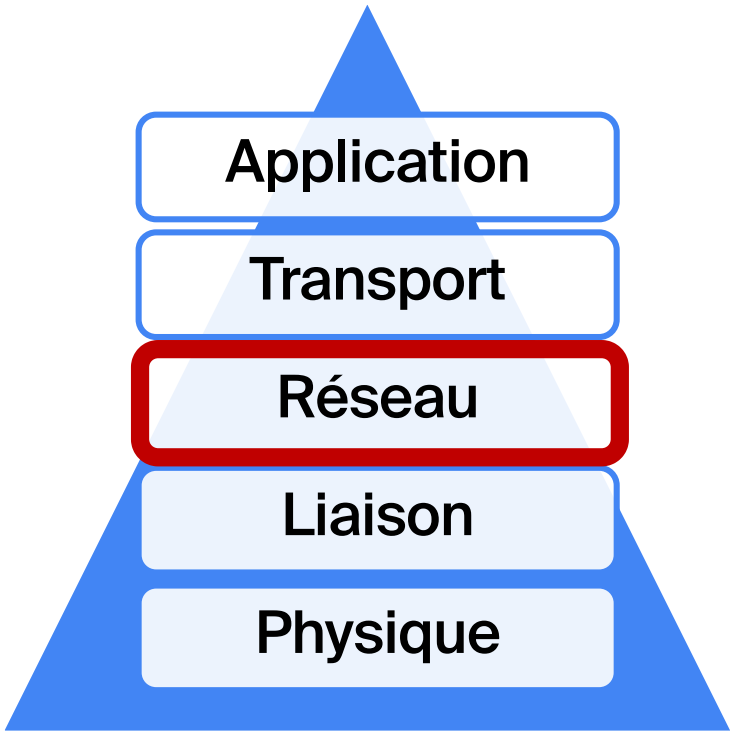
2. La question générale du **transport** d'un paquet d'une source A à une destination B :

Protocole **TCP**

Aujourd'hui

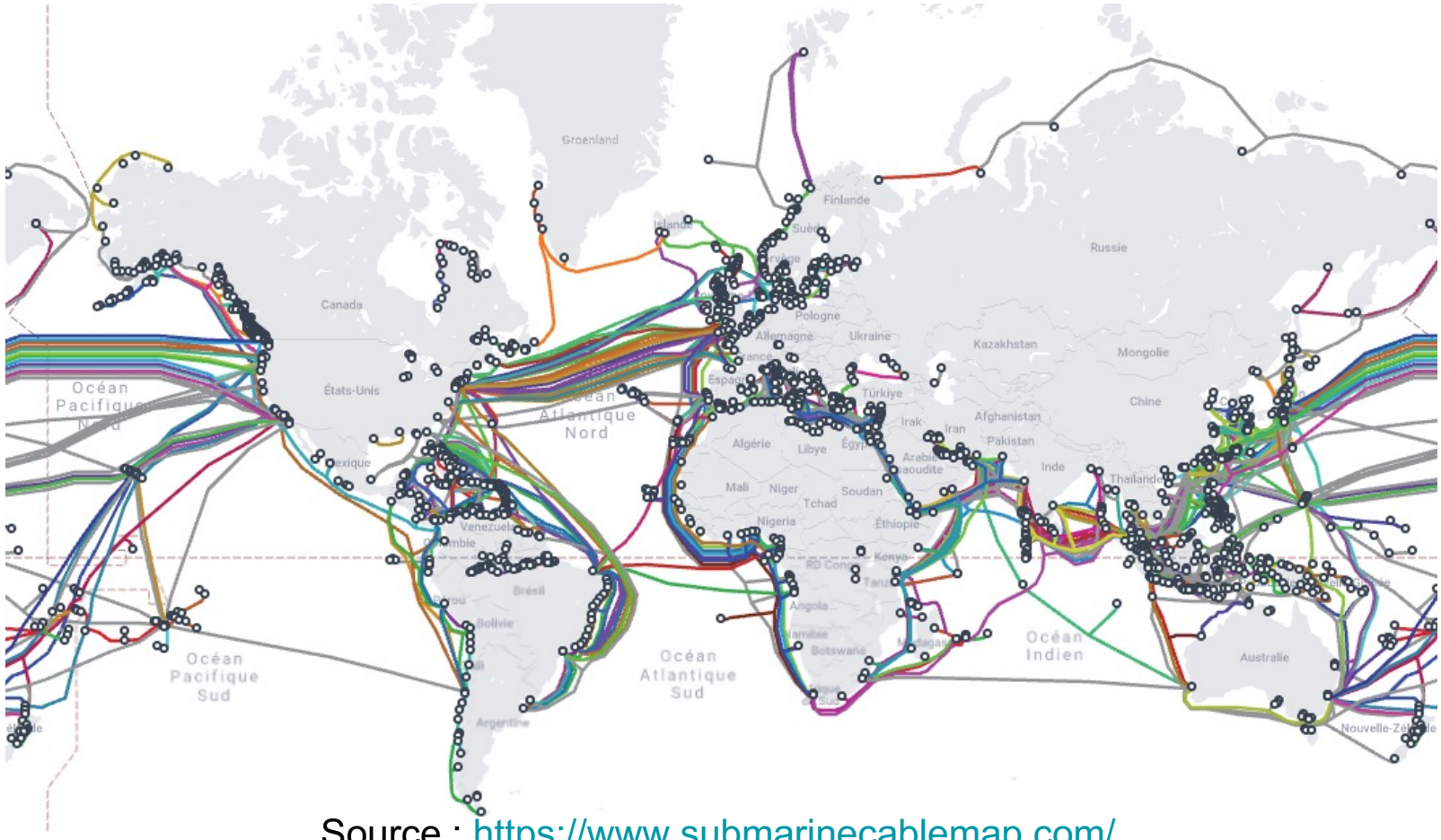
- Introduction aux réseaux informatiques : paquets et couches de protocoles
  - Comment les données circulent (paquets & couches)
- **Routage (Protocole IP)**
  - Comment les données trouvent leur chemin (routage/IP)
- **Congestion (Protocole TCP)**
  - Comment on évite l'engorgement (congestion/TCP)

