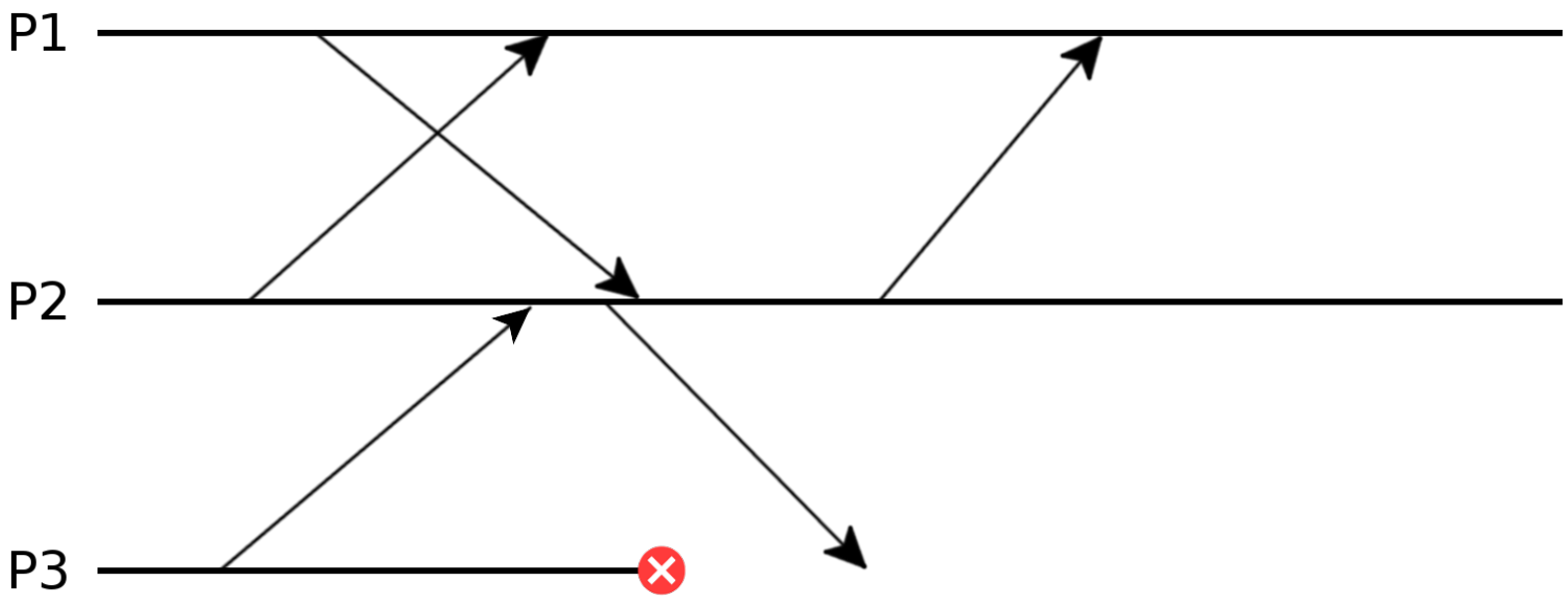
Algorithmique distribuée - MIF12

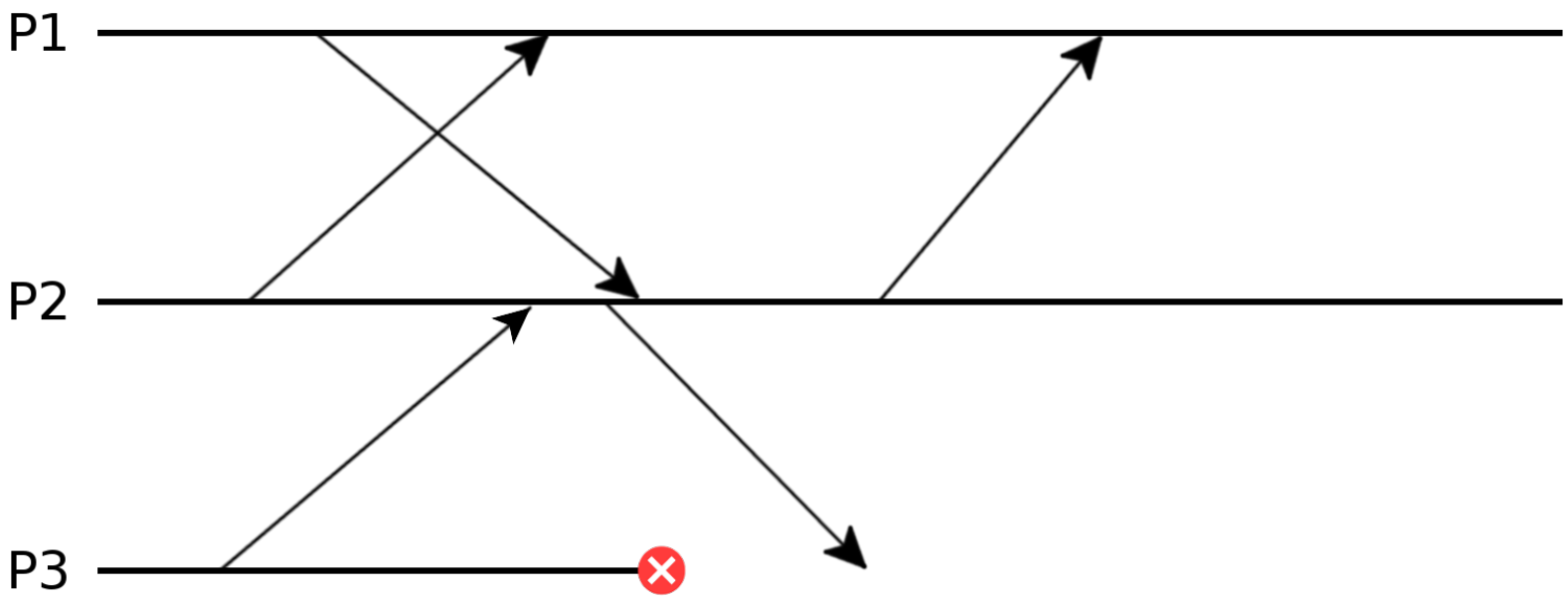
Consensus et détecteurs de fautes

Élise JEANNEAU
elise.jeanneau@univ-lyon1.fr
Adapté du cours de Pierre SENS

Un ou plusieurs processus peuvent tomber en panne.



Un ou plusieurs processus peuvent tomber en panne.



L'algorithme doit fonctionner malgré tout !

- Pannes franches (crash)

Modèles de fautes

- Pannes franches (crash)

- Pannes temporaires

Modèles de fautes

- Pannes franches (crash)

- Pannes temporaires

- Fautes d'omission
  - Pertes de messages

Modèles de fautes

- Pannes franches (crash)

- Pannes temporaires

- Fautes d'omission

  Pertes de messages

- Fautes Byzantines

  Processus malveillant, agit contre l'algorithme

Processus corrects et fautifs

Dans ce cours : pannes franches uniquement.

Dans ce cours : pannes franches uniquement.

Deux types de processus :

- Processus **correct**

P1 ——————————

Dans ce cours : pannes franches uniquement.

Deux types de processus :

- Processus **correct**

  P1 ——————————————

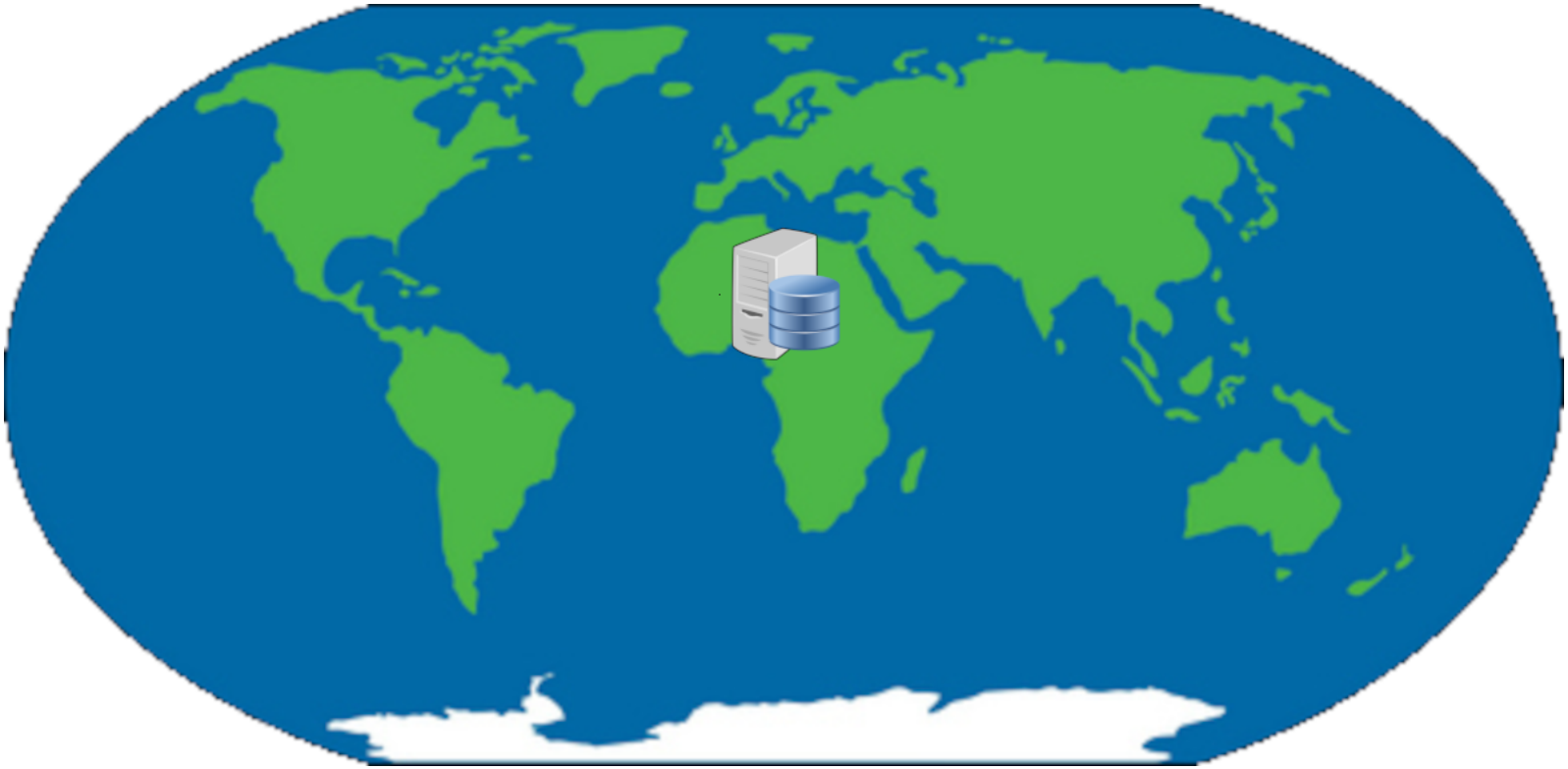
- Processus **fautif**

  P1 —————— 

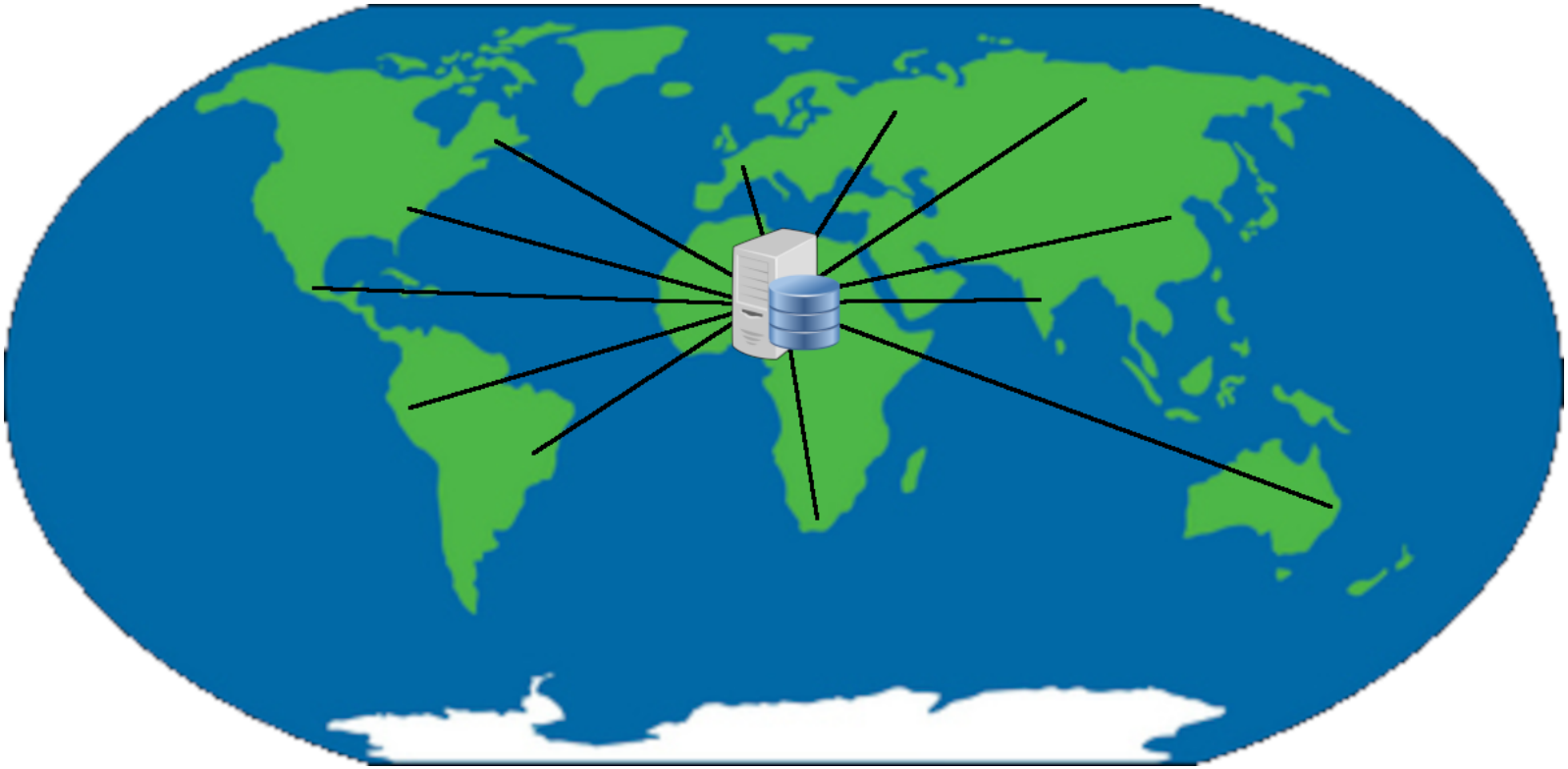
- Un problème central : le **consensus**.
- Plusieurs processus doivent se mettre d'accord.

- Un problème central : le **consensus**.
- Plusieurs processus doivent se mettre d'accord.
- Pourquoi faire ?

Modèle centralisé

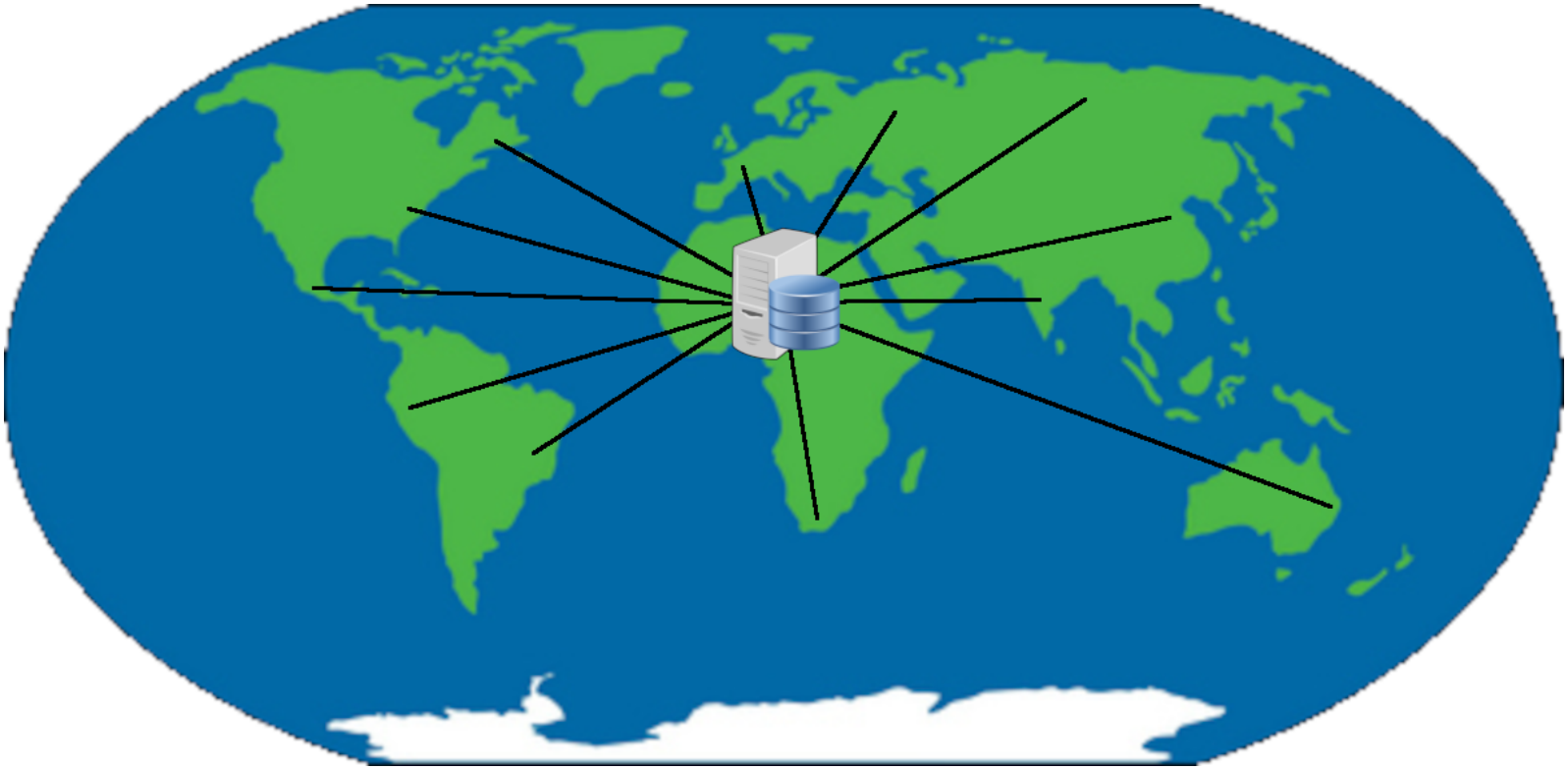


Modèle centralisé



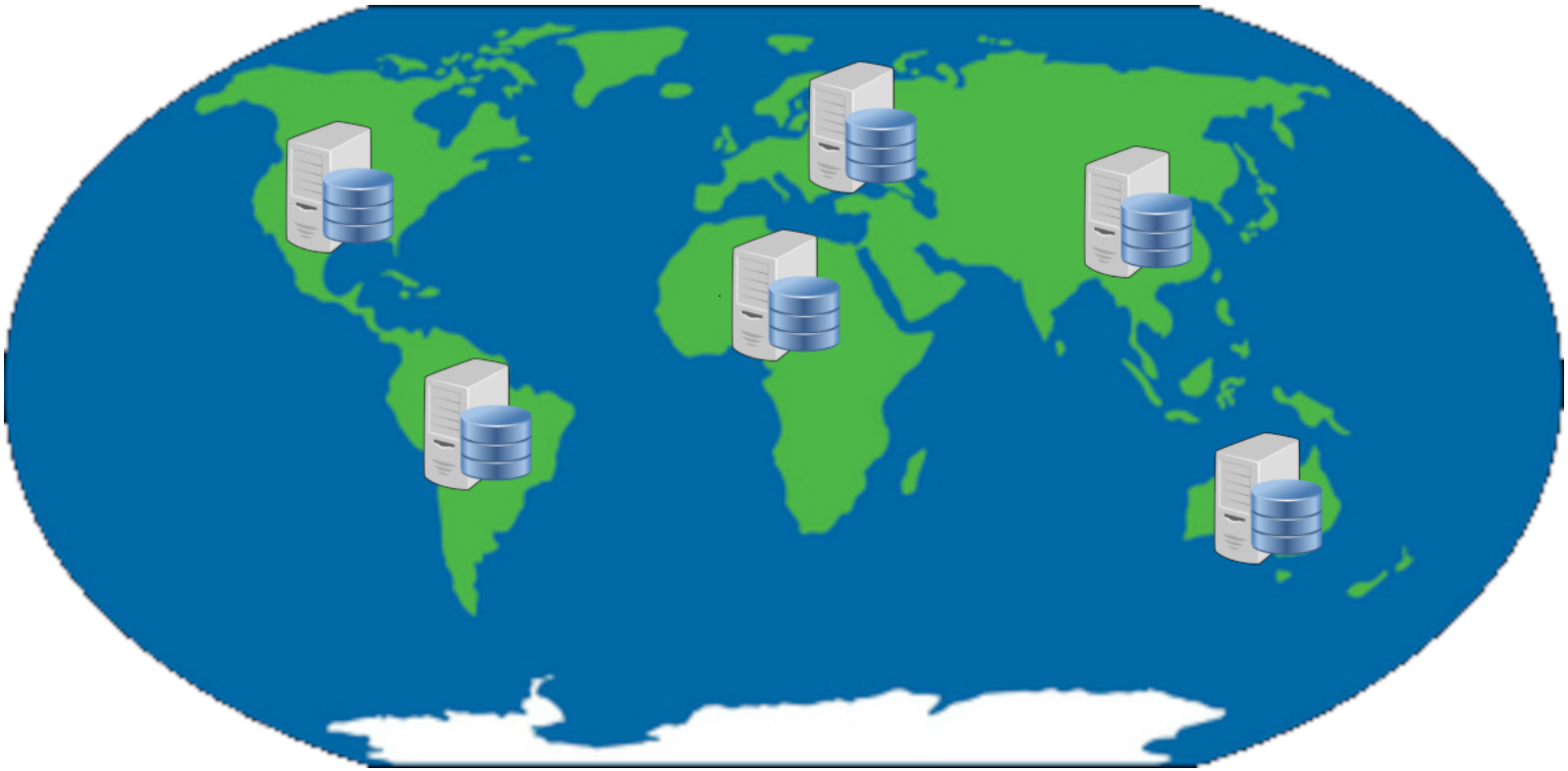
Exemple applicatif : base de données à large échelle

Modèle centralisé



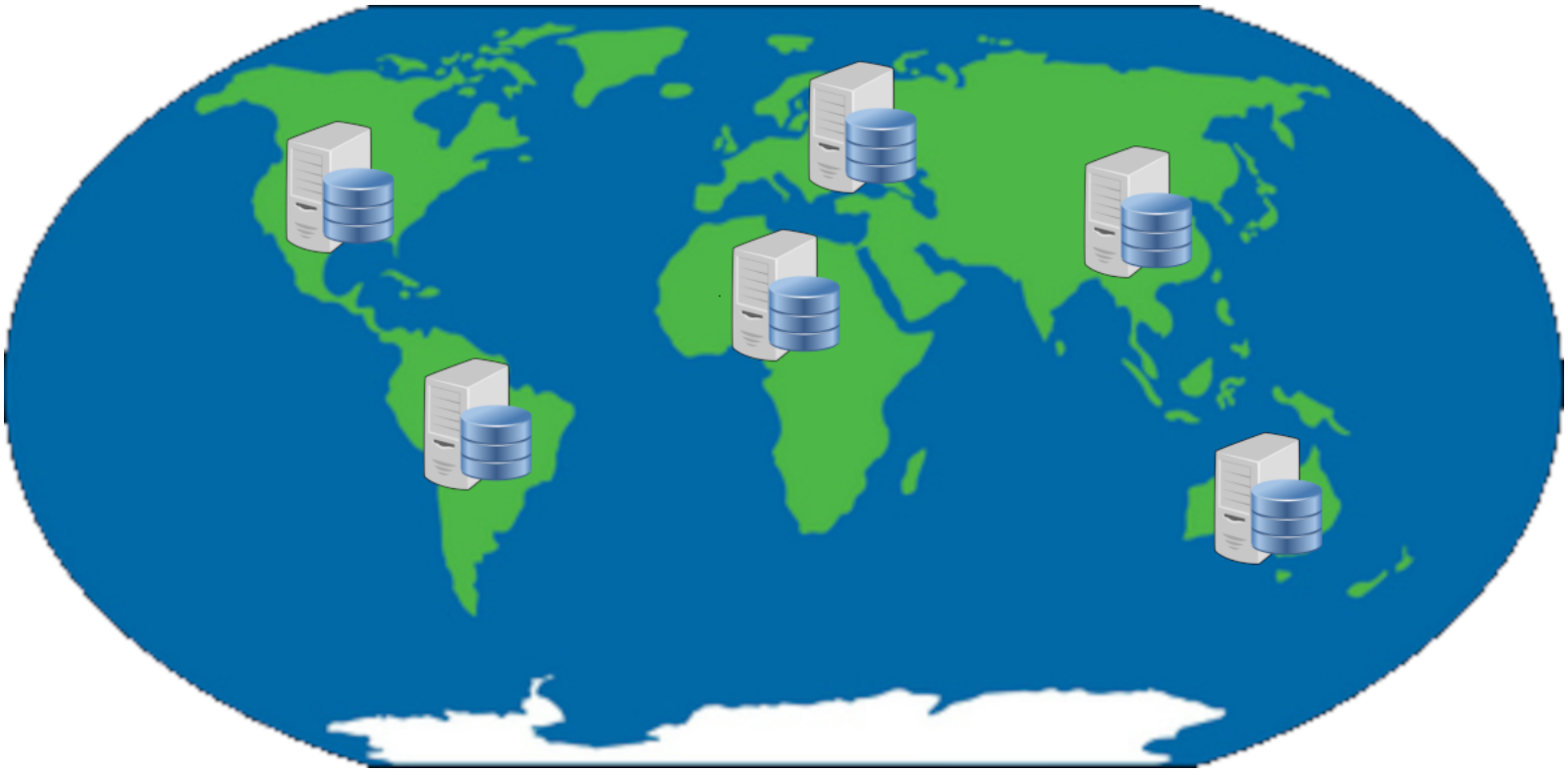
Problèmes : latence, congestion réseau, vulnérabilité à 1 panne...

Modèle distribué : base de données répartie

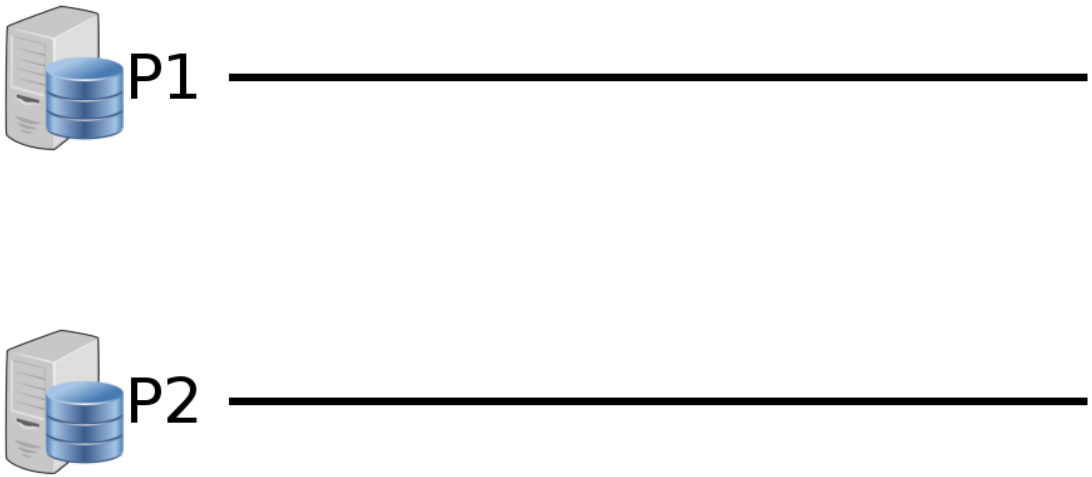


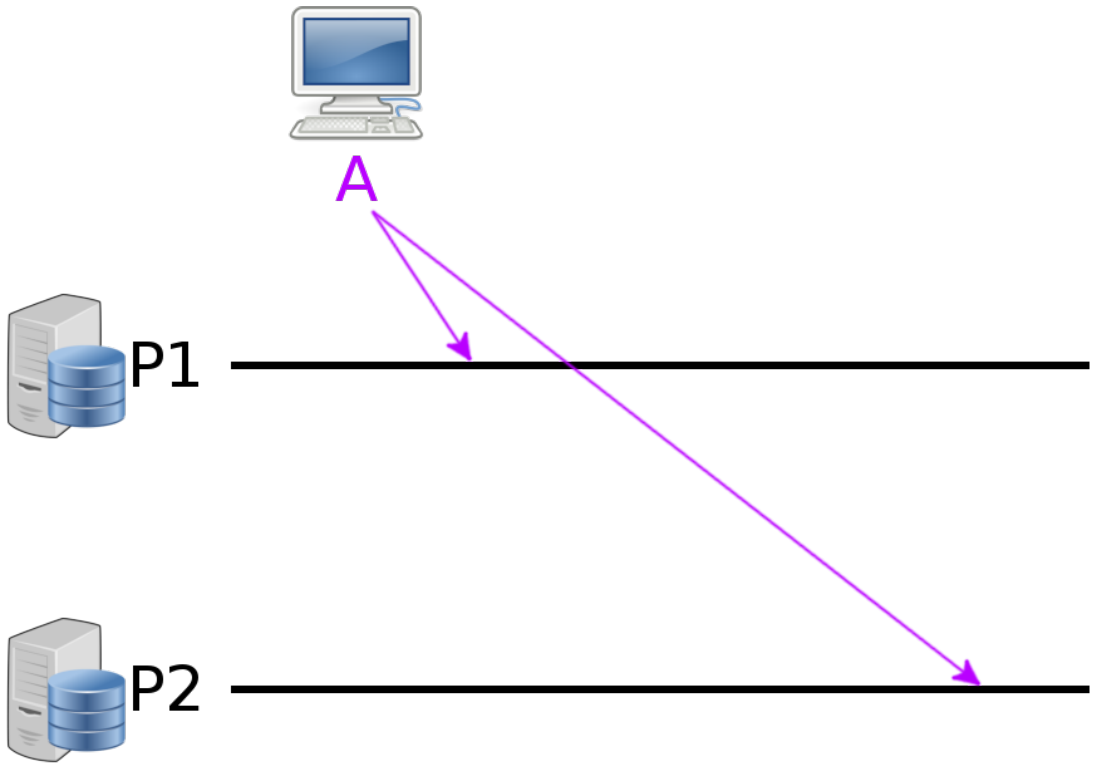
Exemple applicatif : base de données à large échelle

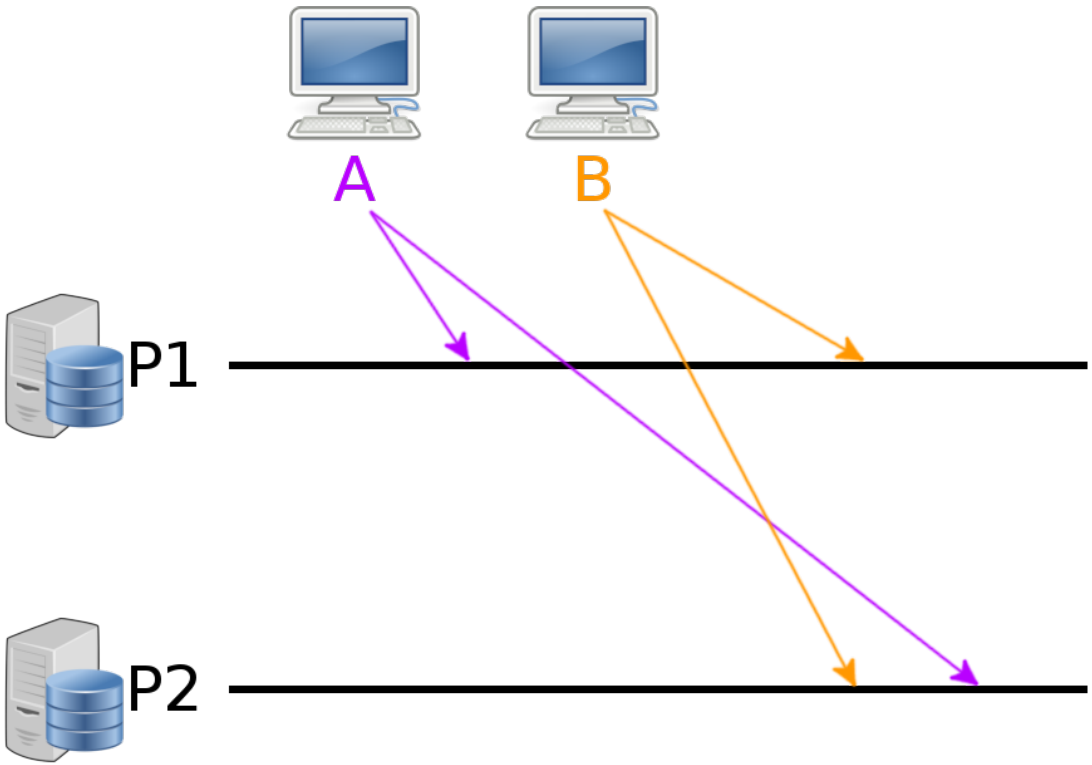
Modèle distribué : base de données répartie

