SDR

# Sondes et échos

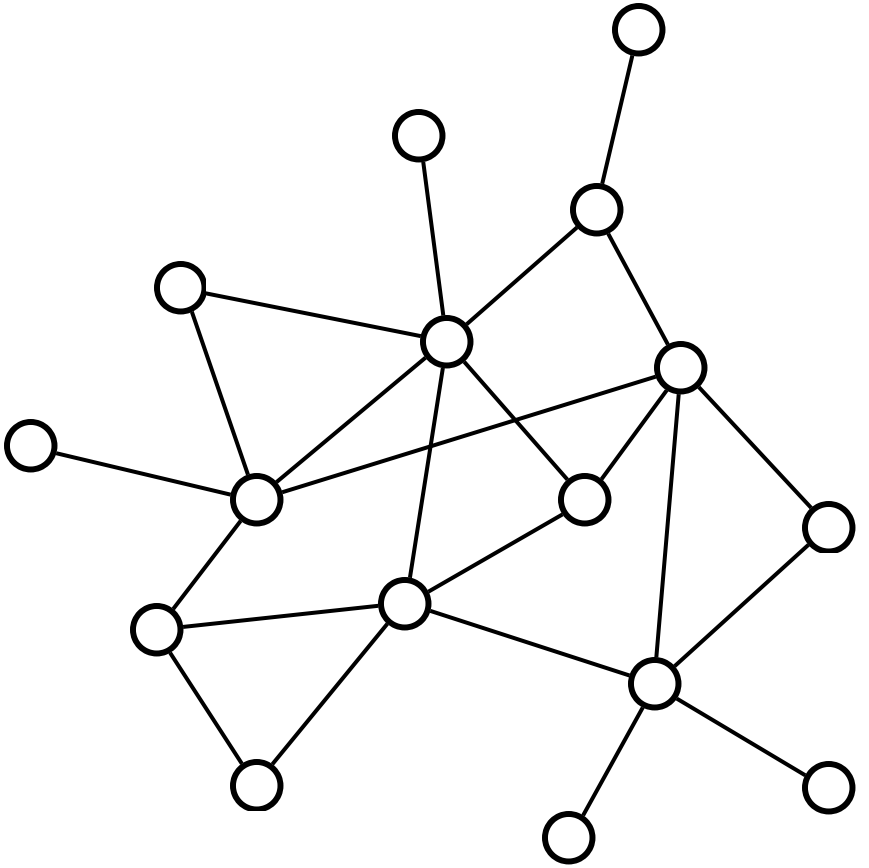
Olivier Lemer

# L'idée

# Programmation par Sondes et Échos

## L'idée

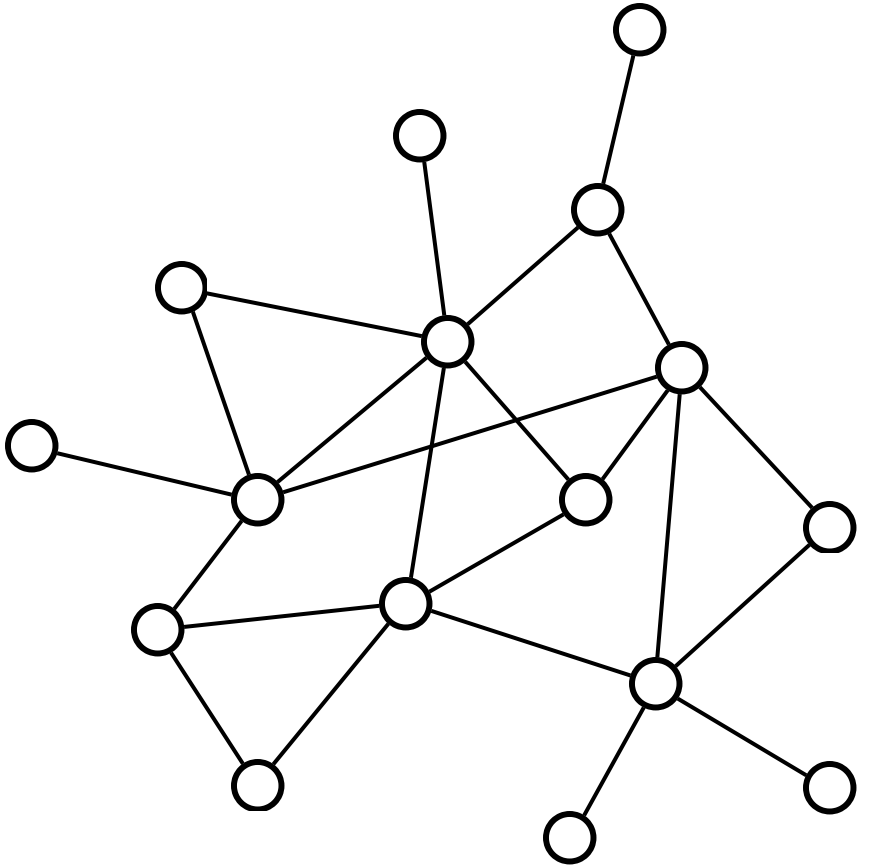
Permettre à un processus donné d'échanger avec tous les autres processus.



# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

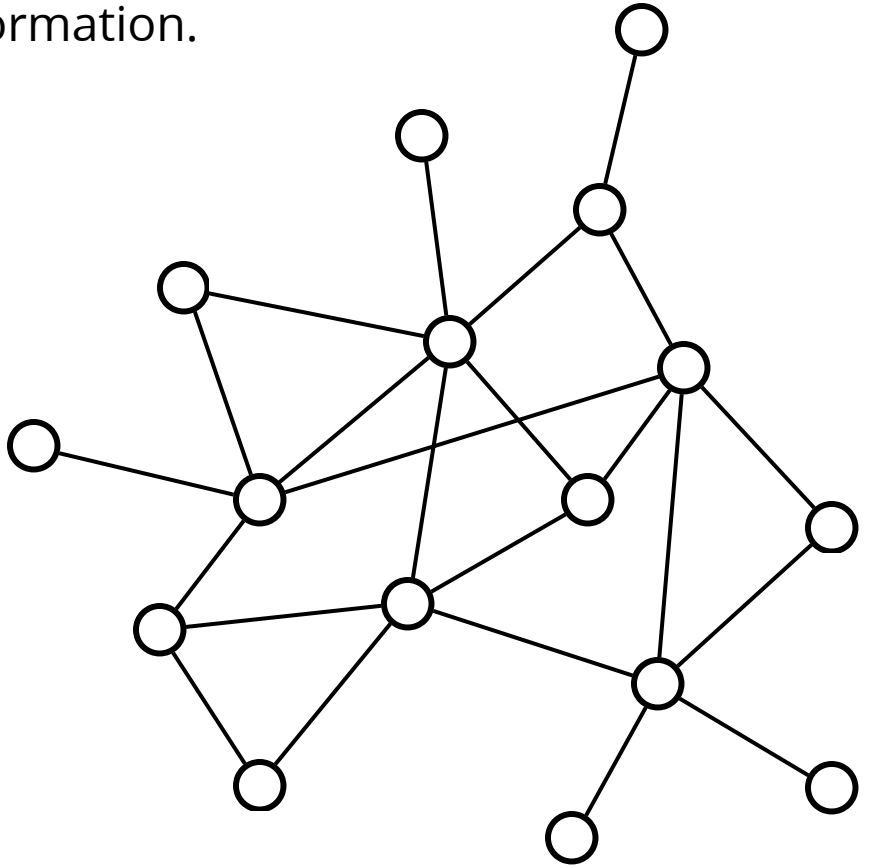


# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

Une phase de **diffusion** propage l'information.

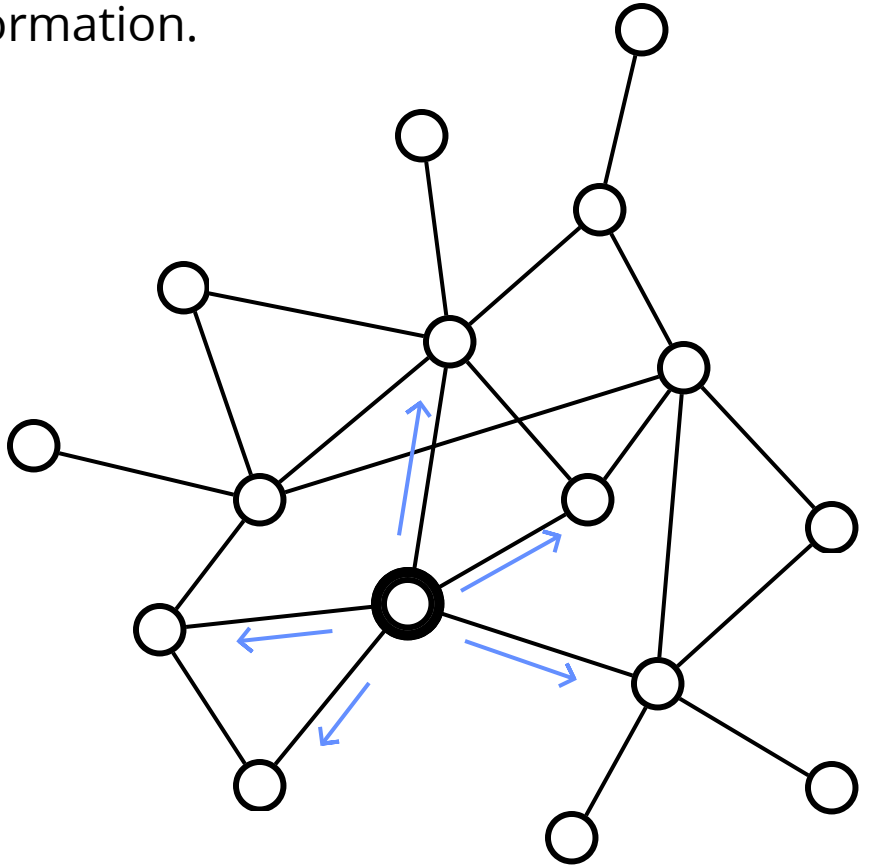


# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

Une phase de **diffusion** propage l'information.

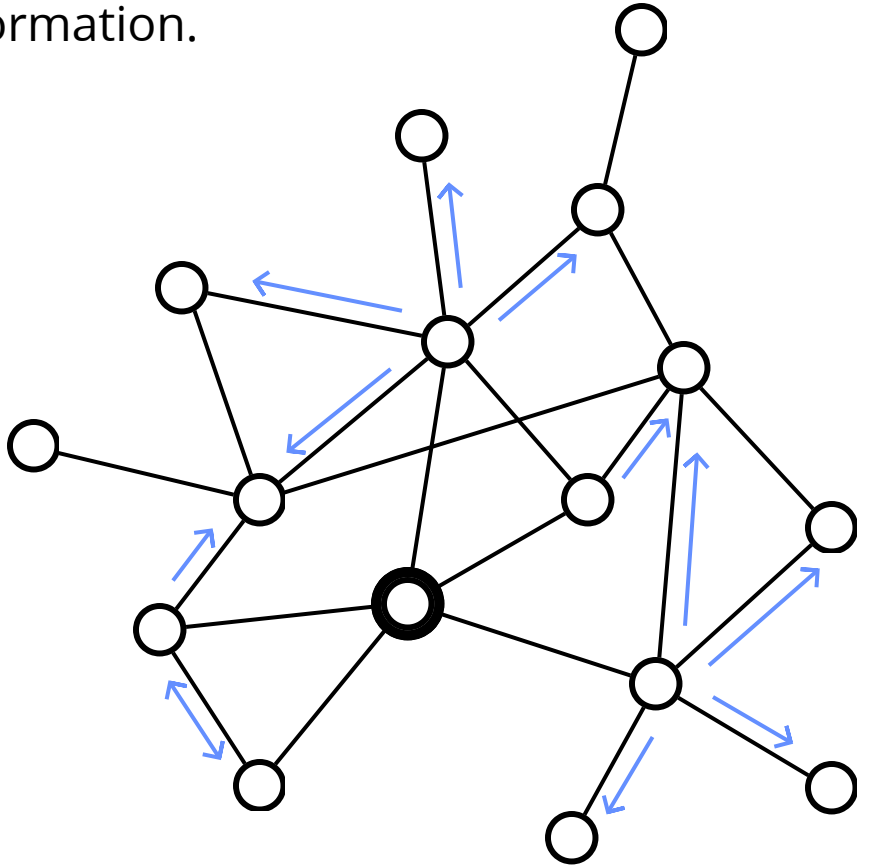


# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

Une phase de **diffusion** propage l'information.



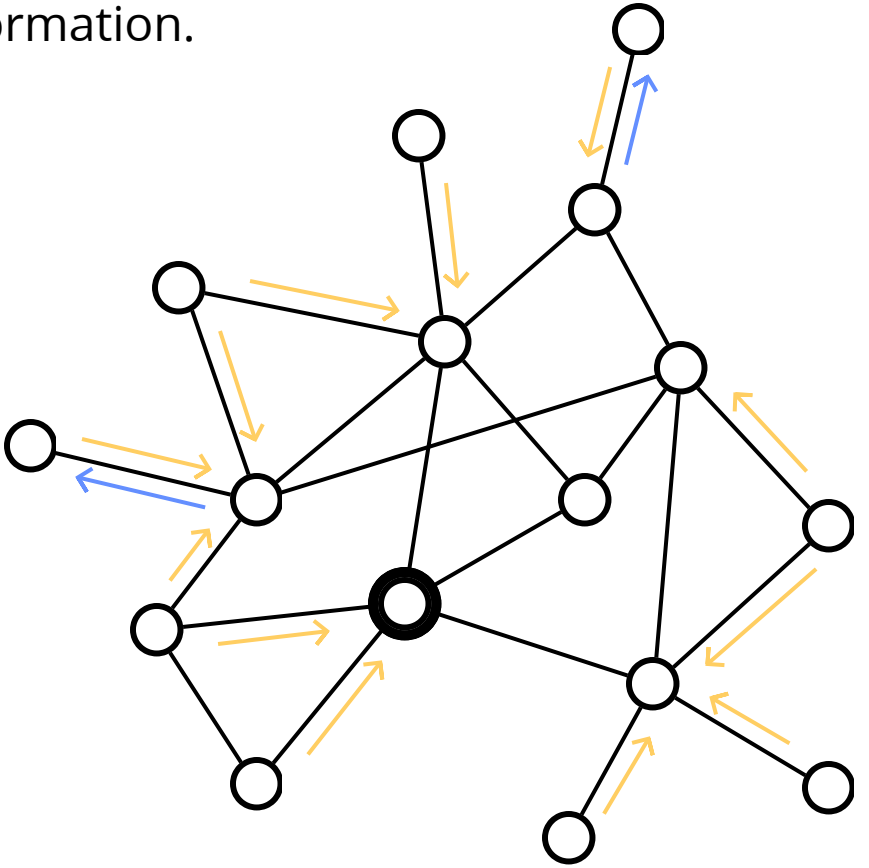
# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

Une phase de **diffusion** propage l'information.

Puis une phase de **contraction**
renvoie une information à la source.



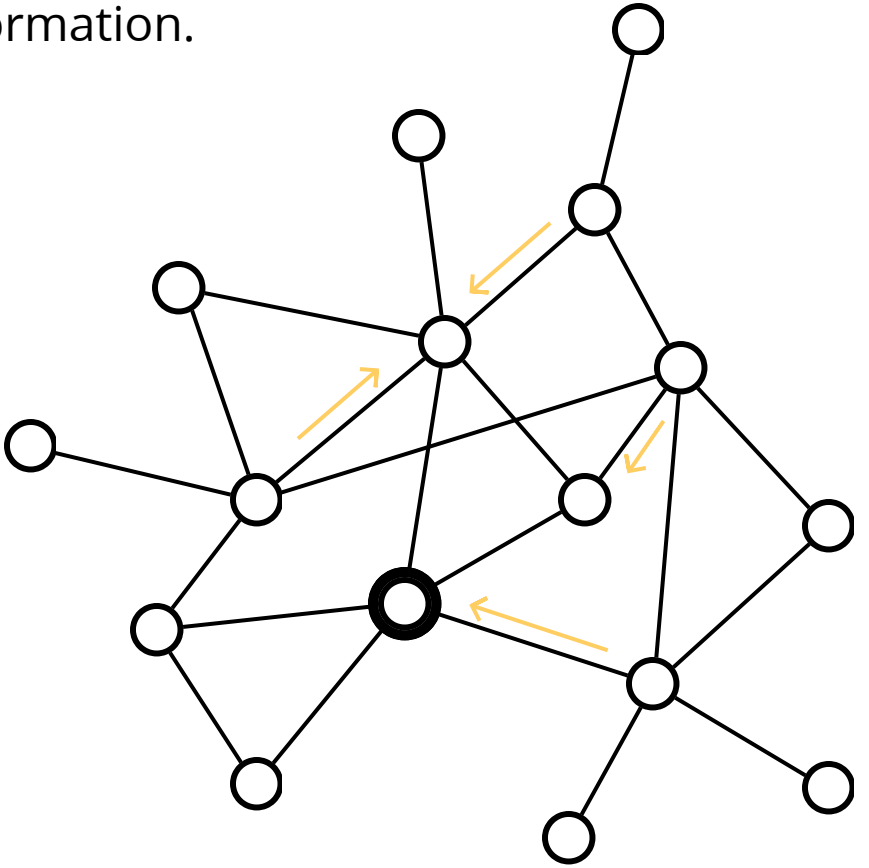
# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

Une phase de **diffusion** propage l'information.

Puis une phase de **contraction**
renvoie une information à la source.



