# Algorithmes Distribués Probabilistes

Akka Zemmari
zemmari@labri.fr

04 novembre 2013

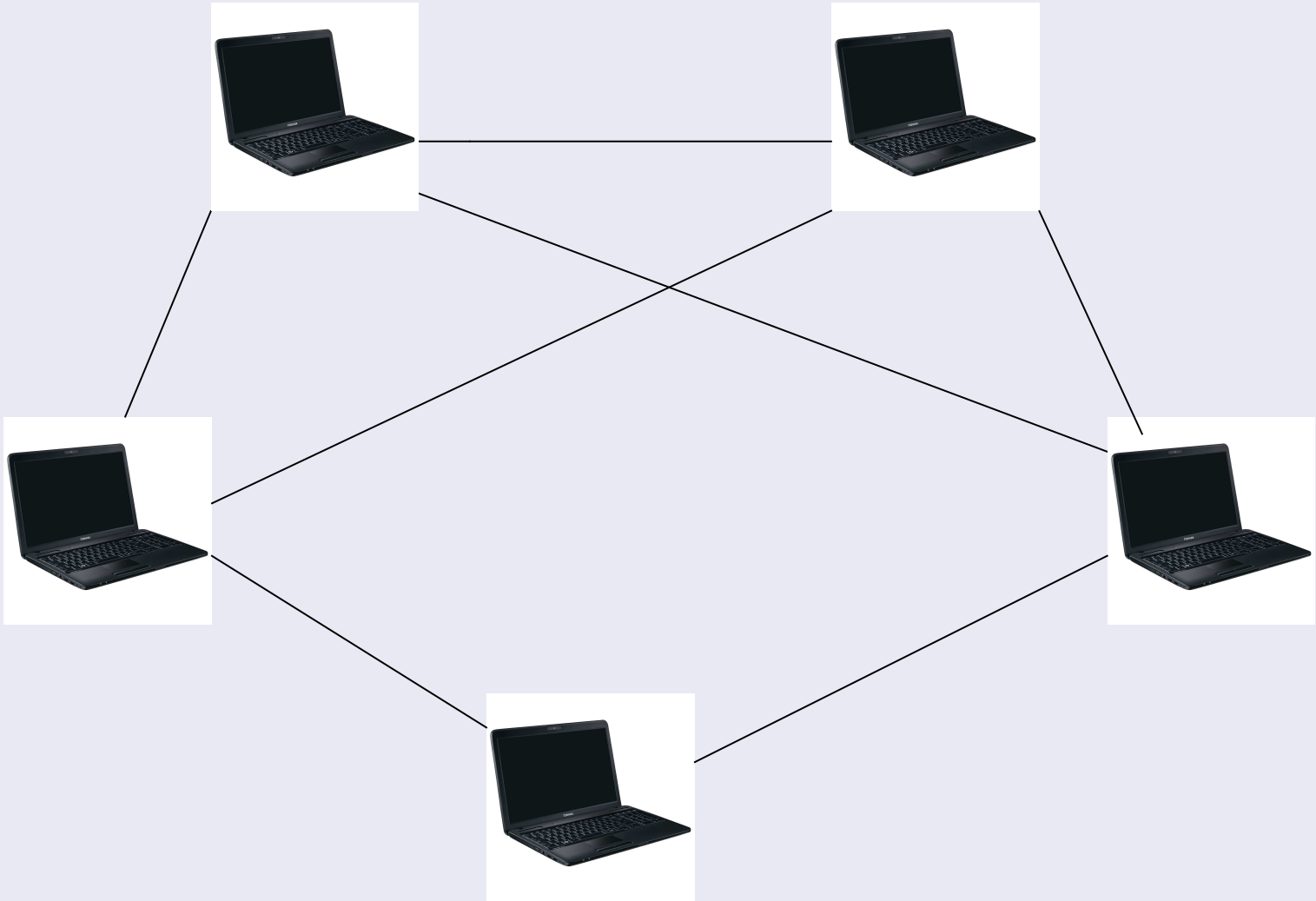
# Pourquoi utiliser les algorithmes distribués probabilistes

**Definition**

Un systme distribué est *anonyme* si les noeuds n'ont pas d'identifiants uniques.

