Information, Calcul et Communication

Correction d'erreurs : principes de base

Olivier Lévêque

École polytechnique fédérale de Lausanne

Comment transmettre des données ?

Il existe deux moyens de transmission :

- A travers le **temps** : enregistrement sur un support
  (et les données peuvent lues plus tard)
- A travers **l'espace** :
  - par câble électrique ou fibre optique (téléphonie, internet)
  - dans l'air (téléphonie sans fil, wifi)

Problème commun : des erreurs surviennent régulièrement !

- lors de l'écriture ou de la lecture de données
- lors de la transmission par câble électrique, fibre optique ou onde électromagnétique

Exemple

- Vous désirez communiquer la direction à prendre (N,S,E,W) à un ami perdu.

Code binaire utilisé :

N	S	E	W
11	10	01	00

- Vous indiquez à votre ami d'aller au nord en envoyant 11.
- Si votre ami reçoit :

11 Tout se passe bien : votre ami se dirige vers le nord.

1? Un **effacement** survient : votre ami ne sait pas s'il doit se diriger vers le nord ou vers le sud.

10 Une **erreur** survient : votre ami se dirige alors vers le sud !

Solution : ajouter de la redondance !

« *Mieux vaut prévenir que guérir* » :

→ Ajouter de la redondance au message que l'on veut envoyer

→ *Mieux :* Ajouter **un minimum de redondance,**
pour prévenir **un maximum d'erreurs**

Exemples de redondance dans la vie courante

- Un père à ses enfants :

« Mettez vos chaussures… mettez vos chaussures, j'ai dit ! »

- Vous à votre ami :

« Pour venir chez moi, c'est simple :
tu tournes dans la 2e rue à droite après le feu rouge, juste après la station-service ;
j'habite au numéro 3, dans l'immeuble bleu avec les grands balcons »

Les informations en rouge sont redondantes (donc inutiles en théorie…).

Corriger un effacement

- Codage par répétition

N	S	E	W
1111	1100	0011	0000

  - 11?1 → N
  - Simple mais coûteux : il faut envoyer 4 bits au lieu de 2 !

- Bit de parité

N	S	E	W
110	101	011	000

  - 1?0 → N
  - Le dernier bit correspond à la somme modulo 2 des deux premiers.

Pour envoyer un message de n bits et corriger un effacement, il faut ajouter :

$$n\ bits \quad | \quad 1\ bit\ de\ parité$$

La taille finale du message est :

$$2n\ bits \quad | \quad n+1\ bits$$

Corriger une erreur

- Doubler chaque bit
  - 1101 → N ou S ??

N	S	E	W
1111	1100	0011	0000

- Tripler chaque bit
  - 111101 → N

N	S	E	W
111111	111000	000111	000000

On applique ici la **règle de la majorité** : 111101 → N 111100 → S

Mais à nouveau, cette solution est très coûteuse :

Chaque bit est triplé, on aura donc besoin de $3n$ bits pour envoyer un message de n bits à l'origine !

Corriger une erreur

Un meilleure méthode : ajouter *plusieurs* bits de parité

Voici un essai :

- Lorsque $n = 4$: on ajoute 2 bits de parité
  - le premier indique la parité de la somme des bits 1 et 2
  - le deuxième indique la parité de la somme des bits 1 et 3

Exemple : Pour envoyer 1101, on envoie 110101.

- Si on reçoit 100101 :
  - bit 1 et 2 : $1 + 0 = 1 \neq 0$
  - bit 1 et 3 : $1 + 0 = 1$

} On déduit que le bit 2 est faux :
Le message d'origine est donc 110101

Corriger une erreur (suite)

Un meilleure méthode : ajouter *plusieurs* bits de parité

Voici un essai :

- Lorsque $n = 4$: on ajoute 2 bits de parité
  - le premier indique la parité de la somme des bits 1 et 2
  - le deuxième indique la parité de la somme des bits 1 et 3

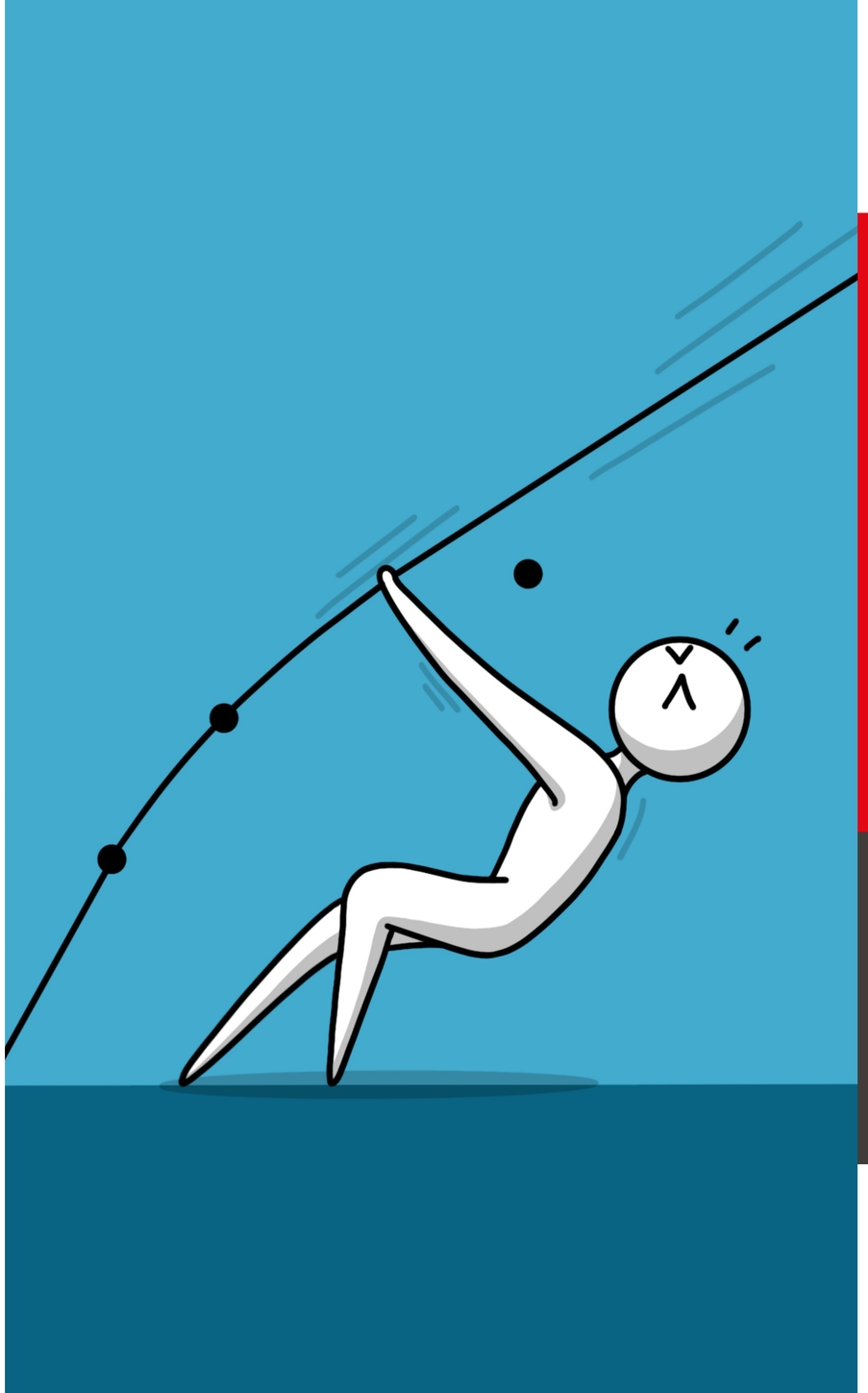
Problème : Pour envoyer 1101, on envoie toujours 110101.

