SDR

Estampilles

Mutex de Lamport

Olivier Lemer

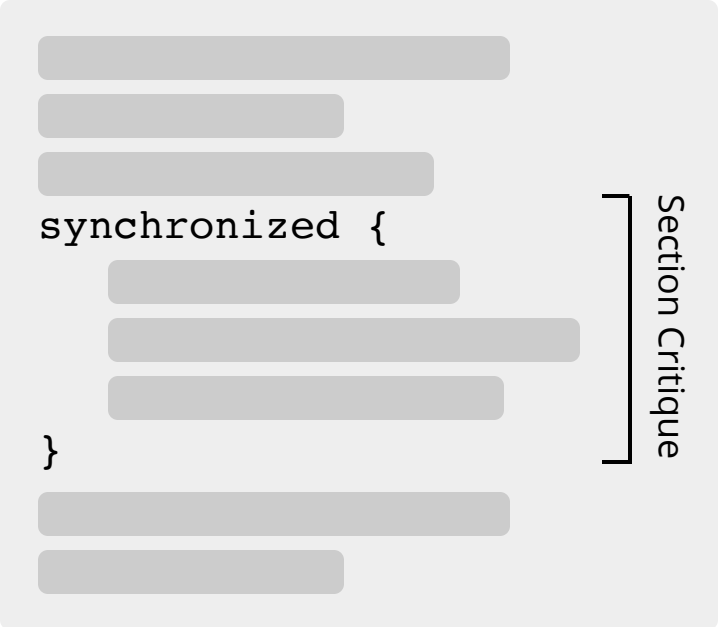
↓ Exclusion Mutuelle ?

Exclusion Mutuelle

Définition | Situation lors de laquelle un bloc d'opérations ne doit être éxecuté que par **un processus à la fois**.

Ce bloc d'opérations est appelé **Section Critique** (SC).

Exemple inspiré de la syntaxe Java

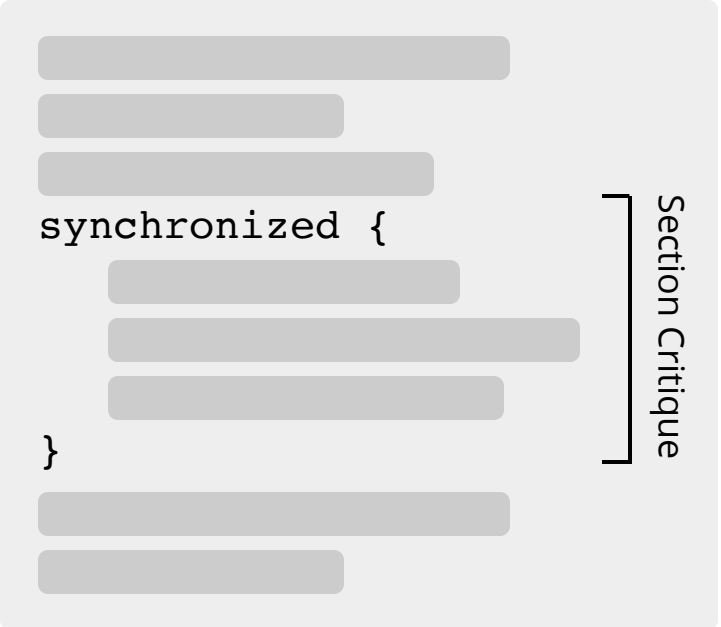


Exclusion Mutuelle

Définition | Situation lors de laquelle un bloc d'opérations ne doit être éxecuté que par **un processus à la fois**.

Ce bloc d'opérations est appelé **Section Critique** (SC).

Exemple inspiré de la syntaxe Java



Synchronisation nécessaire

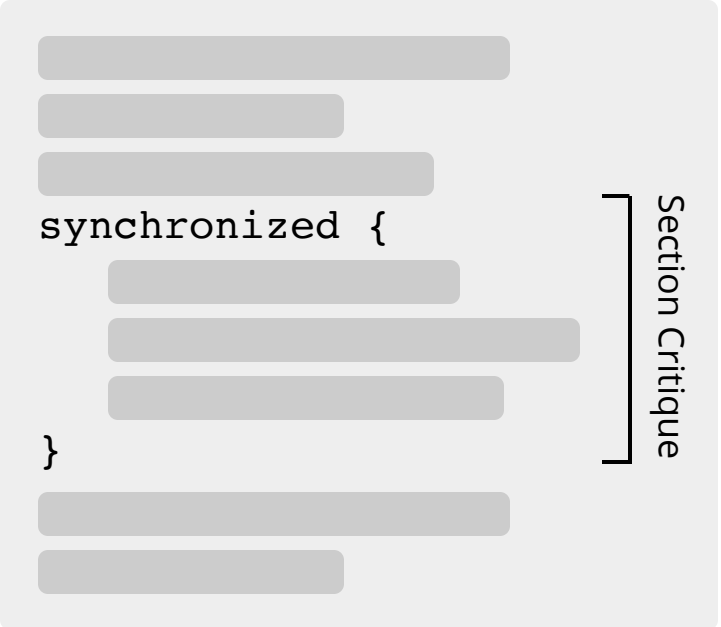
- par locks (**shared memory**), ou
- par messages (**systèmes répartis**).

Exclusion Mutuelle

Définition | Situation lors de laquelle un bloc d'opérations ne doit être éxecuté que par **un processus à la fois**.

Ce bloc d'opérations est appelé **Section Critique** (SC).

Exemple inspiré de la syntaxe Java



Synchronisation nécessaire

- par locks (**shared memory**), ou
- par messages (**systèmes répartis**).

Exemple

Copie synchronisée d'un même compte en banque. Certaines opérations doivent être séquentielles.

↓ Comment ?

Synchronisation par messages

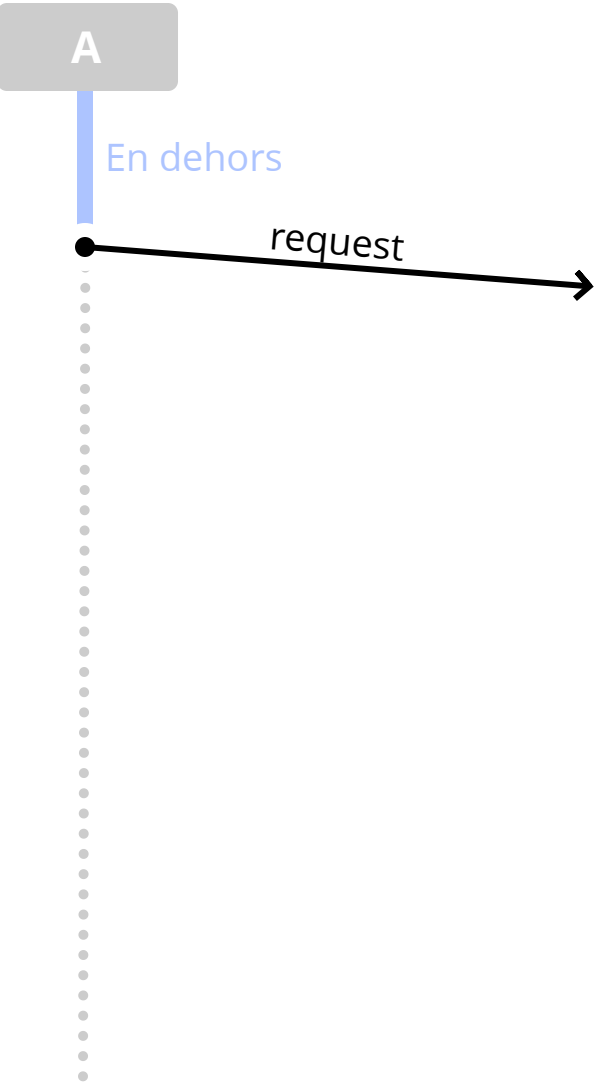
```
synchronized {
}
```

Synchronisation par messages

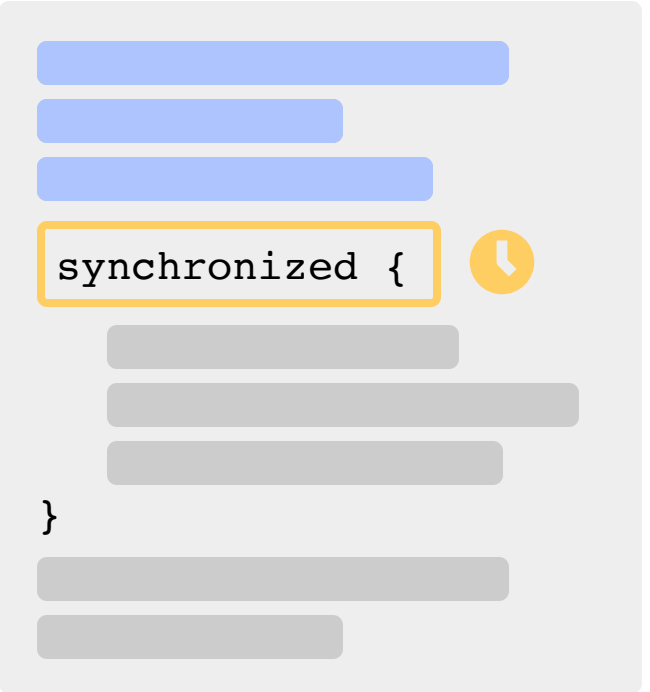
```
synchronized {
}
```

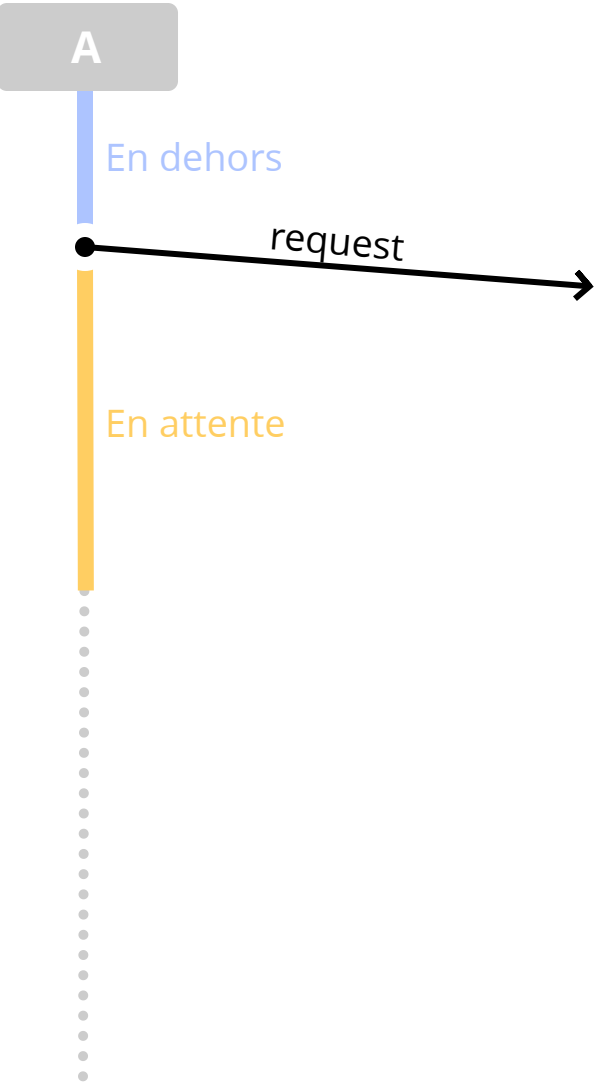
Synchronisation par messages

```
synchronized {
    ...
}
```