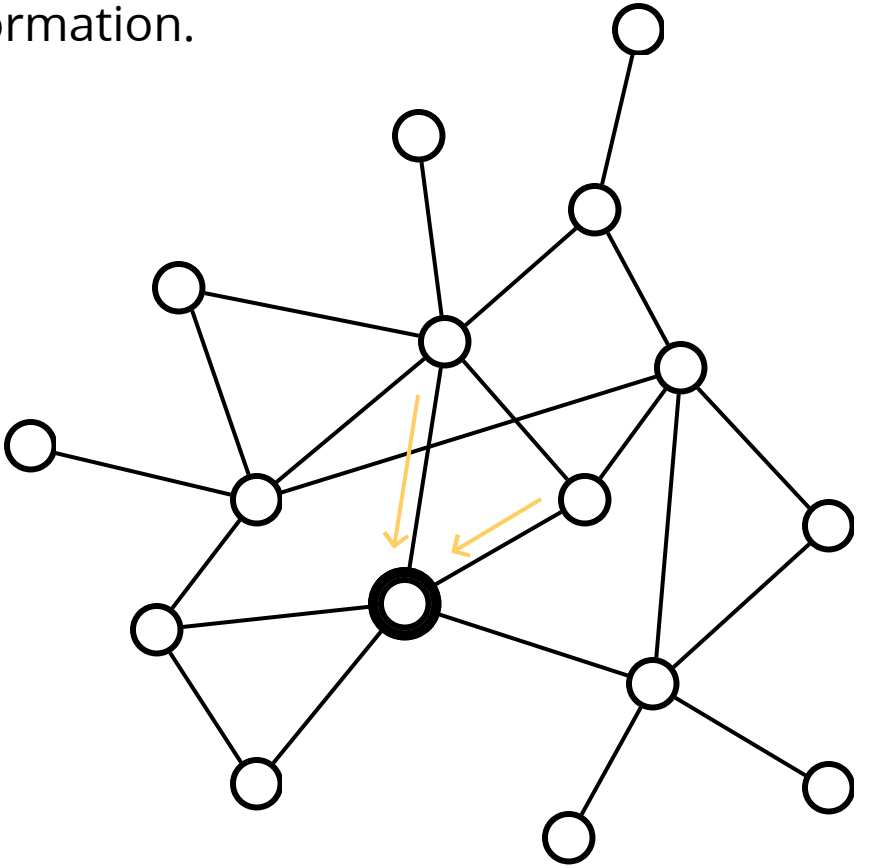
# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

Une phase de **diffusion** propage l'information.

Puis une phase de **contraction**
renvoie une information à la source.



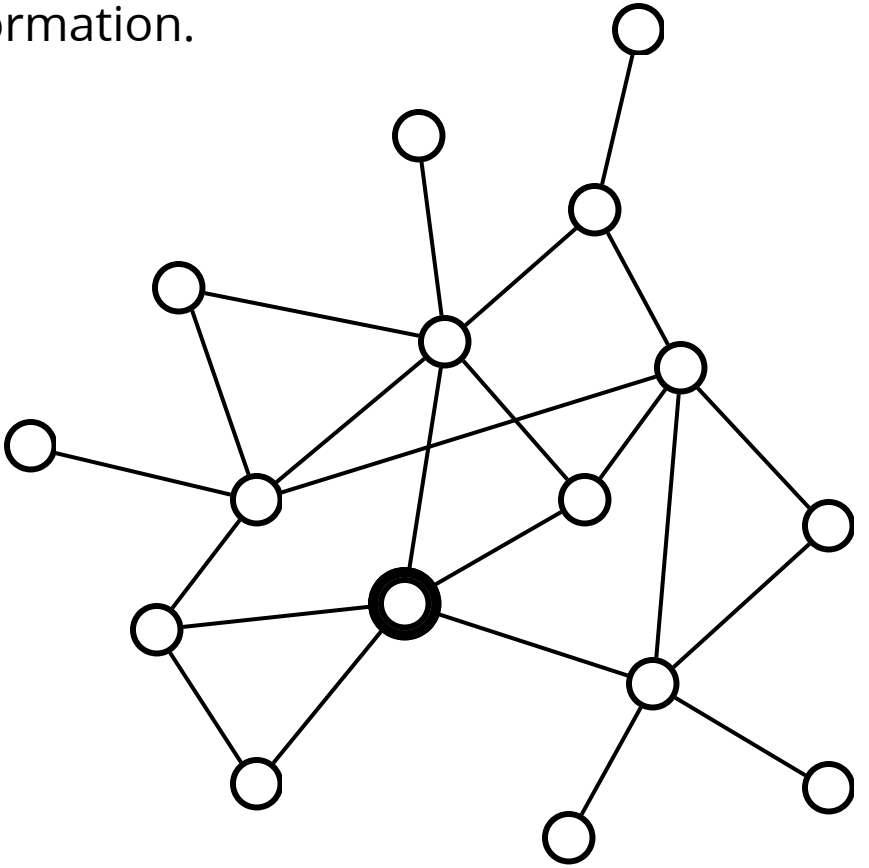
# Programmation par Sondes et Échos

## L'idée

Permettre à un processus donné d'échanger avec tous les autres processus.

Une phase de **diffusion** propage l'information.

Puis une phase de **contraction**
renvoie une information à la source.



# Définition formelle

et exemples

# Sondes et Échos

Un processus **source** déclenche une requête.

# Sondes et Échos

Un processus **source** déclenche une requête.

**Phase de diffusion**

Des **sondes** sont envoyées aux voisins.
*Transmission de requête.*

# Sondes et Échos

Un processus **source** déclenche une requête.

**Phase de diffusion**

Des **sondes** sont envoyées aux voisins.
*Transmission de requête.*

**Phase de contraction**

Des **échos** sont renvoyés dans l'autre sens.
*Transmission de réponse.*

# Sondes et Échos

Un processus **source** déclenche une requête.

**Phase de diffusion**

Des **sondes** sont envoyées aux voisins.
*Transmission de requête.*

**Phase de contraction**

Des **échos** sont renvoyés dans l'autre sens.
*Transmission de réponse.*

**Invariant**

Pour chaque processus, au total :

Nombre de sondes envoyées = Nombre d'échos reçus

# Exemples

**Agrégation d'information** *(source demande, noeuds répondent)*

Sondes : *"Agrégeons les données distribuées"*

Échos : *"Voici mes données et celles que j'ai récoltées"*

**Partage d'information** *(source informe, noeuds acquiescent)*

Sondes : *"Voici de nouveaux identifiants"*

Échos : *"Bien reçu"*

**Synchronisation de processus** *(source orchestre, noeuds calculent puis répondent)*

Sondes : *"Faites ce calcul"*

Échos : *"Voici le résultat"*

# Sémantique des messages

**Sonde**

*"Faites cette chose"*

Par exemple

*"Note cette info"*

*"Donne moi le résultat de cette fonction"*

*"Commence un nouveau round de calculs et dis-moi quand tu as fini"*

**Écho**

*"Mes enfants et moi avons fait cette chose"*

Par exemple

*"C'est fait"*

*"Voici le résultat"*

*"On a fini"*

# Sémantique des messages

**Sonde**

*"Faites cette chose"*

Par exemple

*"Note cette info"*

*"Donne moi le résultat de cette fonction"*

*"Commence un nouveau round de calculs et dis-moi quand tu as fini"*

**Écho**

*"Mes enfants et moi avons fait cette chose"*

Par exemple

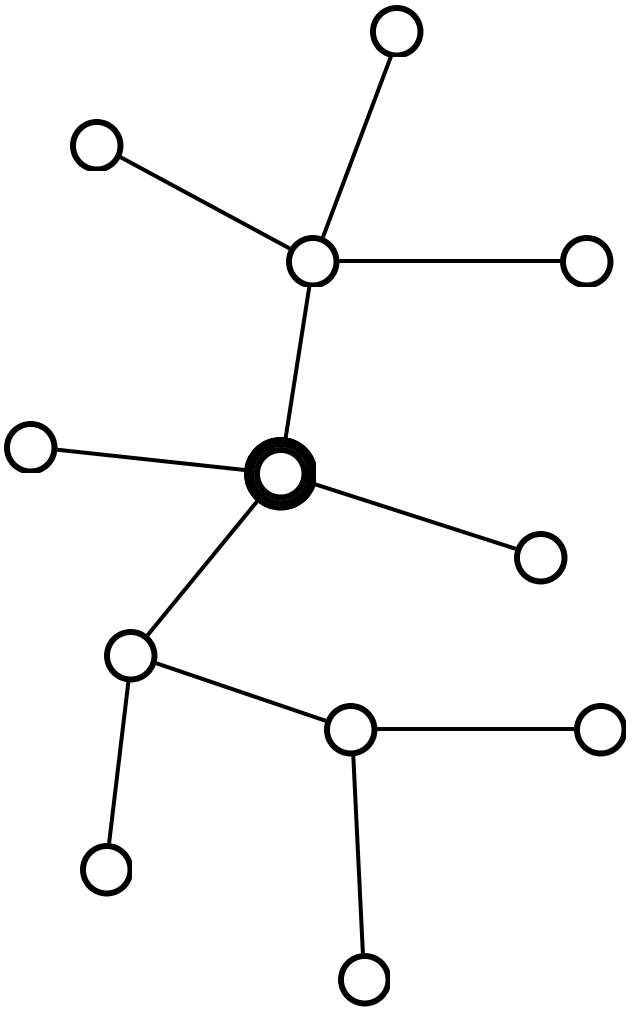
*"C'est fait"* ← *simple ACK*

*"Voici le résultat"* ← *véritable réponse*

*"On a fini"*

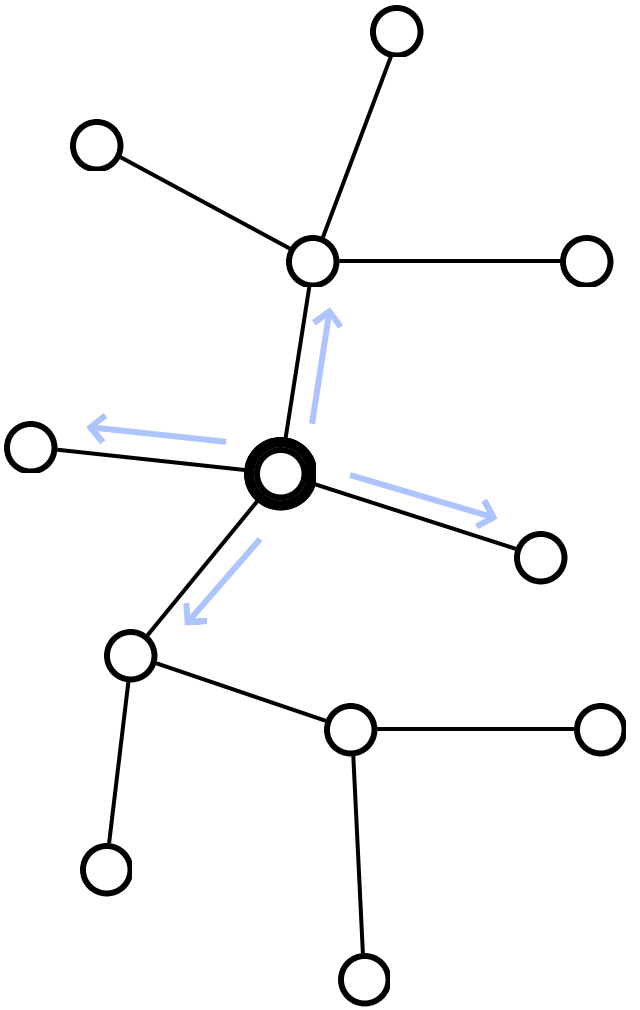
# ↓ Cas particulier - L'arbre

# Solution sur l'arbre



# Solution sur l'arbre

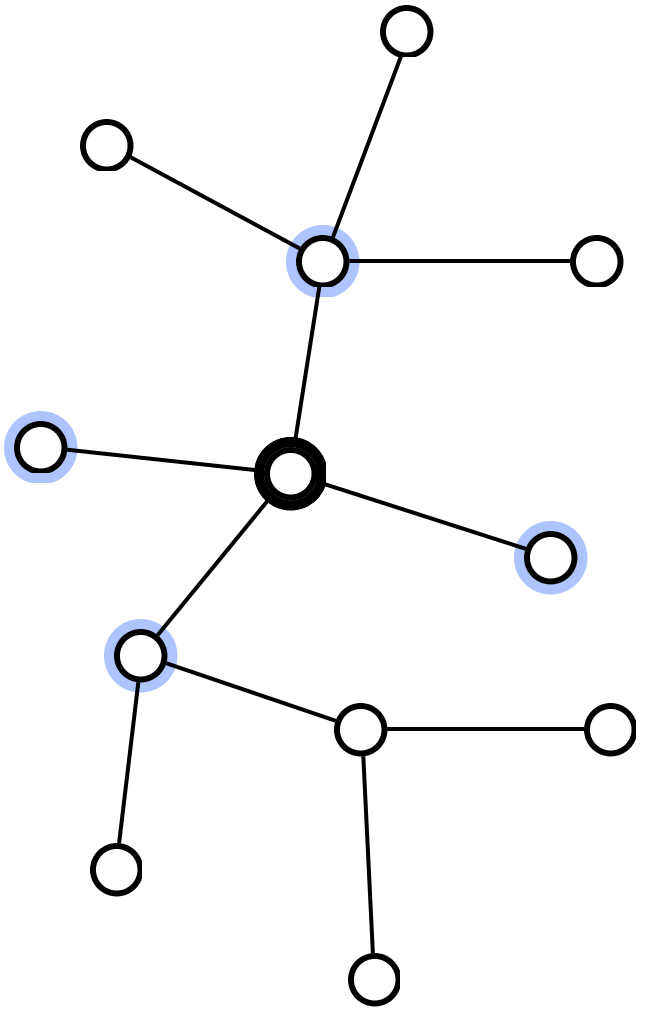
**Source** envoie une `sonde` à ses enfants.



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

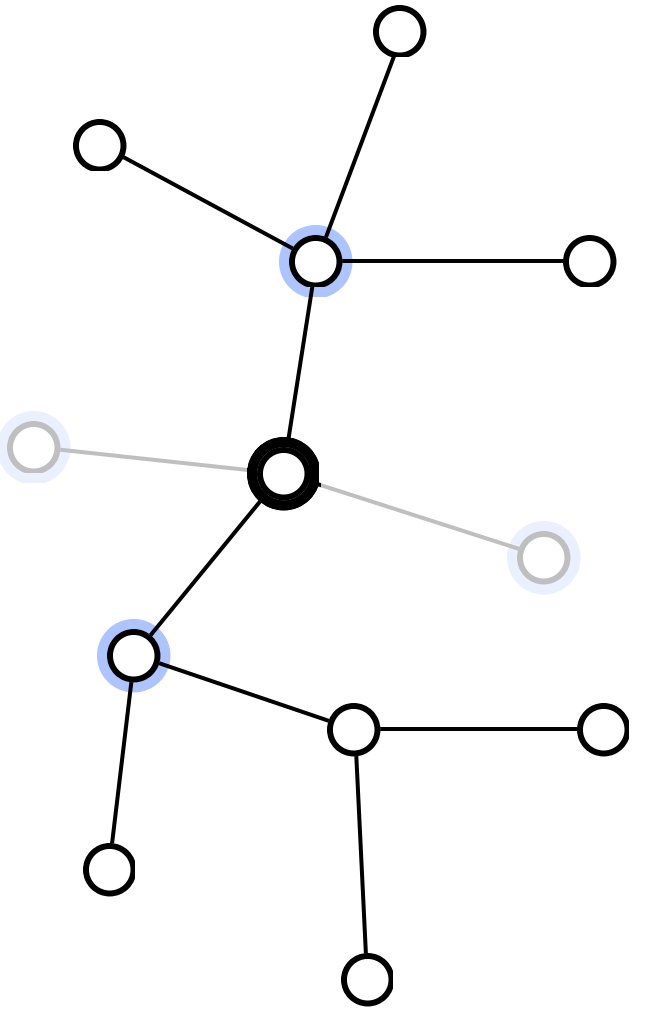
**Réception d'une `sonde`**



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

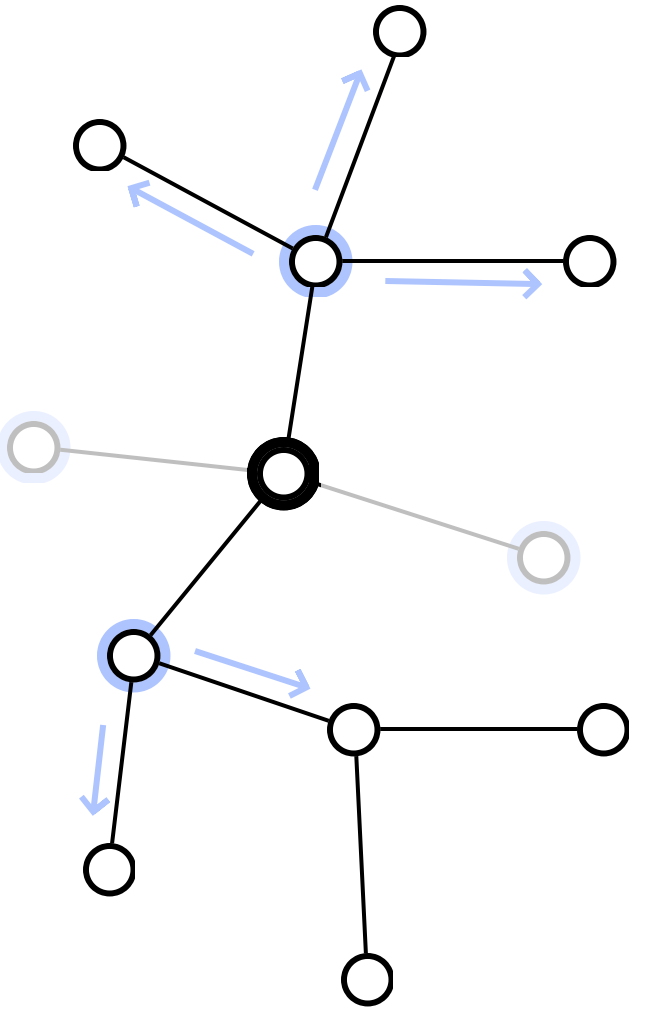


# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants



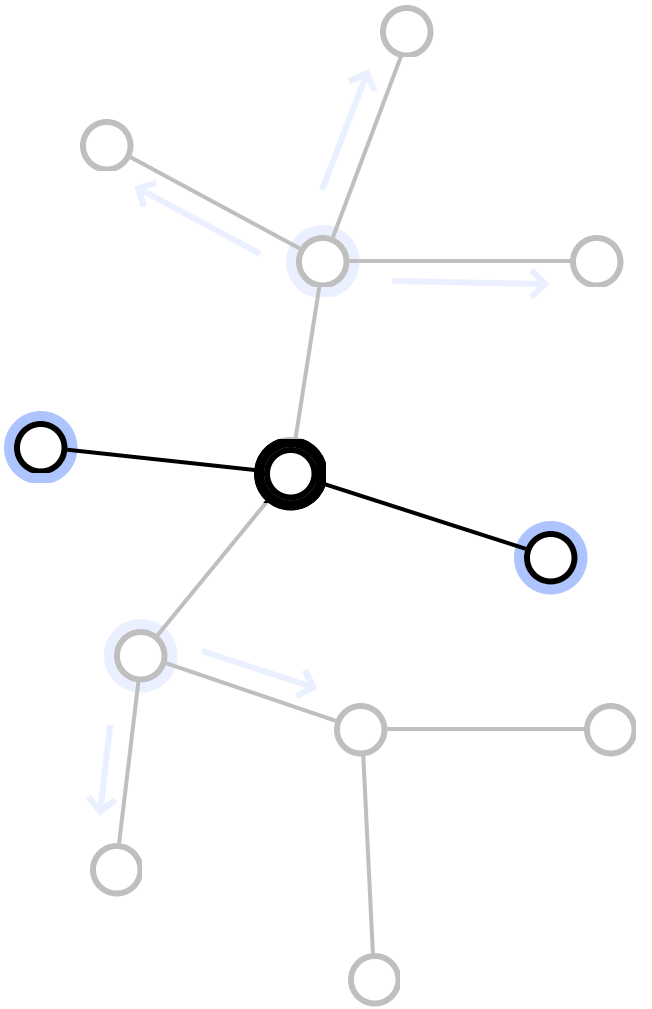
# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