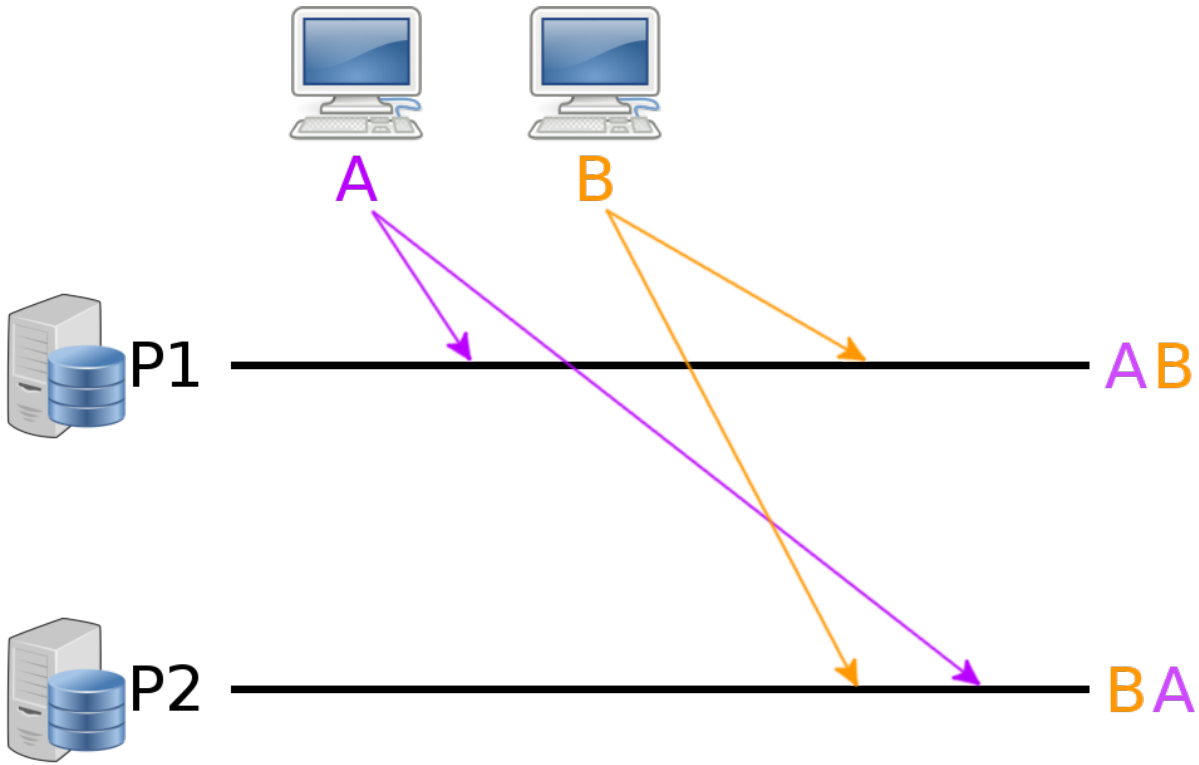


Problème : **cohérence des données**

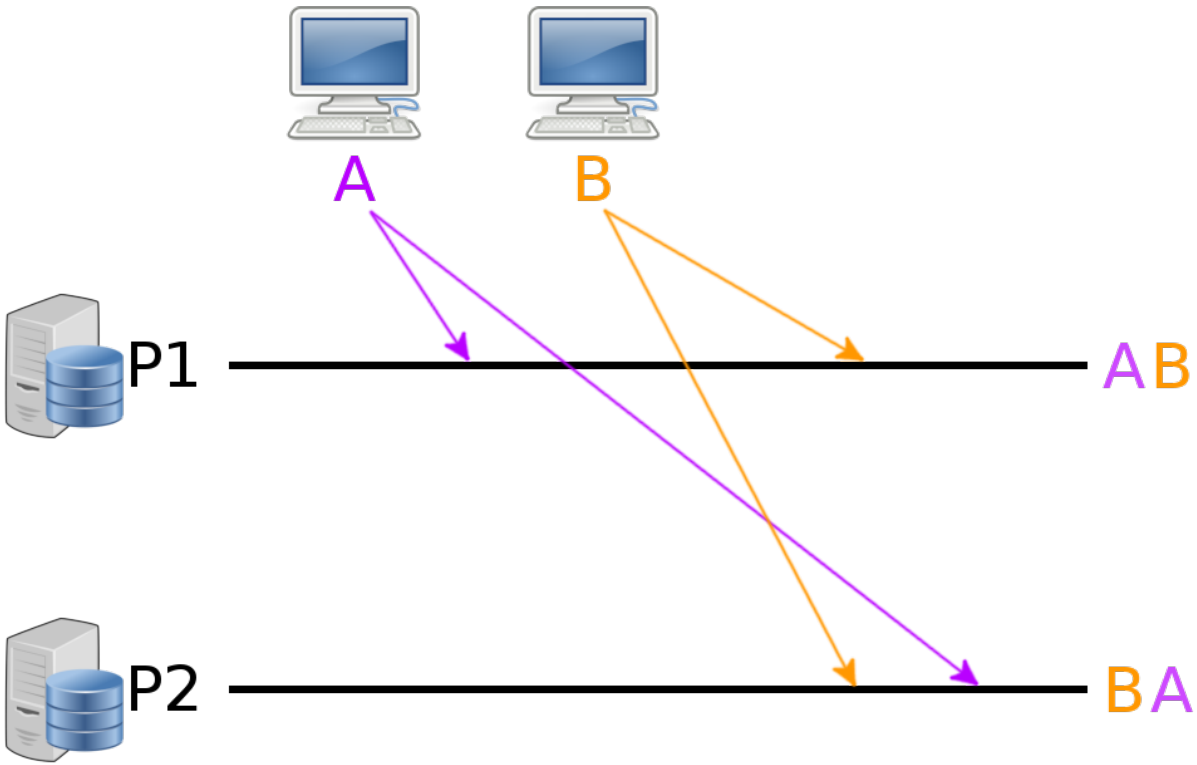

P1
P2


A
P1
P2


A
B
P1
P2

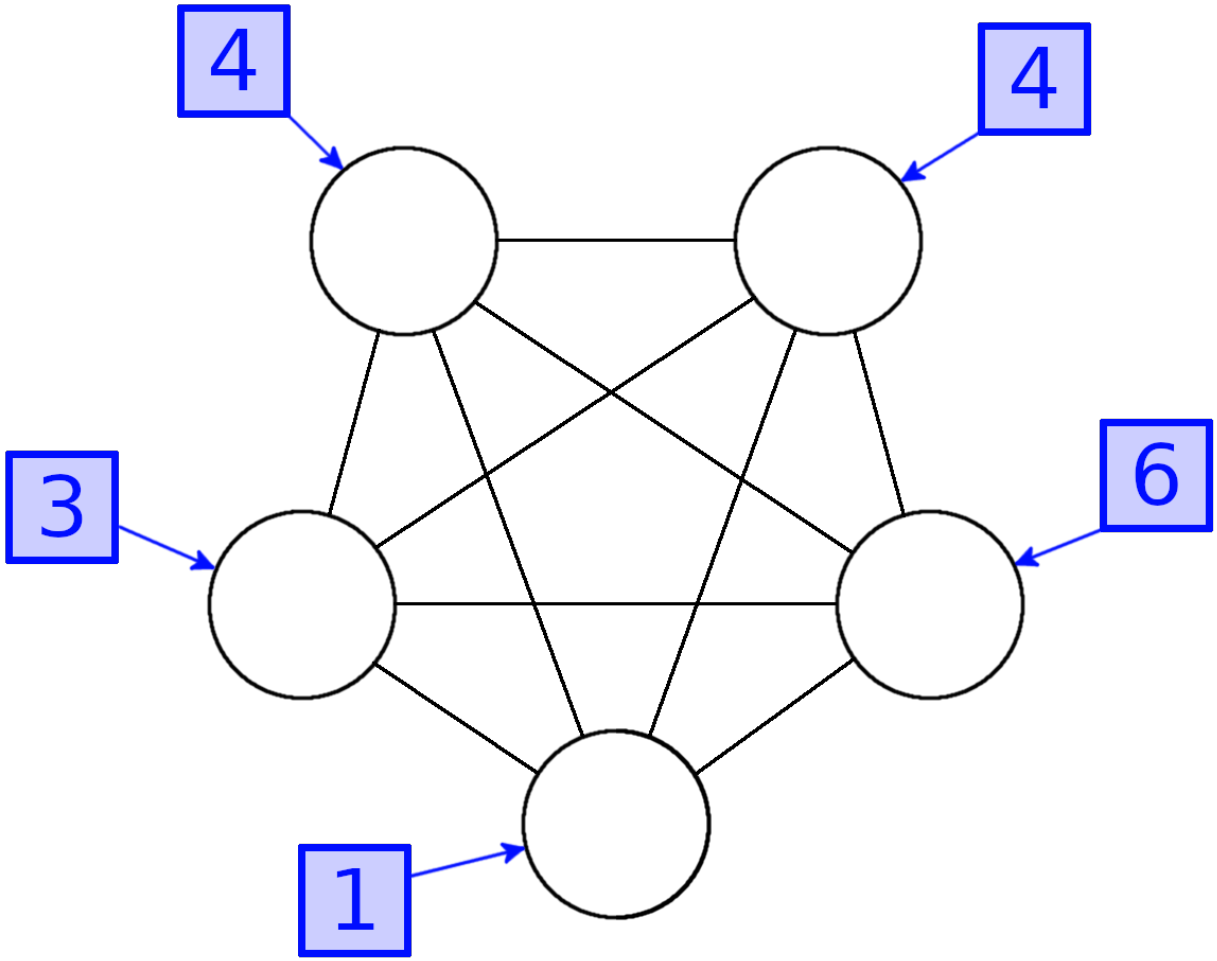


Désaccord entre P1 et P2 sur l'état final.

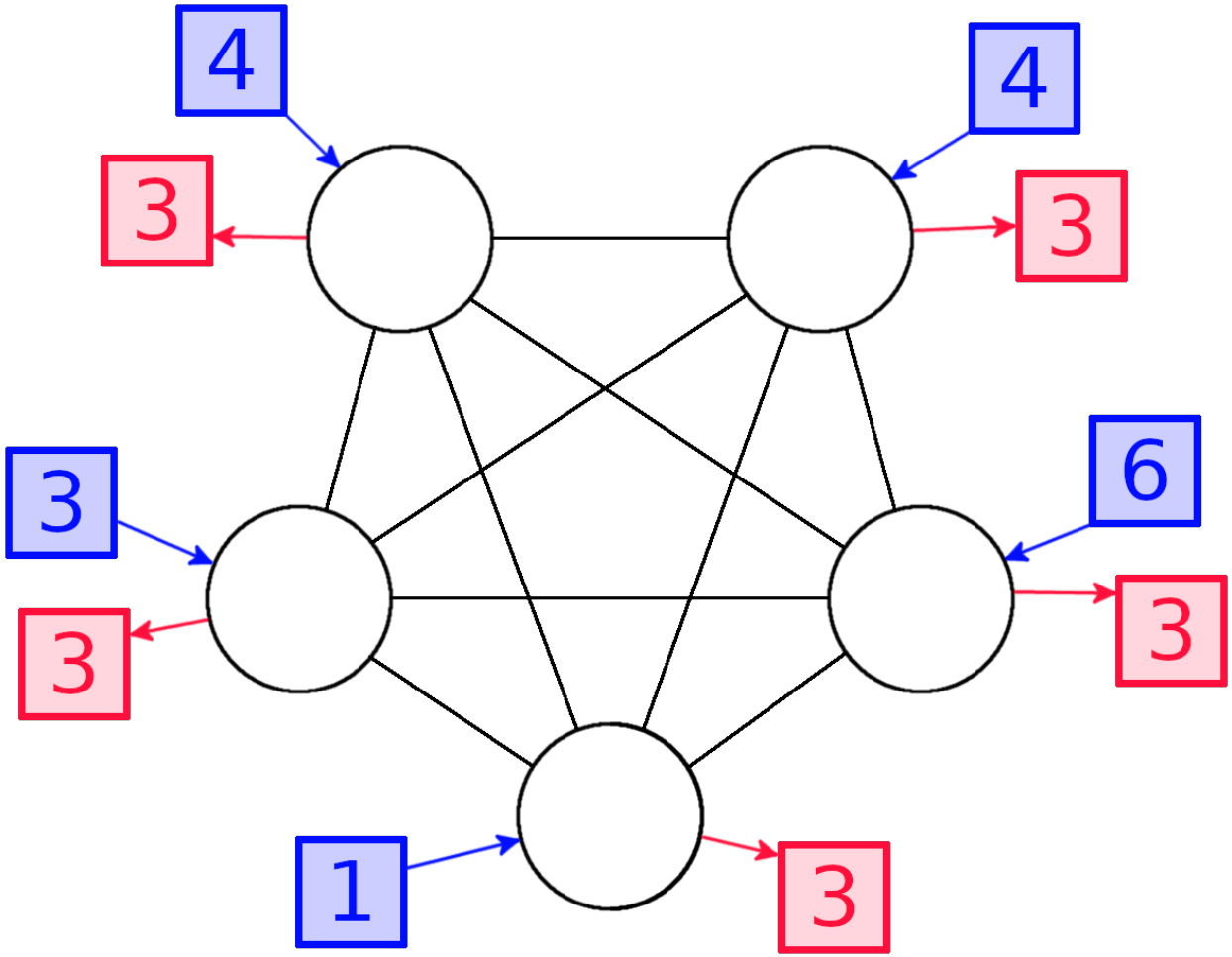


Désaccord entre P1 et P2 sur l'état final.

$\Longrightarrow$ Nécessité d'un **consensus**.



Chaque processus **propose** une valeur.



Chaque processus **propose** une valeur.

Chaque processus **décide** une valeur.

Deux primitives :

- **propose(v)**
  Appelée par l'application

Deux primitives :

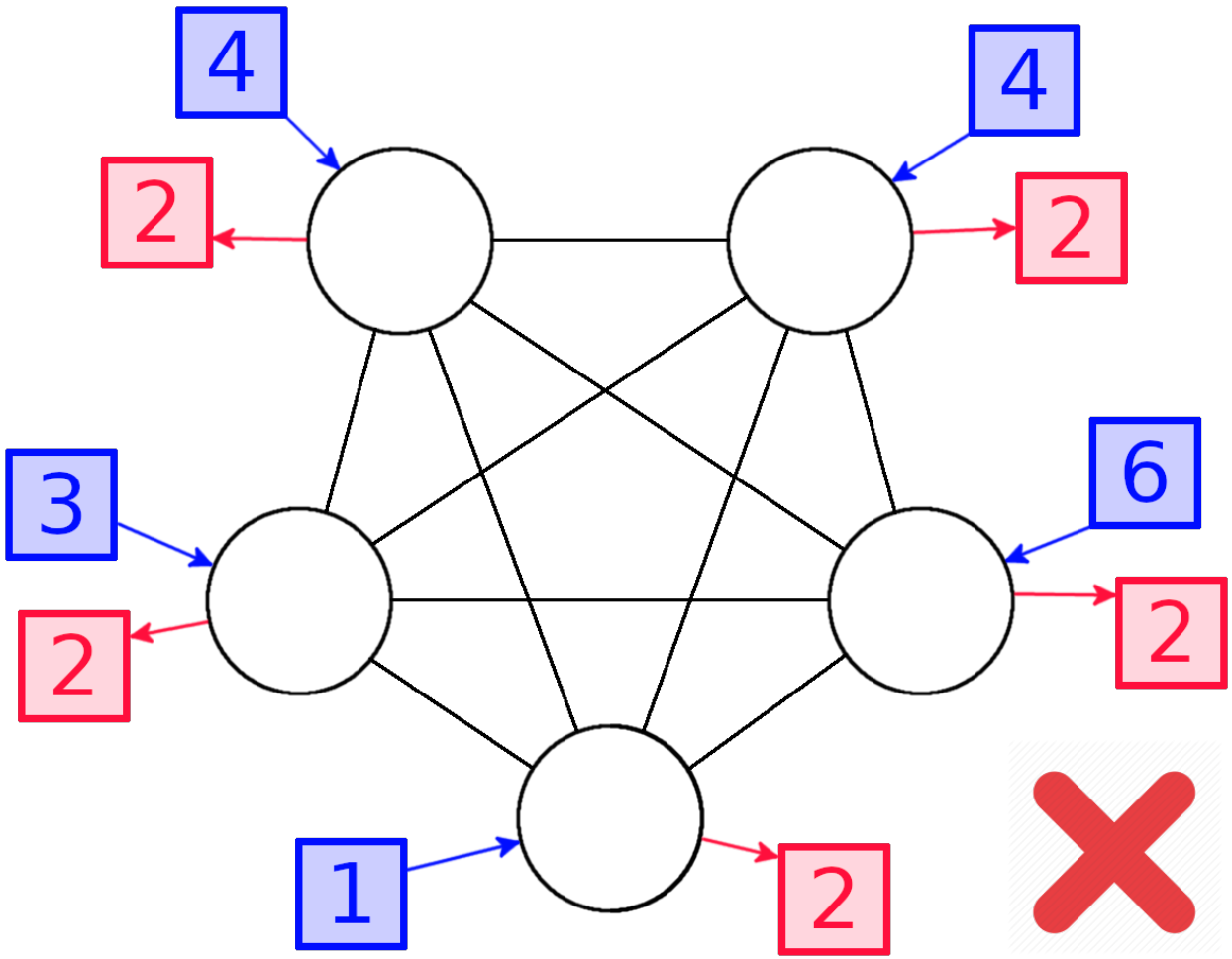
- propose(v)
  Appelée par l'application
- decide(v)
  Appelée par l'algorithme de consensus

- **Validité**
  La valeur décidée doit être l'une des valeurs proposées.

- **Validité**
  La valeur décidée doit être l'une des valeurs proposées.
- **Terminaison**
  Tous les processus corrects décident en un temps fini.

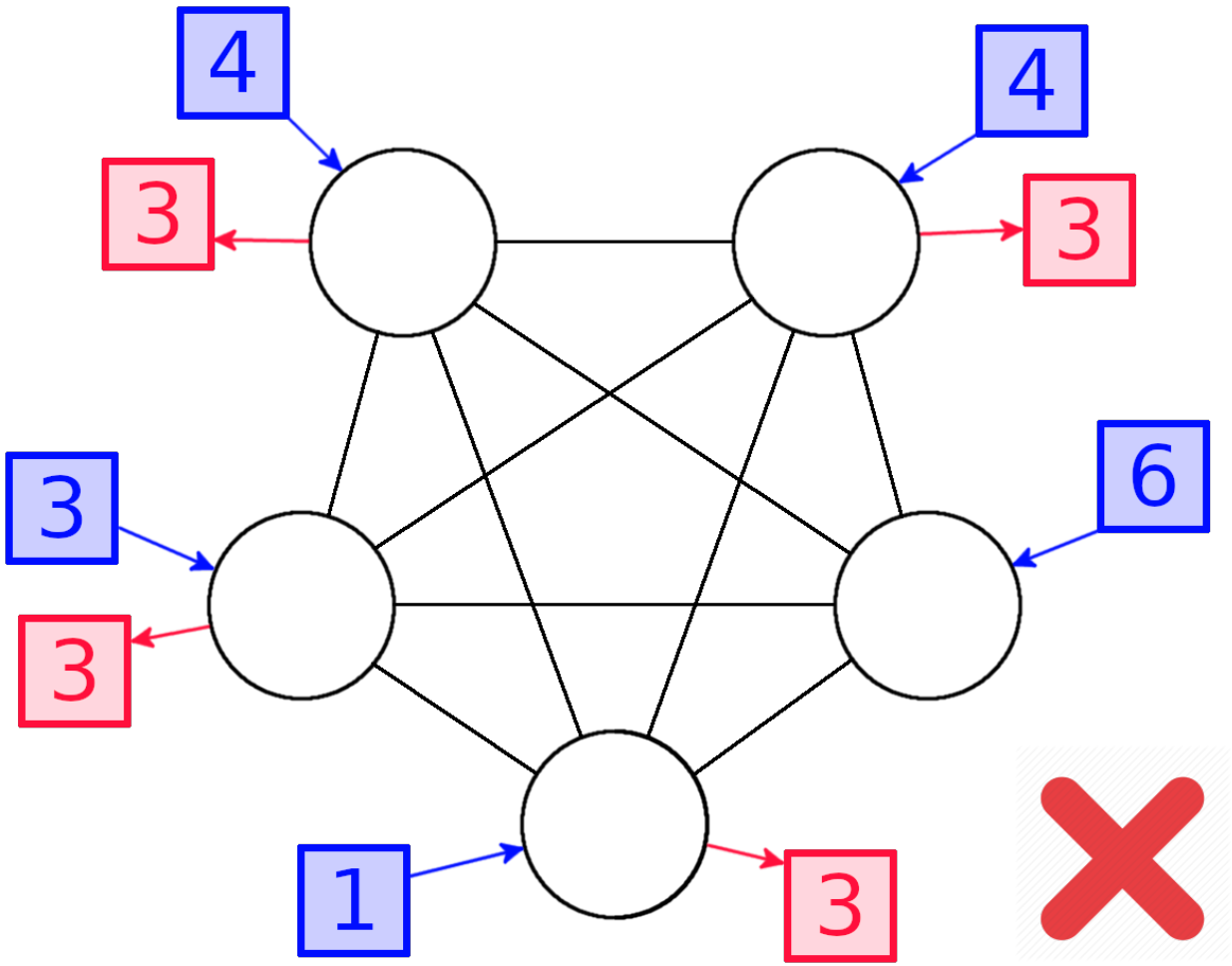
- **Validité**
  La valeur décidée doit être l'une des valeurs proposées.
- **Terminaison**
  Tous les processus corrects décident en un temps fini.
- **Cohérence**
  Deux processus corrects ne peuvent décider différemment.

- **Validité**
  La valeur décidée doit être l'une des valeurs proposées.
- **Terminaison**
  Tous les processus corrects décident en un temps fini.
- **Cohérence**
  Deux processus corrects ne peuvent décider différemment.
- **Intégrité**
  Un processus doit décider au plus une fois.



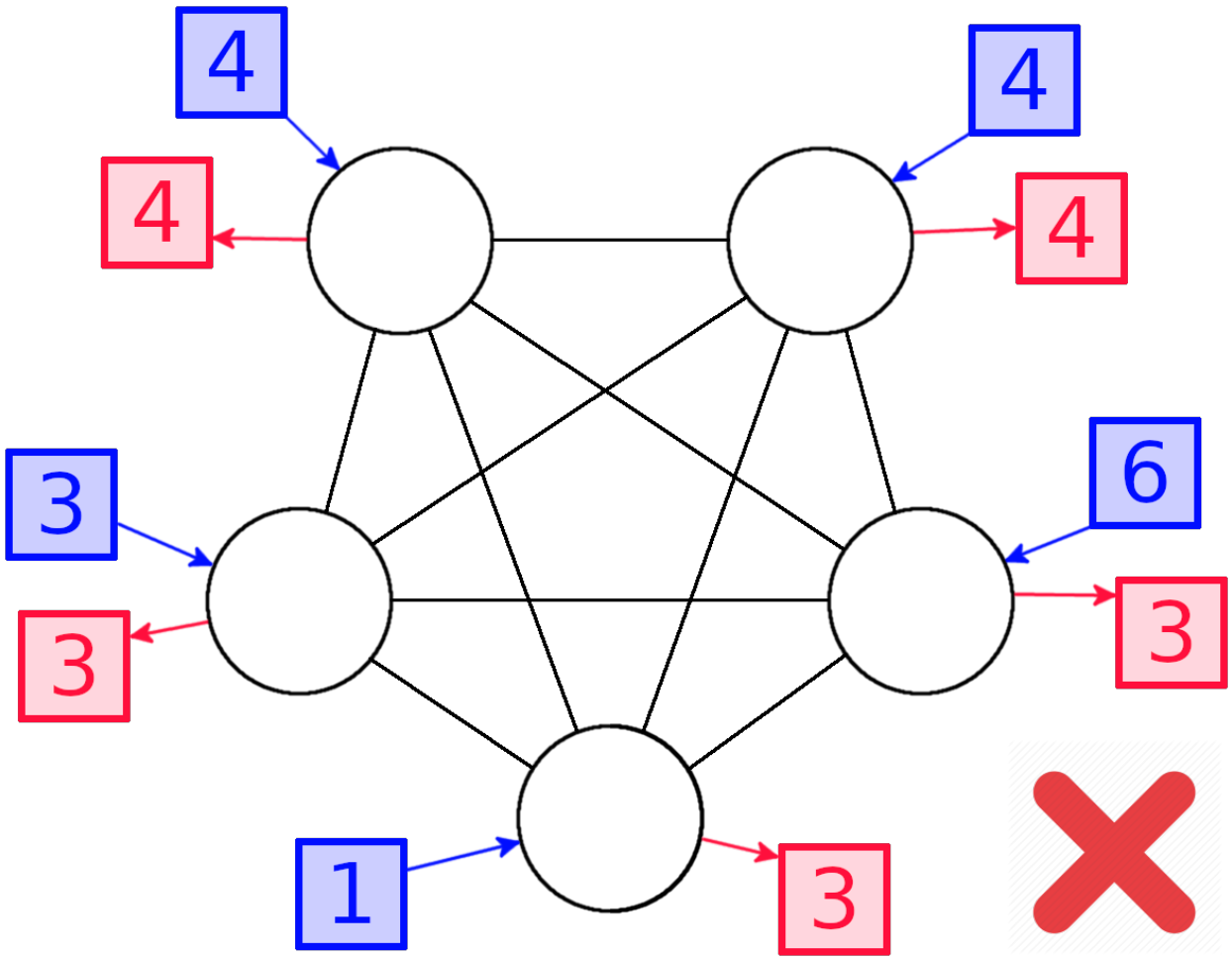
Validité

La valeur décidée doit être l'une des valeurs proposées.



Terminaison

Tous les processus corrects décident en un temps fini.



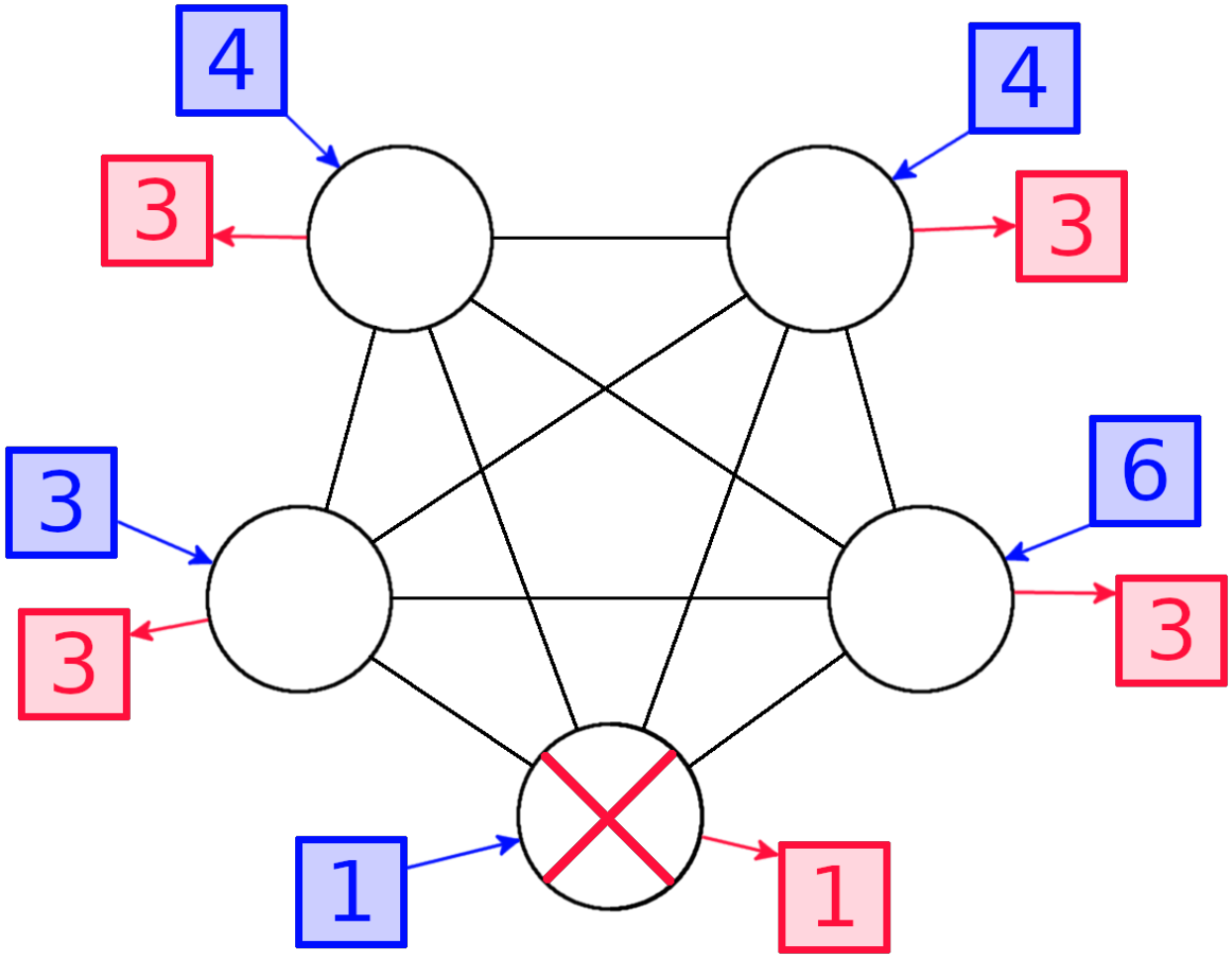
Cohérence

Deux processus corrects ne peuvent décider différemment.

Version plus difficile du consensus.

- **Validité**
  La valeur décidée doit être l'une des valeurs proposées.
- **Terminaison**
  Tous les processus corrects décident en un temps fini.
- **Cohérence uniforme**
  Deux processus, **même fautifs**, ne peuvent décider différemment.
- **Intégrité**
  Un processus doit décider au plus une fois.

Violation de la propriété de cohérence uniforme



La **cohérence** est vérifiée.

La **cohérence uniforme** n'est **pas** vérifiée.

Hypothèses de ce cours

- Graphe **complet**
- Liens de communication bidirectionnels
- n noeuds **statiques**
- Chaque noeud a un **identifiant unique**
- **Messages toujours correctement reçus**
- **Pannes franches**

Algorithme de consensus naïf

1. Envoyer valeur proposée à tous

Algorithme de consensus naïf

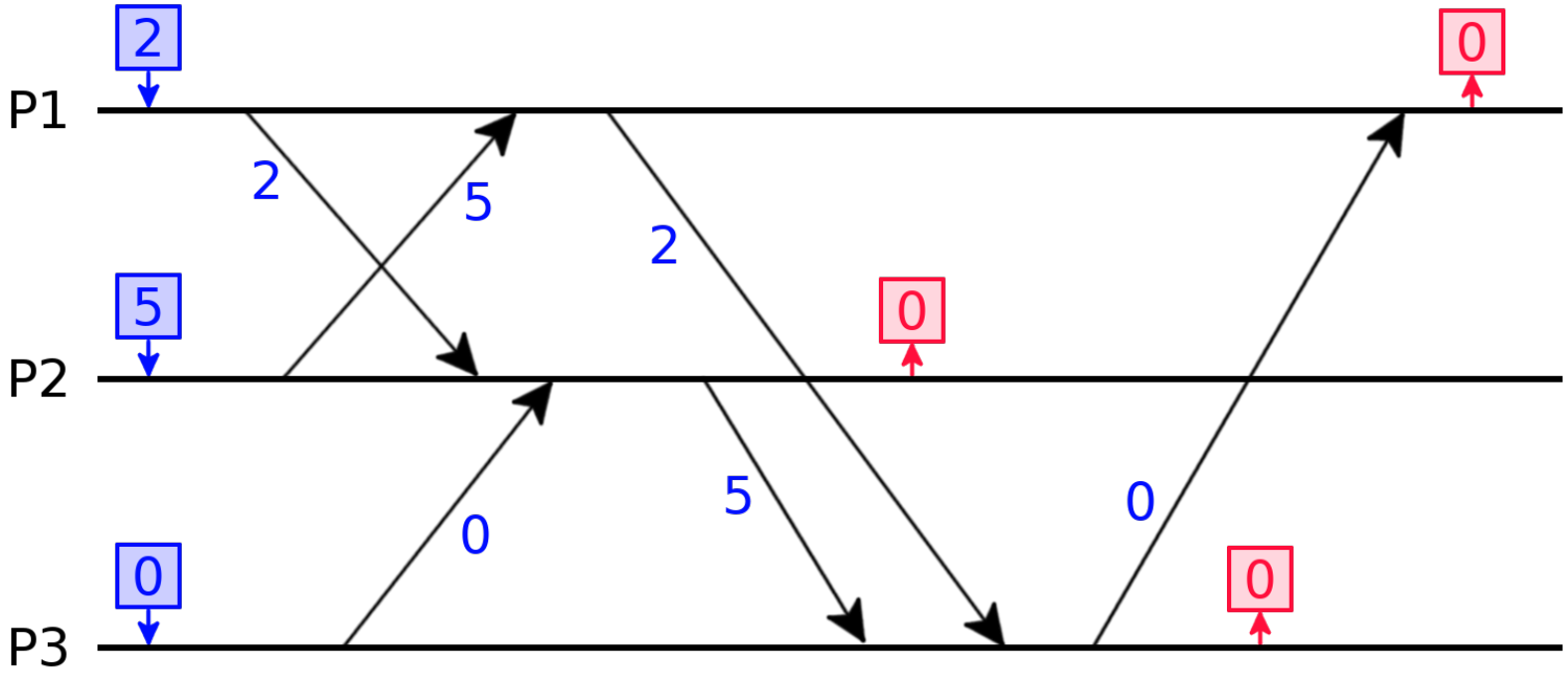
1. Envoyer valeur proposée à tous
2. Attendre de recevoir toutes les valeurs

Algorithme de consensus naïf

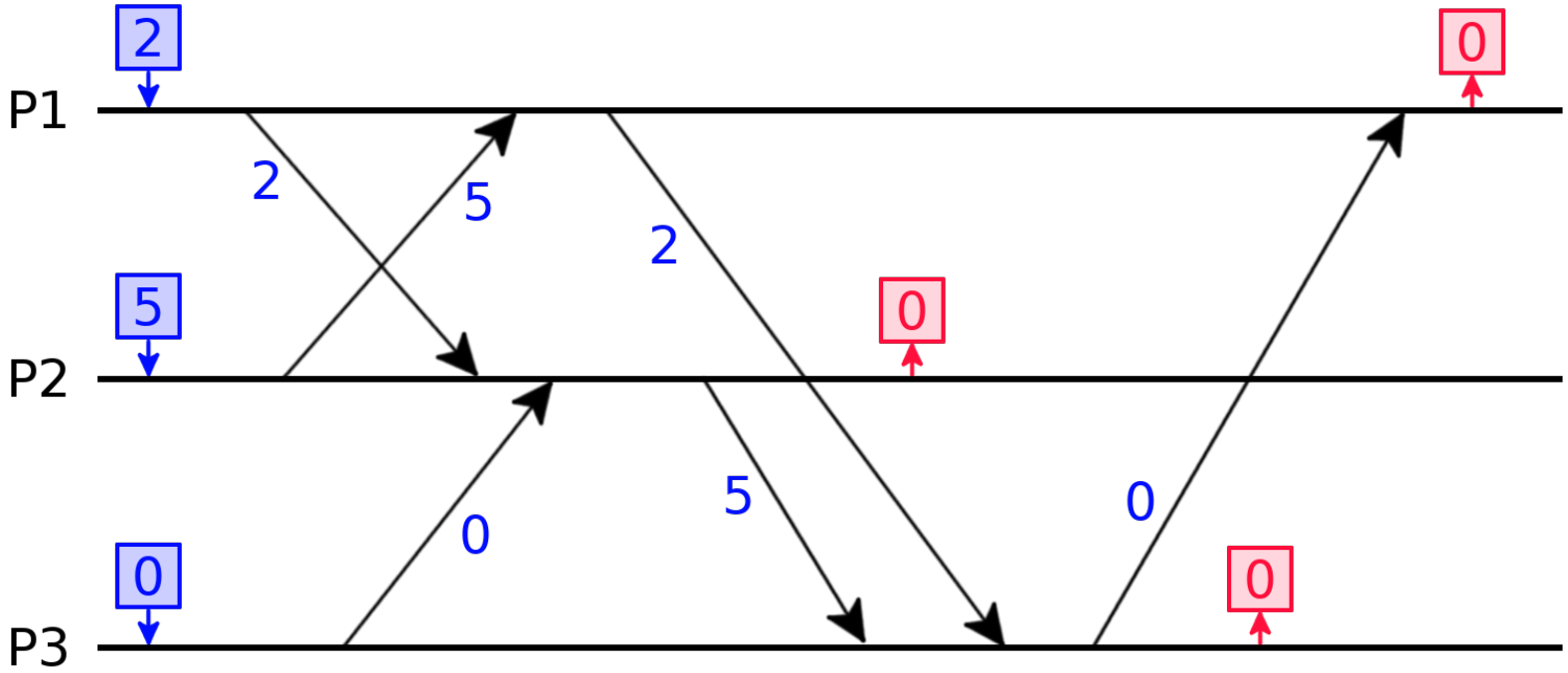
1. Envoyer valeur proposée à tous
2. Attendre de recevoir toutes les valeurs
3. Décider la valeur la plus faible

Algorithme de consensus naïf

1. Envoyer valeur proposée à tous
2. Attendre de recevoir toutes les valeurs
3. Décider la valeur la plus faible

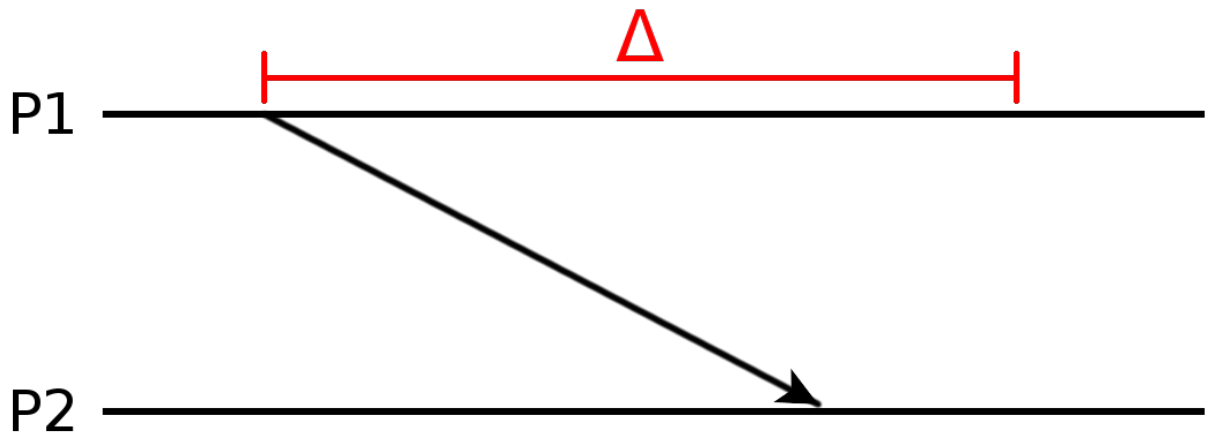


1. Envoyer valeur proposée à tous
2. Attendre de recevoir toutes les valeurs
3. Décider la valeur la plus faible

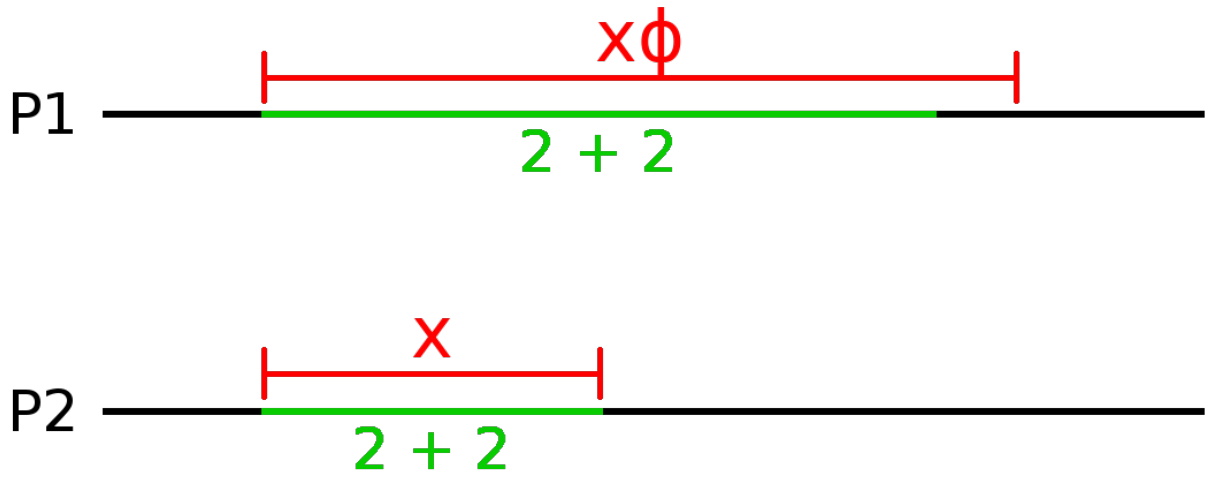


Fonctionne uniquement en l'absence de fautes.