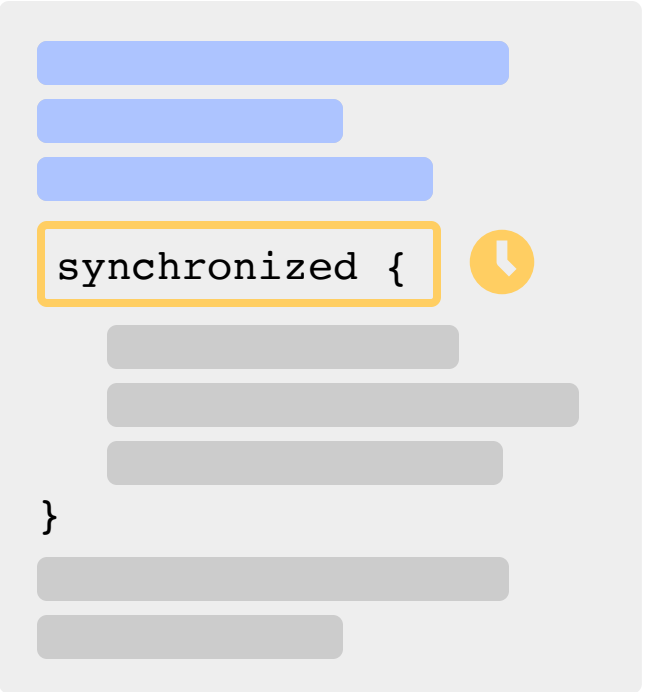


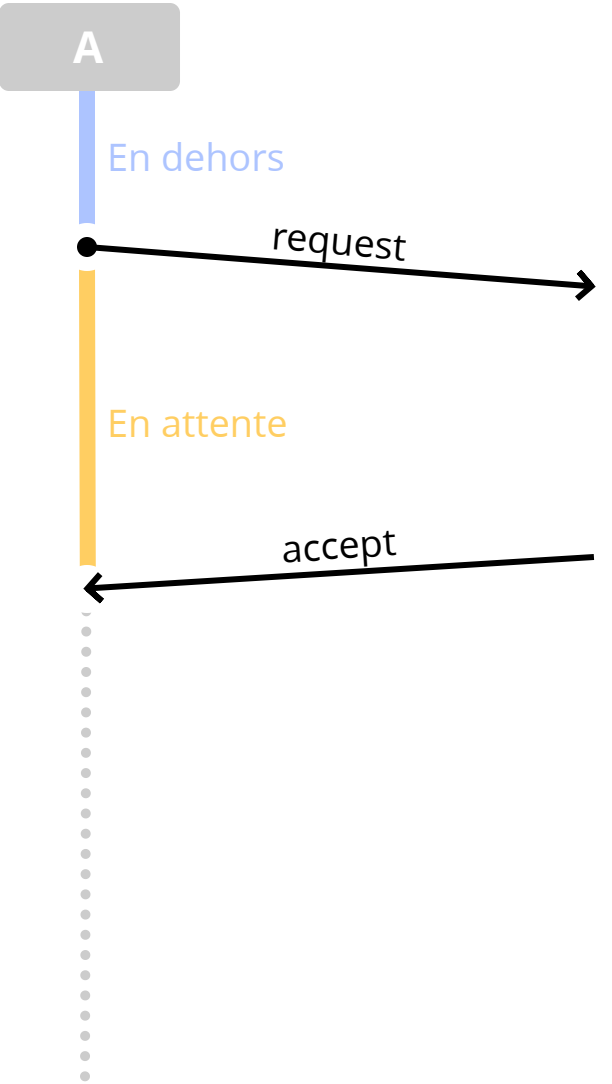
Synchronisation par messages



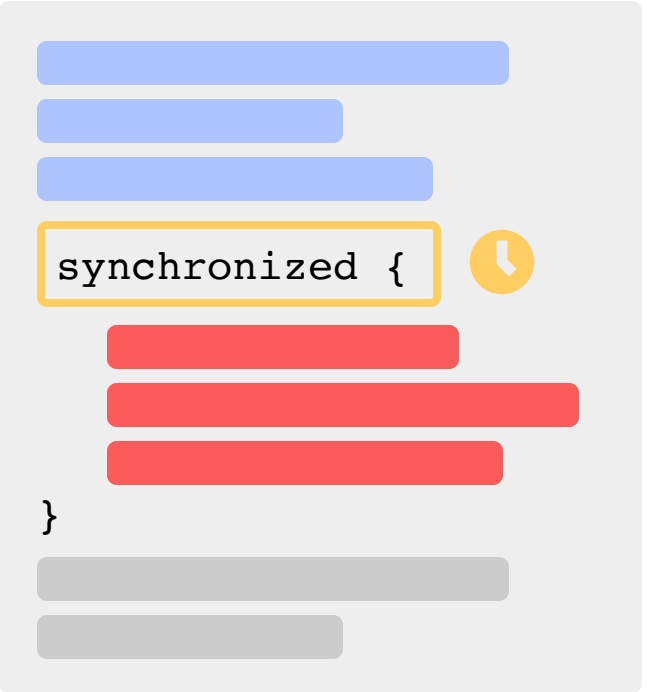


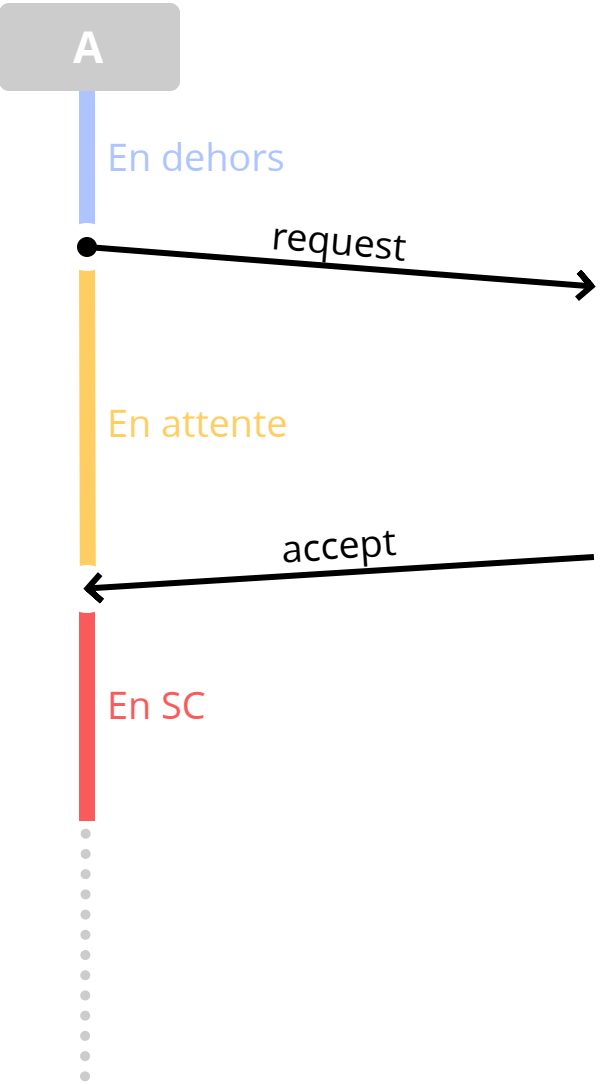
Synchronisation par messages



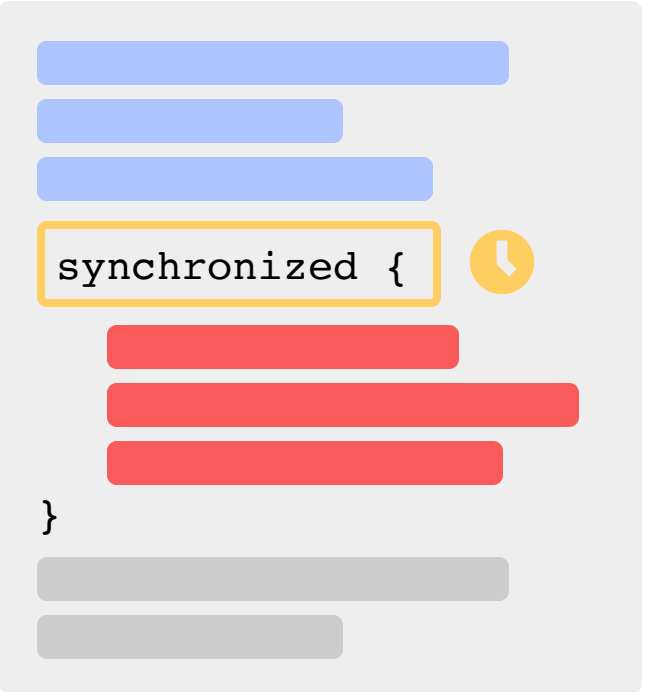


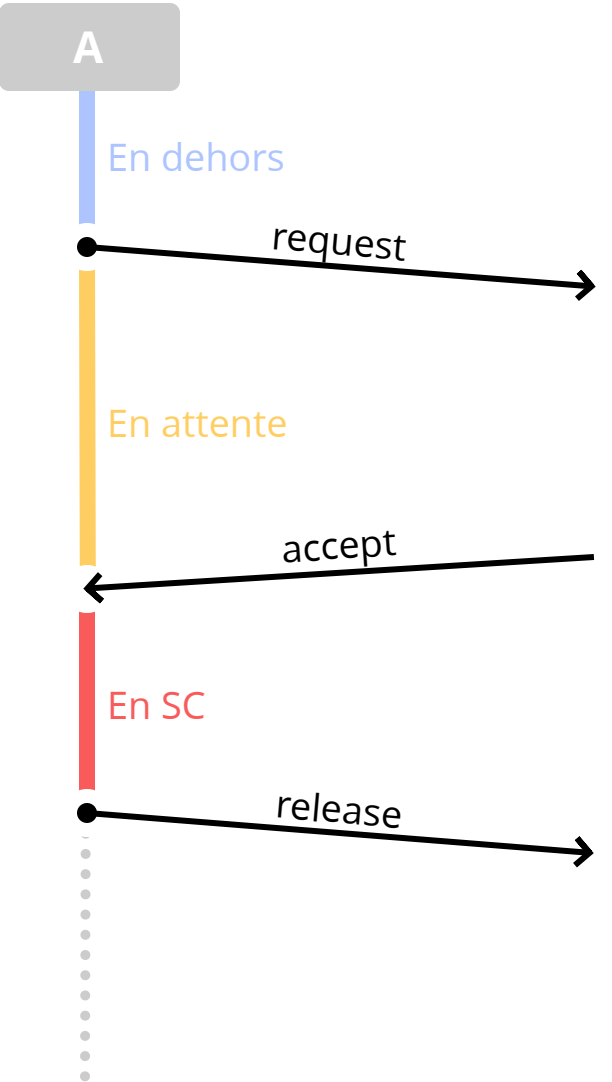
Synchronisation par messages



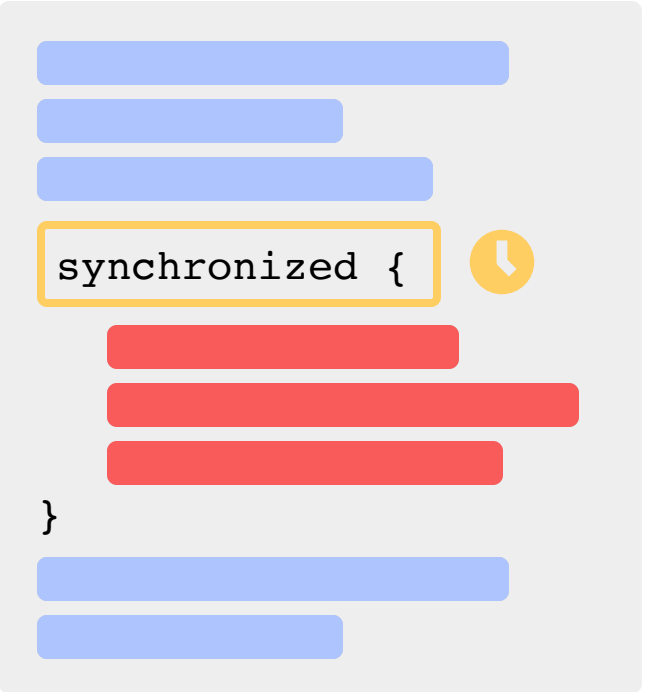


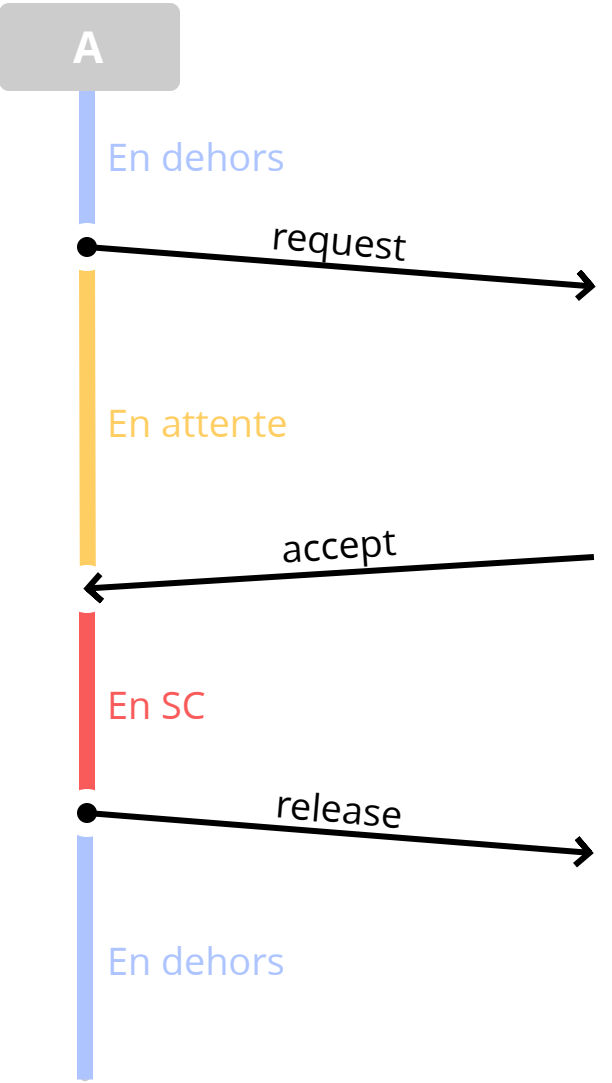
Synchronisation par messages



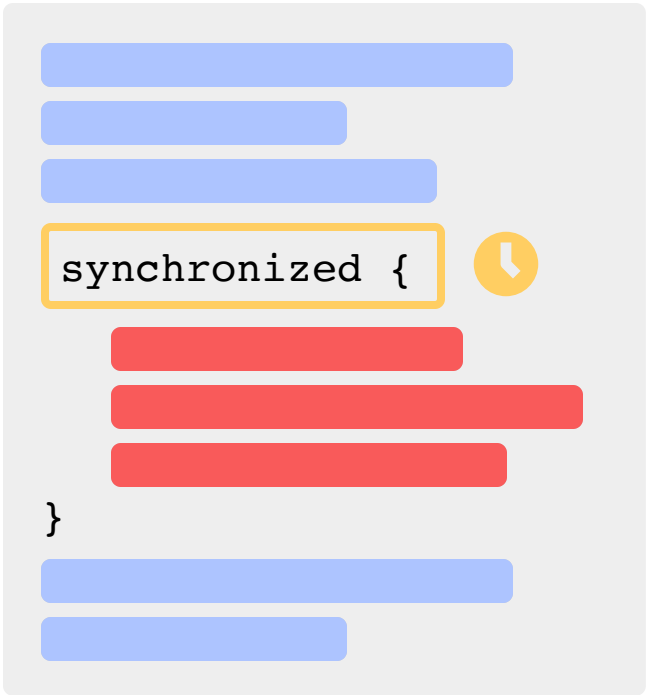


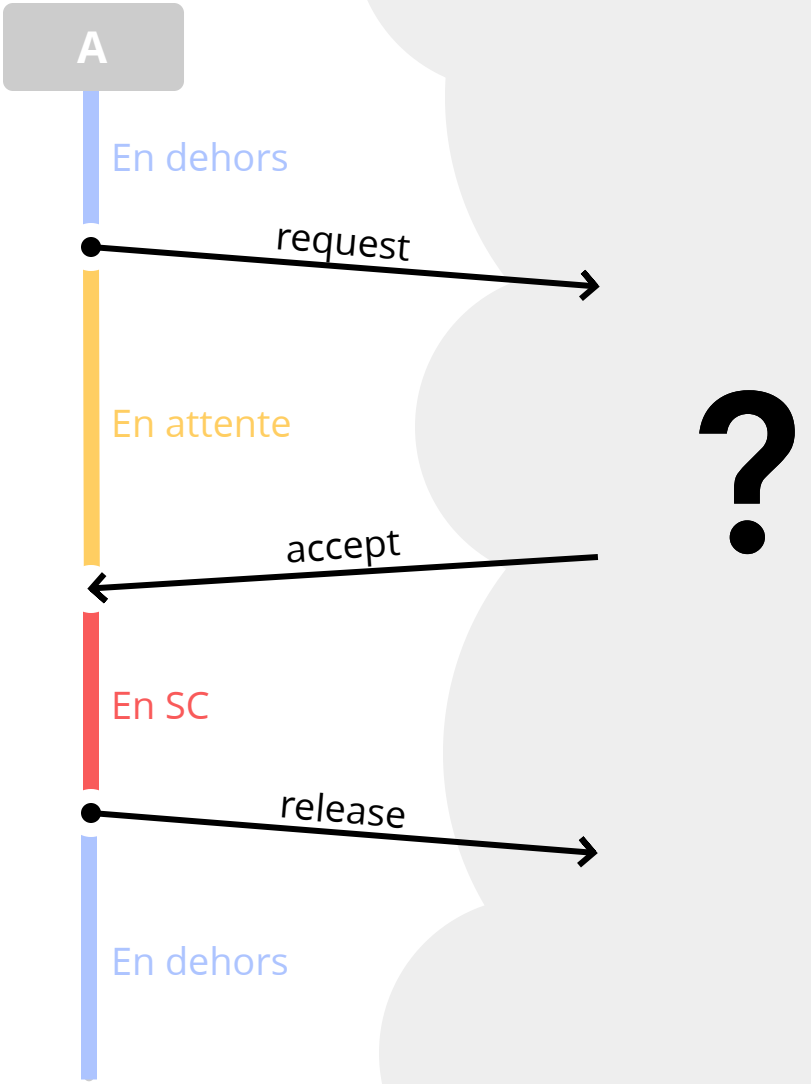
Synchronisation par messages



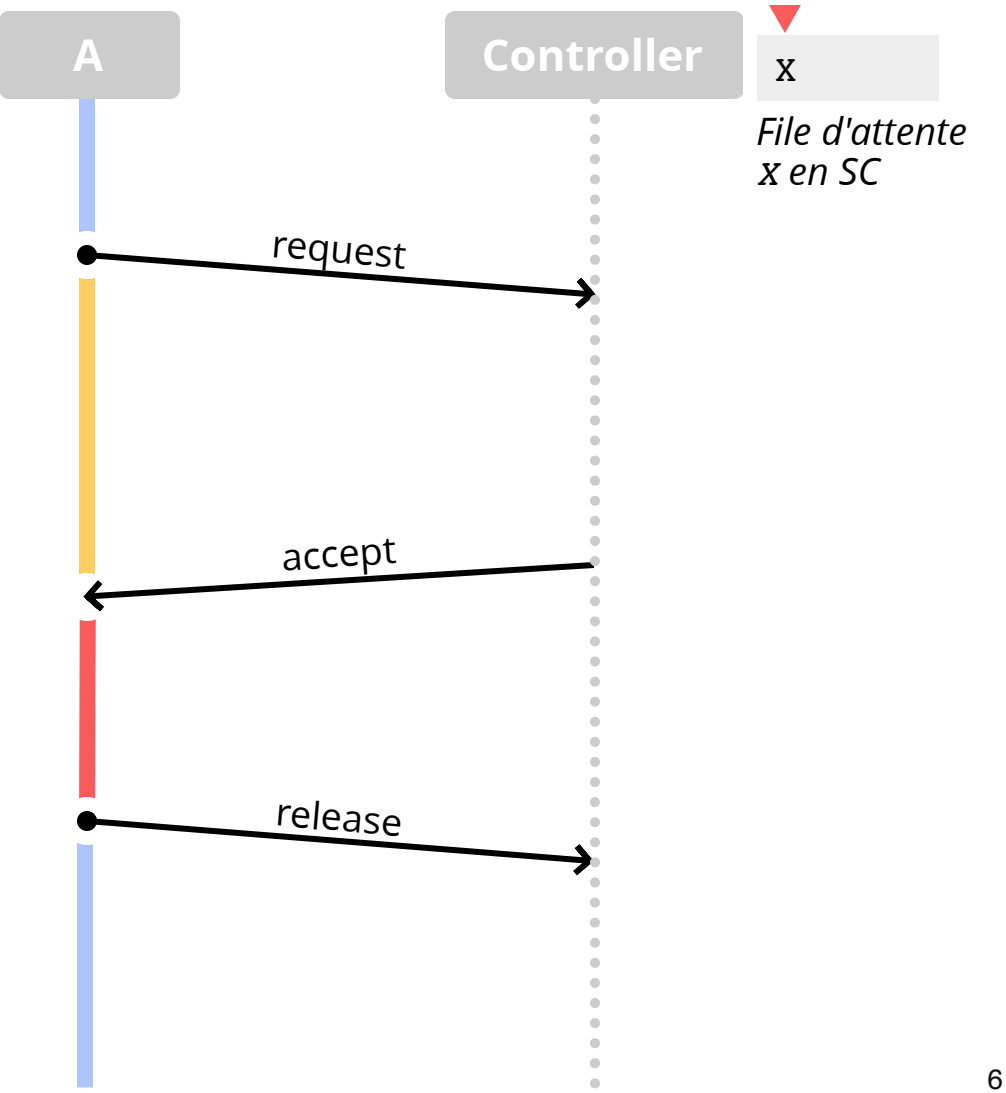


Synchronisation par messages



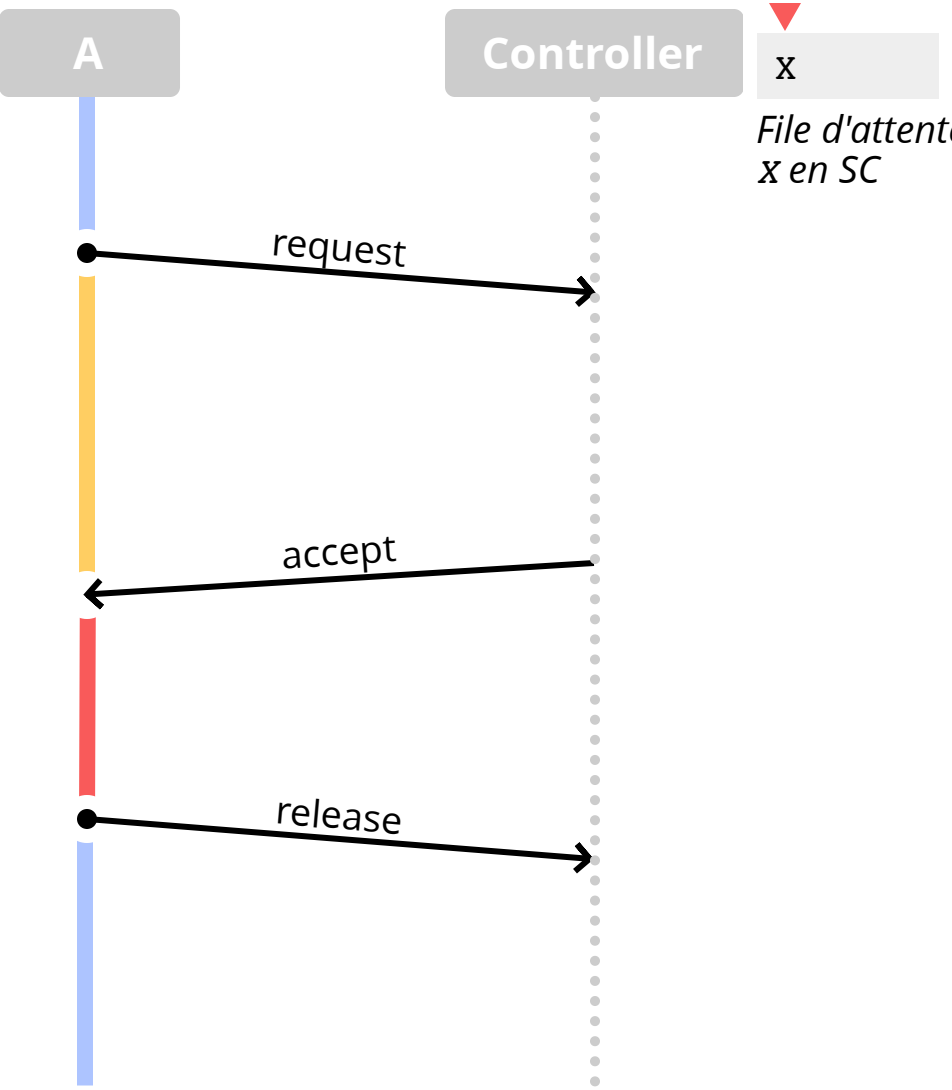


Solution centralisée



Solution centralisée

La tête de la file est toujours celle actuellement en SC.

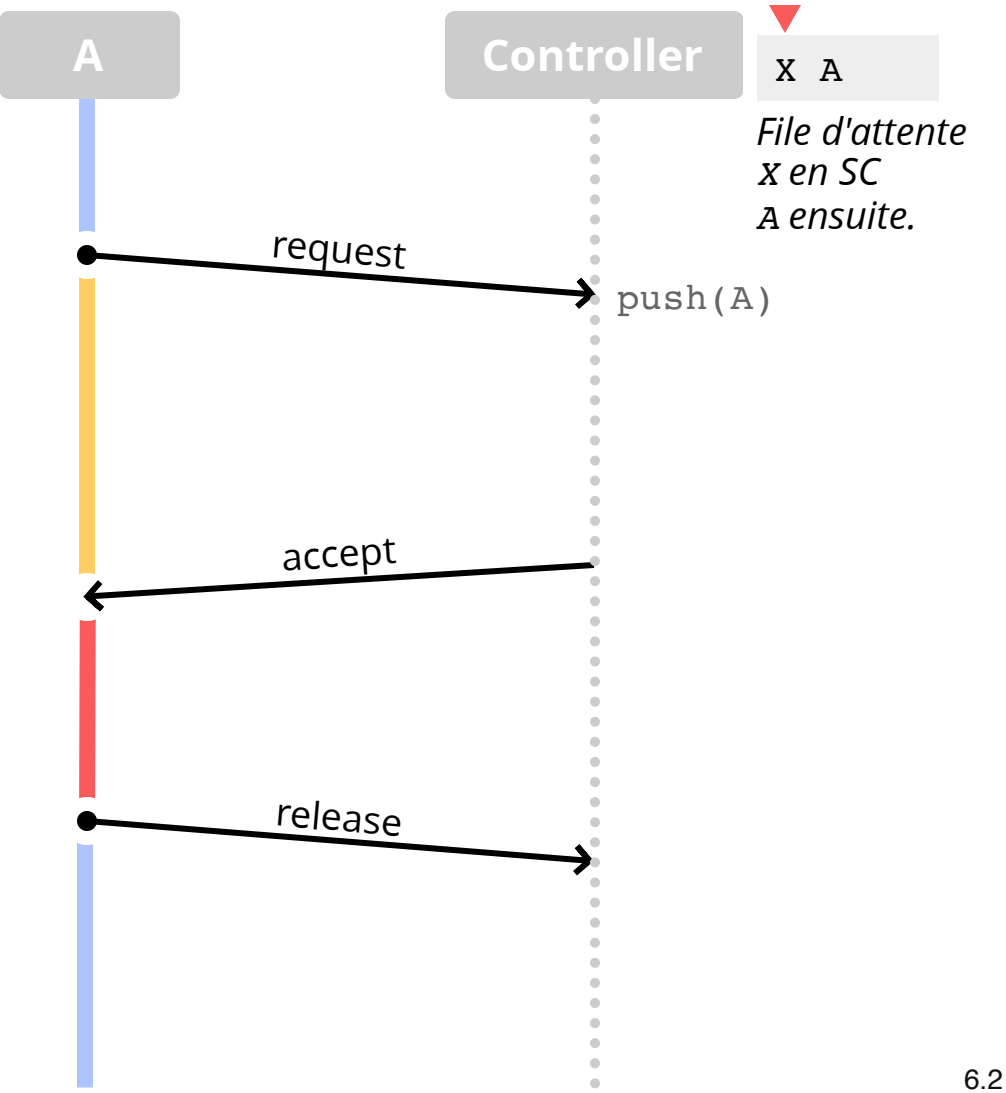


Solution centralisée

La tête de la file est toujours celle actuellement en SC.

On request

Push demandeur dans la file



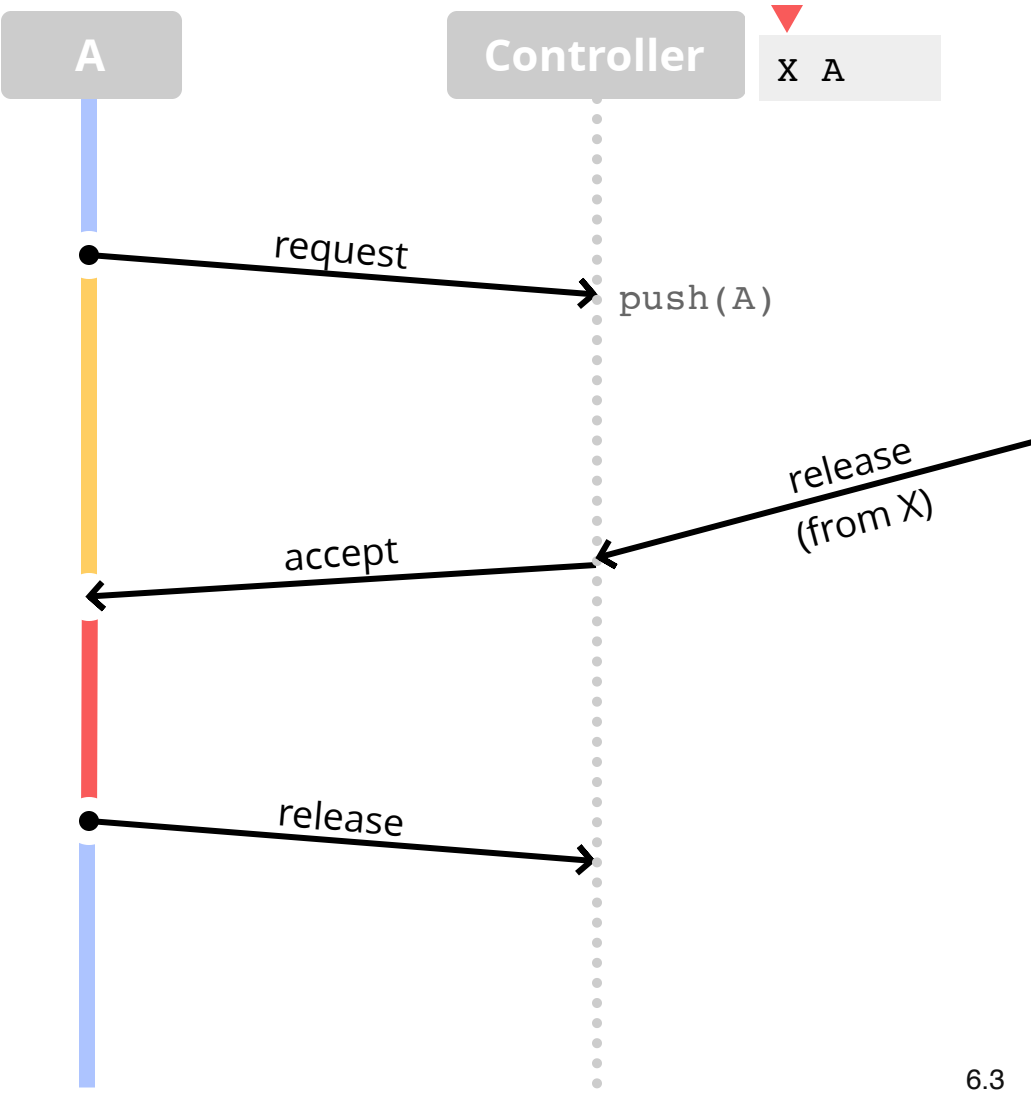
Solution centralisée

La tête de la file est toujours celle actuellement en SC.

On request

Push demandeur dans la file

On release



Solution centralisée

La tête de la file est toujours celle actuellement en SC.

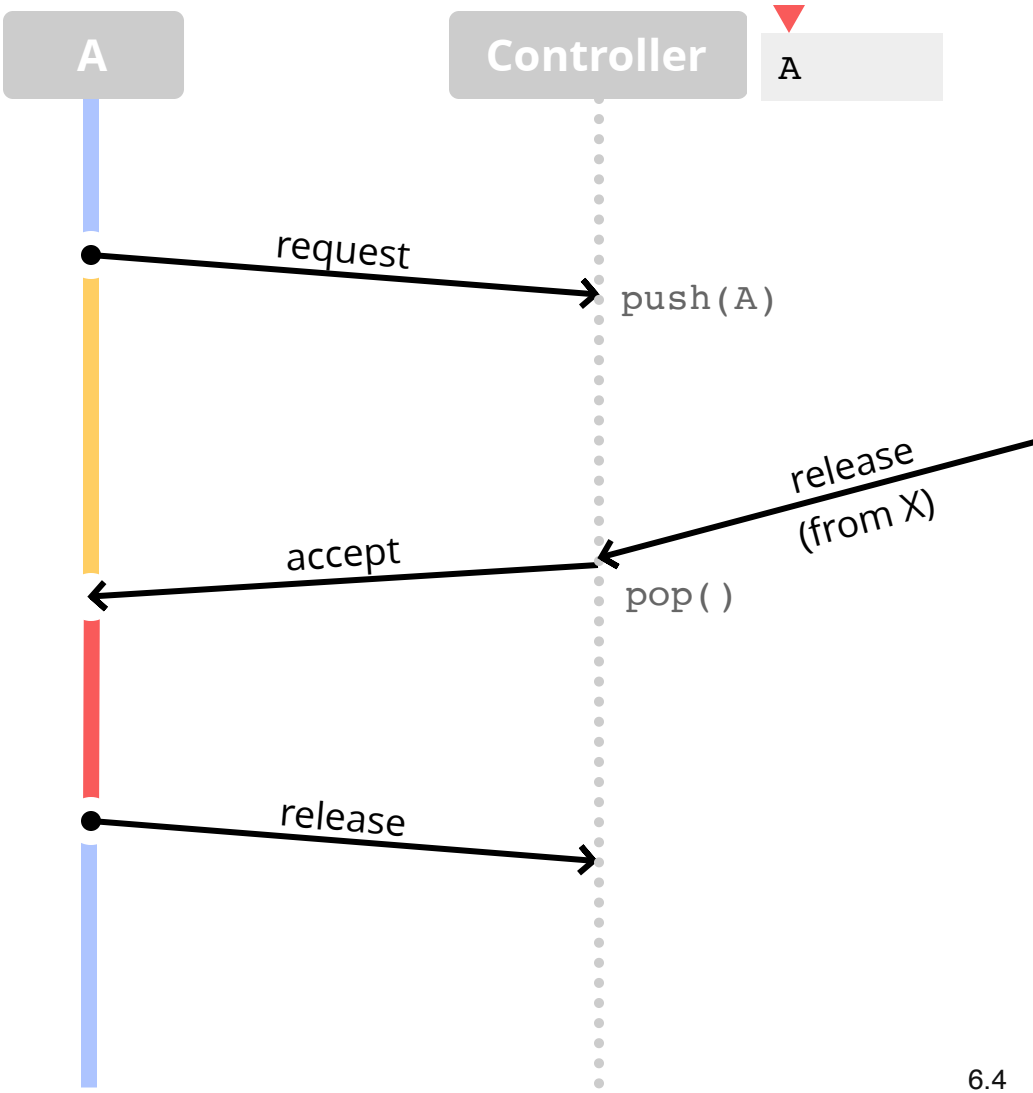
On request

Push demandeur dans la file

On release

Pop tête de file

Envoi de `accept` à nouvelle tête



Solution centralisée

La tête de la file est toujours celle actuellement en SC.