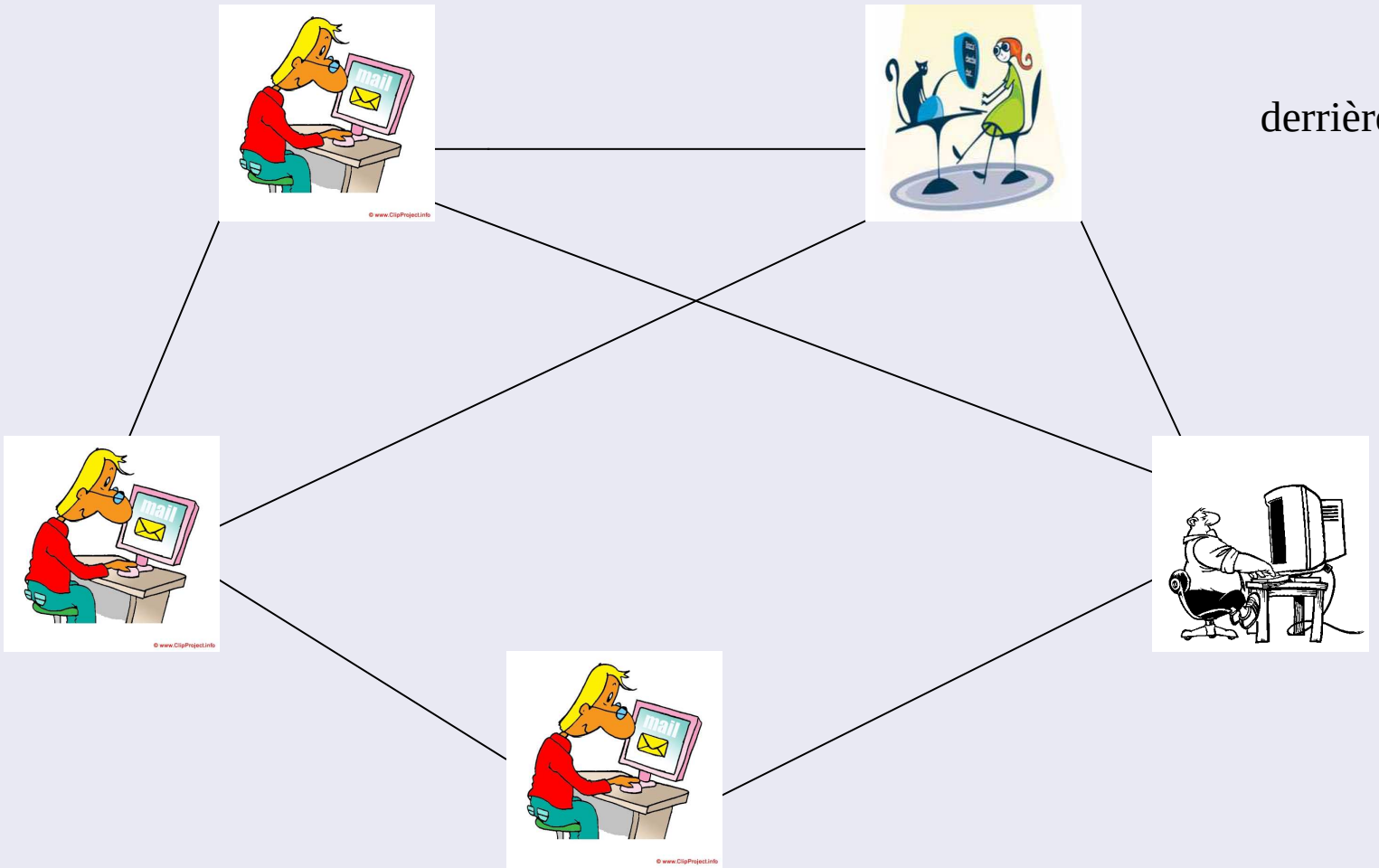
# Pourquoi utiliser les algorithmes distribués probabilistes