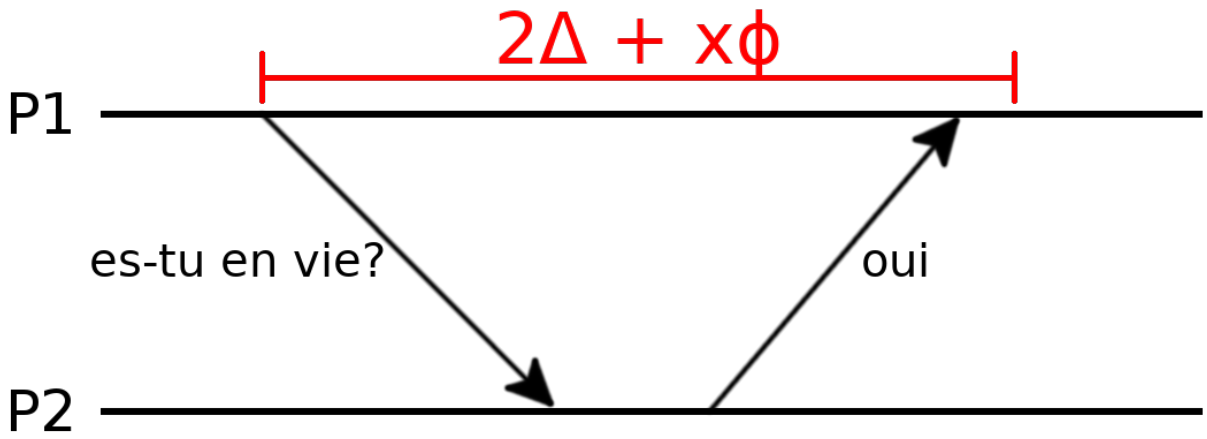
Borne Δ sur le temps de transmission des processus.



Borne Φ sur la vitesse relative des processus.

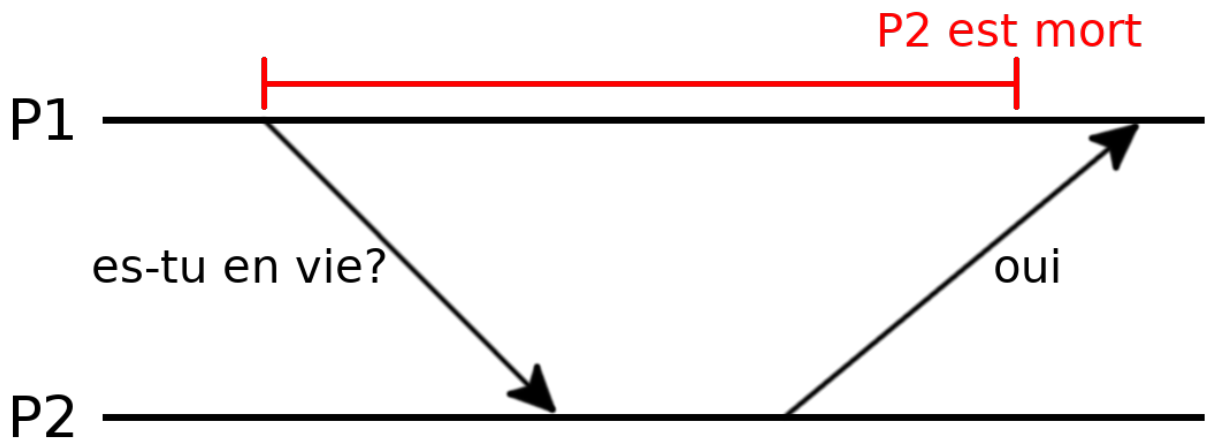


Permet une détection parfaite des fautes.

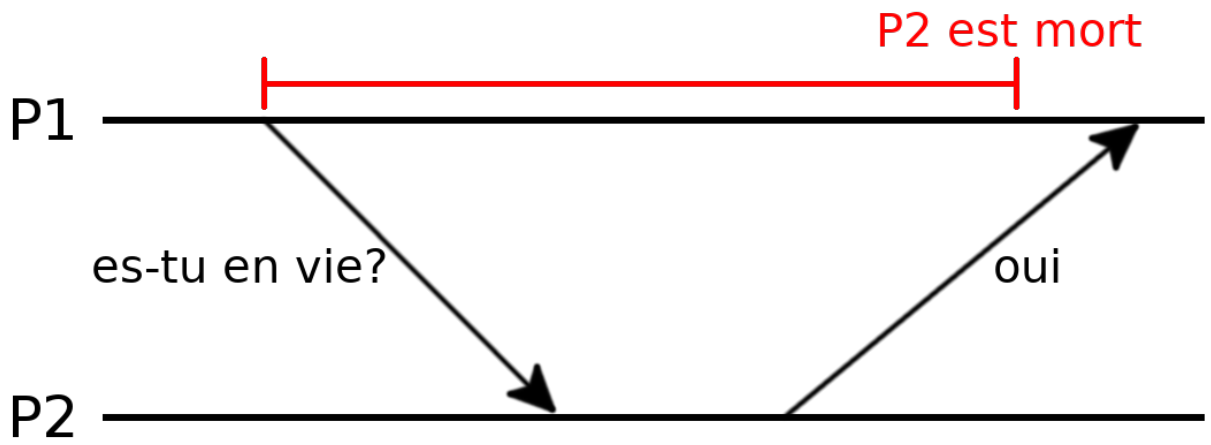


- Pas de borne sur les temps de transmission.
- Pas de borne sur la vitesse relative des processus.

- Pas de borne sur les temps de transmission.
- Pas de borne sur la vitesse relative des processus.



- Pas de borne sur les temps de transmission.
- Pas de borne sur la vitesse relative des processus.

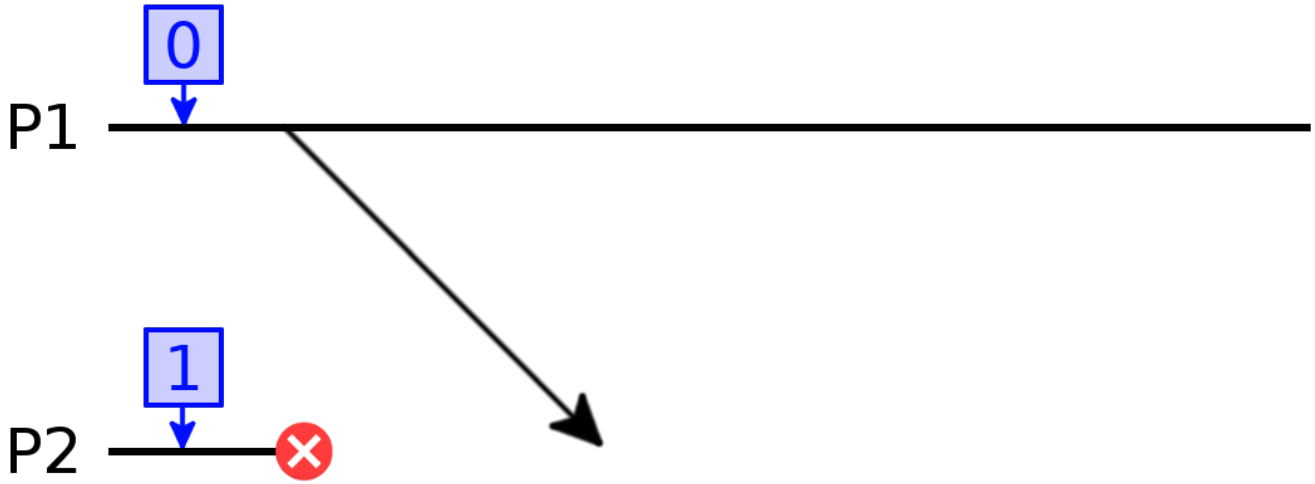


$\Longrightarrow$ Impossible pour P1 de distinguer entre :

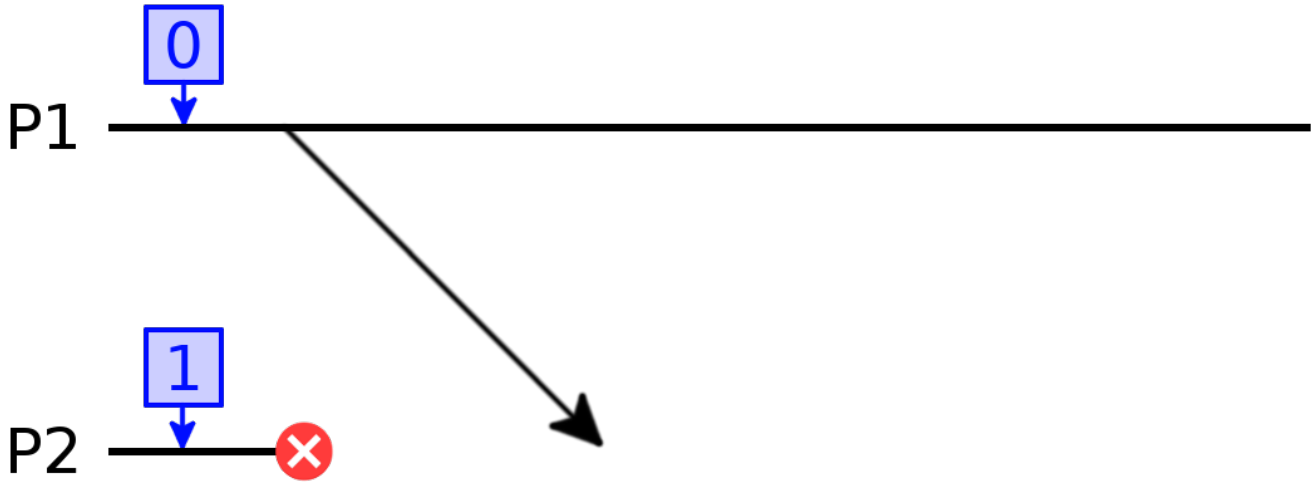
- P2 est lent
- P2 est en panne

Stratégie A : P1 attend le message de P2 avant de **décider**.

Stratégie A : P1 attend le message de P2 avant de **décider**.



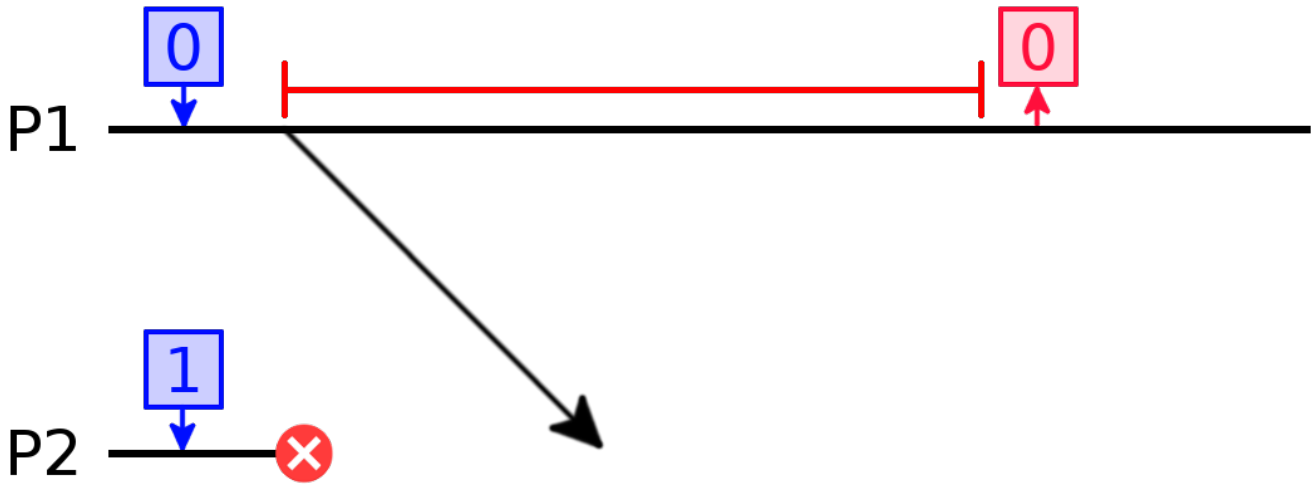
Stratégie A : P1 attend le message de P2 avant de **décider**.



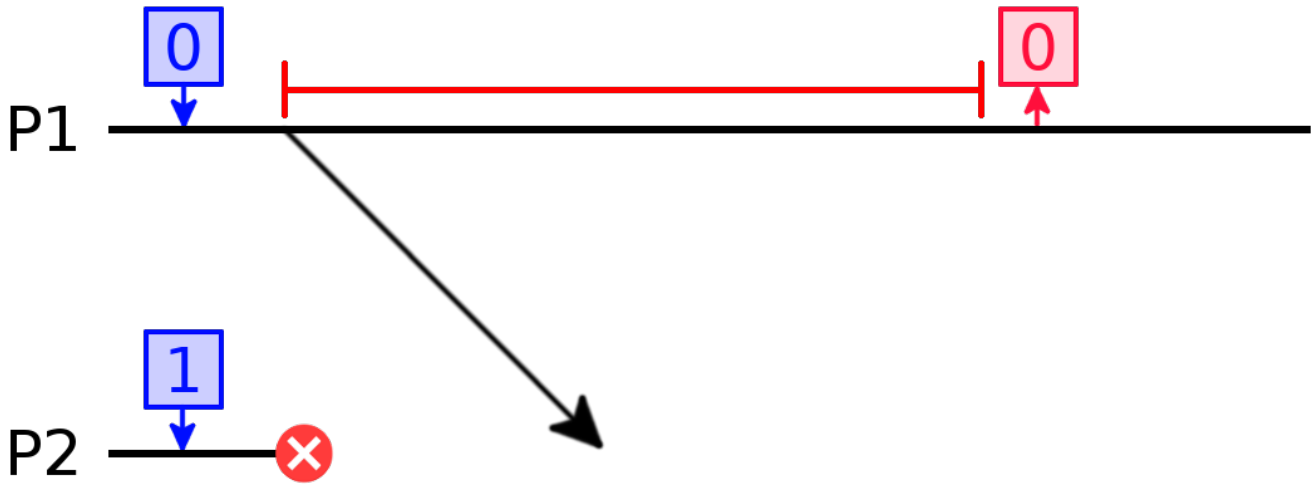
Violation de la propriété de terminaison !

Stratégie B : P1 **décide** après un certain temps.

Stratégie B : P1 **décide** après un certain temps.



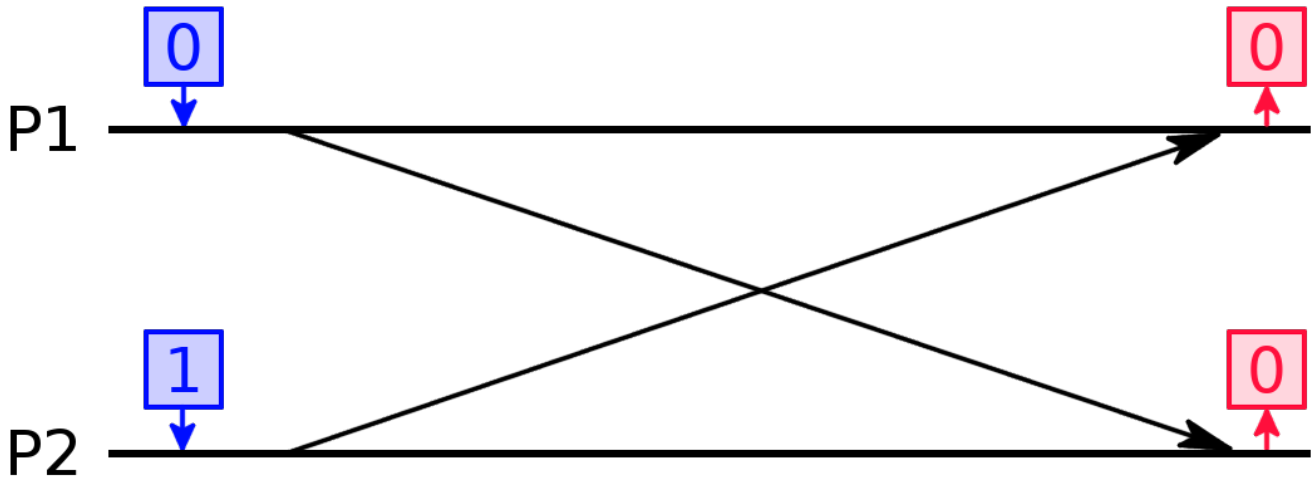
Stratégie B : P1 **décide** après un certain temps.



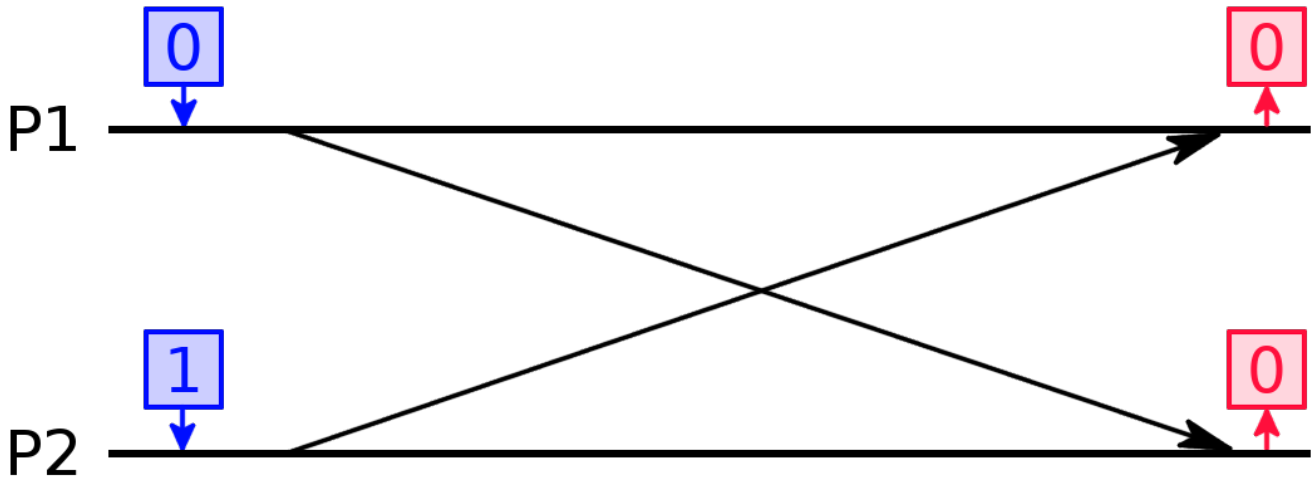
La stratégie B fonctionne.

Stratégie A : P1 attend le message de P2 avant de **décider**.

Stratégie A : P1 attend le message de P2 avant de **décider**.



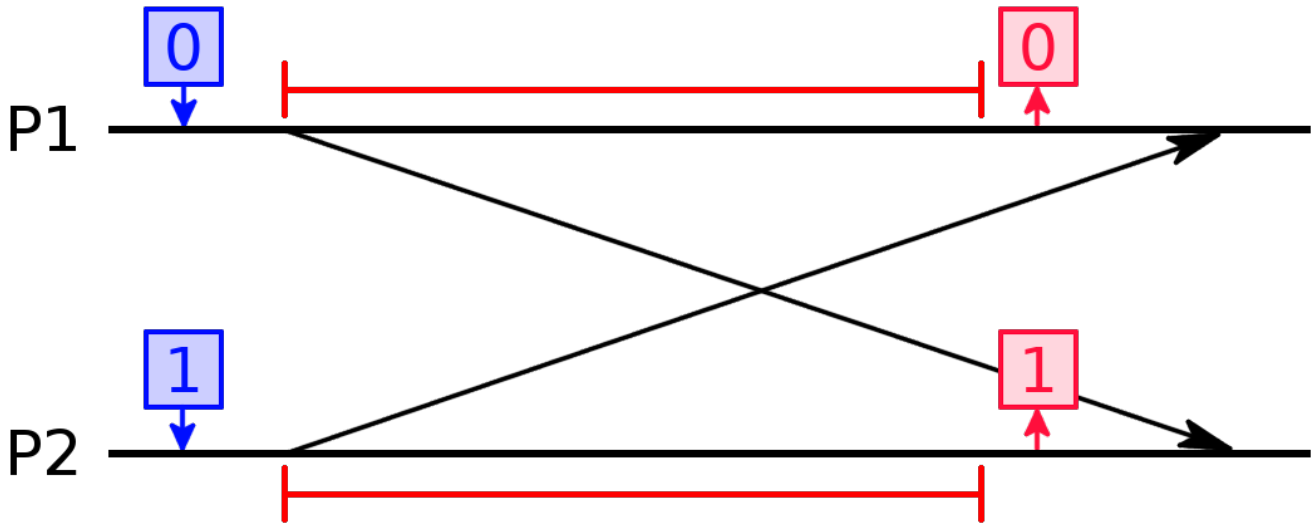
Stratégie A : P1 attend le message de P2 avant de **décider**.



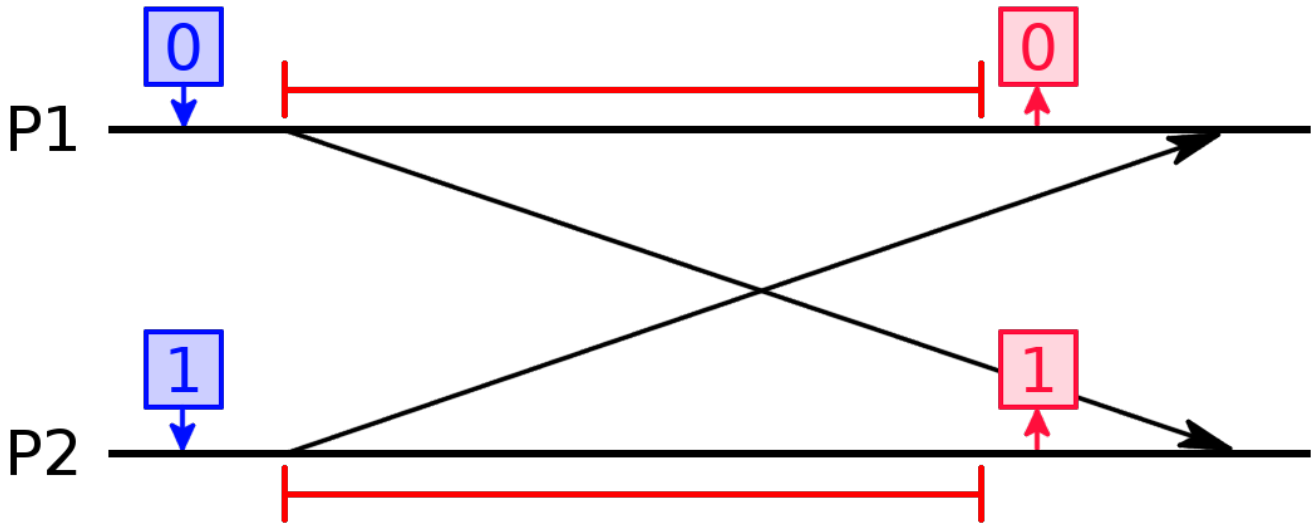
La stratégie A fonctionne.

Stratégie B : P1 **décide** après un certain temps.

Stratégie B : P1 **décide** après un certain temps.



Stratégie B : P1 **décide** après un certain temps.



Violation de la propriété de cohérence !

Impossible de distinguer les scénarios 1 et 2

Impossible de distinguer les scénarios 1 et 2

$\Longrightarrow$ Impossible de choisir entre stratégies A et B !

Impossible de distinguer les scénarios 1 et 2
$\Longrightarrow$ Impossible de choisir entre stratégies A et B !

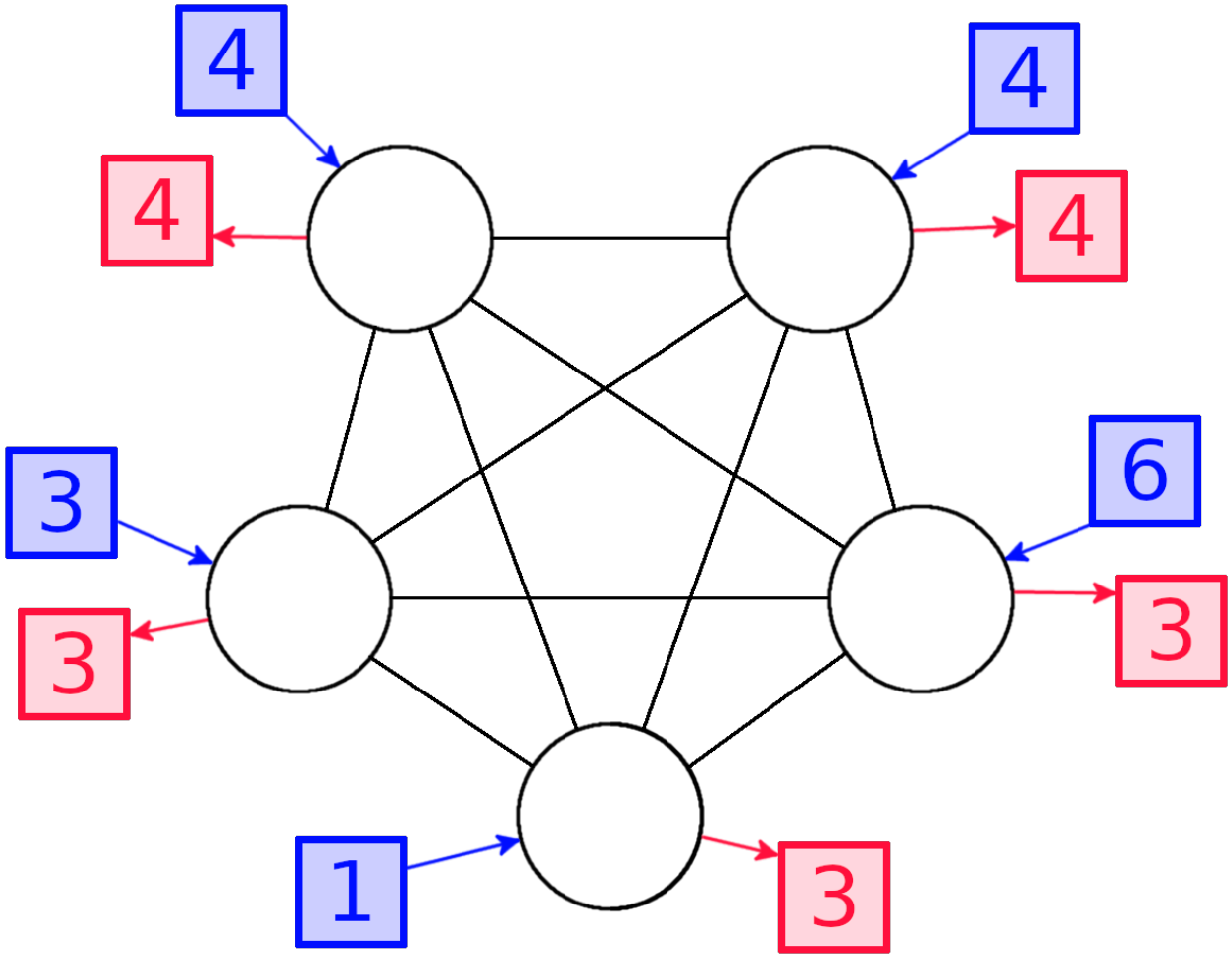
Impossibilité de Fischer, Lynch et Paterson (FLP 85)

Il est impossible de résoudre le consensus de façon déterministe dans un système asynchrone en présence de crashs.

1. **Changer le problème**
   Problème du k-accord

Généralisation du consensus.

- **Validité**
  La valeur décidée doit être l'une des valeurs proposées.
- **Terminaison**
  Tous les processus corrects décident en un temps fini.
- **Cohérence**
  Au plus k valeurs différentes peuvent être décidées.
- **Intégrité**
  Un processus doit décider au plus une fois.



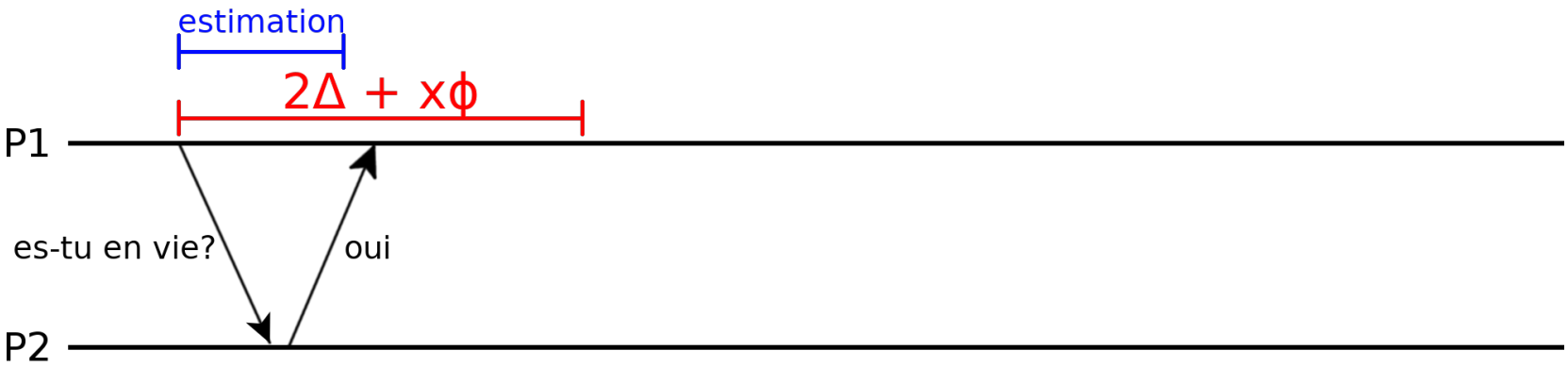
Exécution **valide** pour $k \geq 2$

Exécution **invalide** pour $k = 1$

1. Changer le problème
   - Problème du k-accord
2. **Changer le modèle**
   - Modèle partiellement synchrone

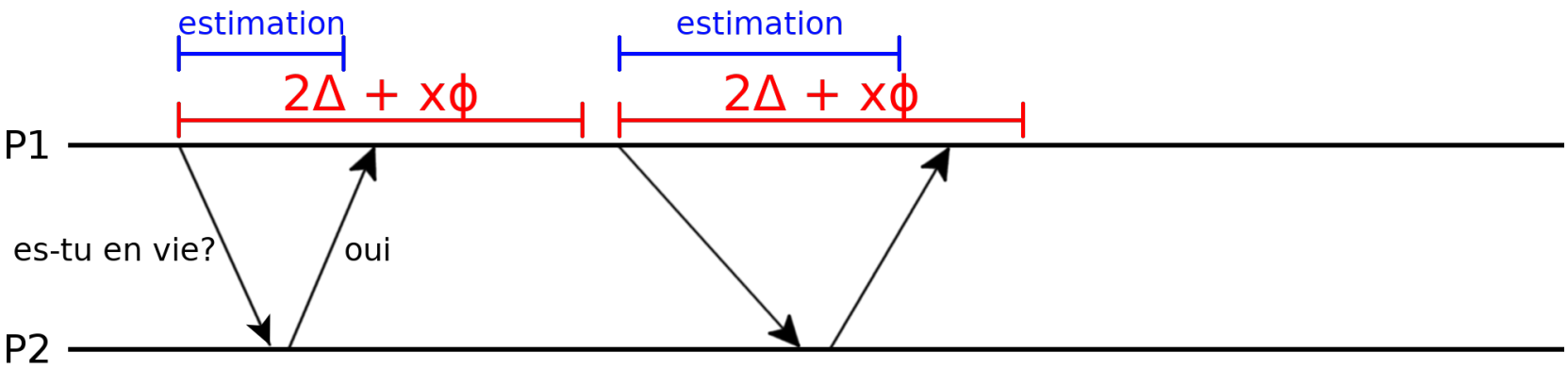
Multiples variantes.
Bornes Δ and Φ :

- Sont connues mais ne s'appliquent qu'après un certain temps, ou
- Sont inconnues.



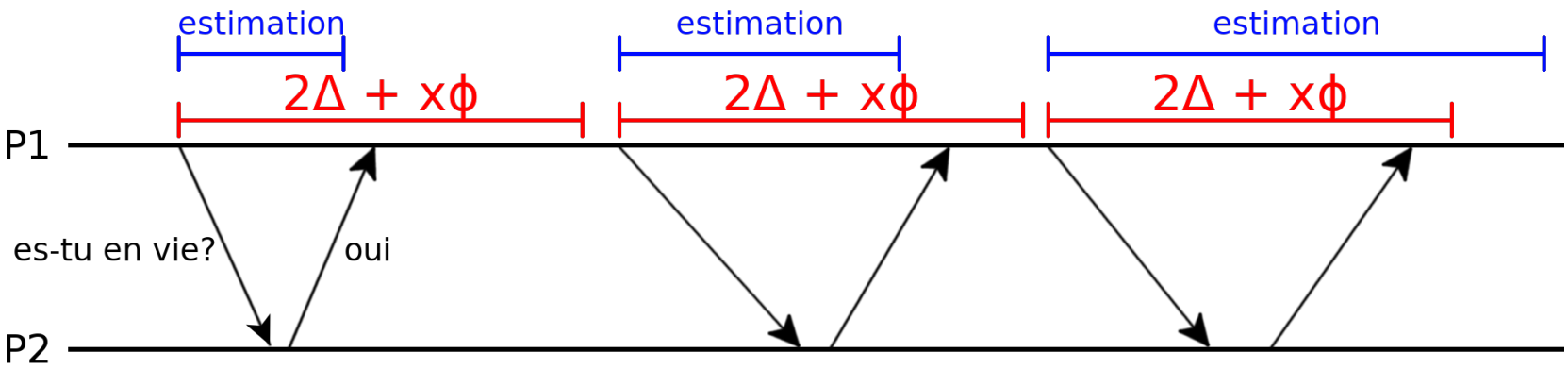
Multiples variantes.
Bornes Δ and Φ :

- Sont connues mais ne s'appliquent qu'après un certain temps, ou
- Sont inconnues.