Si je n'ai pas d'enfants



# Solution sur l'arbre

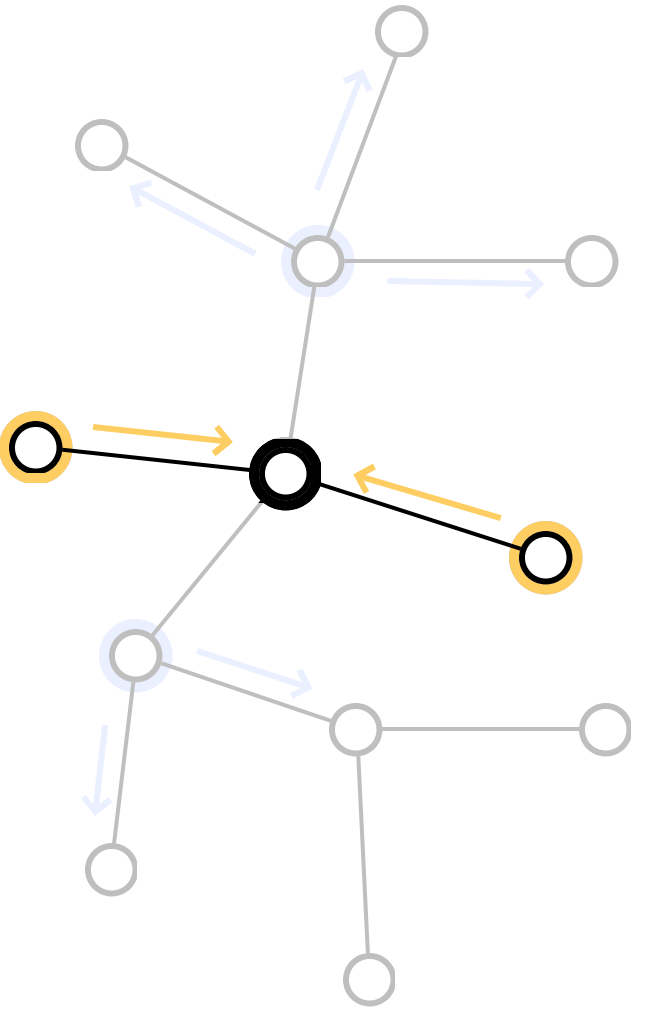
**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

Si je n'ai pas d'enfants

  Je réponds avec `écho`



# Solution sur l'arbre

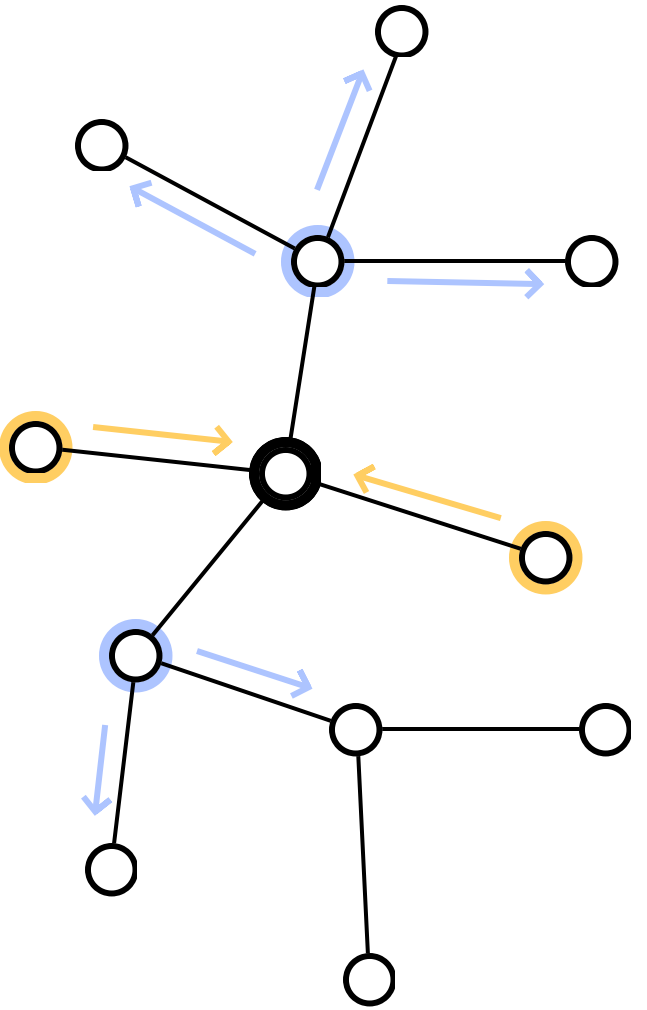
**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

Si je n'ai pas d'enfants

Je réponds avec `écho`



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

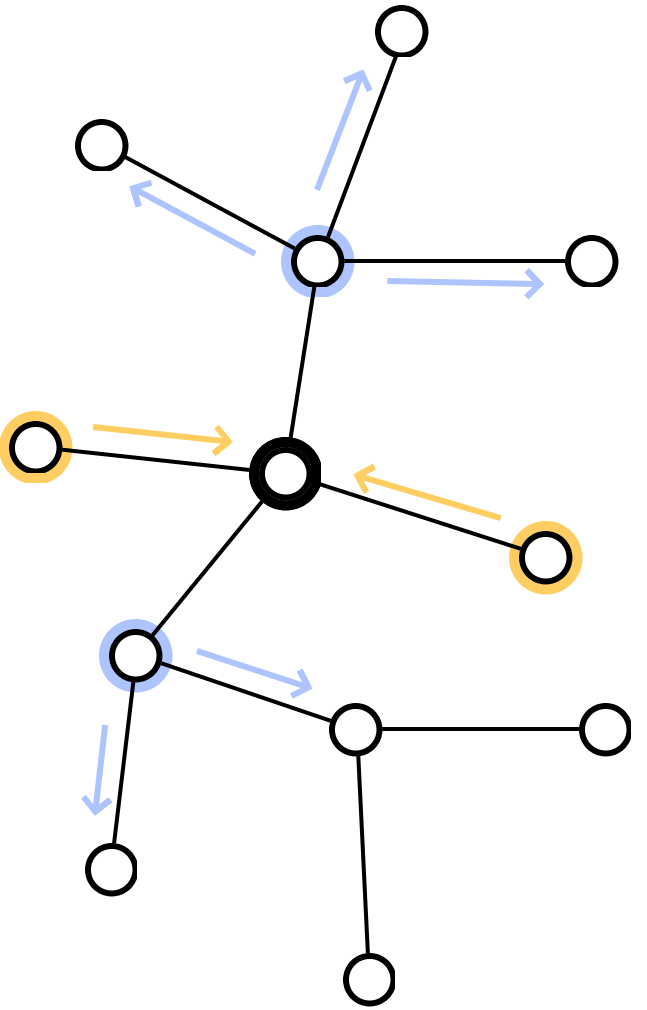
**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

  Je réponds avec `écho`



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

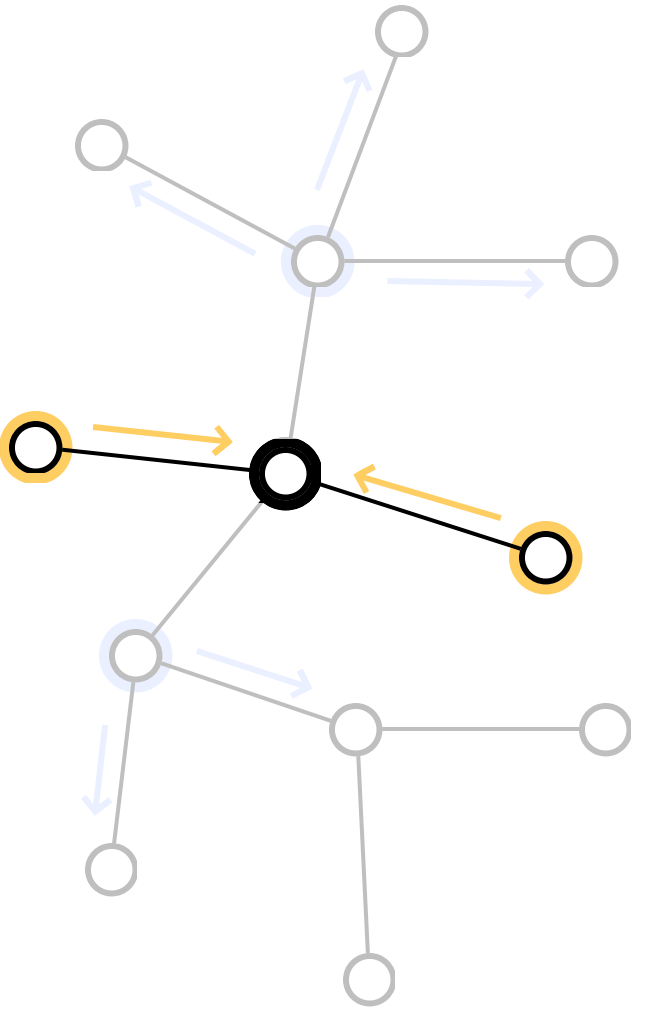
Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

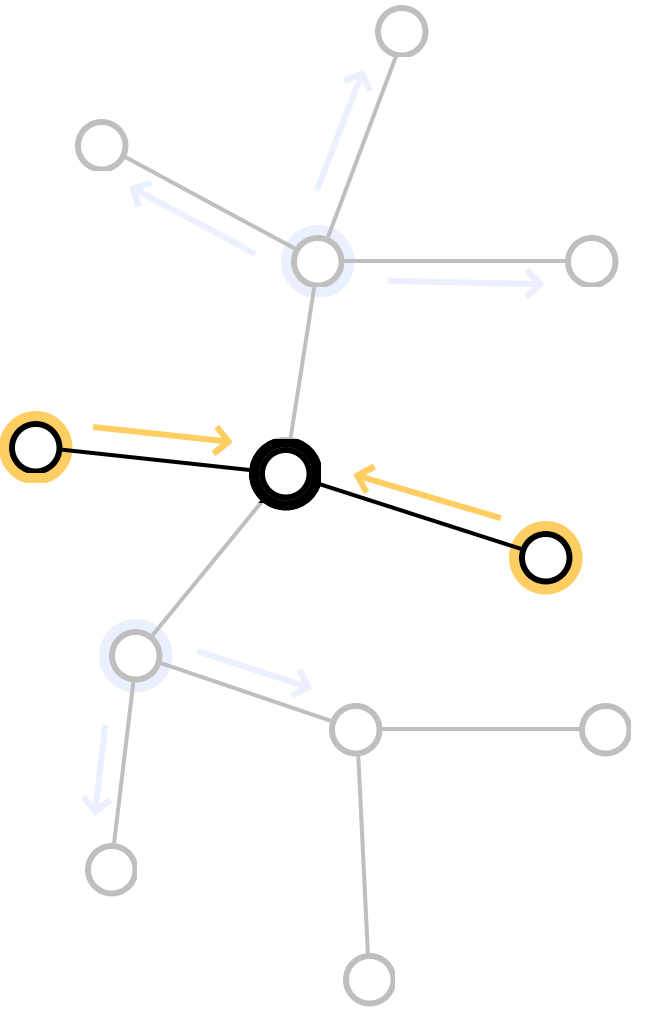
Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

S'il me manque des `échos`

  Je ne fais rien



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

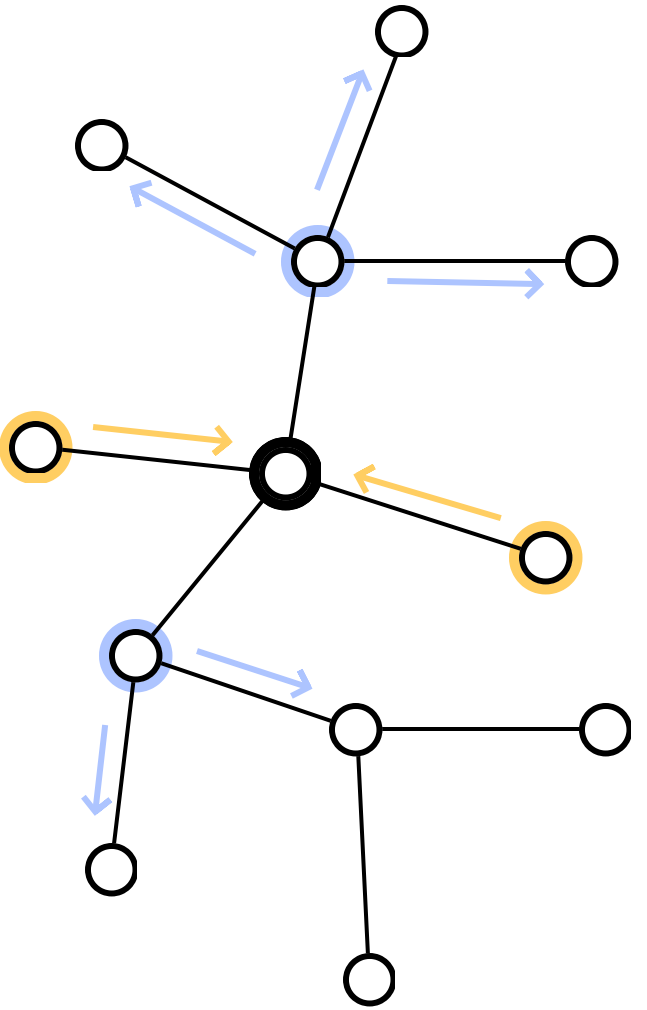
Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

S'il me manque des `échos`

  Je ne fais rien



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

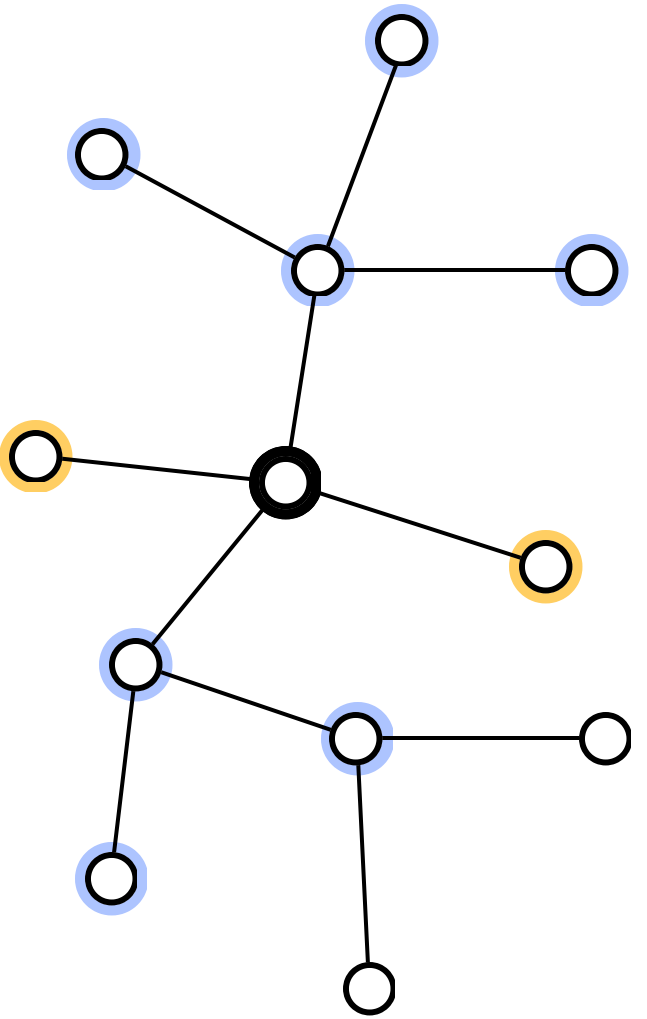
Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