## Et pourquoi l'anonymat ?



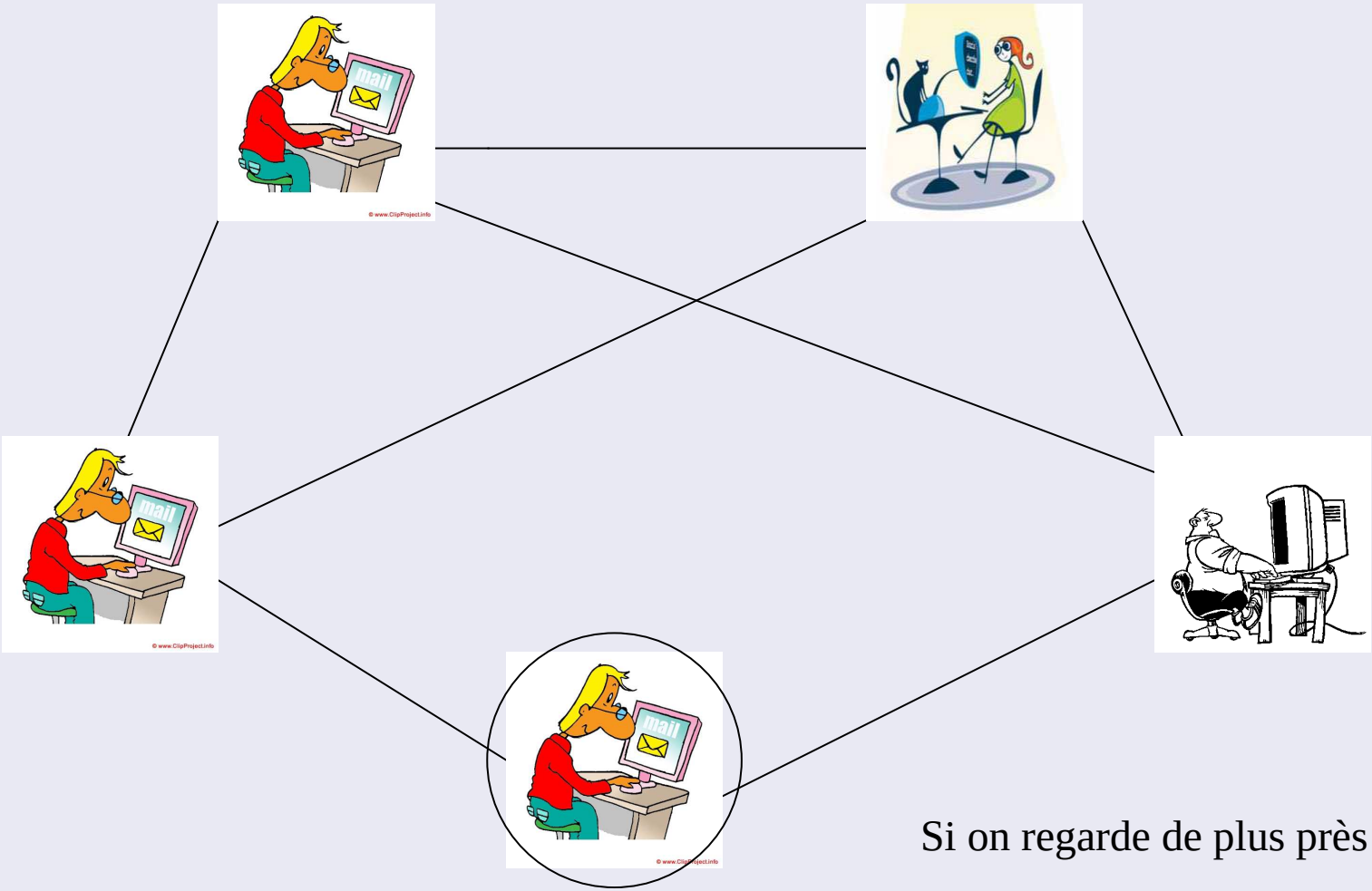
# Pourquoi utiliser les algorithmes distribués probabilistes

## Et pourquoi l'anonymat ?




derrière les machines ...

