SDR

Consensus

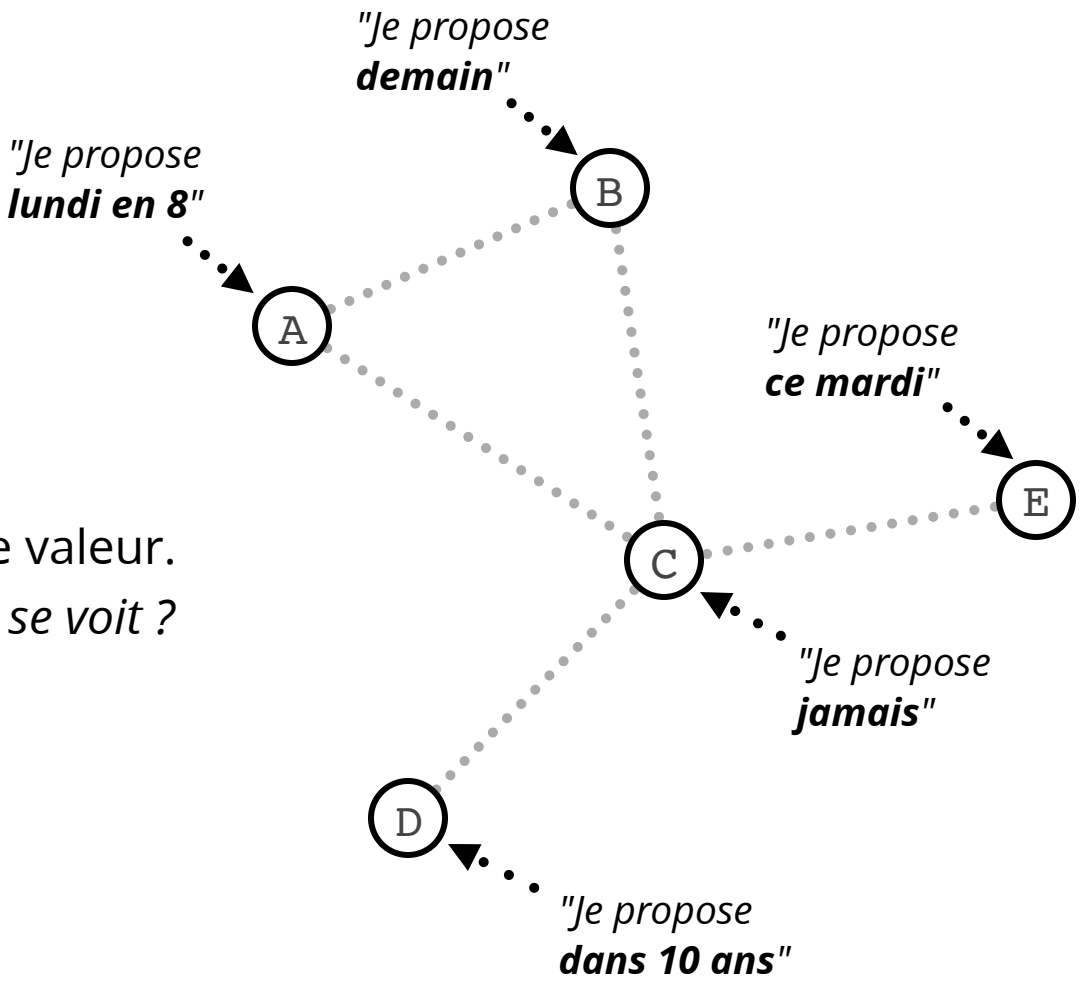
Consensus naïf

Olivier Lemer

↓ Consensus ?

Consensus

Qu'est-ce que c'est ?



Au début

Chaque processus **propose** une valeur.

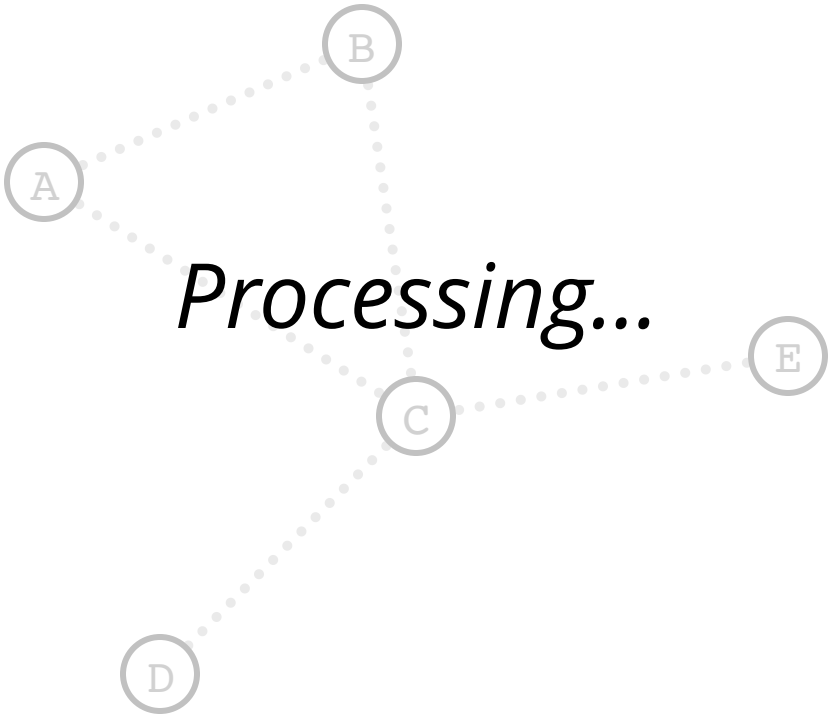
Par exemple : quand est-ce qu'on se voit ?

Consensus

Qu'est-ce que c'est ?