Les 5 couches TCP/IP



- Application (HTTP, mail, etc.)
- Transport (TCP, UDP)
- Réseau (IP)
- Liaison de données (Ethernet, Wi-Fi)
- Physique (câble, ondes)

Routage - intuition



Source : https://www.submarinecablemap.com/

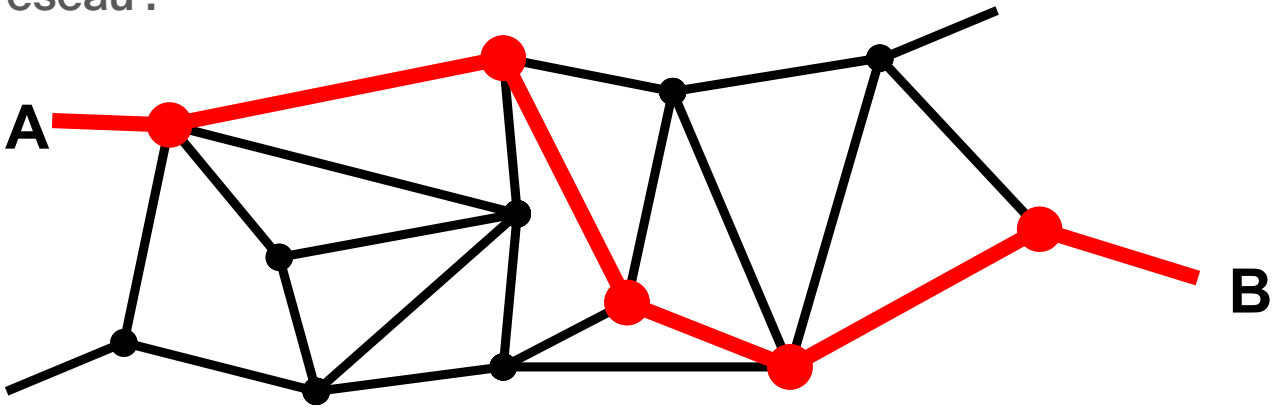
Routage - intuition

- Objectif : trouver le chemin de A à B
- Utilisation de graphes

  (nœuds = machines)
- Mesure : nombre de sauts (hops)

Protocole IP *(internet protocol)*

- Ce protocole est celui utilisé pour acheminer un paquet d'un nœud A vers un nœud B à travers le réseau :

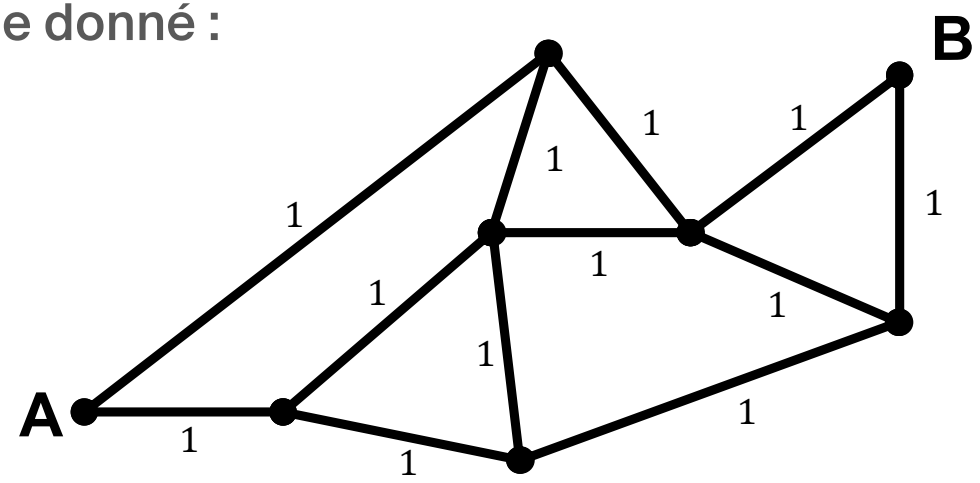


- Pour introduire le sujet, parlons d'abord du problème de **la recherche du plus court chemin dans un graphe**, ainsi que d'un algorithme possible pour la résolution de ce problème : l'algorithme BFS (Breadth First Search)

(« parcours en largeur »)

Algorithme BFS

Considérons un graphe donné :