S'il me manque des `échos`

  Je ne fais rien



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

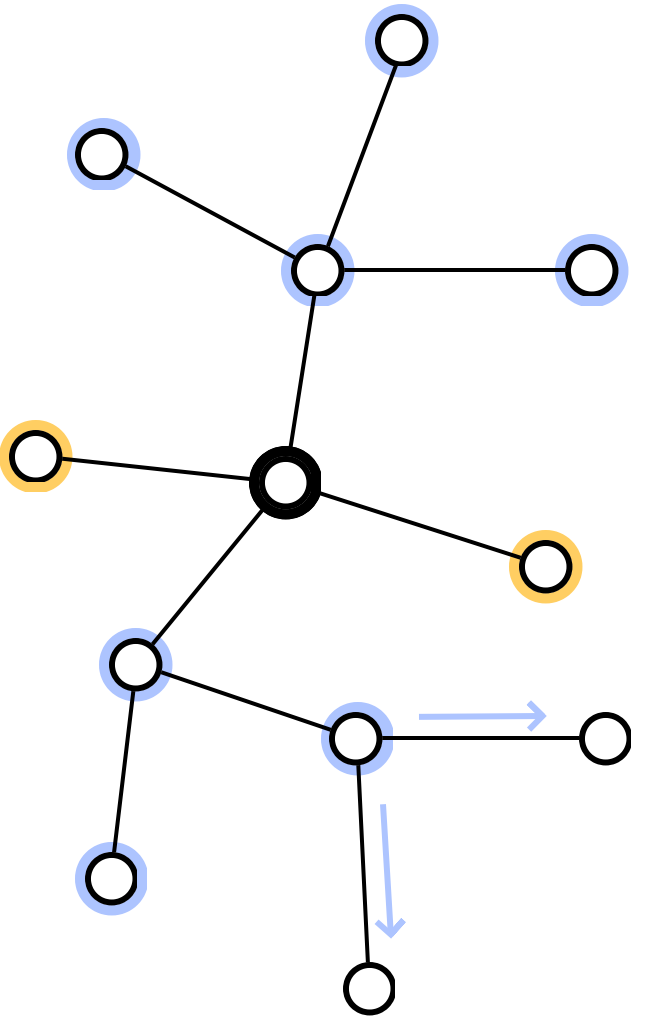
Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

S'il me manque des `échos`

  Je ne fais rien



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

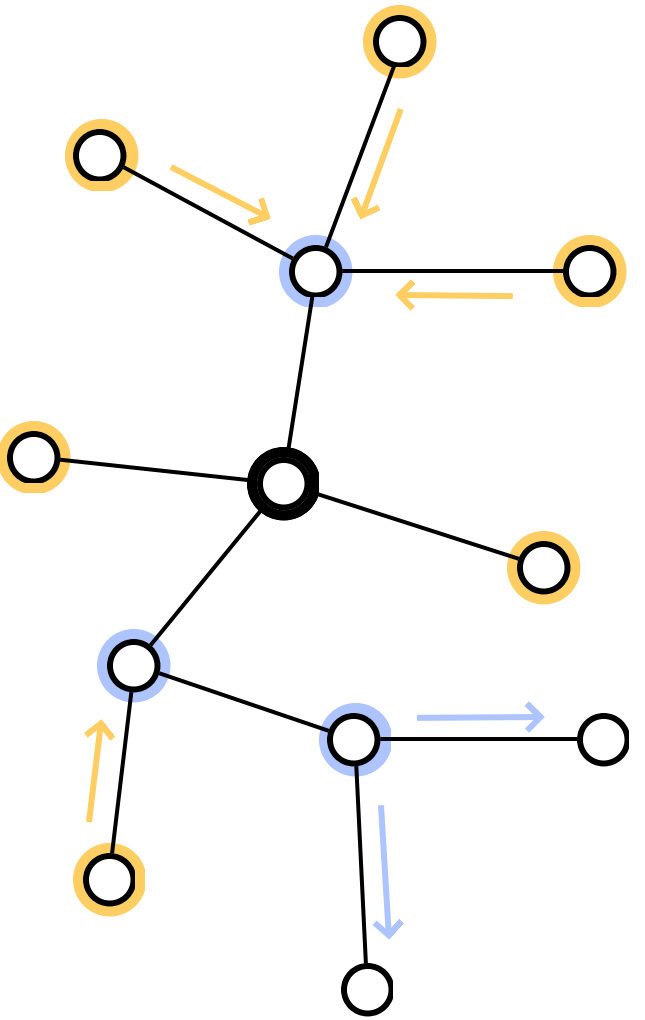
Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

S'il me manque des `échos`

  Je ne fais rien



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

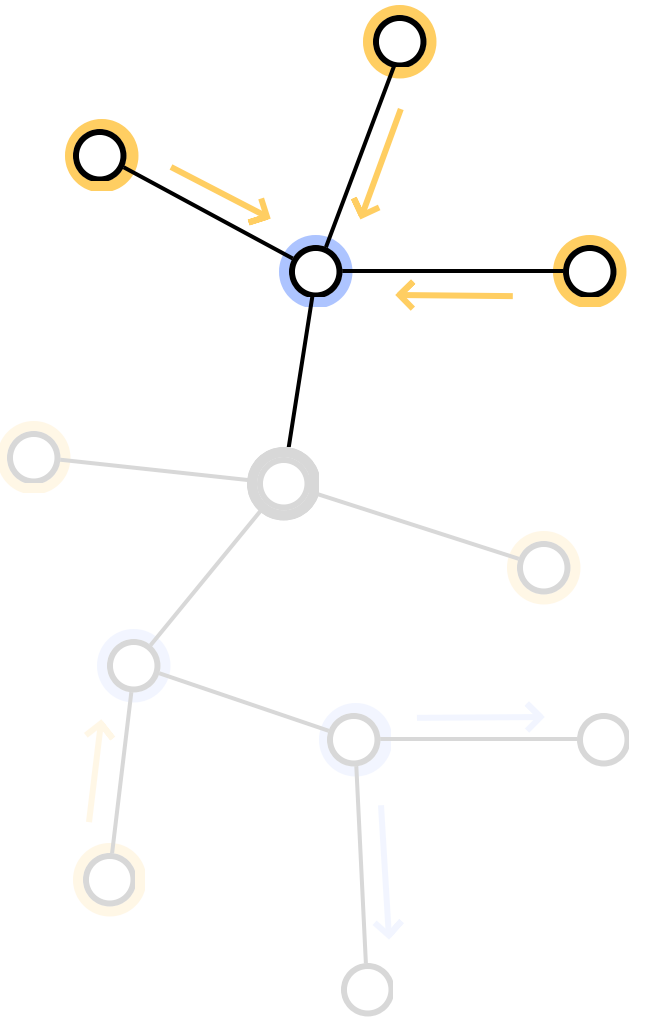
*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

Si j'ai tous les `échos`



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

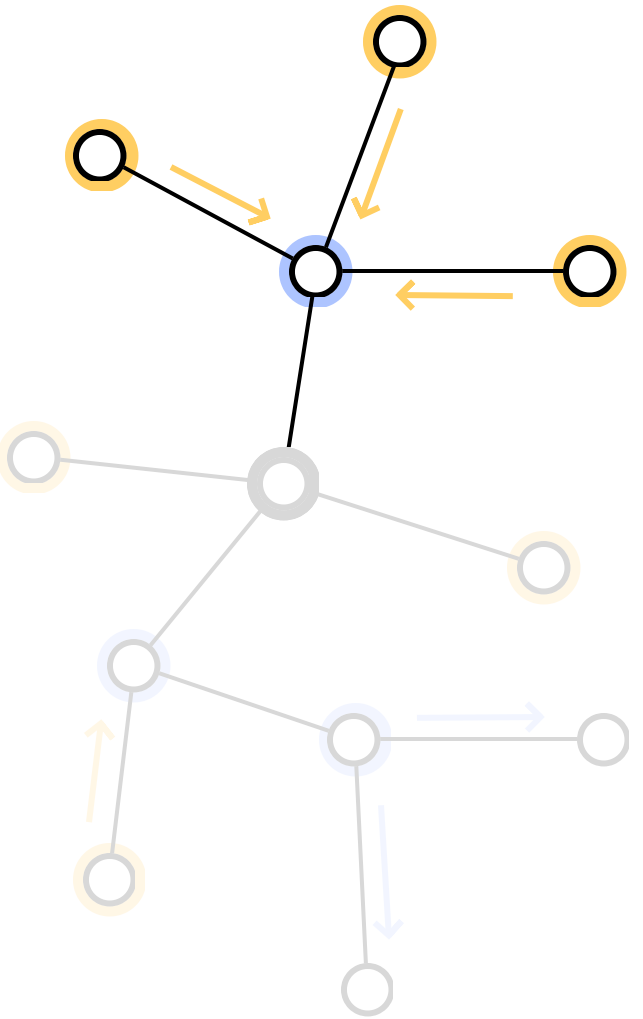
Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