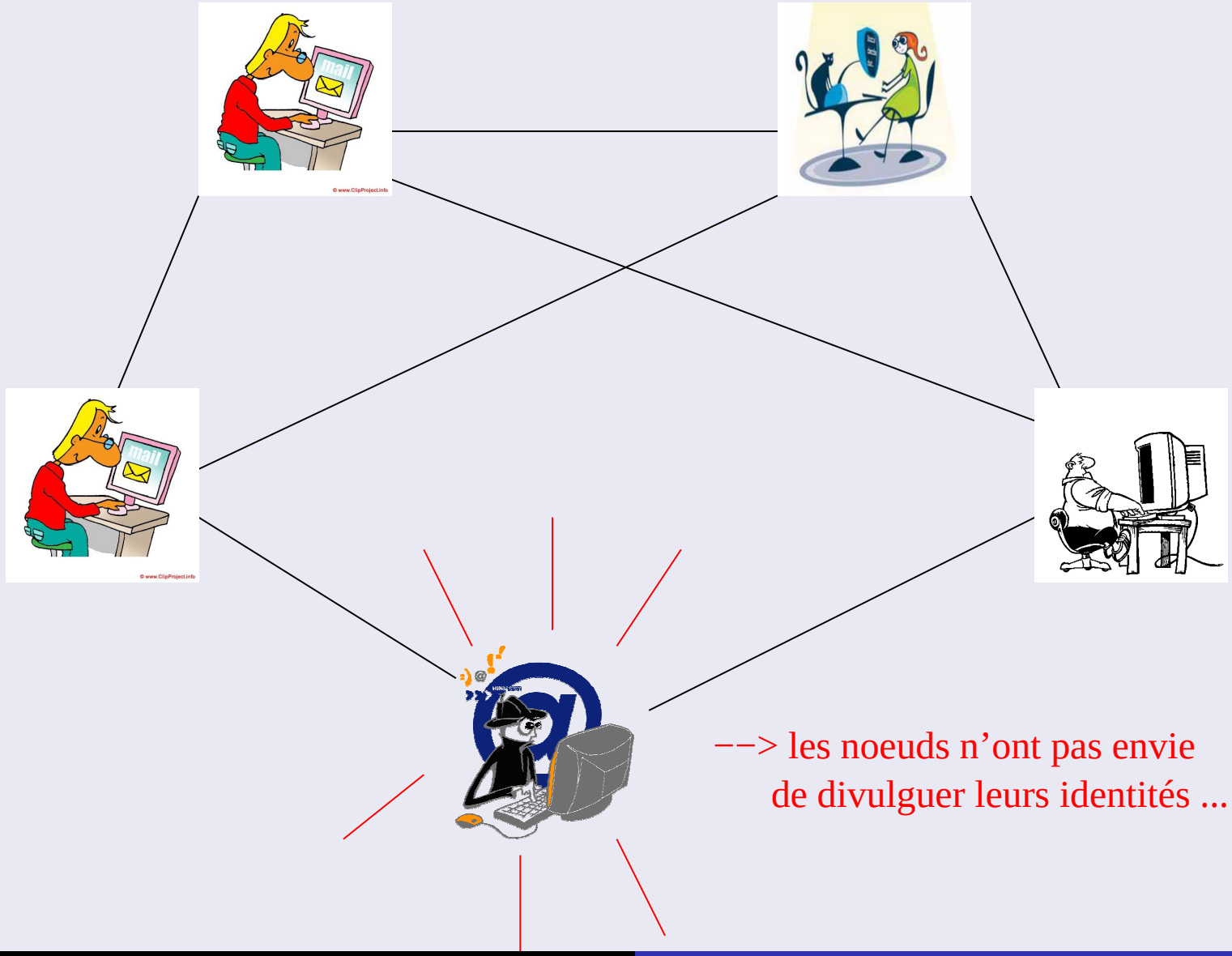
## Et pourquoi l'anonymat ?