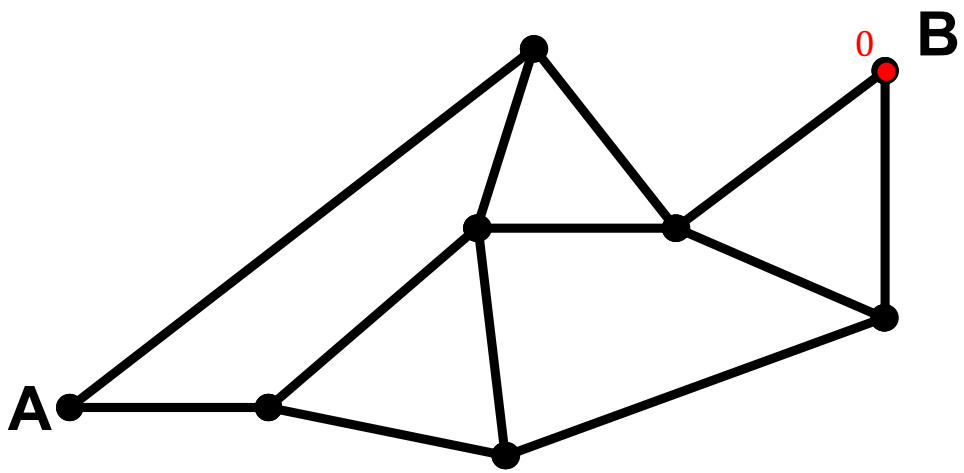
- Remarque :

Nous supposons ici que la distance entre deux nœuds directement reliés entre eux est toujours la même (disons 1).

Algorithme BFS

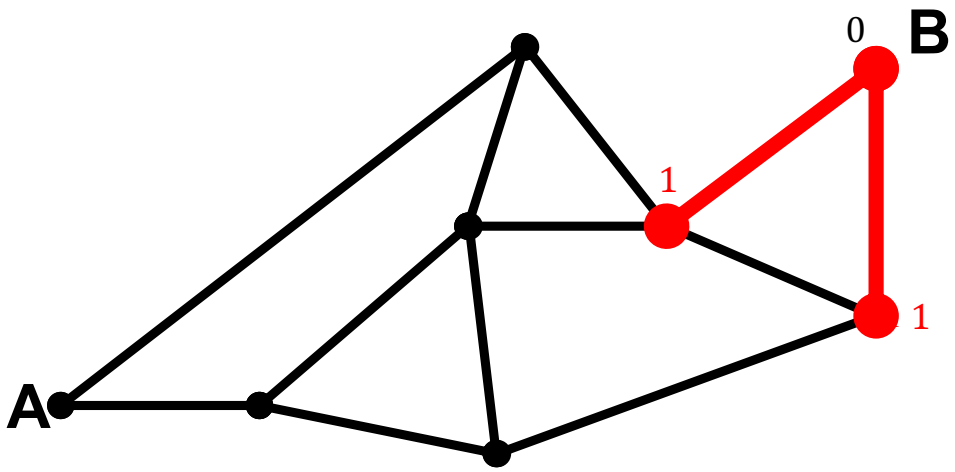
Pour trouver le chemin le plus court de A à B, l'algorithme BFS propose de chercher **tous les chemins les plus courts** de n'importe quel nœud du graphe au nœud B.

0. Initialisation : le nœud B est à distance 0 de lui-même.
   On marque celui-ci comme vu (pas besoin d'y revenir).



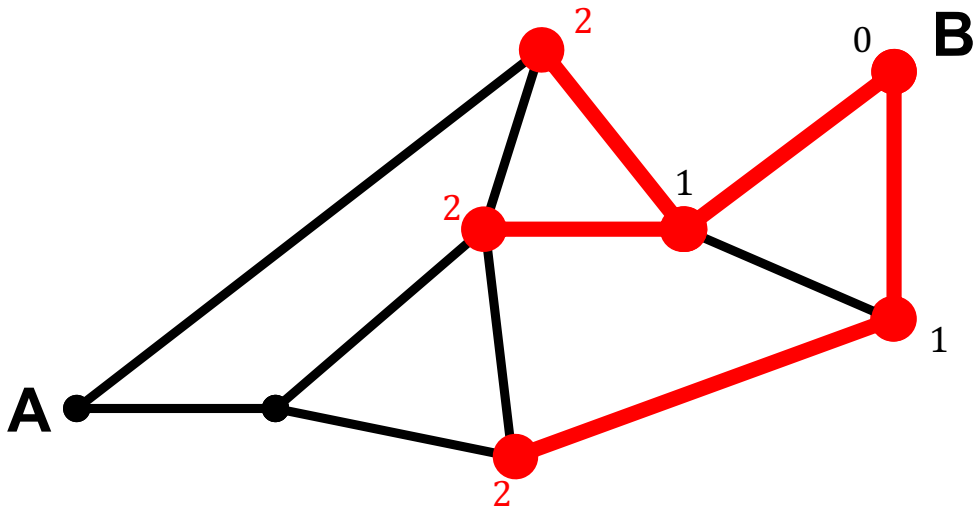
Algorithme BFS

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.