Si j'ai tous les `échos`

  *Je fais des calculs (optionnel)*



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

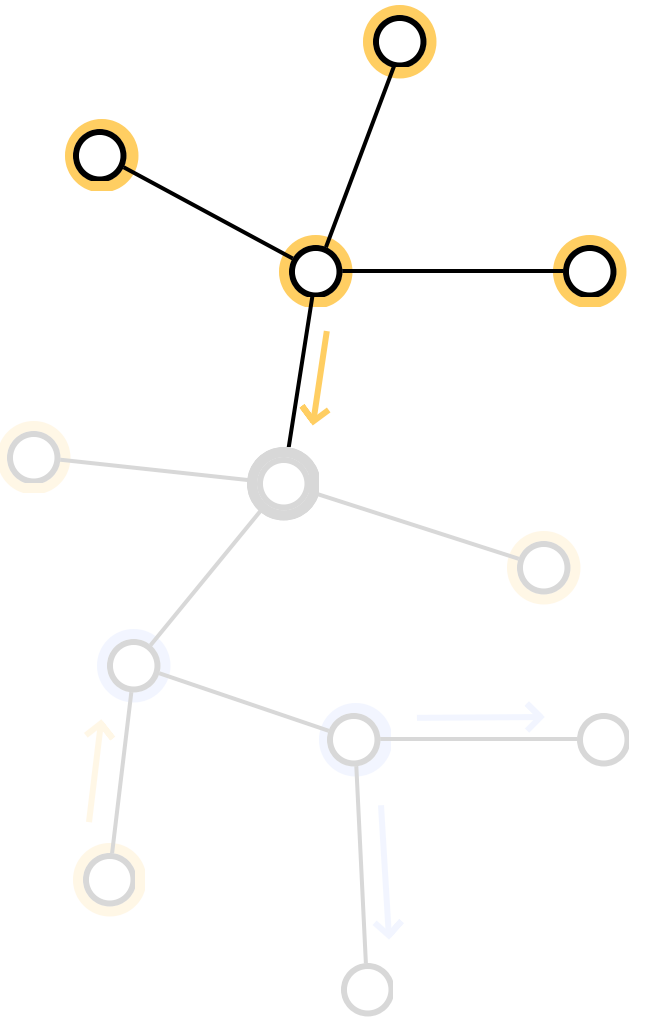
  Je réponds avec `écho`

**Réception d'un `écho`**

Si j'ai tous les `échos`

  *Je fais des calculs (optionnel)*

  Je réponds `écho` à mon parent



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

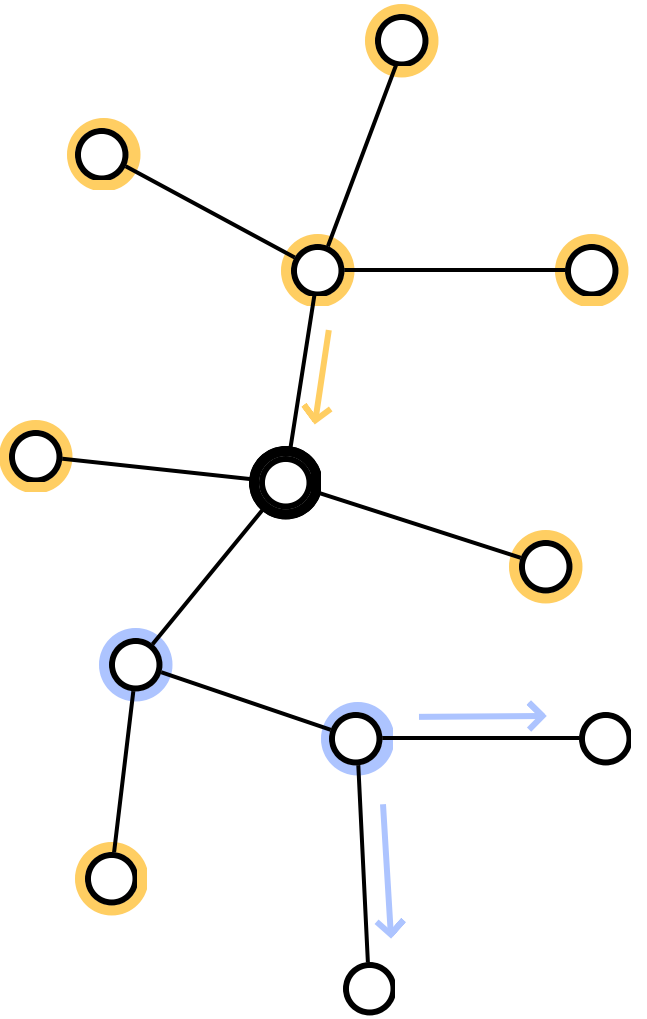
  Je réponds avec `écho`

**Réception d'un `écho`**

Si j'ai tous les `échos`

  *Je fais des calculs (optionnel)*

  Je réponds `écho` à mon parent



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

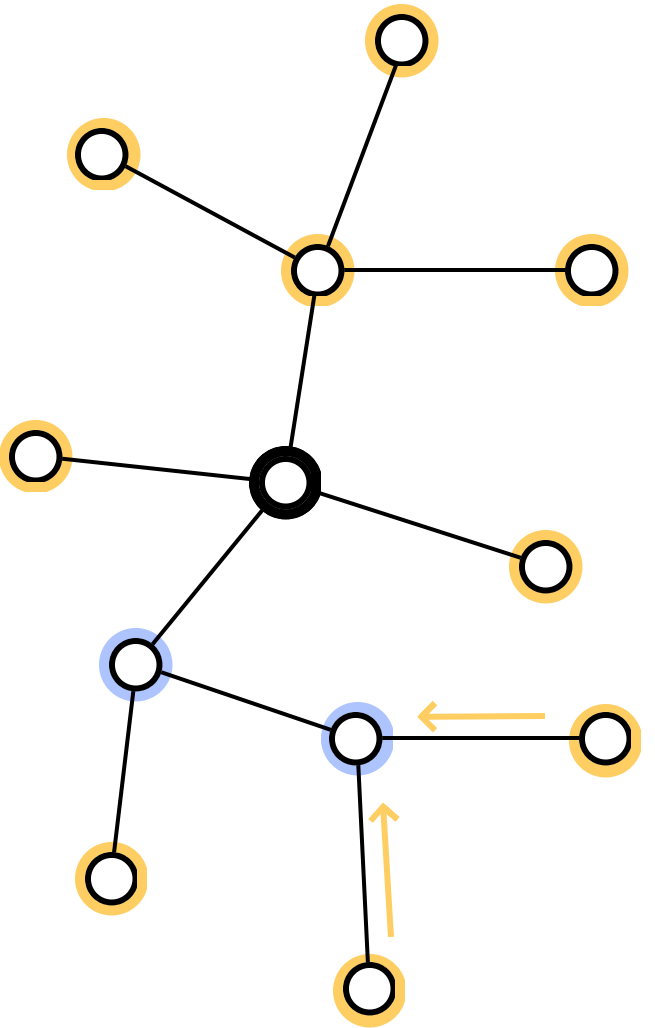
  Je réponds avec `écho`

**Réception d'un `écho`**

Si j'ai tous les `échos`

  *Je fais des calculs (optionnel)*

  Je réponds `écho` à mon parent



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

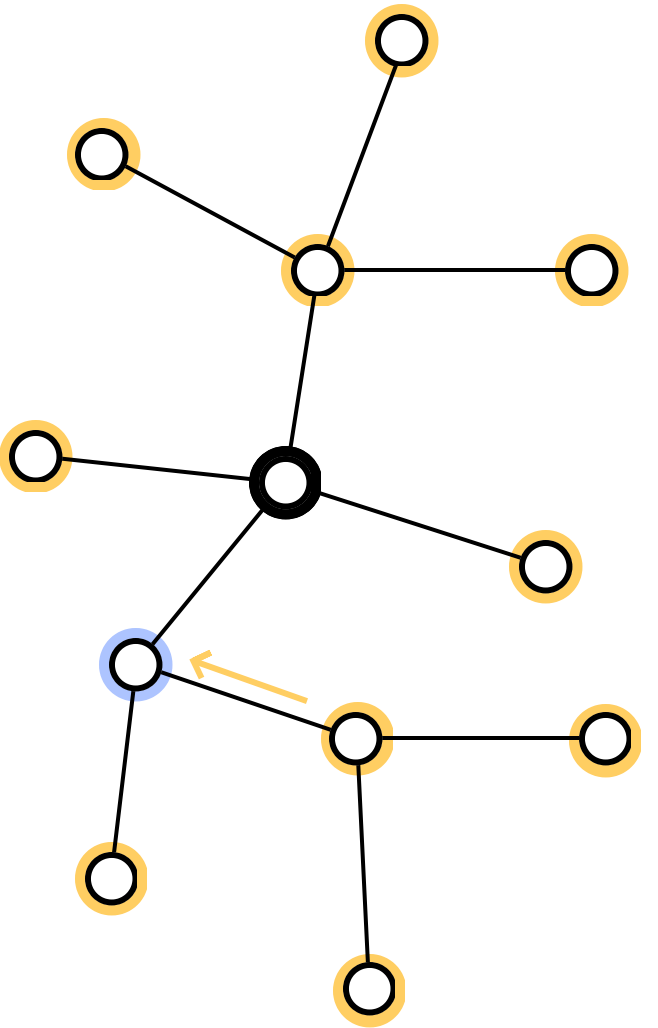
  Je réponds avec `écho`

**Réception d'un `écho`**

Si j'ai tous les `échos`

  *Je fais des calculs (optionnel)*

  Je réponds `écho` à mon parent



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

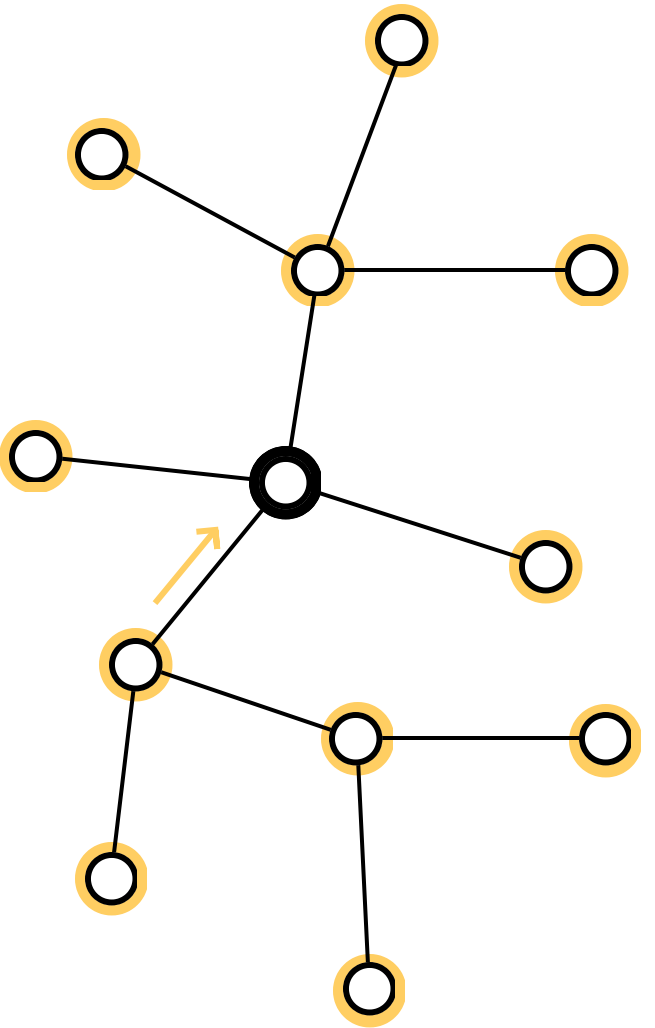
  Je réponds avec `écho`

**Réception d'un `écho`**

Si j'ai tous les `échos`

  *Je fais des calculs (optionnel)*

  Je réponds `écho` à mon parent



# Solution sur l'arbre

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Je la propage à mes enfants

*Je fais des calculs (optionnel)*

Si je n'ai pas d'enfants

  Je réponds avec `écho`

**Réception d'un `écho`**

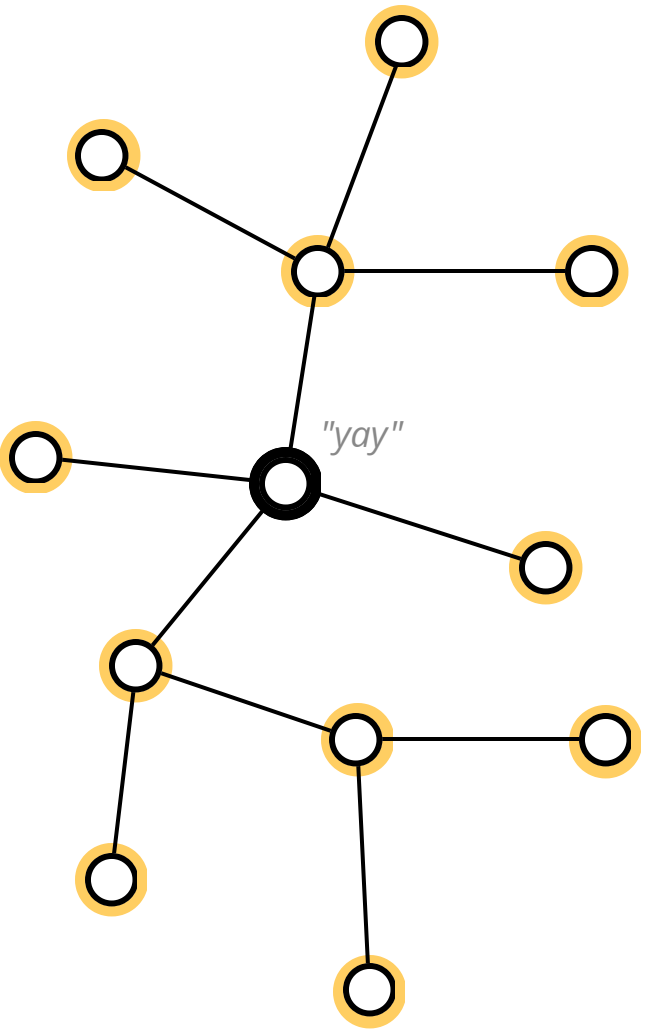
Si j'ai tous les `échos`

  *Je fais des calculs (optionnel)*

  Je réponds `écho` à mon parent

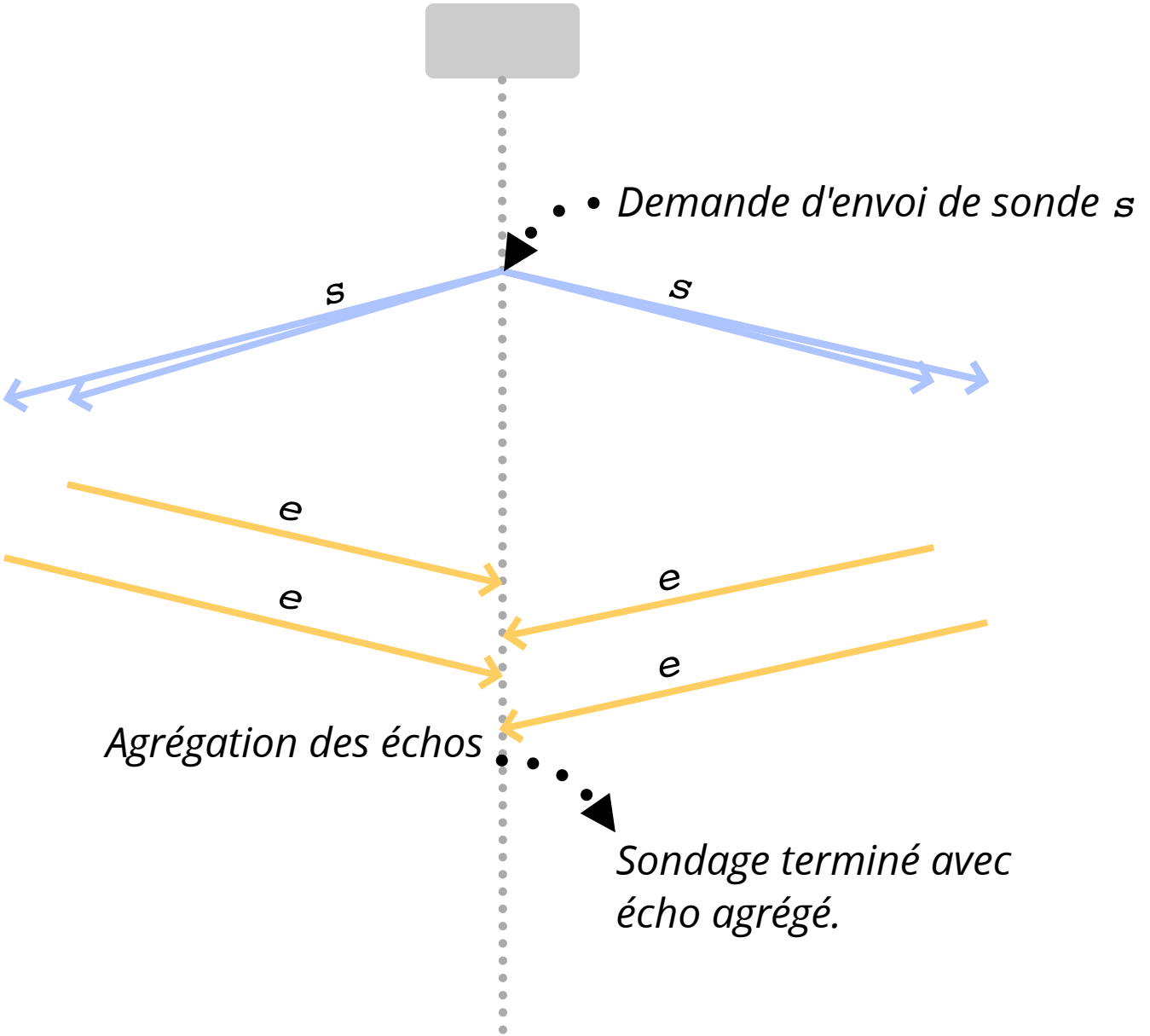
  Si je suis la source

    J'ai fini

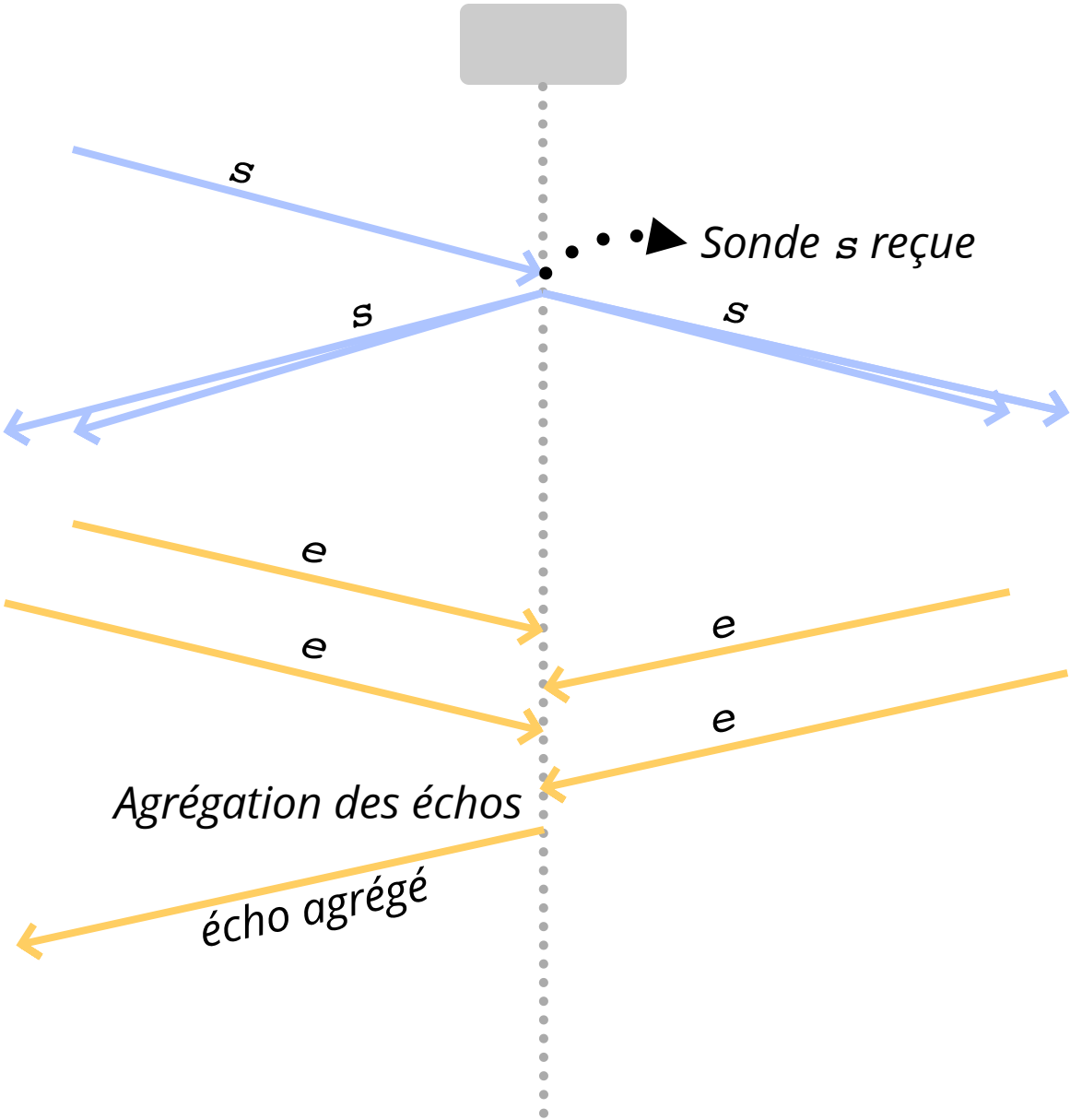


# ↓ Comportement d'un noeud

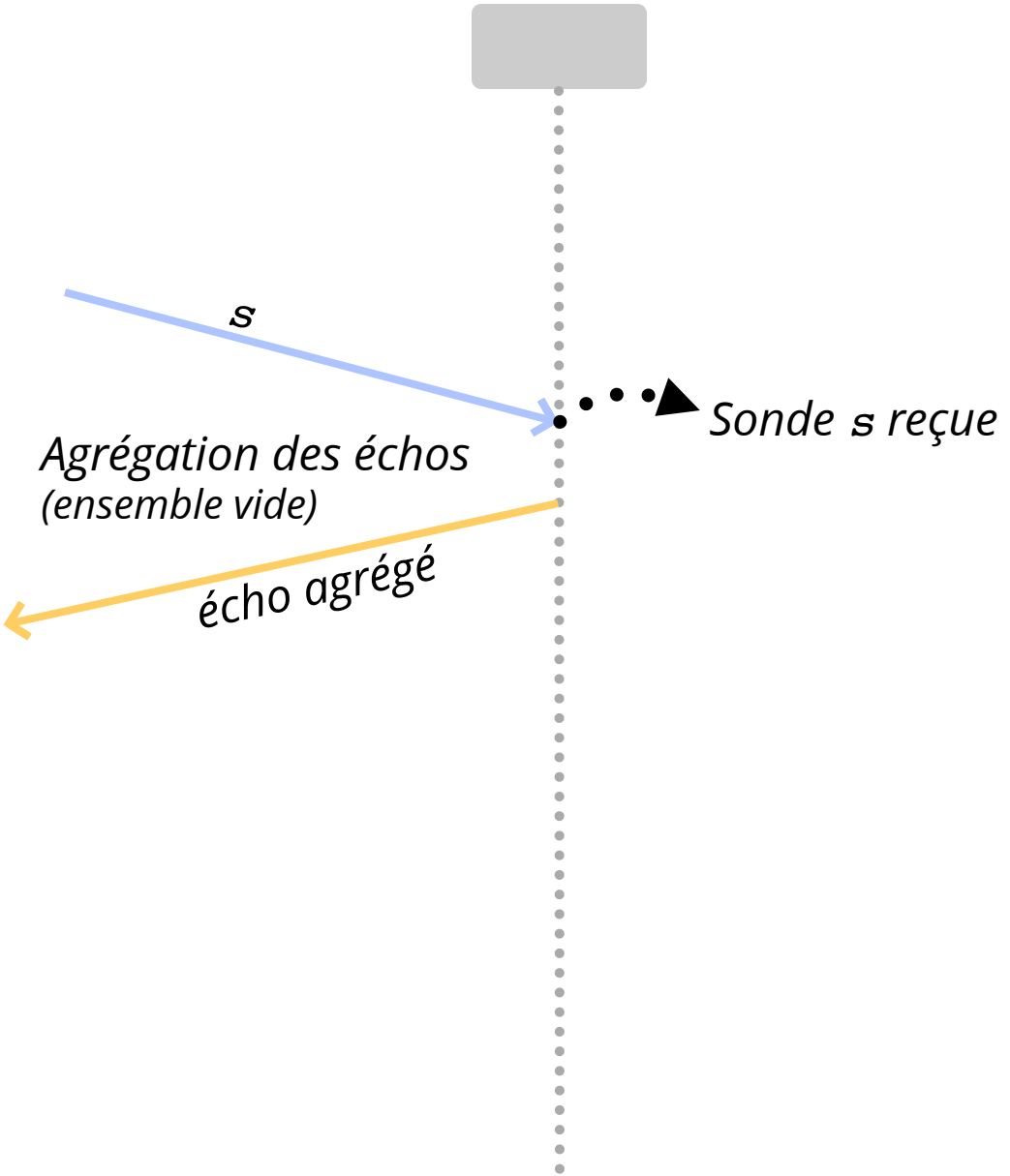
# Comportement de la source



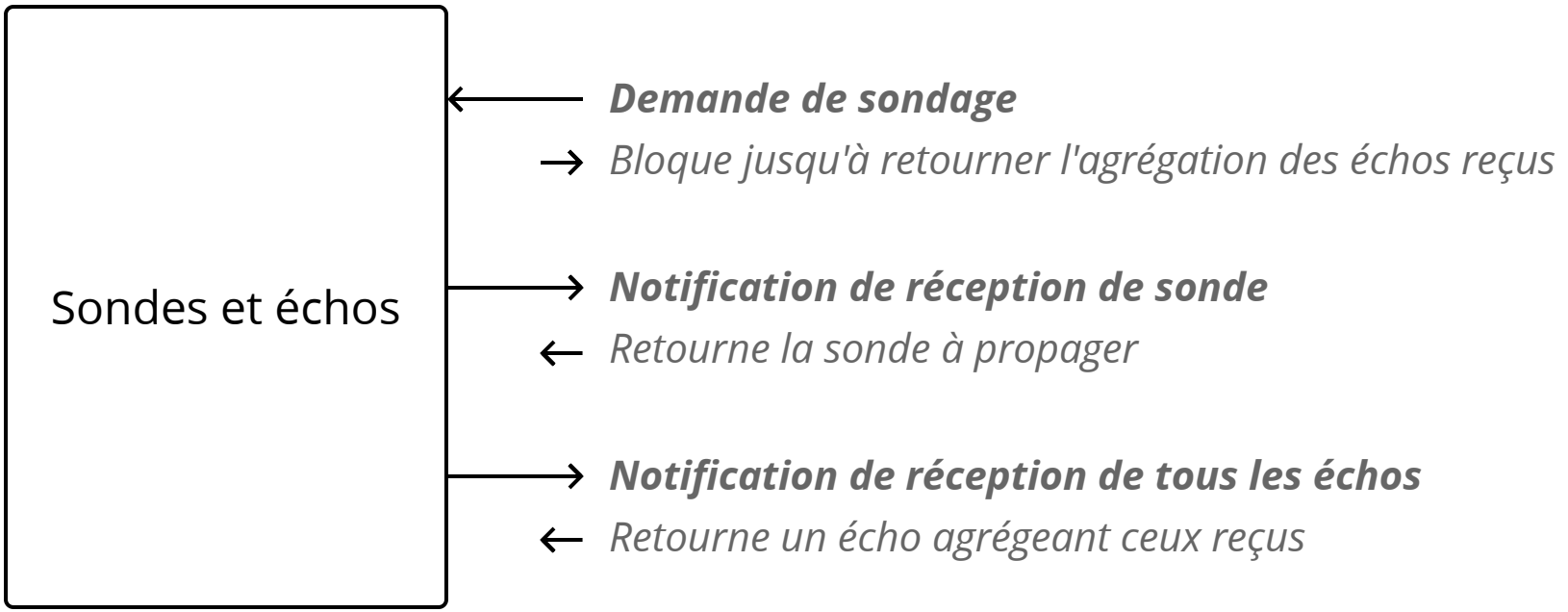
# Comportement d'un intermédiaire



# Comportement d'une "feuille" *(un seul voisin)*



# Interface


Sondes et échos
Demande de sondage
Bloque jusqu'à retourner l'agrégation des échos reçus
Notification de réception de sonde
Retourne la sonde à propager
Notification de réception de tous les échos
Retourne un écho agrégeant ceux reçus