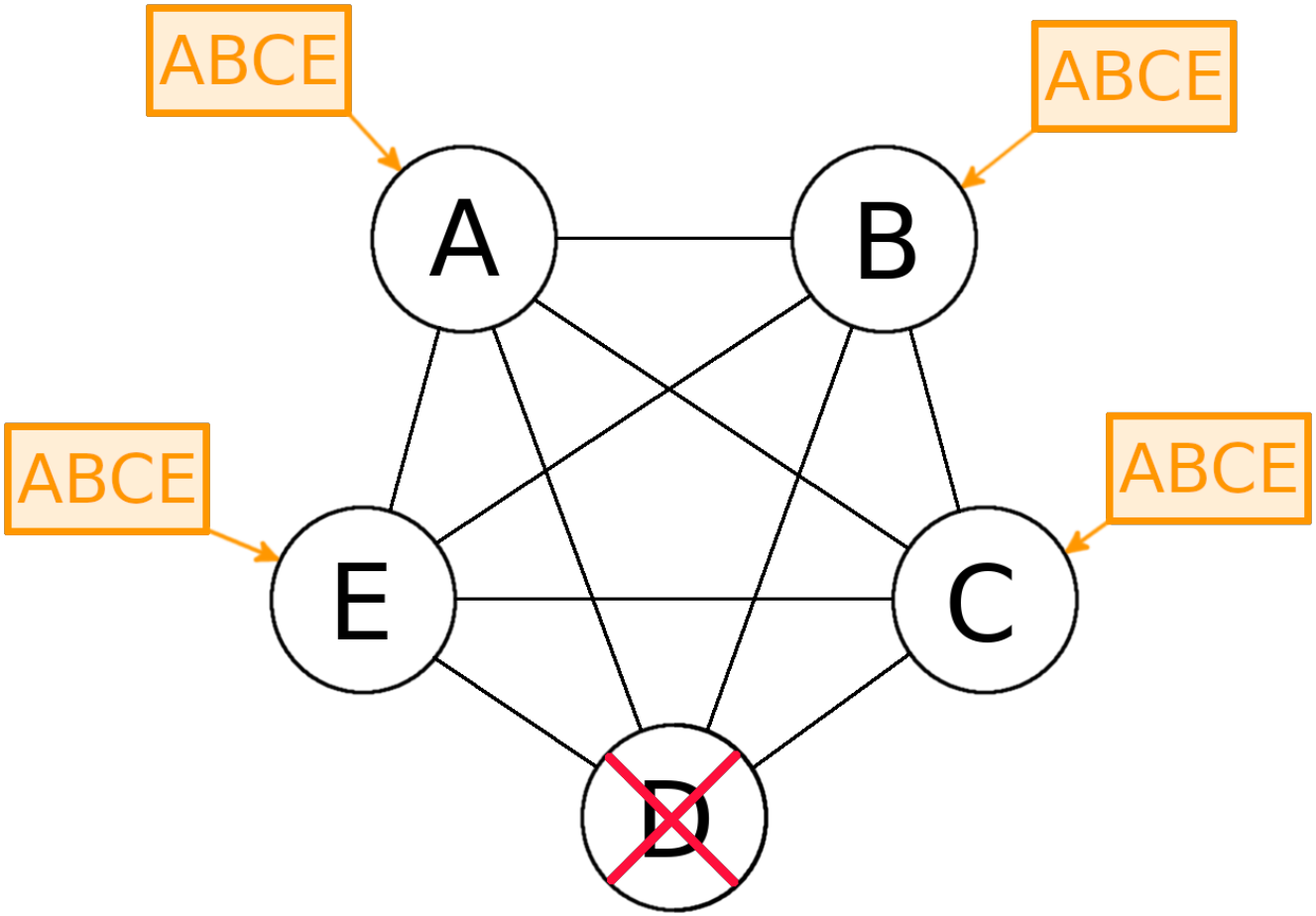
Multiples variantes.
Bornes Δ and Φ :

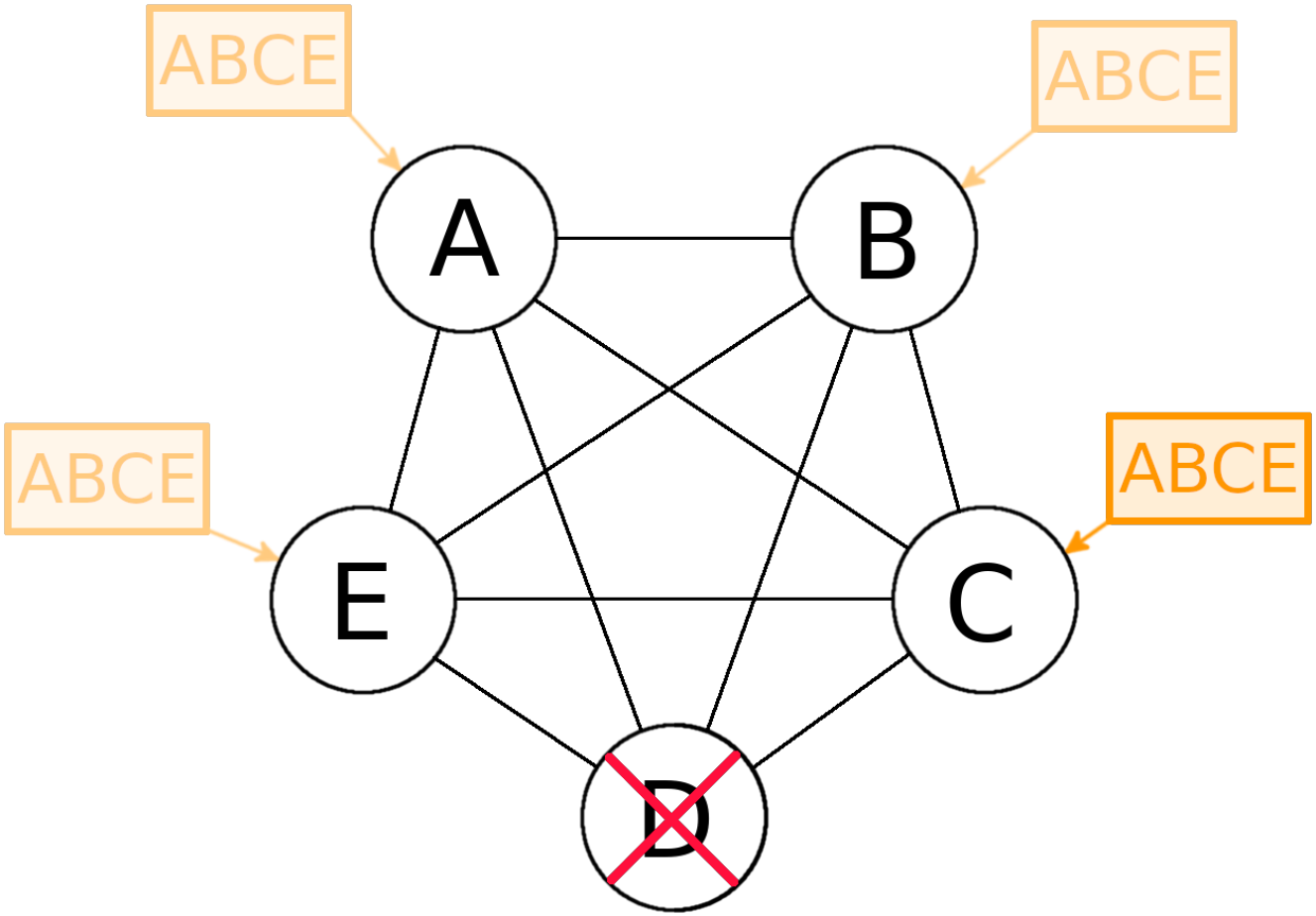
- Sont connues mais ne s'appliquent qu'après un certain temps, ou
- Sont inconnues.



1. **Changer le problème**
   Problème du k-accord
2. **Changer le modèle**
   Modèle partiellement synchrone
3. **Abstraction du modèle**
   Détecteurs de fautes

Détecteurs de fautes

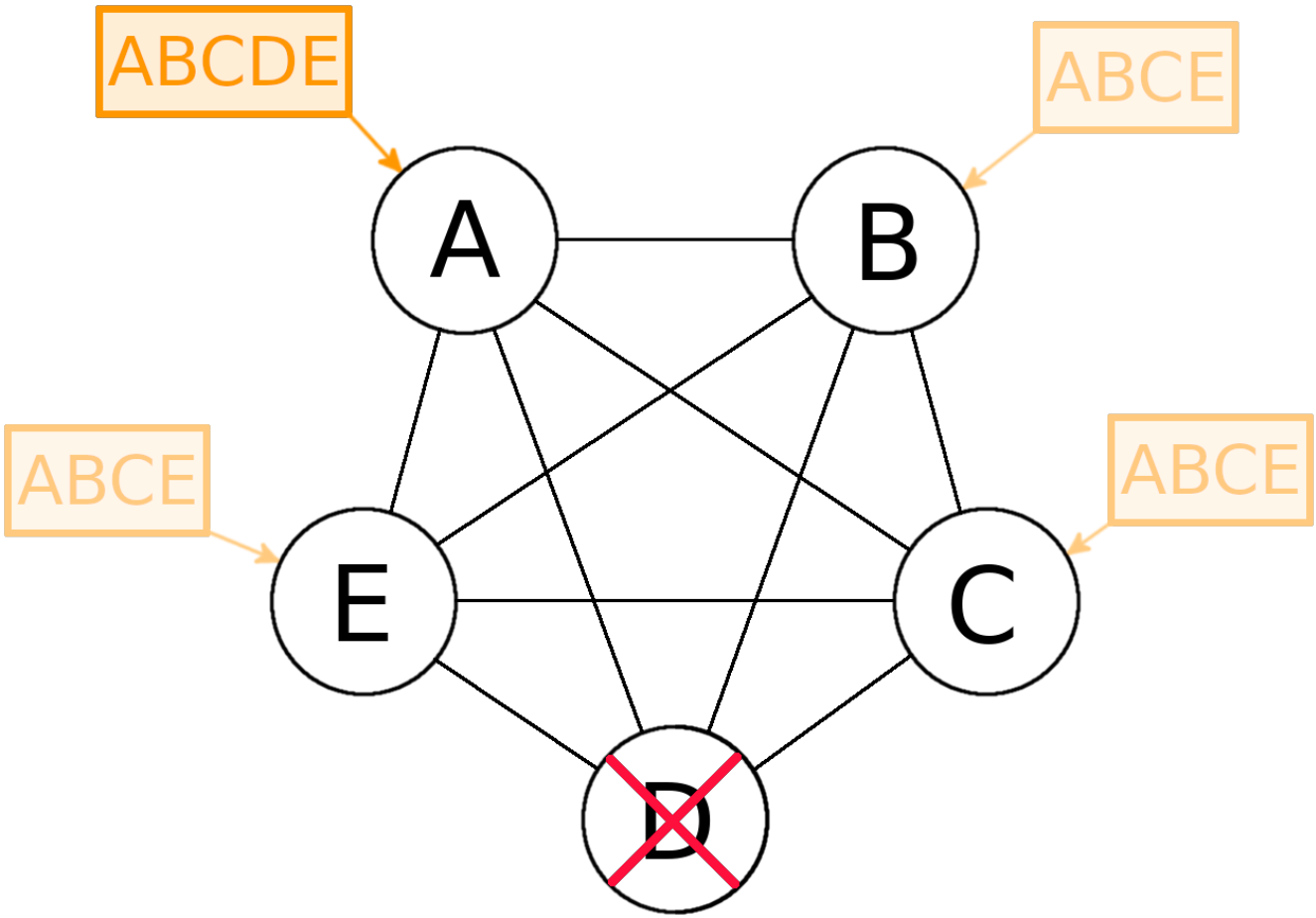


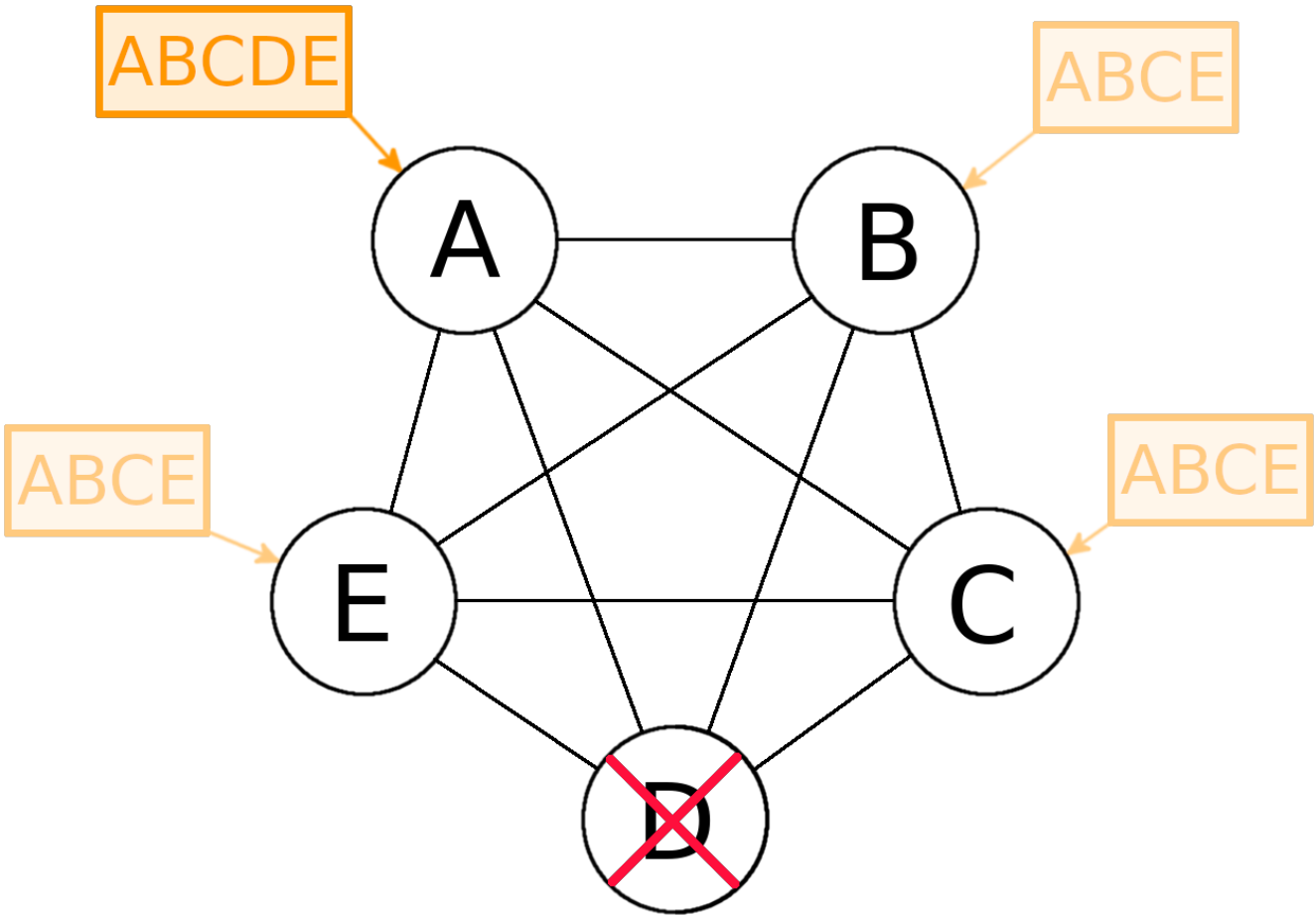


C **fait confiance** à A, B, C et E.

C **suspecte** D.

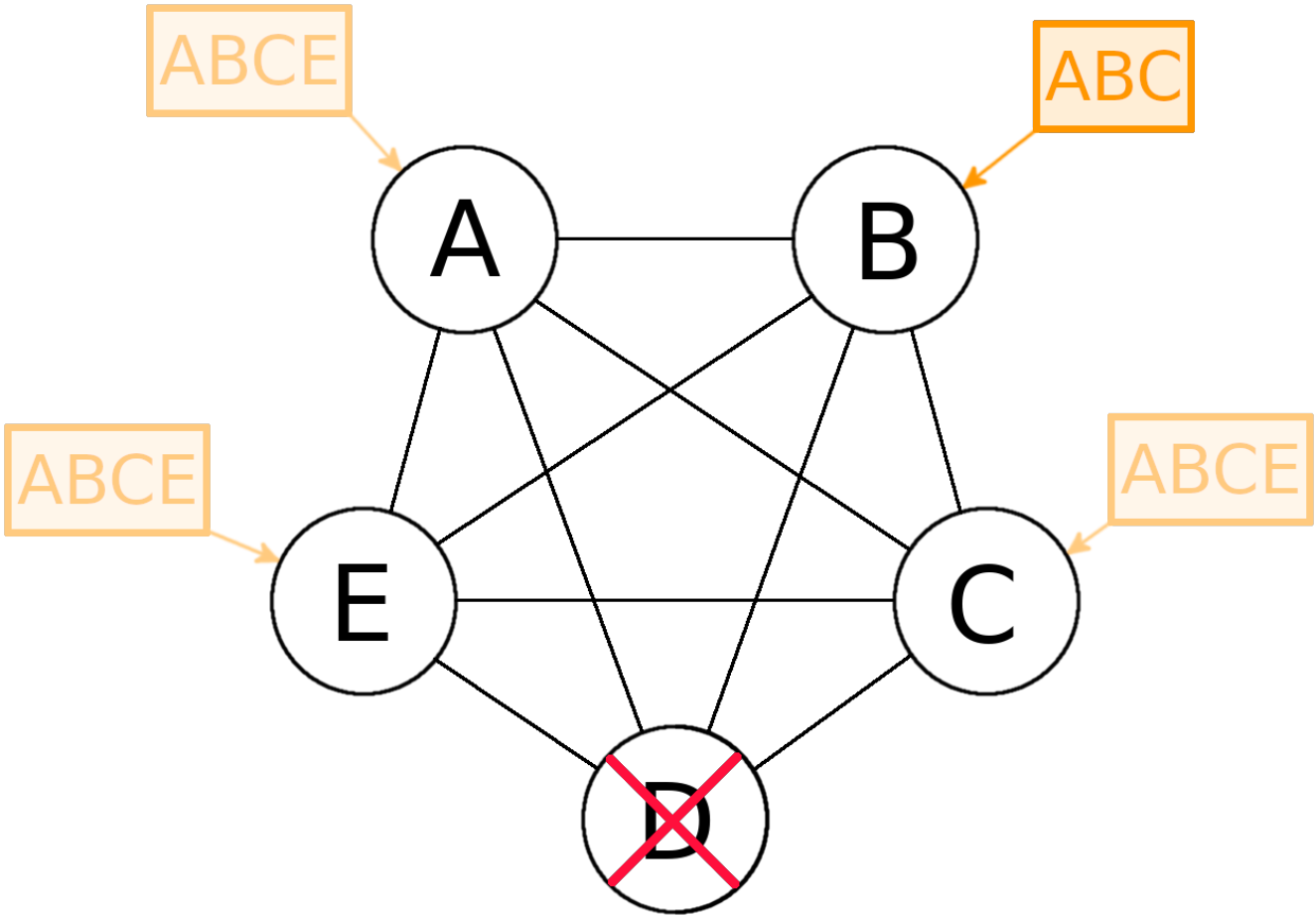
Informations non fiables

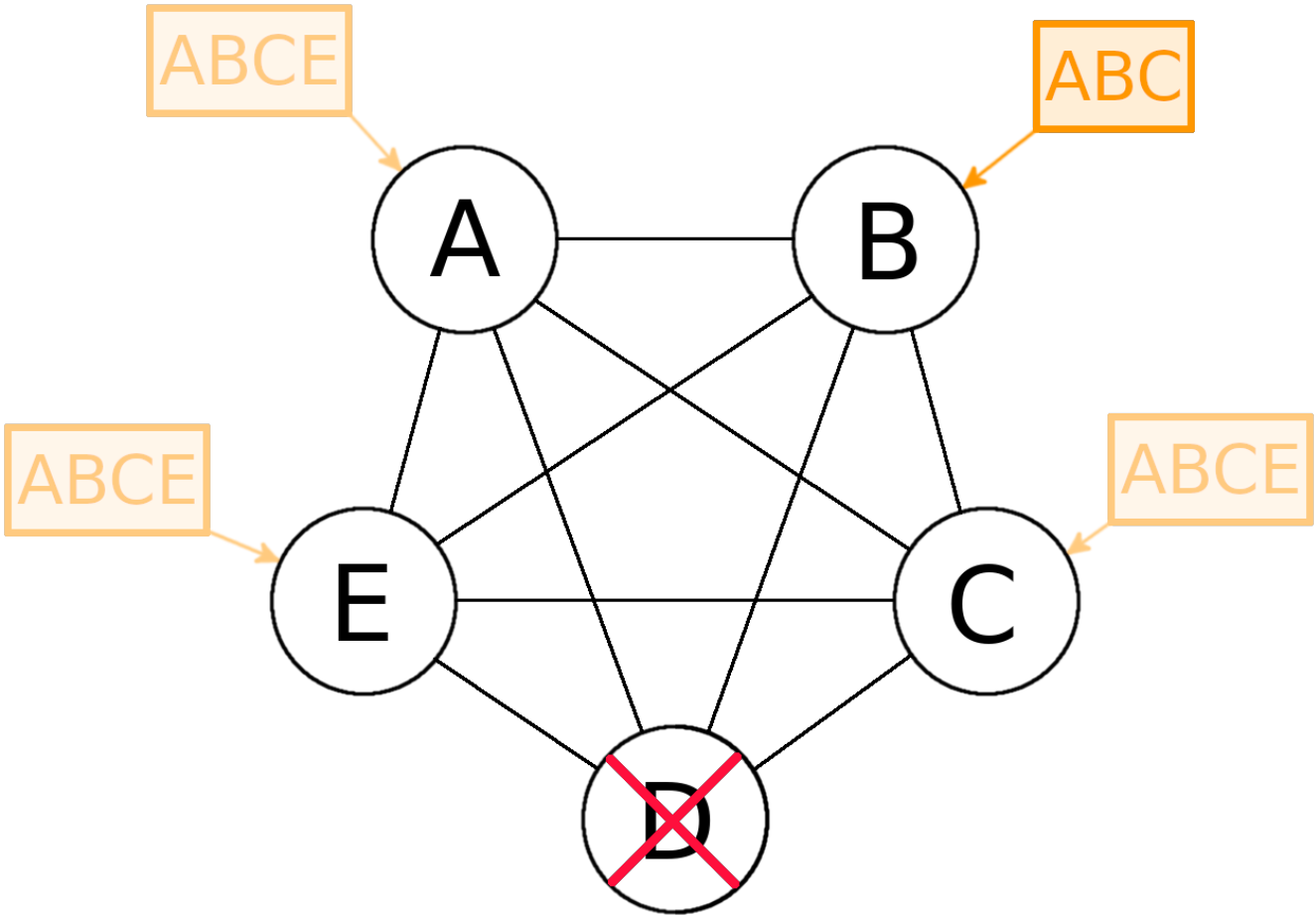




A **fait confiance** à D, à tort.
Retard dans la détection des fautes.

Informations non fiables





B **suspecte** E, à tort.

Fausse suspicion.

Retard dans la détection des fautes

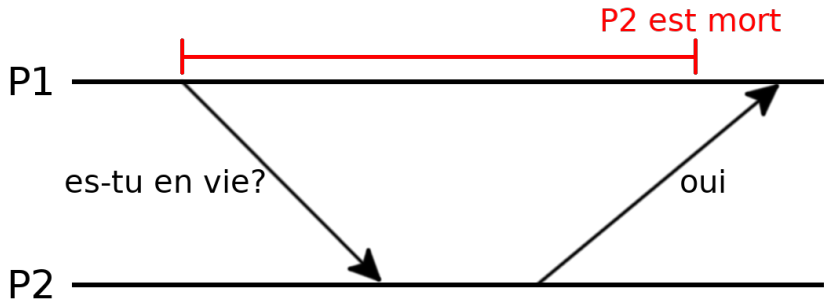
- Inévitable, quel que soit le modèle

Retard dans la détection des fautes

- Inévitable, quel que soit le modèle

Fausses suspicions

- Inévitable en asynchrone

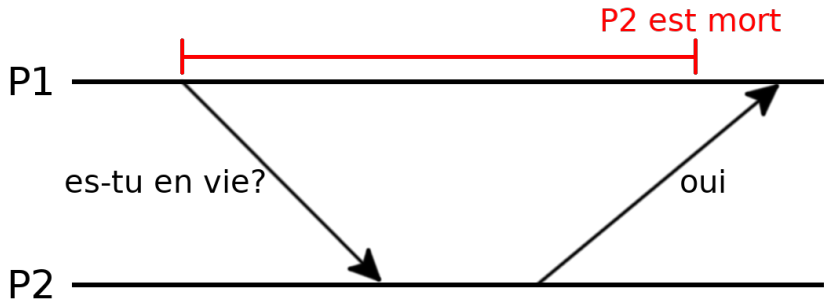


Retard dans la détection des fautes

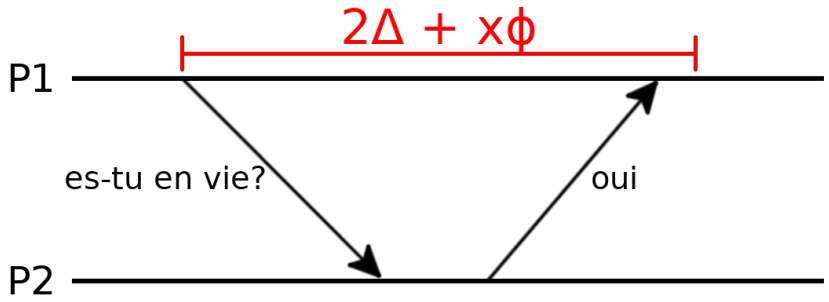
- Inévitable, quel que soit le modèle

Fausses suspicions

- Inévitable en asynchrone



- Évitable en synchrone

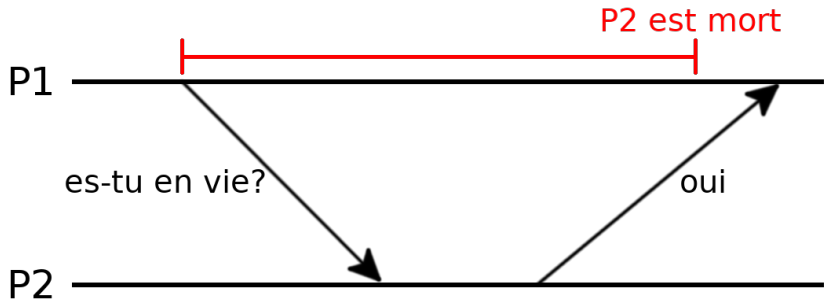


Retard dans la détection des fautes

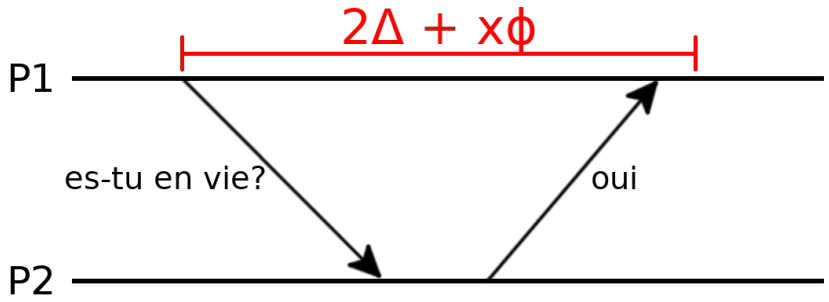
- Inévitable, quel que soit le modèle

Fausses suspicions

- Inévitable en asynchrone



- Évitable en synchrone



- Temporairement inévitable en partiellement synchrone

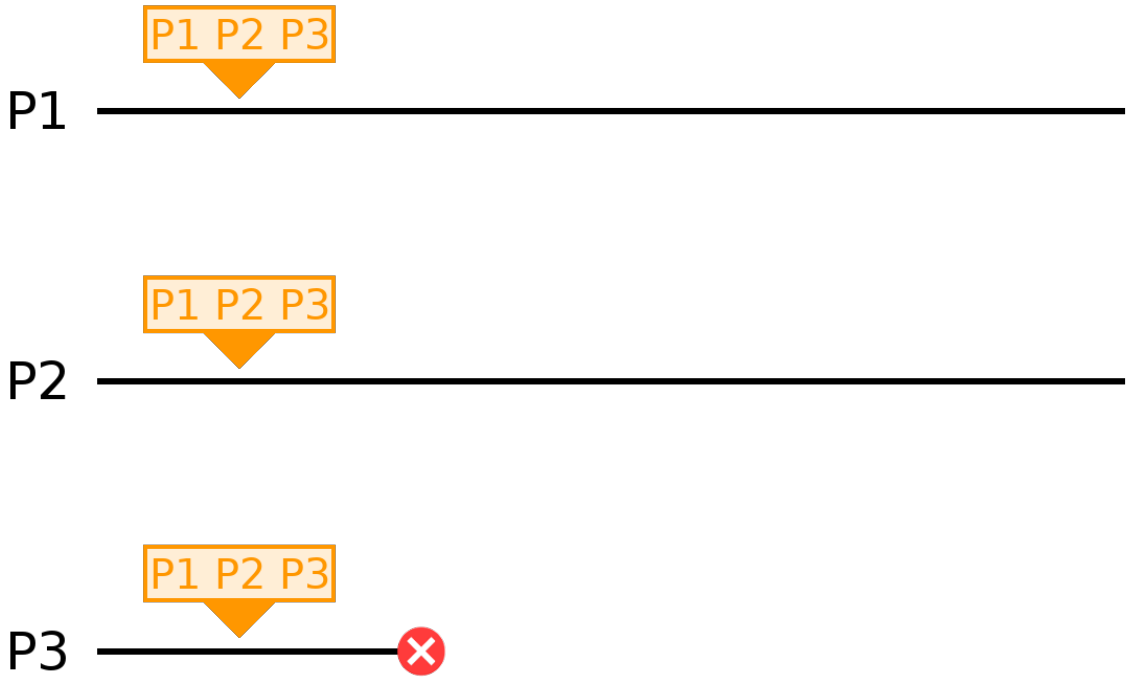
1. **Complétude**
   Les corrects doivent suspecter les fautifs.

Complétude forte

Après un certain temps, tous les fautifs sont suspectés par *tous* les corrects.

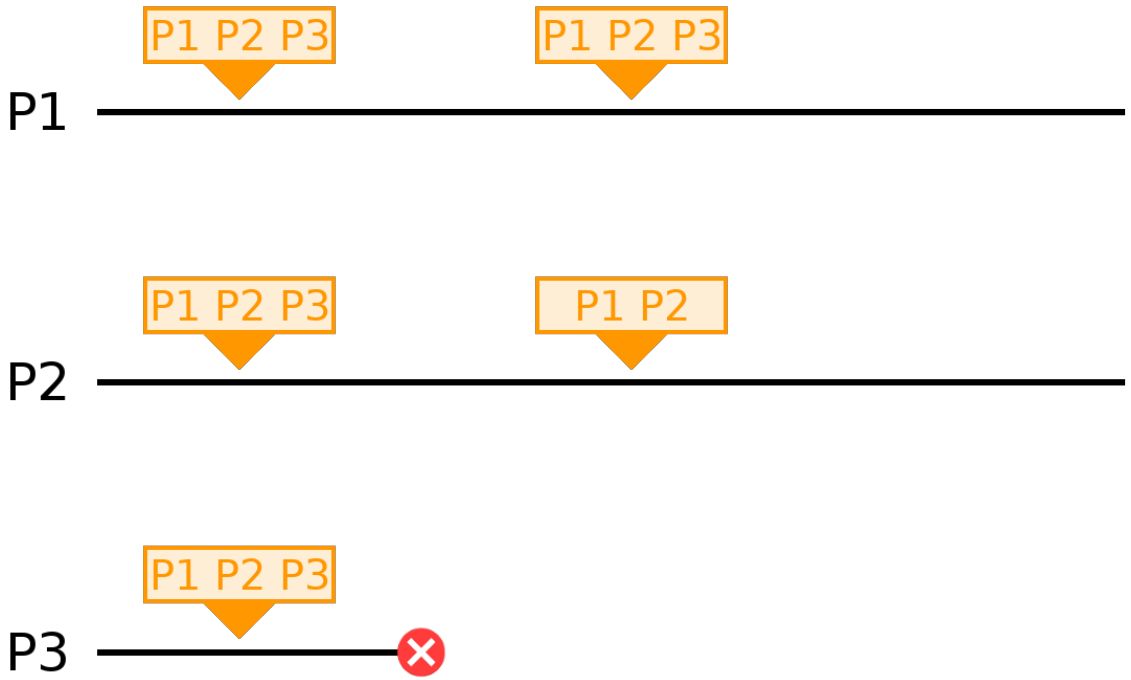
Complétude forte

Après un certain temps, tous les fautifs sont suspectés par *tous* les corrects.



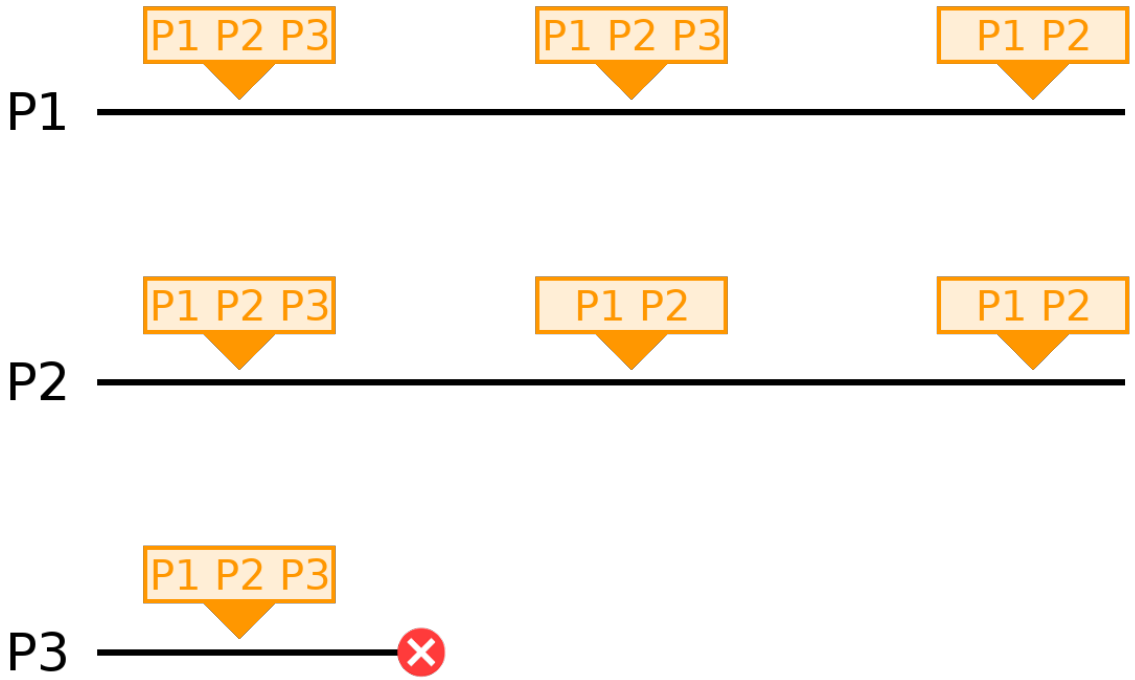
Complétude forte

Après un certain temps, tous les fautifs sont suspectés par *tous* les corrects.



Complétude forte

Après un certain temps, tous les fautifs sont suspectés par *tous* les corrects.

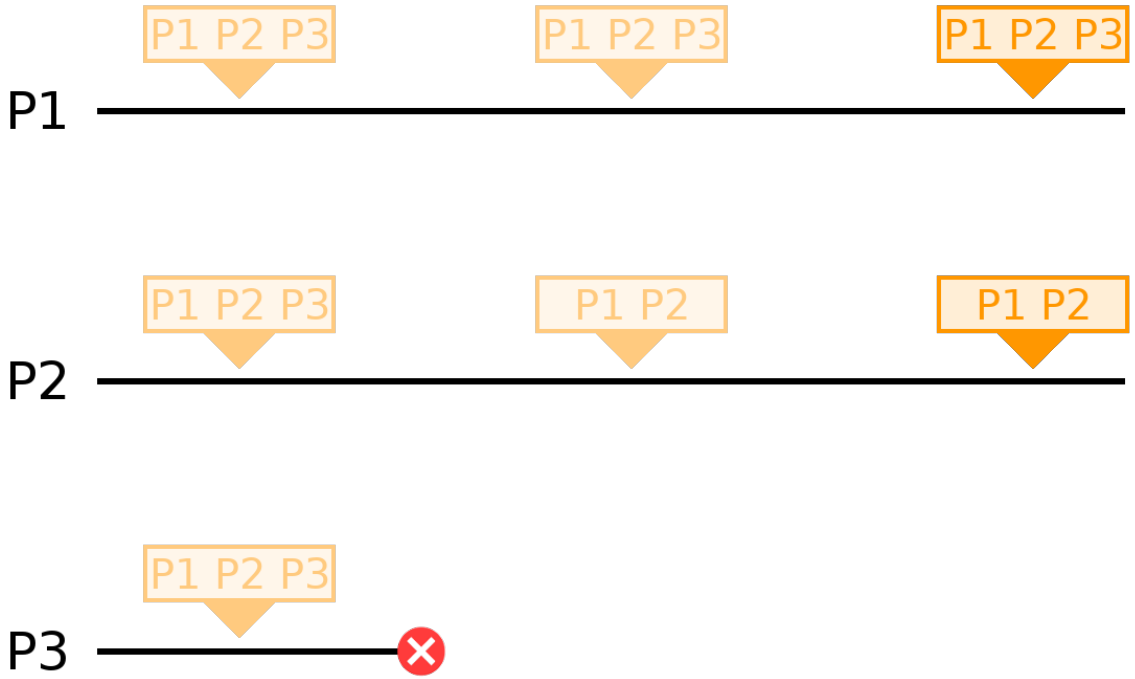


Complétude faible

Après un certain temps, tous les fautifs sont suspectés par *un* correct.