- Si l'erreur survient sur le bit 4, on reçoit alors 110001, mais les deux bits de parité sont corrects → erreur indétectée !

- Pour réparer ce problème, *3 bits de parité* sont en fait nécessaires → code de Hamming

Corriger plusieurs effacements ou erreurs ?

Tout l’art de la théorie du codage (70 ans d’histoire…) consiste à choisir parcimonieusement les bits de parité pour corriger un nombre maximum d’erreurs…



Information, Calcul et Communication

Les codes de Reed-Solomon

École polytechnique fédérale de Lausanne

Olivier Lévêque

Introduction

- Deux personnes (disons Alice et Bob) cherchent à communiquer, mais une proportion non-négligeable des informations transmises par Alice sont effacées/bruitées avant d'être reçues par Bob:

- Les codes de Reed-Solomon sont un bon moyen de gérer une telle situation.
- Pour simplifier leur description, nous allons supposer qu'Alice et Bob travaillent avec des moyens de communication capables de manipuler des *nombres réels* (et non des bits).

Protocole de communication : Envoi

- *Avant de communiquer*, Alice et Bob se mettent d'accord sur un ensemble de n nombres réels *tous différents* :

$$\{t_1, t_2, t_3, \dots, t_n\}$$

- Alice désire envoyer un message x composé d'une suite de nombres réels :

$$x = \{x_1, x_2, x_3, \dots, x_k\} \quad \text{avec} \quad k < n$$

- Elle définit le polynôme suivant :

$$P(t) = \sum_{i=1}^{k} x_i \cdot t^{i-1} \qquad \deg(P) \leq k - 1$$

- et envoie finalement le message :

$$y = \{y_1 = P(t_1)\,,\ y_2 = P(t_2)\,,\ y_3 = P(t_3)\,, \dots,\ y_n = P(t_n)\}$$

Protocole de communication : Réception

- Bob reçoit le message y. On suppose que $n - k$ nombres du message sont effacés; il reçoit donc que k nombres de y. Pour simplifier, admettons de plus que Bob ne reçoive que les k premiers nombres $\{y_1, y_2, y_3, \dots, y_k\}$.
- Le but de Bob est de retrouver $\{x_1, x_2, x_3, \dots, x_k\}$ à partir de $\{y_1, y_2, y_3, \dots, y_k\}$

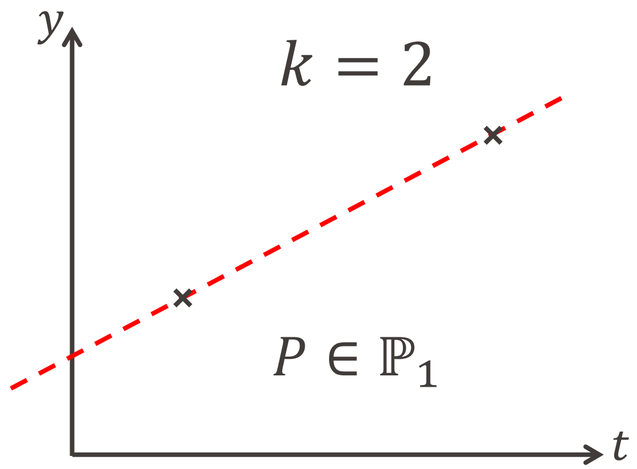
$$\text{Rappelons que } y_j = P(t_j) = \sum_{i=1}^{k} x_i \cdot {t_j}^{i-1} \quad \text{pour} \quad 1 \leq j \leq k$$

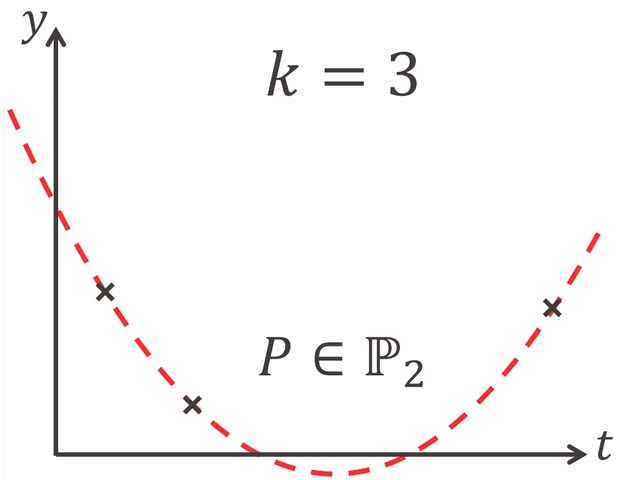
- Il s'agit donc de résoudre un système linéaire de k équations à k inconnues!

$$\text{Exemple pour } k = 3 \text{ et } t = \{1, 2, 3\} : \begin{cases} x_1 \cdot 1 + x_2 \cdot 1 + x_3 \cdot 1 = y_1 \\ x_1 \cdot 1 + x_2 \cdot 2 + x_3 \cdot 4 = y_2 \\ x_1 \cdot 1 + x_2 \cdot 3 + x_3 \cdot 9 = y_3 \end{cases}$$

Une autre façon de visualiser le problème

- En réalité, Bob reçoit les coordonnées des points (t_j, y_j) $\forall\ 1 \leq j \leq k$. Il s'agit donc de trouver l'unique polynôme P tel que $\deg(P) \leq k - 1$ passant par les k points différents.