# Cas particulier - L'arbre

Pseudocode

# Pseudocode

Variables

| | | |
|---|---|---|
| `parent` | *entier, init nil* | id du processus parent |
| `voisins` | *liste d'entiers* | ids de mes processus voisins |
| `echos_reçus` | *liste d'échos* | échos reçus |
| `receptionSonde` | *callback* | à appeler à la réception d'une sonde |
| `agrégerEchos` | *callback* | à appeler une fois tous les échos reçus |

# Pseudocode

Initialisation

> Écouter infiniment les événements suivants :
>
> - Réception d'une sonde de la part de `i`
> - Réception d'un écho de la part de `i`
> - Demande d'envoi d'une sonde

*Note : les traitements de ces événements sont faits de manière séquentielle.*

*Note : on suppose aussi pour l'instant aucune sonde concurrente.*

# Pseudocode

Demande d'envoi d'une sonde `s`

```
parent = nil
Envoyer s à tous les voisins
echos_reçus = []
Si voisins est vide :
    echo := agrégerEchos(echos_reçus)
    Terminer le sondage avec echo
```

# Pseudocode

Réception d'une sonde `s` de `i`

```
parent = i
echos_reçus = []
s := receptionSonde(s)
Envoyer s à tous les voisins sauf i
Si voisins a une taille de 1 :
    echo := agrégerEchos(echos_reçus)
    Envoyer echo à parent
```

# Pseudocode

Réception d'un écho `e` de `i`

```
Ajouter e à echos_reçus
Si parent est nil
    Si echos_reçus a la taille de voisins
        echo := agrégerEchos(echos_reçus)
        Terminer le sondage avec echo
Sinon
    Si echos_reçus a la taille de voisins - 1
        echo := agrégerEchos(echos_reçus)
        Envoyer echo à parent
```

# ↓ Cas général - Graphe

# Stratégie sur graphe arbitraire

# Stratégie sur graphe arbitraire

**Solution 1**

Intégrer un arbre de recouvrement et utiliser l'algorithme précédent.

*(a.k.a. spanning tree)*

# Stratégie sur graphe arbitraire

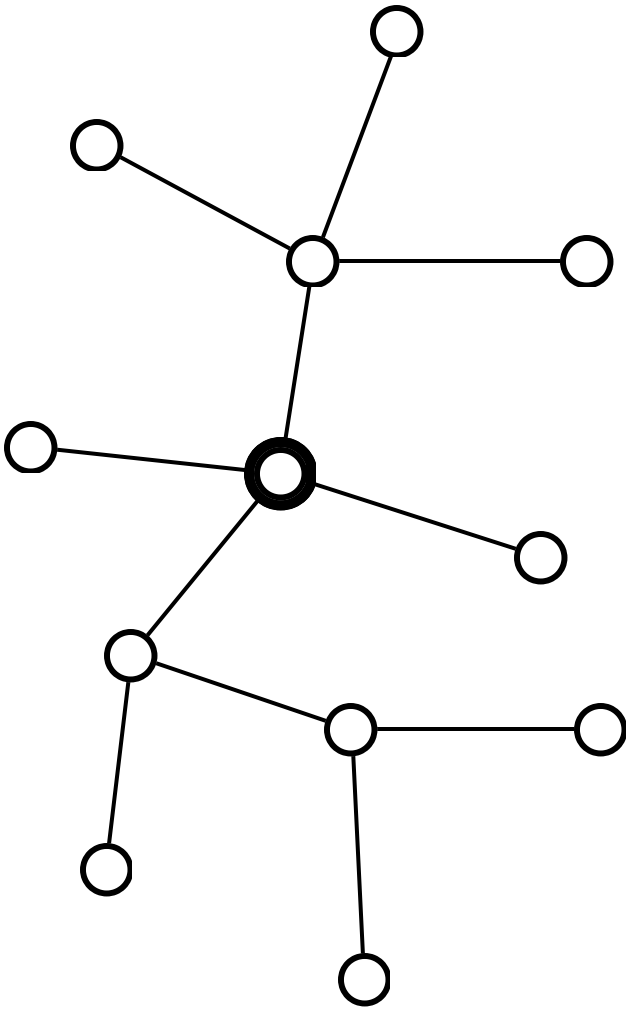
**Solution 1**

Intégrer un arbre de recouvrement et utiliser l'algorithme précédent.
*(a.k.a. spanning tree)*

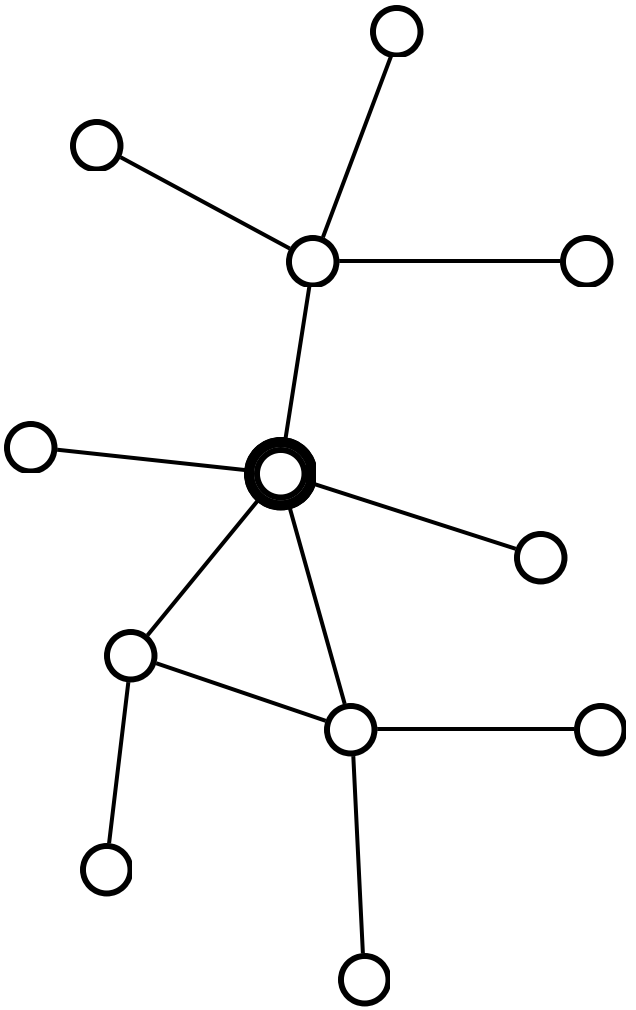
**Solution 2**

Un algorithme spécialisé.

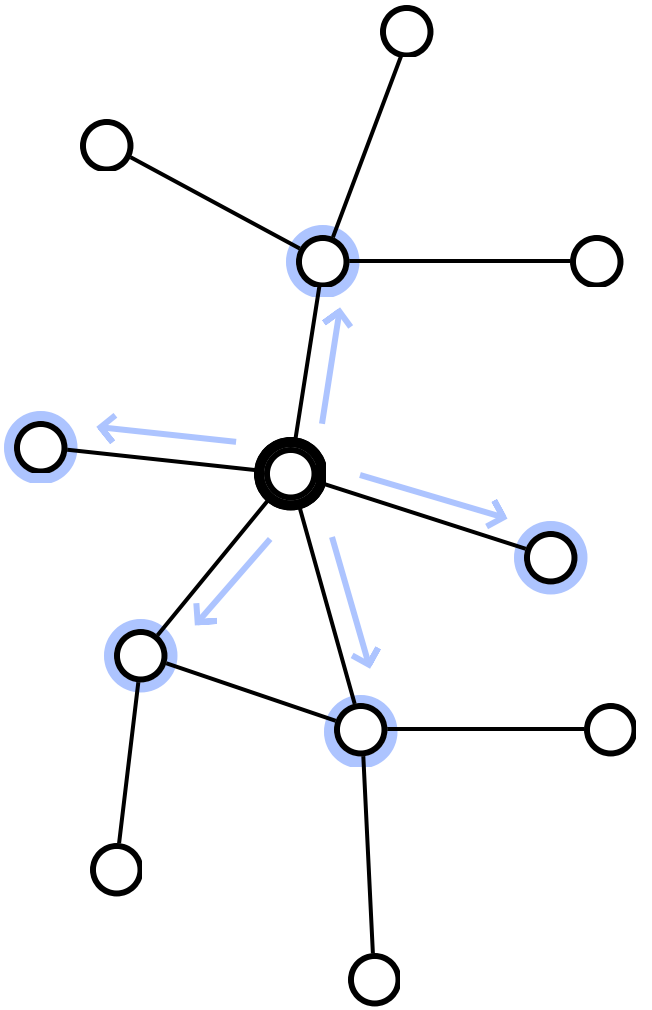
# Problème



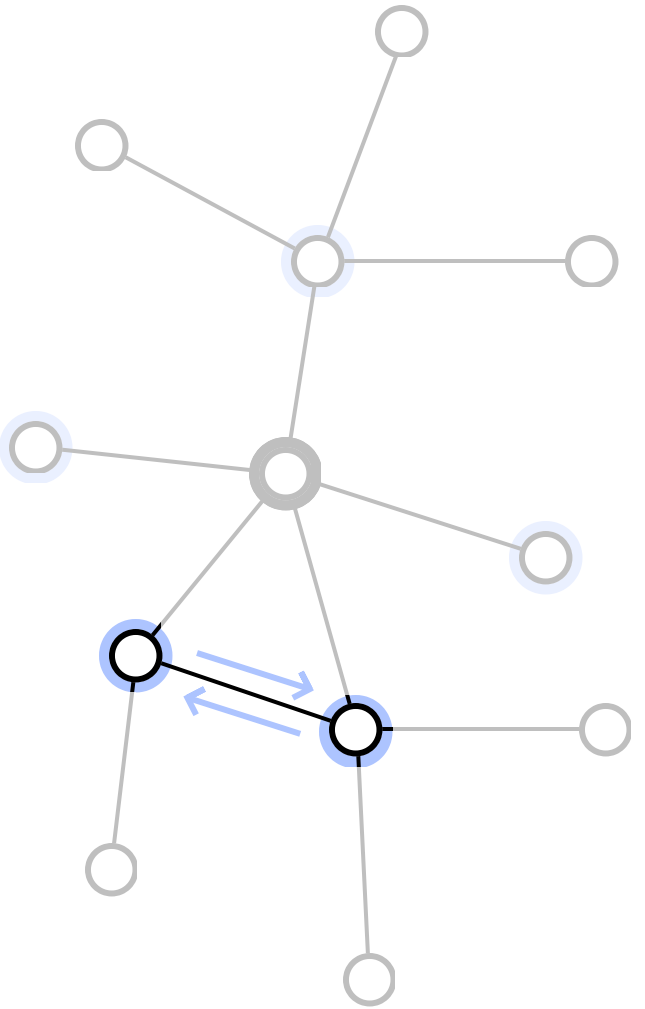
# Problème



# Problème

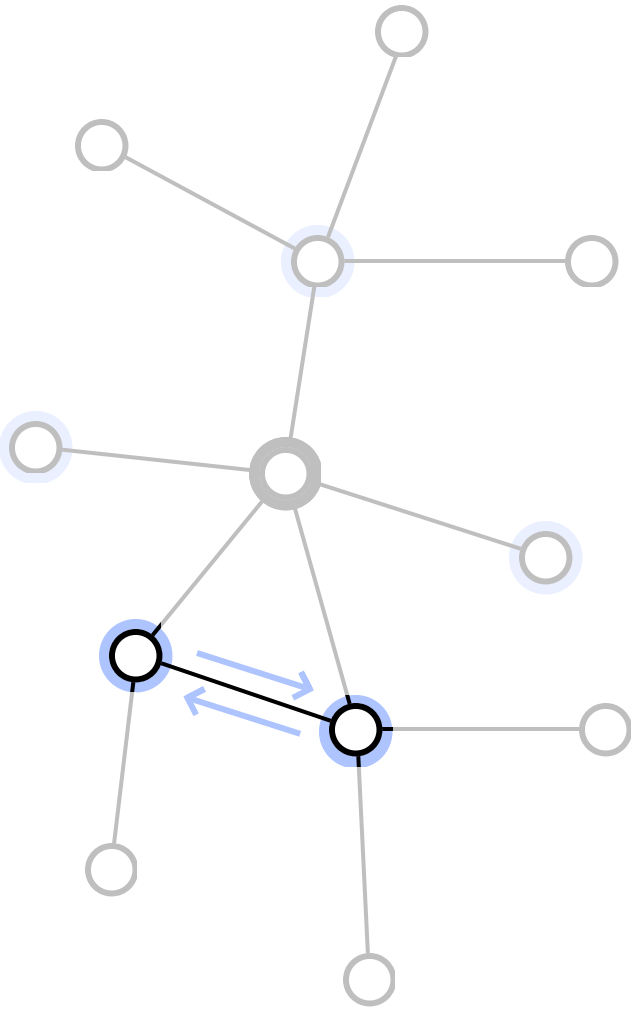


# Problème



# Problème

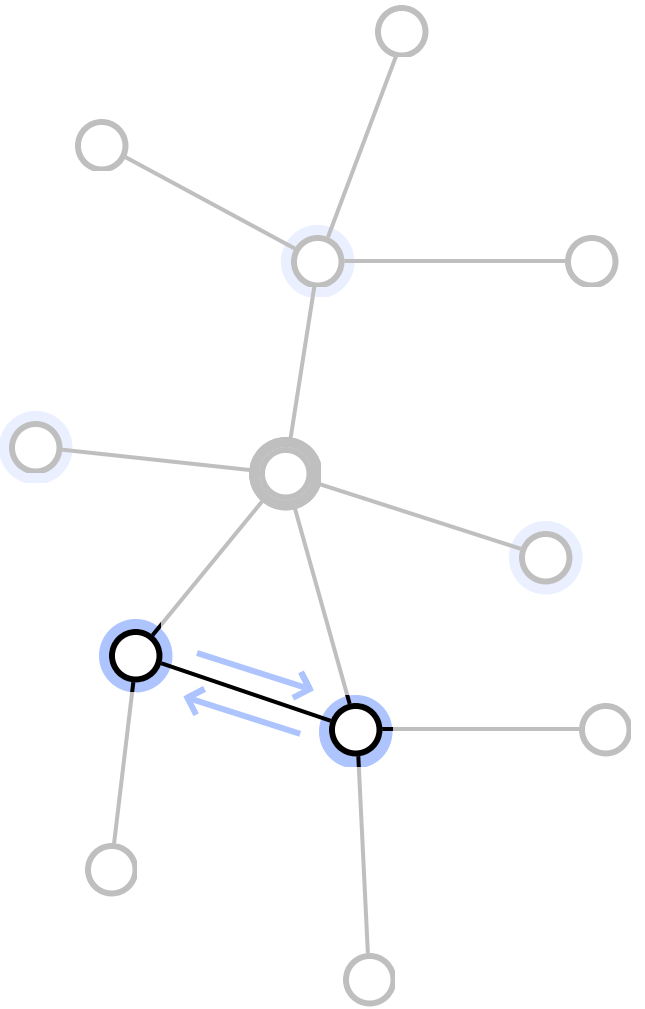
Sur une boucle, deux sondes vont inévitablement se croiser.



# Problème

Sur une boucle, deux sondes vont inévitablement se croiser.

**Que faire ?**

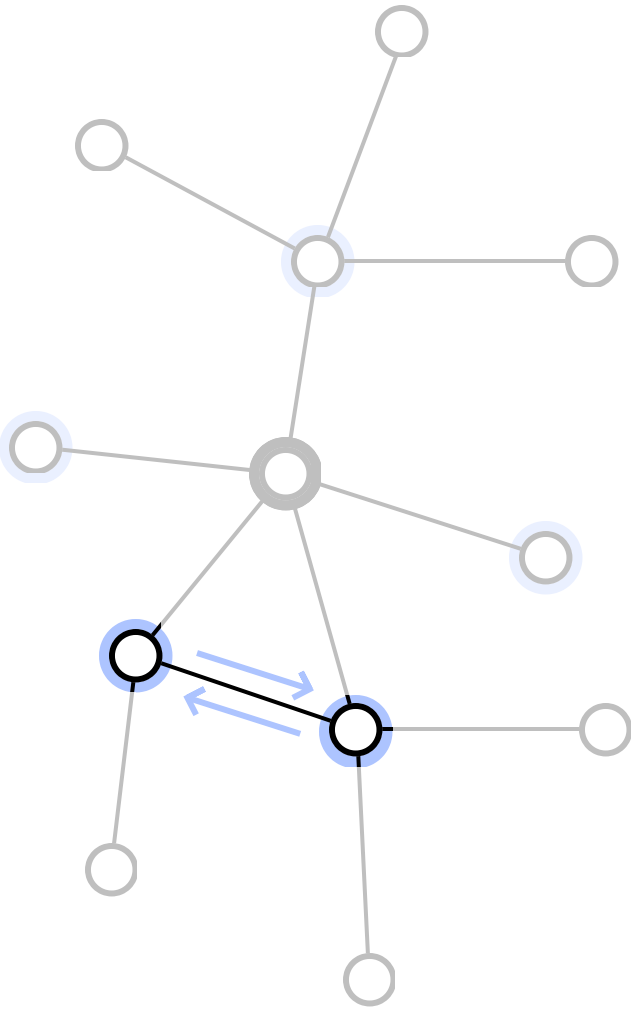


# Problème

Sur une boucle, deux sondes vont inévitablement se croiser.