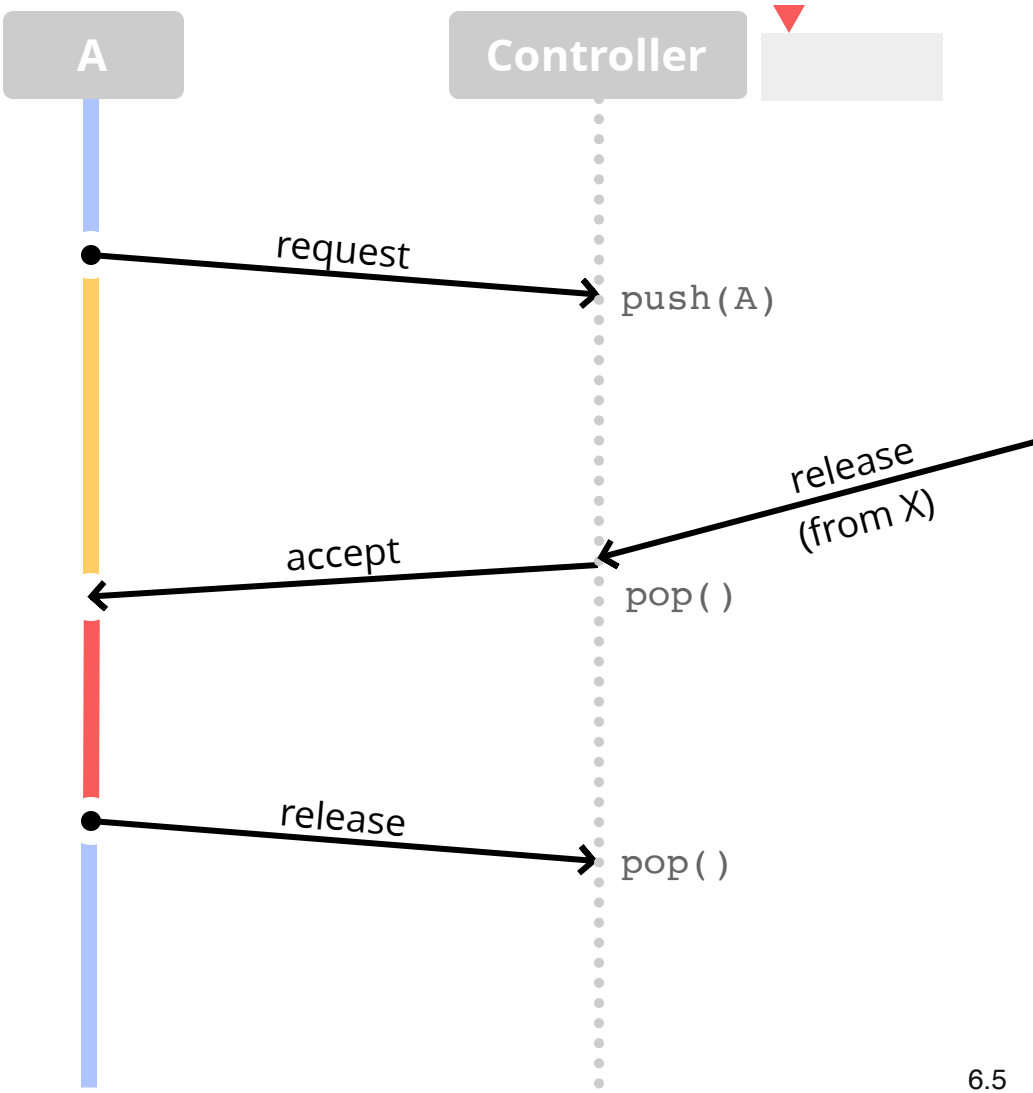
On request

Push demandeur dans la file

On release

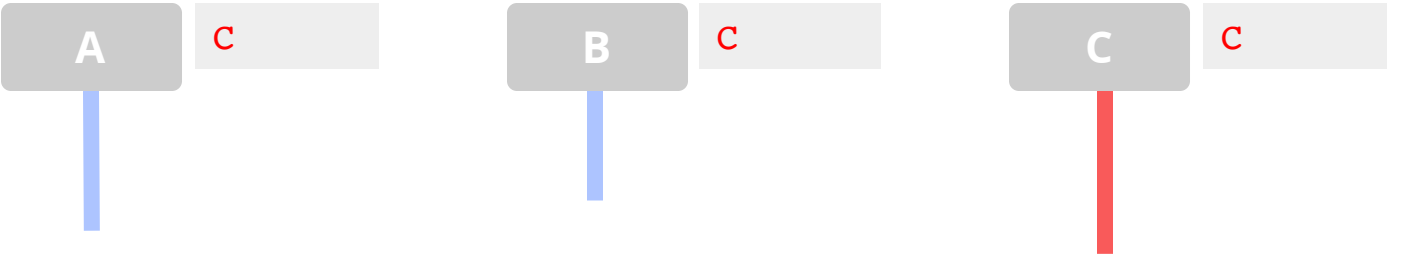
Pop tête de file

Envoi de `accept` à nouvelle tête



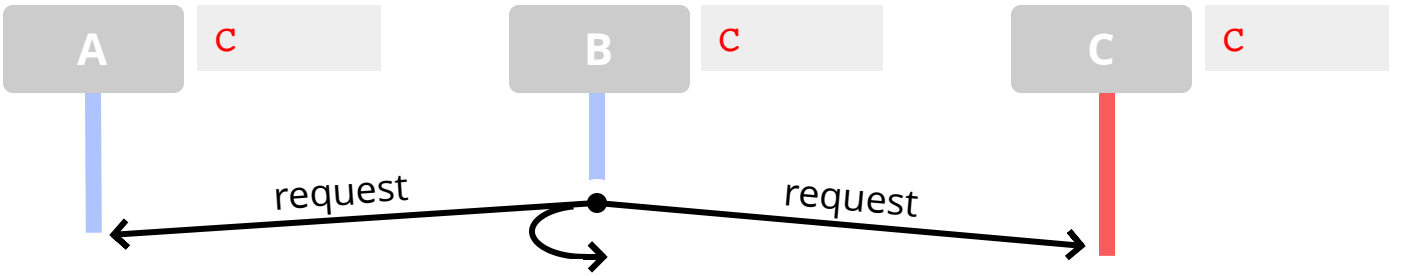
Solution répartie

Version naïve



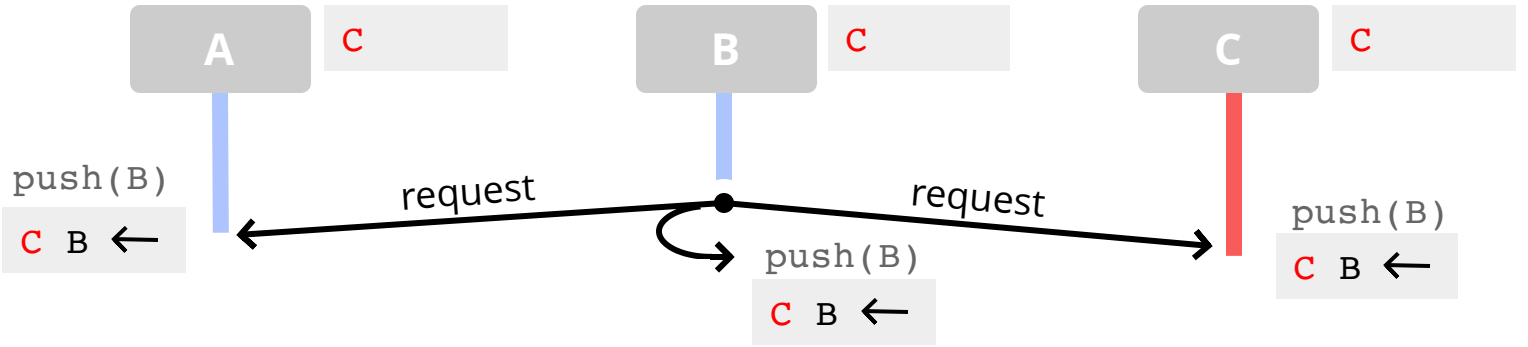
Solution répartie

Version naïve



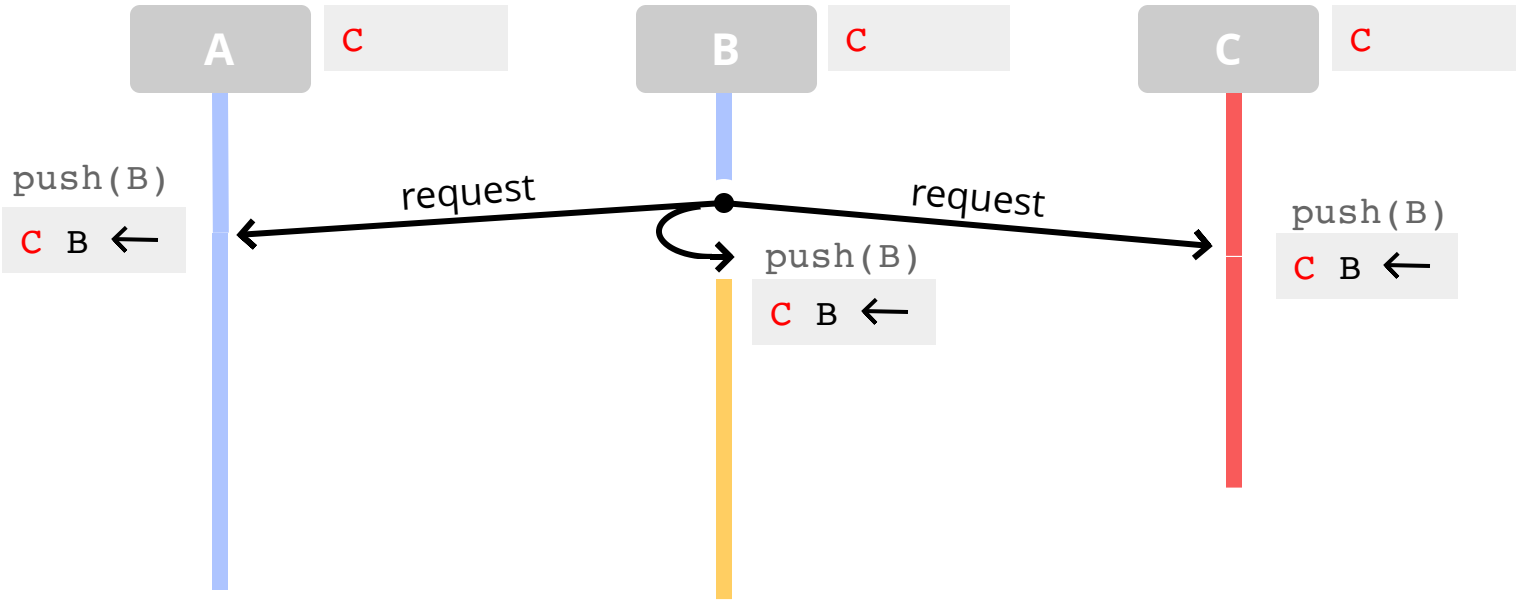
Solution répartie

Version naïve



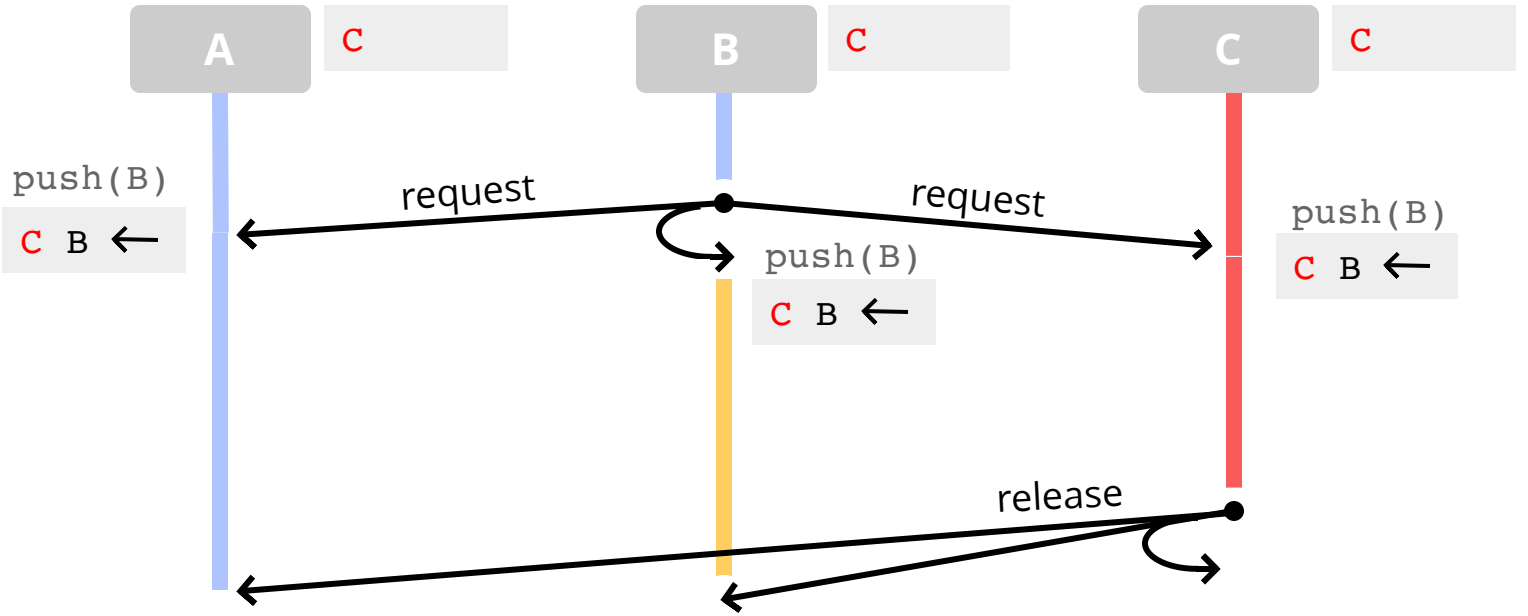
Solution répartie

Version naïve



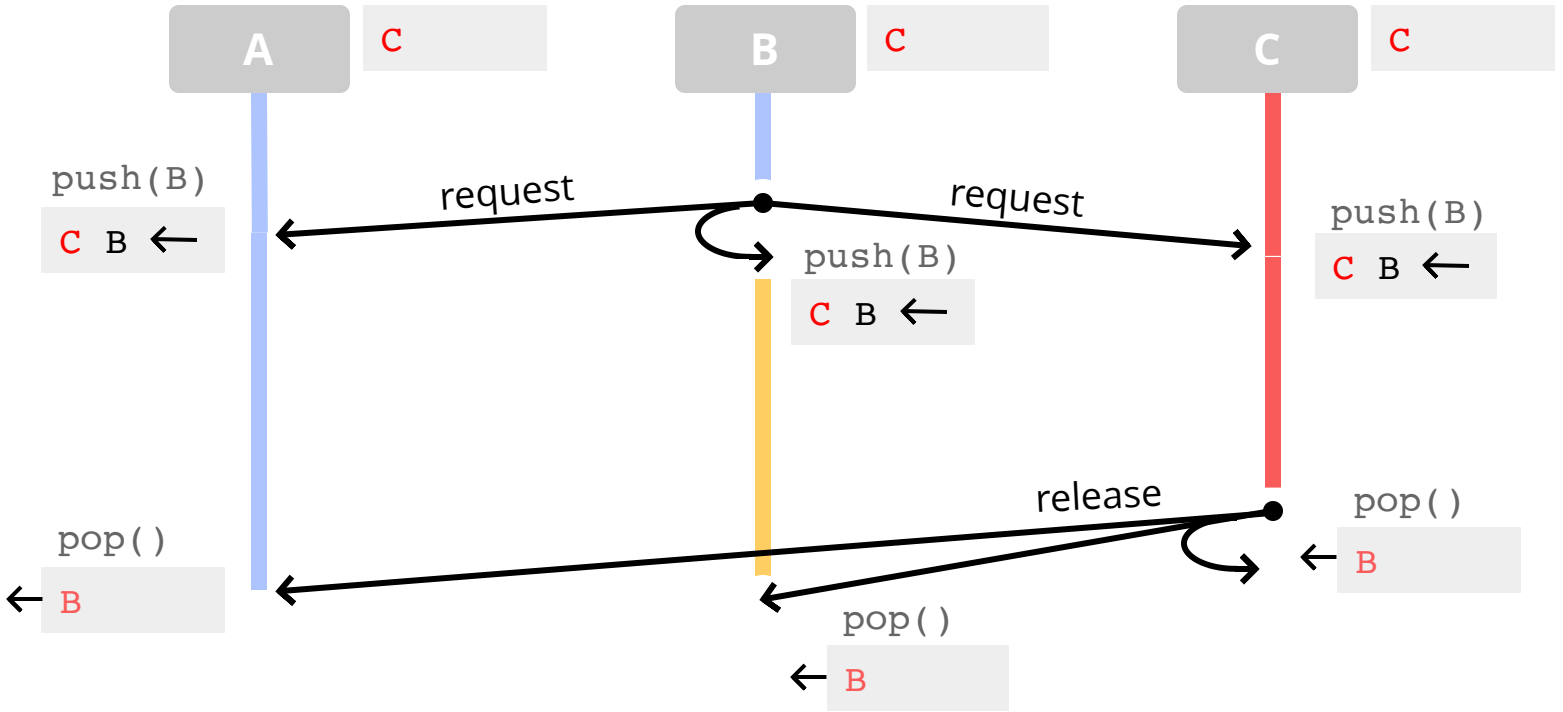
Solution répartie

Version naïve



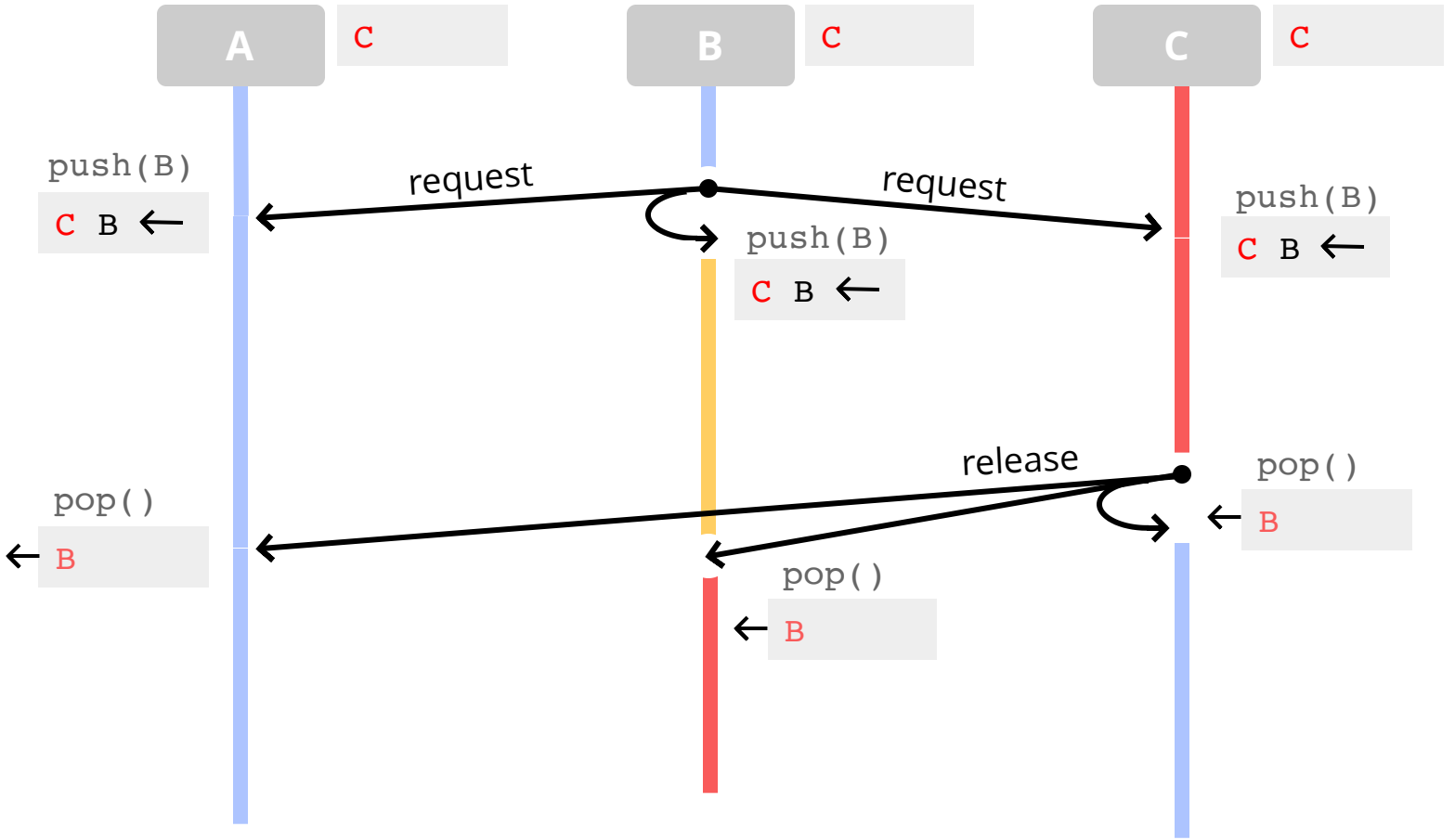
Solution répartie

Version naïve



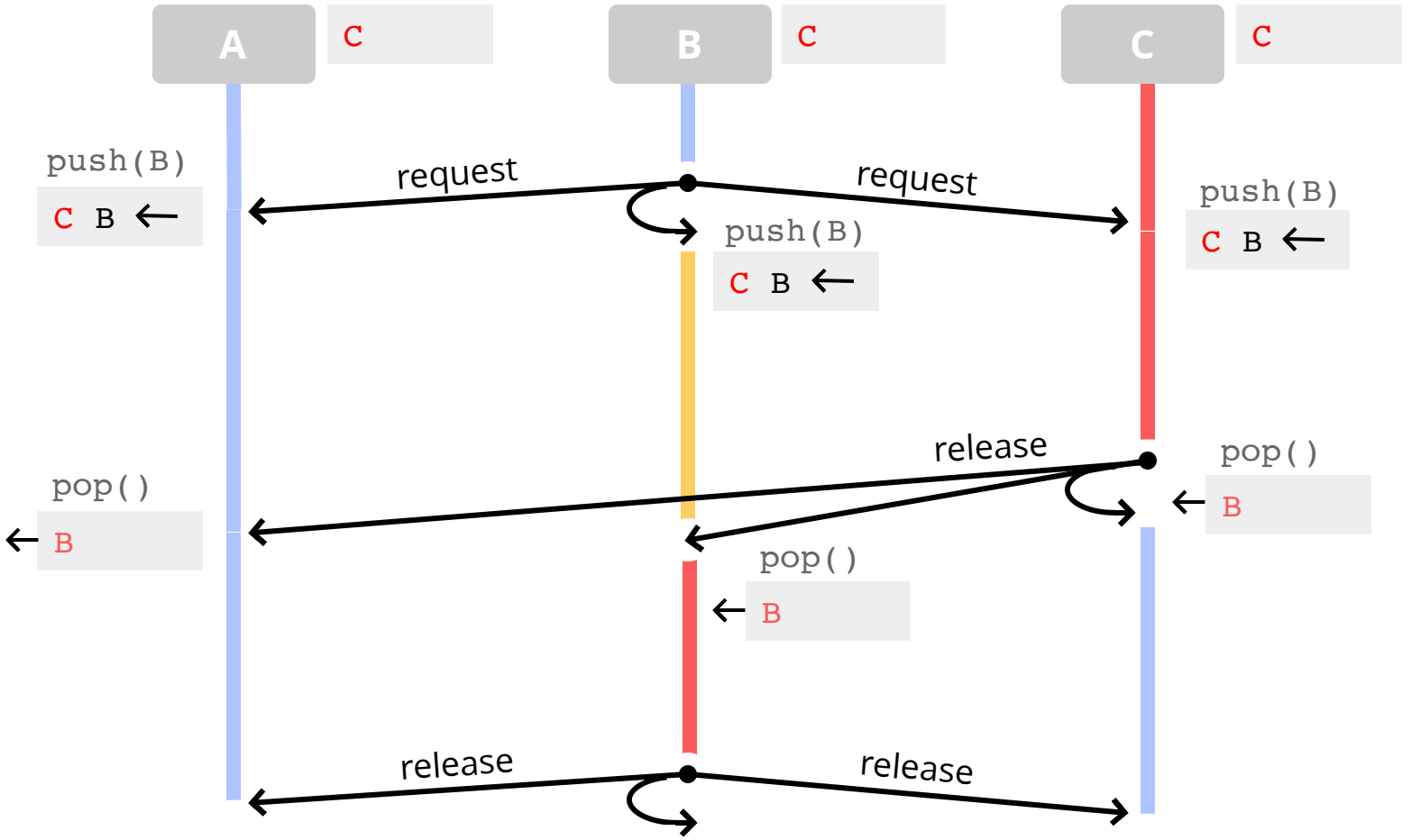
Solution répartie

Version naïve



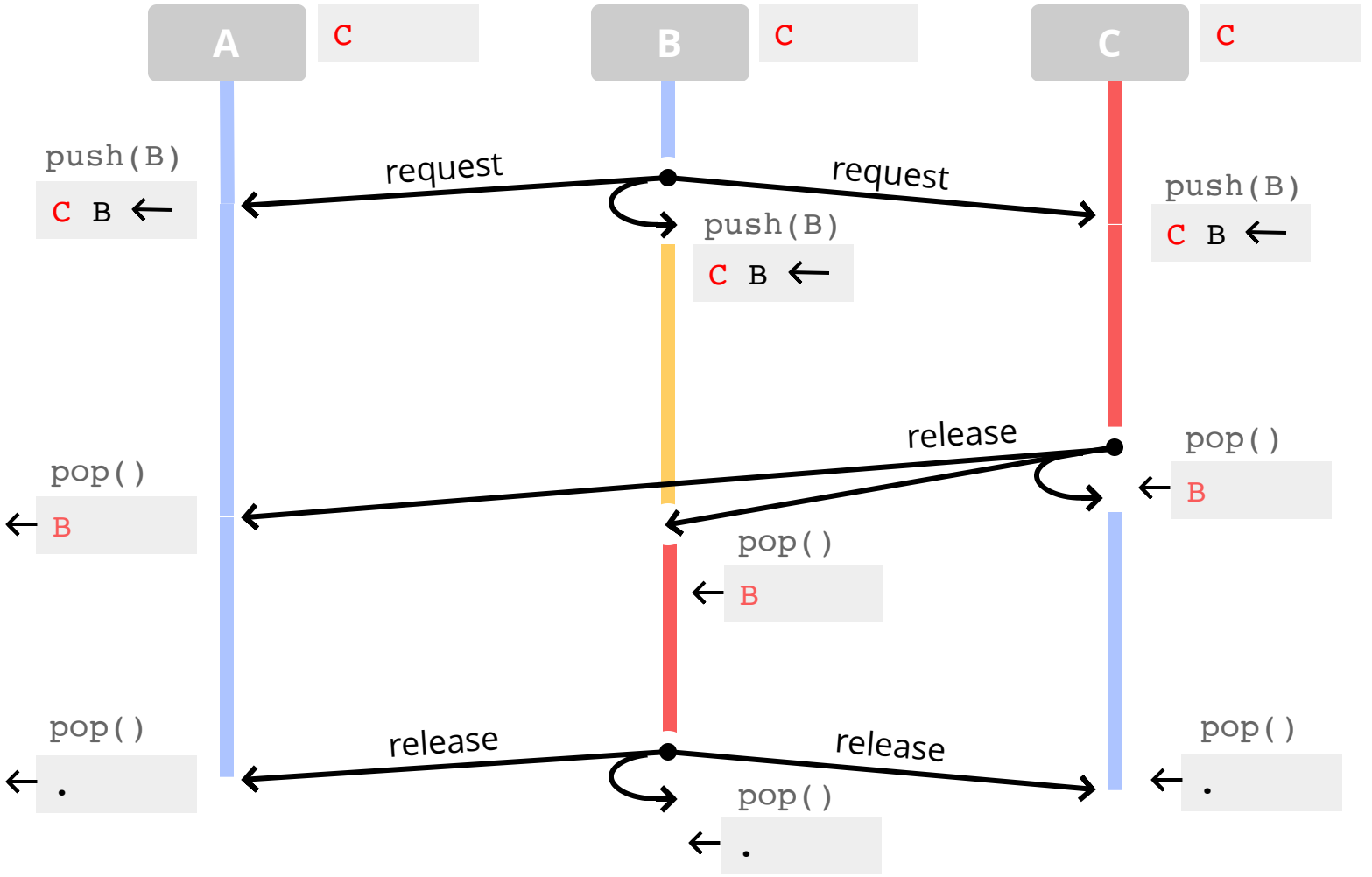
Solution répartie

Version naïve



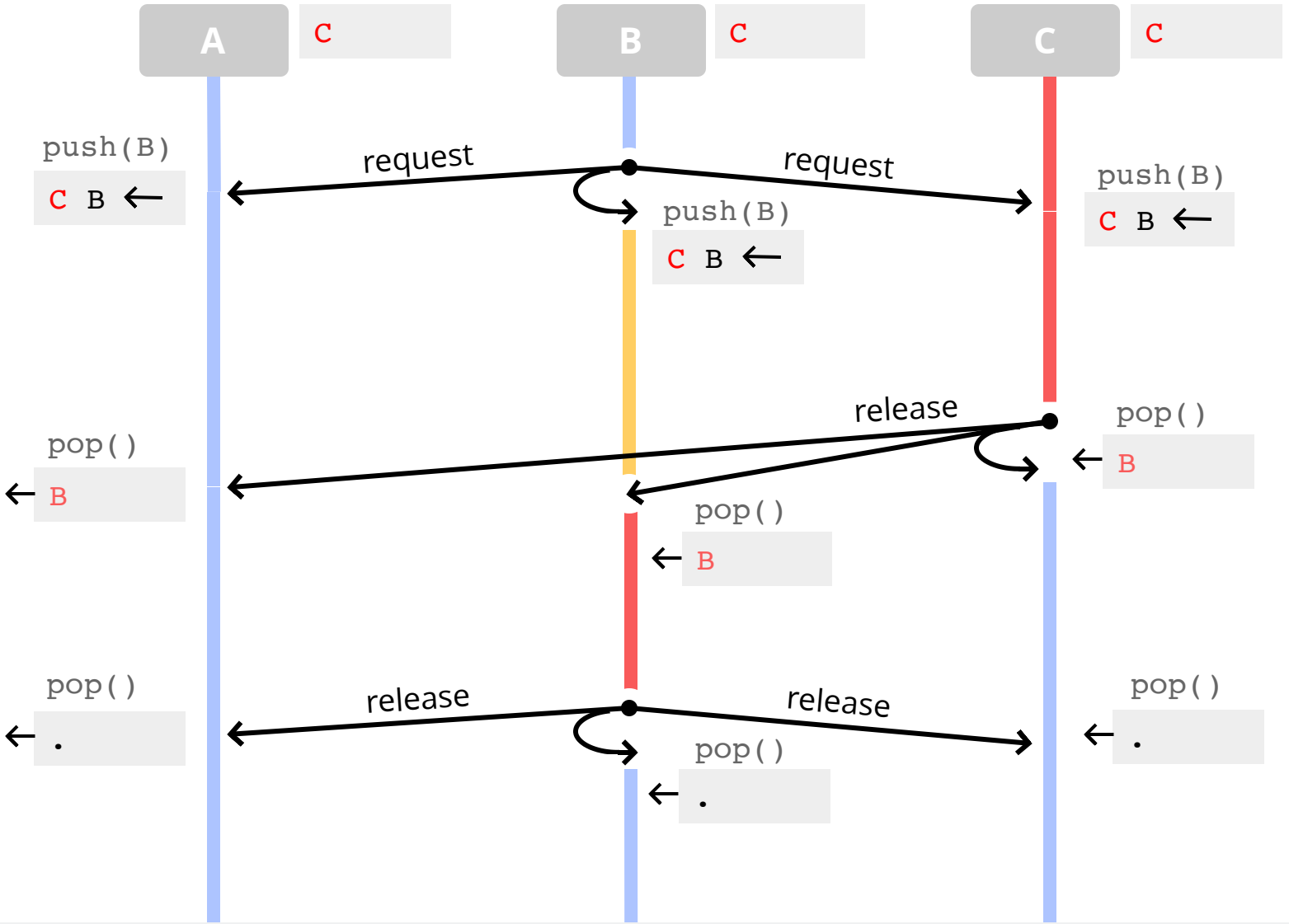
Solution répartie

Version naïve



Solution répartie

Version naïve



Solution répartie

Version naïve - Résumé

La tête de la file est toujours
celle actuellement en SC.

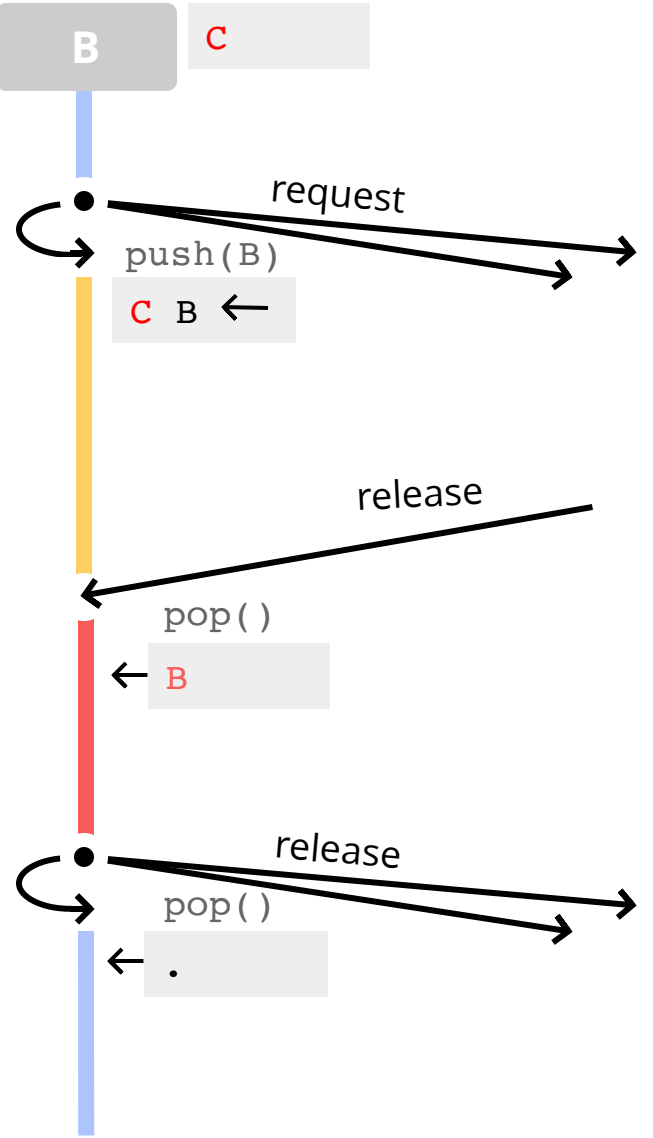
On request

Push demandeur dans la file

On release

Pop tête de file

Si je suis nouvelle tête, entrer en SC.



Solution répartie

Version naïve - Résumé

La tête de la file est toujours
celle actuellement en SC.

On request

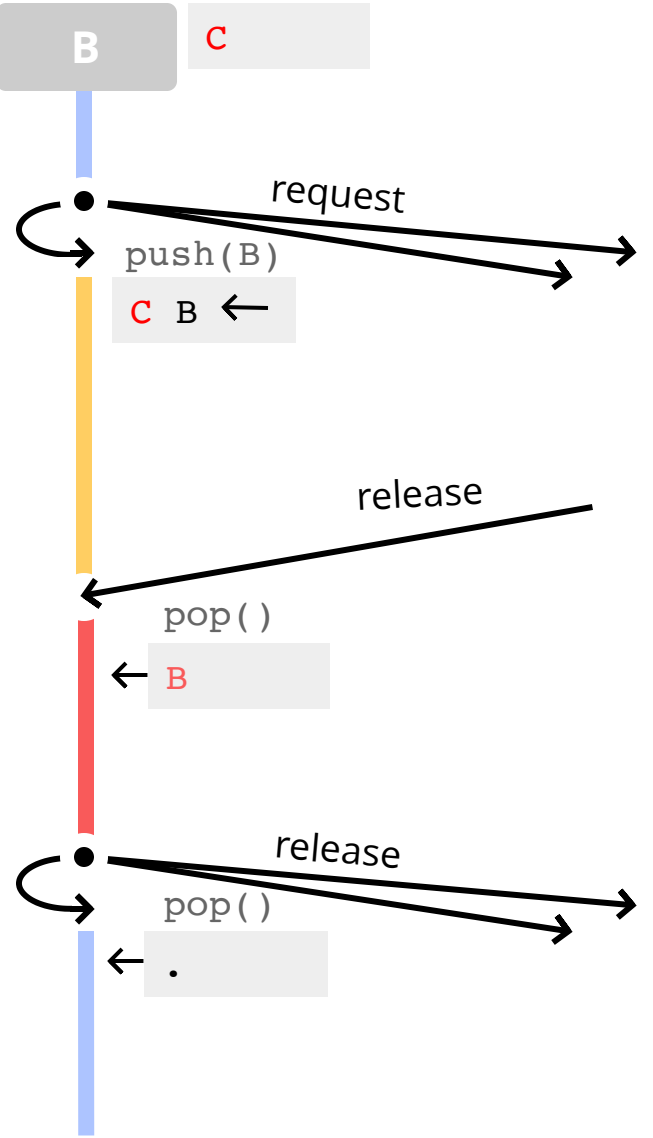
Push demandeur dans la file

On release

Pop tête de file

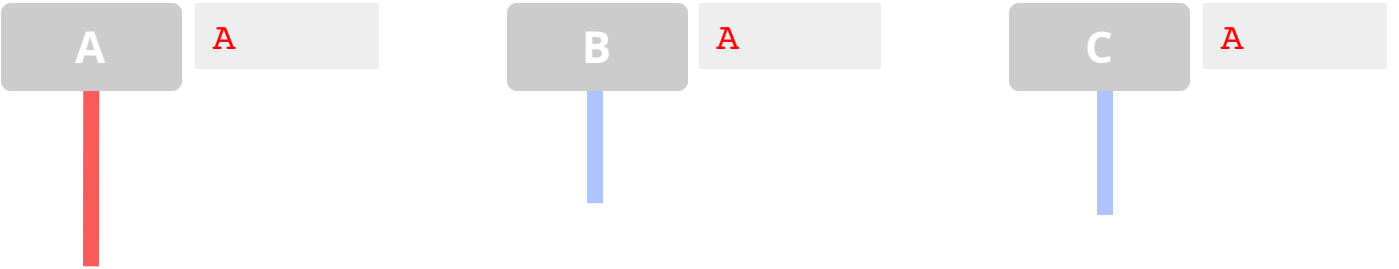
Si je suis nouvelle tête, entrer en SC.

Problème ?