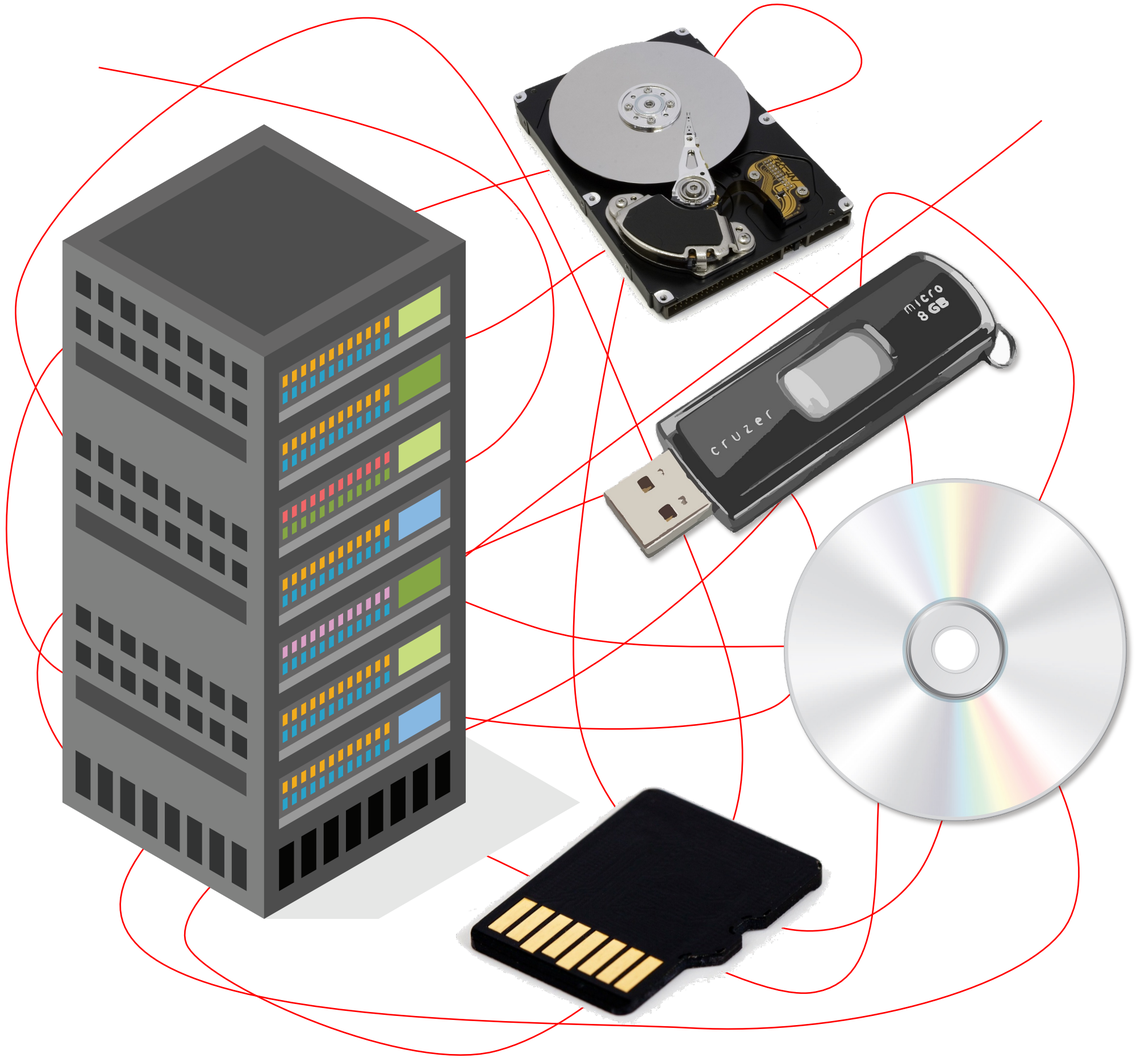


- Ainsi, Bob retrouve le message envoyé x (composé des coefficients du polynôme P).

En pratique

- On ne travaille pas avec des nombres réels, mais avec des nombres entiers modulo p (où p est un nombre premier) : $\{0,1,2,\ldots,p-1\}$, ou plus généralement des nombres appartenant à un groupe fini.
- Et les mêmes principes concernant les polynômes s'appliquent dans ce cadre.

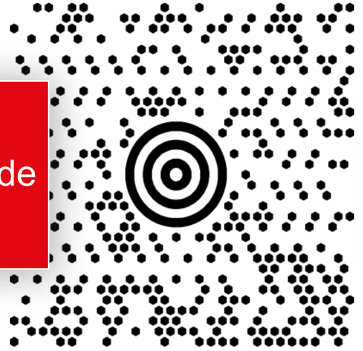
Application : Stockage de données et codes-barres 2D



QR code

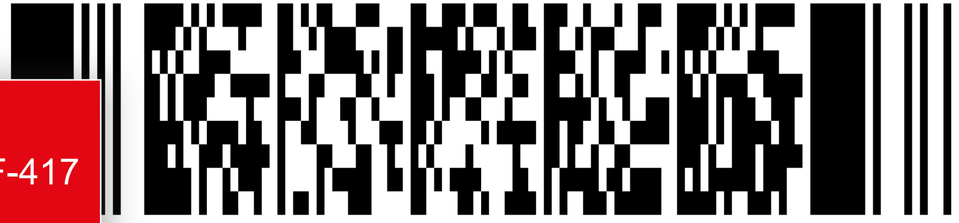
Aztec Code

MaxiCode



Data Matrix

PDF-417



Information, Calcul et Communication

Réseaux : Protocole TCP

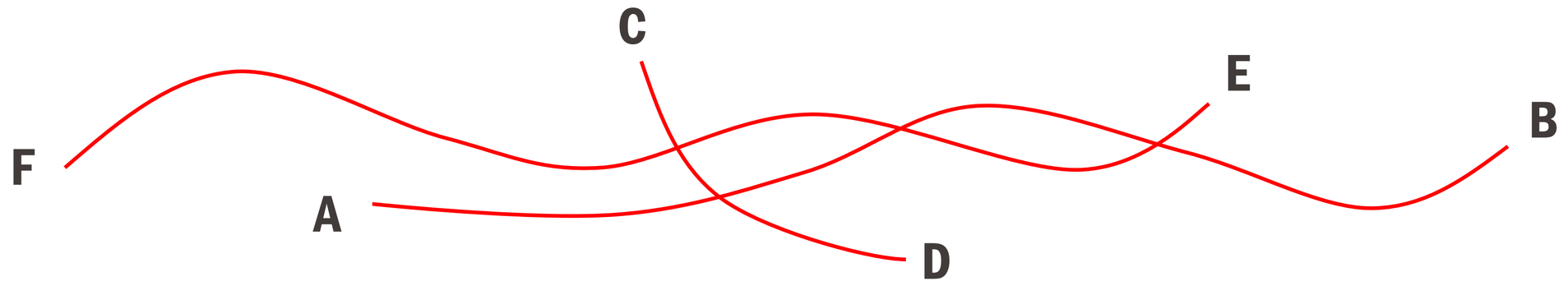
École polytechnique fédérale de Lausanne

Olivier Lévêque

Réseau : Le téléphone

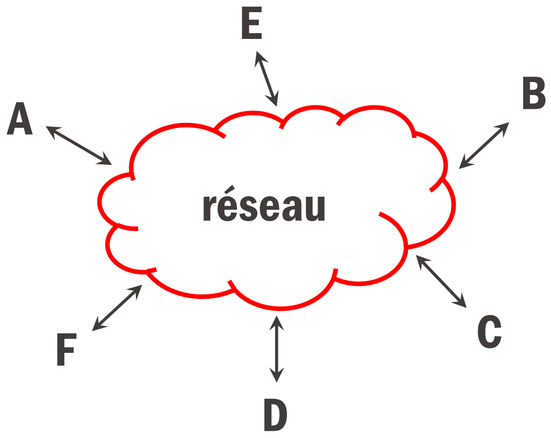
Principe :