**Que faire ?**

Les ignorer…



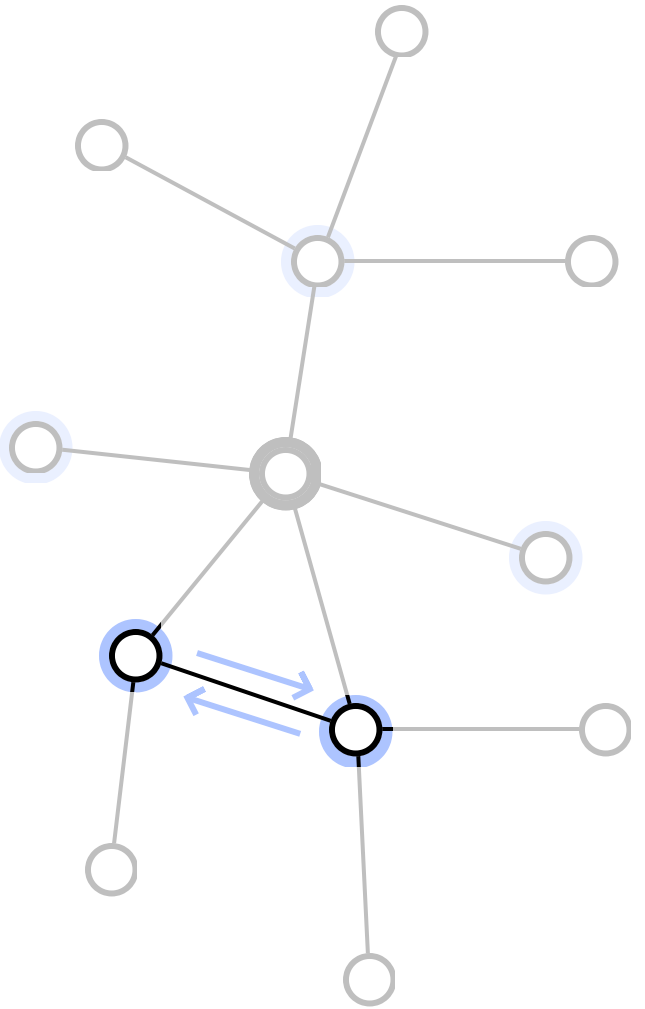
# Problème

Sur une boucle, deux sondes vont inévitablement se croiser.

**Que faire ?**

Les ignorer…

Problème : on compte les échos reçus.



# Problème

Sur une boucle, deux sondes vont inévitablement se croiser.

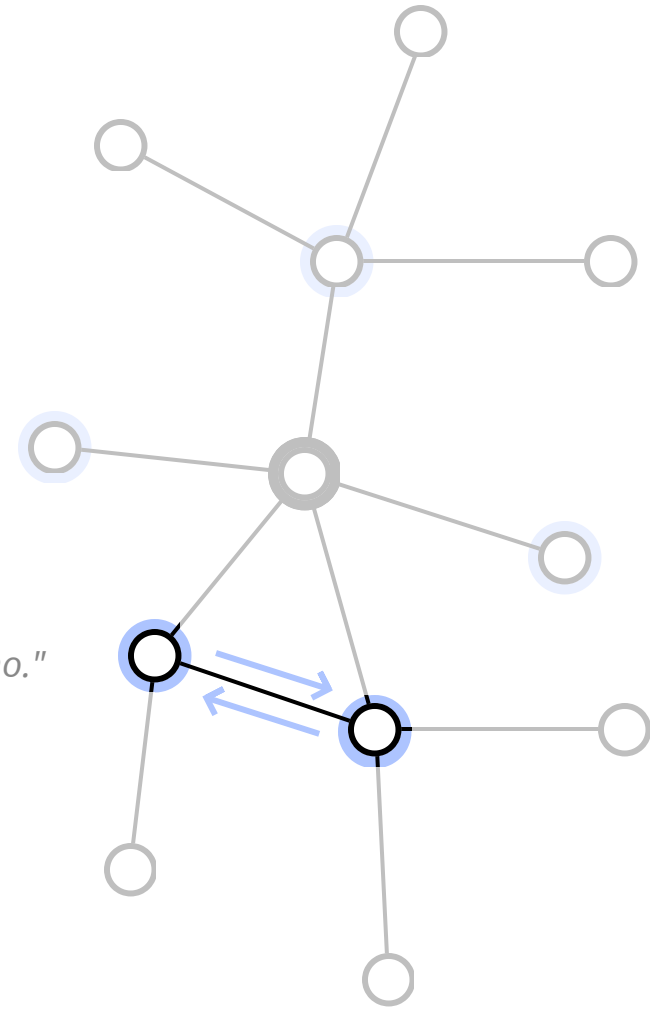
**Que faire ?**

Les ignorer…

Problème : on compte les échos reçus.

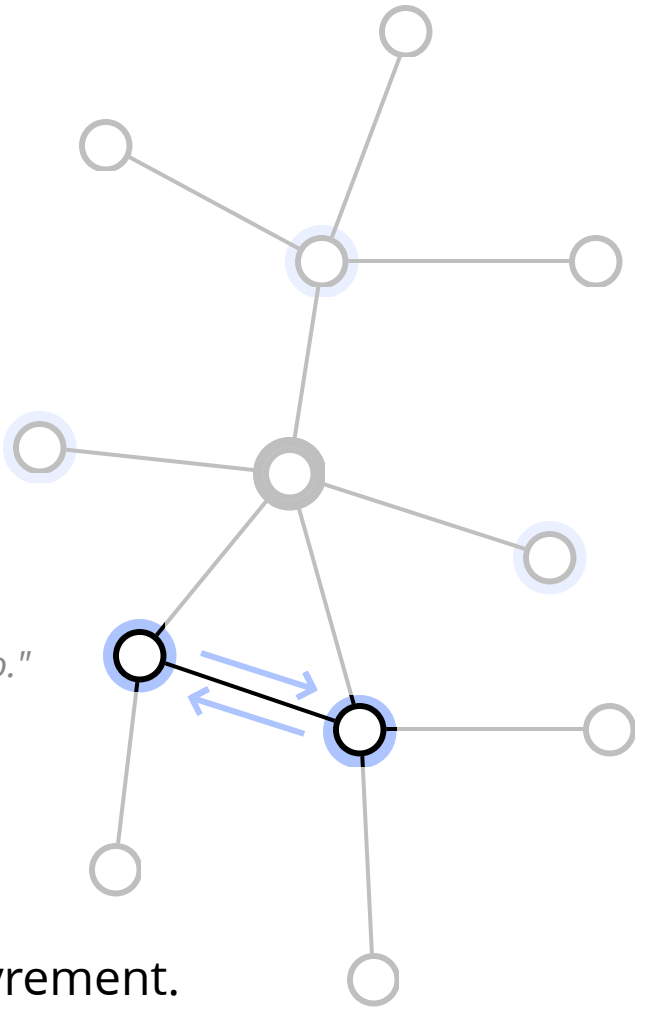
Ne plus attendre d'écho de ce voisin.

*"Si je reçois de ta part une sonde que j'ai déjà,*
*c'est que tu as reçu une sonde avant la mienne.*
*Tu es donc déjà l'enfant de quelqu'un, je n'attends plus ton écho."*



# Problème

Sur une boucle, deux sondes vont inévitablement se croiser.



**Que faire ?**

Les ignorer…

Problème : on compte les échos reçus.

Ne plus attendre d'écho de ce voisin.

*"Si je reçois de ta part une sonde que j'ai déjà,*
*c'est que tu as reçu une sonde avant la mienne.*
*Tu es donc déjà l'enfant de quelqu'un, je n'attends plus ton écho."*

**Remarque :**

Les sondes traitées forment un arbre de recouvrement.

=> *Retour à l'algorithme initial.*

# Solution

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Si je connais déjà cette sonde
  J'arrête d'attendre l'écho de ce voisin.
Sinon
  Je la propage à mes enfants
  *Je fais des calculs (optionnel)*
  Si je n'ai pas d'enfants
    Je réponds avec `écho`

**Réception d'un `écho`**

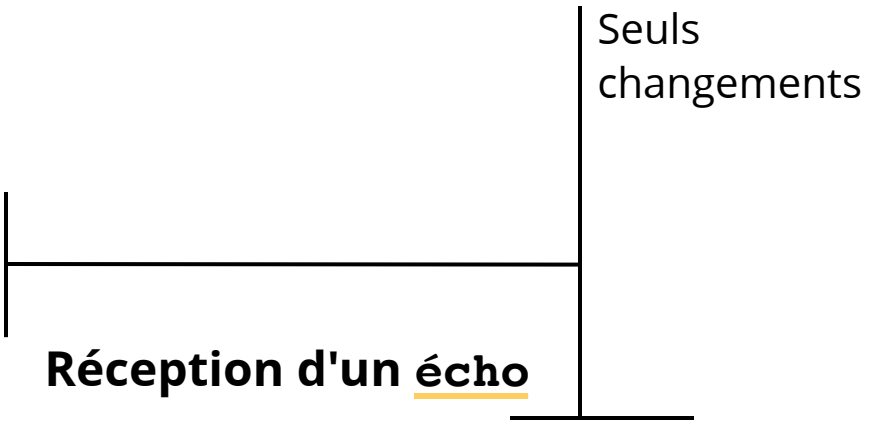
Si j'ai tous les `échos` attendus
  *Je fais des calculs (optionnel)*
  Je réponds `écho` à mon parent
  Si je suis la source
    J'ai fini

# Solution

**Source** envoie une `sonde` à ses enfants.

**Réception d'une `sonde`**

Si je connais déjà cette sonde
  J'arrête d'attendre l'écho de ce voisin.
Sinon
  Je la propage à mes enfants
  *Je fais des calculs (optionnel)*
  Si je n'ai pas d'enfants
    Je réponds avec `écho`




**Réception d'un `écho`**

Si j'ai tous les `échos` attendus
  *Je fais des calculs (optionnel)*
  Je réponds `écho` à mon parent
  Si je suis la source
    J'ai fini

# Cas général - Graphe

Pseudocode

# Pseudocode

Variables

| | | |
|---|---|---|
| `parent` | *entier, init nil* | id du processus parent |
| `voisins` | *liste d'entiers* | ids de mes processus voisins |
| `echos_reçus` | *liste d'échos* | échos reçus |
| `echos_attendus` | *entier* | nombre d'échos attendus |
| `sonde_reçue` | *bool, init false* | si j'ai déjà reçu une sonde |
| `receptionSonde` | *callback* | à appeler à la réception d'une sonde |
| `agrégerEchos` | *callback* | à appeler une fois tous les échos reçus |

# Pseudocode

Demande d'envoi d'une sonde `s`

```
parent = nil
Envoyer s à tous les voisins
echos_reçus = []
sonde_reçue = true
echos_attendus = taille de voisins
Vérifier nombre d'échos reçus
```

Nouveau

# Pseudocode

Réception d'une sonde `s` de `i`

```
Si sonde_reçue
    echos_attendus -= 1                    | Nouveau
Sinon
    sonde_reçue = true                     | Nouveau
    parent = i
    echos_reçus = []
    echos_attendus = taille de voisins – 1 | Nouveau
    s := receptionSonde(s)
    Envoyer s à tous les voisins sauf i
Vérifier nombre d'échos reçus
```

# Pseudocode

Réception d'un écho `e` de `i`

Ajouter `e` à `echos_reçus`

*Vérifier nombre d'échos reçus*

Vérification du nombre d'échos reçus

Si `echos_reçus` a une taille de `echos_attendus`

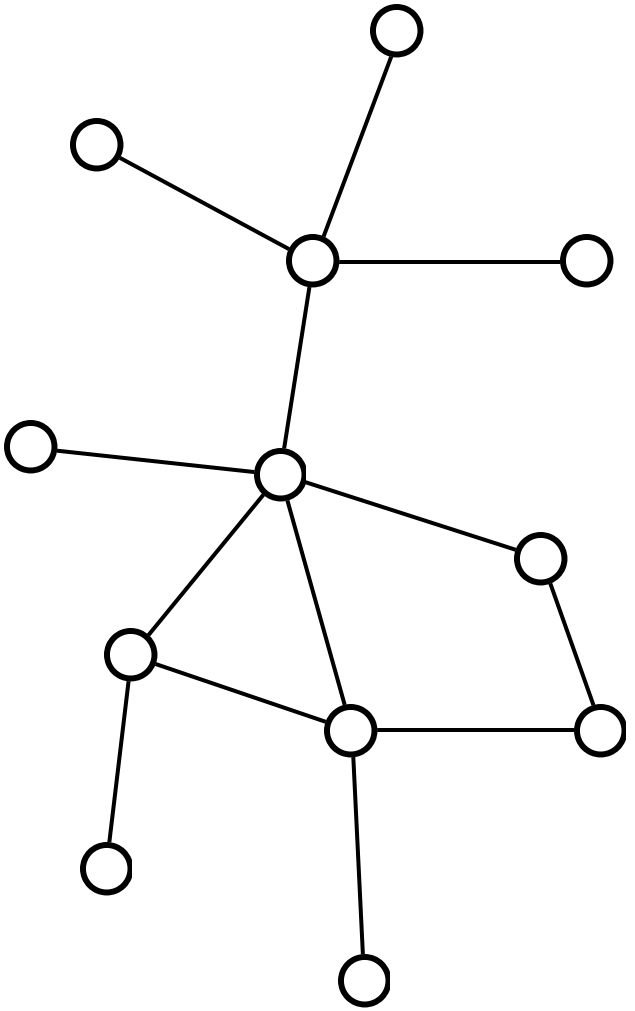
```
    sonde_reçue = false
    echo := agrégerEchos(echos_reçus)
    Si parent est nil
        Terminer le sondage avec echo
    Sinon
        Envoyer echo à parent
```