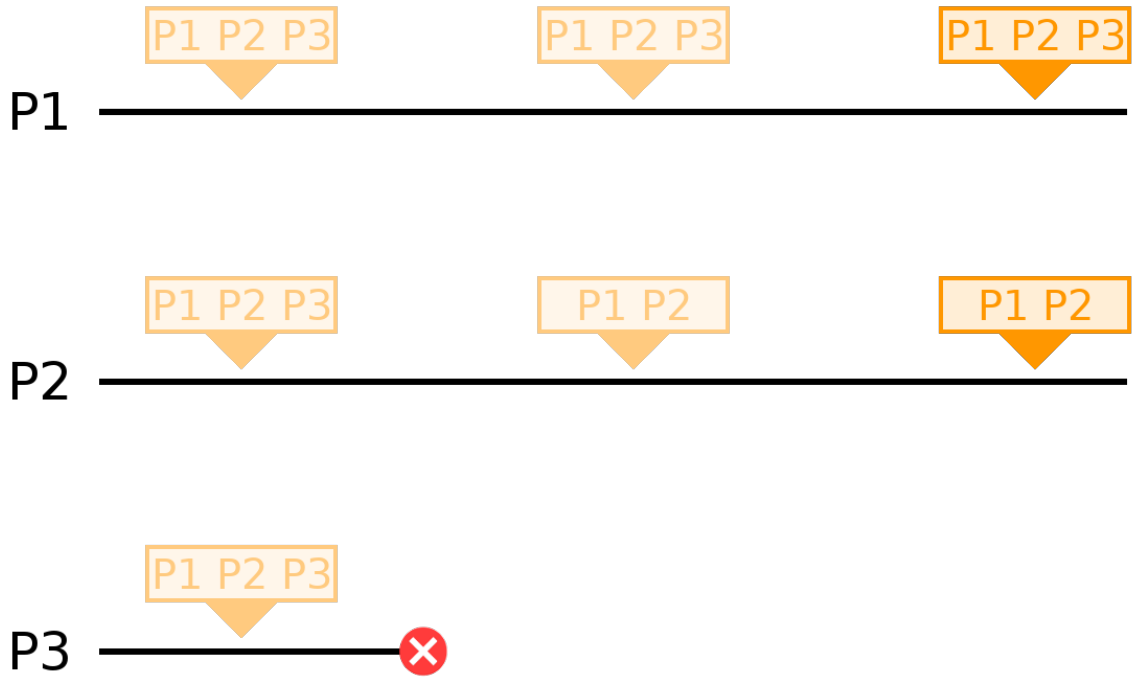
Complétude faible

Après un certain temps, tous les fautifs sont suspectés par *un* correct.



Construire la complétude forte à partir de la faible

P2 informe P1 de la panne.



Construire la complétude forte à partir de la faible

P2 informe P1 de la panne.

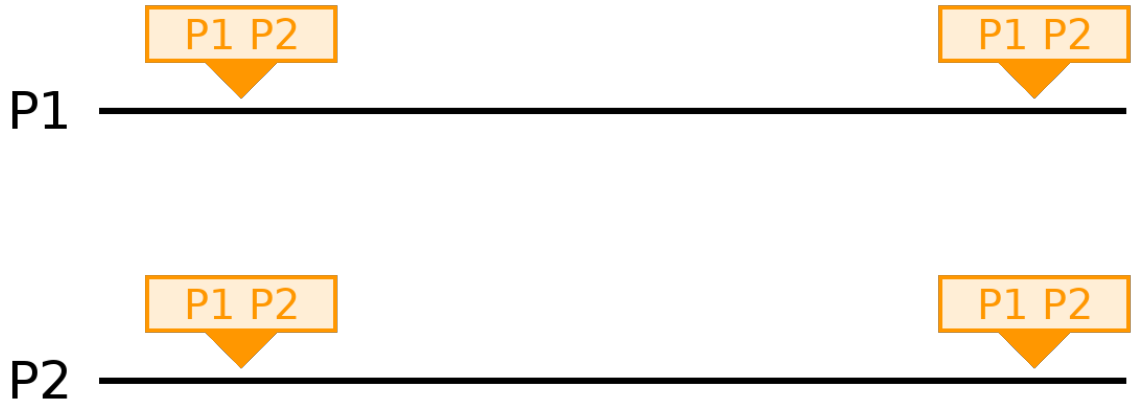
1. **Complétude**
   Les corrects doivent suspecter les fautifs.
2. **Justesse**
   Les corrects ne doivent pas être suspectés.

Justesse forte

Les corrects ne sont jamais suspectés.

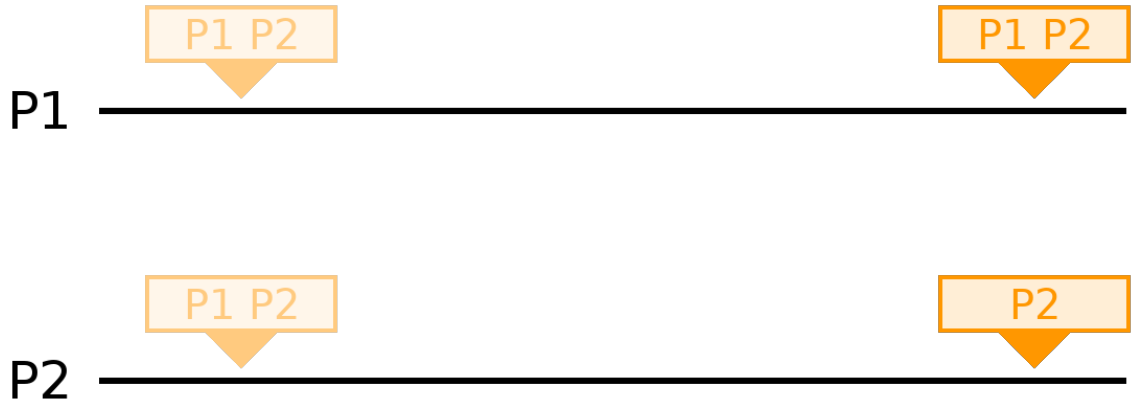
Justesse forte

Les corrects ne sont jamais suspectés.



Justesse faible

Un correct n'est jamais suspecté.

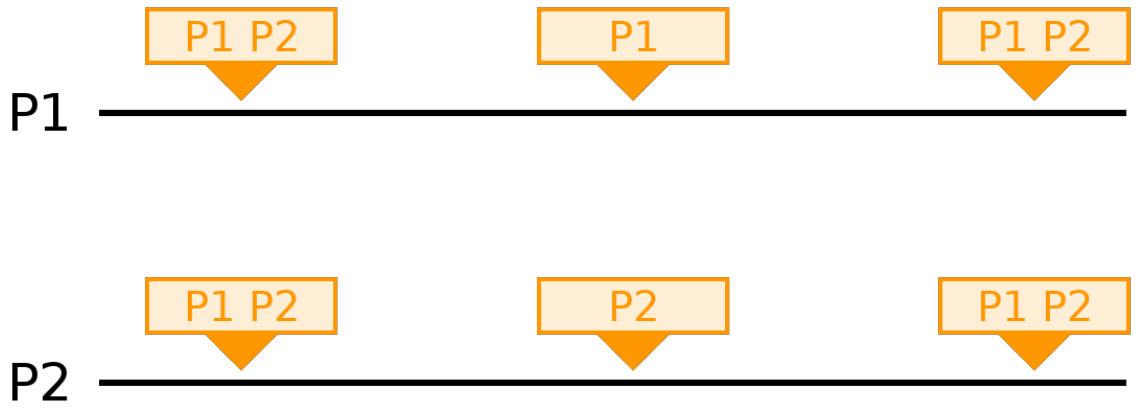


Justesse finalement forte

Après un certain temps, les corrects ne sont jamais suspectés.

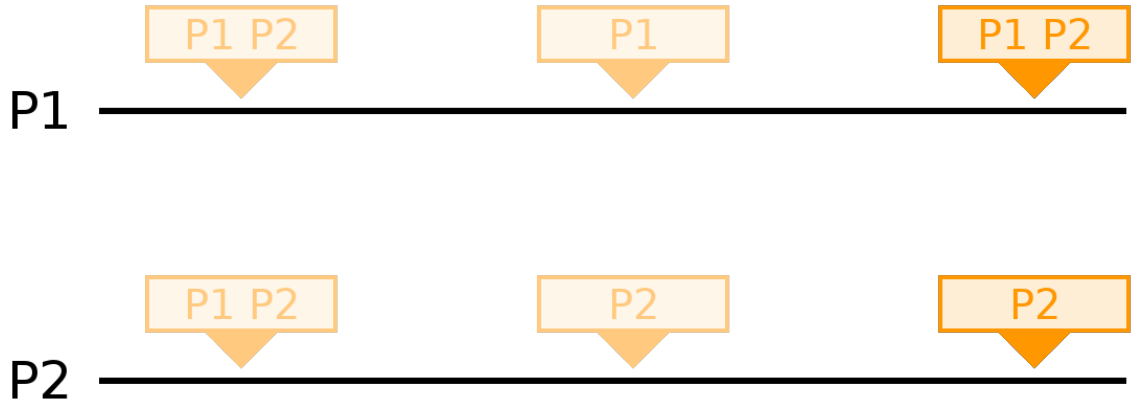
Justesse finalement forte

Après un certain temps, les corrects ne sont jamais suspectés.



Justesse finalement faible

Après un certain temps, *un* correct n'est jamais suspecté.



Classes de détecteurs

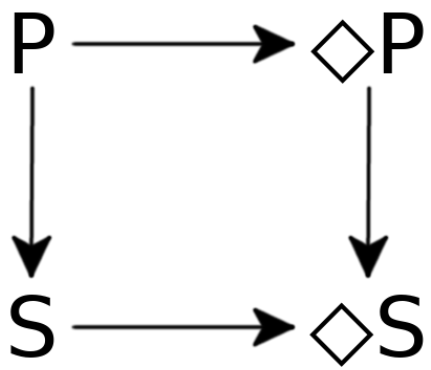
	Justesse			
	Forte	Faible	Finalement forte	Finalement faible
Complétude forte	P	S	$\diamond P$	$\diamond S$
Complétude faible	Q	W	$\diamond Q$	$\diamond W$

Classes de détecteurs

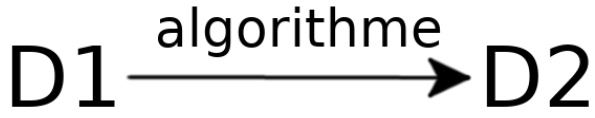
	Justesse			
	Forte	Faible	Finalement forte	Finalement faible
Complétude forte	P	S	$\diamond P$	$\diamond S$
Complétude faible	Q	W	$\diamond Q$	$\diamond W$

	Justesse			
	Forte	Faible	Finalement forte	Finalement faible
Complétude forte	P	S	$\diamond P$	$\diamond S$
Complétude faible	Q	W	$\diamond Q$	$\diamond W$

Force des détecteurs



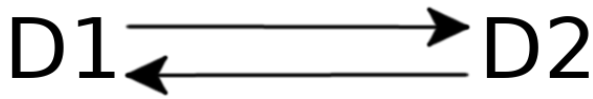
D1 est **plus fort** que D2 :
$\exists$ un algorithme qui construit D2 à partir de D1.



D1 est **plus fort** que D2 :
$\exists$ un algorithme qui construit D2 à partir de D1.

Si un problème peut être résolu avec D2, alors il peut être résolu avec D1.

D1 est **équivalent** à D2 :
D1 est plus fort que D2, et vice-versa.



D est **suffisant** pour résoudre $\mathcal{P}$:
$\exists$ un algorithme qui résout $\mathcal{P}$ en s'appuyant uniquement sur D.

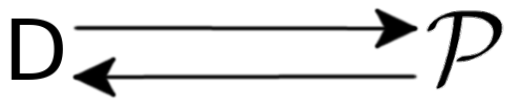
D est le **détecteur minimal** pour résoudre $\mathcal{P}$:
D est suffisant pour résoudre $\mathcal{P}$.
et
D est plus faible que tous les autres détecteurs suffisants pour résoudre $\mathcal{P}$.

D est le **détecteur minimal** pour résoudre $\mathcal{P}$:
D est suffisant pour résoudre $\mathcal{P}$.
et
D est plus faible que tous les autres détecteurs suffisants pour résoudre $\mathcal{P}$.

Autrement dit :
$\exists$ un algorithme qui construit D à partir de $\mathcal{P}$, et vice-versa.

D est le **détecteur minimal** pour résoudre $\mathcal{P}$:
D est suffisant pour résoudre $\mathcal{P}$.
et
D est plus faible que tous les autres détecteurs suffisants pour résoudre $\mathcal{P}$.

Autrement dit :
$\exists$ un algorithme qui construit D à partir de $\mathcal{P}$, et vice-versa.
$\Longrightarrow$ D est nécessaire et suffisant pour résoudre $\mathcal{P}$.



Hypothèse : majorité de corrects.

Hypothèse : majorité de corrects.

1. $\diamond S$ est **suffisant** pour résoudre le consensus.
   (et donc P, S et $\diamond P$ aussi)

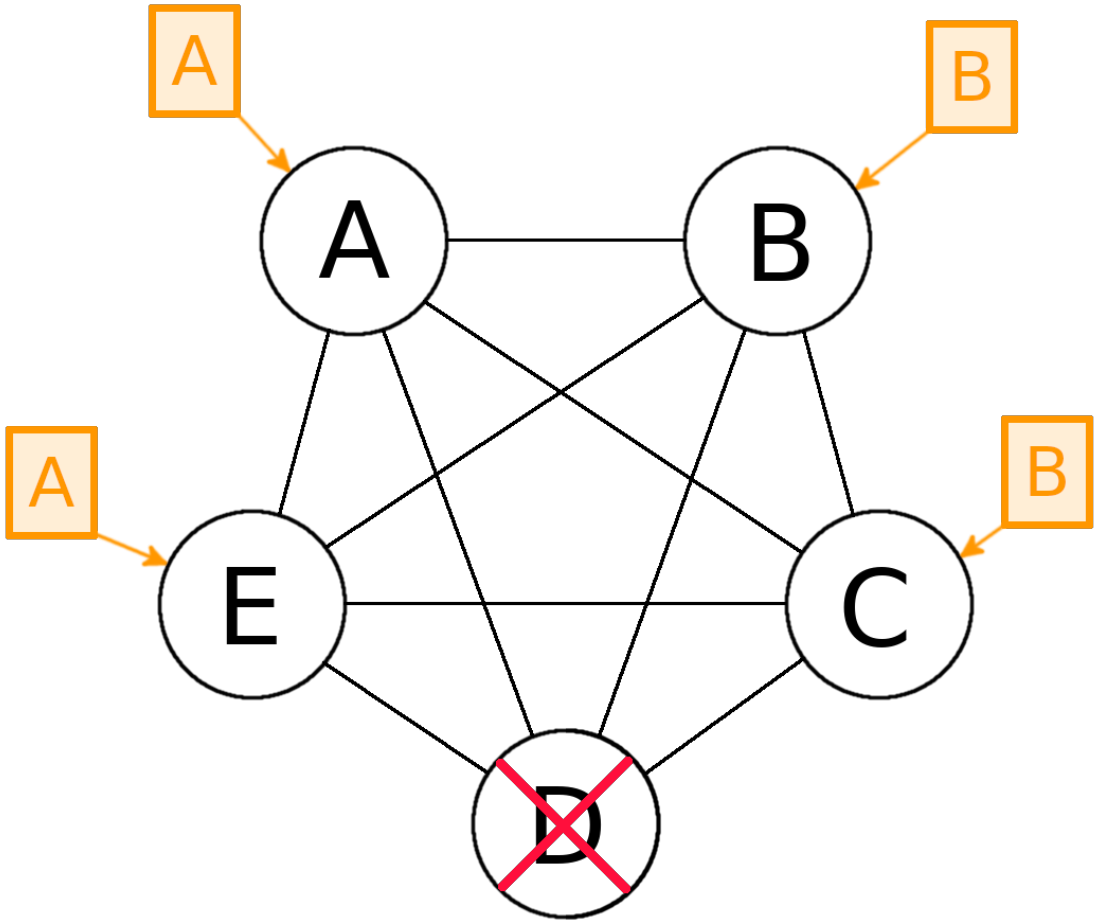
Hypothèse : majorité de corrects.

1. $\diamond S$ est **suffisant** pour résoudre le consensus.
   (et donc P, S et $\diamond P$ aussi)
   $\Longrightarrow$ Impossible d'implémenter $\diamond S$ en asynchrone (FLP) !

Hypothèse : majorité de corrects.

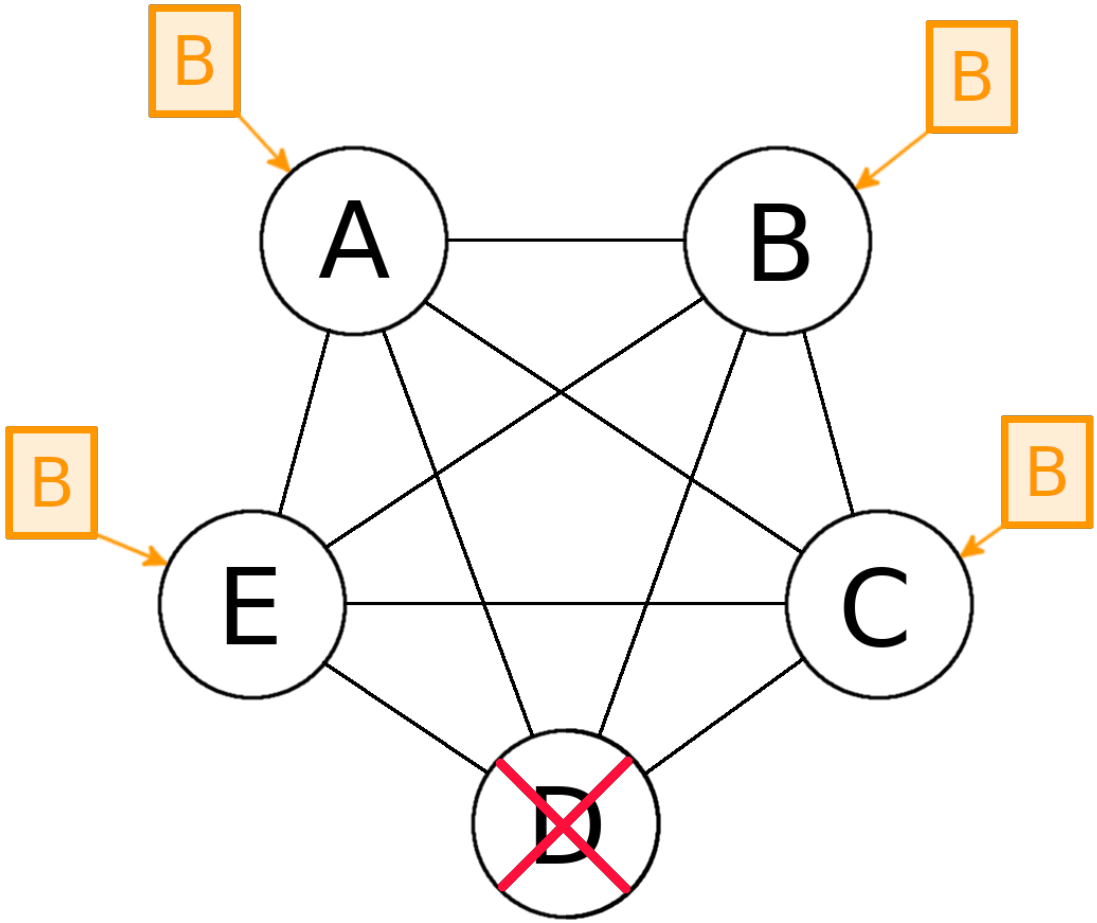
1. $\diamond S$ est **suffisant** pour résoudre le consensus.
   (et donc P, S et $\diamond P$ aussi)
   $\Longrightarrow$ Impossible d'implémenter $\diamond S$ en asynchrone (FLP) !
2. $\diamond S$ est le **détecteur minimal** pour résoudre le consensus.

Le détecteur de leader Ω



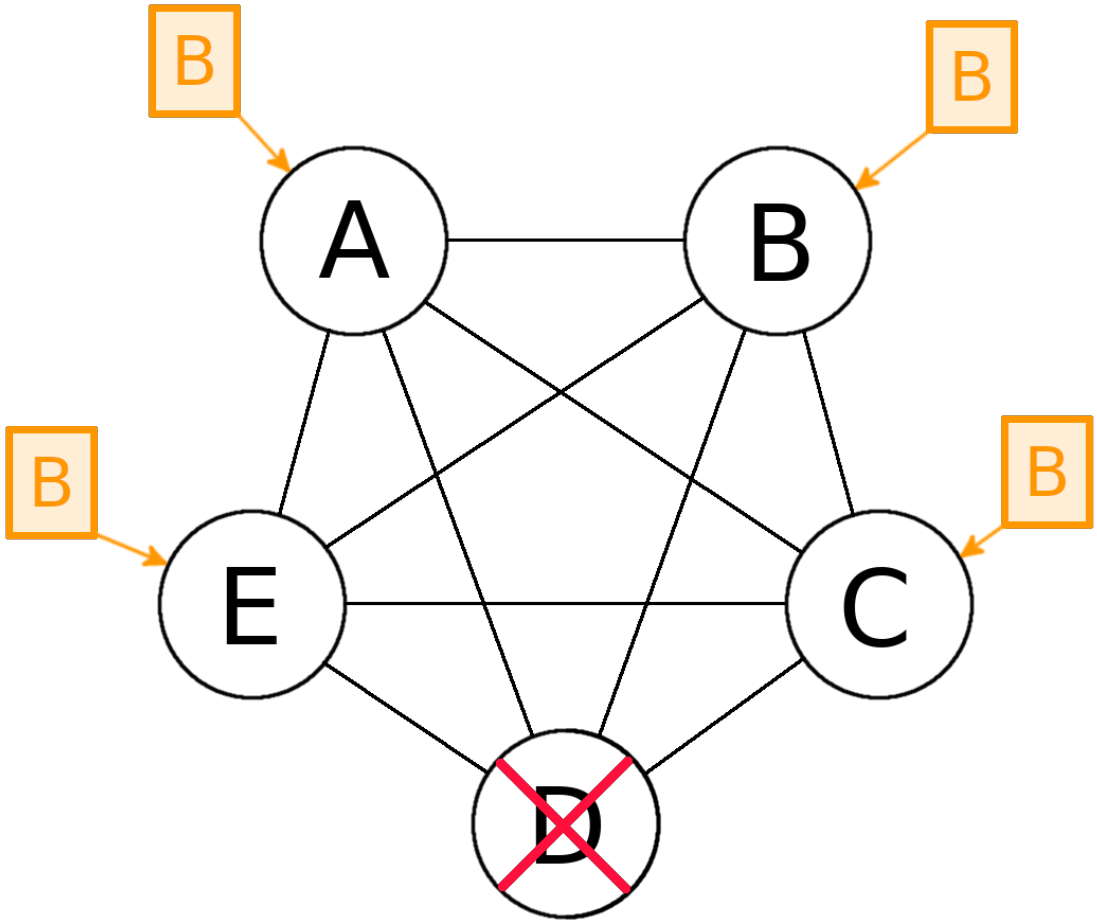
Indique **un seul** processus correct.

Le détecteur de leader Ω



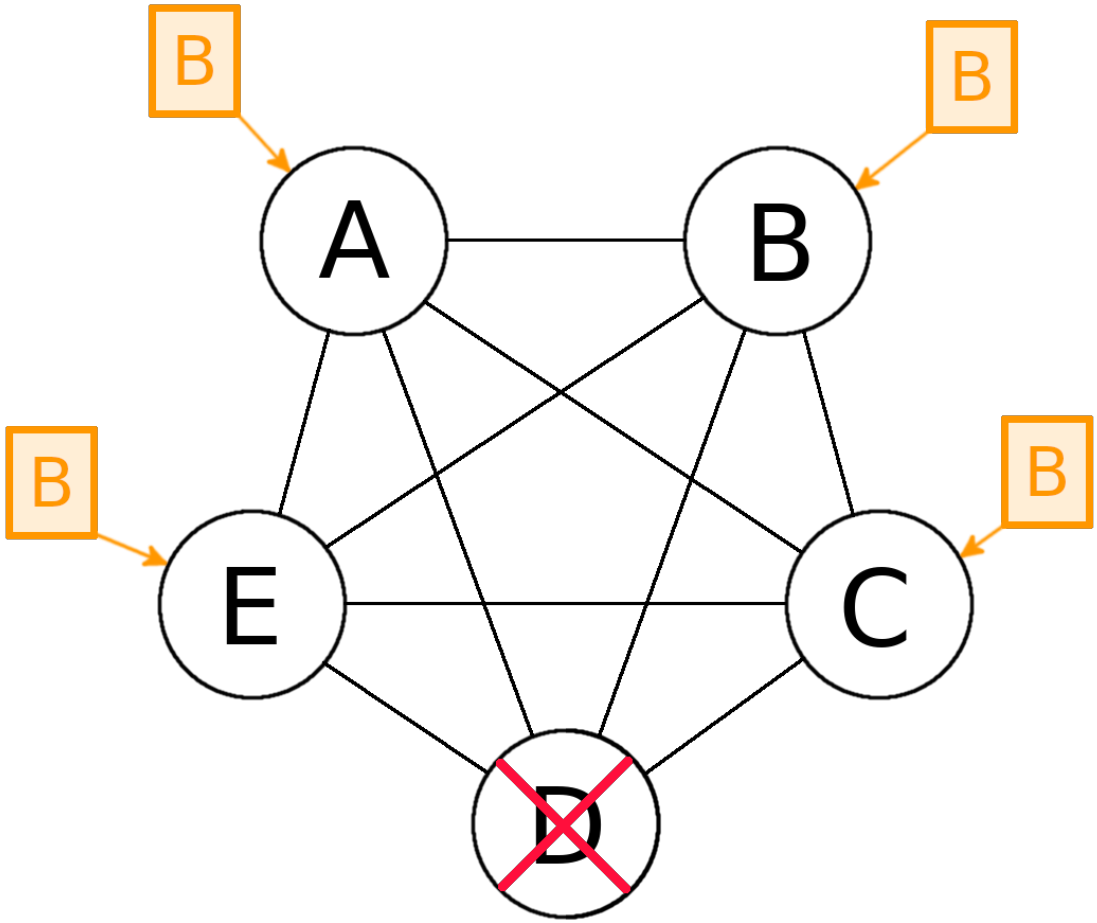
Indique **un seul** processus correct.
À terme, le même pour tout le monde.

Le détecteur de leader Ω



Indique **un seul** processus correct.
À terme, le même pour tout le monde.
Les processus ne savent pas quand l'état stable est atteint !.

Le détecteur de leader Ω



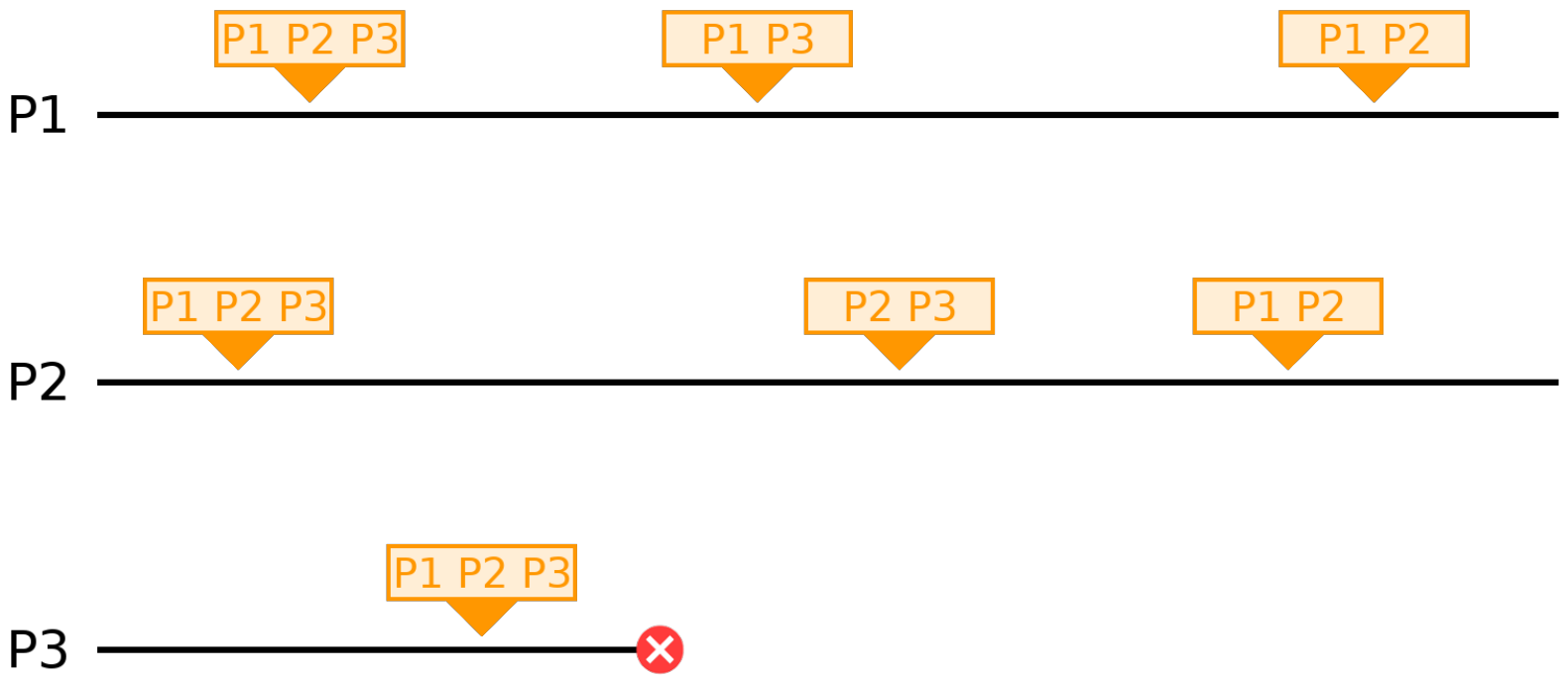
Indique **un seul** processus correct.

À terme, le même pour tout le monde.

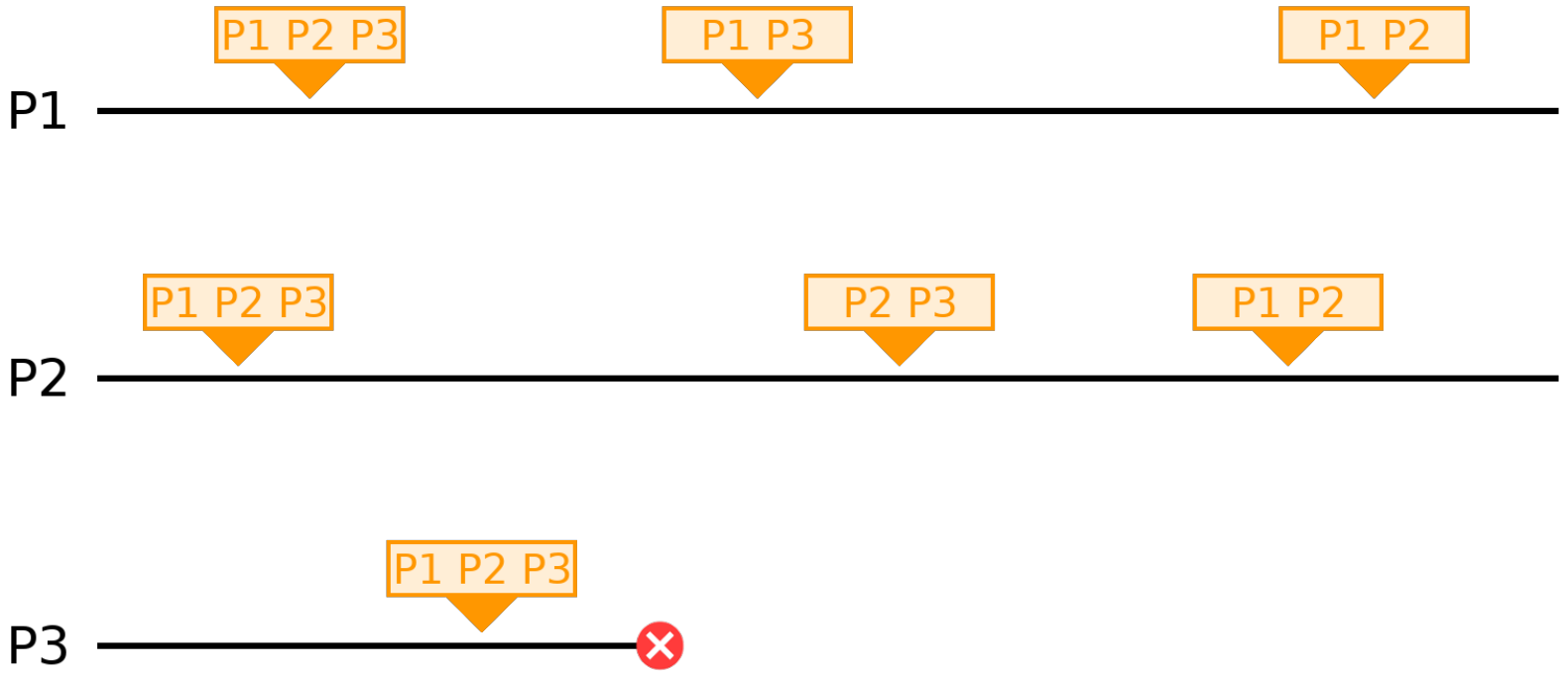
Les processus ne savent pas quand l'état stable est atteint !.

Ω est **équivalent** à $\diamond S$.

Le détecteur de quorums Σ



Le détecteur de quorums Σ



- **Complétude forte**
- **Intersection** : les listes fournies par Σ s'intersectent deux à deux

Détecteurs de fautes minimaux

	Consensus	k-accord	Exclusion mutuelle
Majorité de corrects	Ω	$\overline{\Omega}_k$	$\mathcal{T}$
Jusqu'à $n-1$ fautes	(Ω, Σ)	?	$(\mathcal{T}, \Sigma')$

Algorithme de consensus avec détecteur P

Hypothèse : détecteur P

Algorithme de consensus avec détecteur P

Hypothèse : détecteur P

- Rondes asynchrones

Algorithme de consensus avec détecteur P

Hypothèse : détecteur P

- Rondes asynchrones
- 1 leader différent à chaque ronde
  id leader = numéro ronde

Algorithme de consensus avec détecteur P

Hypothèse : détecteur P

- Rondes asynchrones
- 1 leader différent à chaque ronde
  id leader = numéro ronde

À chaque ronde :

- Leader **décide** sa valeur et la diffuse à tous

Hypothèse : détecteur P

- Rondes asynchrones
- 1 leader différent à chaque ronde
  id leader = numéro ronde

À chaque ronde :

- Leader **décide** sa valeur et la diffuse à tous
- Chaque autre processus attend :
  - Soit de recevoir la valeur du leader (puis adopte la valeur du leader)

Algorithme de consensus avec détecteur P

Hypothèse : détecteur P

- Rondes asynchrones
- 1 leader différent à chaque ronde
  id leader = numéro ronde

À chaque ronde :

- Leader **décide** sa valeur et la diffuse à tous
- Chaque autre processus attend :
  - Soit de recevoir la valeur du leader (puis adopte la valeur du leader)
  - Soit le crash du leader

Hypothèse : détecteur P

- Rondes asynchrones
- 1 leader différent à chaque ronde
  id leader = numéro ronde

À chaque ronde :

- Leader **décide** sa valeur et la diffuse à tous
- Chaque autre processus attend :
  - Soit de recevoir la valeur du leader (puis adopte la valeur du leader)
  - Soit le crash du leader

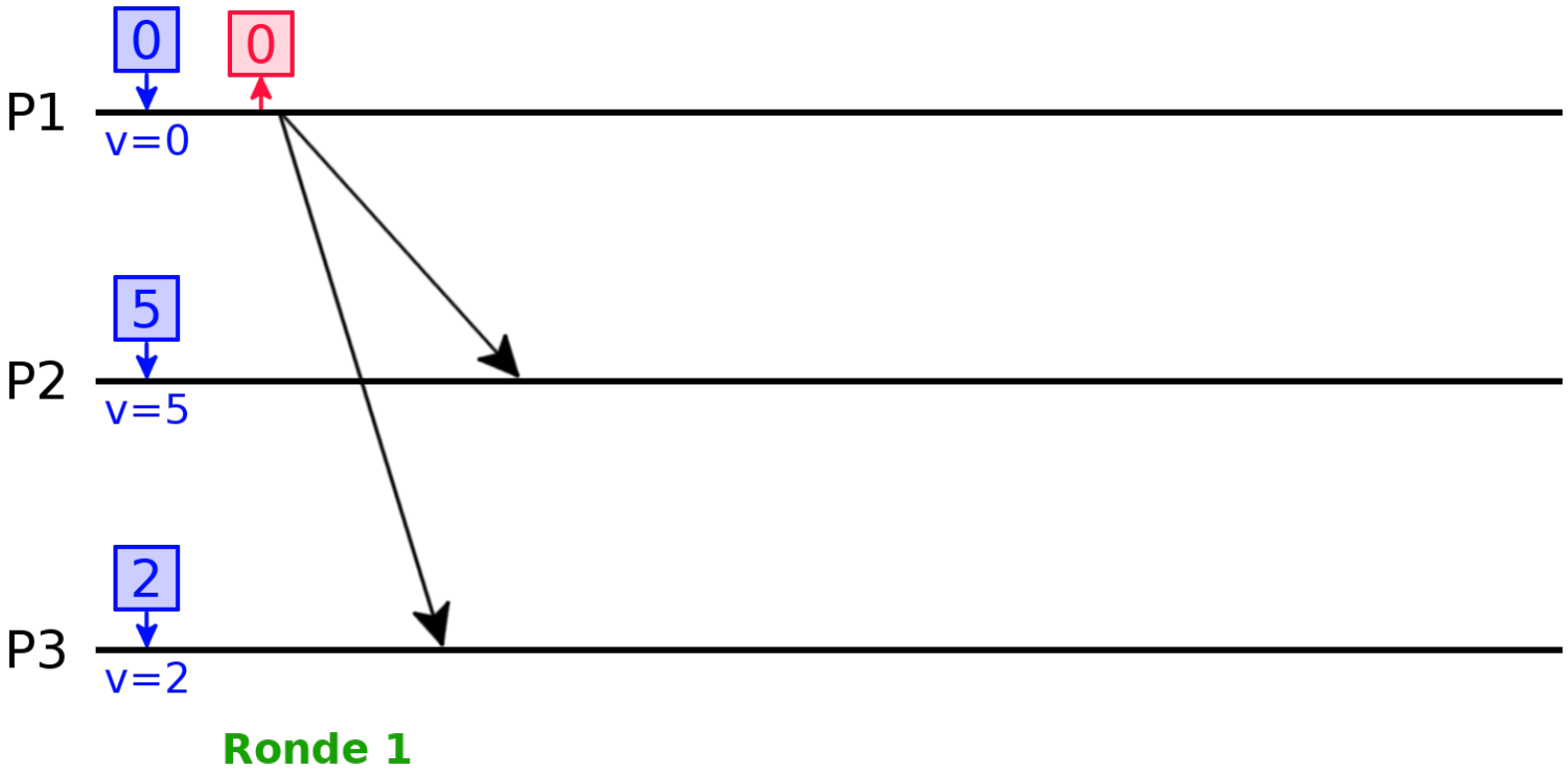
Tous les corrects décident en n rondes.