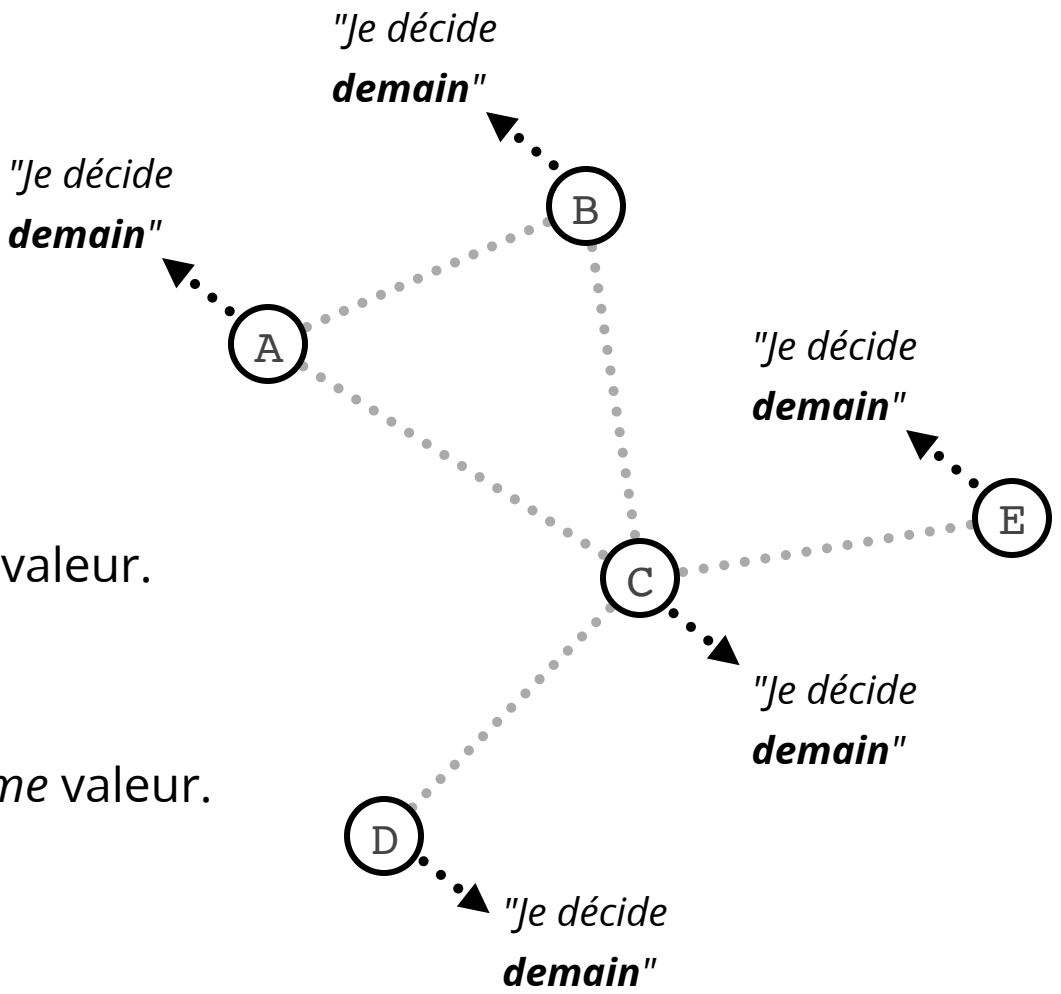
Au début

Chaque processus **propose** une valeur.



Consensus

Qu'est-ce que c'est ?



Au début

Chaque processus **propose** une valeur.

À la fin

Chaque processus **décide** la *même* valeur.

(Laquelle ? C'est arbitraire.)

↓ Suppositions du jour

Suppositions du jour

Pas de perte, de duplication, ou de réordonnancement des messages.

Toute panne est **permanente**.

Il existe une borne supérieure constante **`T`** sur la durée de transit de tout message.
On a donc un système "synchrone".
*(À ne pas confondre avec un **algorithme** synchrone)*

Les durées de traitement sont négligeables.

*Comprendre : on pourra utiliser un détecteur de pannes **parfait***

Le réseau est une clique (tous connectés à tous).

Pourquoi cette supposition sur `T` ?

> *"C'est une supposition irréaliste d'avoir une borne sur la durée de transit des messages"*

Et pourtant on l'utilise quand même ! Pourquoi ?

> *"Sans garantie sur la durée de transit des messages,*
> ***il est impossible d'atteindre un consensus**,*
> *dès que même un seul processus peut tomber en panne."*
>
> *Fischer, M., N. Lynch, and M. Paterson (1985).*

On a donc besoin d'un système au moins *partiellement synchrone*.

On fait ici une supposition plus forte (système synchrone) pour simplifier.



Propriétés

Ce qui définit le consensus

Propriétés

Terminaison

Tout processus correct **décide** d'une valeur, *un jour*.

(on finira par décider)

Validité

Si un processus décide $\mathtt{v}$, alors un processus a proposé $\mathtt{v}$.

(on n'invente pas de valeur)

Unicité

Aucun processus ne décide deux fois.

(une seule décision par consensus)

Accord

Tous deux processus décident la même valeur

(on décide tous pareil)

↓ Solution naïve

Solution naïve

Quand je propose

Envoyer ma proposition à tout le monde.

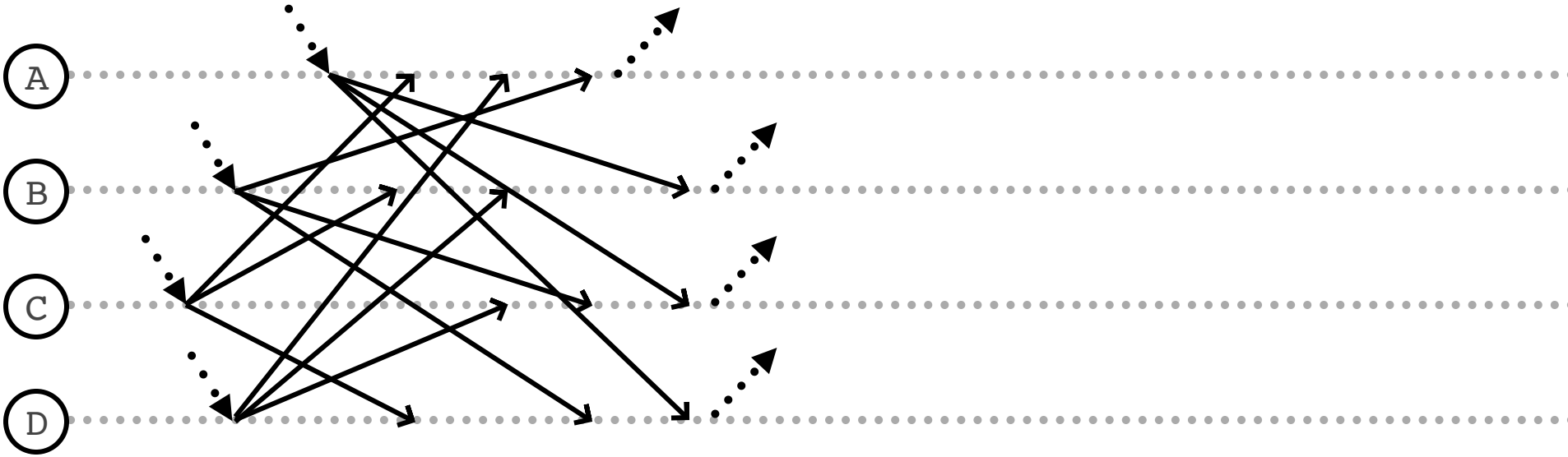
Problème ?

Si un processus tombe en panne ?

Quand j'ai reçu toutes les propositions

J'en choisis une de manière déterministe.

(i.e. en suivant tous le même algorithme)



Solution naïve

Quand je propose

Envoyer ma proposition à tout le monde.