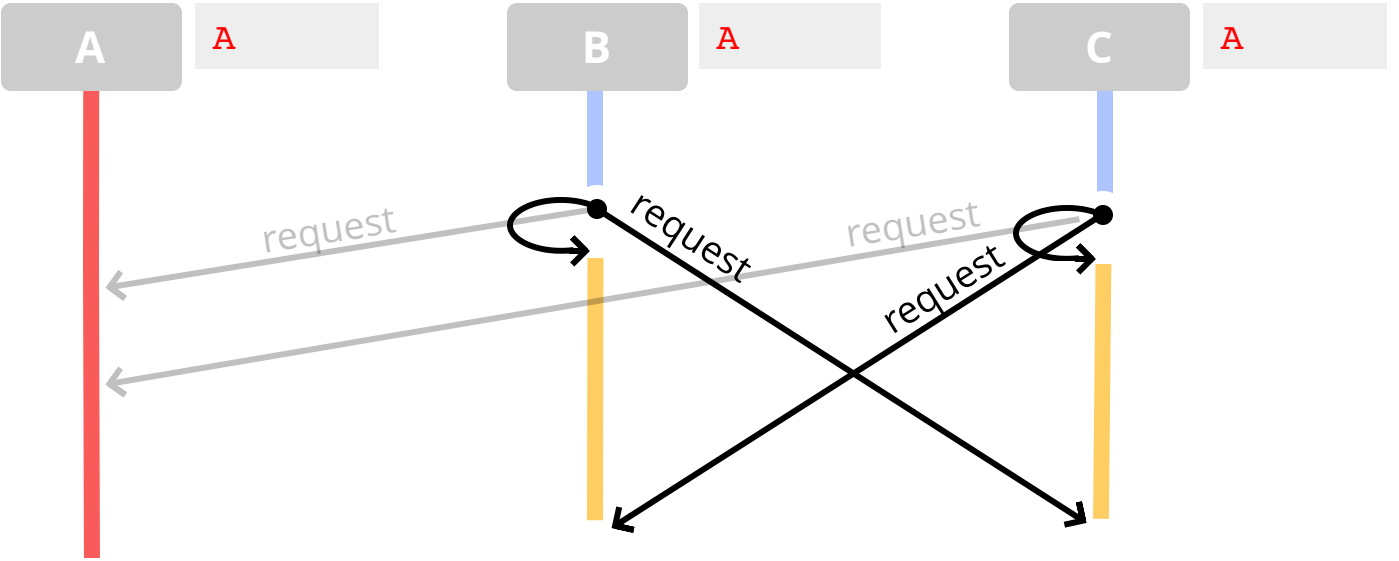
Solution répartie

Version naïve - Problème



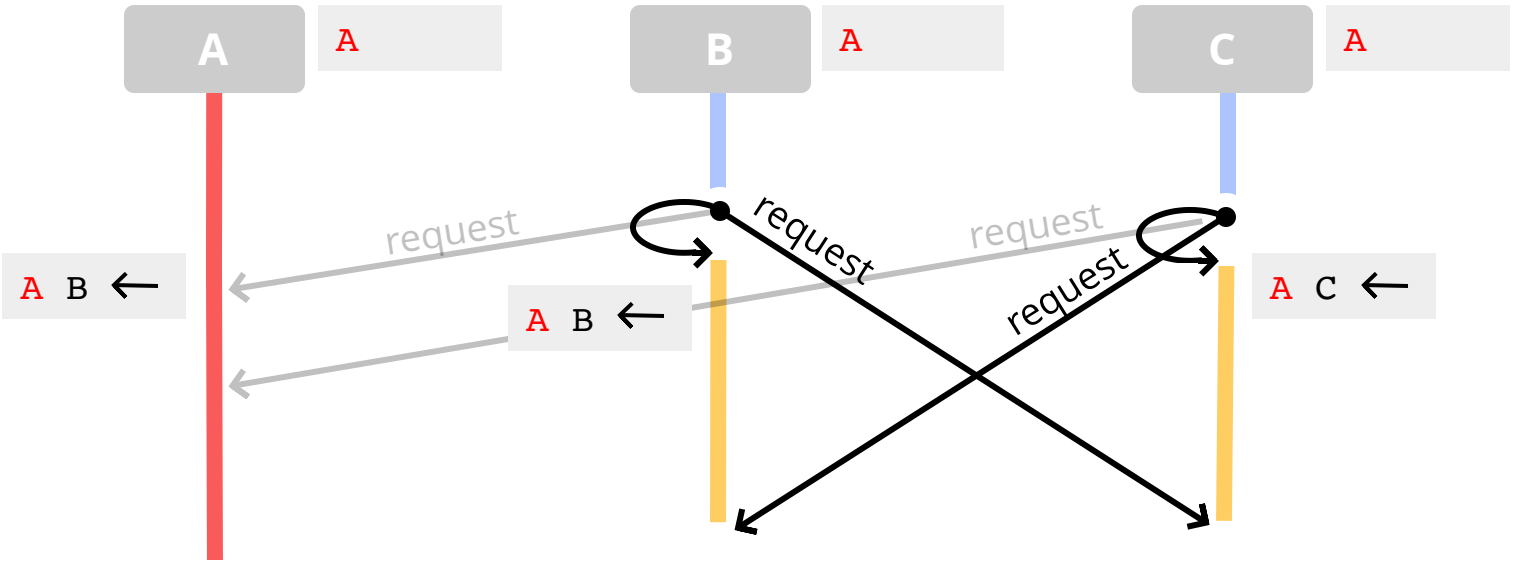
Solution répartie

Version naïve - Problème



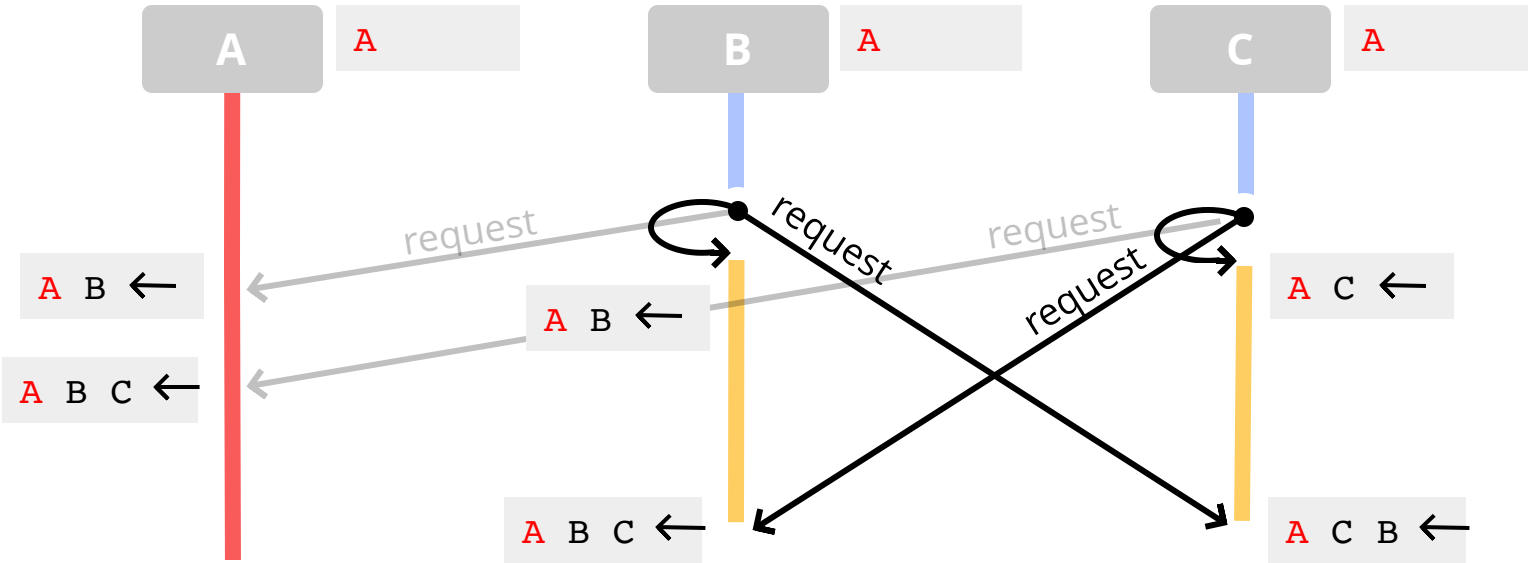
Solution répartie

Version naïve - Problème



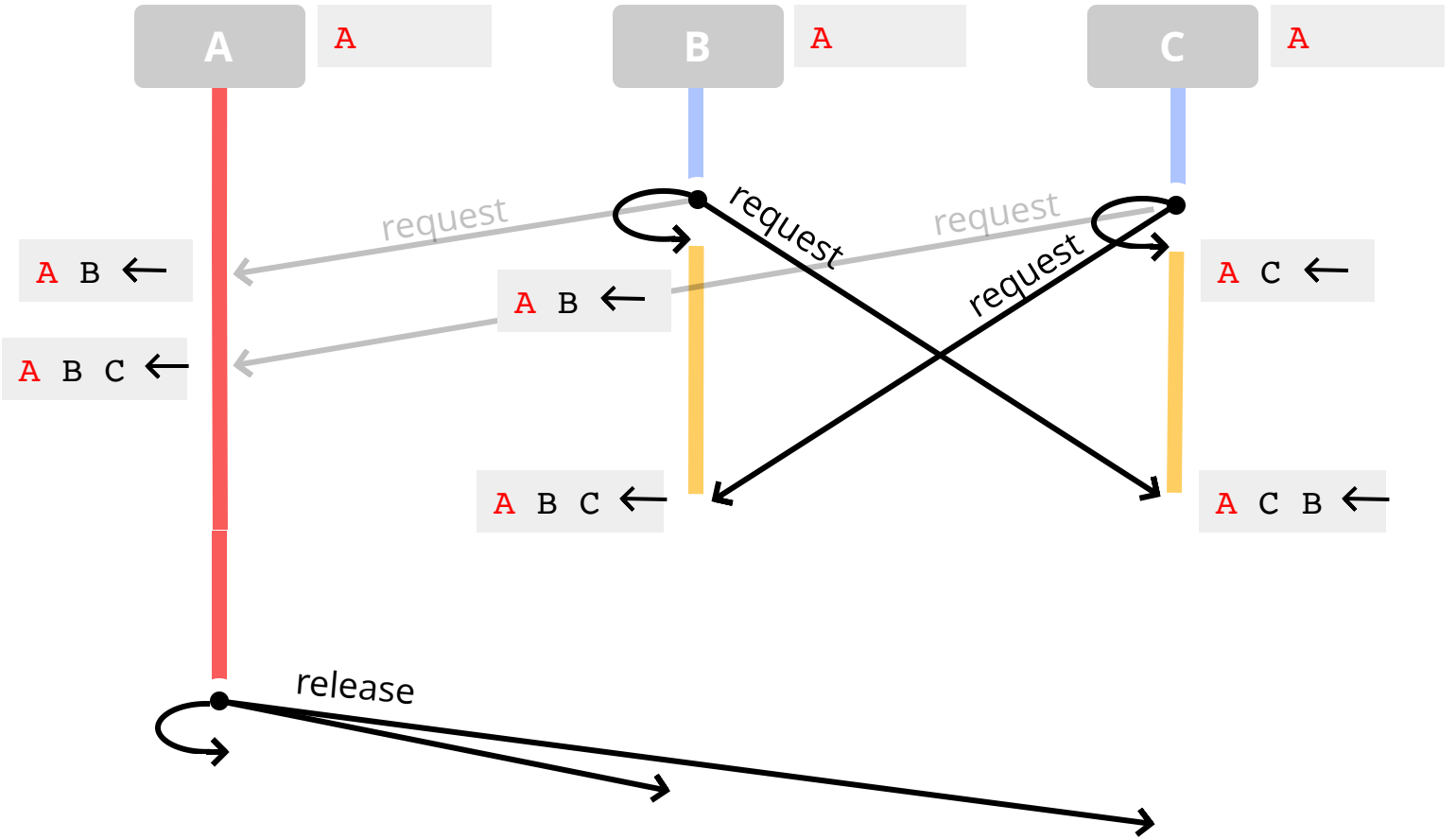
Solution répartie

Version naïve - Problème



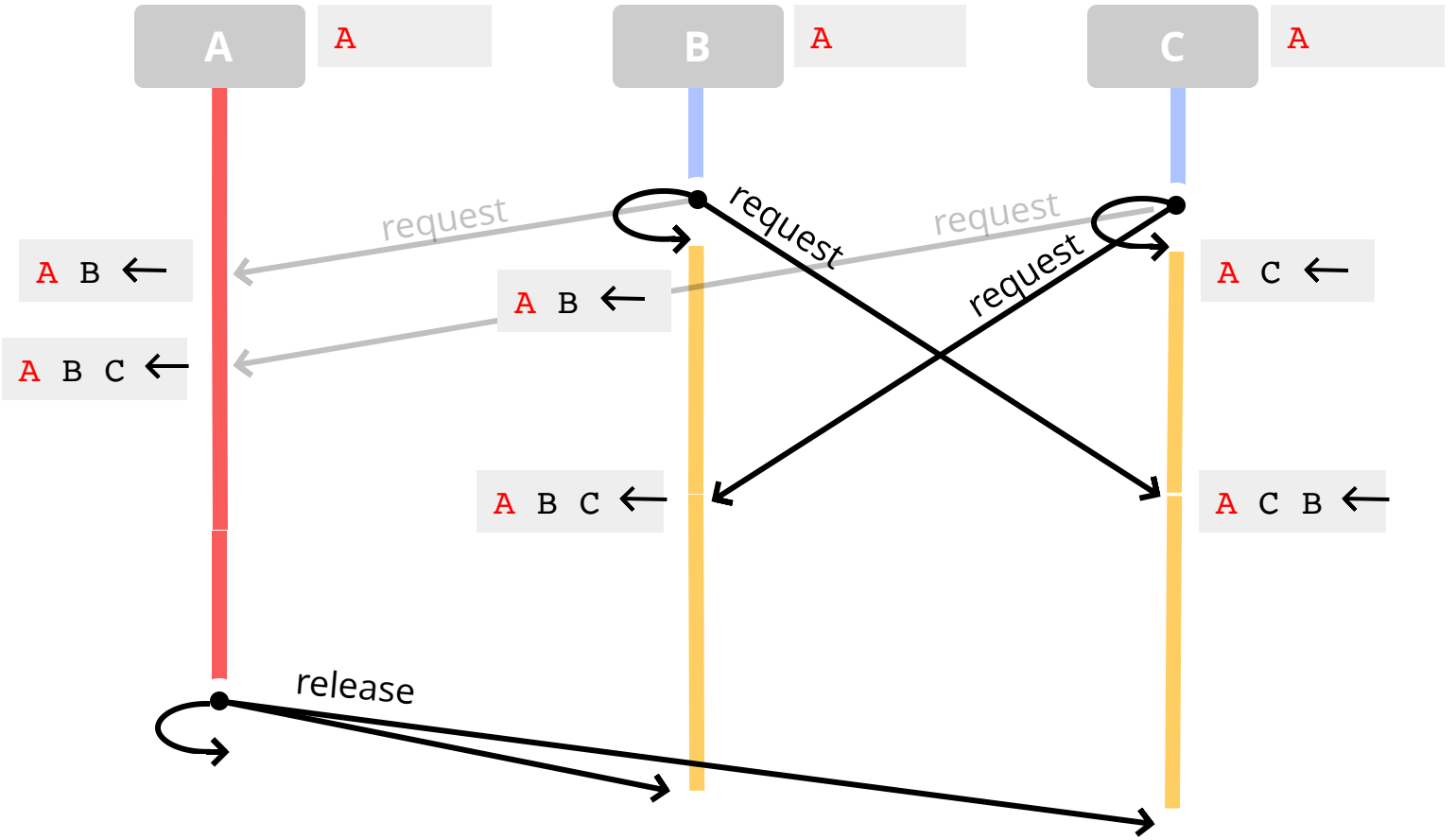
Solution répartie

Version naïve - Problème



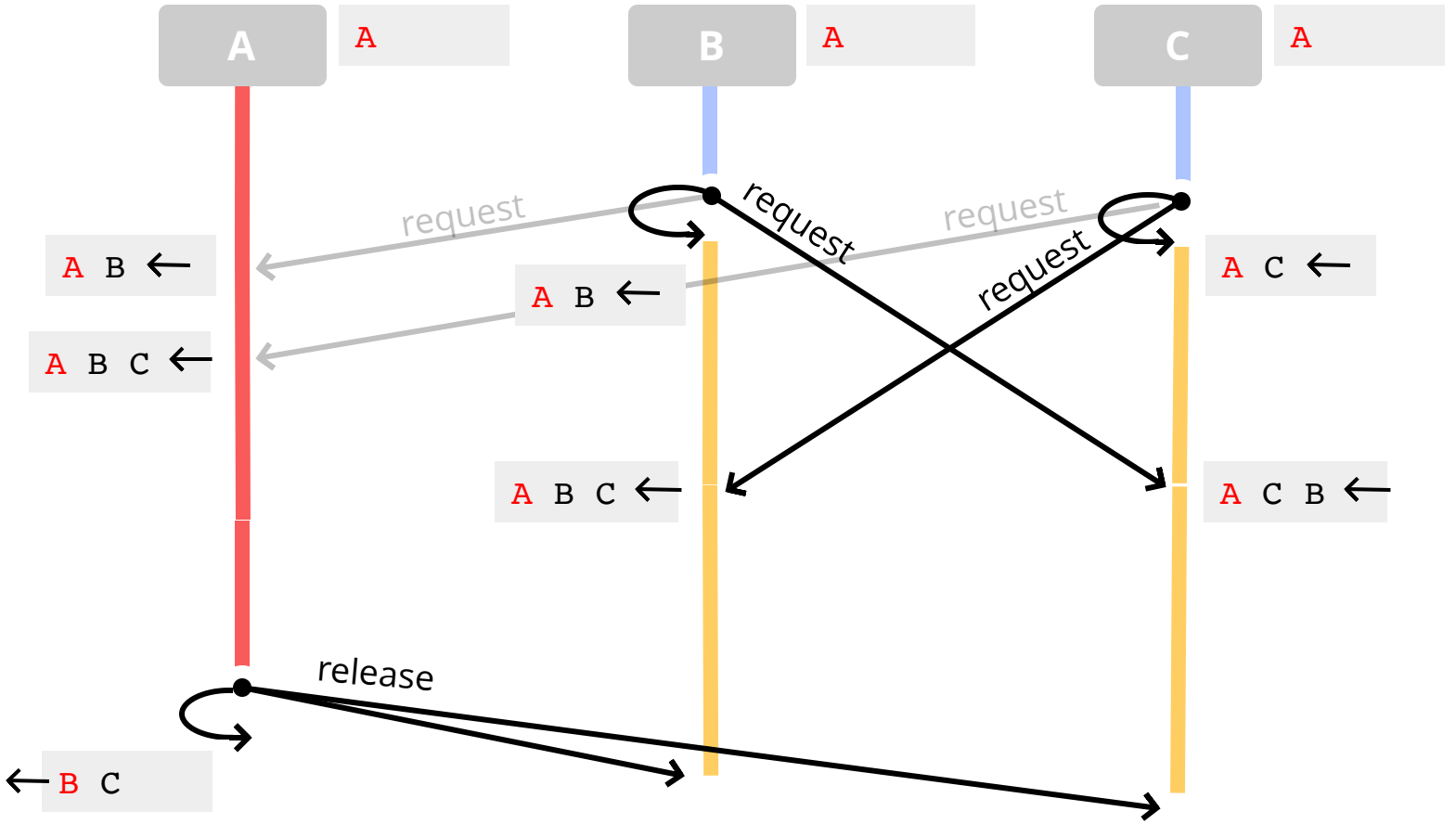
Solution répartie

Version naïve - Problème



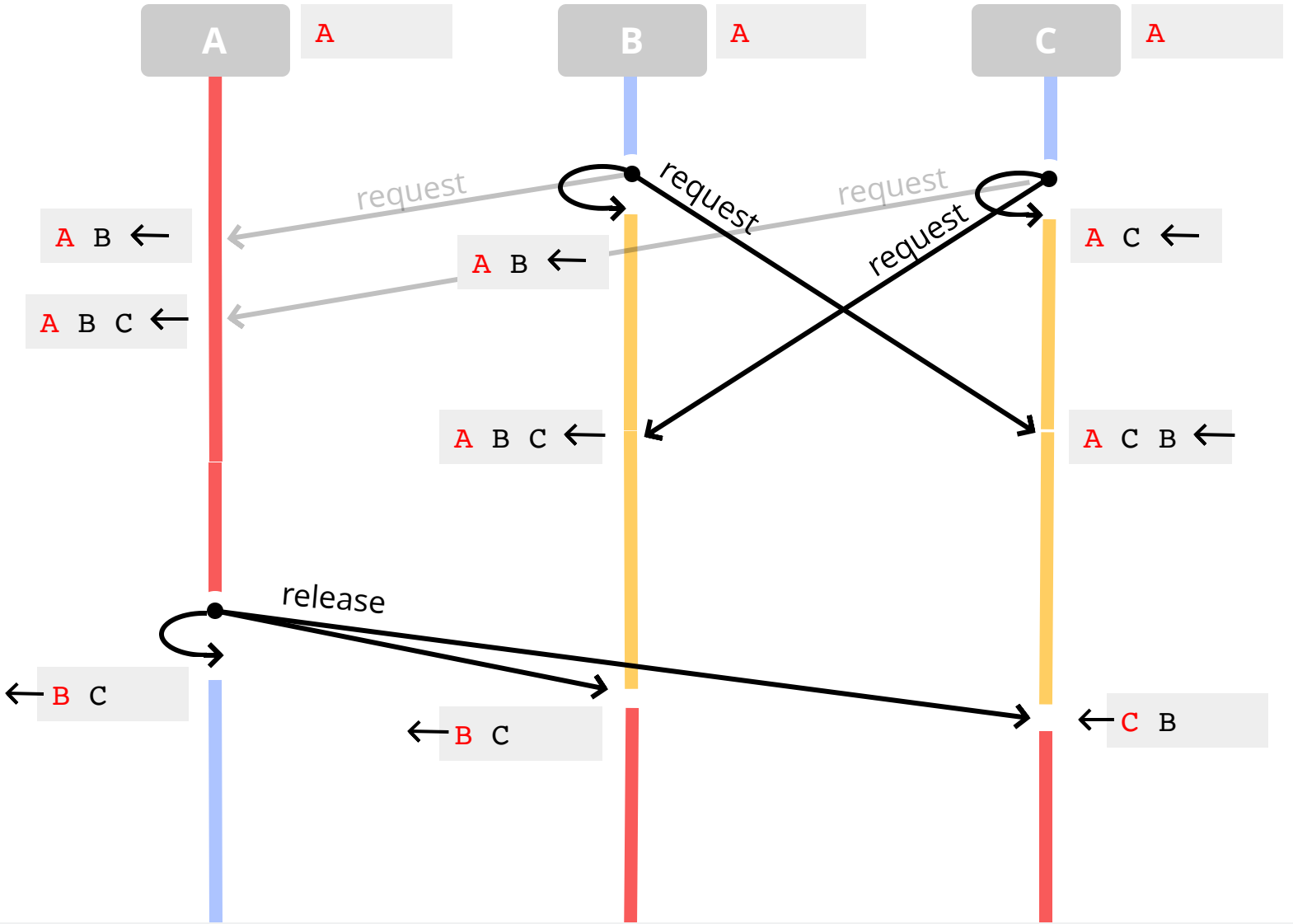
Solution répartie

Version naïve - Problème



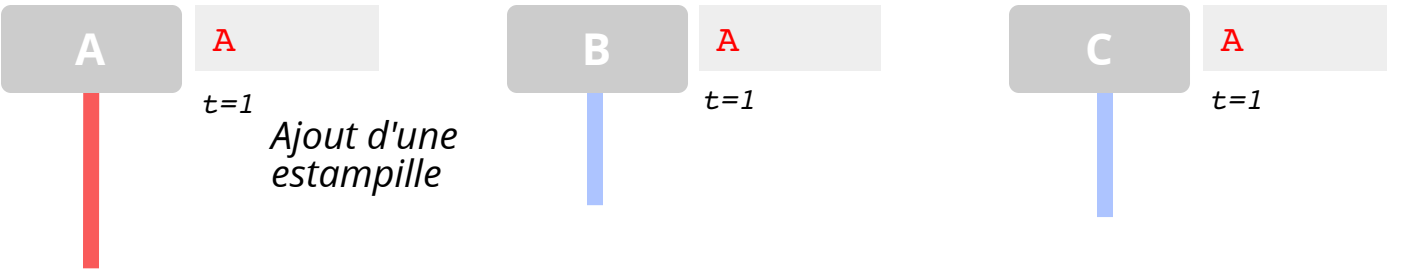
Solution répartie

Version naïve - Problème



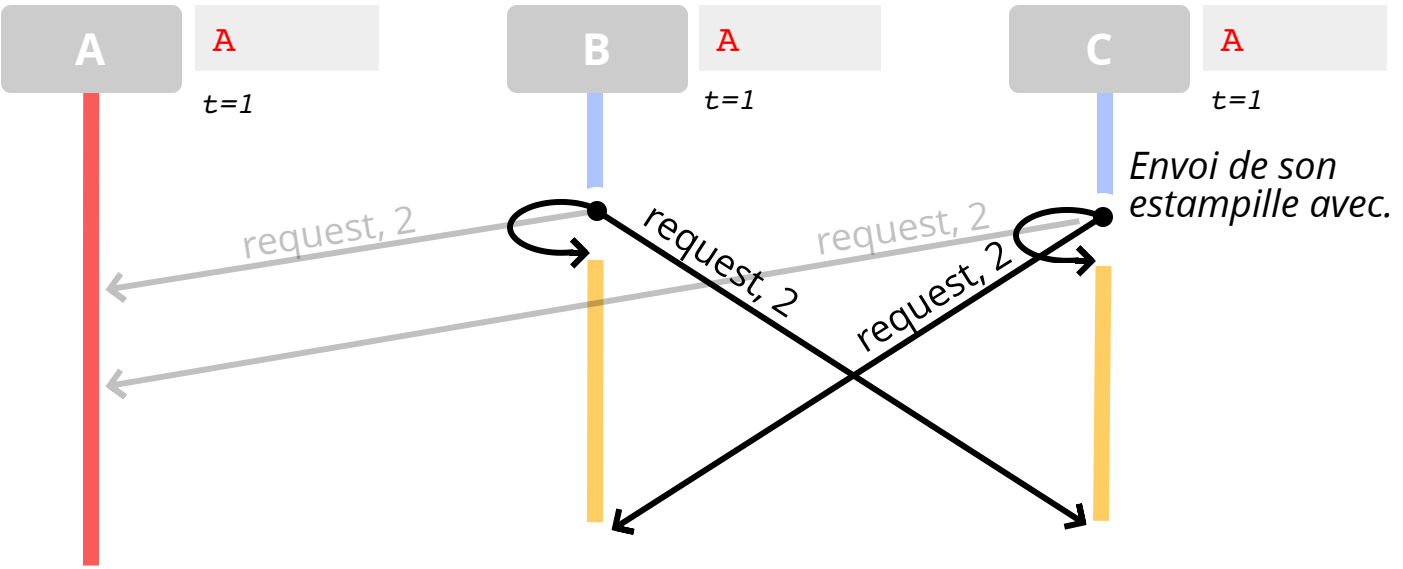
Solution répartie

Version naïve *corrigée*



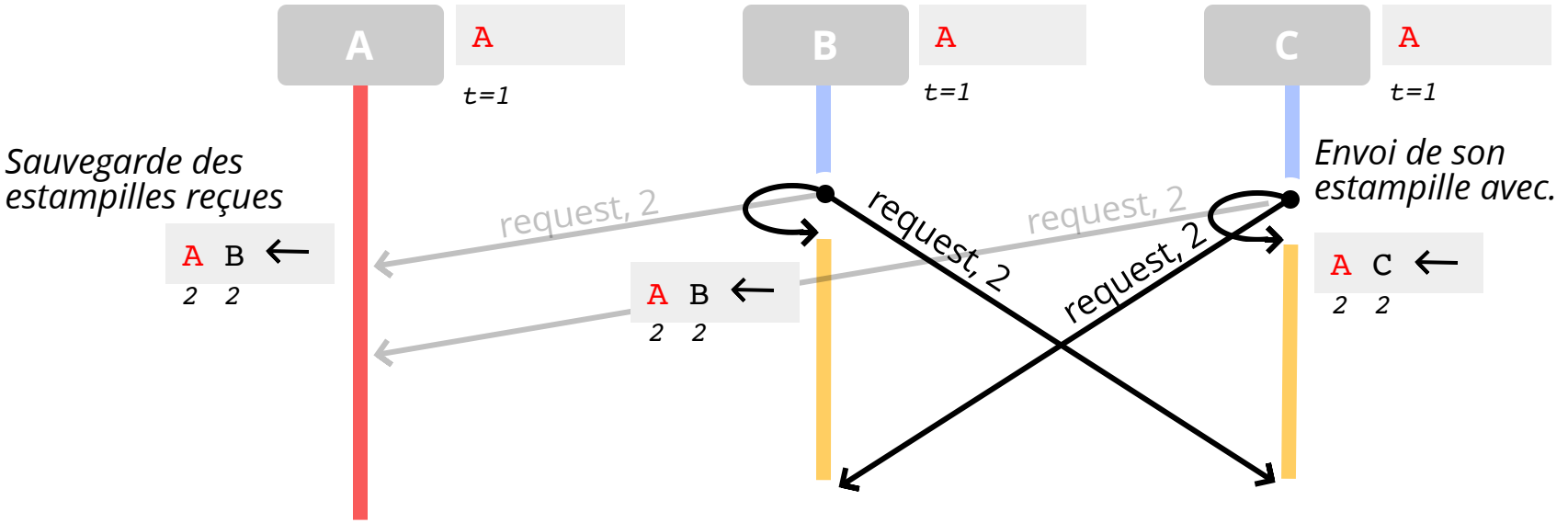
Solution répartie

Version naïve *corrigée*



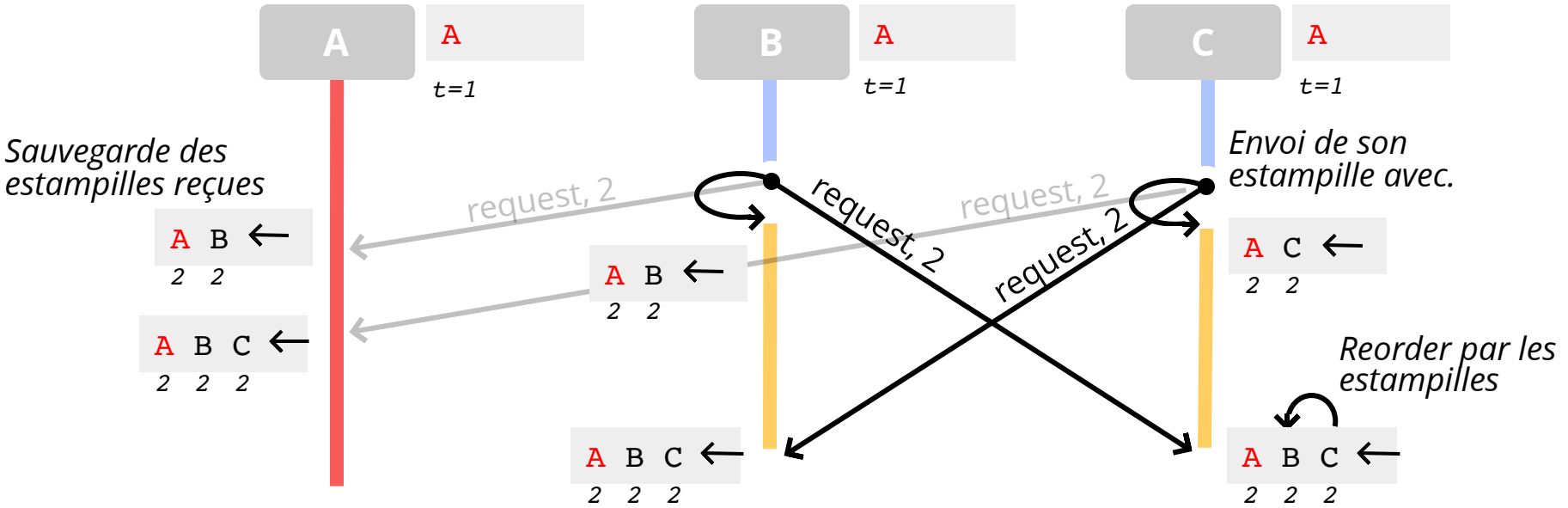
Solution répartie

Version naïve *corrigée*



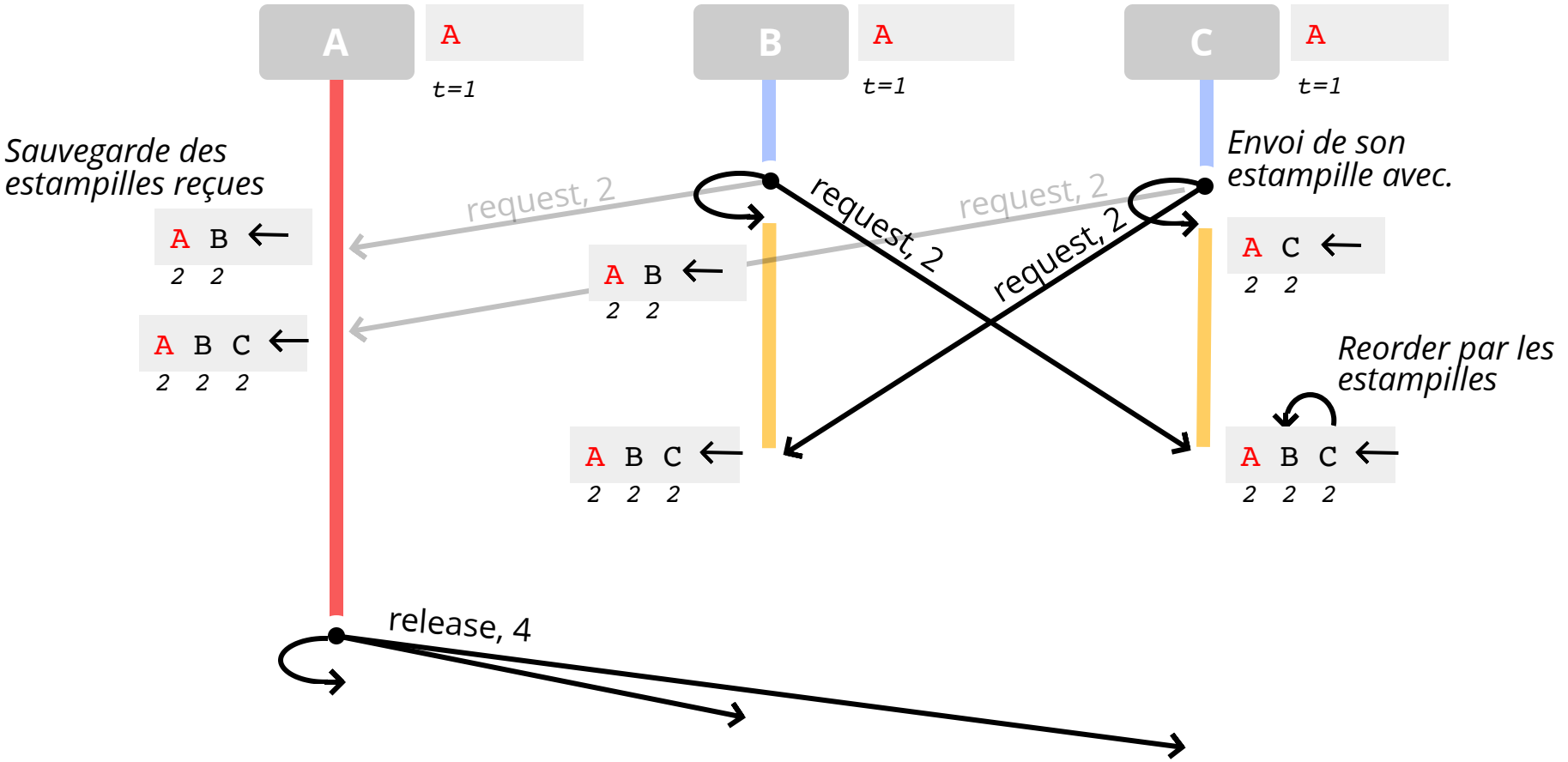
Solution répartie

Version naïve *corrigée*



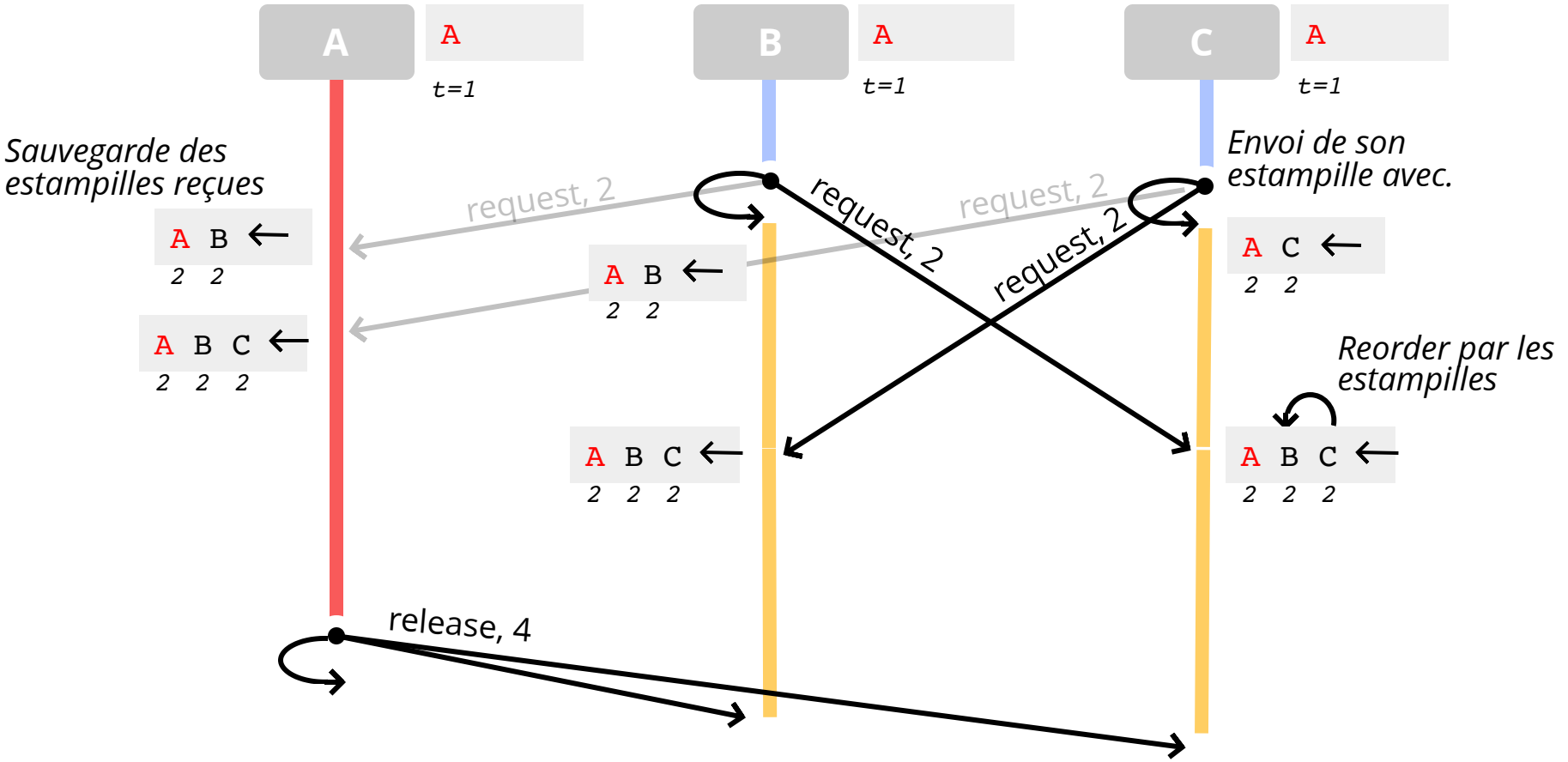
Solution répartie