- 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)



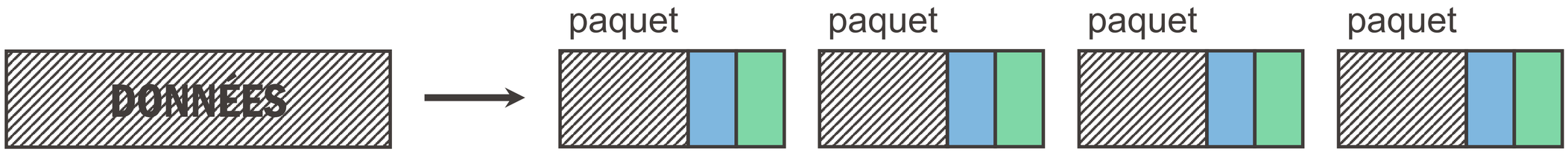
Avec l'avènement des communications entre ordinateurs (**beaucoup plus** de données à échanger), ce système ne pouvait plus tenir…

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 *deux informations* :

- Sa **destination**
- Son **identifiant**
  (pour que le destinataire puisse remettre les paquets dans l'ordre à l'arrivée)

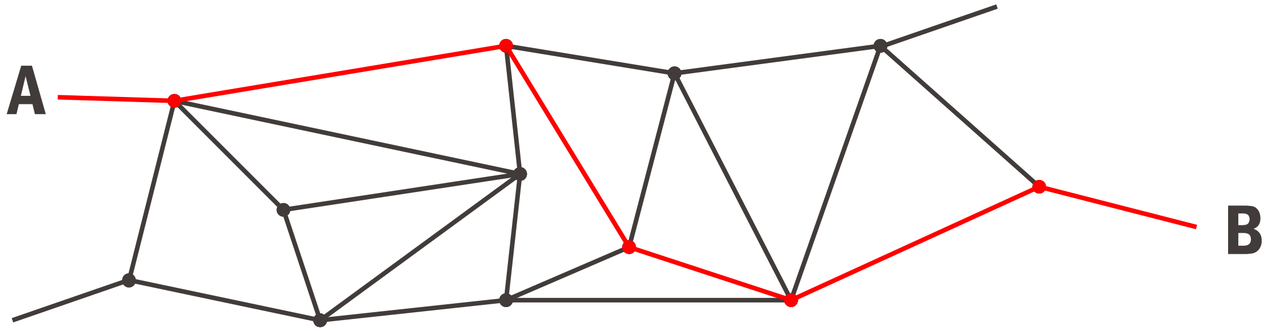
Deux protocoles

Nous allons maintenant étudier plus attentivement :

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

Protocole TCP

2. Et plus en détail, la question de l'acheminement du paquet à travers le réseau par un algorithme de **routage** :



Protocole IP

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 ?

- Pour éviter la **détérioration** du paquet, nous avons vu précédemment comment protéger celui-ci au moyen d'algorithmes de **correction d'erreurs** (bits de parité).

- Si par contre 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

     $$W \longrightarrow W + a \qquad \text{avec } a > 0 \text{ fixé}$$

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

     $$W \longrightarrow b \cdot W \qquad \text{avec } 0 < b < 1 \text{ fixé également}$$

(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

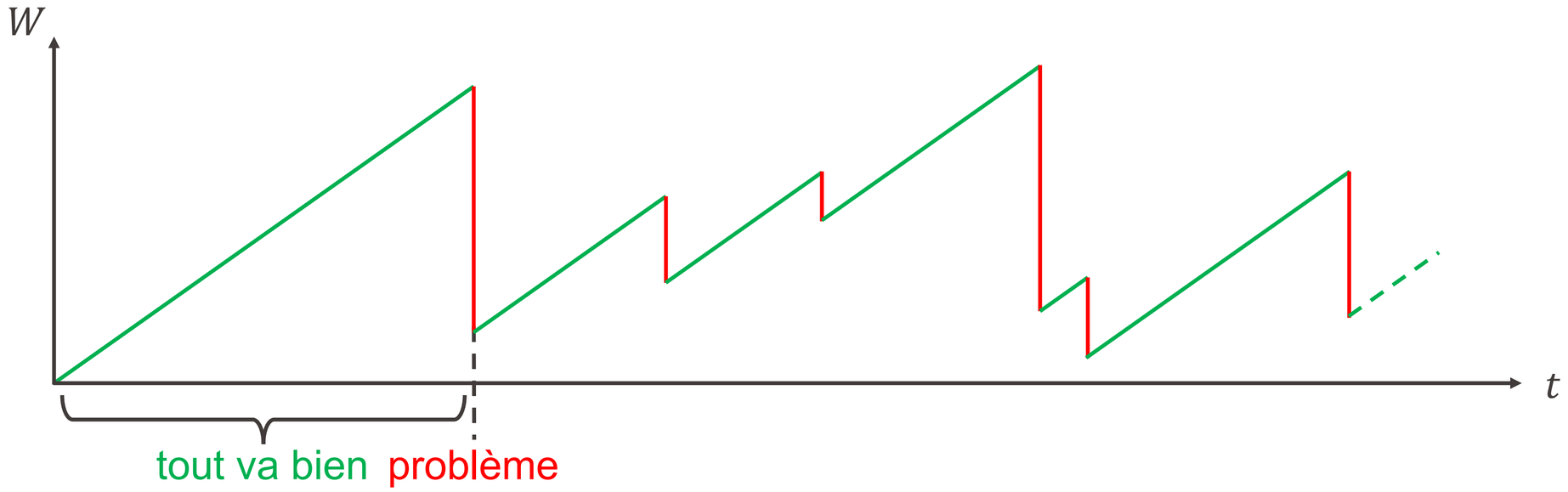
(pour utiliser au mieux le réseau)

- Dès qu'un problème survient :

On **diminue fortement** le nombre de transmissions

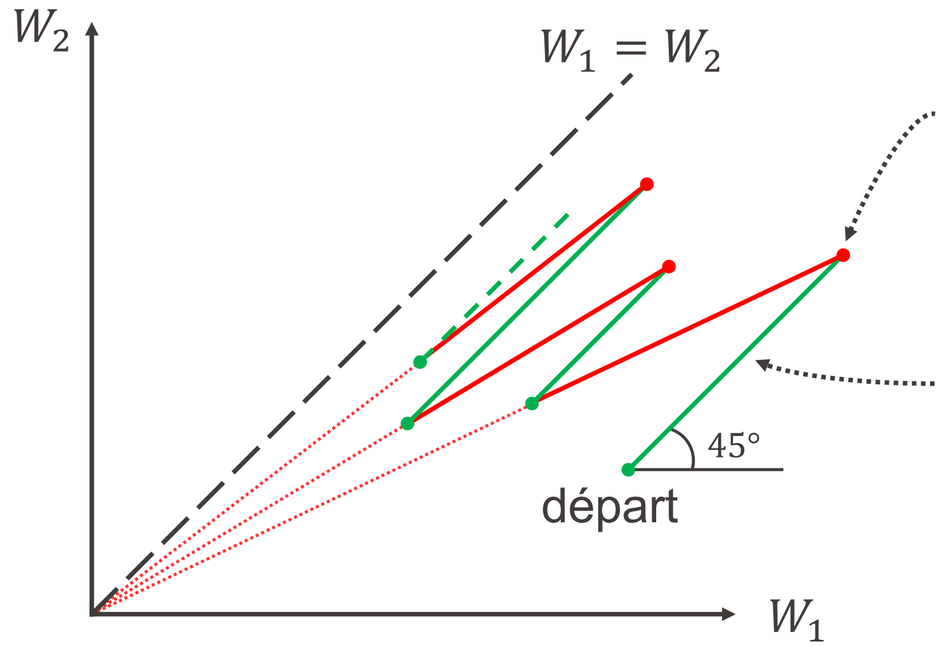
(pour laisser sa place aux autres)