Algorithme de consensus avec détecteur P

Exemple 1

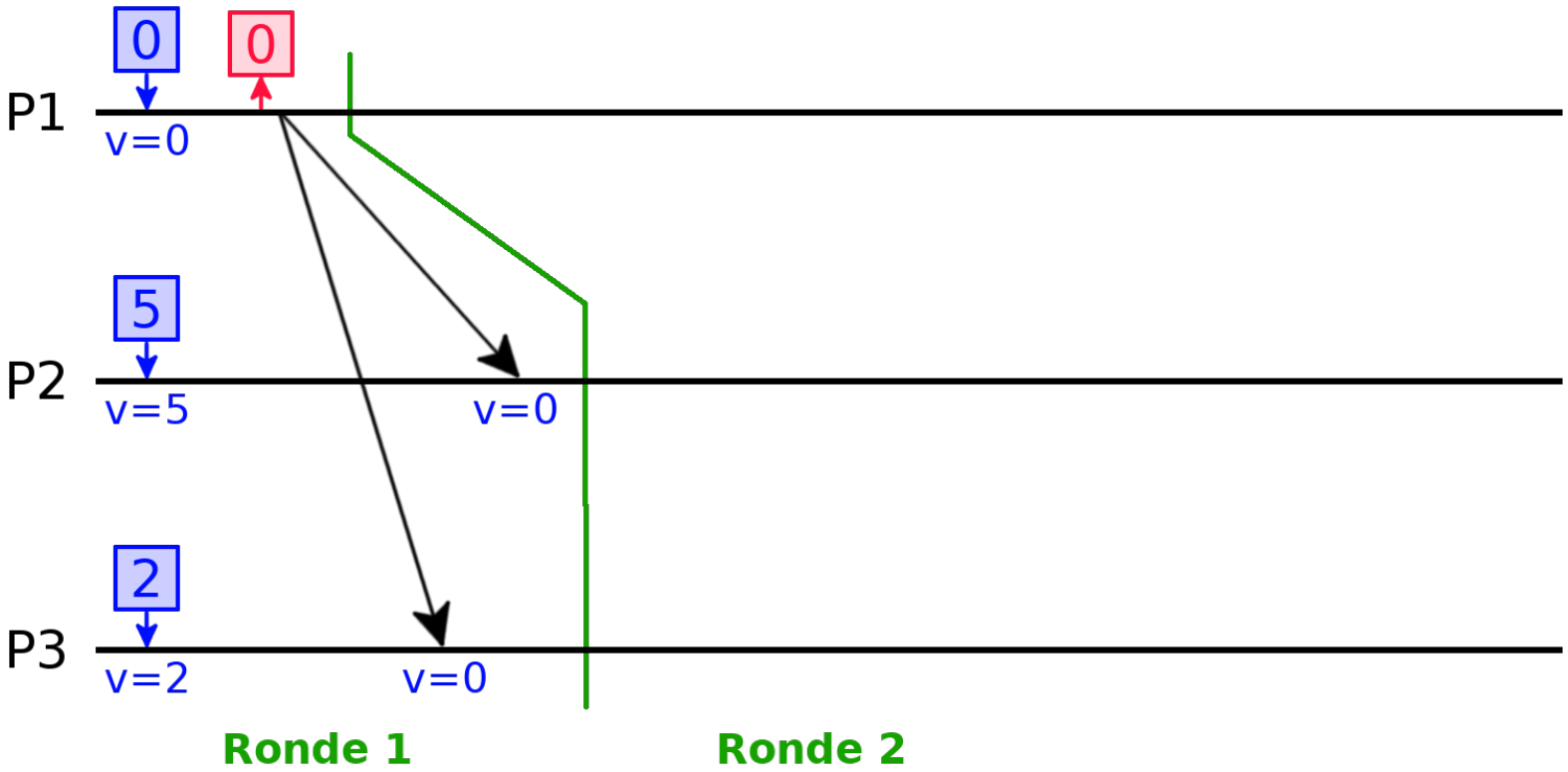
Algorithme de consensus avec détecteur P

Exemple 1



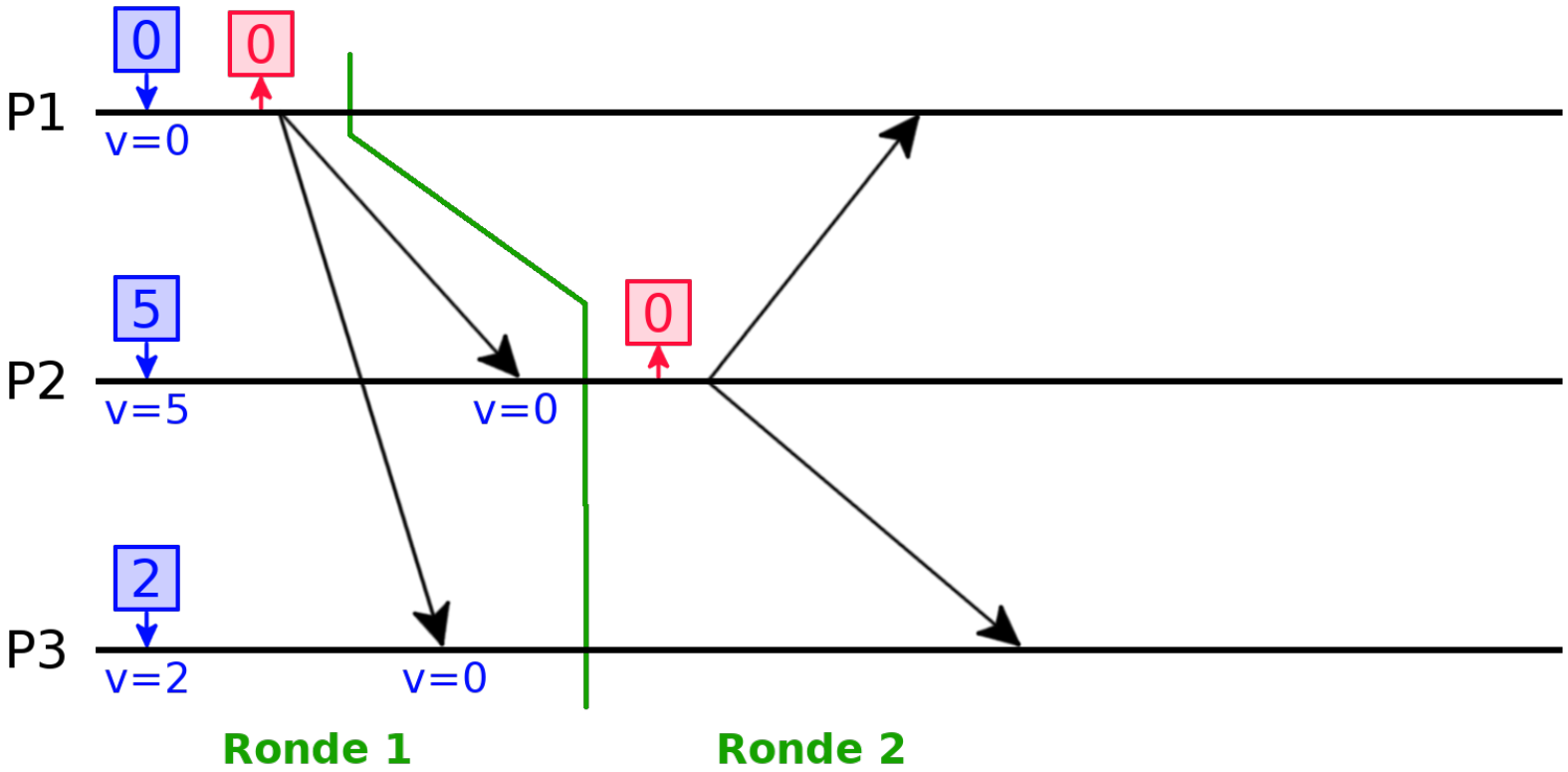
Algorithme de consensus avec détecteur P

Exemple 1



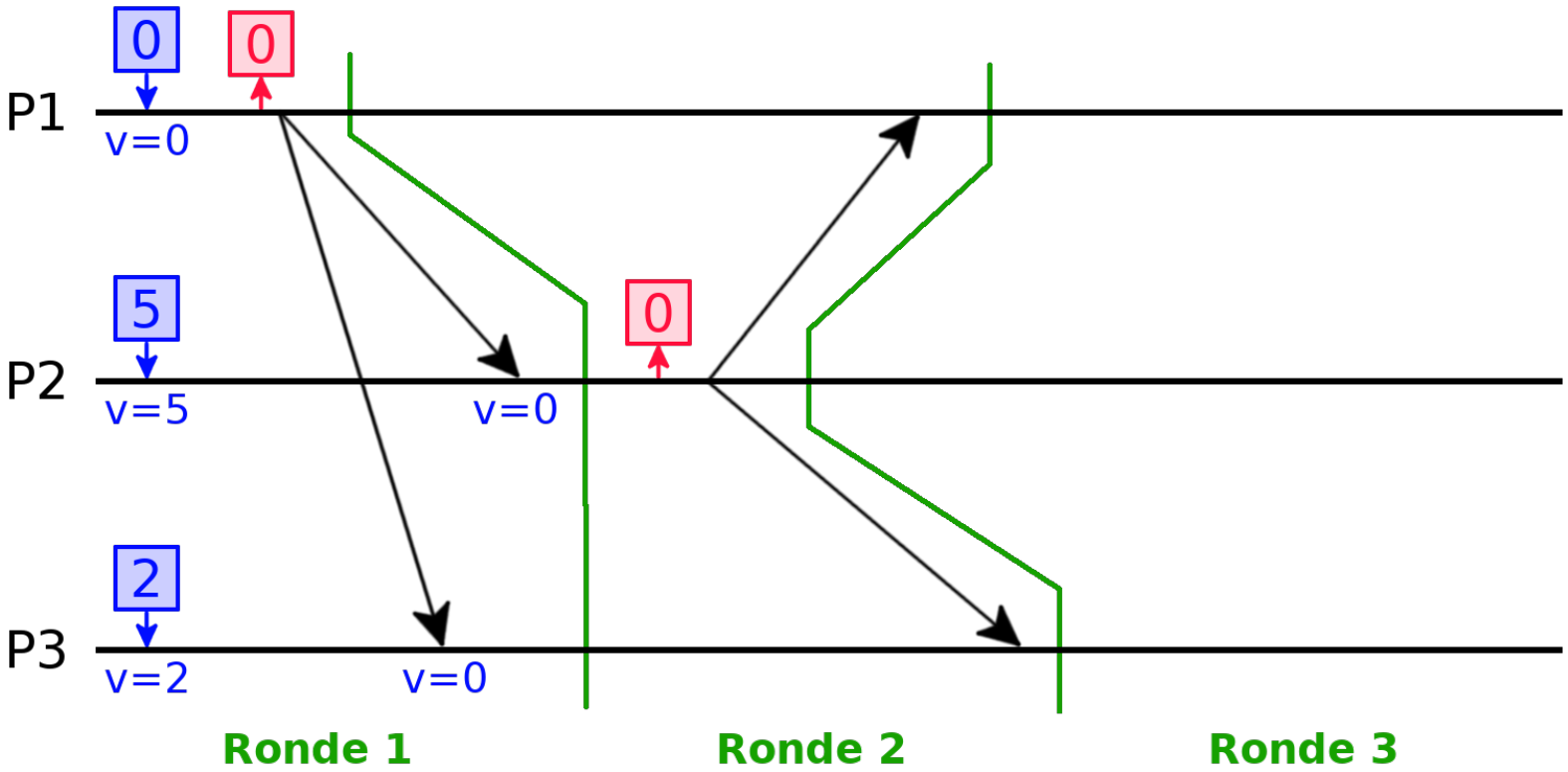
Algorithme de consensus avec détecteur P

Exemple 1



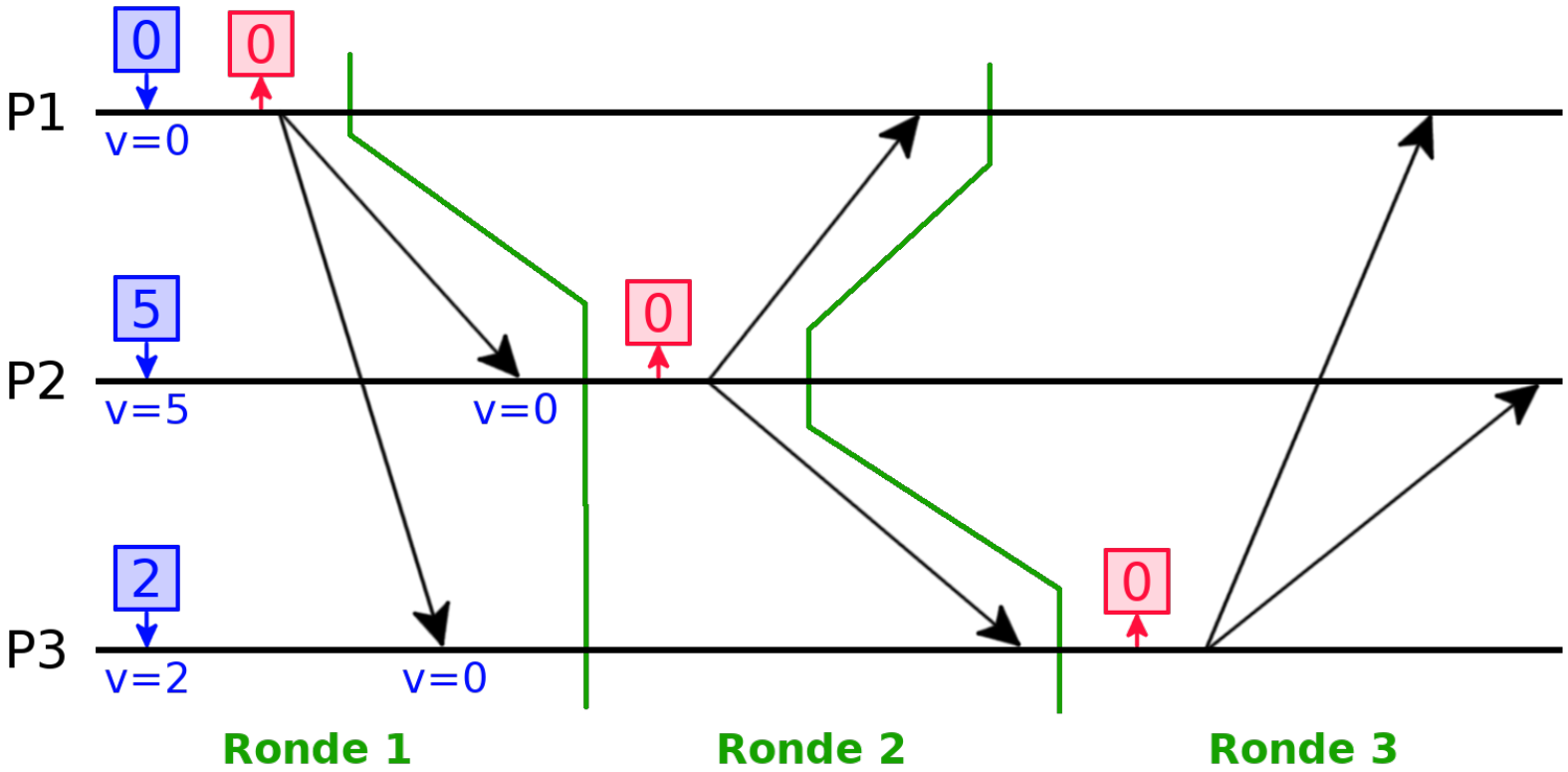
Algorithme de consensus avec détecteur P

Exemple 1



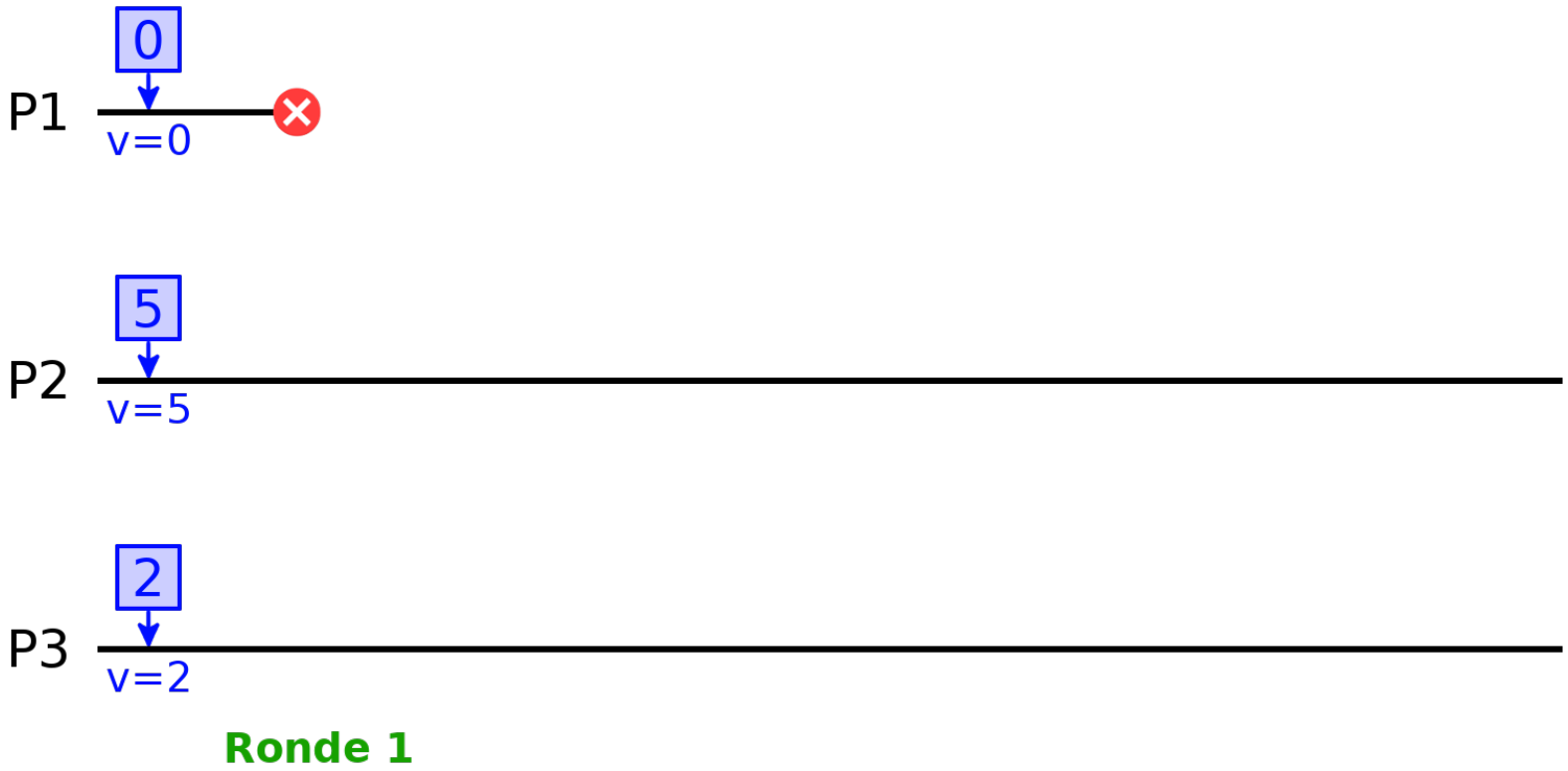
Algorithme de consensus avec détecteur P

Exemple 1



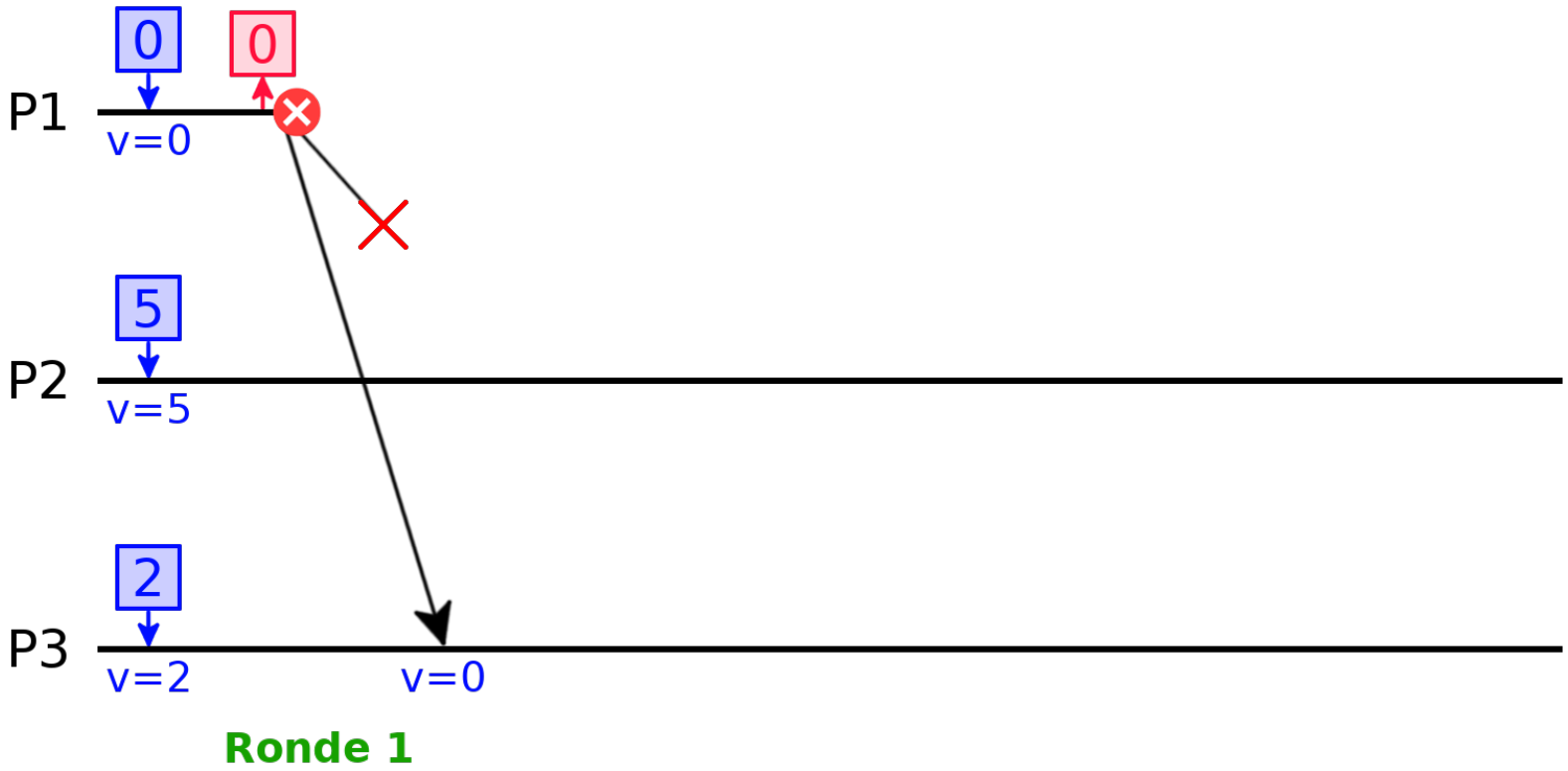
Algorithme de consensus avec détecteur P

Exemple 2



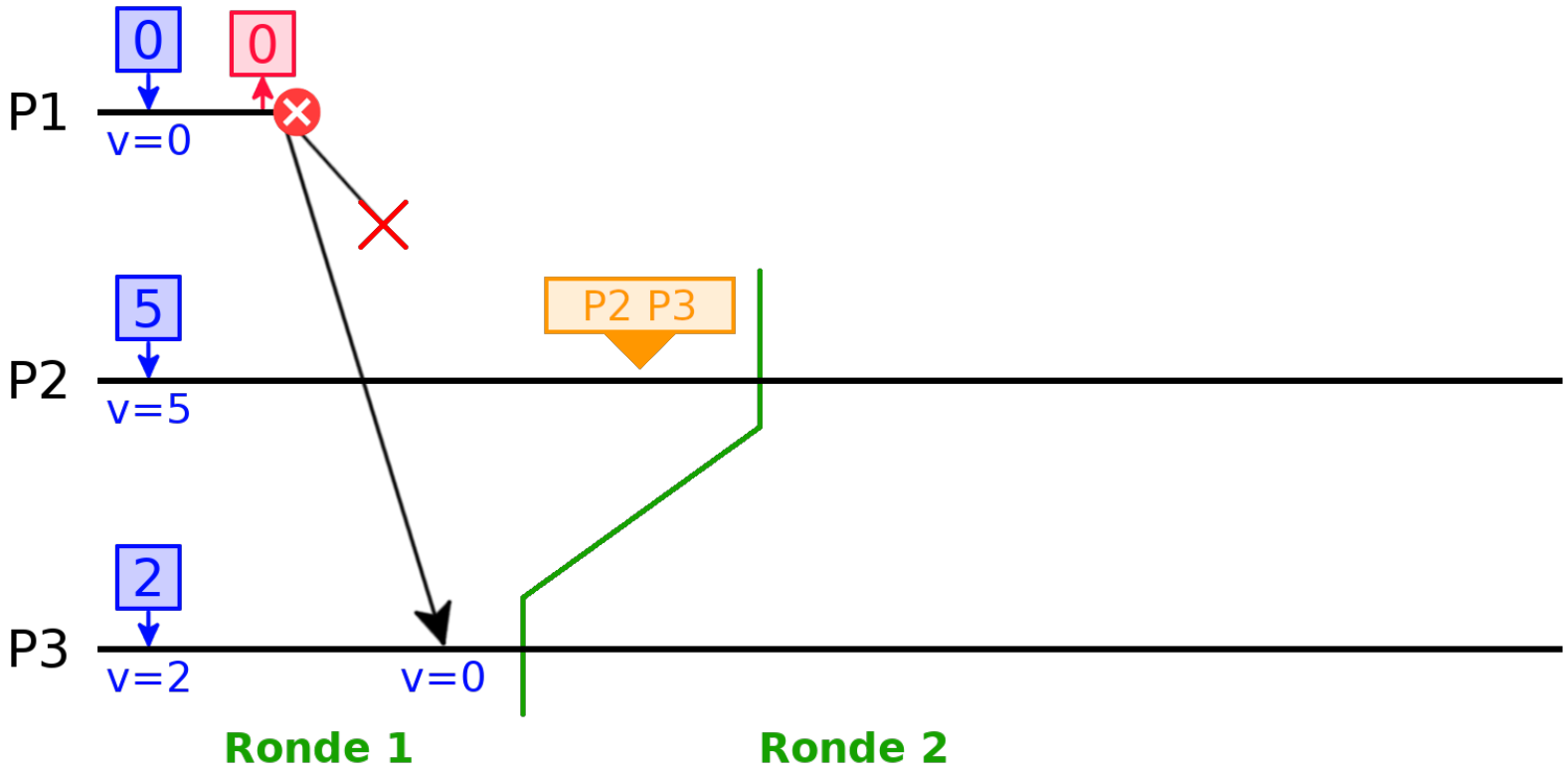
Algorithme de consensus avec détecteur P

Exemple 2



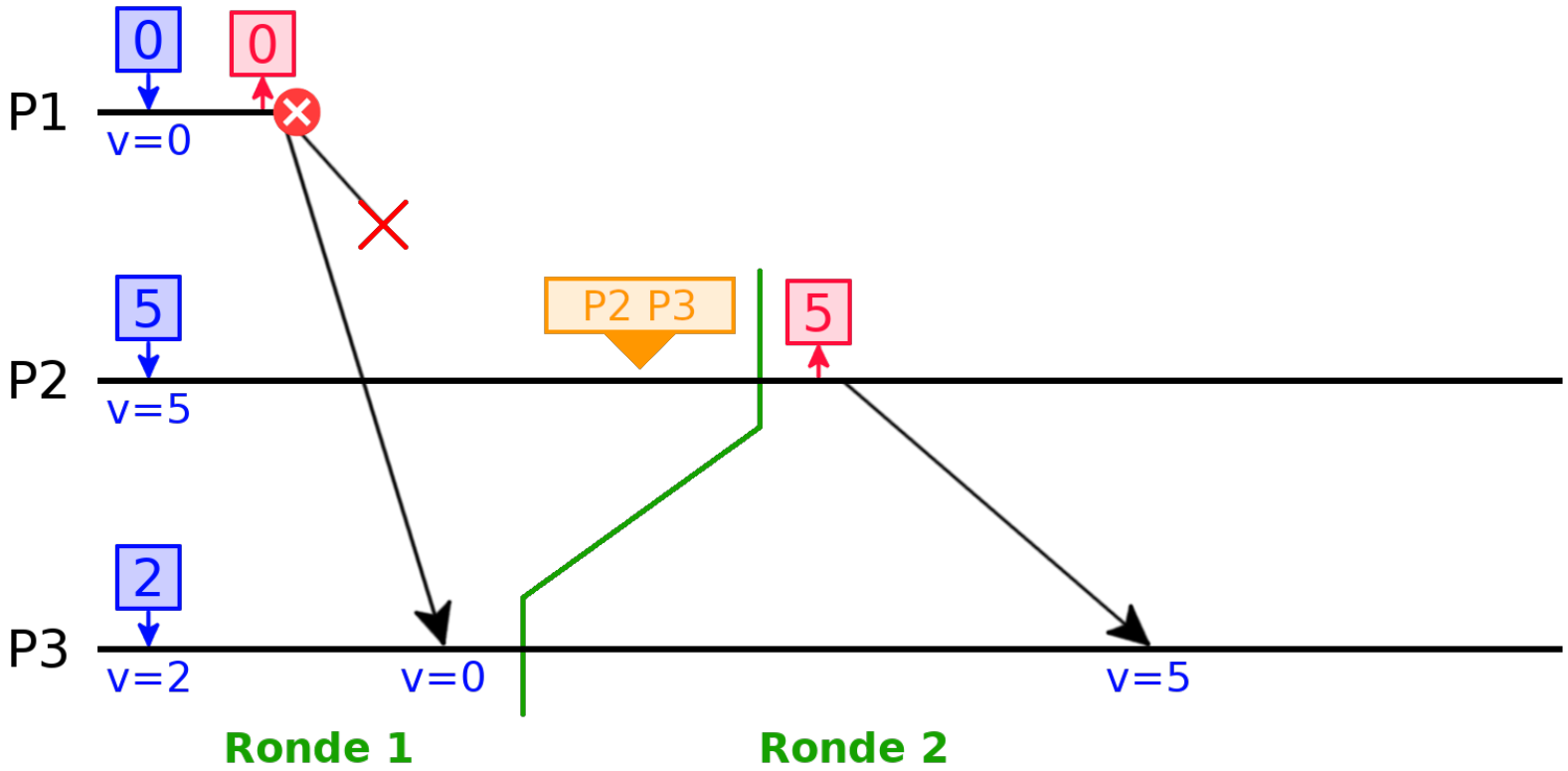
Algorithme de consensus avec détecteur P

Exemple 2



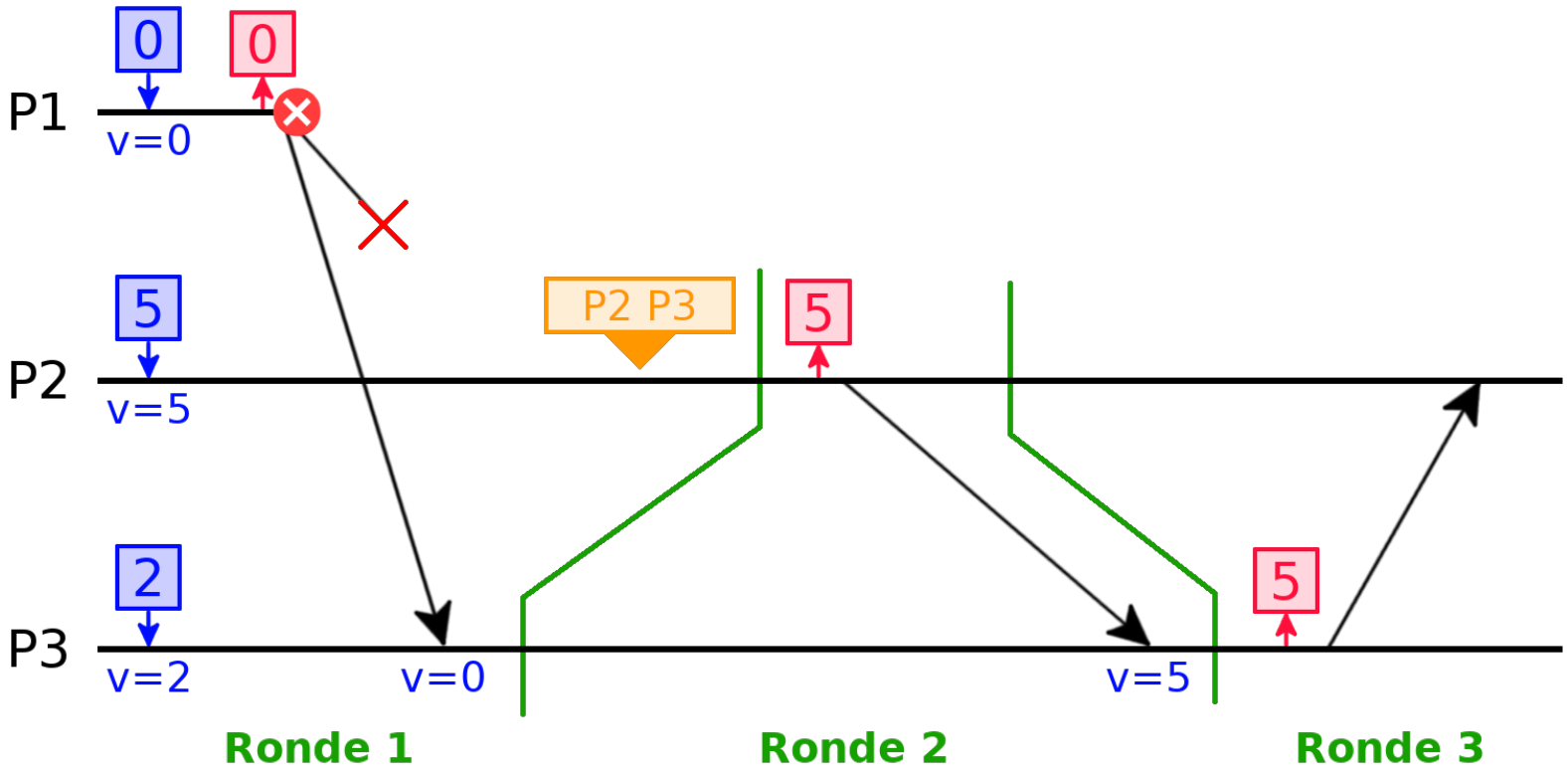
Algorithme de consensus avec détecteur P

Exemple 2



Algorithme de consensus avec détecteur P

Exemple 2



Un autre algorithme de consensus avec détecteur P

Hypothèse : détecteur P

Un autre algorithme de consensus avec détecteur P

Hypothèse : détecteur P

- f = nombre maximum de fautes tolérées

Un autre algorithme de consensus avec détecteur P

Hypothèse : détecteur P

- f = nombre maximum de fautes tolérées
- Chaque processus maintient un vecteur contenant les valeurs proposées

Hypothèse : détecteur P

- f = nombre maximum de fautes tolérées
- Chaque processus maintient un vecteur contenant les valeurs proposées

$f + 1$ rondes :

- Chaque processus diffuse son vecteur

Hypothèse : détecteur P

- f = nombre maximum de fautes tolérées
- Chaque processus maintient un vecteur contenant les valeurs proposées

$f + 1$ rondes :

- Chaque processus diffuse son vecteur
- Attend la réception des vecteurs de tous les processus non suspectés

Hypothèse : détecteur P

- f = nombre maximum de fautes tolérées
- Chaque processus maintient un vecteur contenant les valeurs proposées

$f + 1$ rondes :

- Chaque processus diffuse son vecteur
- Attend la réception des vecteurs de tous les processus non suspectés

Après $f + 1$ rondes :

- Chaque processus **décide** la première valeur non vide de son vecteur

Hypothèse : détecteur P

- f = nombre maximum de fautes tolérées
- Chaque processus maintient un vecteur contenant les valeurs proposées

$f + 1$ rondes :

- Chaque processus diffuse son vecteur
- Attend la réception des vecteurs de tous les processus non suspectés

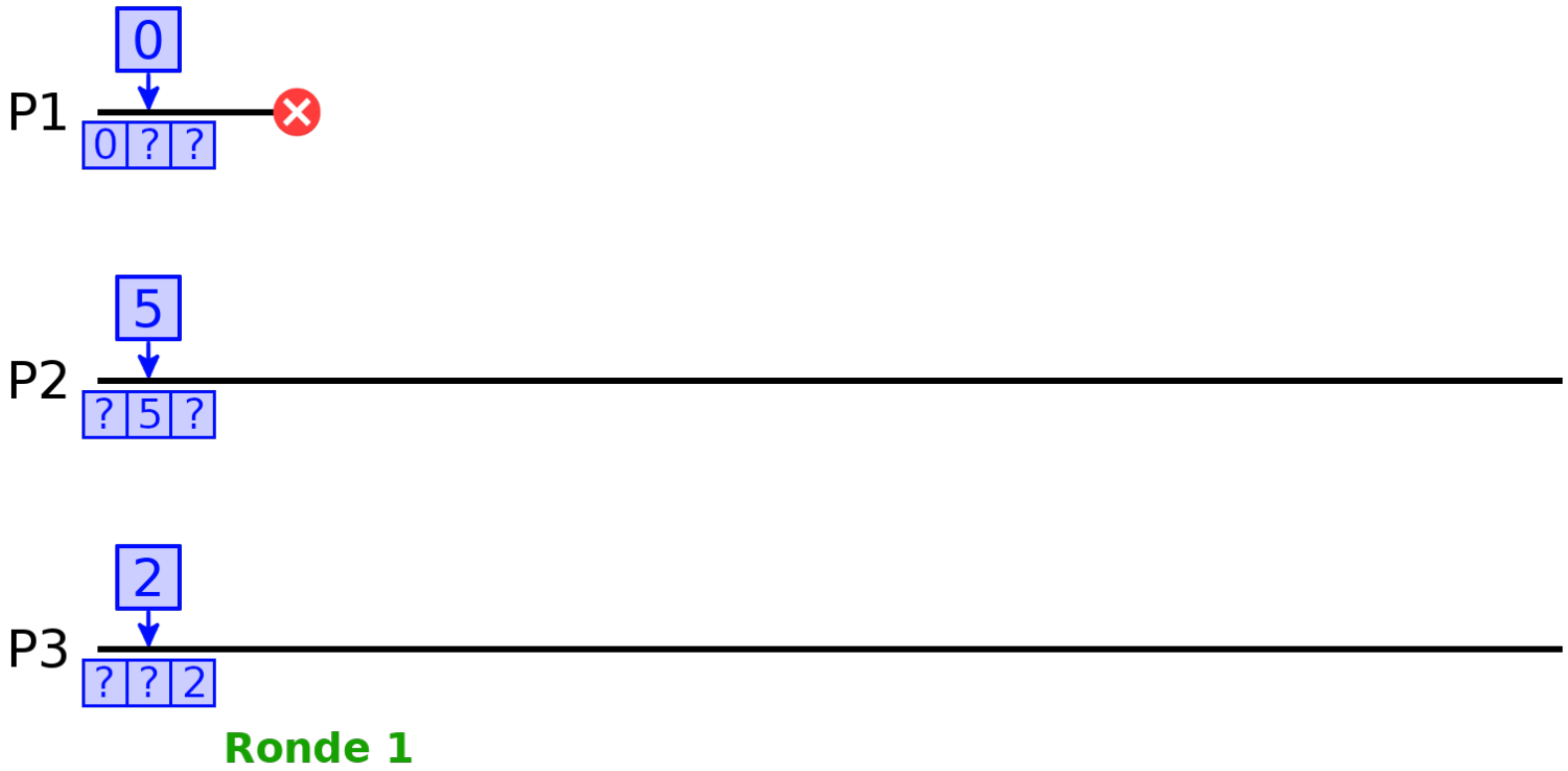
Après $f + 1$ rondes :

- Chaque processus **décide** la première valeur non vide de son vecteur

Tous les corrects décident en $f + 1$ rondes.