Version naïve *corrigée*



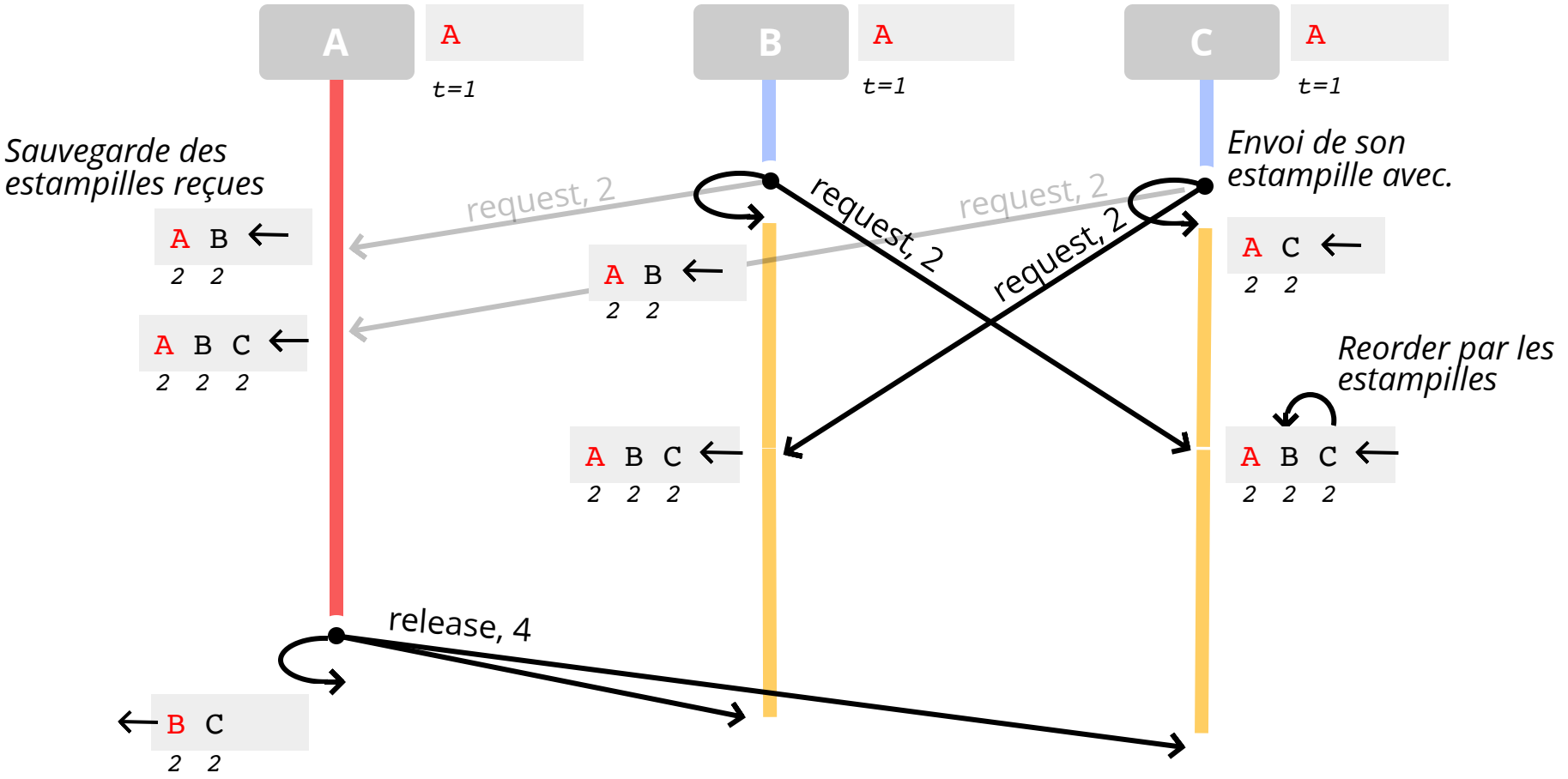
Solution répartie

Version naïve *corrigée*



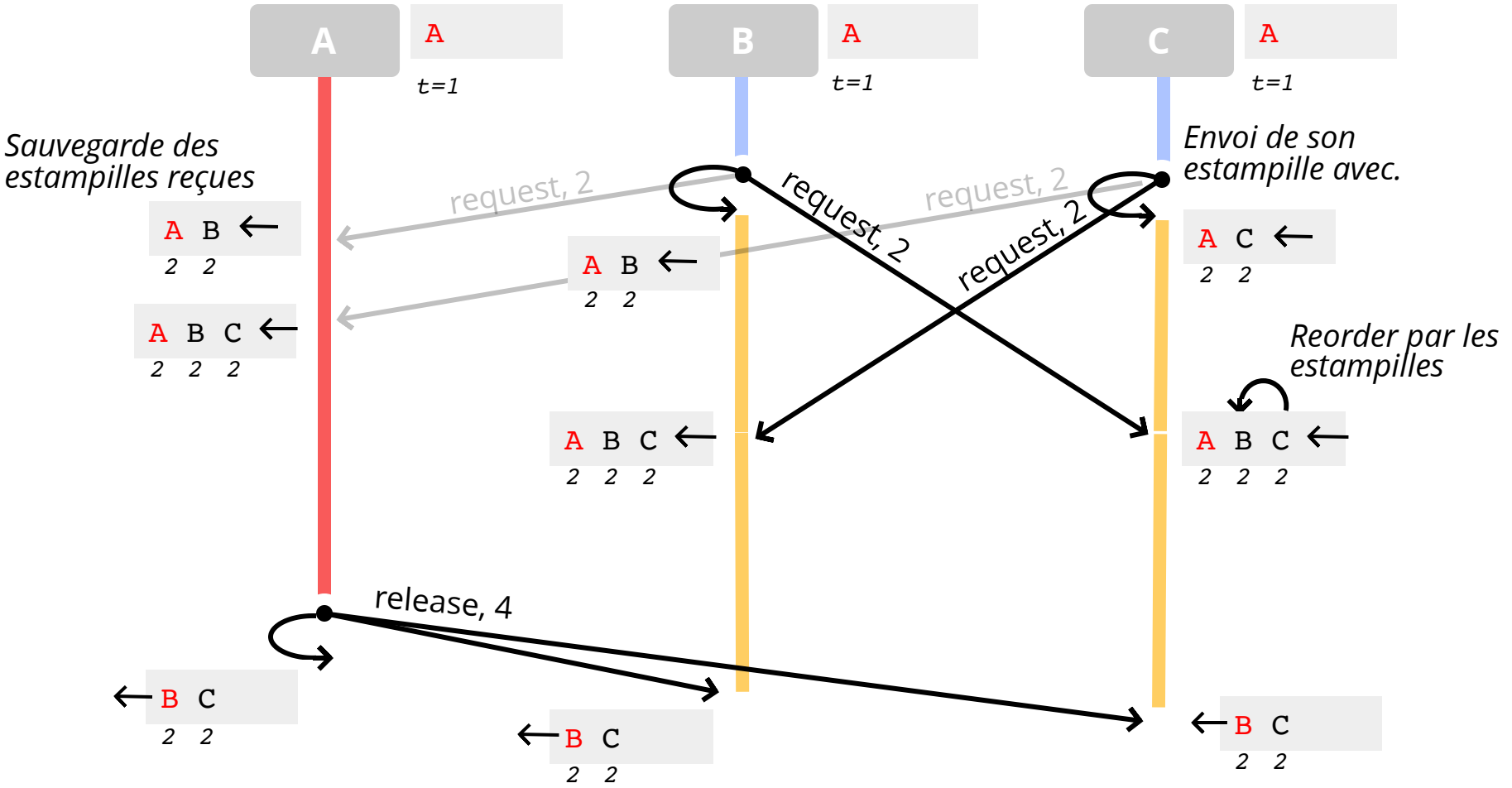
Solution répartie

Version naïve *corrigée*



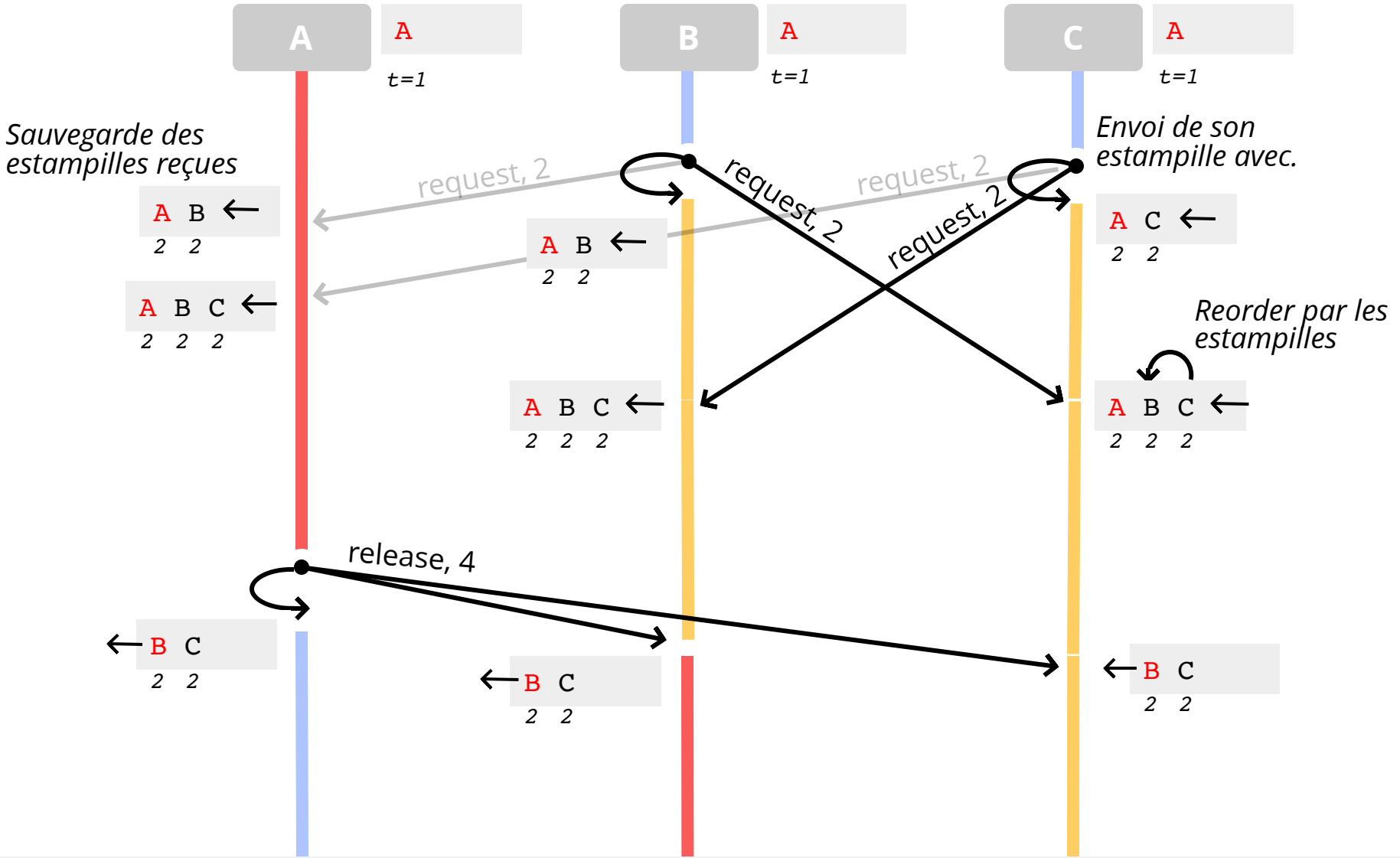
Solution répartie

Version naïve *corrigée*



Solution répartie

Version naïve *corrigée*



Solution répartie
Version naïve corrigée - Résumé

La tête de la file est toujours
celle actuellement en SC.

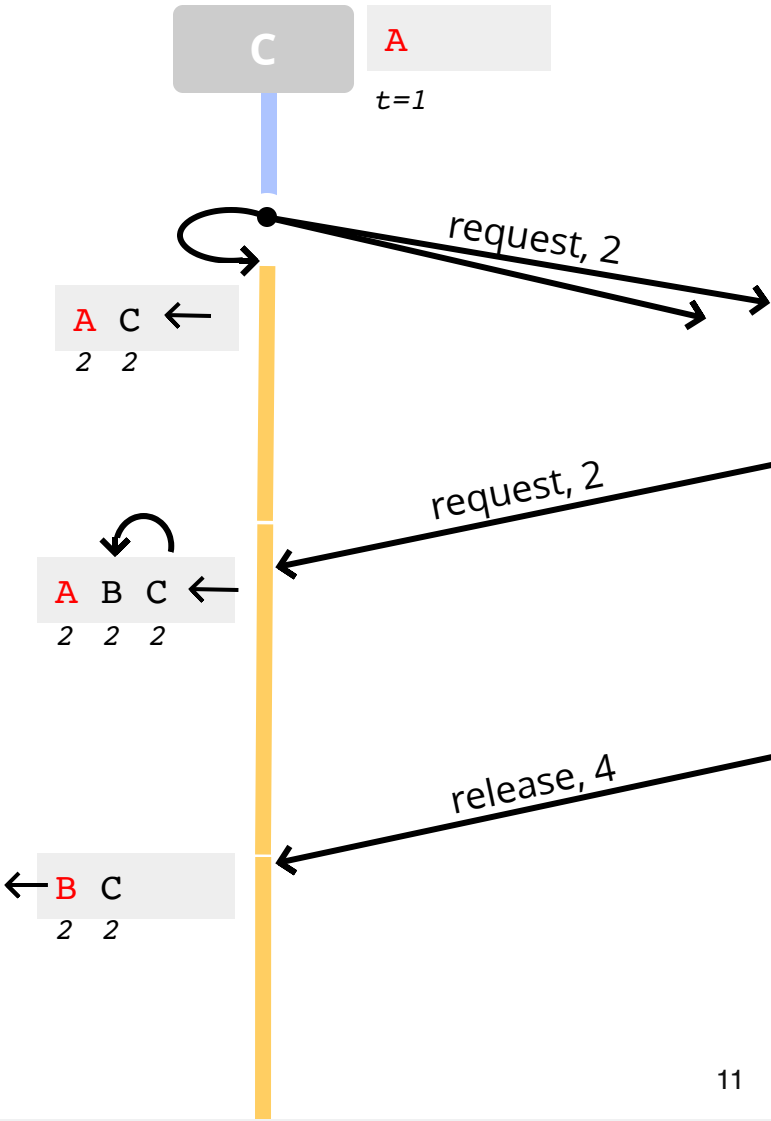
On request

Push demandeur dans la file

On release

Pop tête de file

Si je suis nouvelle tête, entrer en SC.



Solution répartie

Version naïve corrigée - Résumé

La tête de la file est toujours
celle actuellement en SC.

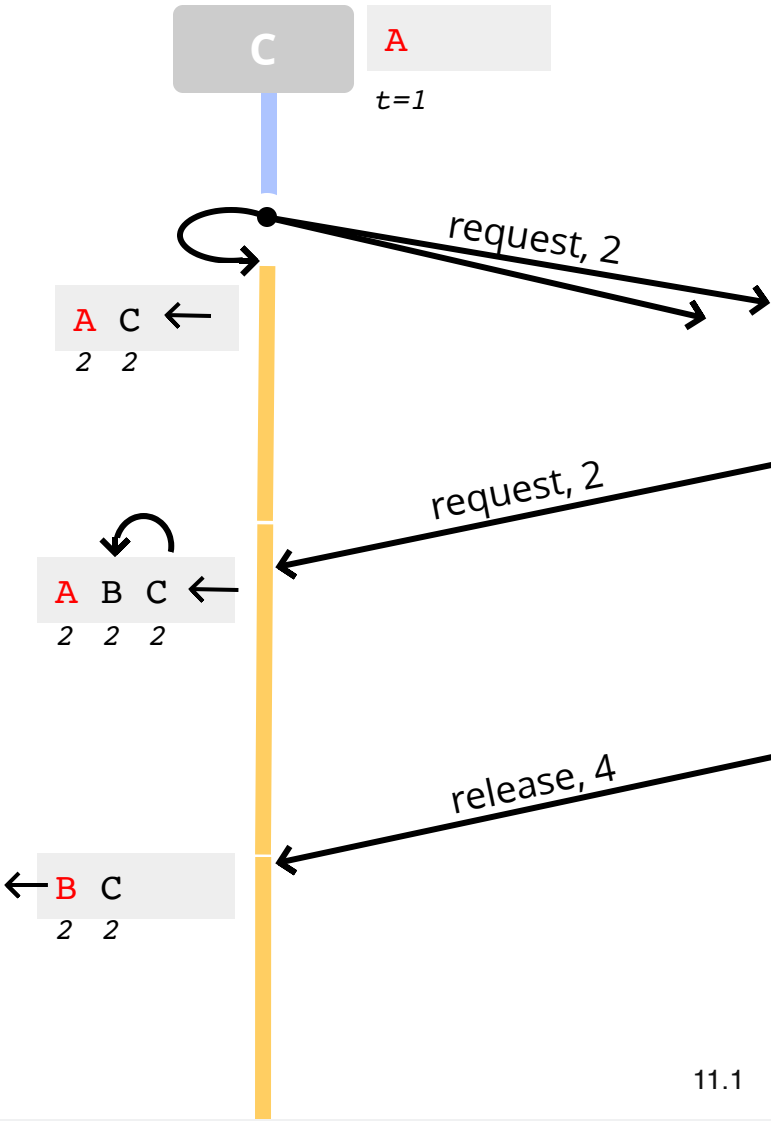
On request

Push demandeur dans la file
avec son timestamp.

On release

Pop tête de file

Si je suis nouvelle tête, entrer en SC.



Solution répartie

Version naïve corrigée - Résumé

La tête de la file est toujours
celle actuellement en SC.

On request

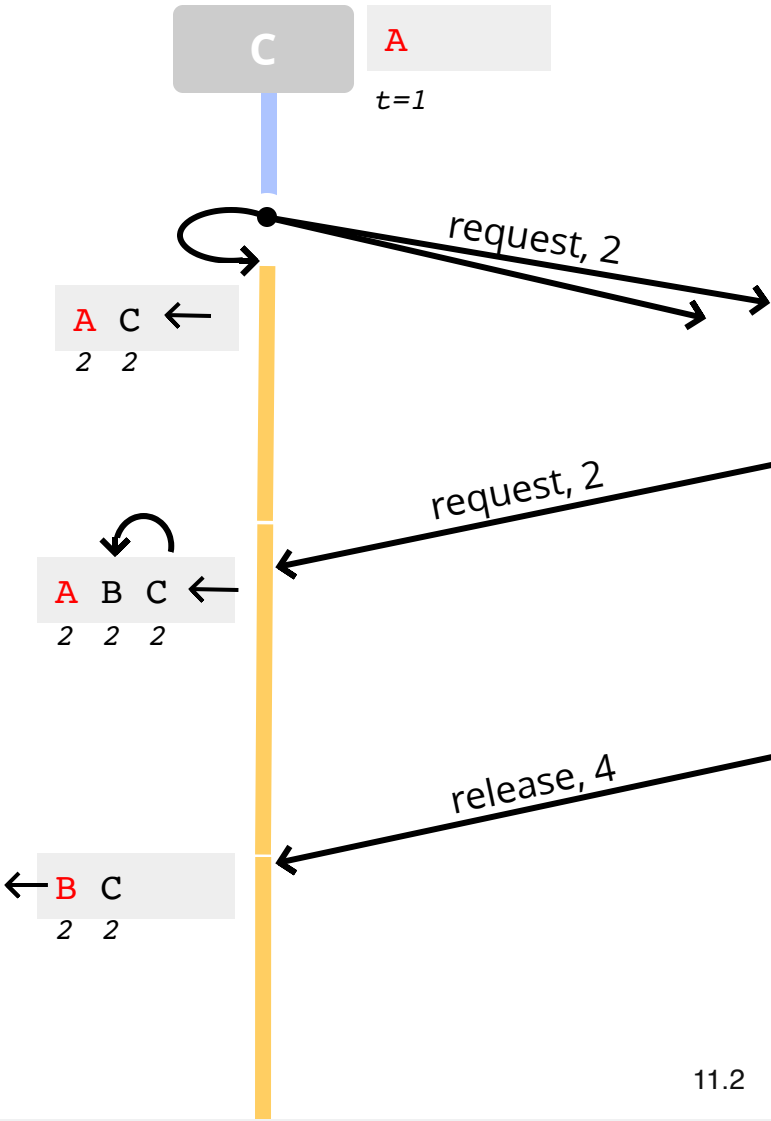
Push demandeur dans la file
avec son timestamp.