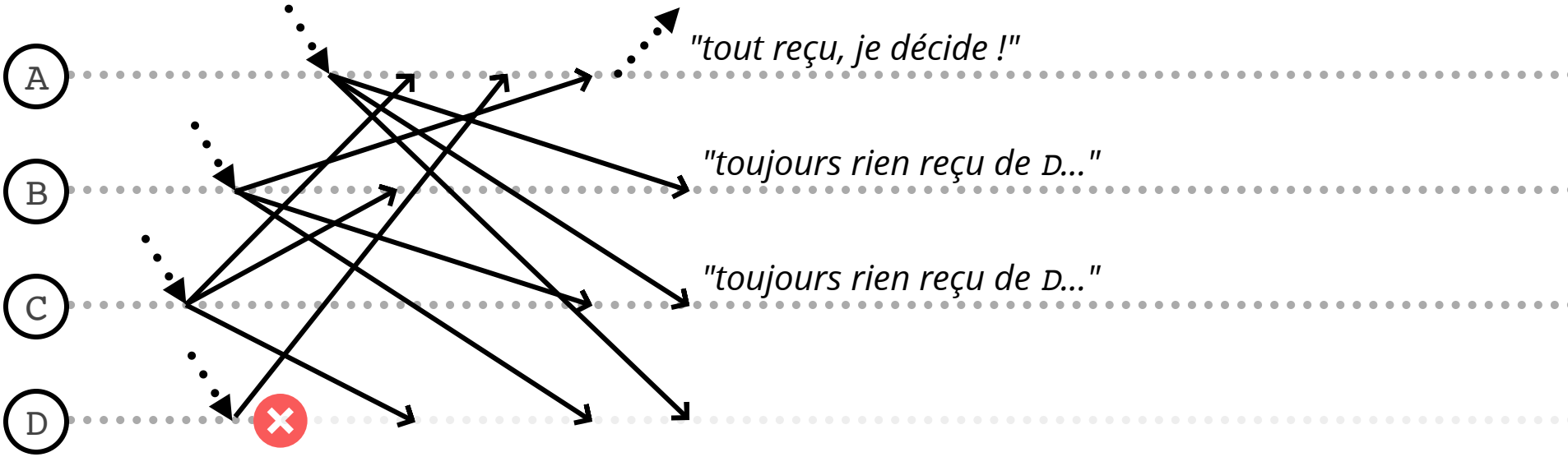
Quand j'ai reçu toutes les propositions

J'en choisis une de manière déterministe.

Problème ?

Si un processus tombe en panne ?
Durant un envoi ?

*Risque de perte de **terminaison**.*



Solution naïve

Quand je propose

Envoyer ma proposition à tout le monde.

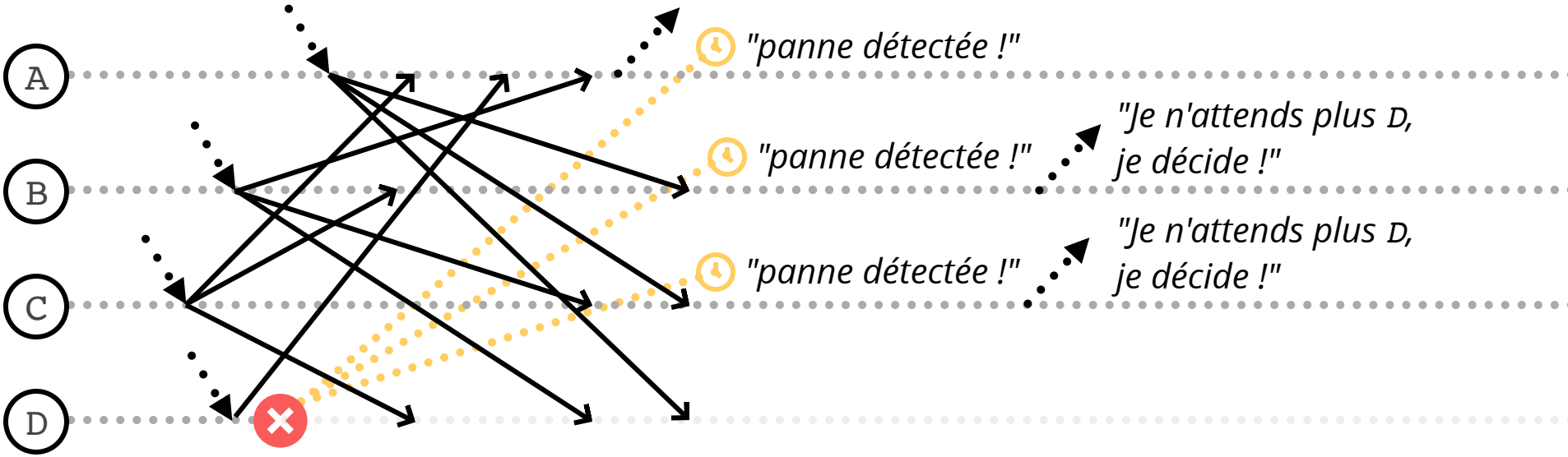
Quand j'ai reçu toutes les propositions

J'en choisis une de manière déterministe.

Problème ?

Si un processus tombe en panne ?
Durant un envoi ?

*Risque de perte de **terminaison**,*
*ou d'**accord** (si détection de pannes).*

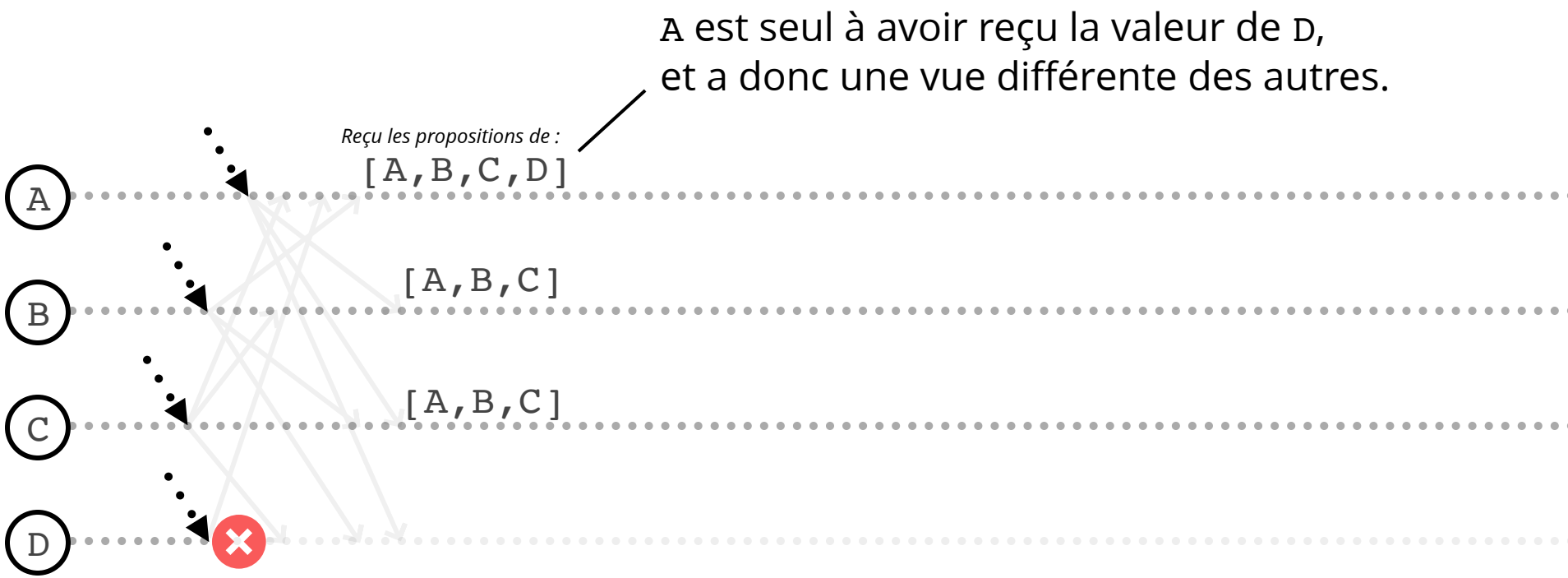


Solution naïve

Le problème

Si un processus tombe en panne

Possible que tout le monde n'ait pas toutes les valeurs.