→ 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 :

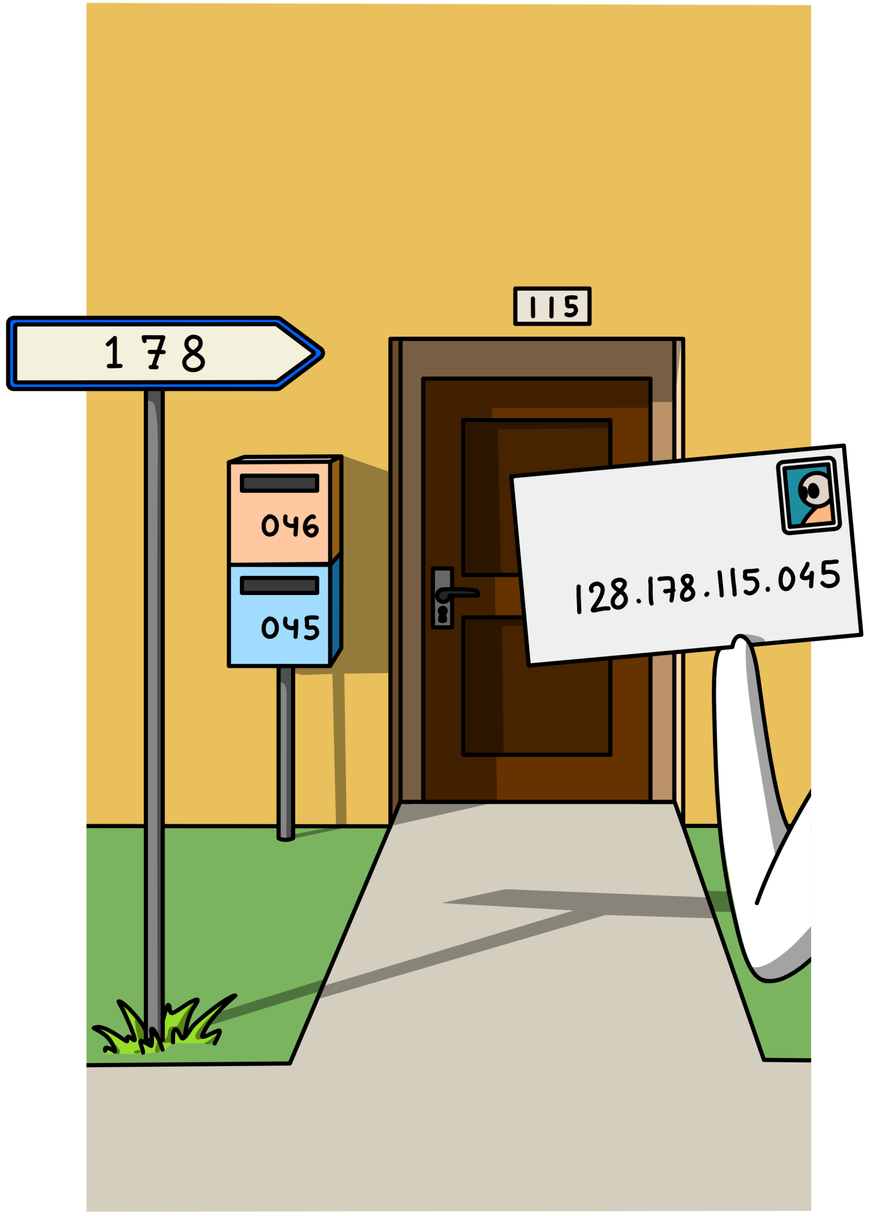


Un problème survient : A_1 et A_2 **divisent** W_1 et W_2 par la même valeur

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)