# Avec plusieurs sources

Sur graphe arbitraire

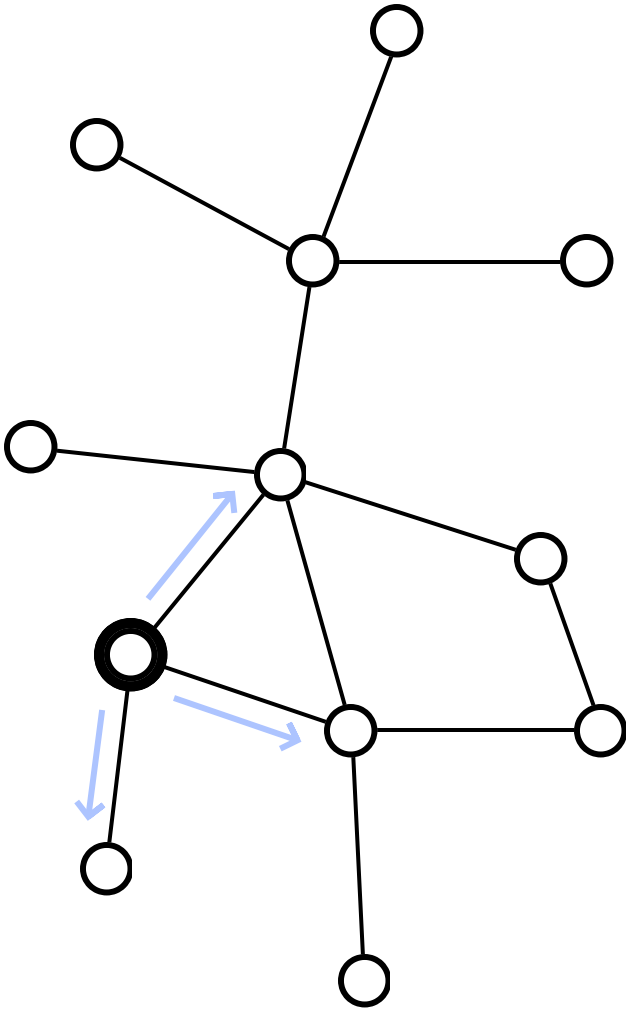
# Cas général avec plusieurs sources

Problème



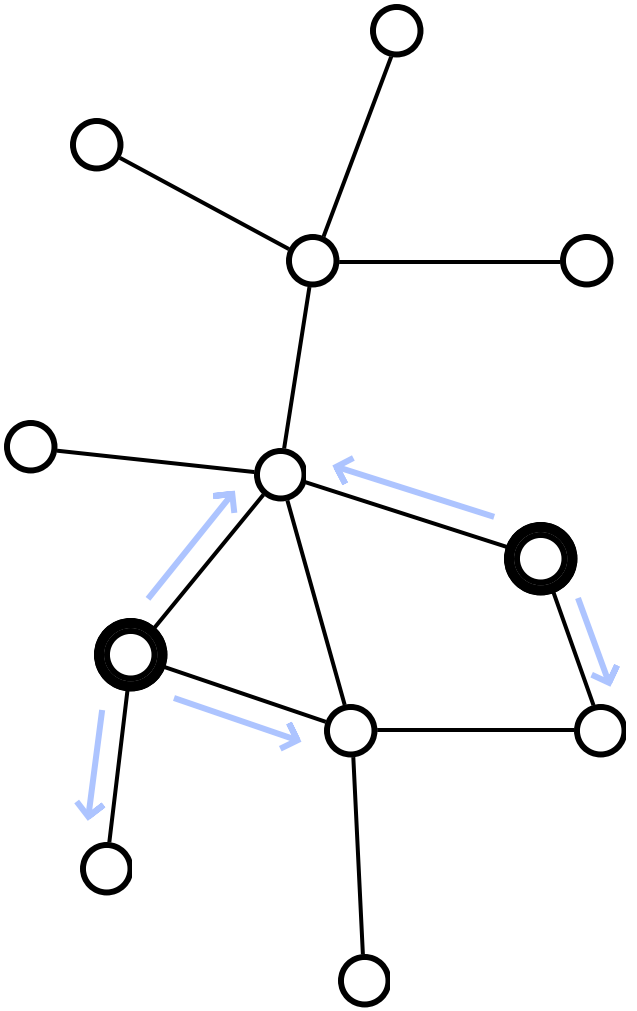
# Cas général avec plusieurs sources

## Problème



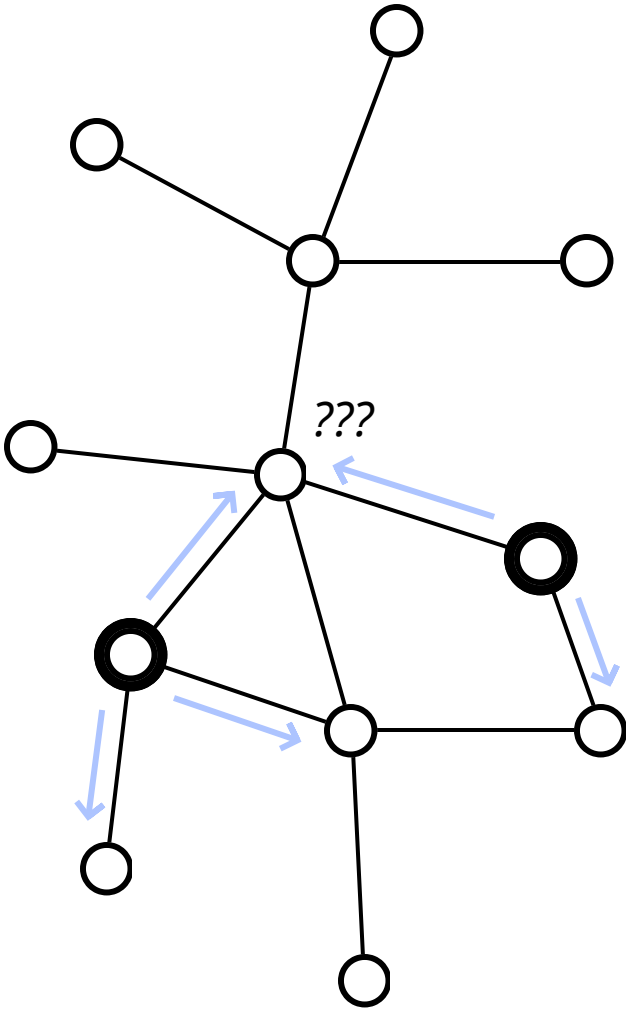
# Cas général avec plusieurs sources

## Problème



# Cas général avec plusieurs sources

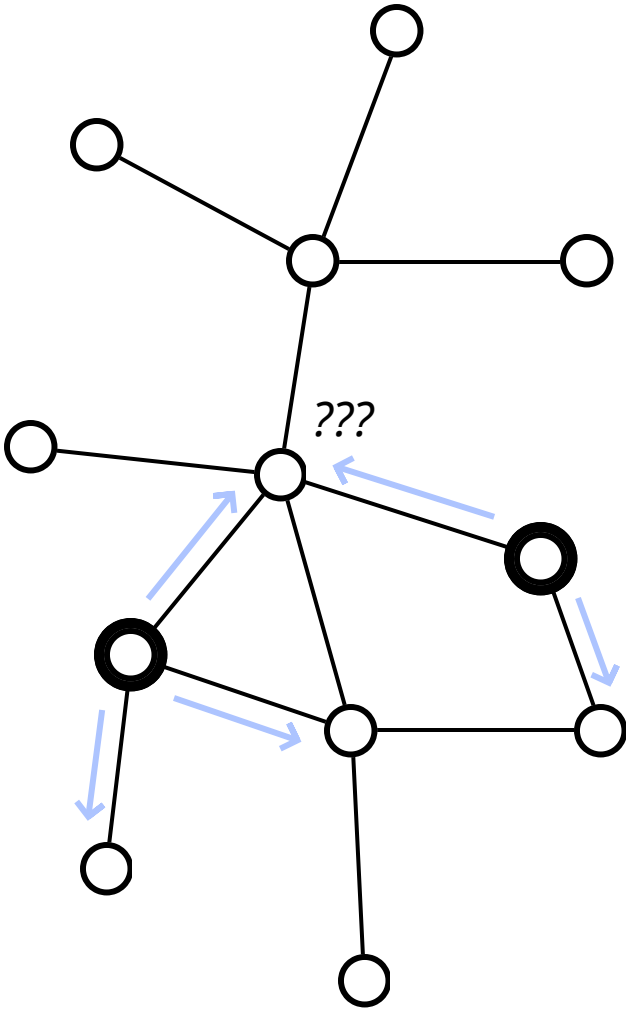
## Problème



# Cas général avec plusieurs sources

## Problème

Réception de deux sondes différentes.

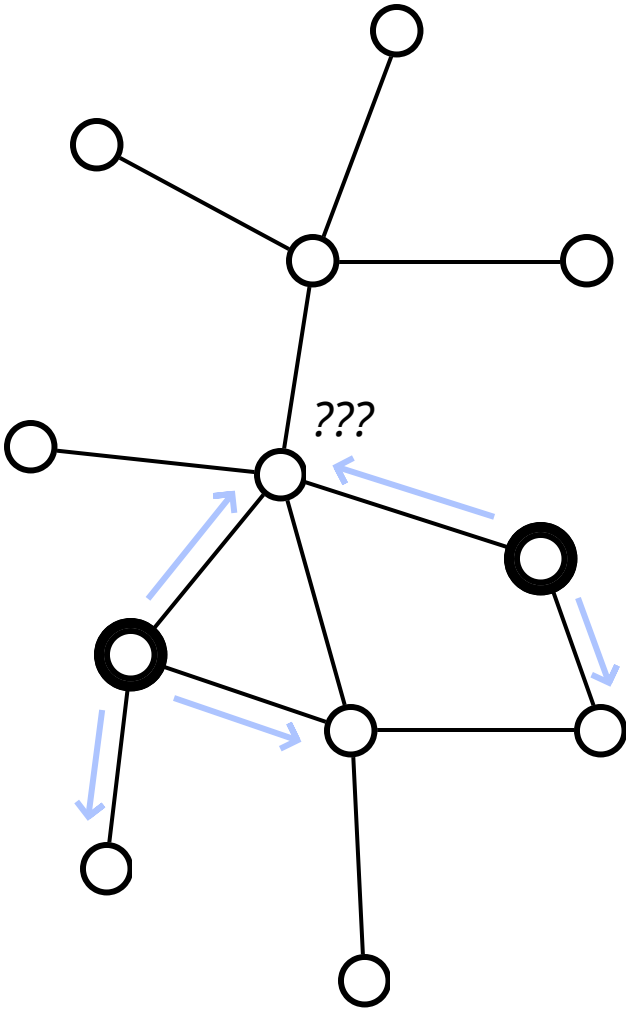


# Cas général avec plusieurs sources

## Problème

Réception de deux sondes différentes.

**Que faire ?**



# Cas général avec plusieurs sources

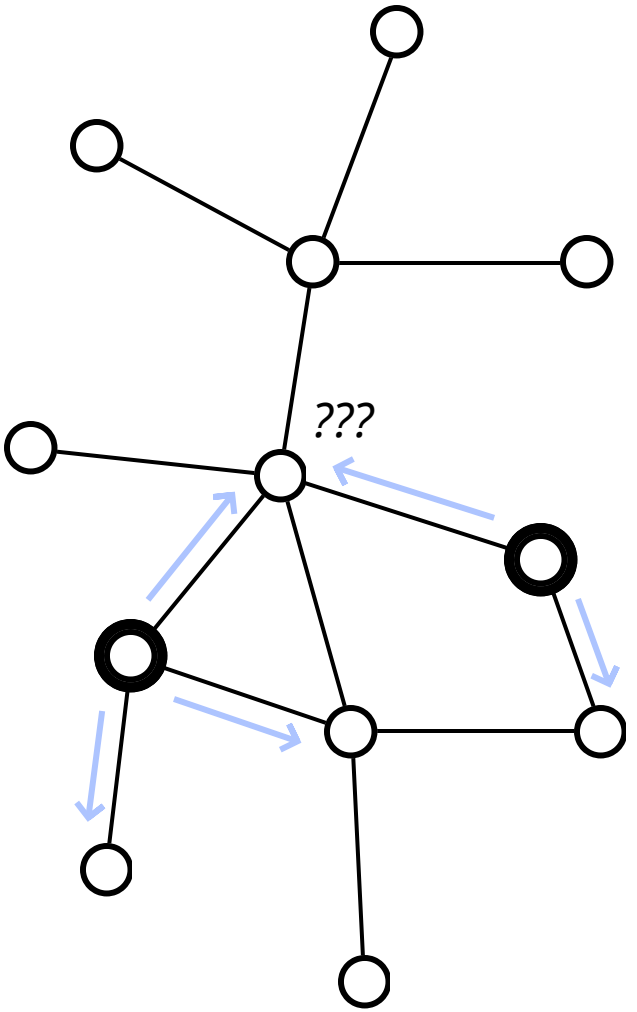
## Problème

Réception de deux sondes différentes.

**Que faire ?**

Marquer la sonde d'un id globalement unique.

Les traiter indépendamment.



# Avec plusieurs sources

Pseudocode

# Parenthèse - id globalement unique

# Parenthèse - id globalement unique

**Comment générer localement un id globalement unique ?**

# Parenthèse - id globalement unique

**Comment générer localement un id globalement unique ?**

Générer un id localement unique (e.g. un compteur)

```
count
```

# Parenthèse - id globalement unique

**Comment générer localement un id globalement unique ?**

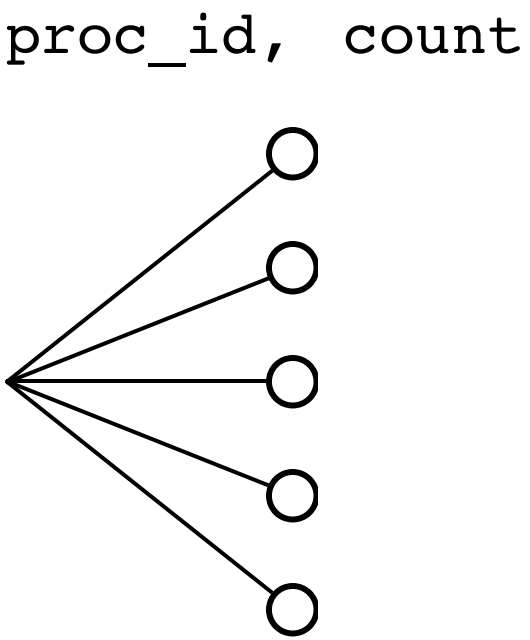
Générer un id localement unique (e.g. un compteur)
Y préfixer un id globalement unique du processus.

```
proc_id, count
```

# Parenthèse - id globalement unique

**Comment générer localement un id globalement unique ?**

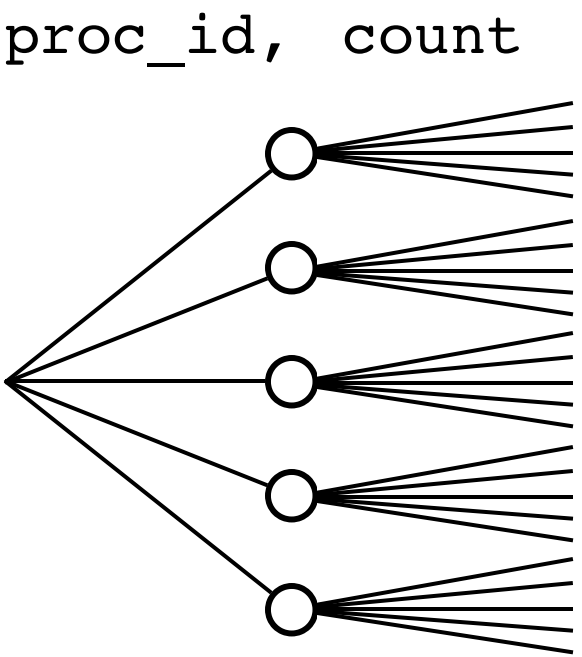
Générer un id localement unique (e.g. un compteur)
Y préfixer un id globalement unique du processus.



# Parenthèse - id globalement unique

**Comment générer localement un id globalement unique ?**

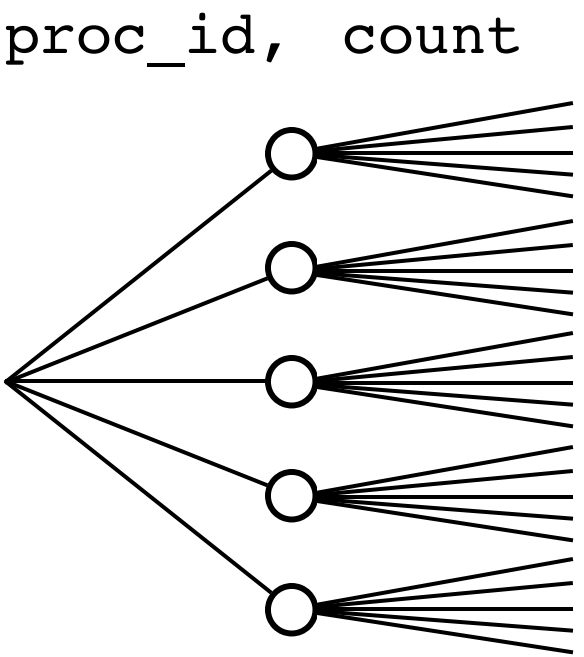
Générer un id localement unique (e.g. un compteur)
Y préfixer un id globalement unique du processus.



# Parenthèse - id globalement unique

**Comment générer localement un id globalement unique ?**

Générer un id localement unique (e.g. un compteur)
Y préfixer un id globalement unique du processus.




*"`proc_id` est la dizaine*
*`count` est l'unité "*

# Pseudocode

Variables

| | | |
|---|---|---|
| `parent` | *dict d'entiers* | id du parent pour chaque id de sonde. |
| `voisins` | *liste d'entiers* | ids de mes processus voisins |
| `echos_reçus` | *dict de listes d'échos* | échos reçus par id de sonde |
| `echos_attendus` | *dict d'entiers* | nombre d'échos attendus par id de sonde |
| `dernier_id` | *dict d'entiers* | dernier sondage terminé, par source |
| `receptionSonde` | *callback* | à appeler à la réception d'une sonde |
| `agrégerEchos` | *callback* | à appeler une fois tous les échos reçus |

*`sonde_reçue` est remplacé par la présence d'une clé dans `echos_attendus`*

# Pseudocode

Demande d'envoi d'une sonde `s`

Générer un nouveau `s_id`

`parent[s_id] = nil`

Envoyer `s` à tous les `voisins`

`echos_reçus[s_id] = []`

`echos_attendus[s_id]` = taille de `voisins`

*Vérifier nombre d'échos reçus pour `s_id`*

# Pseudocode

Réception d'une sonde `s` avec id `s_id` de `i`

Si `s_id` ≤ `dernier_id[proc_id de s_id]`
    Ignorer la sonde
Si `s_id` est dans `echos_attendus`
    `echos_attendus[s_id] -= 1`
Sinon
    `echos_attendus[s_id] =` taille de `voisins` – 1
    `echos_reçus[s_id] = []`
    `parent[s_id] = i`
    `s := receptionSonde(s)`
    Envoyer `s` et `s_id` à tous les `voisins` sauf `i`
*Vérifier nombre d'échos reçus pour `s_id`*

# Pseudocode

Réception d'un écho `e` avec id `s_id` de `i`

Ajouter `e` à `echos_reçus[s_id]`

*Vérifier nombre d'échos reçus pour `s_id`*

Vérification du nombre d'échos reçus pour `s_id`

Si `echos_reçus[s_id]` a une taille de `echos_attendus[s_id]`

```
    echo := agrégerEchos(echos_reçus[s_id])
    Si parent[s_id] est nil
        Terminer le sondage s_id avec echo
    Sinon
        Envoyer echo avec s_id à parent[s_id]
    dernier_id[proc_id de s_id] = s_id
    Oublier s_id dans echos_attendus, echos_reçus et parent
```

# Pseudocode

Que garder après un sondage terminé ?

**Tout garder ?**

Correct, mais coûteux en mémoire

**Tout oublier ?**

Incorrect.

Une sonde retardataire arrive après la fin du sondage.

Sans trace, elle passe pour neuve et attend indéfiniment des échos.

**Tout oublier, sauf `dernier_id` ?**

Correct

Speaker notes

Les ids d'une même source sont croissants : c'est le schéma (proc_id, compteur) de la parenthèse précédente. Un seul entier par source suffit donc à distinguer une sonde retardataire d'une sonde neuve. Le coût de « tout garder », c'est echos_reçus, qui n'est jamais libéré. Le marqueur, lui, ne libère pas de mémoire : il empêche d'en recréer.

# Exercices

# Exercices

**1**

Écrire un algorithme qui détermine la topologie d'un réseau garanti comme étant un arbre.

Un processus démarre l'algorithme et obtient à la fin un dictionnaire contenant pour chaque processus la liste de ses enfants, si la source était la racine.

**2**

Écrire un algorithme qui détermine un arbre de recouvrement d'un réseau quelconque (donc avec potentiellement des cycles).

Un processus démarre l'algorithme, et à la fin de son exécution, chaque processus contient *uniquement* la liste de ses enfants dans cet arbre de recouvrement. Aucun processus ne connait l'arbre complet.