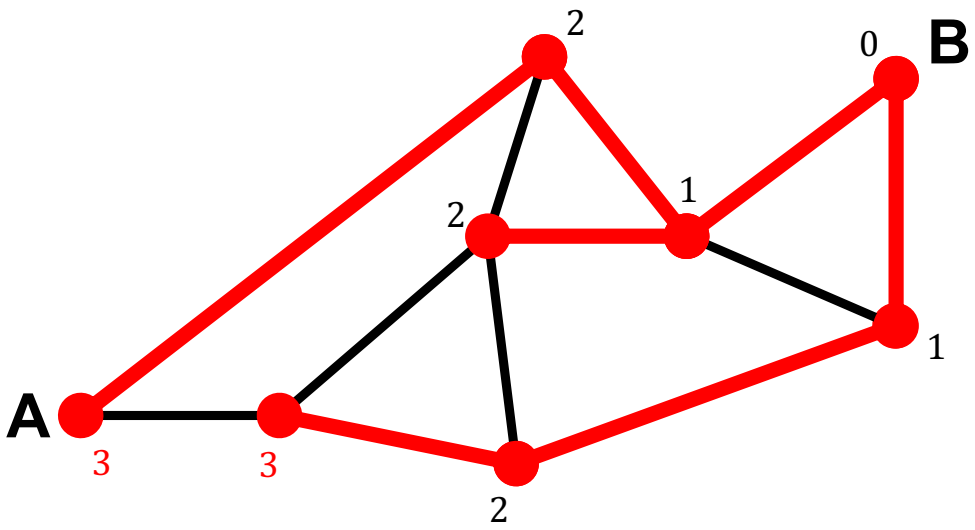
Algorithme BFS

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.
2. Les **voisins des voisins** sont à distance 2 de celui-ci. On les marque également.



Algorithme BFS

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.
2. Les **voisins des voisins** sont à distance 2 de celui-ci. On les marque également.
3. Et ainsi de suite... jusqu'à atteindre **tous les nœuds du graphe**, inclus le nœud A (ici, 3 étapes suffisent).



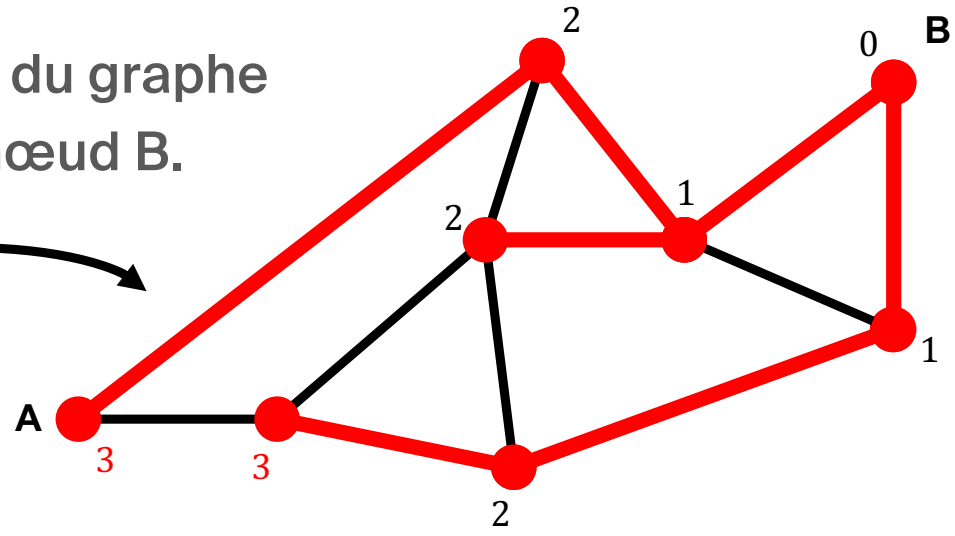
Algorithme BFS

0. Initialisation : le nœud B est à distance 0 de lui-même. On marque celui-ci comme vu.

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.
2. Les **voisins des voisins** sont à distance 2 de celui-ci. On les marque également.
3. Et ainsi de suite... jusqu'à atteindre **tous les nœuds du graphe**, inclus le nœud A.

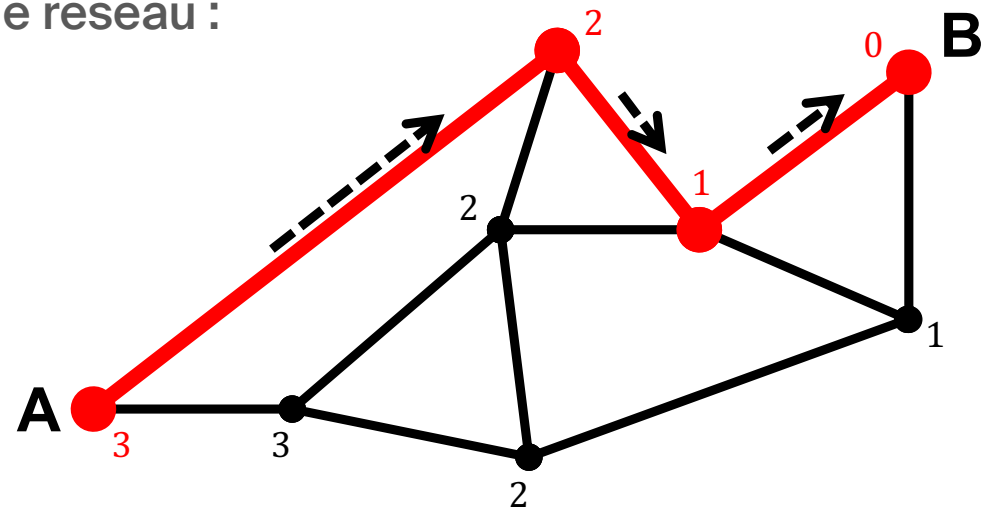
- À la fin de l'algorithme, tous les nœuds du graphe connaissent leur plus petite distance du nœud B.

arbre couvrant minimal !