Information, Calcul et Communication

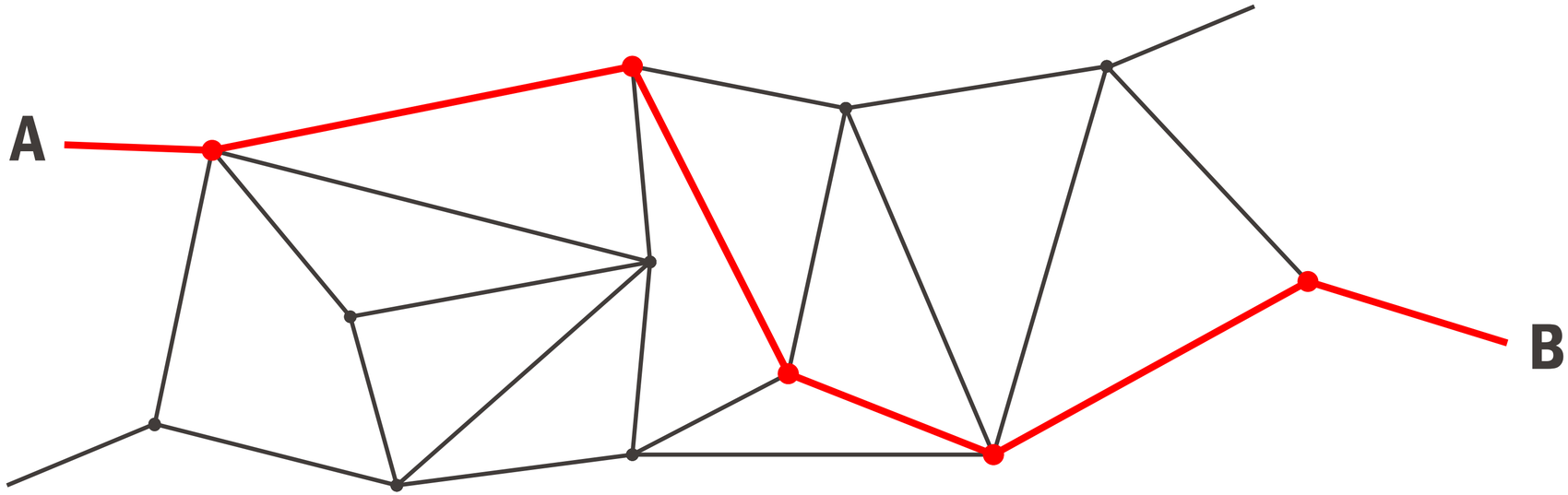
Réseaux : Protocole IP

École polytechnique fédérale de Lausanne

Olivier Lévêque

Protocole IP *(internet protocol)*

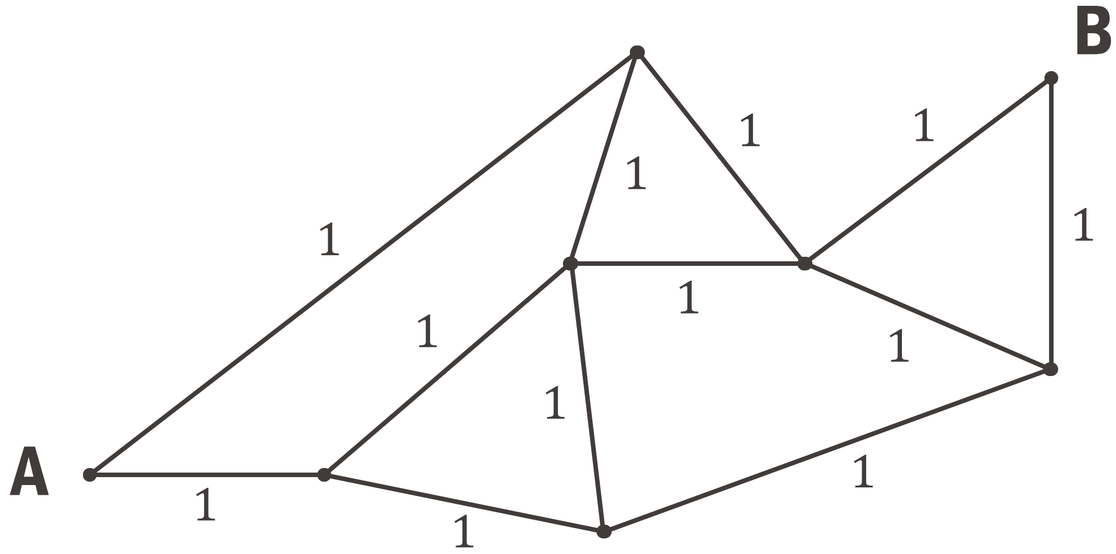
- Ce protocole est celui utilisé pour acheminer un paquet d'un nœud A vers un nœud B à travers le réseau :



- 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 :



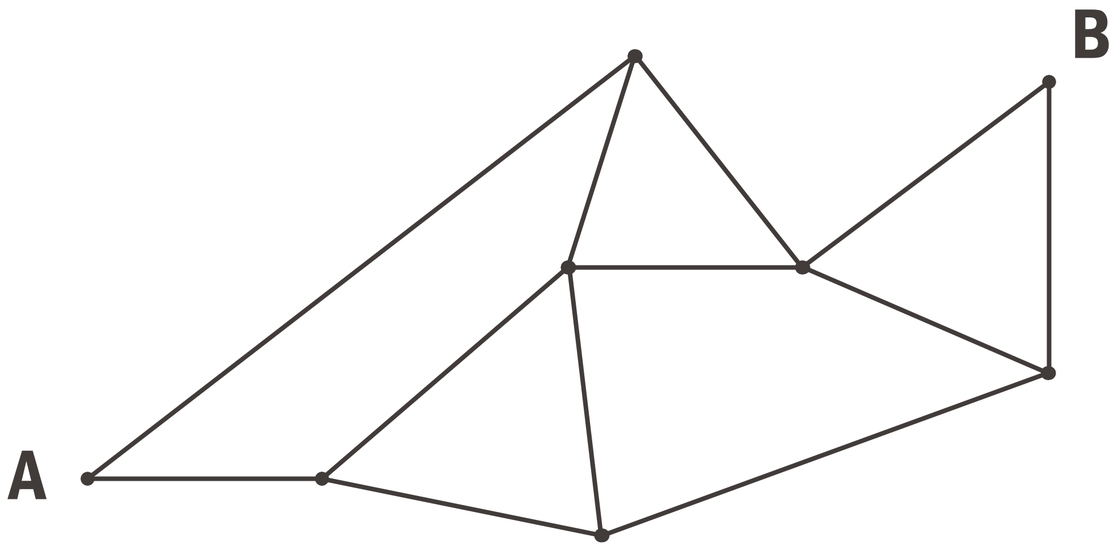
- Remarque importante :

Comme ce graphe est censé représenter un réseau de machines communiquant entre elles, nous supposons ici que la distance entre deux nœuds directement reliés entre eux est toujours la même (disons 1).

Ceci est justifié par le fait que le temps de transmission entre deux machines est négligeable par rapport au temps de relais à l'intérieur d'une machine donnée (qui est lui supposé à peu près identique pour chaque machine).

Algorithme BFS

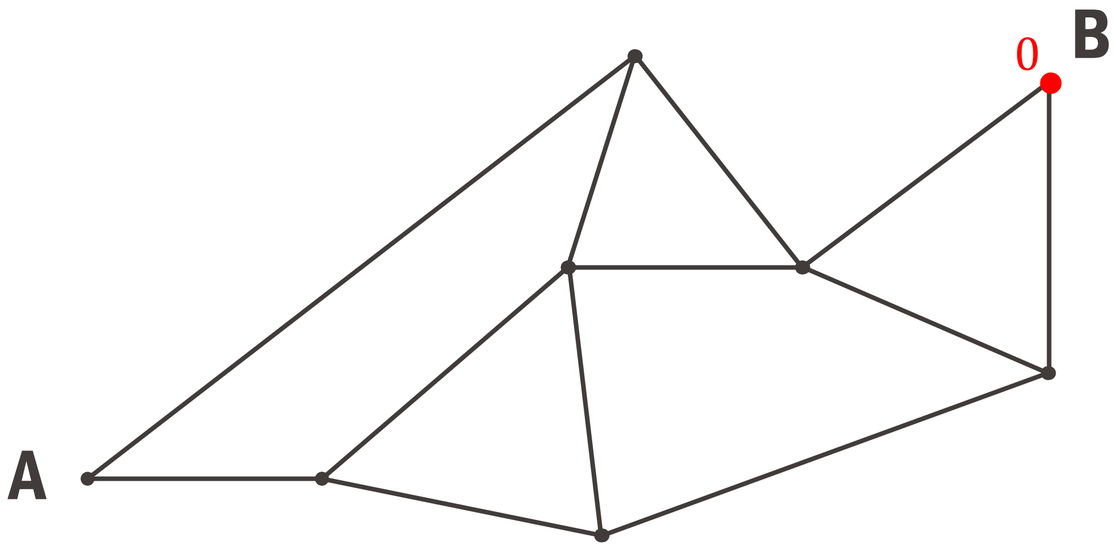
Pour trouver le chemin le plus court de A à B, l'algorithme BFS propose de chercher **tous les chemins les plus courts** de n'importe quel nœud du graphe au nœud B.