Solution naïve

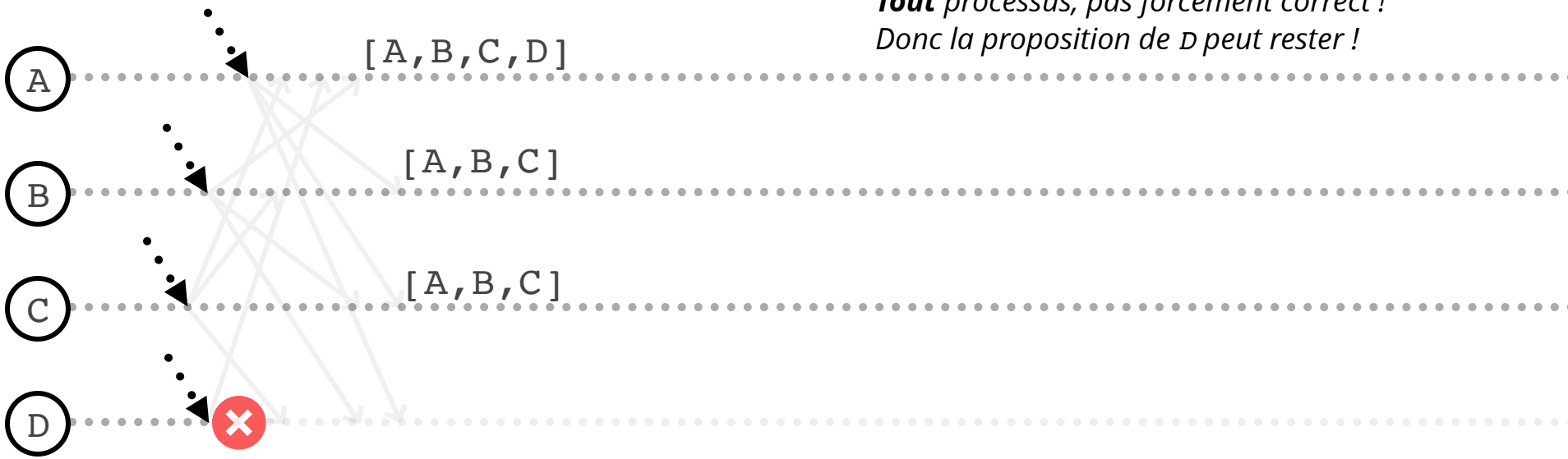
Le problème

Si un processus tombe en panne

Possible que tout le monde n'ait pas toutes les valeurs.

Solution ? Qu'est-ce qu'on veut ? On garde la proposition de `D` ou pas ?
Souvenez-vous de la propriété de "Validité" :
"Si un processus décide `v`, alors un processus a proposé `v`."

***Tout** processus, pas forcément correct !*
Donc la proposition de `D` peut rester !



Solution naïve

Le problème

Si un processus tombe en panne

Possible que tout le monde n'ait pas toutes les valeurs.

Solution ? Il faut propager ce qu'on sait aux autres !

Quand je n'attends plus de proposition, je lance un nouveau round.

Pour chaque processus, soit j'ai reçu sa proposition, soit j'ai détecté sa panne.



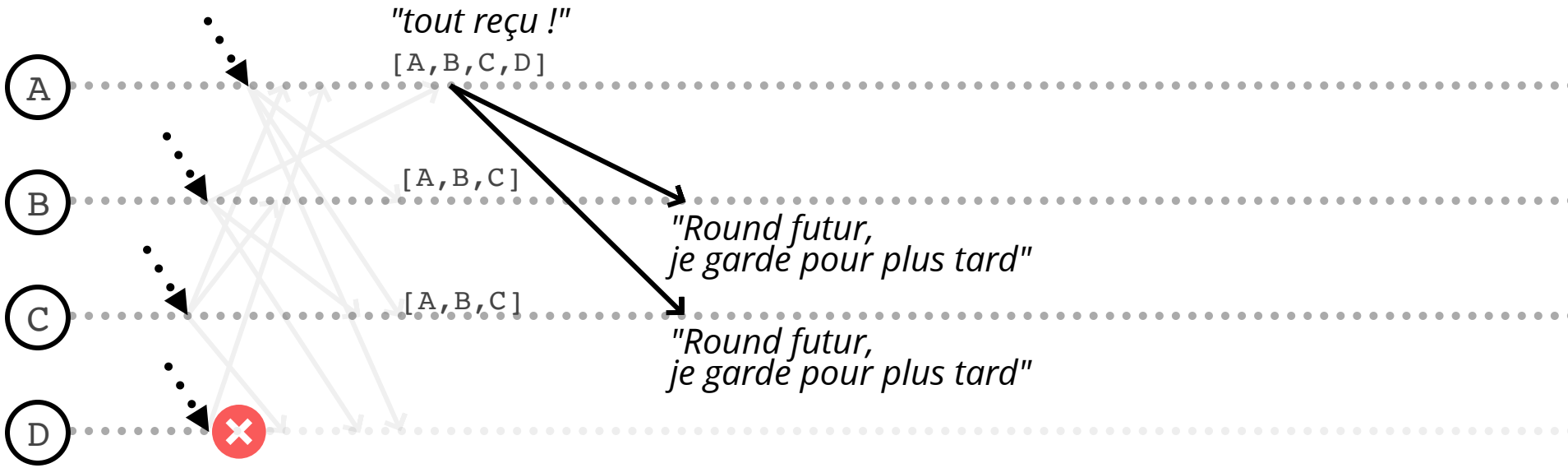
Plusieurs rounds

Comme solution à notre problème

Deuxième round

Lorsque je n'ai plus rien à recevoir

Je lance un deuxième round, avec toutes les valeurs connues.

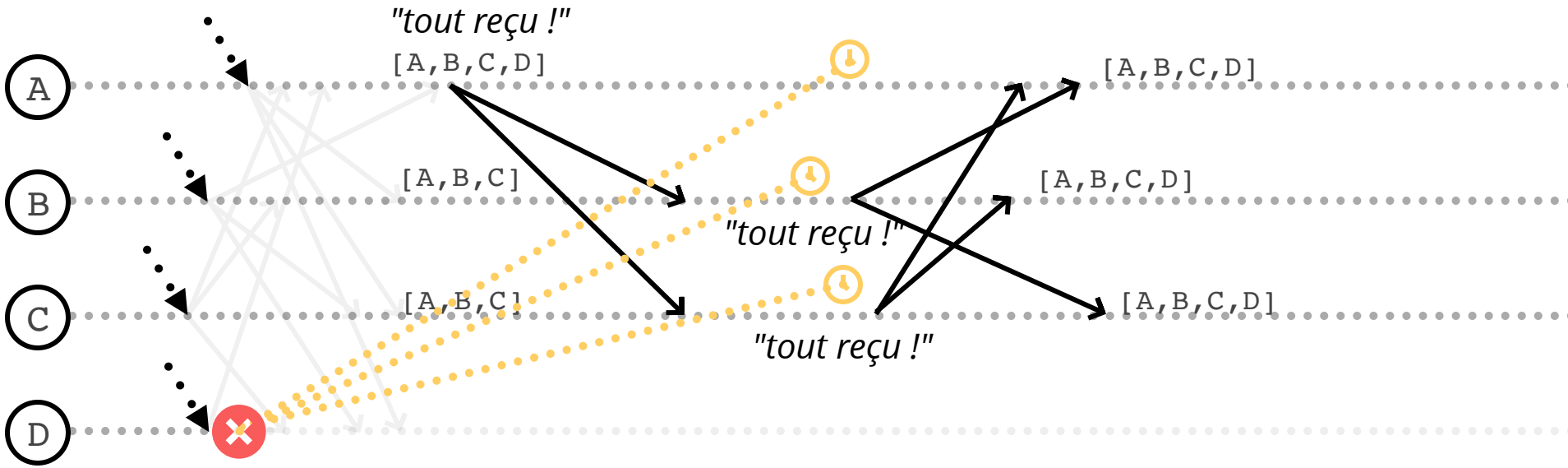


Deuxième round

Lorsque je n'ai plus rien à recevoir

Je lance un deuxième round, avec toutes les valeurs connues.