Si on regarde de plus près ...

# Pourquoi utiliser les algorithmes distribués probabilistes

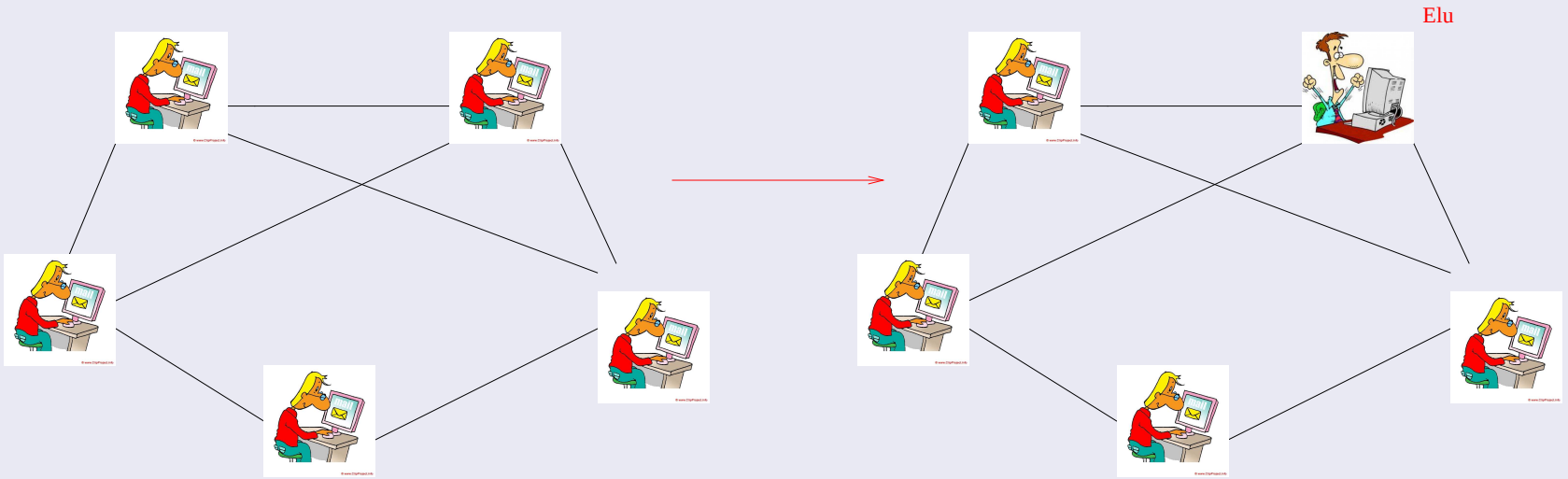
## Et pourquoi l'anonymat ?




−−> les noeuds n'ont pas envie
de divulguer leurs identités ...

# Pourquoi utiliser les algorithmes distribués probabilistes

## Un exemple: Elir dans un réseau anonyme



# Pourquoi utiliser les algorithmes distribués probabilistes

## Un exemple : Elir dans un réseau anonyme

Plus fomellement:
(Leader Election Problem [Tel, Lynch, ...]) Each node eventually decides whether it is a leader or not, subject to the constraint that there is exactly one leader.

# Pourquoi utiliser les algorithmes distribués probabilistes

## Pourquoi élir ...

- choisir un noeud pour centraliser des informations,
- choisir un noeud pour redémarrer le système,
- gérer les accès concurents à des ressources critiques,
- ...

Le problème est "facile" à résoudre si le réseau n'est pas symétrique ...