Algorithme BFS

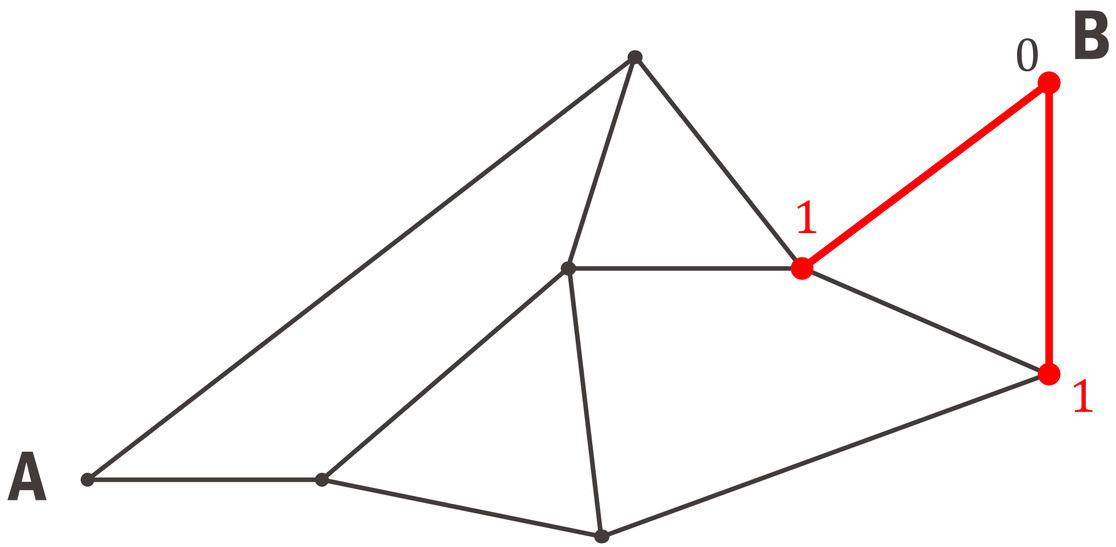
Pour trouver le chemin le plus court de A à B, l'algorithme BFS propose de chercher **tous les chemins les plus courts** de n'importe quel nœud du graphe au nœud B.

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



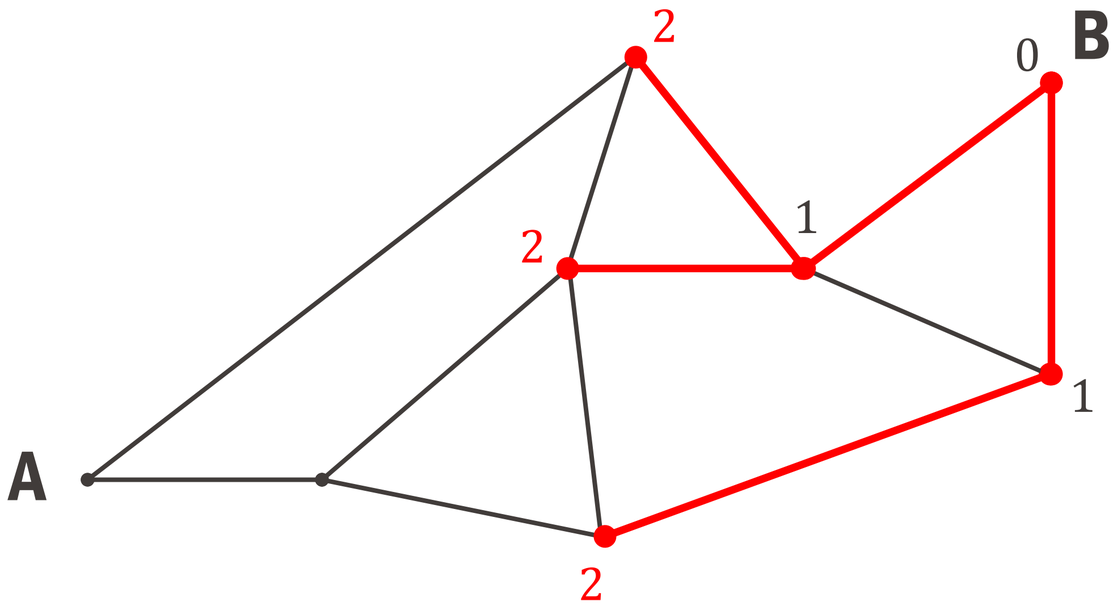
Algorithme BFS

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



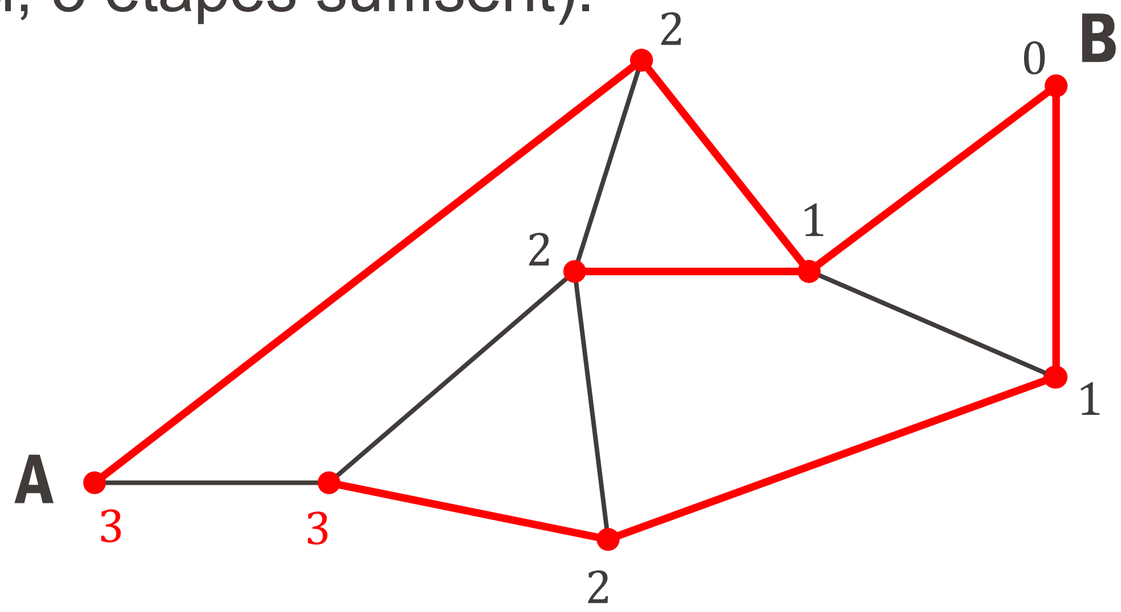
Algorithme BFS

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).