On est bon ?



Deuxième round

Lorsque je n'ai plus rien à recevoir

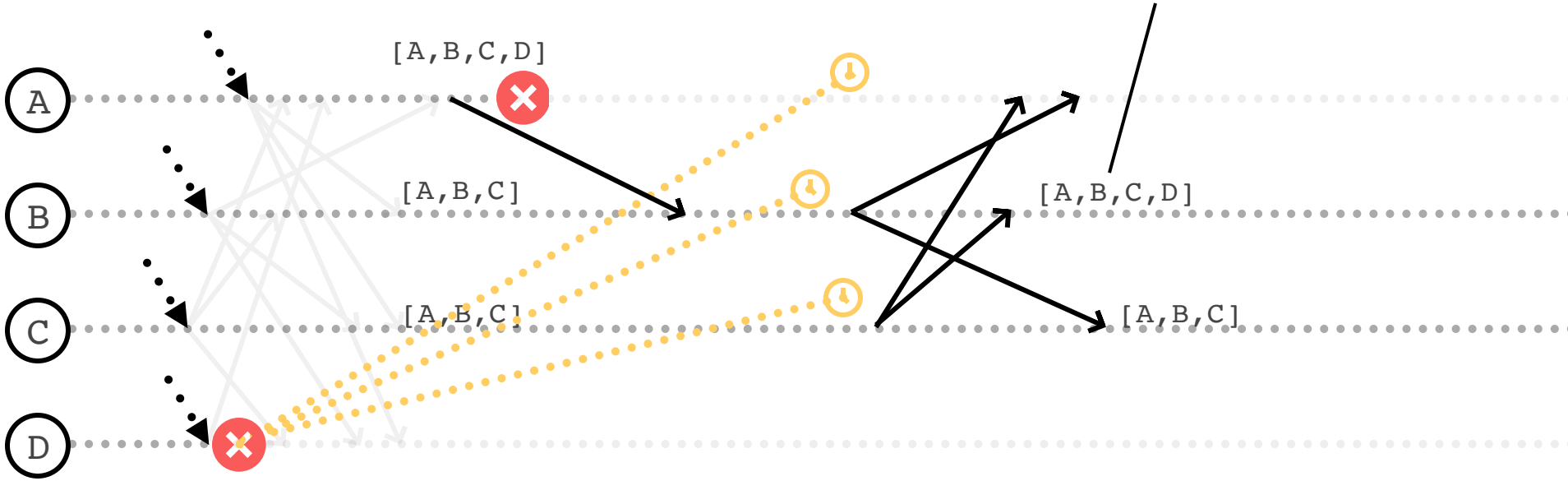
Je lance un deuxième round, avec toutes les valeurs connues.

On est bon ?

Non ! Même problème !

Solution ? *On refait pareil !*

Seul `B` a reçu les données de `A`,
et donc celles de `D`.
`C` a encore une vision incomplète.



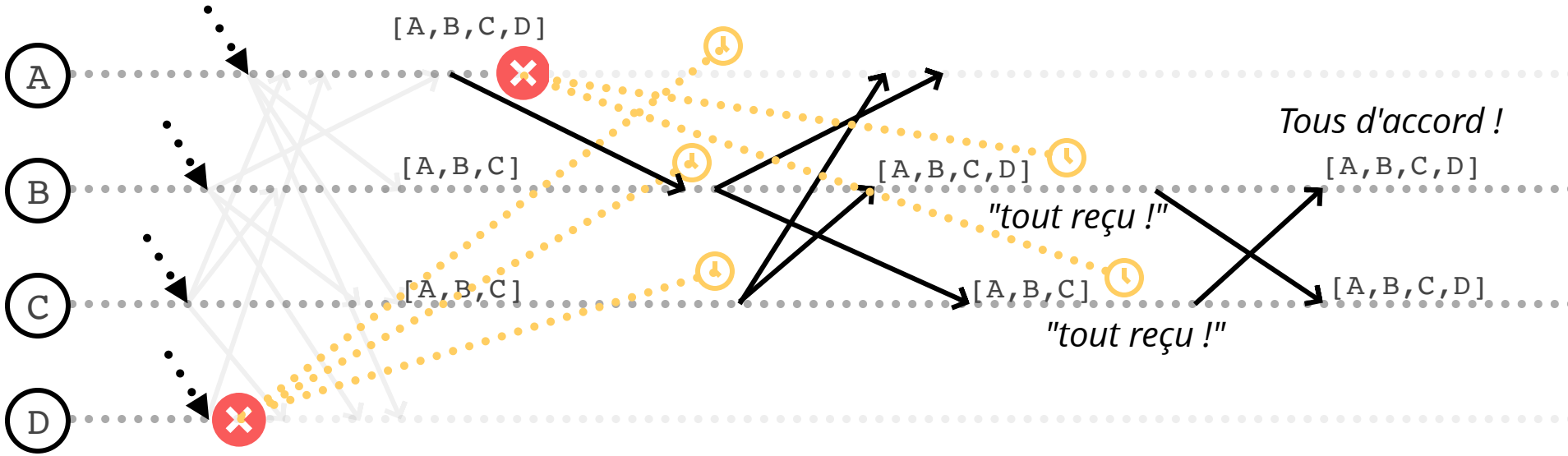
Plusieurs rounds

Quand j'ai la proposition de chaque processus non détecté

Je lance un nouveau round, avec toutes les valeurs connues.

Et maintenant, on est bon ?

Peut-être… Mais quand s'arrêter ?



Combien de rounds ?

Ou "quand s'arrêter" ?

Quand s'arrêter

Quand suis-je sûr d'avoir toutes les propositions ?

Quand il n'y a pas eu de pannes pendant le round précédent.

Utiliser le détecteur de pannes ?