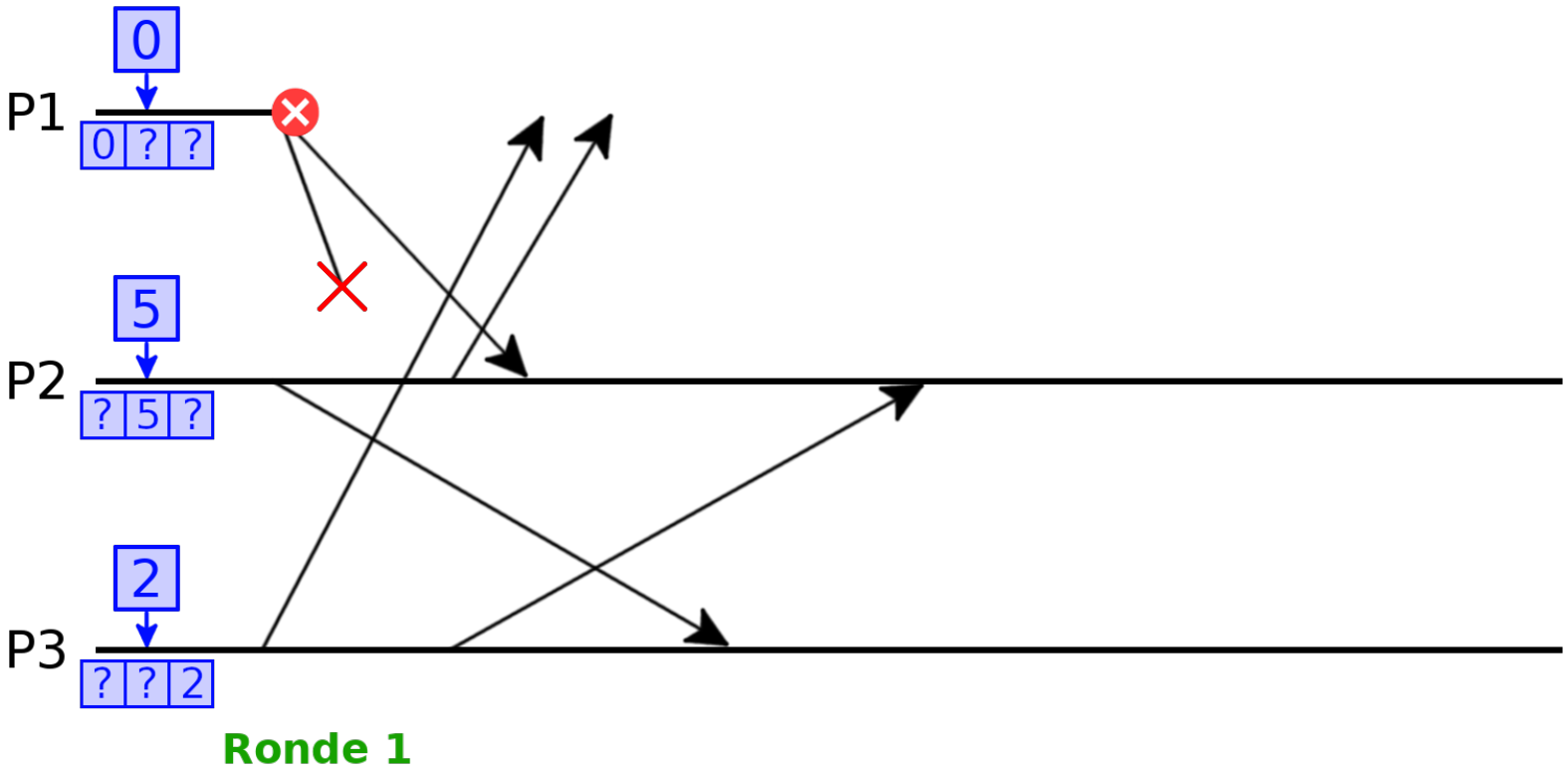
Un autre algorithme de consensus avec détecteur P

Exemple avec $f = 1$



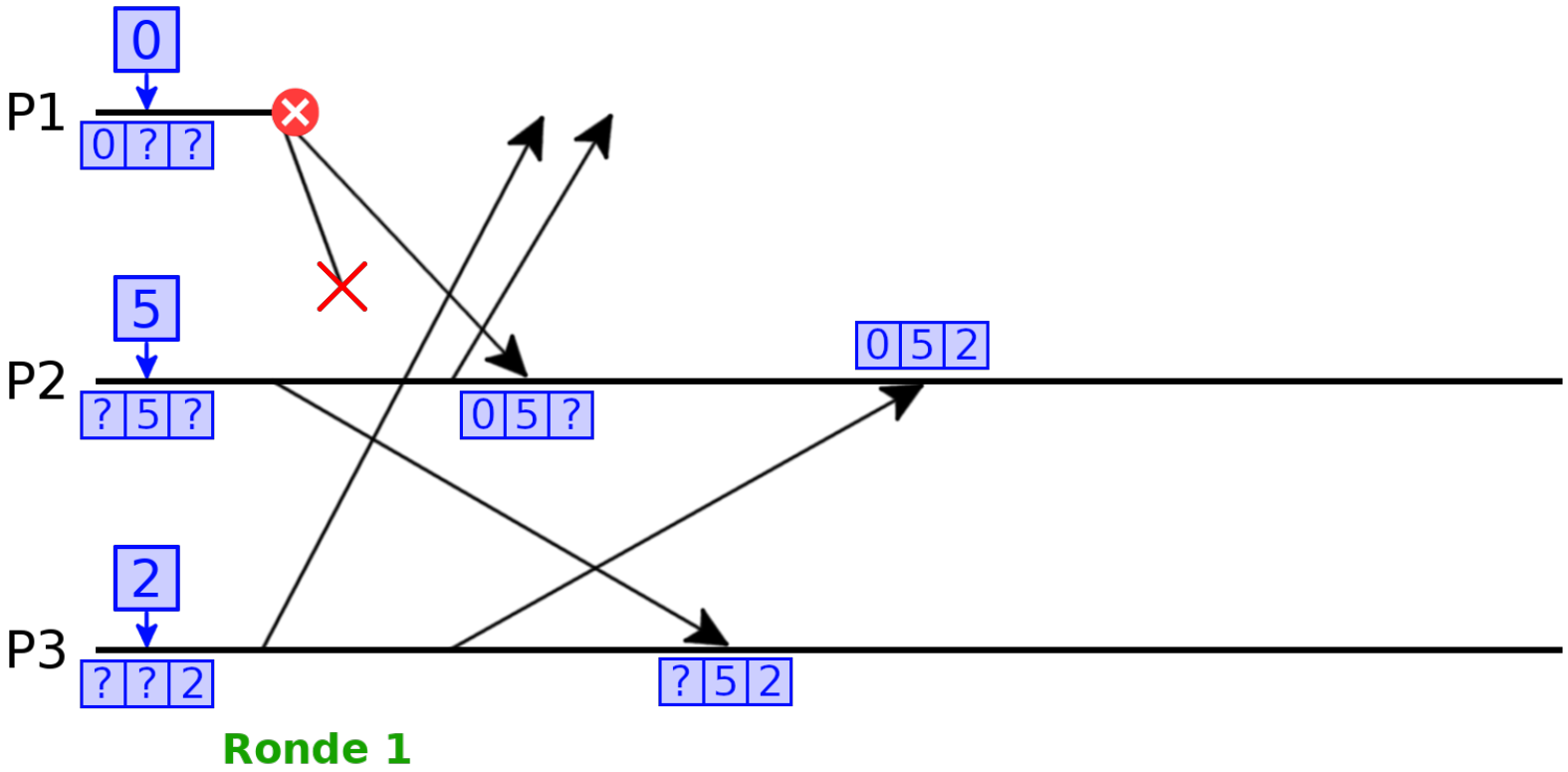
Un autre algorithme de consensus avec détecteur P

Exemple avec $f = 1$



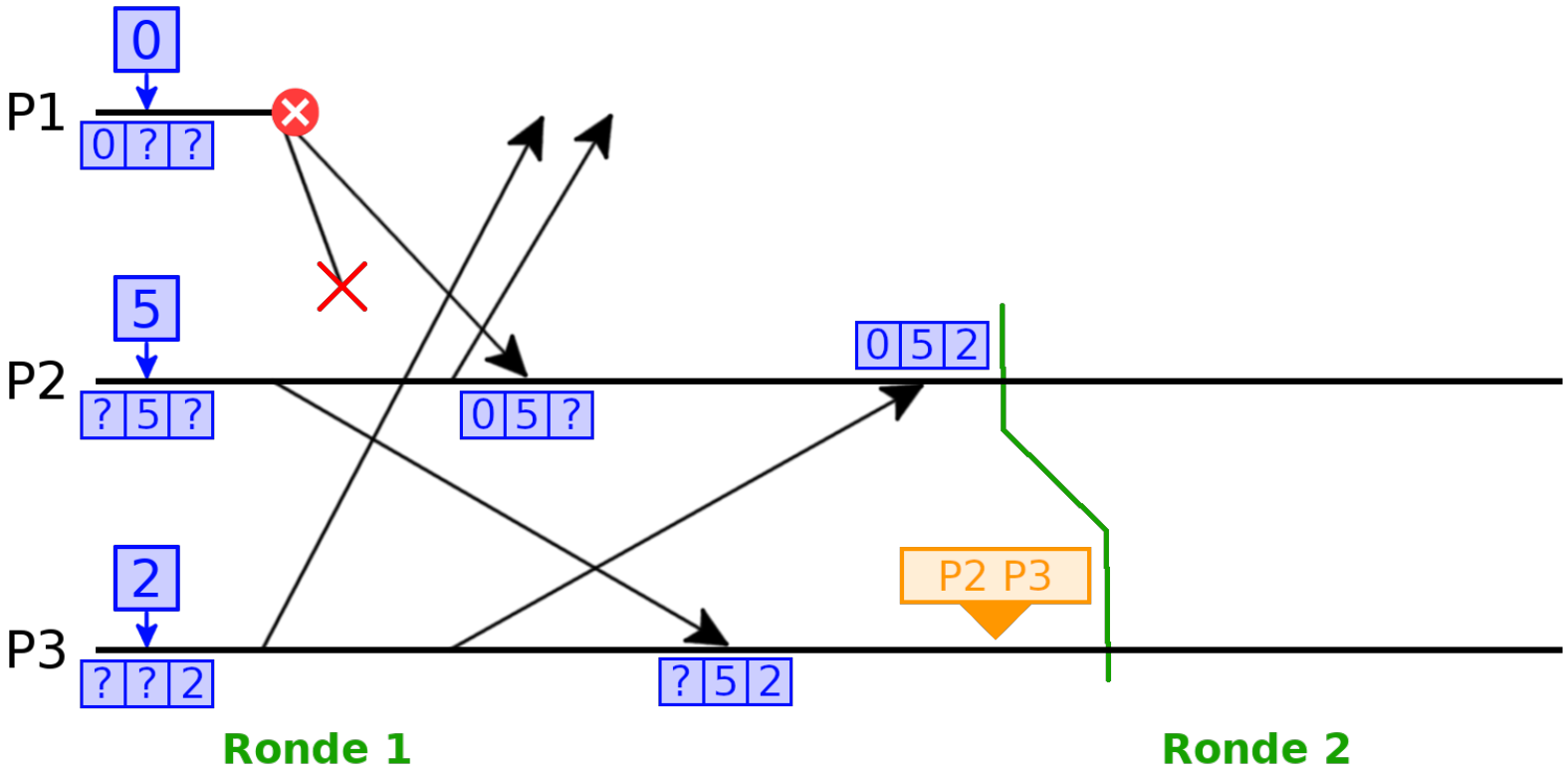
Un autre algorithme de consensus avec détecteur P

Exemple avec $f = 1$



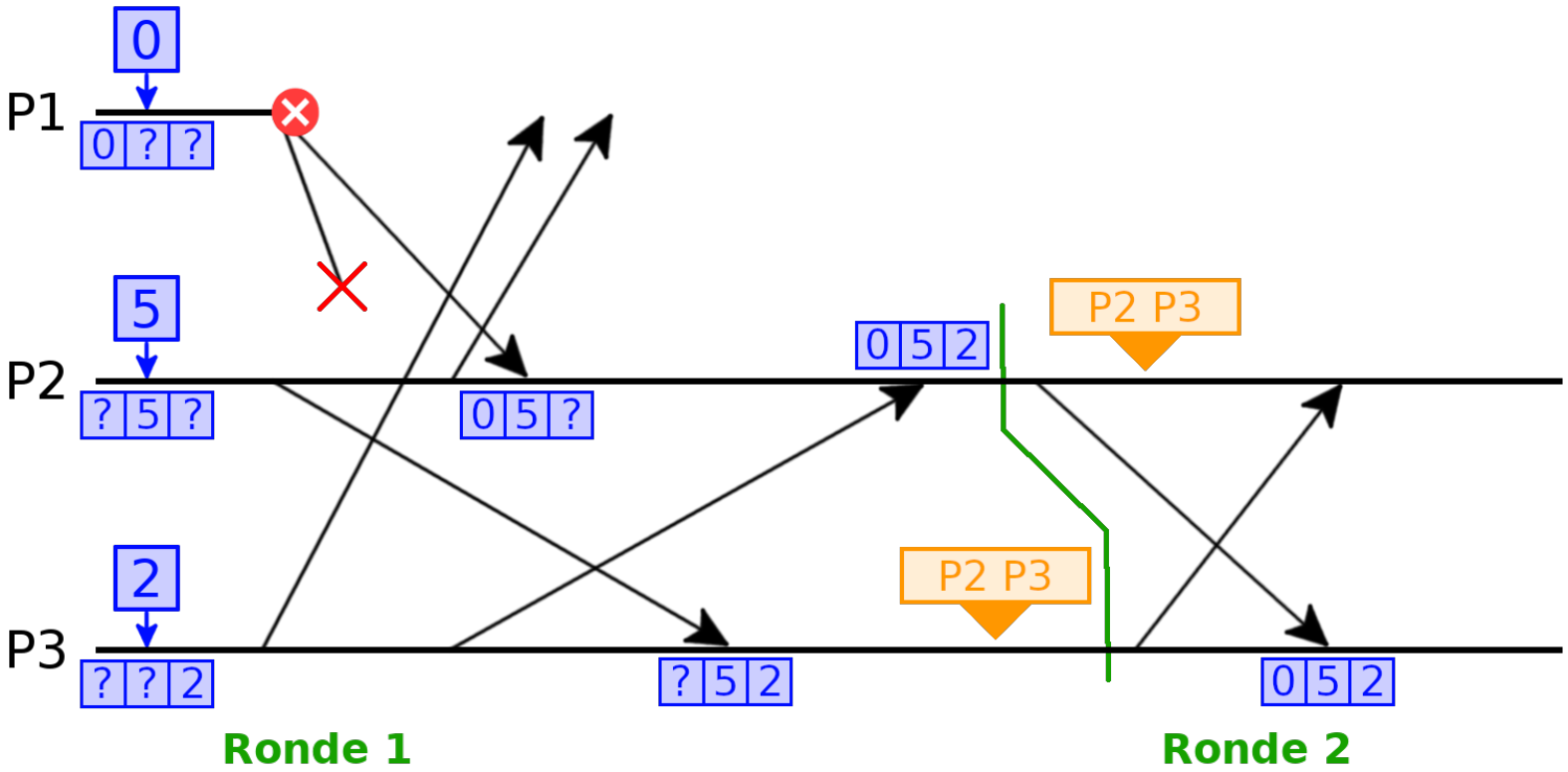
Un autre algorithme de consensus avec détecteur P

Exemple avec $f = 1$



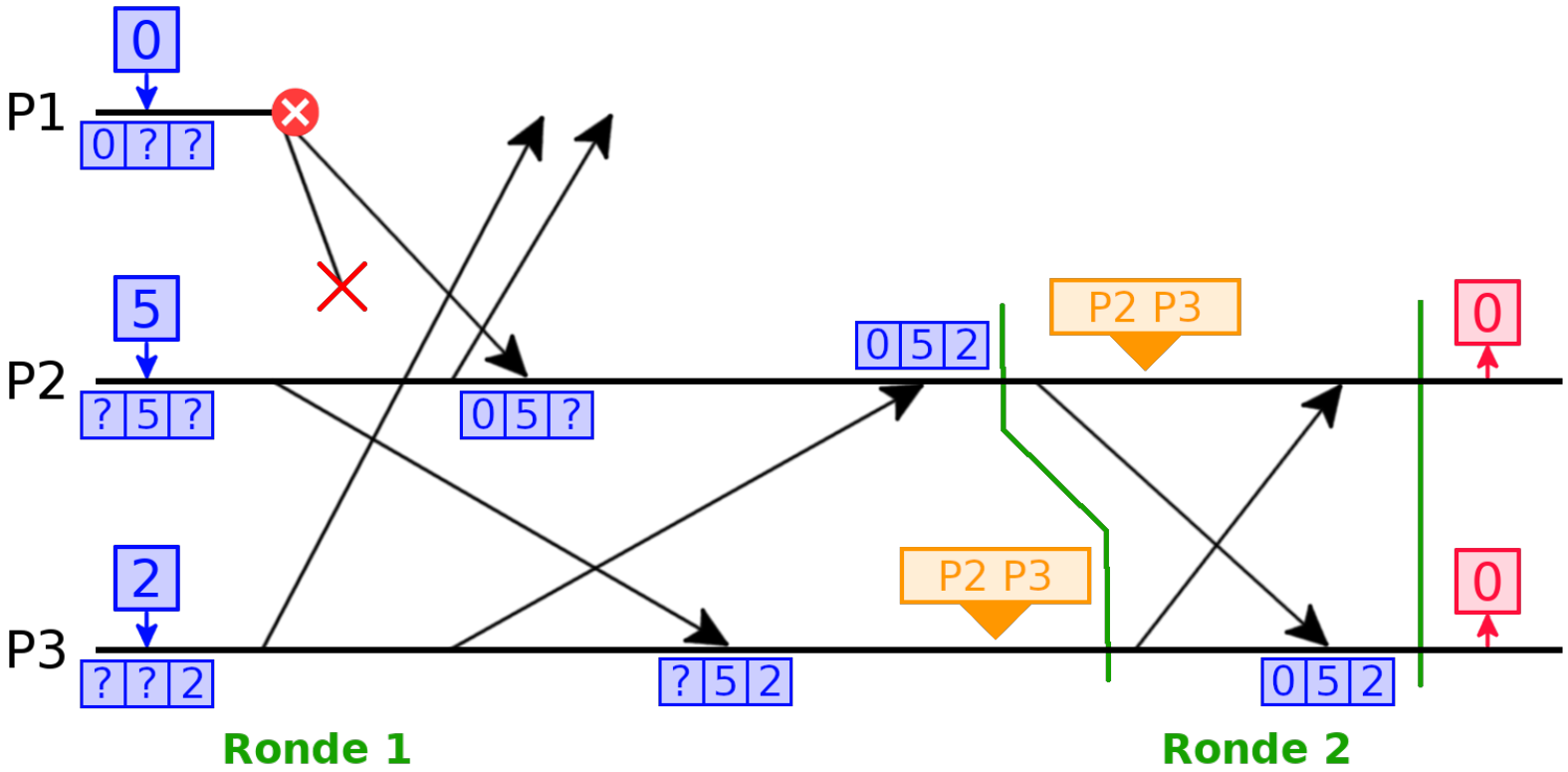
Un autre algorithme de consensus avec détecteur P

Exemple avec $f = 1$



Un autre algorithme de consensus avec détecteur P

Exemple avec $f = 1$



Algorithme du coordinateur tournant

Hypothèses :

- Majorité de corrects
- Détecteur $\diamond S$

Hypothèses :

- Majorité de corrects
- Détecteur $\diamond S$

1 coordinateur (leader) différent à chaque ronde
id coordinateur = numéro ronde $\%n$

Hypothèses :

- Majorité de corrects
- Détecteur $\diamond S$

1 coordinateur (leader) différent à chaque ronde
id coordinateur = numéro ronde $\%n$

Le coordinateur impose sa valeur, qui est choisie si il n'est pas suspecté.

- **Phase 1: estimation**
  Chaque processus envoie au coordinateur sa valeur courante.

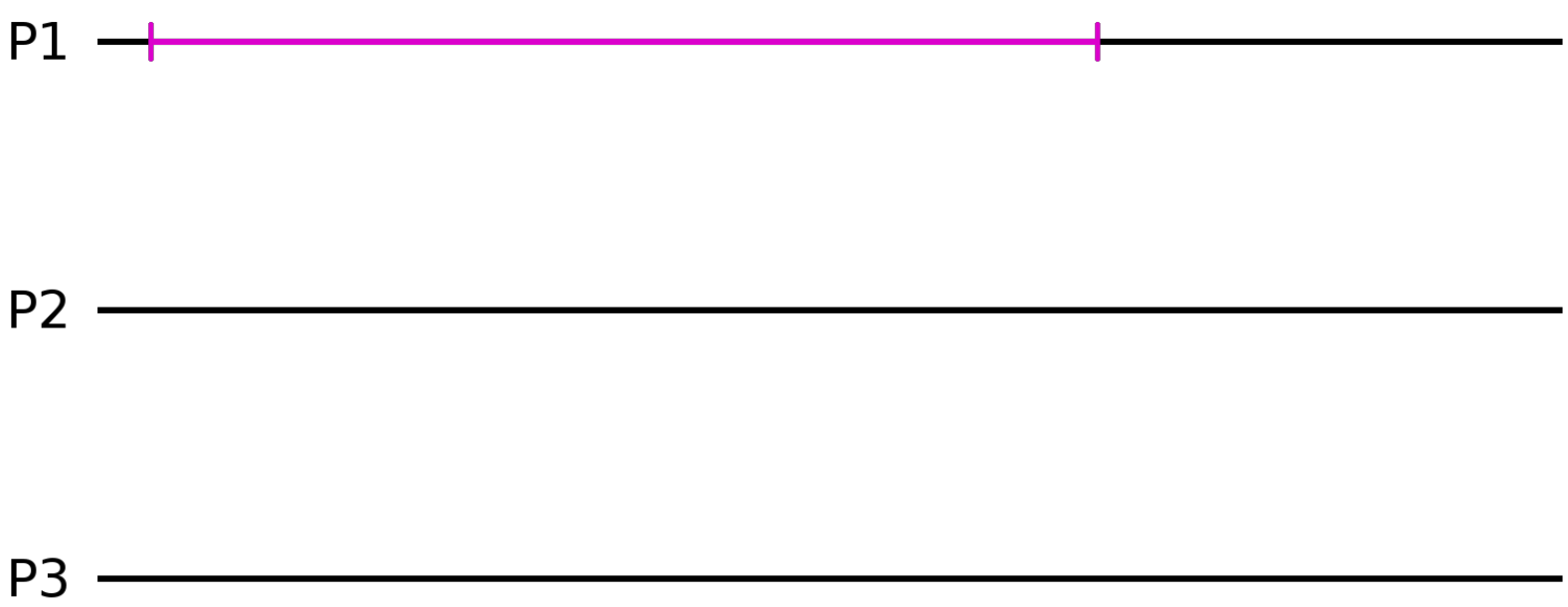
- Phase 1: estimation
  Chaque processus envoie au coordinateur sa valeur courante.
- **Phase 2: proposition**
  Le coordinateur attend une majorité de valeurs.
  Il choisit la plus à jour, puis la diffuse.

- **Phase 1: estimation**
  Chaque processus envoie au coordinateur sa valeur courante.
- **Phase 2: proposition**
  Le coordinateur attend une majorité de valeurs.
  Il choisit la plus à jour, puis la diffuse.
- **Phase 3: ack**
  Pour chaque correct:
  - Si réception de la valeur, renvoyer ack au coordinateur.
  - Si le coordinateur est soupçonné, renvoyer nack.

- **Phase 1: estimation**
  Chaque processus envoie au coordinateur sa valeur courante.
- **Phase 2: proposition**
  Le coordinateur attend une majorité de valeurs.
  Il choisit la plus à jour, puis la diffuse.
- **Phase 3: ack**
  Pour chaque correct:
  - Si réception de la valeur, renvoyer ack au coordinateur.
  - Si le coordinateur est soupçonné, renvoyer nack.
- **Phase 4: décision**
  Le coordinateur attend une majorité de réponses (ack ou nack).
  - Si majorité de ack, **décision** puis diffusion de la valeur.
    Les autres processus **décident** à la réception.
  - Sinon, on passe à la ronde suivante.