Routage

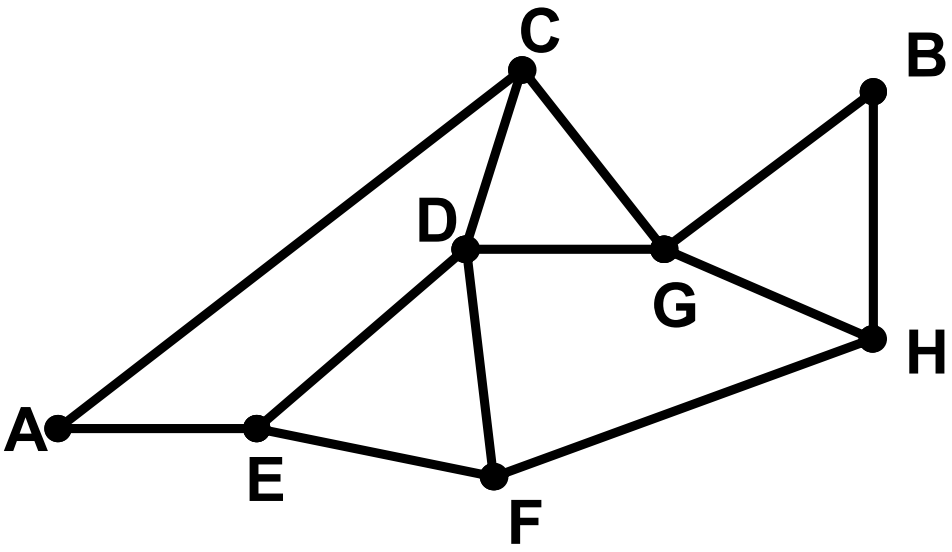
L'algorithme BFS nous donne maintenant une idée de comment acheminer les paquets à travers le réseau :



Supposons que A désire envoyer un paquet à B :
À quel voisin direct va-t-il choisir de transmettre le paquet ? À celui dont la distance à B est la plus petite ! (et ainsi de suite jusqu'à la destination)

Routage

Pour que cela fonctionne en pratique, il importe que chaque nœud du réseau maintienne à jour une **table de routage** contenant les informations suivantes :



Nœud A		
Destination	**Direction**	**Distance**
B	C	3
C	C	1
D	C ou E	2
E	E	1
...	...	...

Nœud C		
Destination	**Direction**	**Distance**
A	A	1
B	G	2
D	D	1
E	A ou D	2
...	...	...

Quelques remarques

- La mise à jour des tables de routage s'effectue **de manière distribuée**, chaque nœud vérifiant à intervalles réguliers les informations de ses voisins directs.

- En pratique, chaque nœud ne tient pas à jour une table avec toutes les destinations possibles, mais **seulement les destinations proches de lui** ; la gestion des destinations plus lointaines est déléguée à un nœud plus important (« gateway ») → **hiérarchie dans le réseau**

Adresses IP *(v4 sur 32 bits)*

Chaque nœud dans le réseau possède sa propre adresse IP, qui reflète la hiérarchie de celui-ci.

- Exemple :

$$172.16.254.1$$

(chaque nombre $\in \{0 \dots 255\}$)

8 bits 8 bits 8 bits 8 bits

$\rightarrow 32$ bits au total

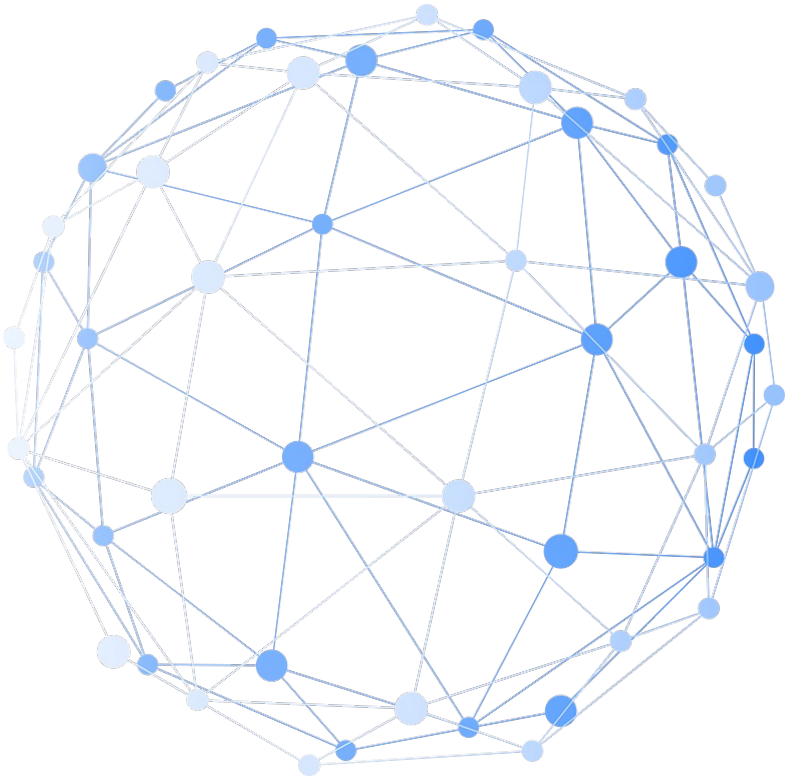
$\rightarrow 2^{32}$ adresses différentes

~ 4 milliards

réseau principal — sous-réseau — … — …

Mais pas assez en 2020 ! → IP v6, sur 128 bits (On espère assez pour quelque temps !)