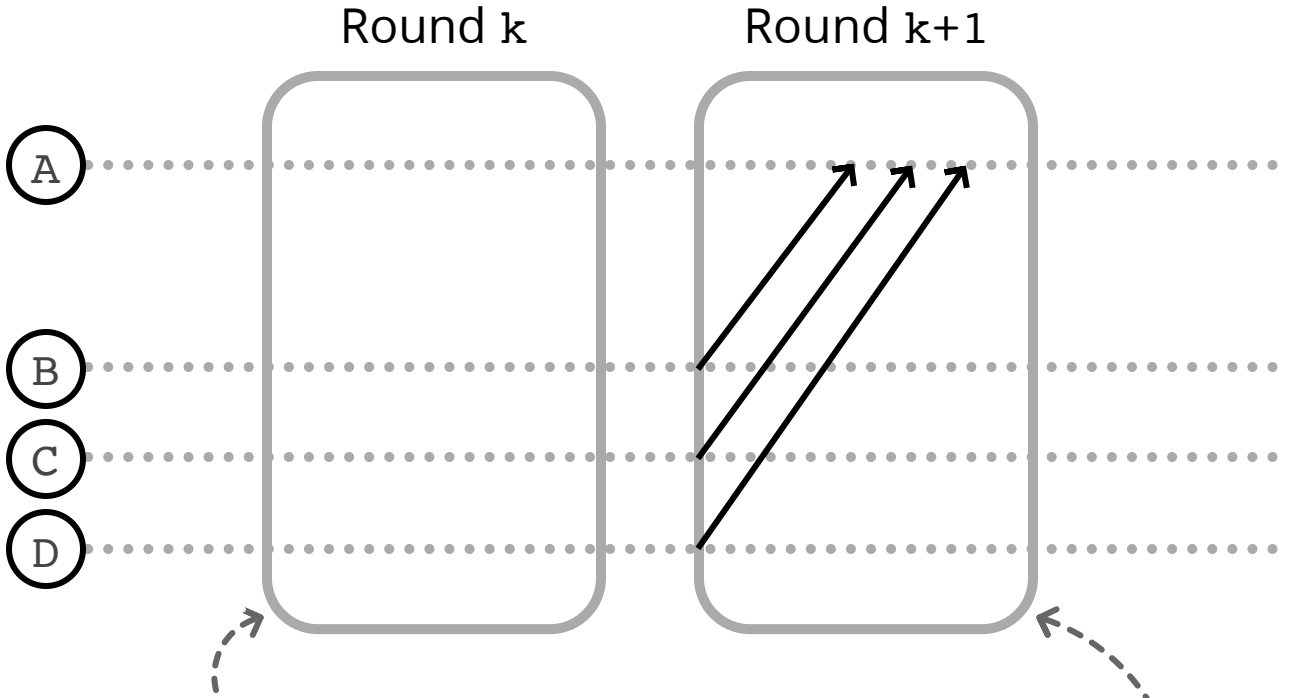
Il peut prendre 2T à détecter ; chaque round devrait donc durer 2T.

Comment, alors ?

> *Si je reçois de la part des mêmes processus deux fois de suite, alors aucune panne n'a eu lieu entre temps.*

Quand s'arrêter

Si je reçois de la part des mêmes processus deux fois de suite, alors aucune panne n'a eu lieu entre temps.



À la fin du round k, l'union de ce que tout le monde sait correspond à ce que je veux savoir, et est garantie de m'arriver durant le round k+1.

Quand s'arrêter

Quand suis-je sûr d'avoir toutes les propositions ?

Quand il n'y a pas eu de pannes pendant le round précédent.

Utiliser le détecteur de pannes ?

Il peut prendre 2T à détecter ; chaque round devrait donc durer 2T.

Comment, alors ?

> *Si je reçois de la part des mêmes processus deux fois de suite, alors aucune panne n'a eu lieu entre temps.*

Donc je suis sûr d'avoir toutes les propositions.

Est-ce que je peux m'arrêter, alors ?

Pas encore, je suis peut-être le seul dans cette situation !

Quand s'arrêter

Round 1

A reçoit de B, C, D

B reçoit de A, C

C reçoit de A, B

Round 2

B reçoit de A, C

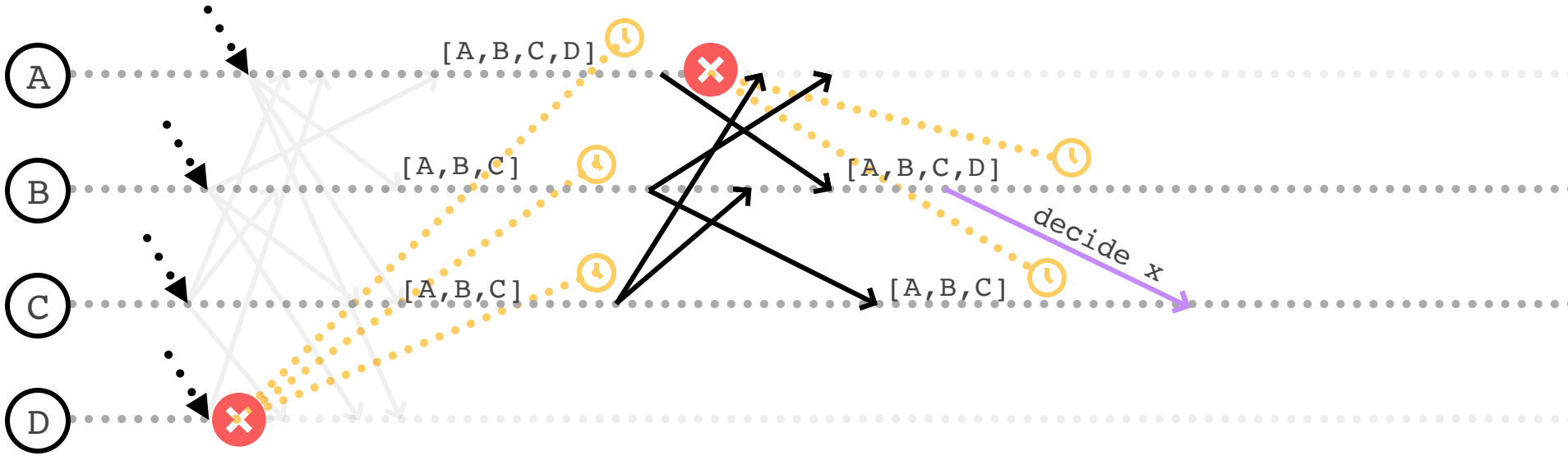
C reçoit de B

B a une vision complète !

Que faire ?

Le dire à tout le monde !

(ici, il n'y a que C qui est encore en marche)



Résumé intermédiaire

Jusqu'au "partage de décision"

Résumé intermédiaire

Initialement

Ma liste de propositions ne contient que la mienne.
J'entre en round 1.

Quand j'entre en round `x`

J'envoie ma liste de propositions avec l'id de round.

Quand je reçois un message pour le round `x`

Si je ne suis pas encore en round `x`, je repousse sa réception.
Sinon, je mets à jour ma liste de propositions.

Quand j'ai reçu un message de tous ceux dont je n'ai pas détecté la panne

Si j'ai reçu, ce round, de la part des mêmes qu'au round précédent,
    Je décide parmi mes propositions *(de manière déterministe).*
    J'envoie ma décision à tous les processus en ligne, et je m'arrête.
Sinon,
    Je passe au round suivant.

Phase de décision

Phase de décision

Quand j'ai reçu un message de tous ceux dont je n'ai pas détecté la panne

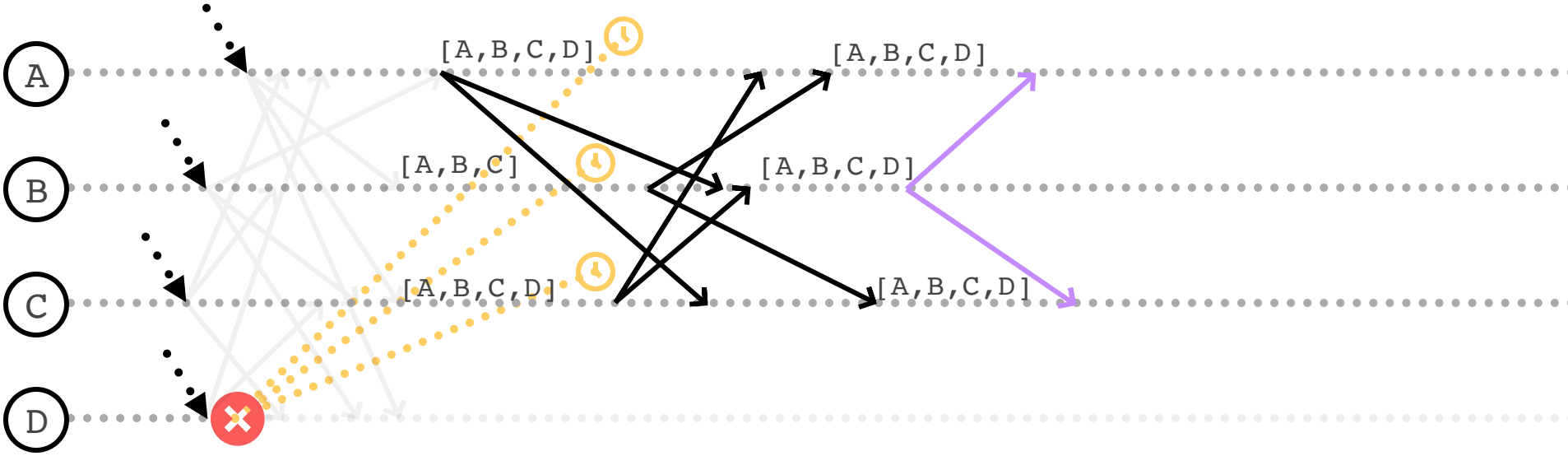
Si j'ai reçu, ce round, de la part des mêmes qu'au round précédent,