# Pourquoi utiliser les algorithmes distribués probabilistes

**Théorème [Angluin'81]**

Il n'existe pas d'algorithme déterministe permettant de réaliser une élection dans un réseau anonyme ayant une topologie en anneau

Remarque : le même résultat peut être énnoncé pour tout réseau ayant une topologie symétrique (graphe complet, graphe bipartie complet, ...)

$\Rightarrow$ Utliser des algorithmes probabilistes ...

# Définitions

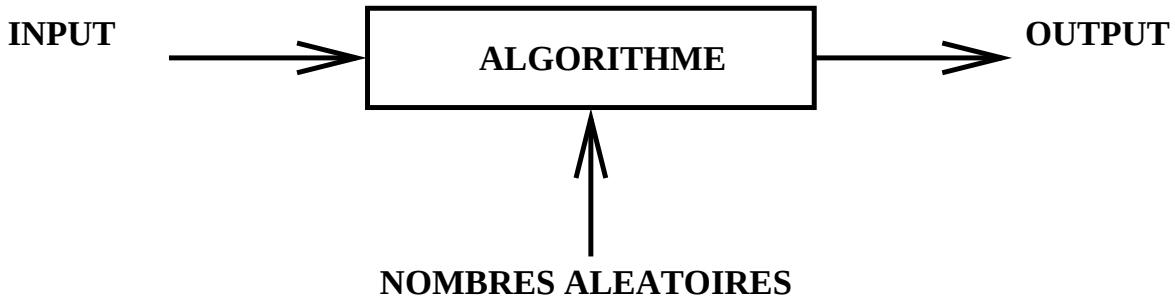
- Un *algorithme probabiliste* est un algorithme qui utilise des tirages de type pile ou face ou des générateurs de nombres aléatoires. A la différence des algorithmes *déterministes*, le *hasard* joue un rôle fondamental dans l'exécution de tels algorithmes.

# Définitions

- Un *algorithme probabiliste* est un algorithme qui utilise des tirages de type pile ou face ou des générateurs de nombres aléatoires. A la différence des algorithmes *déterministes*, le *hasard* joue un rôle fondamental dans l'exécution de tels algorithmes.
- Motwani & Raghavan:

# Définitions

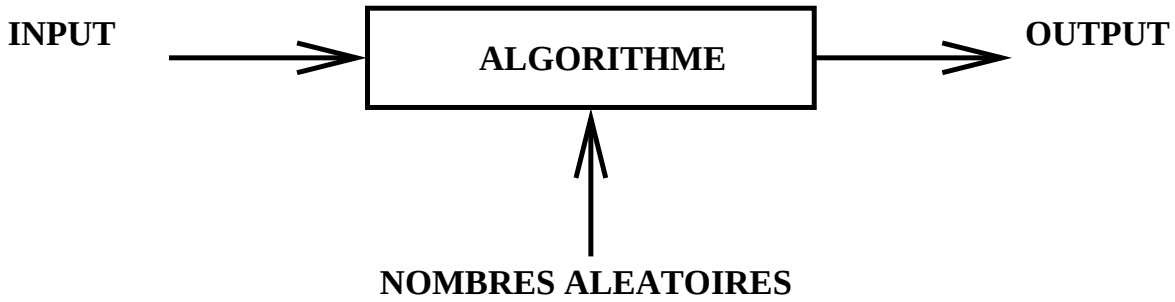
- Un *algorithme probabiliste* est un algorithme qui utilise des tirages de type pile ou face ou des générateurs de nombres aléatoires. A la différence des algorithmes *déterministes*, le *hasard* joue un rôle fondamental dans l'exécution de tels algorithmes.
- Motwani & Raghavan:




En plus de l'entrée, l'algorithme utilise une source de nombres aléatoires et fait des choix aléatoires lors de son exécution.
$\Rightarrow$ le comportement de l'algorithme peut varier même avec une même entrée.