Algorithme BFS (Breadth-First Search)

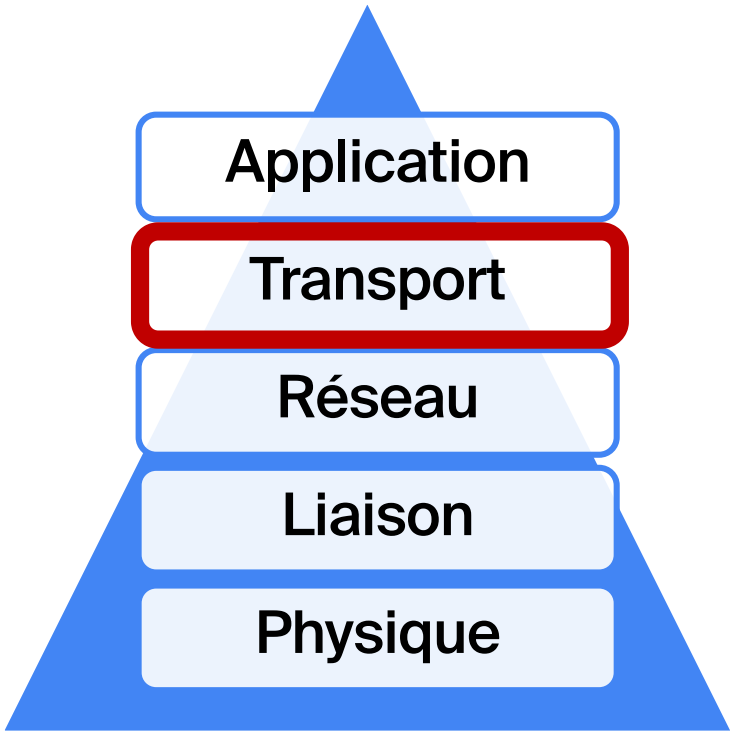


- Distance minimale depuis la destination
- Construction d'un arbre couvrant
- Permet à chaque nœud de connaître le meilleur voisin pour atteindre la cible
- Chaque machine stocke :
  - Destination
  - Voisin suivant (next hop)
  - Distance (nombre de sauts)
- Mise à jour par ouï-dire (périodique)

Aujourd'hui

- Introduction aux réseaux informatiques : paquets et couches de protocoles
  - Comment les données circulent (paquets & couches)
- Routage (Protocole IP)
  - Comment les données trouvent leur chemin (routage/IP)
- **Congestion (Protocole TCP)**
  - Comment on évite l'engorgement (congestion/TCP)

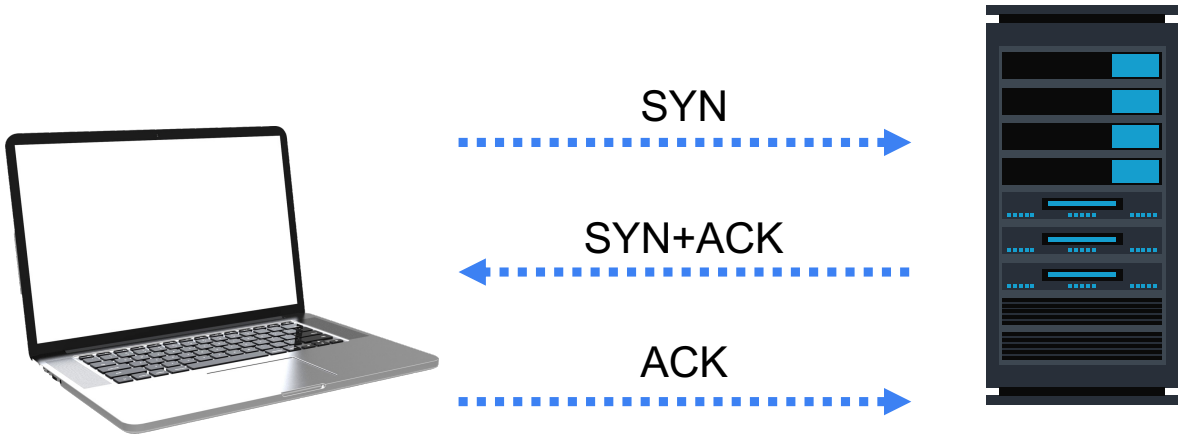
Les 5 couches TCP/IP



- Application (HTTP, mail, etc.)
- Transport (TCP, UDP)
- Réseau (IP)
- Liaison de données (Ethernet, Wi-Fi)
- Physique (câble, ondes)

Protocole TCP

- Communication fiable entre 2 machines
- ACK (accusés de réception)
- Time-out et retransmission
- Risques : perte de paquets ou d'ACK



Protocole TCP *(transmission control protocol)*

Pour communiquer d'un nœud A à un nœud B dans le réseau, les choses se passent **schématiquement** ainsi :

1. A envoie un paquet à B.
2. Si B reçoit le paquet sans erreurs, B renvoie un autre paquet pour notifier à A que celui-ci a été bien reçu.
3. De son côté, A attend la notification de B.

« Time out » : Si **après un certain temps** T, cette notification n'arrive pas, A tente alors de renvoyer le premier paquet à B.

Quels sont les problèmes qui peuvent survenir ?

- Le premier paquet envoyé par A se **détériore** ou **se perd**.
- Ce même paquet reste bloqué quelque part dans le réseau à cause d'un problème de **congestion** et arrive trop tard à B.
- Le paquet arrive entier et à l'heure à B, mais la **notification** de B elle-même détériore, se perd ou arrive trop tard à A (i.e. après le temps T).