Trie la file par heure de Lamport.
(avec résolution arbitraire des égalités)

On release

Pop tête de file

Si je suis nouvelle tête, entrer en SC.



Solution répartie
Version naïve corrigée - Résumé

La tête de la file est toujours
celle actuellement en SC.

On request

Push demandeur dans la file
avec son timestamp.

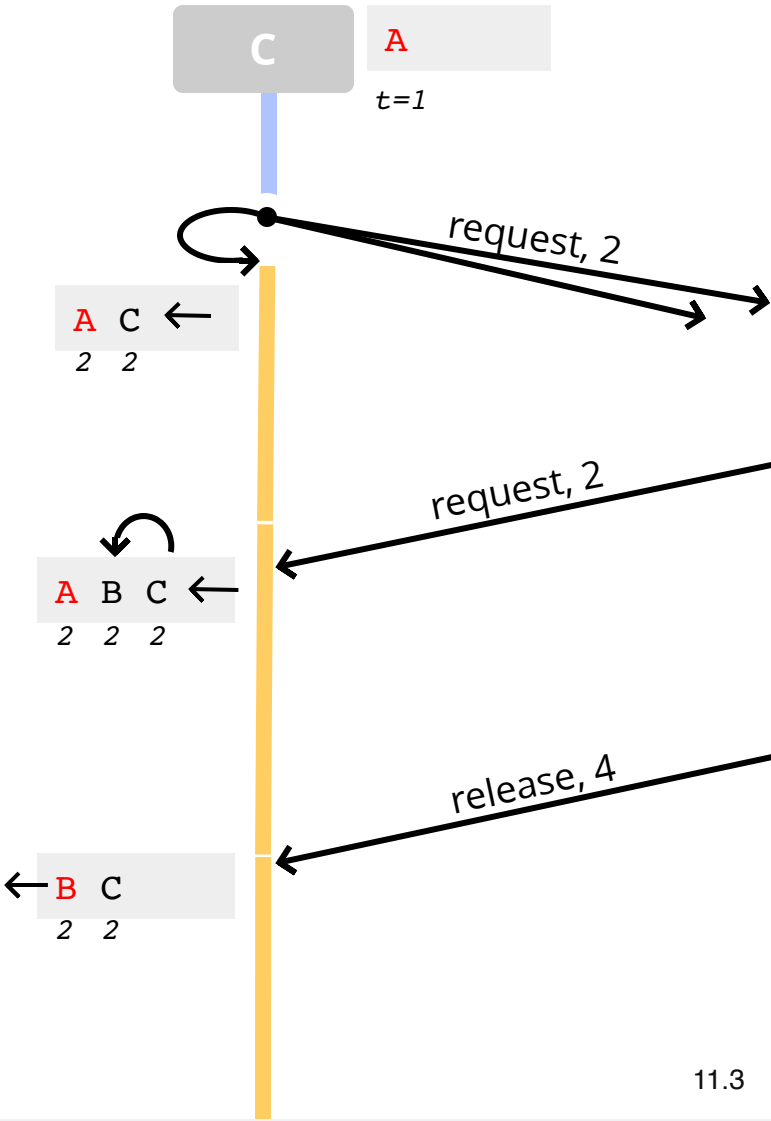
Trie la file par heure de Lamport.
(avec résolution arbitraire des égalités)

On release

Pop tête de file

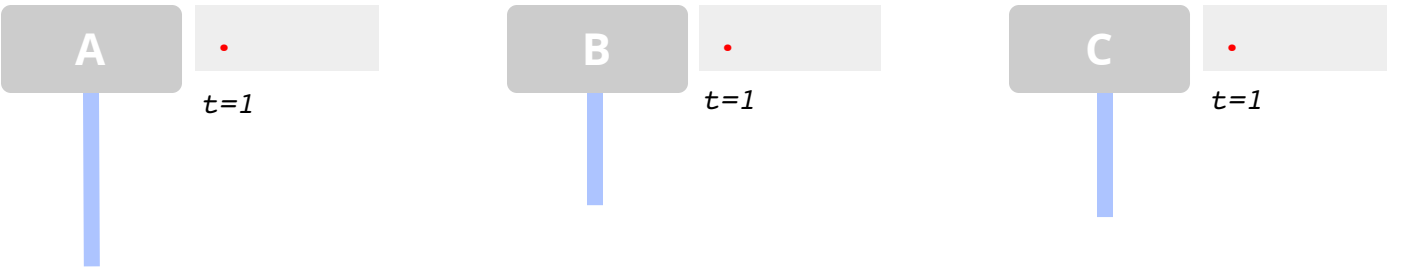
Si je suis nouvelle tête, entrer en SC.

Problème ?



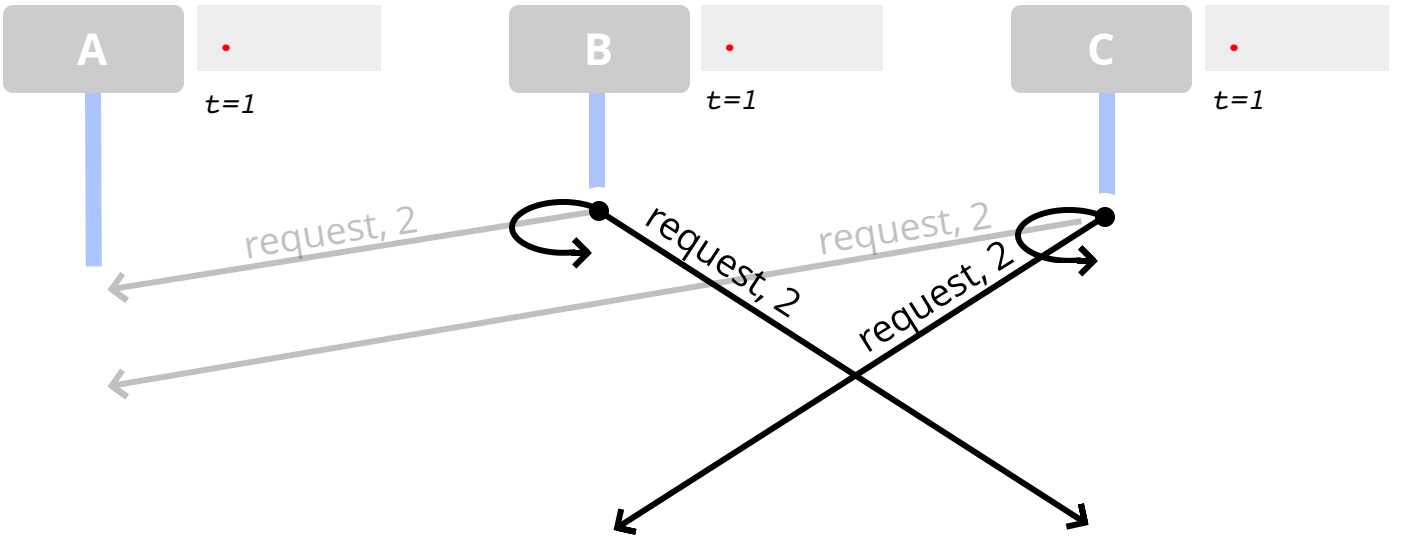
Solution répartie

Version naïve corrigée - Problème



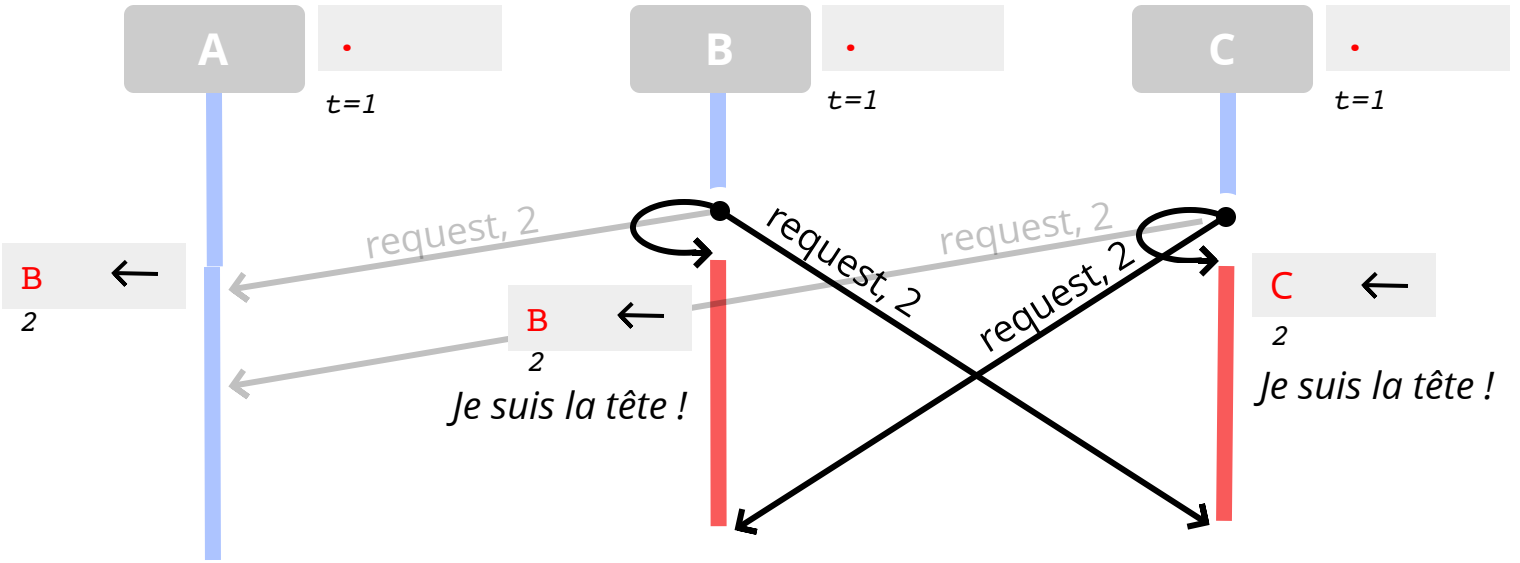
Solution répartie

Version naïve corrigée - Problème



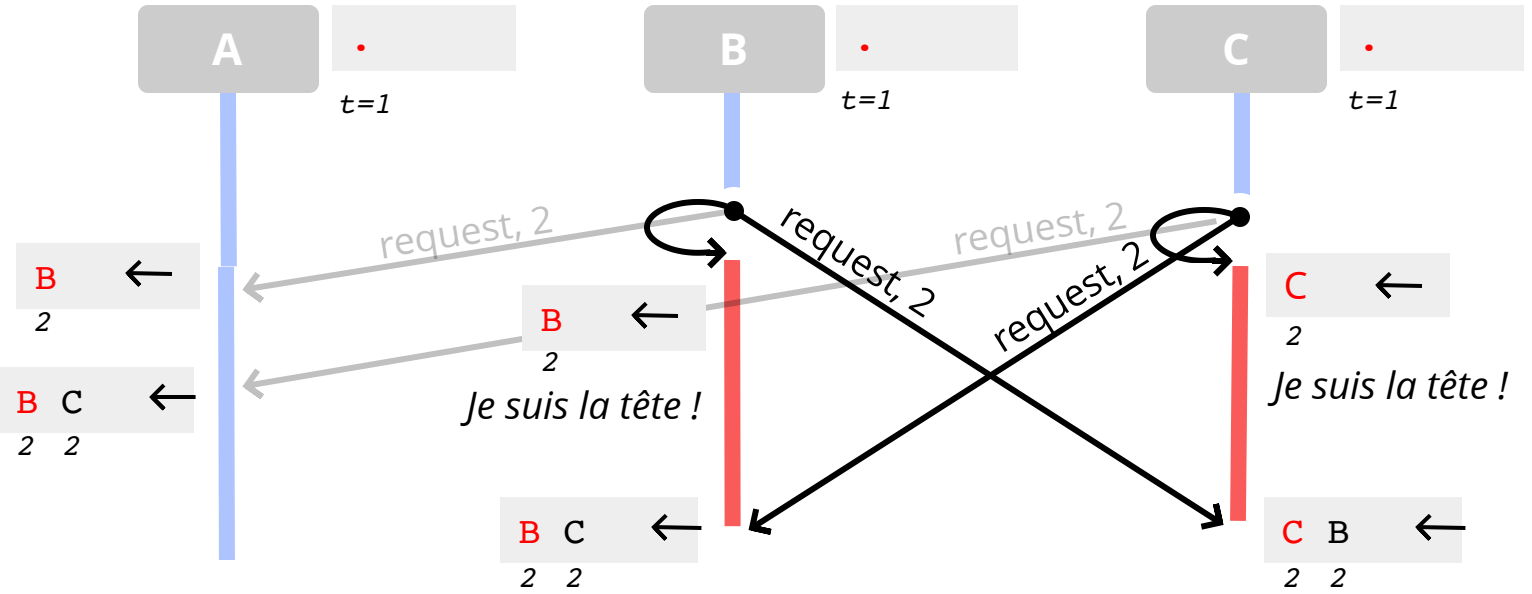
Solution répartie

Version naïve corrigée - Problème



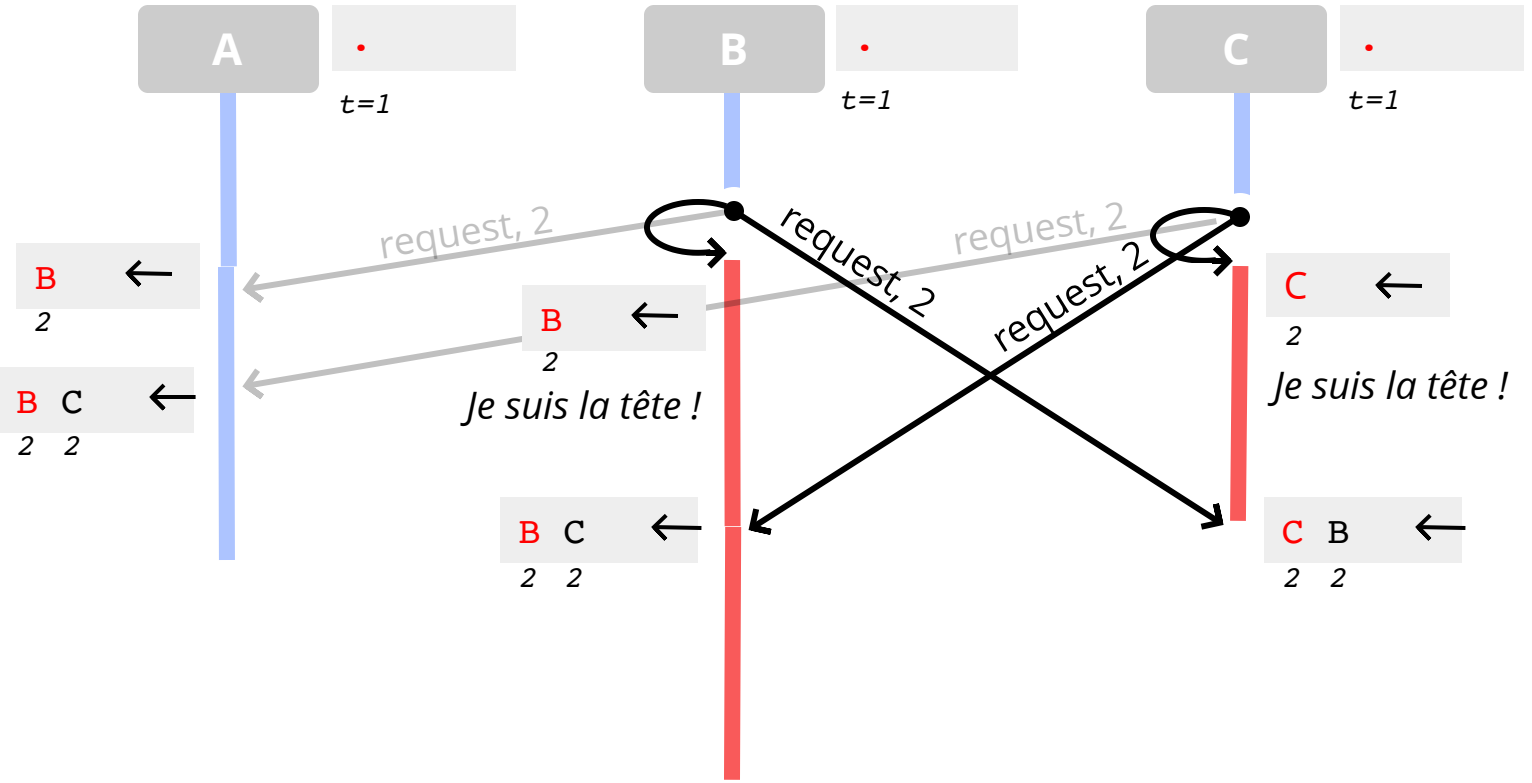
Solution répartie

Version naïve corrigée - Problème



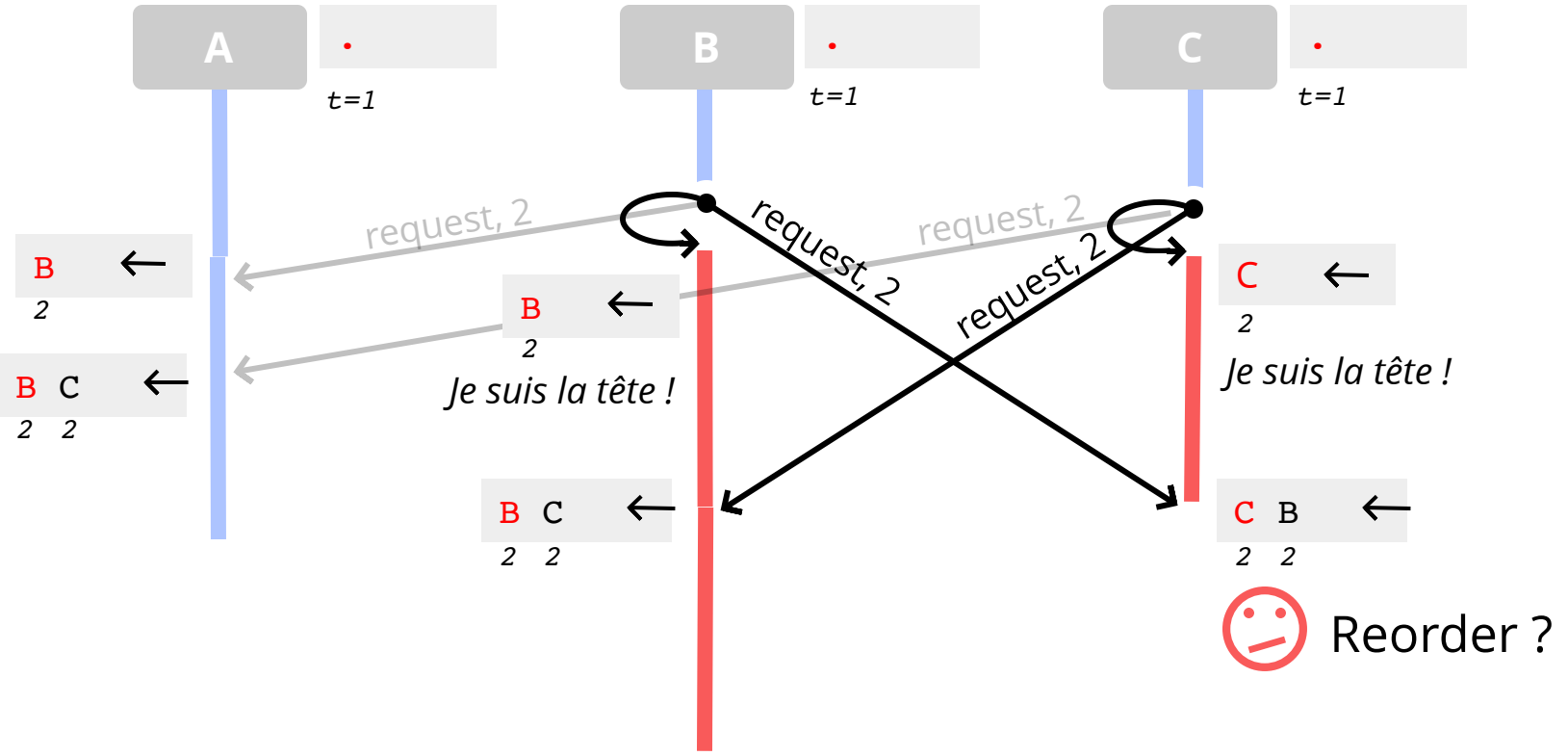
Solution répartie

Version naïve corrigée - Problème



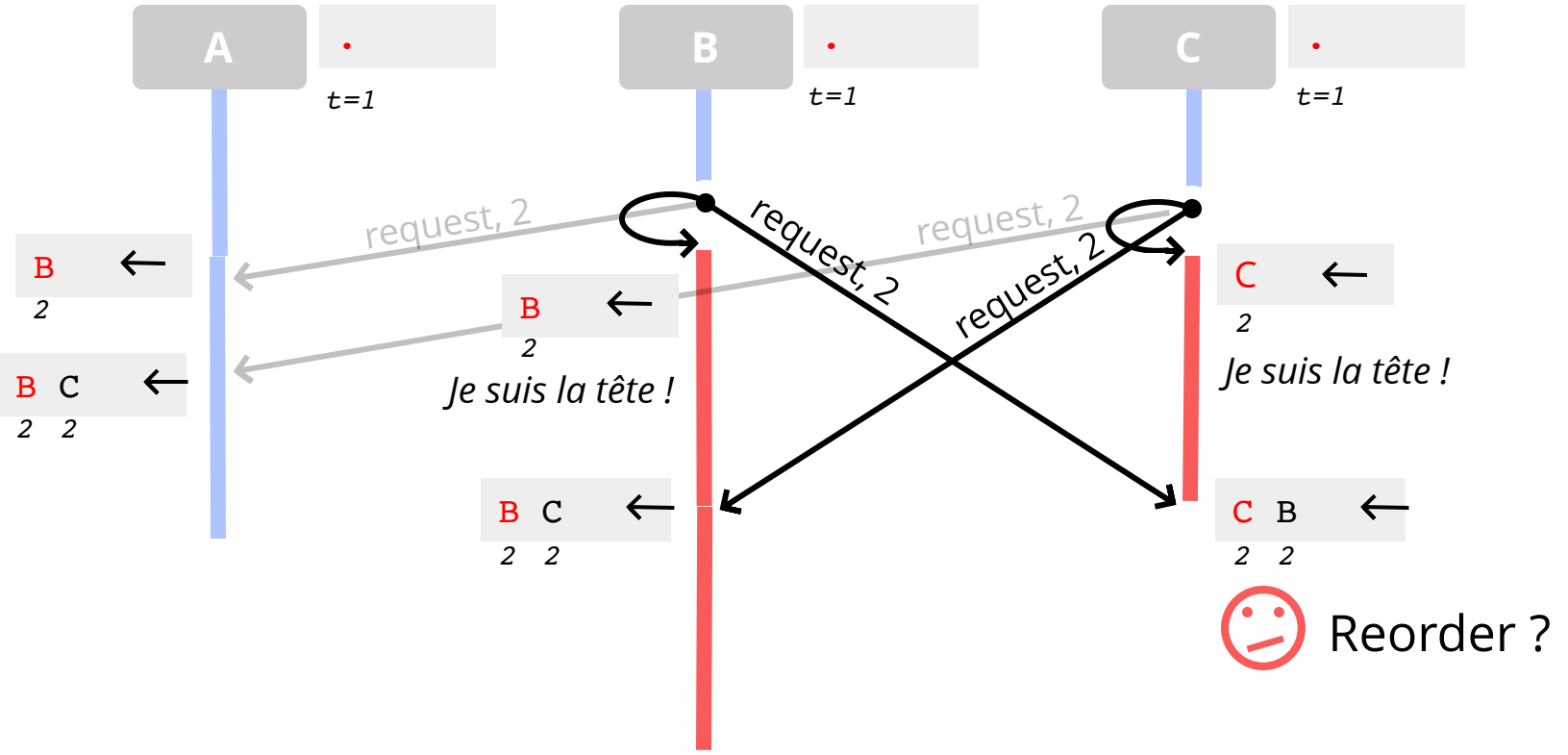
Solution répartie

Version naïve corrigée - Problème



Solution répartie

Version naïve corrigée - Problème



Problème :

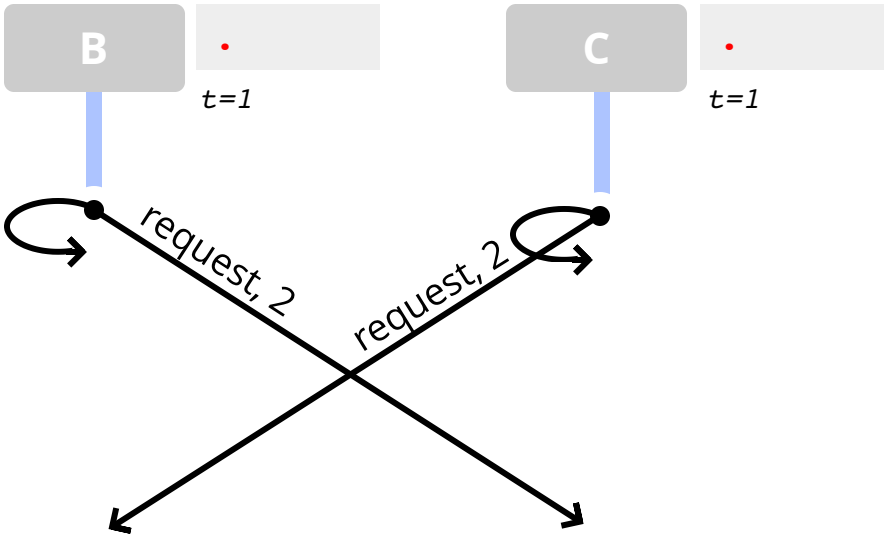
Les timestamps de Lamport n'ordonnancent pas les messages immédiatement.
Ils assurent seulement qu'un ordre strict sera obtenu **un jour**.