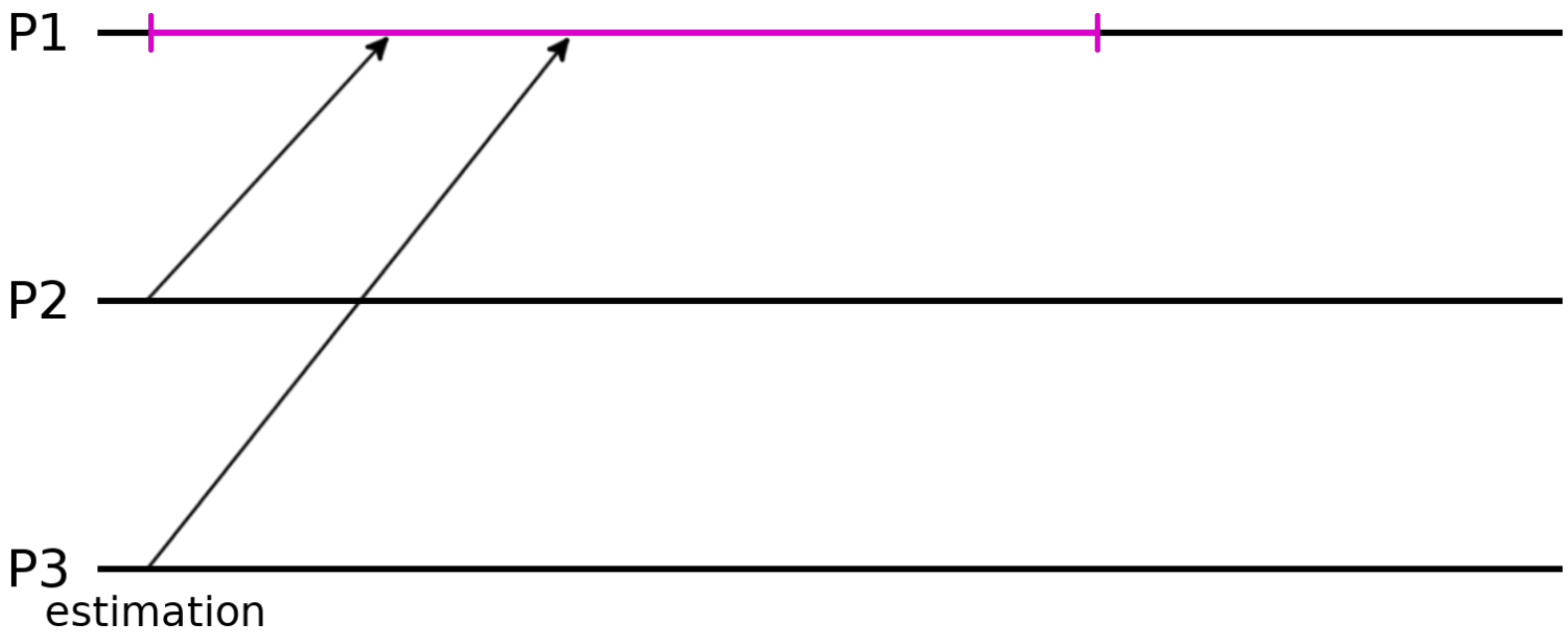
Algorithme du coordinateur tournant

Exemple 1



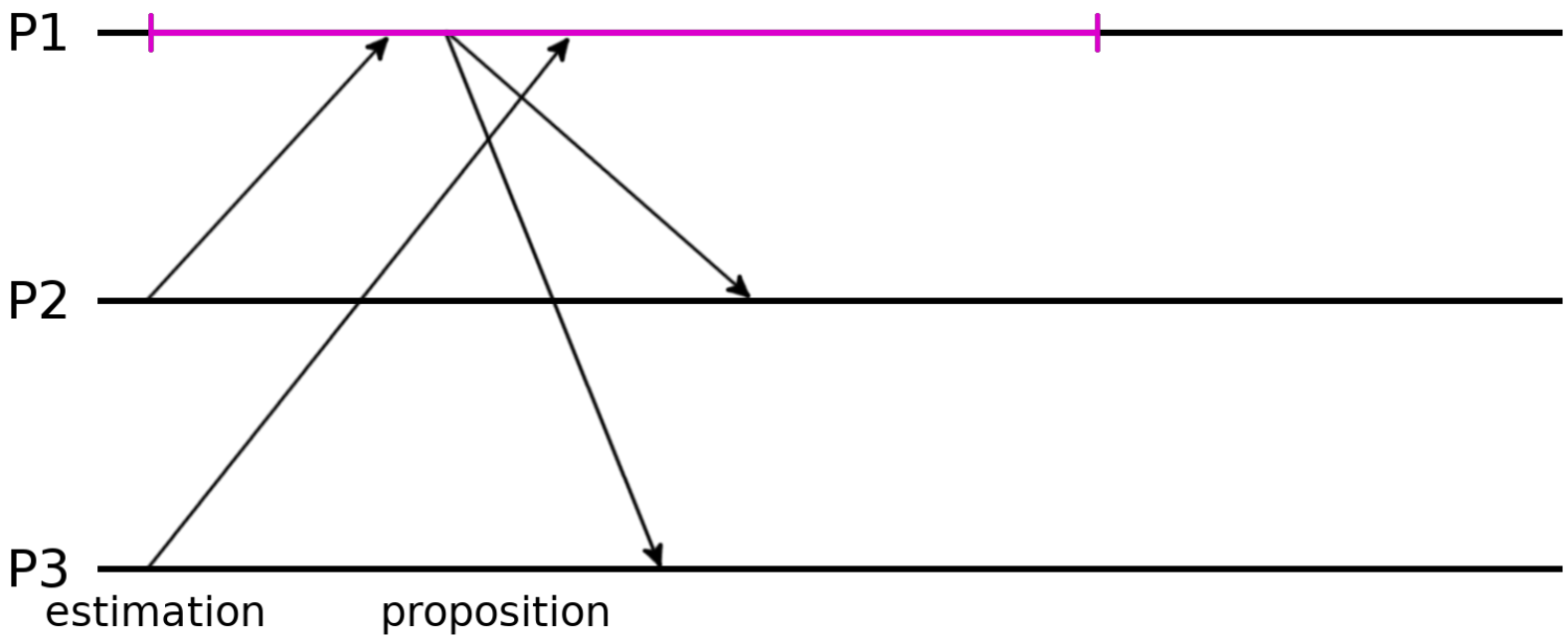
Algorithme du coordinateur tournant

Exemple 1



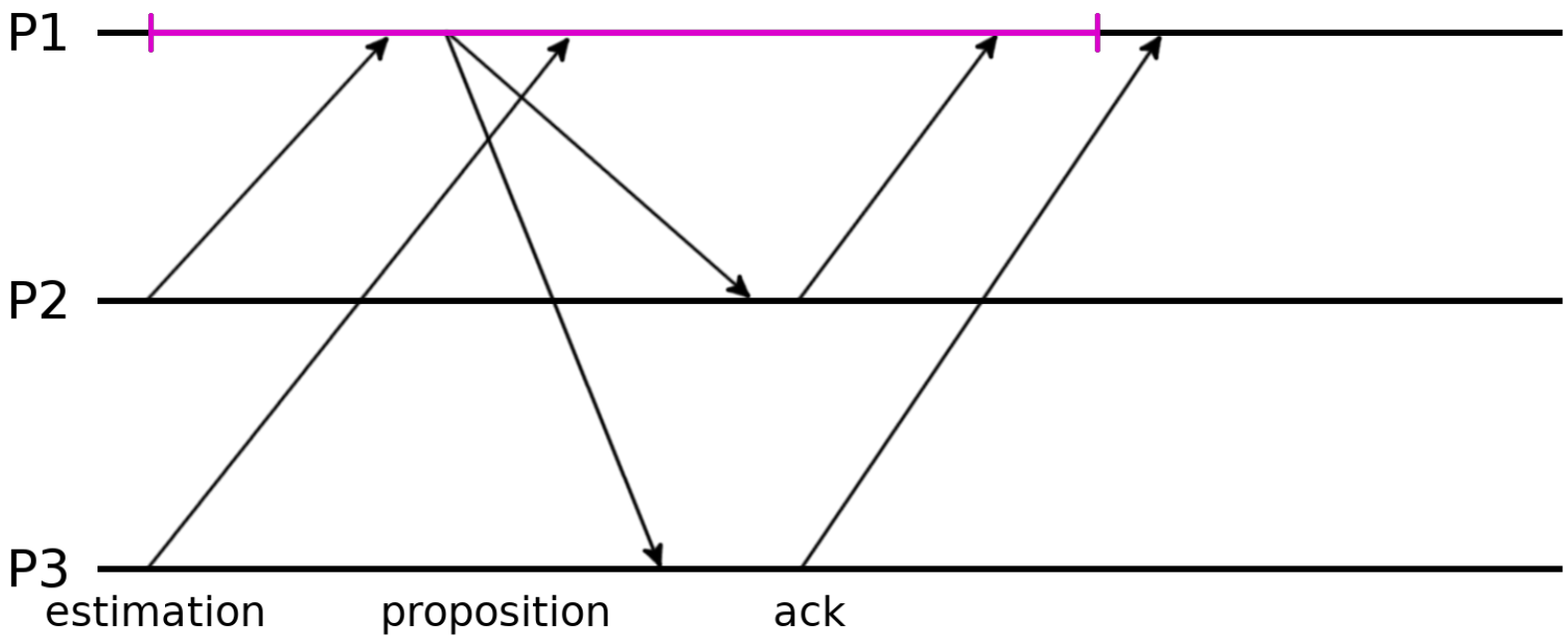
Algorithme du coordinateur tournant

Exemple 1



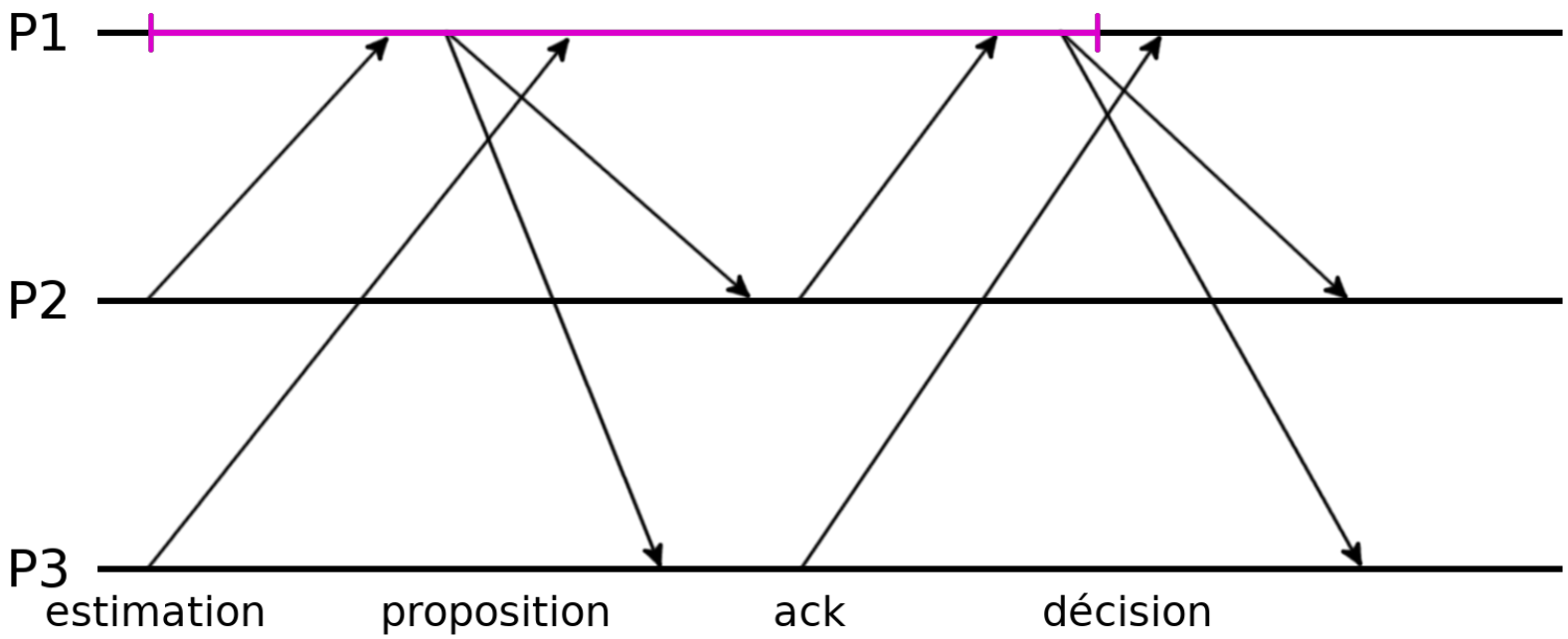
Algorithme du coordinateur tournant

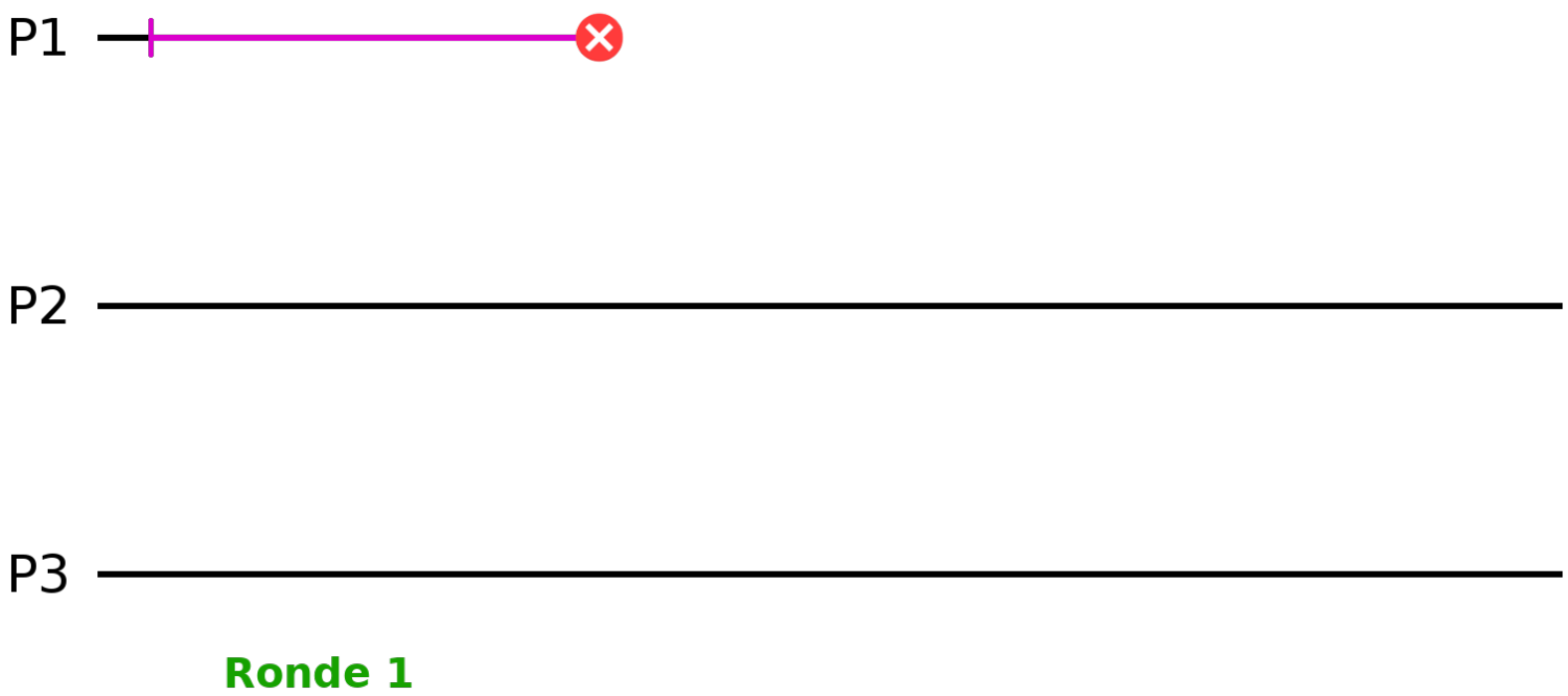
Exemple 1



Algorithme du coordinateur tournant

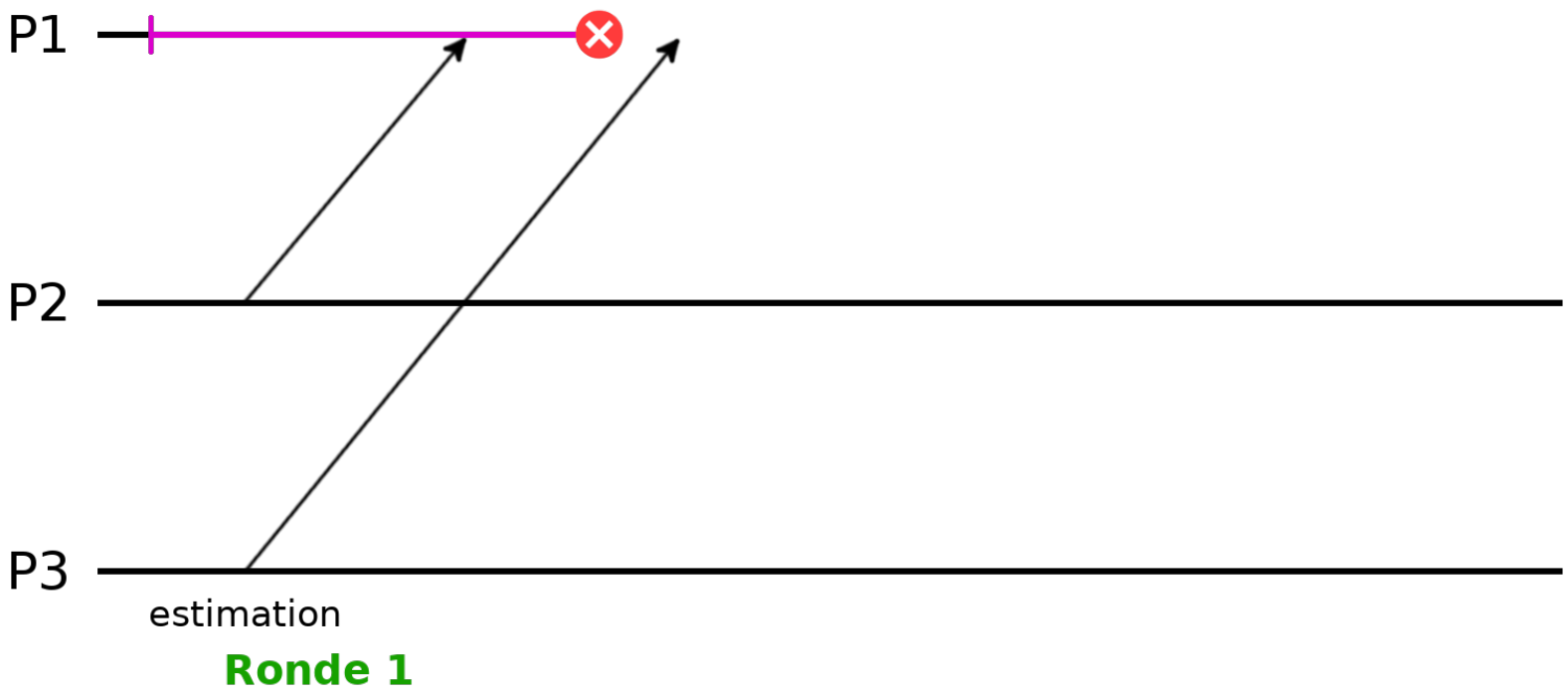
Exemple 1




P1
P2
P3
Ronde 1

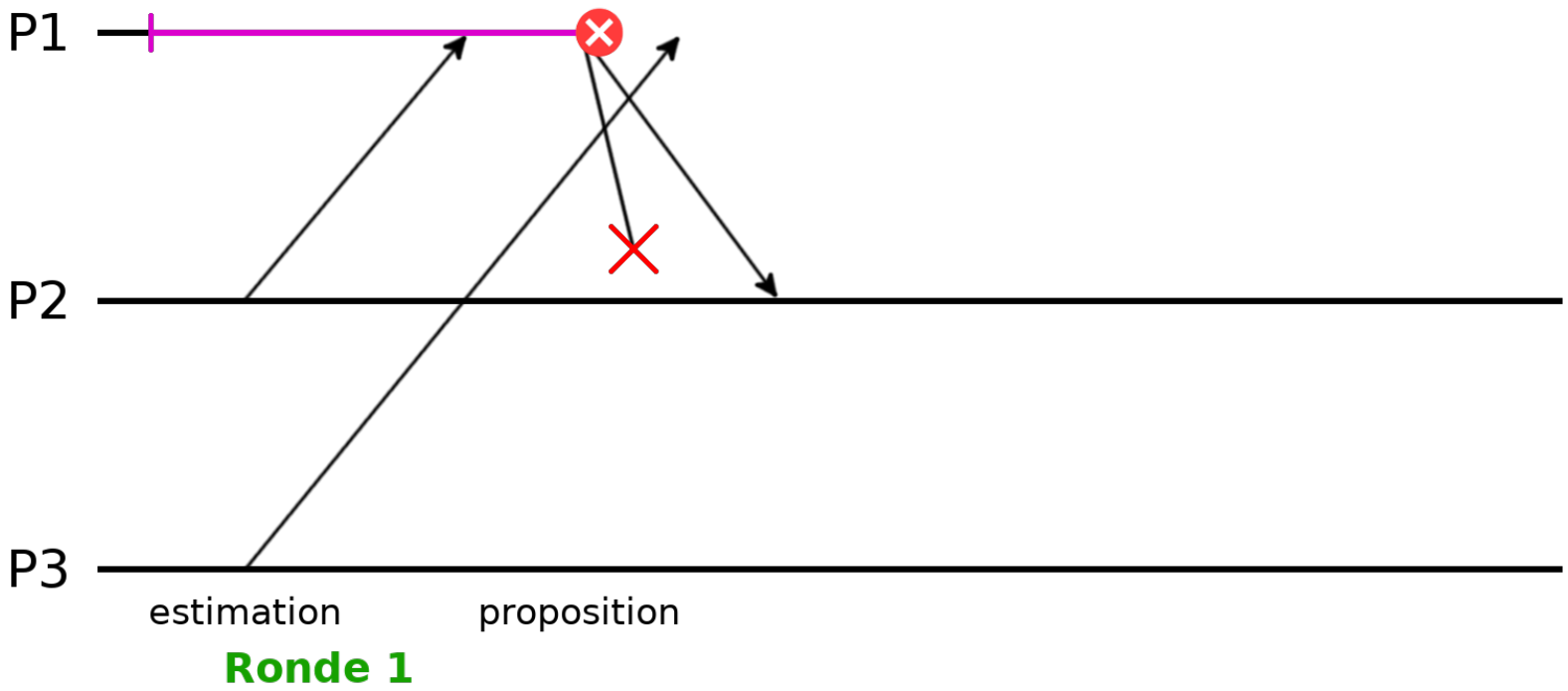
Algorithme du coordinateur tournant

Exemple 2



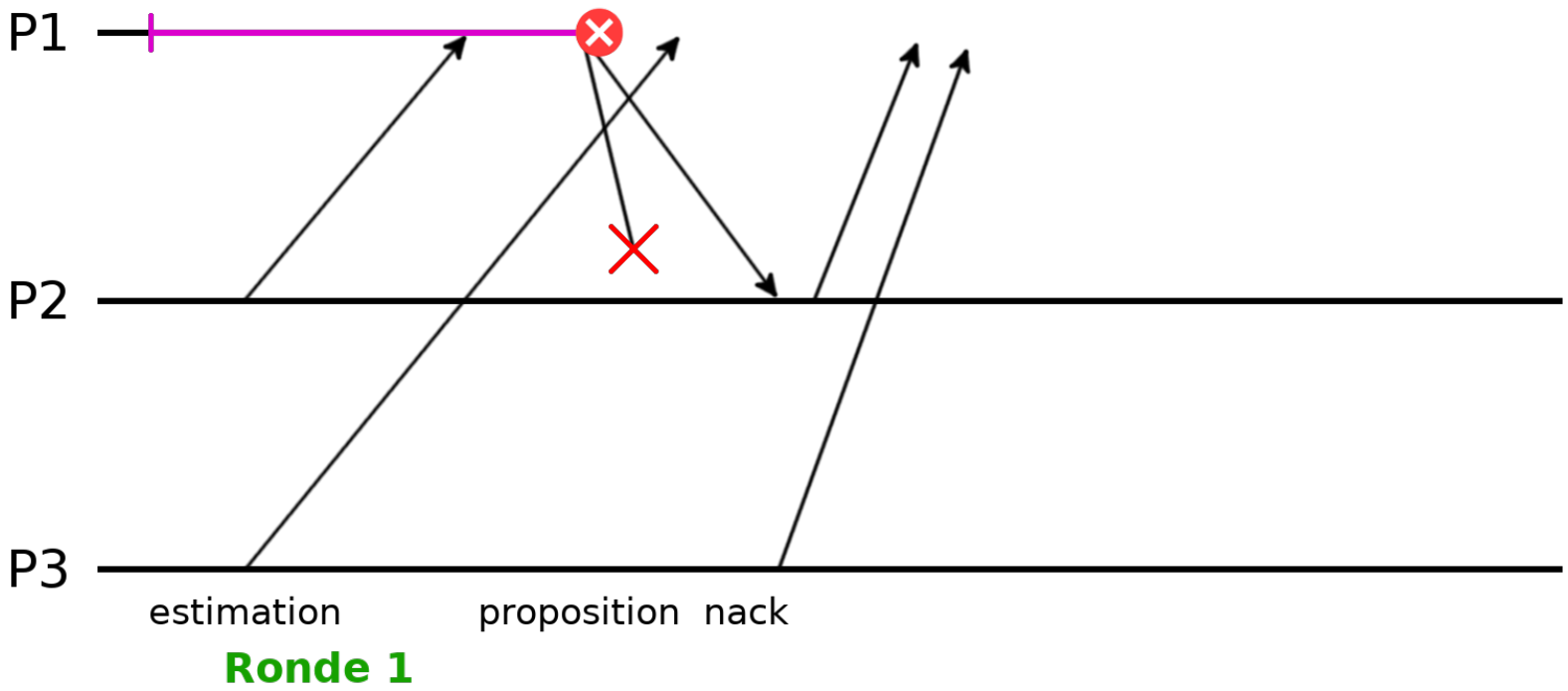
Algorithme du coordinateur tournant

Exemple 2



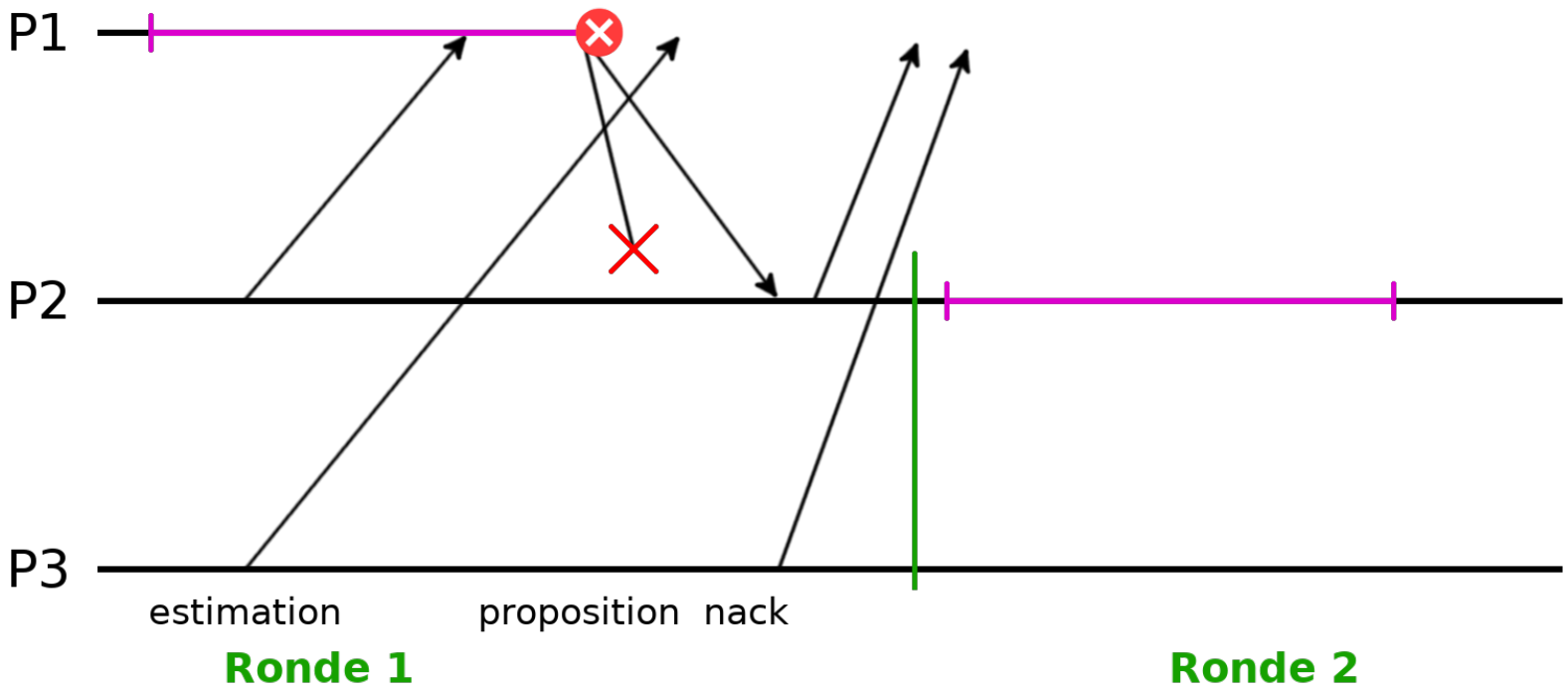
Algorithme du coordinateur tournant

Exemple 2



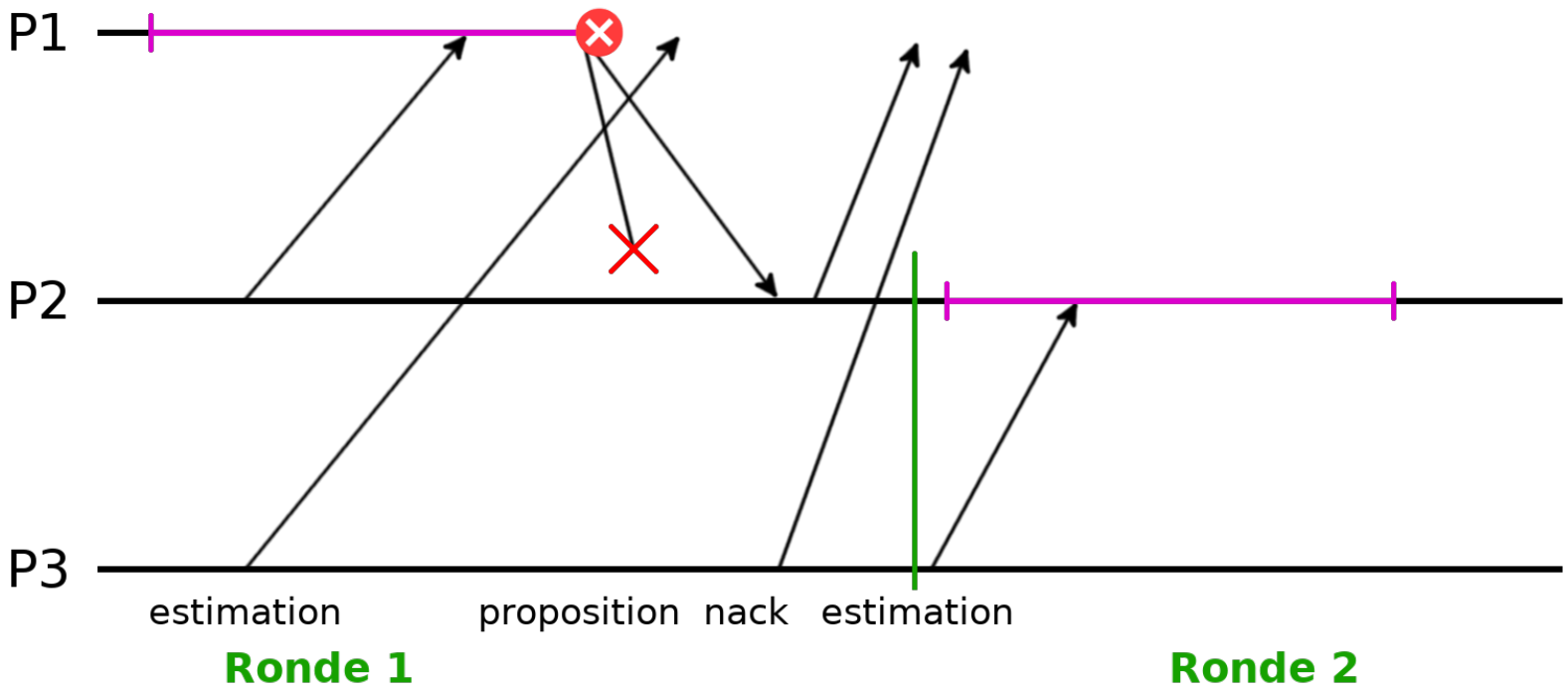
Algorithme du coordinateur tournant

Exemple 2



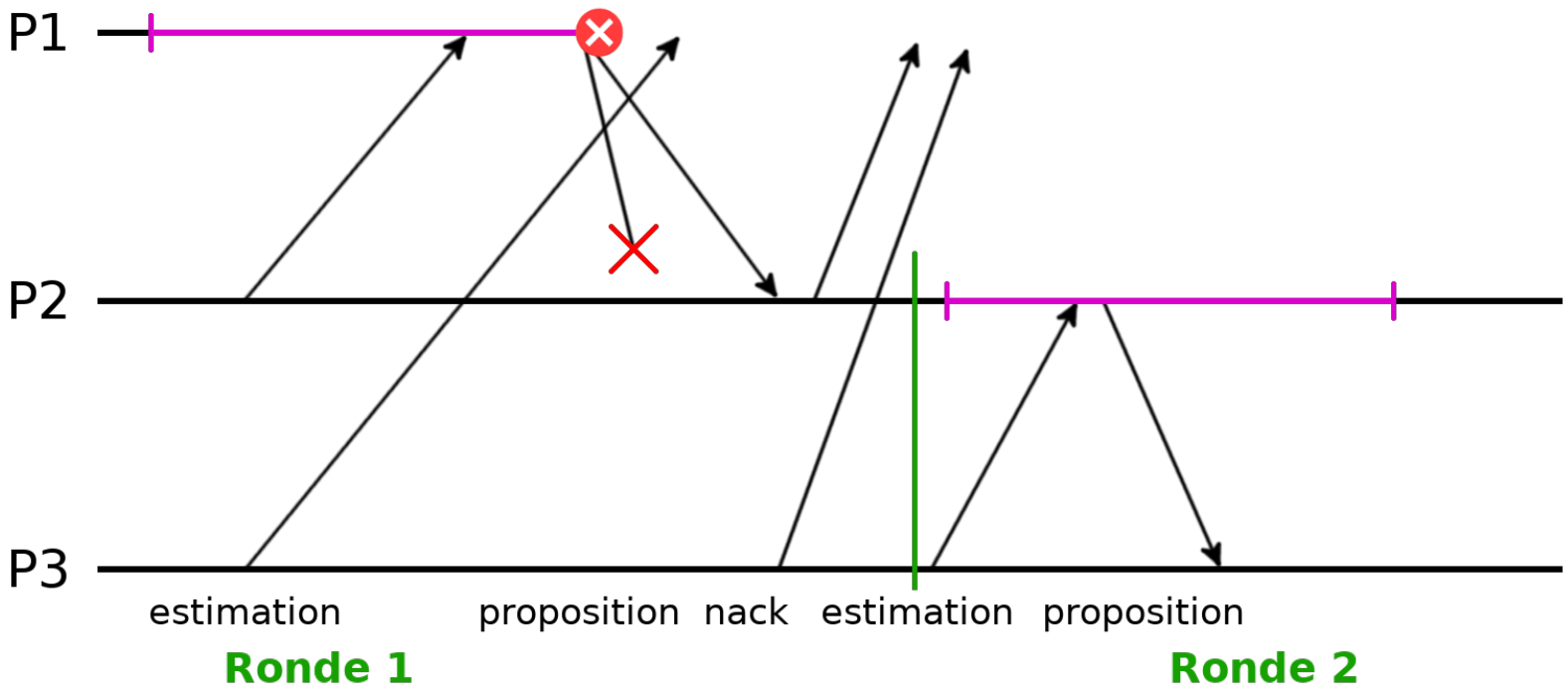
Algorithme du coordinateur tournant

Exemple 2



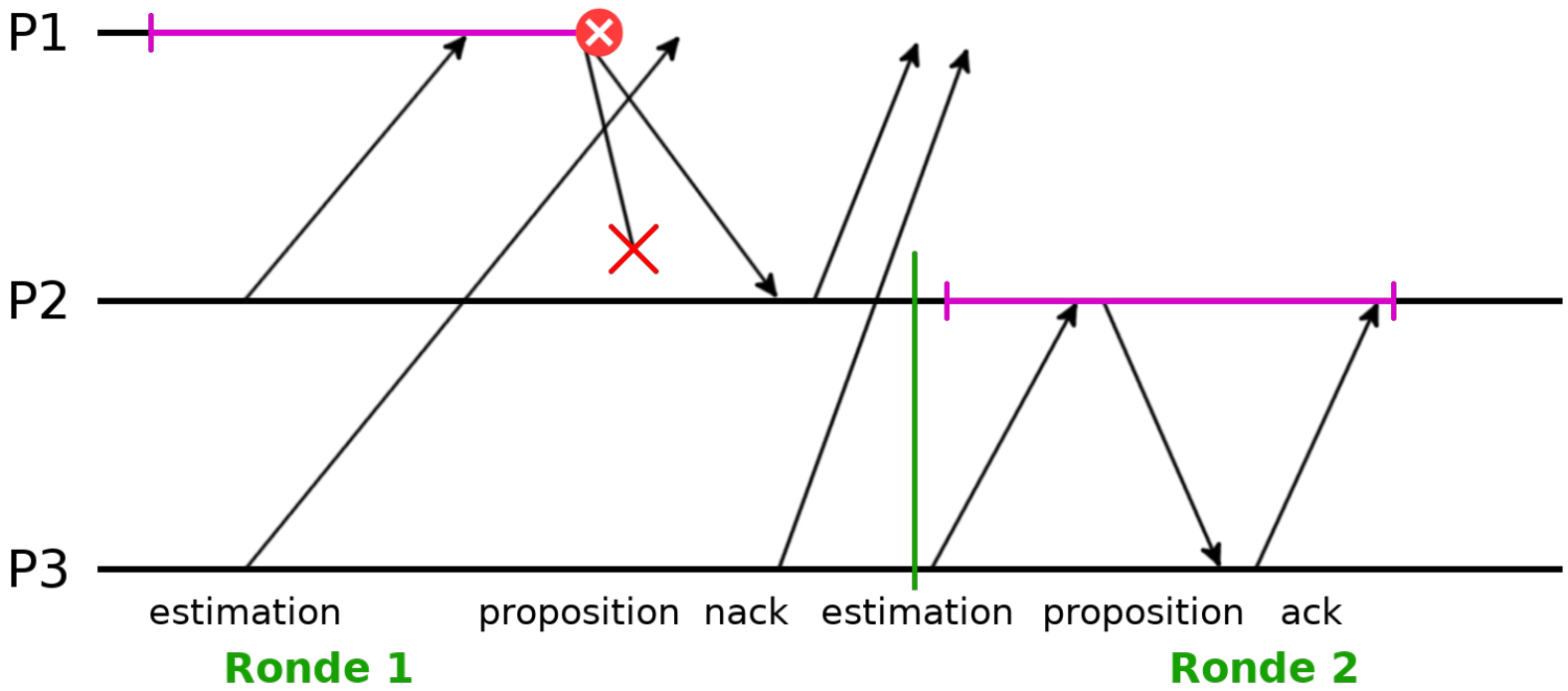
Algorithme du coordinateur tournant

Exemple 2



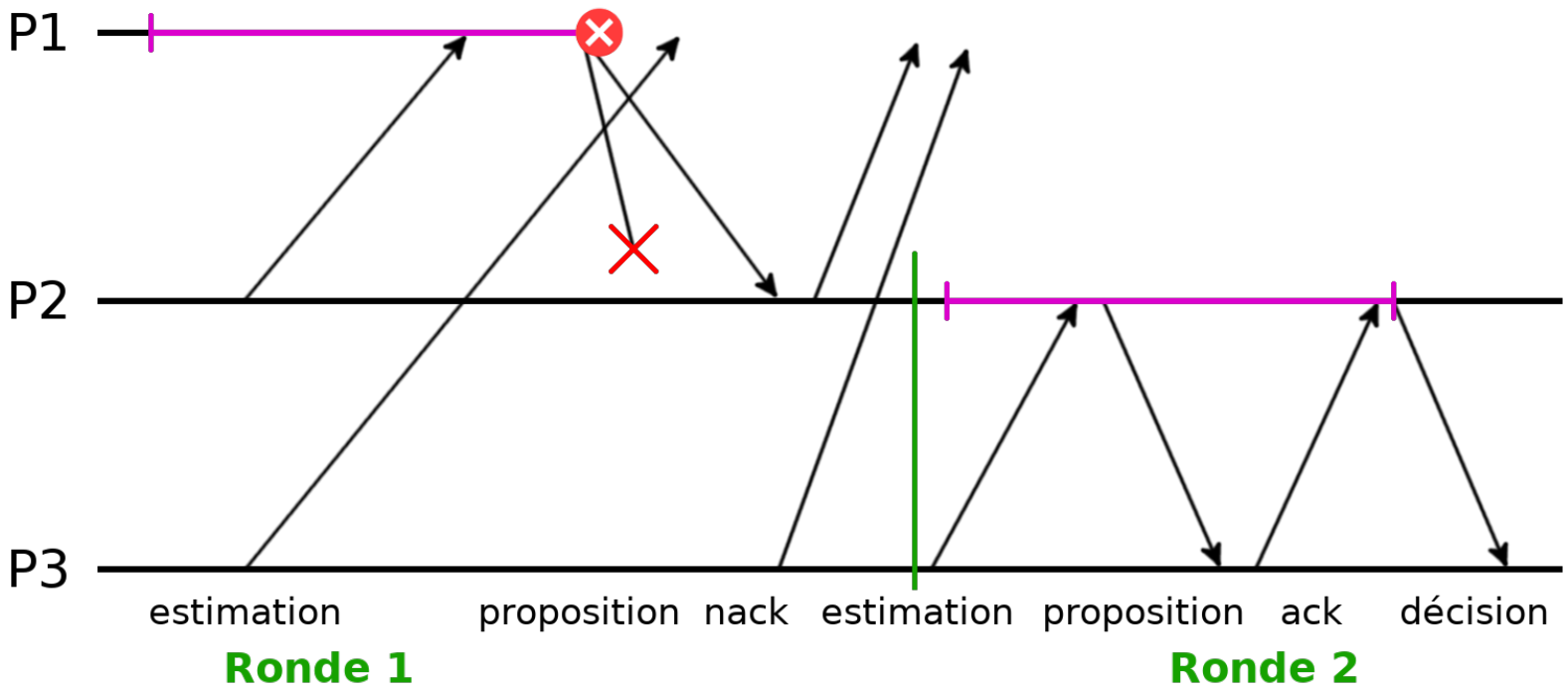
Algorithme du coordinateur tournant

Exemple 2



Algorithme du coordinateur tournant

Exemple 2



1. **Changer le problème**
   Problème du k-accord
2. **Changer le modèle**
   Modèle partiellement synchrone
3. **Abstraction du modèle**
   Détecteurs de fautes
4. **Consensus «imparfait»**
   Consensus probabiliste, Paxos

Déterministe : validité, cohérence, intégrité
Probabiliste : terminaison

Déterministe : validité, cohérence, intégrité
Probabiliste : terminaison

Un algo important : **Paxos**.

Déterministe : validité, cohérence, intégrité
Probabiliste : terminaison

Un algo important : **Paxos**.

Certains processus peuvent ne pas terminer !
Mais la probabilité de terminer tend vers 1 en fonction du temps.