Dans tous les cas décrits, A pensera (à juste titre ou non) que B n'a pas reçu le paquet et tentera de retransmettre le premier paquet au temps T.

Comment remédier à ces problèmes ?

- Si le paquet se **perd** (lors d'une collision ou autre), alors on ne peut pas faire grand chose...
- Et que faire pour éviter les problèmes de **congestion**, tout en utilisant au mieux la capacité du réseau ?
- Il s'agit de **bien régler le temps** T entre deux transmissions ! (et ceci pour chaque nœud du réseau)

Ici, il y a un compromis à faire :

- Si T est trop grand :

Il est très probable qu'une grande majorité des notifications seront reçues à temps (et donc que les paquets arriveront à destination), mais le temps entre deux transmissions sera trop long

→ **réseau sous-utilisé**

- Si T est trop petit :

Le temps entre deux transmissions devient très court et le réseau doit relayer beaucoup trop de paquets en même temps

→ **plus de congestion**

De plus, souvent les mêmes paquets sont renvoyés, augmentant par là-même la congestion !

Augmentation additive – retrait multiplicatif *(AIMD)*

- Comment bien régler la valeur de T ?
- Soit $W = \frac{1}{T}$ le nombre de paquets transmis par unité de temps, pour un nœud donné.

- **Schématiquement**, l'idée de l'algorithme AIMD est la suivante :

1. Tant que le nœud reçoit des notifications à temps, il **augmente légèrement** W

   $W \longrightarrow W + a$ **avec** $a > 0$ **fixé**

2. Dès qu'un paquet est perdu, le nœud **diminue fortement** W

   $W \longrightarrow b \cdot W$ **avec** $0 < b < 1$ **fixé également**

(Notez bien : on peut choisir librement les paramètres a et b)

Augmentation additive – retrait multiplicatif *(AIMD)*

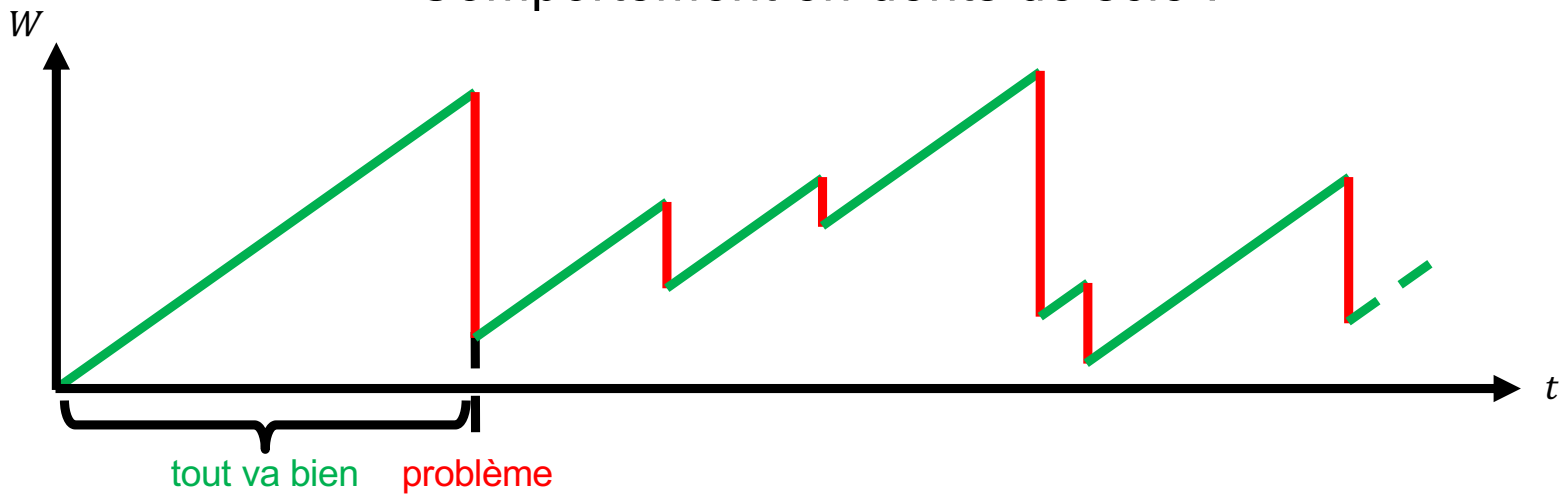
- Tant que tout va bien :

On **augmente légèrement** le nombre de transmissions
(pour utiliser au mieux le réseau)

- Dès qu'un problème survient :

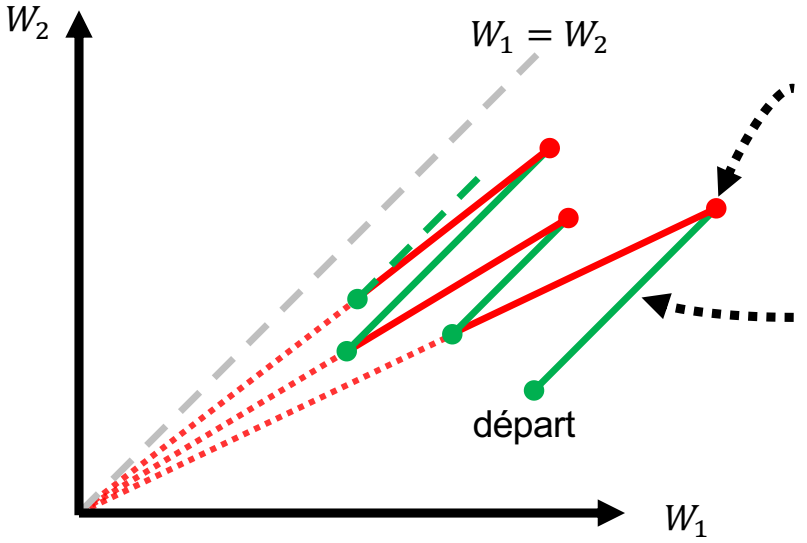
On **diminue fortement** le nombre de transmissions
(pour laisser sa place aux autres)

→ Comportement en dents de scie :



Pourquoi donc un tel algorithme ?