Solution répartie

Version naïve corrigée *corrigée*

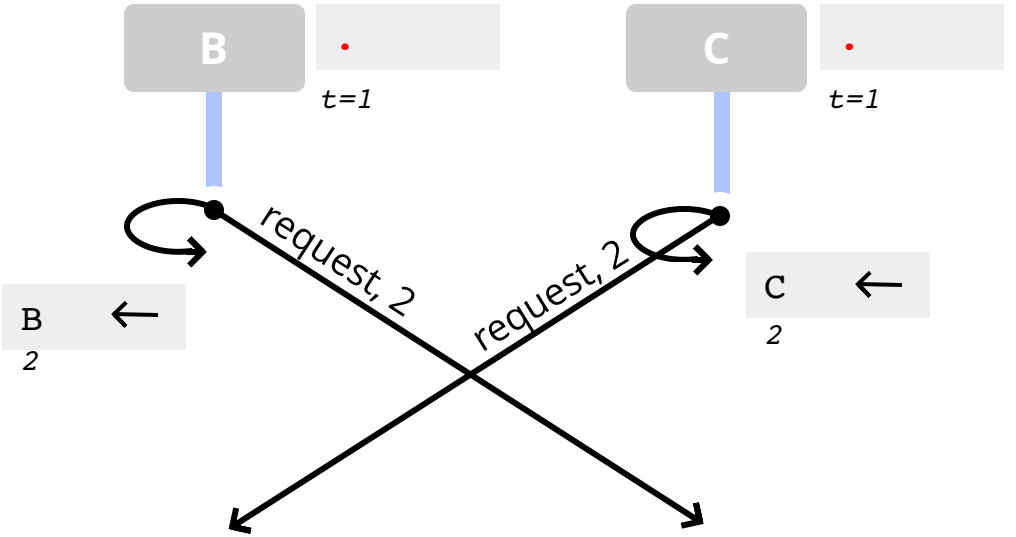
Solution répartie

Version naïve corrigée *corrigée*



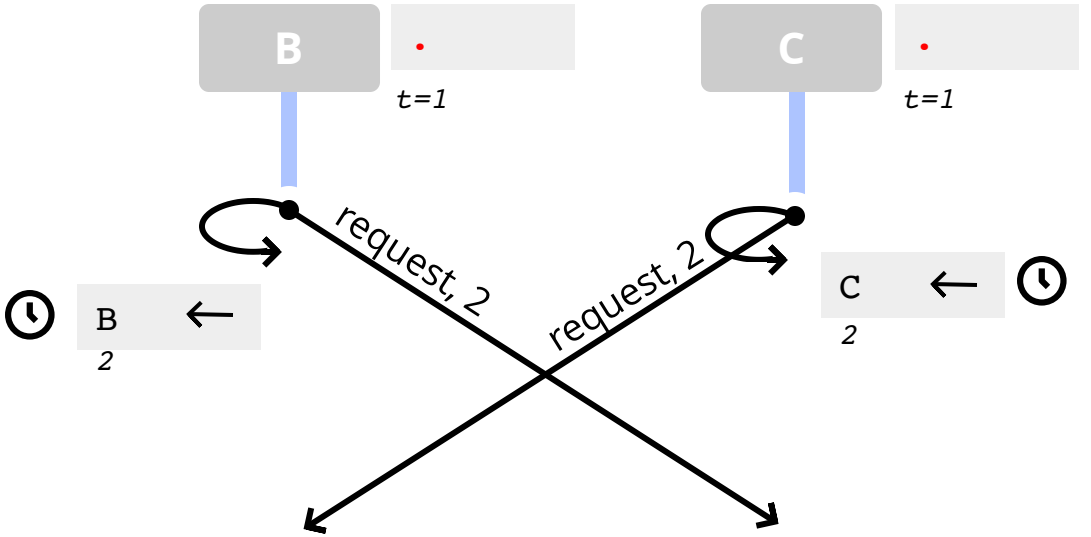
Solution répartie

Version naïve corrigée *corrigée*



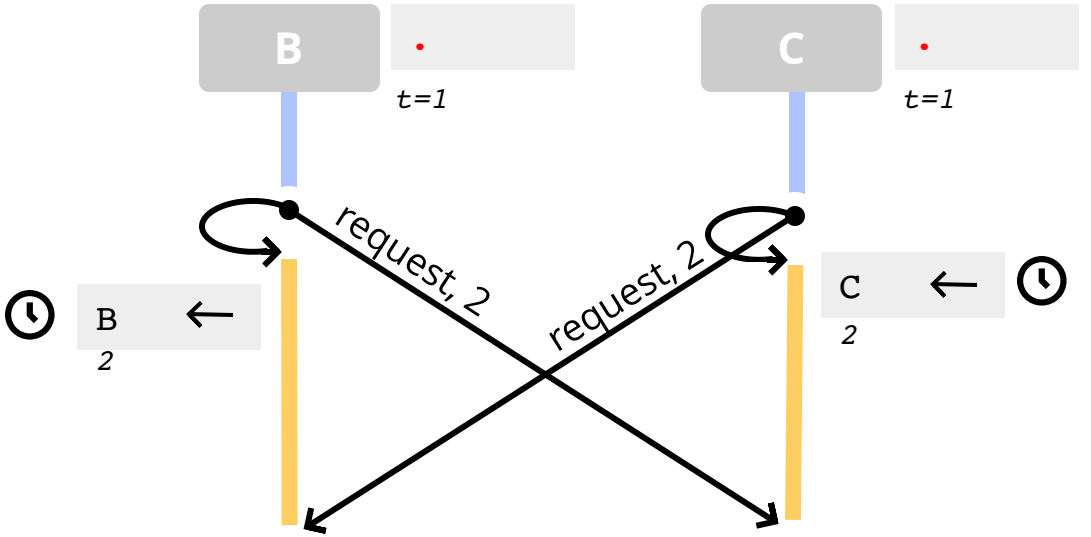
Solution répartie

Version naïve corrigée *corrigée*



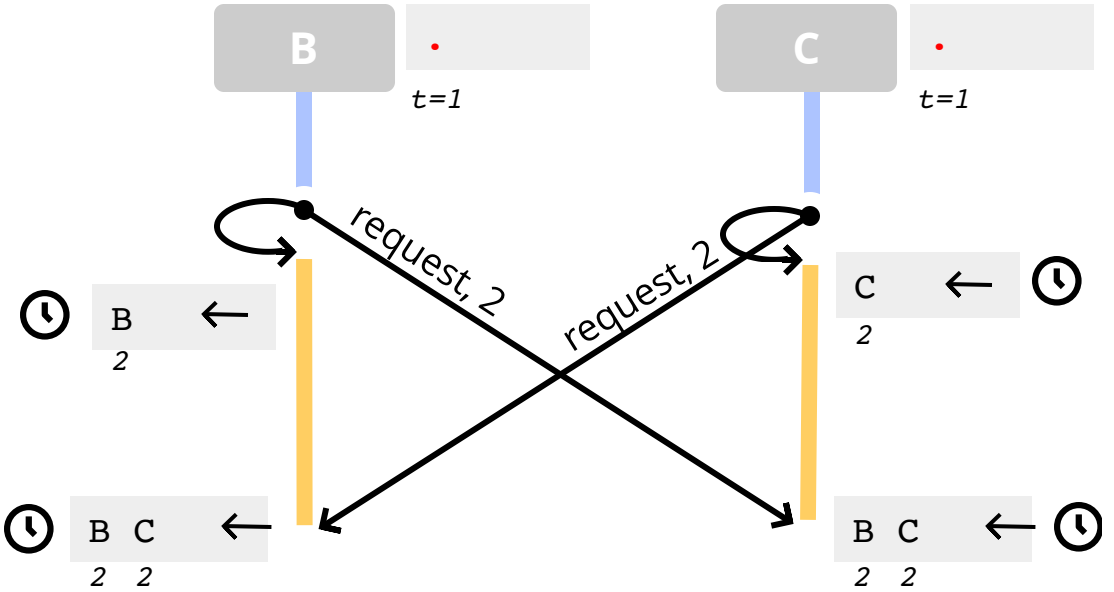
Solution répartie

Version naïve corrigée *corrigée*



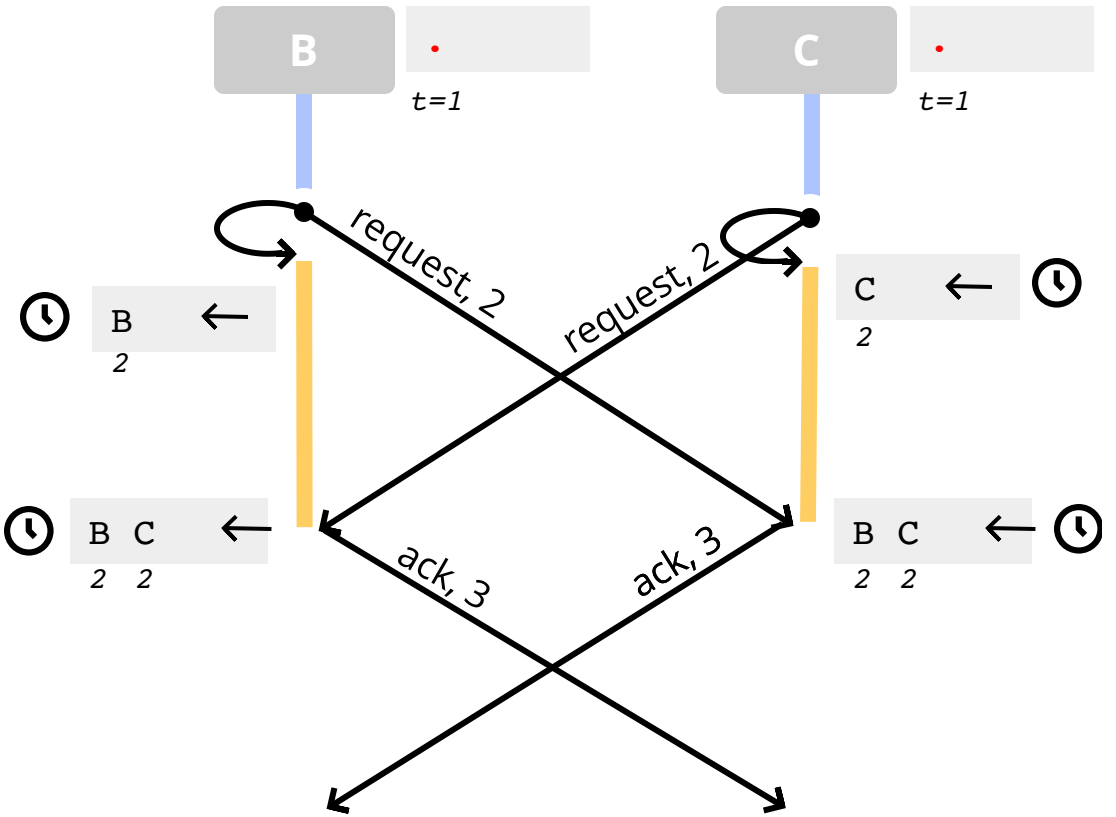
Solution répartie

Version naïve corrigée *corrigée*



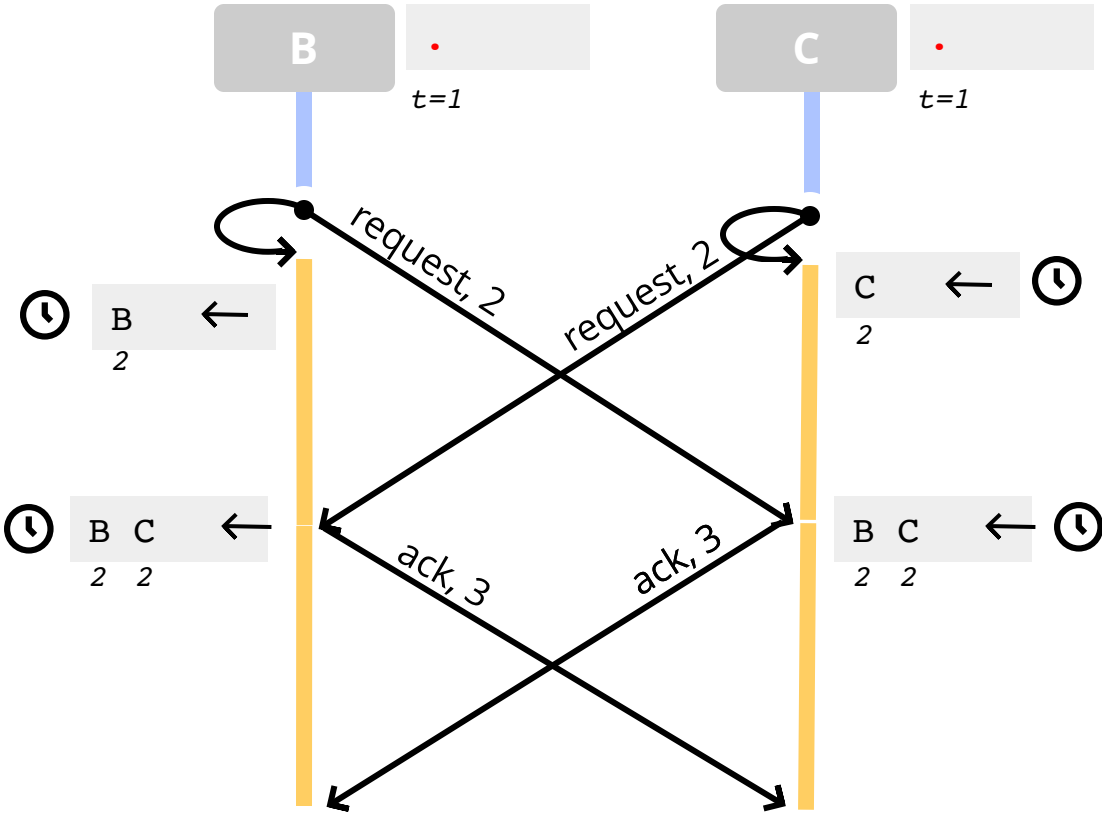
Solution répartie

Version naïve corrigée *corrigée*



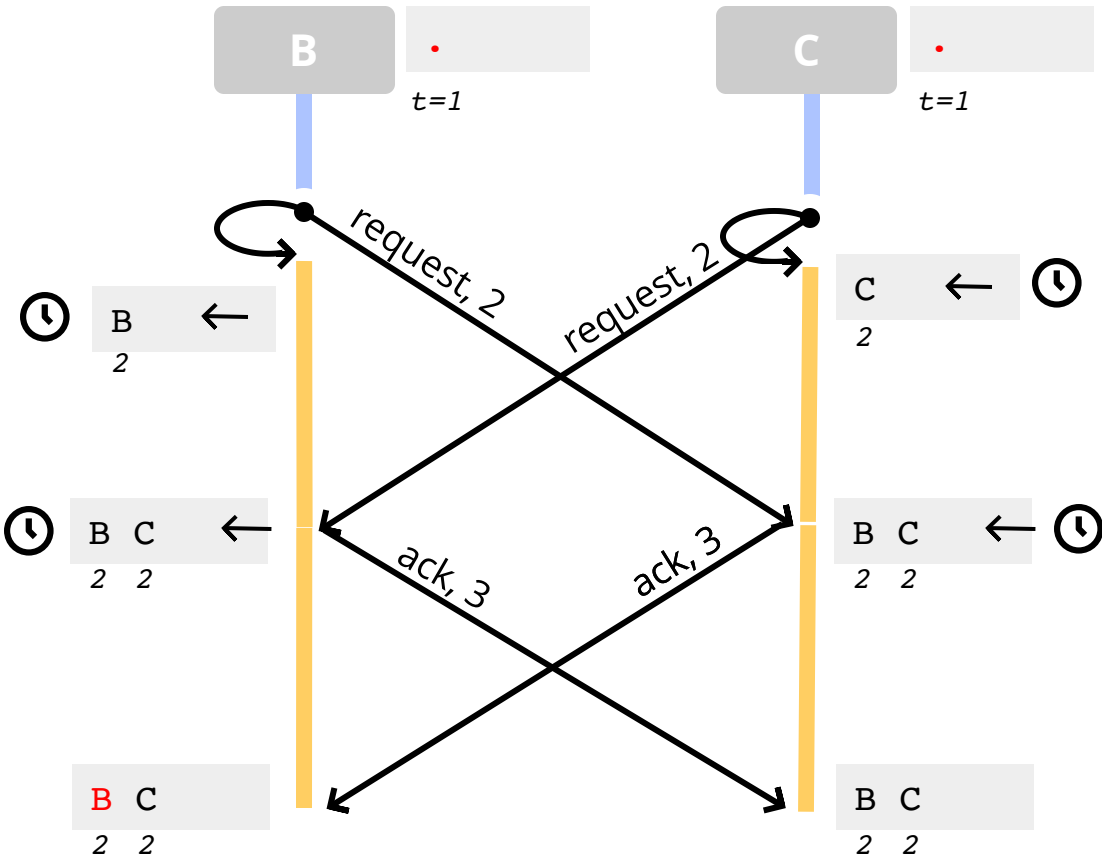
Solution répartie

Version naïve corrigée *corrigée*



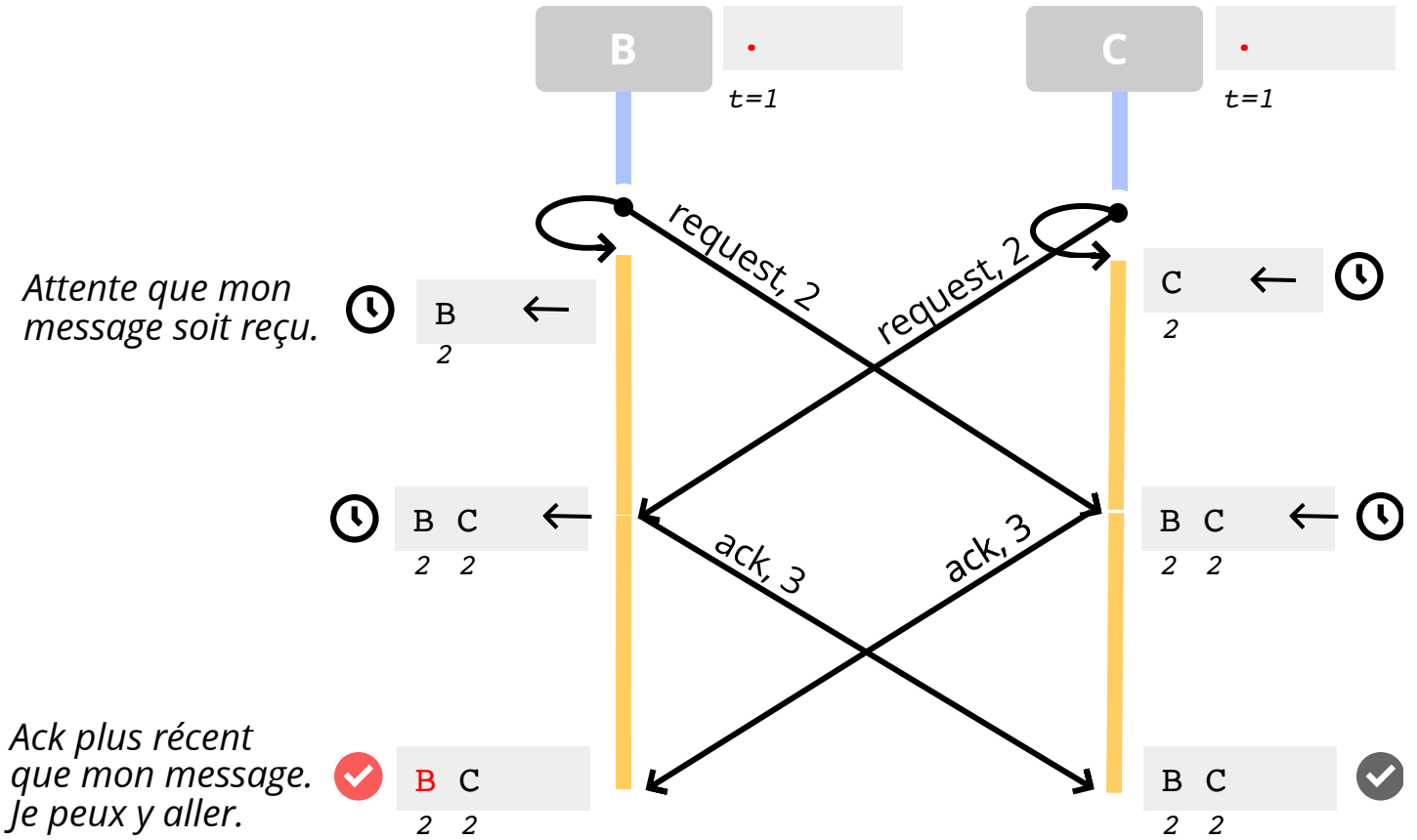
Solution répartie

Version naïve corrigée *corrigée*



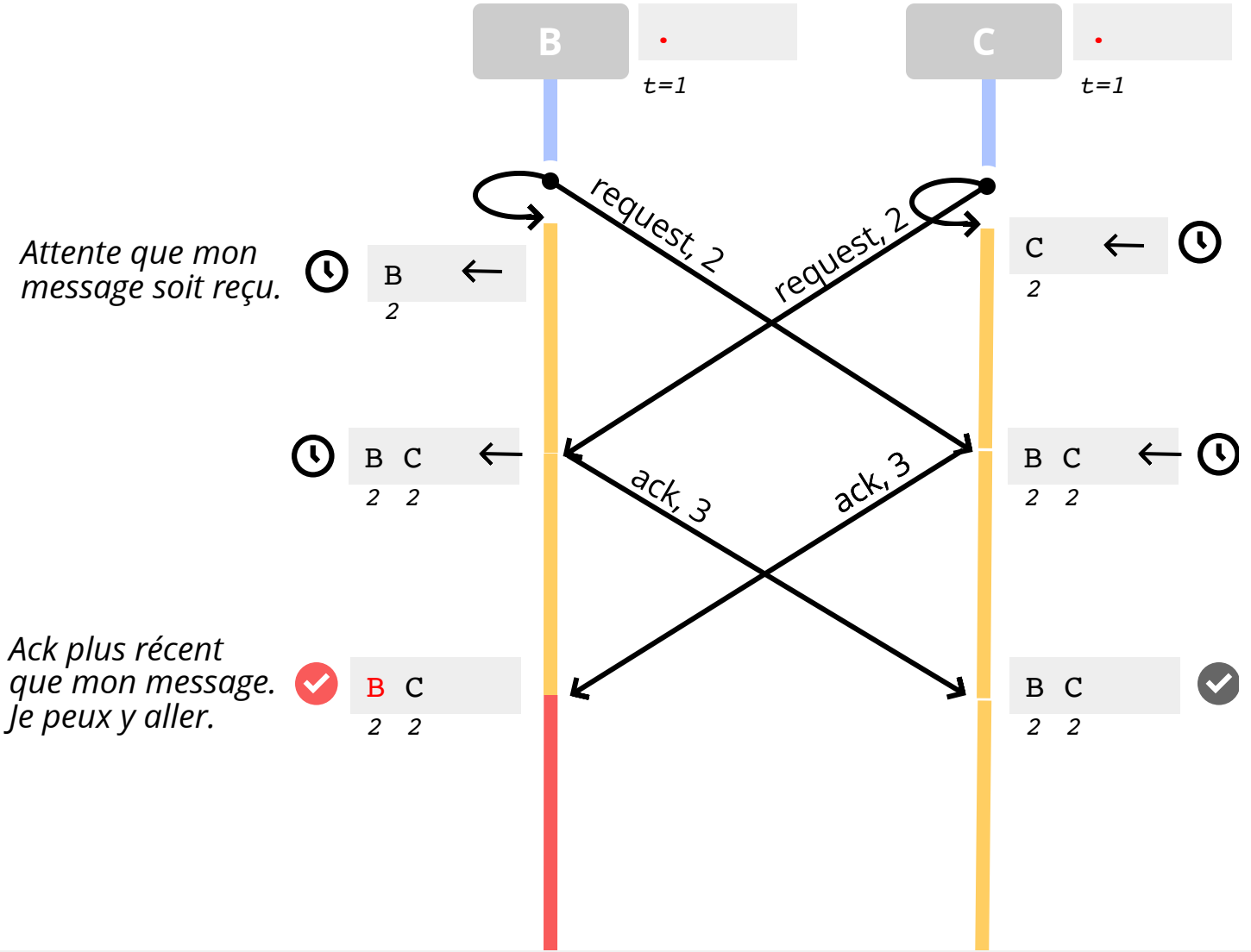
Solution répartie

Version naïve corrigée *corrigée*



Solution répartie

Version naïve corrigée *corrigée*



Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

On release

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

On release

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

On release

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Pop tête de file

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Pop tête de file

Entrer en SC

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Pop tête de file

Entrer en SC
si je suis la nouvelle tête, et que j'ai reçu tous les `acks`.

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Pop tête de file

Entrer en SC
si je suis la nouvelle tête, et que j'ai reçu tous les `acks`.

La tête de la file est celle actuellement en SC,

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Pop tête de file

Entrer en SC
si je suis la nouvelle tête, et que j'ai reçu tous les `acks`.

La tête de la file est celle actuellement en SC,
si elle a reçu un ack de toutes les autres machines.

Solution répartie
Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Pop tête de file

Entrer en SC
si je suis la nouvelle tête, et que j'ai reçu tous les `acks`.

La tête de la file est celle actuellement en SC,
si elle a reçu un ack de toutes les autres machines.

Problème ?

Solution répartie

Version naïve corrigée *corrigée* - Résumé

On request

Push demandeur dans la file avec son timestamp.

Réordonne la file par heure de Lamport.

Envoie `ack` avec nouveau timestamp.

On release

Pop tête de file

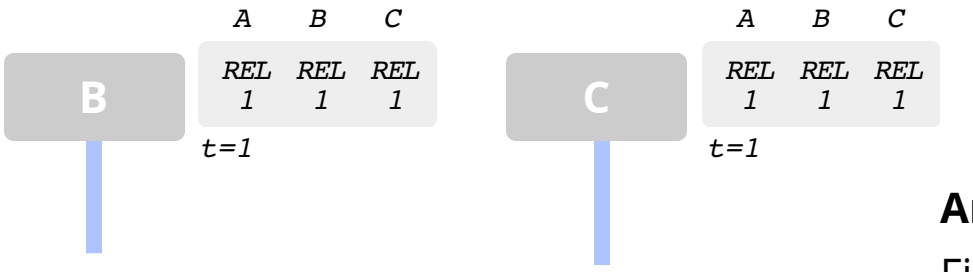
Entrer en SC
si je suis la nouvelle tête, et que j'ai reçu tous les `acks`.

La tête de la file est celle actuellement en SC,
si elle a reçu un ack de toutes les autres machines.

Problème ?
La structure de donnée n'est plus adaptée

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

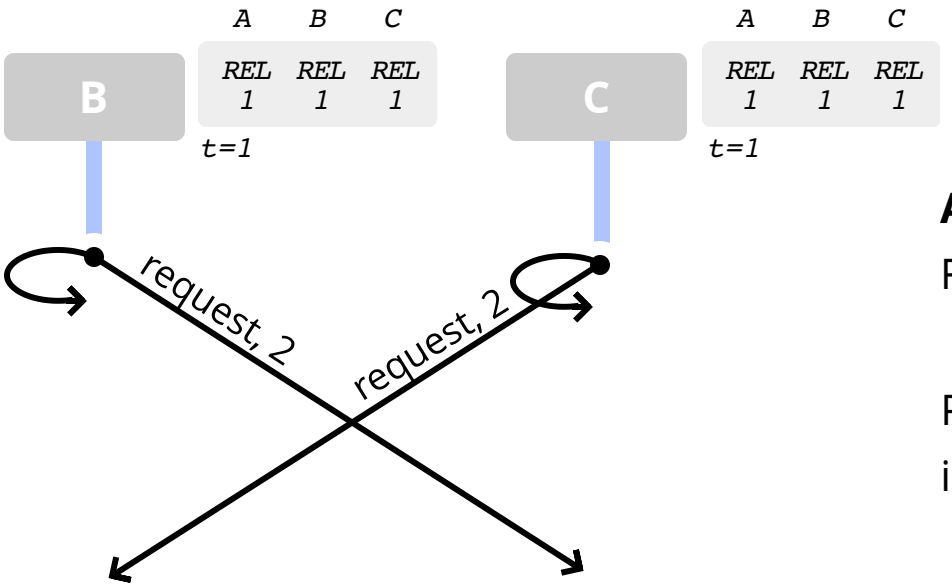
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

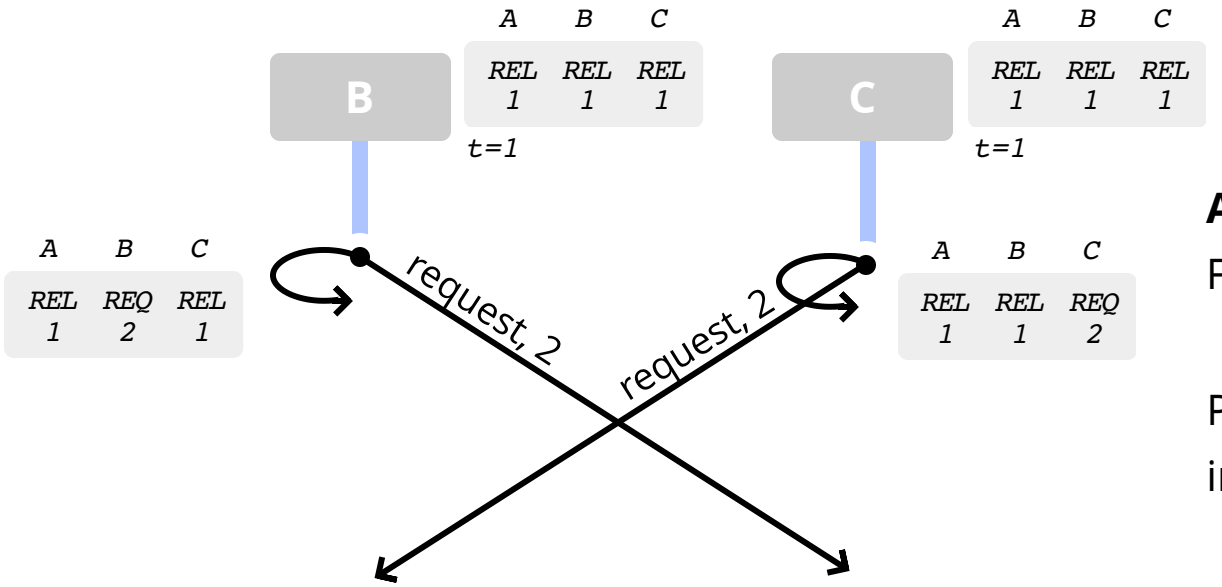
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

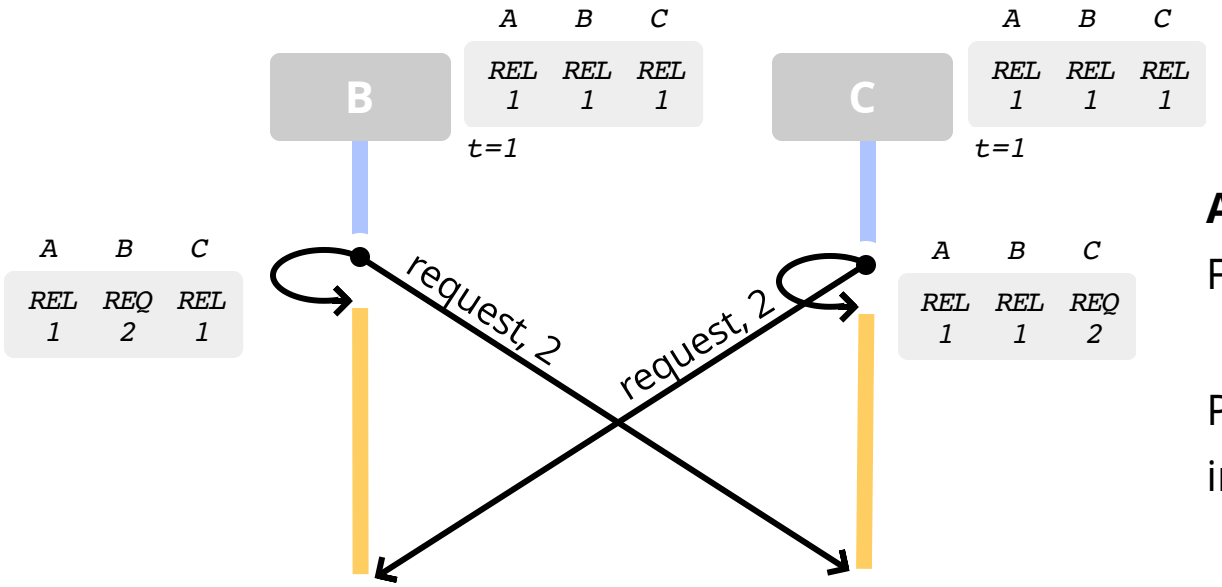
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