Je décide parmi mes propositions *(de manière déterministe).*

J'envoie ma décision à tous les processus en ligne, et je m'arrête.

Quand je reçois une décision

Je note et je m'arrête ? *Non ! Même risque qu'avant*



Phase de décision

Quand j'ai reçu un message de tous ceux dont je n'ai pas détecté la panne

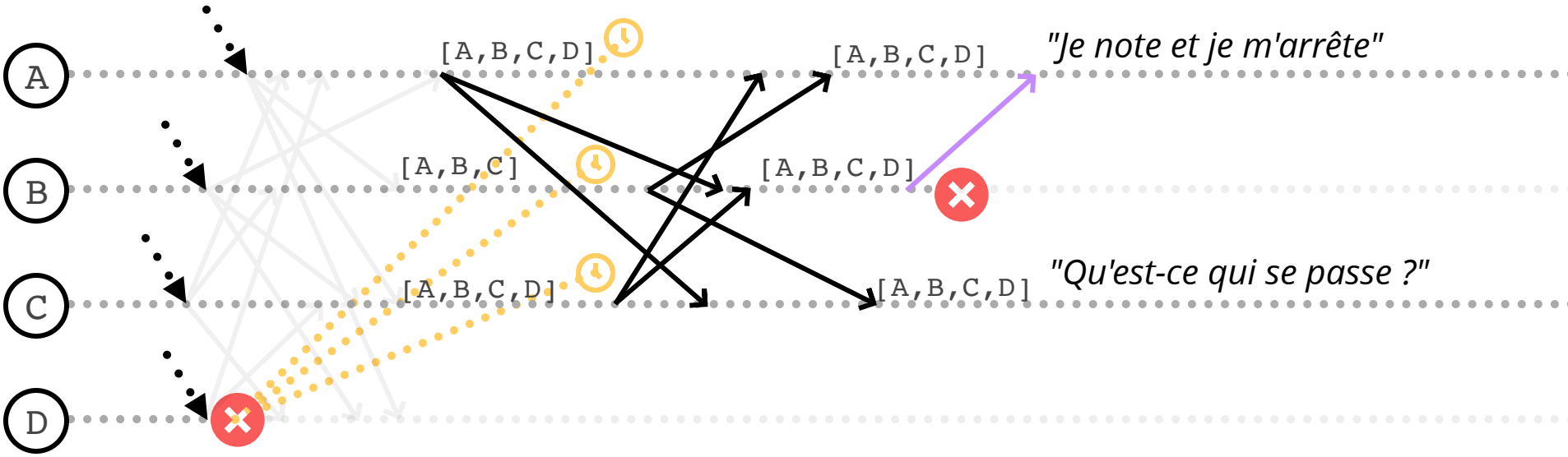
Si j'ai reçu, ce round, de la part des mêmes qu'au round précédent,

Je décide parmi mes propositions *(de manière déterministe).*

J'envoie ma décision à tous les processus en ligne, et je m'arrête.

Quand je reçois une décision

Je note et je m'arrête ? *Non ! Même risque qu'avant*



Phase de décision

Quand j'ai reçu un message de tous ceux dont je n'ai pas détecté la panne

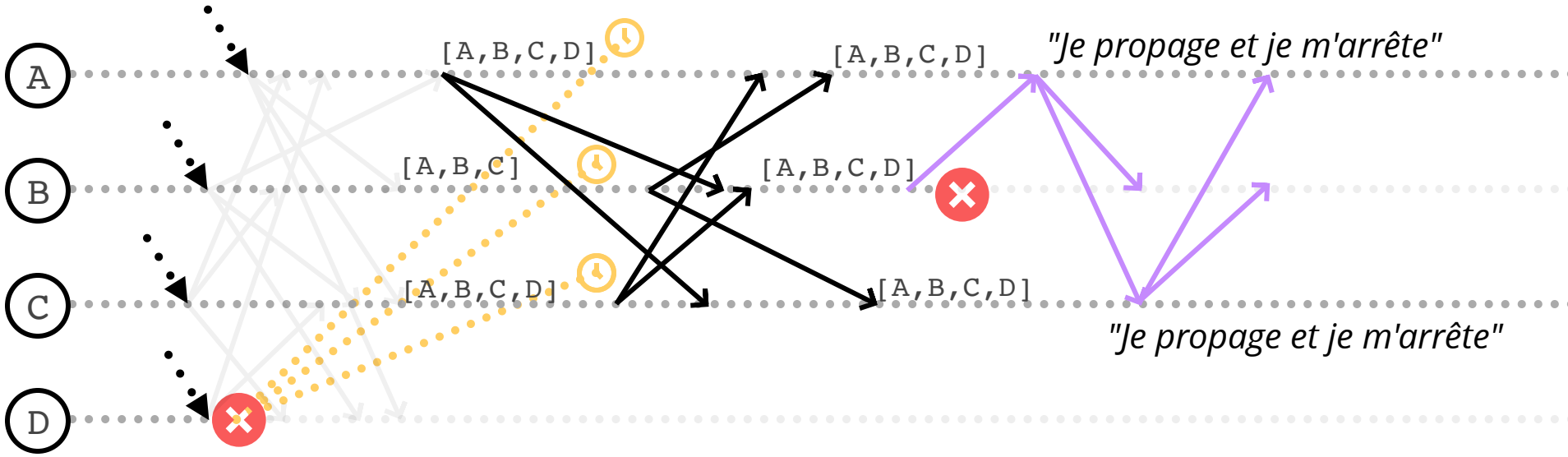
Si j'ai reçu, ce round, de la part des mêmes qu'au round précédent,

Je décide parmi mes propositions *(de manière déterministe).*

J'envoie ma décision à tous les processus en ligne, et je m'arrête.

Quand je reçois une décision

Je *propage*, et je m'arrête.