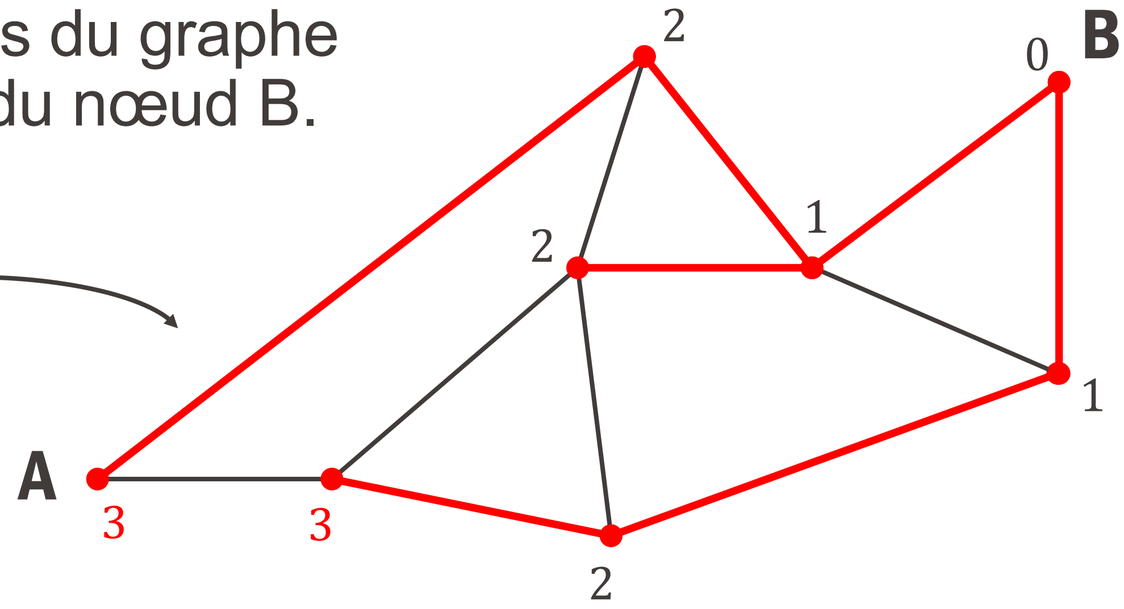


Algorithme BFS

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.

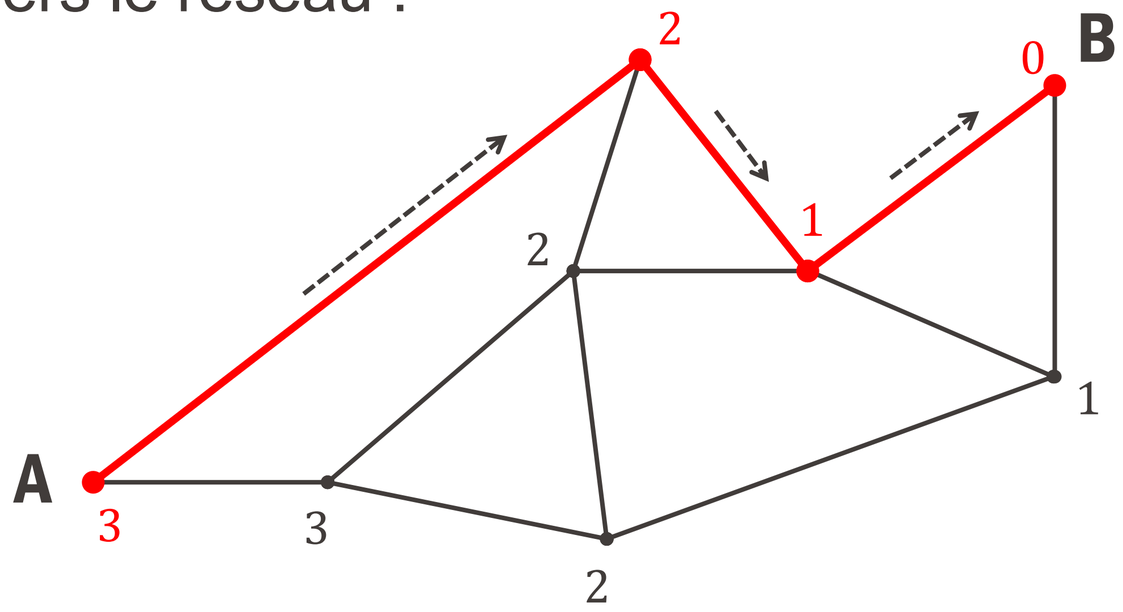
- À la fin de l'algorithme, tous les nœuds du graphe connaissent leur plus petite distance du nœud B.

arbre couvrant minimal !



Routage

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

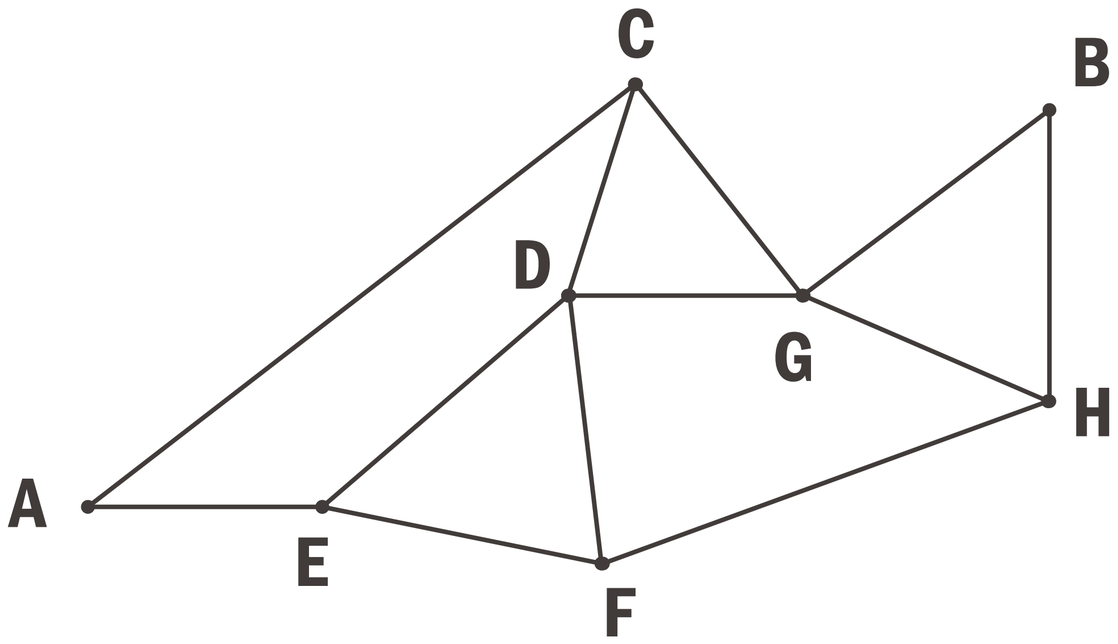


Supposons que A désire envoyer un paquet à B :

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 $\in \{0 \dots 255\}$)

$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
8 bits	8 bits	8 bits	8 bits	$\rightarrow 32$ bits au total
réseau principal	sous-réseau	…	…	$\rightarrow 2^{32}$ adresses différentes
				~ 4 milliards

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