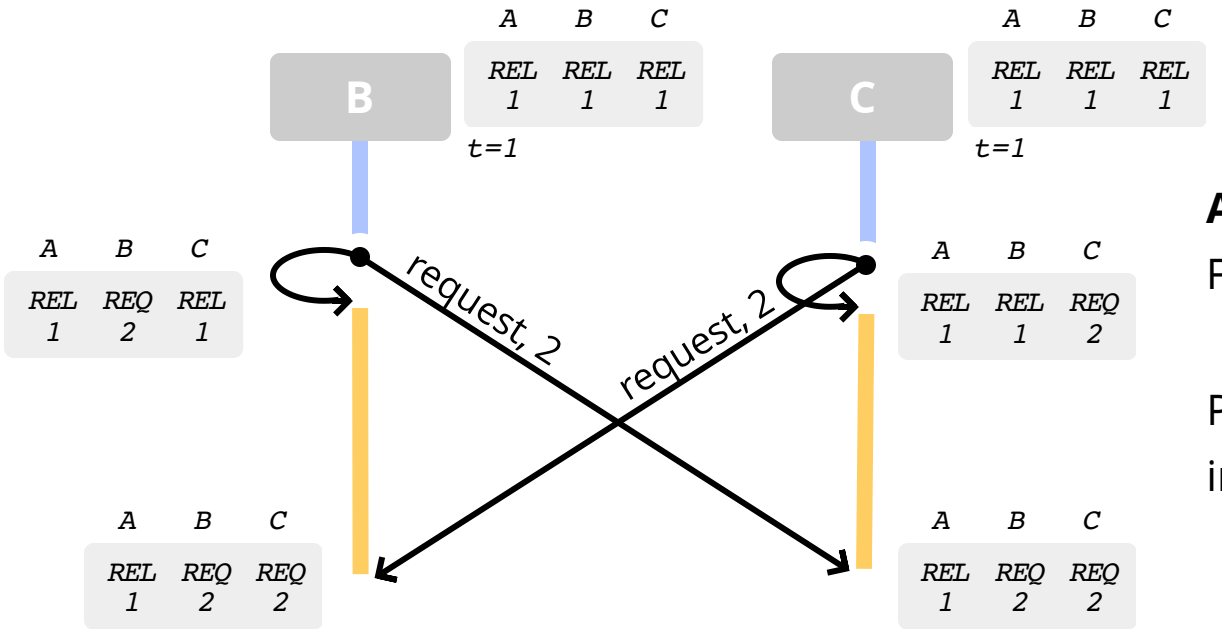
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

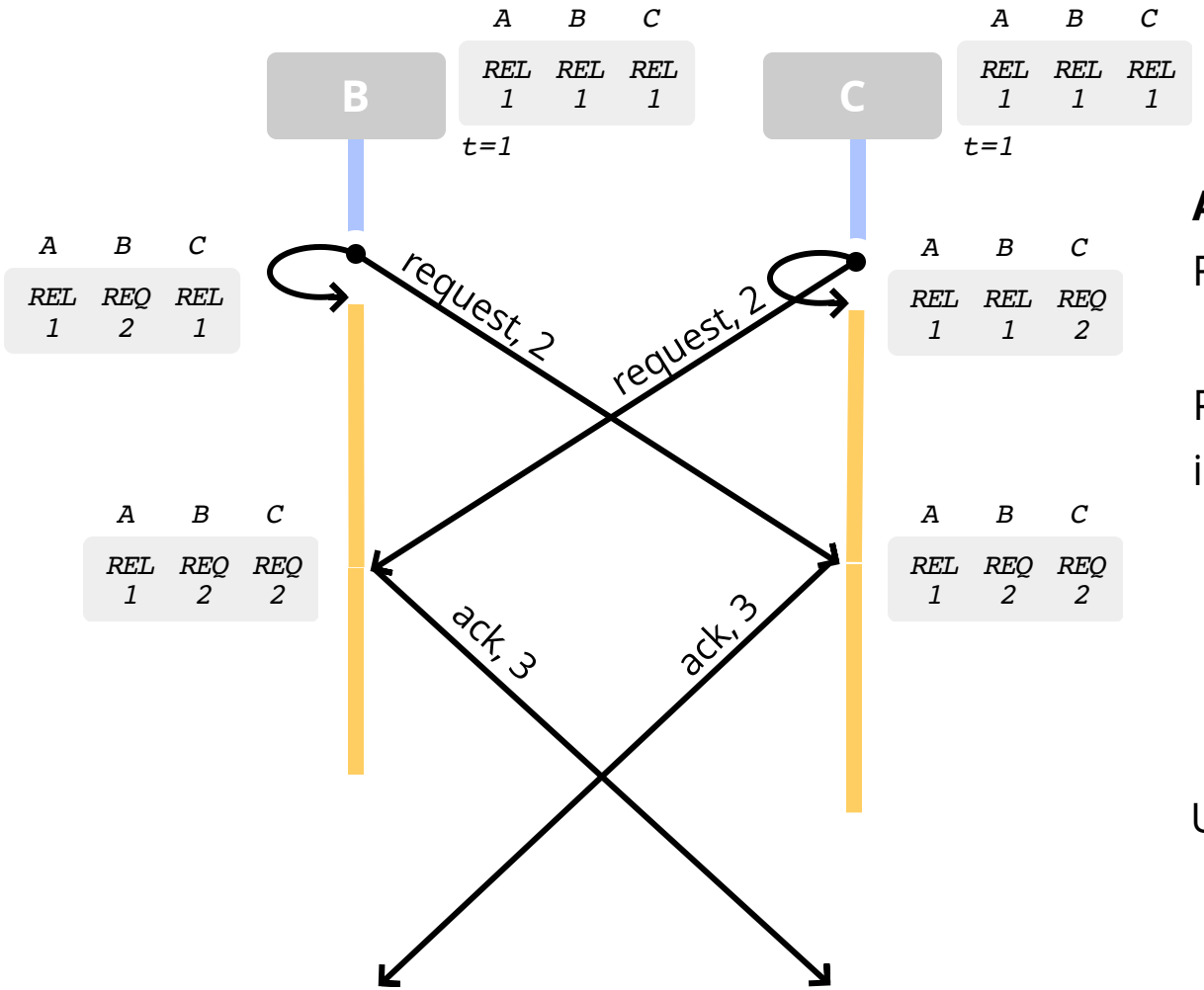
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

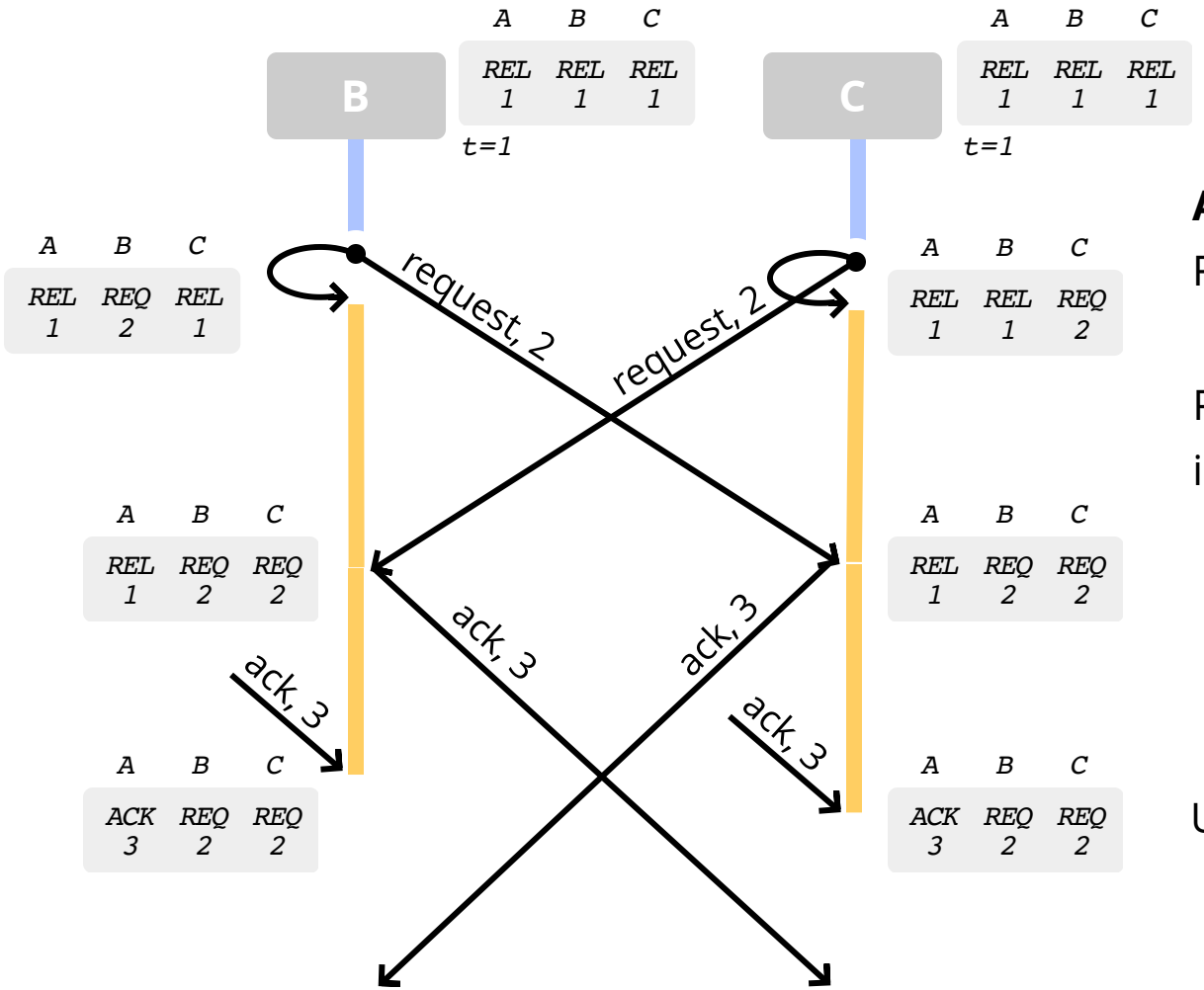
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

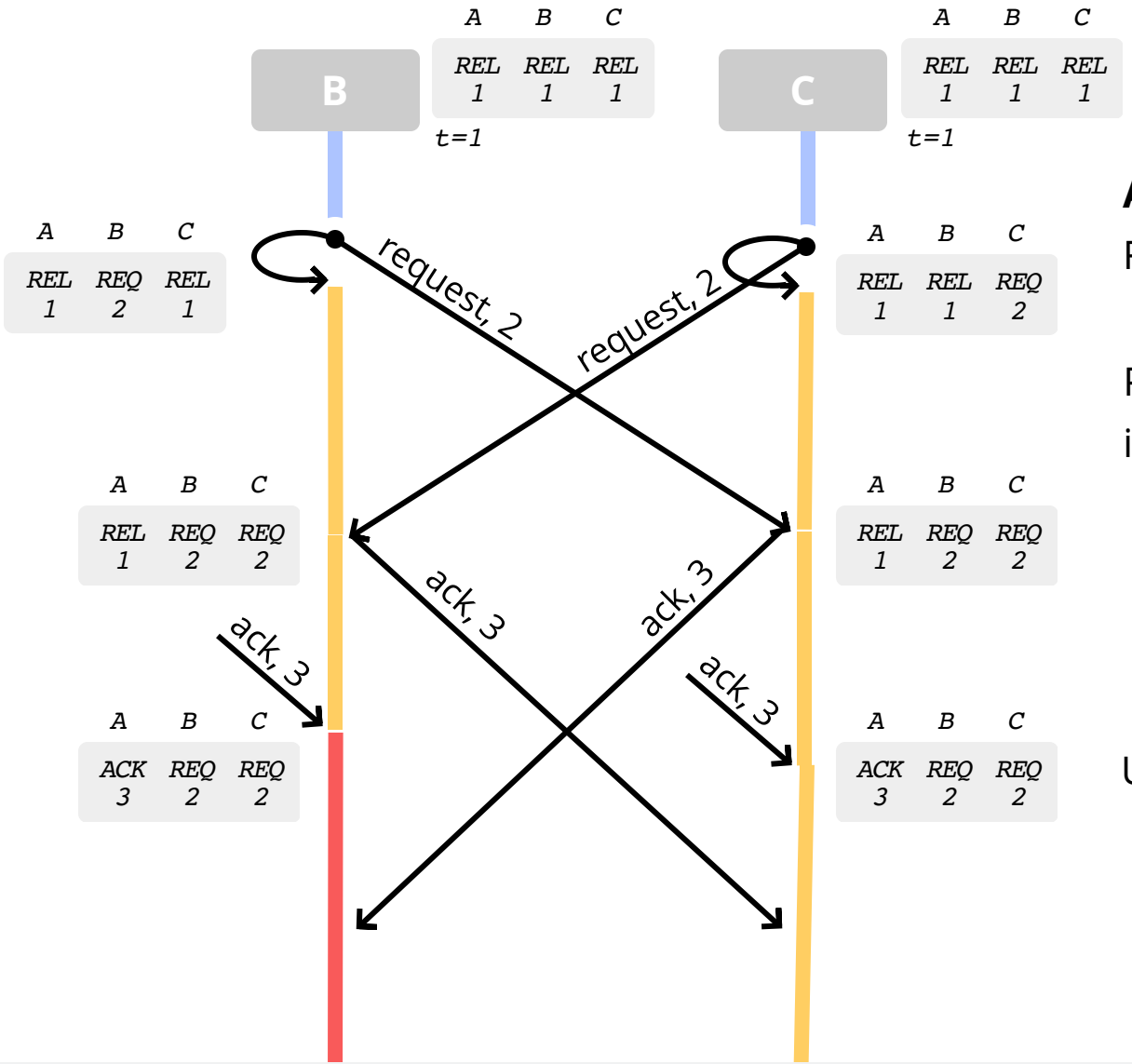
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

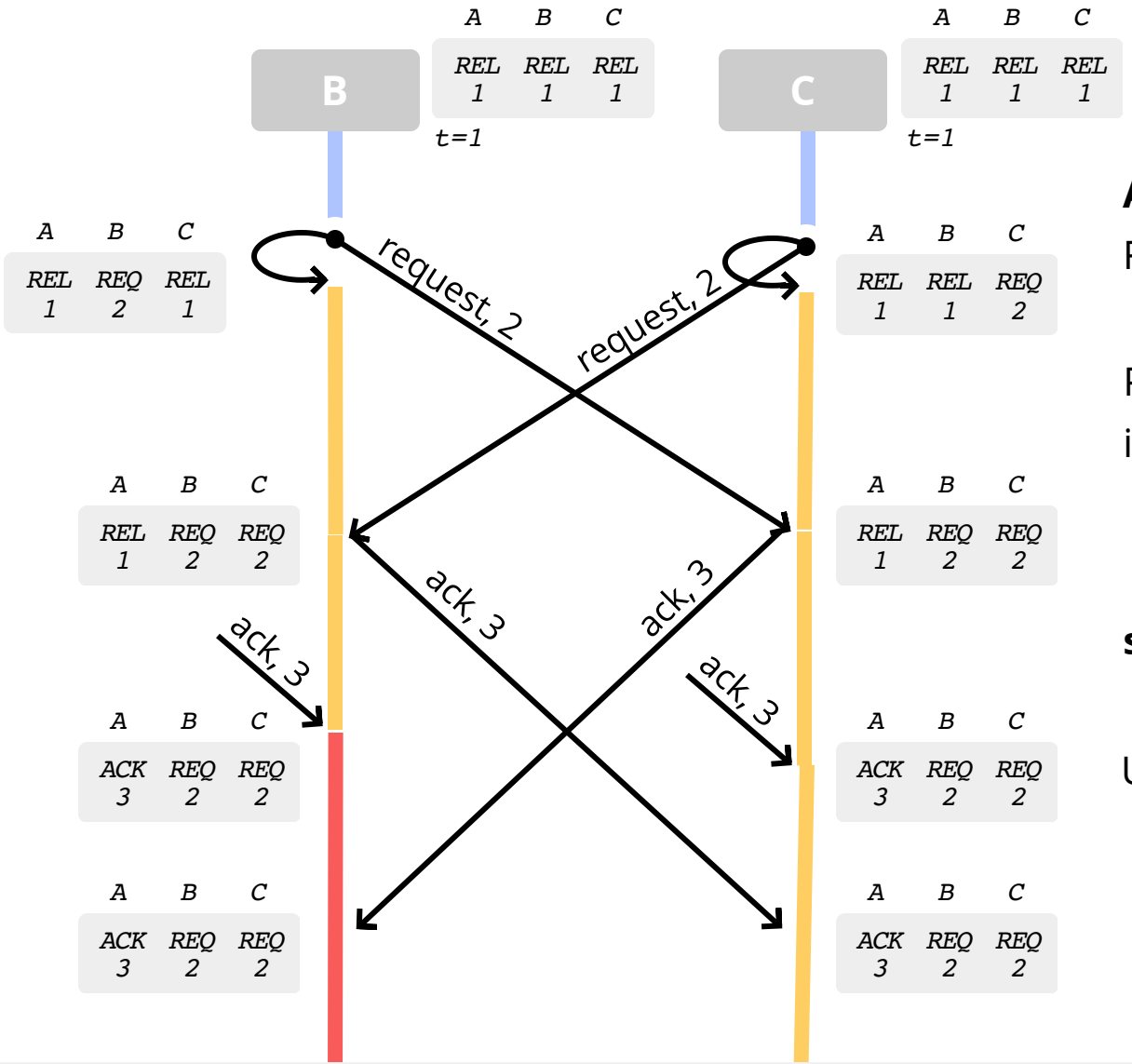
- le dernier message reçu
- le timestamp associé

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie

Version finale (?)



Amélioration :

File remplacée par tableau.

Pour chaque processus,

initialement `{REL, 1}`, puis

- le dernier message reçu
- le timestamp associé

sauf si `ACK` remplacerait `REQ`.

Un processus entre en SC si

- son dernier message est `REQ`,
- et son heure logique est **strictement la plus petite**.

Solution répartie
Version finale (?) - Résumé

On `REQ`

On `REL`

On `ACK`

Après chaque message

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.

On `REL`

On `ACK`

Après chaque message

Solution répartie

Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

On `REL`

Après chaque message

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,

On `REL`

Après chaque message

Solution répartie

Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,
sauf si un `REQ` y est déjà.

On `REL`

Après chaque message

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,
sauf si un `REQ` y est déjà.

On `REL`

Stocke message dans tableau.

Après chaque message

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,
sauf si un `REQ` y est déjà.

On `REL`

Stocke message dans tableau.

Après chaque message

Entre en SC, si

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,
sauf si un `REQ` y est déjà.

On `REL`

Stocke message dans tableau.

Après chaque message

Entre en SC, si

- j'ai la plus petite heure logique, et

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,
sauf si un `REQ` y est déjà.

On `REL`

Stocke message dans tableau.

Après chaque message

Entre en SC, si

- j'ai la plus petite heure logique, et
- c'est un `REQ`

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,
sauf si un `REQ` y est déjà.

On `REL`

Stocke message dans tableau.

Après chaque message

Entre en SC, si

- j'ai la plus petite heure logique, et
- c'est un `REQ`

Suggestions ?

Solution répartie
Version finale (?) - Résumé

On `REQ`

Stocke message dans tableau.
Envoi `ack` avec nouveau timestamp.

On `ACK`

Stocke message dans tableau,
sauf si un `REQ` y est déjà.

On `REL`

Stocke message dans tableau.

Après chaque message

Entre en SC, si

- j'ai la plus petite heure logique, et
- c'est un `REQ`

Suggestions ? *To be continued...*

Algorithme

a.k.a. Algorithme de Lamport

Résumé

Les `N` processus maintiennent un tableau de taille `N`, et
un timestamp `ts` de Lamport.

Le timestamp est incrémenté en entrée et sortie de SC,
et à toute réception de la part d'un autre processus.

Pour **demander l'entrée en SC**, le processus
- envoie un `REQ` à tous, lui-même compris, puis attend.

À la **réception de tout message,** le processus
- stocke ce message dans le tableau, dans la case de l'envoyeur,
  sauf si cela remplacerait un `REQ` par un `ACK`.
- entre en SC si sa case est `REQ` et a la plus petite heure logique.

À la **réception d'un `REQ`**, le processus
- envoie un `ACK` à l'émetteur.

Au **sortir de SC**, le processus
- envoie un `REL` à tous, lui-même compris.

Pseudocode

Variables

`n`	*entier, constant*	nombre de processus
`self`	*entier, entre 0 et n-1*	numéro de ce processus
`ts`	*entier, initialement 0*	timestamp Lamport de ce processus
`enSC`	*booléen*	vrai ssi autorisé à être en SC
`tab`	*tableau de messages*	messages de chaque processus de format `{[REQ|ACK|REL], ts}`, initialement `{REL, 0}`.

Pseudocode

Initialisation

Écouter infiniment les événements suivants :

- Demande de SC de la couche applicative
- Passage de `enSC` à `true`
- Sortie de SC dans la couche applicative
- Réception `REQ(ts)`
- Réception `ACK(ts)`
- Réception `REL(ts)`

(Avec traitement séquentiel de chaque événement)

Pseudocode

Traitement : Demande de SC de la couche applicative.

`ts` incrémenté.

Envoi de `{REQ, ts}` à tous les processus, `self` inclus.

Traitement : Passage de `enSC` à `true`.

Autorisation de la couche applicative à entrer en SC.

Traitement : Sortie de SC dans la couche applicative.

`ts` incrémenté.

Envoi de `{REL, ts}` à tous les processus, `self` inclus.

Pseudocode

Traitement : Réception `{REQ, tsi}` du processus `i`.

`ts` ← max(`tsi`, `ts`) + 1, si `i` n'est pas `self`.
`tab[i]` ← `{REQ, tsi}`.
Envoi de `{ACK, ts}` à `i`.
Tentative d'entrée en SC.

Traitement : Réception `{REL, tsi}` du processus `i`.

`ts` ← max(`tsi`, `ts`) + 1, si `i` n'est pas `self`.
`tab[i]` ← `{REL, tsi}`.
Tentative d'entrée en SC.

Traitement : Réception `{ACK, tsi}` du processus `i`.

`ts` ← max(`tsi`, `ts`) + 1, si `i` n'est pas `self`.
Si `tab[i]` n'a pas comme type `REQ`,
    `tab[i]` ← `{ACK, tsi}`.
Tentative d'entrée en SC.

Pseudocode

Tentative d'entrée en SC.

```
Si tab[self] est de type REQ
    Extraire toutes les heures logiques dans tab.
    Si la plus ancienne appartient à self,
        enSC ← true
```

Propriétés

Propriétés
Algorithme de Lamport

Propriétés
Algorithme de Lamport

Correctness Jamais plus d'un processus sera en SC à un instant T.

Propriétés
Algorithme de Lamport

Correctness Jamais plus d'un processus sera en SC à un instant T.

Progrès Toute demande d'entrée en SC sera autorisée un jour.

Propriétés
Algorithme de Lamport

Correctness Jamais plus d'un processus sera en SC à un instant T.

Progrès Toute demande d'entrée en SC sera autorisée un jour.

Complexité

Propriétés
Algorithme de Lamport

Correctness Jamais plus d'un processus sera en SC à un instant T.

Progrès Toute demande d'entrée en SC sera autorisée un jour.

Complexité

Communication par processus souhaitant entrer en SC :

Propriétés
Algorithme de Lamport

Correctness Jamais plus d'un processus sera en SC à un instant T.

Progrès Toute demande d'entrée en SC sera autorisée un jour.

Complexité

Communication par processus souhaitant entrer en SC : $3(n-1)$

Propriétés
Algorithme de Lamport

Correctness Jamais plus d'un processus sera en SC à un instant T.

Progrès Toute demande d'entrée en SC sera autorisée un jour.

Complexité

- **Communication** par processus souhaitant entrer en SC : $3(n-1)$
- **Calcul** par événement :

Propriétés
Algorithme de Lamport

Correctness Jamais plus d'un processus sera en SC à un instant T.

Progrès Toute demande d'entrée en SC sera autorisée un jour.

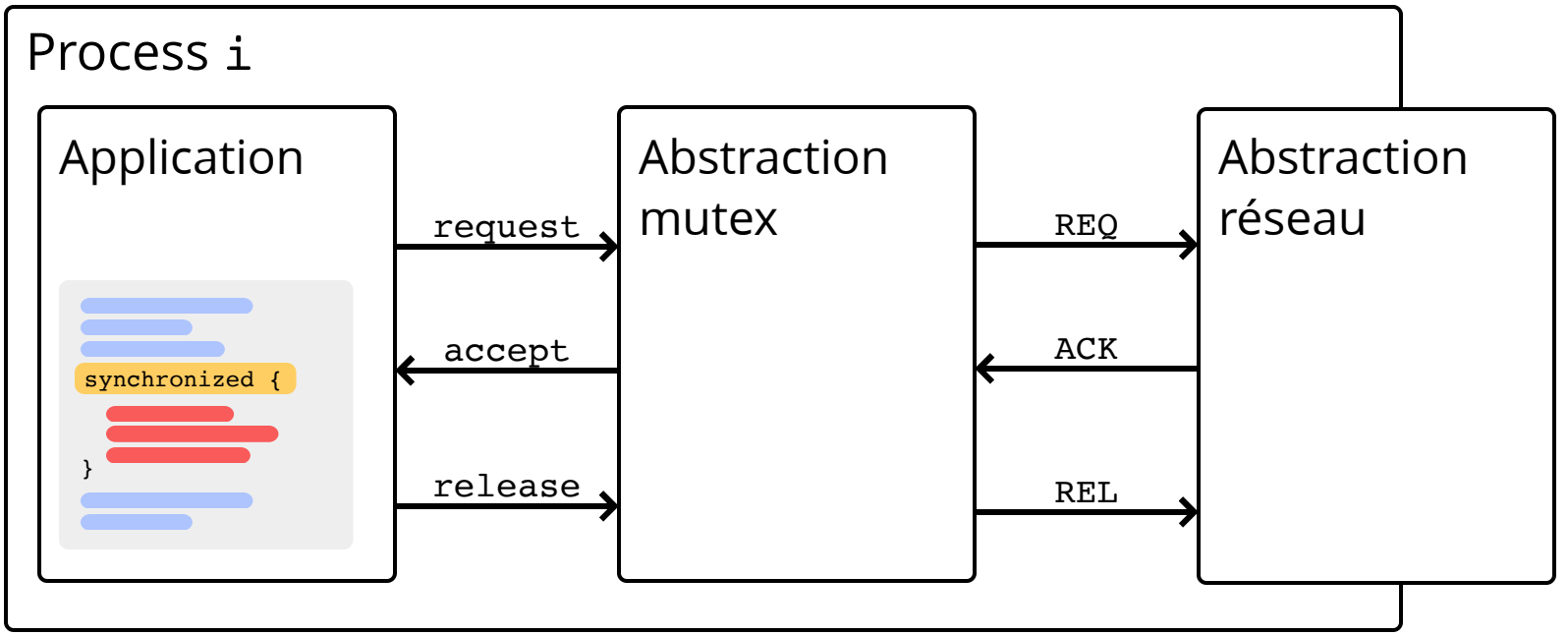
Complexité

Communication par processus souhaitant entrer en SC : $3(n-1)$

Calcul par événement : O(n) par réception de message
O(1) pour le reste.

↓ Implémentation logicielle

Implémentation logicielle



Implémenté comme une couche d'abstraction de mutex.

Couche d'abstraction réseau : indépendance de l'abstraction mutex et du système de communication choisi (tcp, udp, channels, etc).

↓ Exercices

Exercice 1

Suppositions:

- Un délai de transit de 2s existe pour toute communication.
- Tout autre délai est relativement négligeable (e.g. traitement, copie, etc.)
- Une SC dure 5s.

- Les événements simultanés sont priorisés dans l'ordre suivant :
  - demande de SC, sortie de SC, gestion de message
  - le nom de l'émetteur sert à trier deux messages reçus simultanément

- Deux processus dans le système, et initialement
  - `A` a un timestamp à `2`,
  - `B` a un timestamp à `10`.

`A` demandera la SC au bout d'une seconde, et `B` au bout de 4s.

Exercice 2

Mêmes conditions, mais

Trois processus, et initialement

- `A` a un timestamp à 10,
- `B` a un timestamp à 6,
- `C` a un timestamp à 4.

Demandes de SC

- `A` au bout de 1 seconde,
- `C` au bout de 2 secondes.

Exercice 3
Amélioration ?

Ça fait beaucoup de messages…

Y a-t-il des messages qui peuvent être omis ?

Modifiez le pseudocode en conséquence.