Supposons que deux nœuds, disons A_1 et A_2, utilisent simultanément cet algorithme, et observons l'évolution de W_1 et W_2 sur un même graphe :



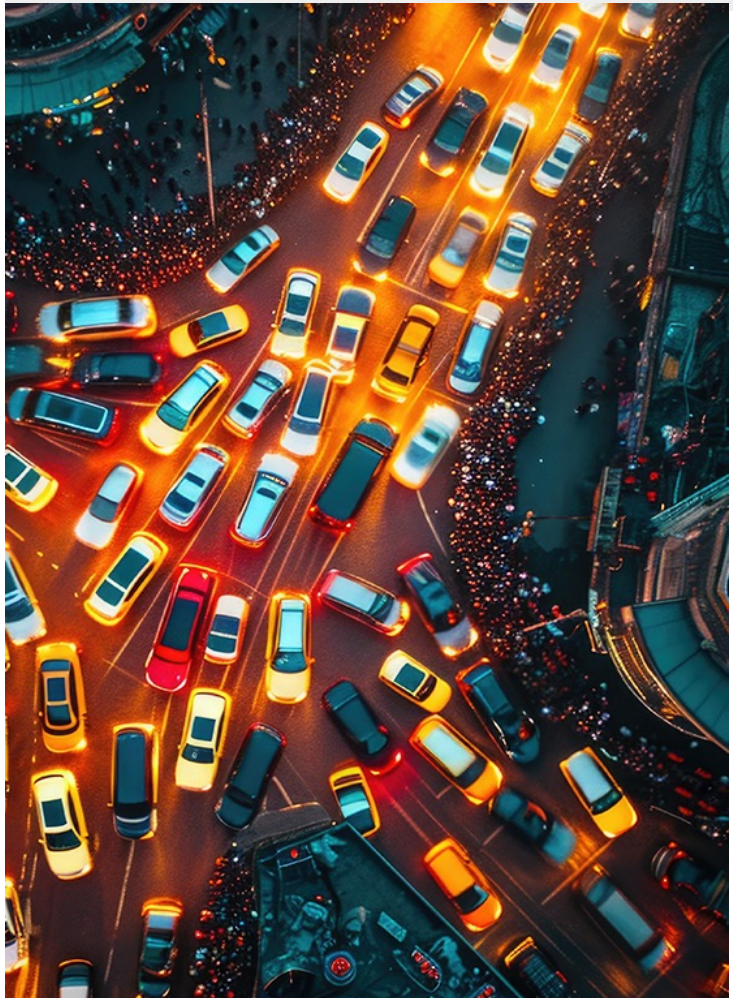
Un problème survient : A_1 et A_2 **divisent** W_1 et W_2 par la même valeur

Phase où tout va bien : A_1 et A_2 augmentent les valeurs W_1 et W_2 au même rythme, **additivement**

Au fur et à mesure, les valeurs de W_1 et W_2 se rapprochent :

→ **répartition équitable !** (et utilisation optimale du réseau)

Congestion et réglage du débit



- Problème : comment réagir face à la congestion ?
- Objectif : utiliser efficacement le réseau sans le saturer
- **AIMD (Additive Increase, Multiplicative Decrease)**
  - Si tout va bien : augmenter W lentement ($W \rightarrow W + a$)
  - Si congestion : diminuer W fortement ($W \rightarrow b \times W$)
  - Paramètres : $a > 0$, $0 < b < 1$
- Comportement en dents de scie
- Stabilisation dynamique
- Équité : convergence si plusieurs utilisateurs l'utilisent

Résumé Semaine 13 – ICC-T

- Un **réseau** permet à des machines de communiquer par **échange de paquets**
- La communication suit une **pile de couches** (TCP/IP) pour structurer les échanges
- Chaque couche a un **rôle spécifique** et ajoute (ou retire) un **en-tête**
- Les **adresses** changent selon la couche : MAC (locale), IP (globale), ports (processus)
- Le **routage** permet de trouver le chemin à travers le réseau (IP + tables)
- Le protocole **TCP** assure une communication fiable (ACK, retransmission, timeout)
- En cas de congestion, le réseau s'auto-régule avec **AIMD** : équité et stabilité

Évaluation approfondie

- Jusqu'au dimanche 7 juin
  - Connectez-vous à Moodle et restez sur la page d'accueil (tableau de bord, pas la page du cours)
  - Cliquer sur la flèche en haut à droite de l'écran qui fera apparaître un bloc contenant la tuile intitulée « Évaluation approfondie »

- Donnez un retour sur le cours
  - Feedback anonyme
  - Dites ce qui vous a plu, ce qui vous a déplu, et vos suggestions d'amélioration
  - Soyez constructifs : c'est votre opportunité d'aider à améliorer ce cours à l'avenir

rafael.pires@epfl.ch

Merci