Phase de décision

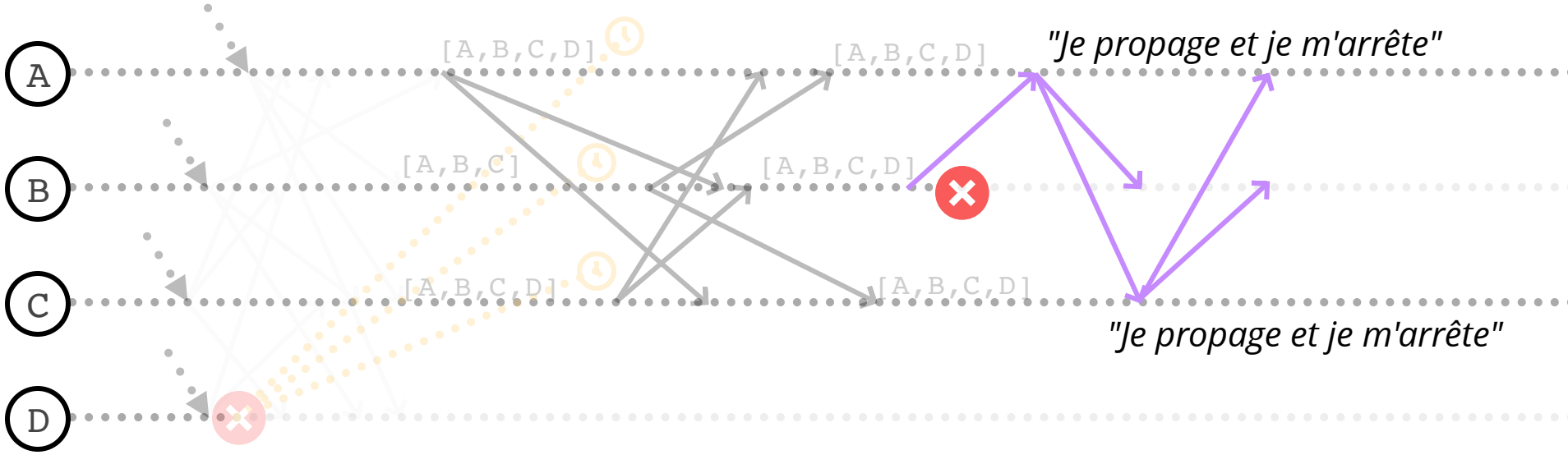
Remarque

L'approche utilisée ici,

"À la réception d'un broadcast, je broadcast à mon tour"

n'est pas spécifique au consensus.

Elle permet un broadcast sûr dans tout contexte de broadcast.



Résumé complet

et analyse de tolérance aux pannes

Résumé

Initialement

Ma liste ne contient que ma proposition.
J'entre en round 1.

Quand je reçois une décision

Je *propage*, et je m'arrête.

Quand j'entre en round `x`

J'envoie ma liste de propositions avec l'id de round.

Quand je reçois un message pour le round `x`

Si je ne suis pas encore en round `x`, je repousse sa réception.
Sinon, je mets à jour ma liste de propositions.

Quand j'ai reçu un message de tous ceux dont je n'ai pas détecté la panne

Si j'ai reçu, ce round, de la part des mêmes qu'au round précédent,
    Je décide parmi mes propositions *(de manière déterministe).*
    J'envoie ma décision à tous les processus en ligne, et je m'arrête.
Sinon,
    Je passe au round suivant.

Tolérance aux pannes

Combien de pannes puis-je tolérer ?

Jusqu'à N-1 (!)

C'est excellent. En général, difficile de tolérer plus qu'une constante, ou N/2-1.

Consensus IRL

Un problème central

Un problème central

Tout type d'information peut être proposé

"Qui de nous sera l'élu ?"
"Qui de nous aura la SC ?"

Voire même
"Quelle modification fait-on ensuite de l'état global"

Ou de manière plus abstraite
*"Quelle transition applique-t-on sur **une machine d'état partagée**"*

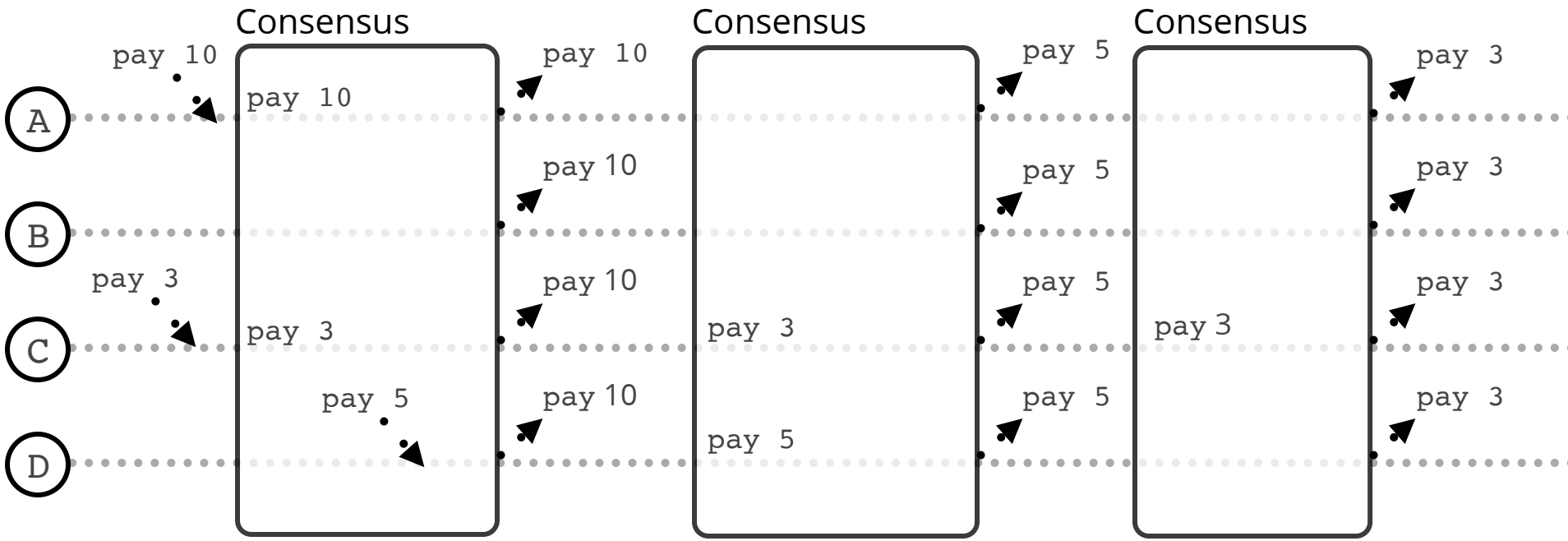
Machine d'état partagée

Tous les processus maintiennent une copie de la même machine d'état :

- Des états *(par exemple, un ensemble de comptes bancaires)*
- Des transitions entre états *(par exemple, des commandes `pay`)*

Si je souhaite modifier l'état, je lance un consensus et propose la modification

Tant que ma proposition n'a pas été acceptée, je recommence.



Machine d'état partagée

Exemples

États	**Transitions**
Graphe de dettes	Ajouts de payements
Qui a le jeton de la SC	Passation du jeton à un autre
Qui est l'élu	Passation du rôle à un autre
Comptes en banque	Transaction d'un compte à un autre

La solution à tout ?

Tout ce qui peut être modélisé par une machine d'état, donc oui.

Mais, bien plus lourd.

Consensus IRL

Suppositions

Jusqu'à N/2 pannes récupérables
Pas de panne arbitraire
Système partiellement asynchrone
Sur les messages : intégrité uniquement

Paxos
Lamport (1989)
Encore lui, bien sûr

Raft *(prochain cours)*
Diego Ongaro and John Ousterhout (2014)
Améliorations de Paxos, spécialisé dans la simulation de machines d'état

Et d'autres bien sûr...