# Définitions

- Un *algorithme probabiliste* est un algorithme qui utilise des tirages de type pile ou face ou des générateurs de nombres aléatoires. A la différence des algorithmes *déterministes*, le *hasard* joue un rôle fondamental dans l'exécution de tels algorithmes.
- Motwani & Raghavan:




Objectif : concevoir des algorithmes probabilistes et analyser leur comportement.

# Définitions

- Un *algorithme probabiliste distribué* $\mathcal{A}_p$ est une collection d'algorithmes locaux $\mathcal{A}_p = (\mathcal{A}_i)_{1 \leq i \leq n}$ telle que chaque algorithme $\mathcal{A}_i$ est un algorithme (ou processus) probabiliste.

# Notations

- $\mathcal{S}$ : un système distribué synchrone à $n \geq 1$ noeuds modélisé par un graphe $G = (V, E)$ :
  - $V$ : ensemble de sommets représentant les noeuds du système
  - $E$ : ensemble d'arêtes représentant les liens de communication
  - $\mathcal{T}$ : une tâche à réaliser sur $\mathcal{S}$
  - $\mathcal{A}$ : un algorithme probabiliste permettant de réaliser la tâche $\mathcal{T}$.

# Hypothèses

- Le système (distribué) est synchrone $\Rightarrow$ les noeuds du système ont accès à une horloge globale (hypothèse trop forte ...)
- Sans perte de généralité, on suppose que l'algorithme $\mathcal{A}$ démarre à l'instant $t = 0$

# Analyse des algorithmes distribués probabilistes

On s'interesse (la plus part des cas) aux mesures :

- Temps d'exécution de l'algorithme.
- Nombre de messages envoyés.
- Taille des messages envoyés.

# Complexité en temps

## Définition

On définit la variable aléatoire (v.a.) $T$ comme le nombre d'unités de temps qui s'écoulent entre l'instant $t = 0$ et la plus petite valeur de $t$ telle que tous les noeuds du système ont fini d'exécuter l'algorithme $\mathcal{A}$.

$$T = \min\{t \geq 0 \mid \text{ tous les noeuds de } \mathcal{S} \text{ ont terminé l'exécution de l'algorithme } \mathcal{A}\}$$

Dans l'analyse de l'algorithme $\mathcal{A}$, on s'interesse :

- à l'espérance de $T$ : complexité en moyenne de $\mathcal{A}$.
- à la distribution de $T$ (plus difficile ...)
- Si $\mathbb{E}(T) = O(f(n))$, on s'intéresse également à la probabilité que $T = O(f(n))$.

# Complexité avec forte probabilité (w.h.p)

**Définition**

On dit que l'algorithme $\mathcal{A}$ a une complexité en temps $O(t)$ avec forte probabilité si il términe en $O(t)$ avec probabilité $1 - \frac{1}{n^c}$ pour tout $c > 1$.

# Catégorie des algorithmes probabilistes

Il existe deux (grandes catégories) d'algorithmes probabilistes :

- Algorithmes Monte Carlo : Un algorithme probabiliste est dit de type Monte Carlo s'il termine toujours et que le résultat qu'il calcule est correct avec probabilité $\geq 1/3$
- Algorithmes Las Vegas : Un algorithme probabiliste est dit de type Las Vegas si le résultat qu'il calcule est correct avec probabilité 1 mais que son temps d'exécution est une variable aléatoire dont l'espérance est au plus polynomiale.

# Election dans les réseaux anonymes

- élection dans un graphe complet
- L'algorithme est paramétré par un entier $N > 0$
- Nous verrons que selon la valeur de $N$, l'algorithme peut ne pas terminer, ou avoir un temps logarithmique avec forte probabilité ou encore être de type Monte Carlo.
- Nous verrons également comment transformer cet algorithme en un algorithme de type Las Vegas

# Election dans les réseaux anonymes

## $ElectionProba_1()$

1. $v$ tire uniformément un élement $r(v)$ de l'ensemble $\{1, 2, \cdots, N\}$;
2. $v$ envoie $r(v)$ à tous ses voisins;
3. $v$ reçoit $r(u)$ de chaque voisin $u$;
4. **Si**$(r(v) > r(u)$ pour tout sommet $u$ de $V)$ **alors** $etat_v := Elu$;
5. **Sinon** $etat_v := NonElu$;
6. **Fin Si**
7. $v$ envoie $etat_v$ à tous ses voisins;
8. $v$ reçoit $etat_u$ de chaque voisin $u$;

## Analyse de l'algorithme :

- Probabilité pour un sommet d'être élu :
  Proposition : Soit $v$ un sommet quelconque de $K_n$. Soit $\mathcal{E}(v)$ l'événement $v$ est élu. Alors:

$$\mathbb{P}(\mathcal{E}(v)) = \frac{1}{N} \sum_{i=2}^{N} \left( \frac{i-1}{N} \right)^{n-1}. \quad (1)$$

- Probabilité d'aboutir à une élection :
  Soit $\mathcal{E}$ l'évenement *Il y a un sommet élu dans le graphe*. Il est alors clair que :

$$\mathcal{E} = \{\exists v \in V \text{ tel que } \mathcal{E}(v)\}.$$

Par ailleurs, pour tous $u, v \in V$, $\mathcal{E}(u) \cap \mathcal{E}(v) = \emptyset$. Il en résulte que :

$$\begin{aligned} \mathbb{P}(\mathcal{E}) &= \mathbb{P}(\{\exists v \in V \text{ tel que } \mathcal{E}(v)\}) \\ &= \sum_{v \in V} \mathbb{P}(\mathcal{E}(v)) \\ &= \frac{n}{N} \sum_{i=2}^{N} \left(\frac{i-1}{N}\right)^{n-1}. \end{aligned}$$

## Analyse de l'algorithme (selon les valeurs de $N$):

1. Probabilité pour un sommet d'être élu
   - Si $N = 1$, l'algorithme échoue
   - Si $N = 2$,
   $$\mathbb{P}(\mathcal{E}(v)) = \frac{1}{2^n}.$$
   - Si $N = n$,
   $$\begin{aligned} \mathbb{P}(\mathcal{E}(v)) &\sim \frac{1}{n}\left((1-\frac{1}{N})^n - \frac{1}{N^n}\right) \\ &\to \frac{1}{n}, \quad \text{quand } N \to \infty. \end{aligned}$$

## Analyse de l'algorithme (selon les valeurs de $N$):

2. Probabilité d'une élection

- Si $N = n$,

$$\begin{aligned} \mathbb{P}(\mathcal{E}) &\sim \frac{1}{e} - \frac{1}{n^n}, \quad \text{quand } n \to \infty. \\ &\sim \frac{1}{e} \geq 1/3. \end{aligned}$$

  $\Rightarrow$ algorithme de type Monte Carlo !!

- Si $N = n^2$, alors $ElectionProba_1()$ réalise une élection avec forte probabilité.
- Si $N = \infty$, alors $ElectionProba_1()$ réalise une élection avec probabilité 1.

# Election dans les réseaux anonymes

## Un deuxième algorithme :

- Repeter
  $ElectionProba_1()$;
- Jusqu'à $etat_v = Elu$ ou $\exists u$ tel que $etat_u = Elu$

## Analyse de l'algorithme

Notation : $T$ la v.a. qui compte le nombre d'itérations

$$\mathcal{E}_i = \{\text{il y a une élection à la } i^{\text{ème}} \text{ itération, }\} \text{ pour tout } i \geq 1.$$

Il est alors clair que :

$$\text{Pour tout } i \geq 1,\ T = i \text{ si, et seulement si, } \mathcal{E}_i.$$

$\Rightarrow$ $T$ est une v.a. qui suit une loi géométrique de paramètre $p = \mathbb{P}(\mathcal{E})$

## Analyse de l'algorithme

Affirmation : en choisissant bien $N$, on peut obtenir un algorithme qui a une complexité en $O(\log n)$ et ce avec forte probabilité.
$\Rightarrow$ algorithme de type Las Vergas !!