SDR

Estampilles

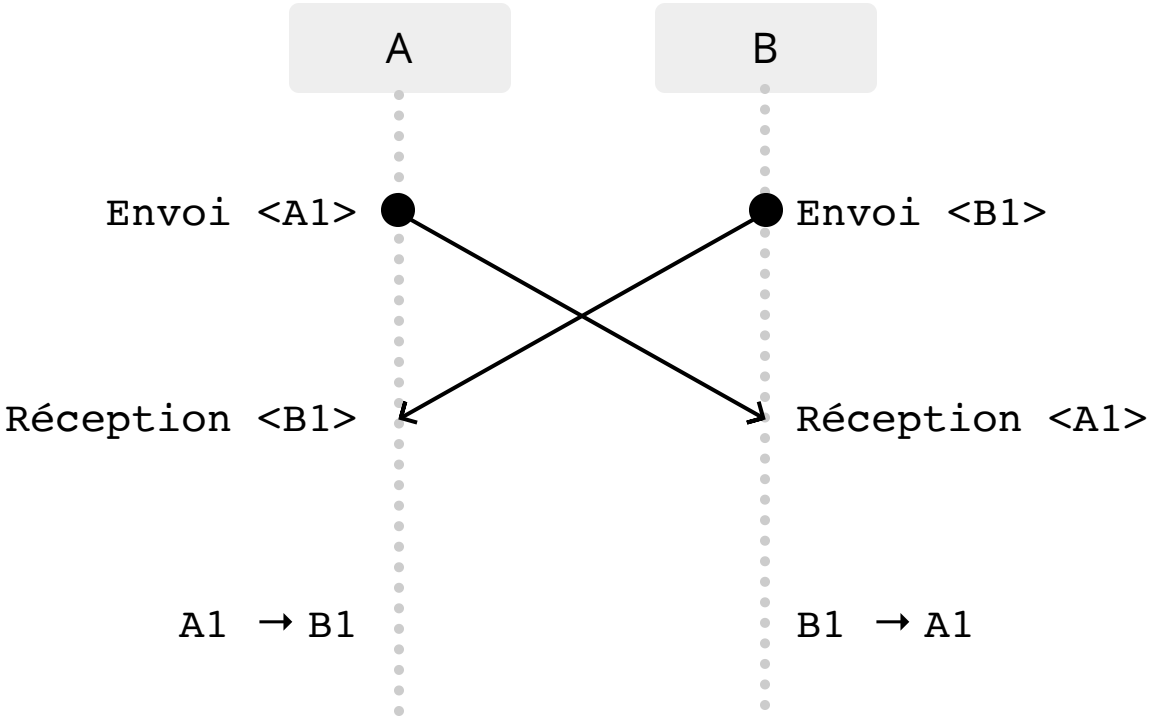
Horloges logiques

Olivier Lemer

↓ Le problème des horloges

Le problème des horloges

L'ordre des évènements n'apparait pas le même partout.



La synchronisation comme solution ?

Créer une horloge commune ?

Problème : Horloges à quartz dérivent différemment selon la machine.

Solutions : protocoles de mise à l'heure.

- Network Time Protocol : longue distance, ms precision.
- Precision Time Protocol : courte distance, µs precision.

Pas de solution permettant une horloge globale parfaite :
toujours soumis aux délais et variabilités du réseau.

Nos besoins

Résolution par requirements

Quels sont les besoins ?

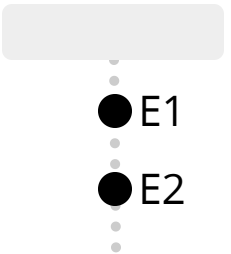
On veut connaitre ~~l'heure~~ ~~exacte~~ ~~de chaque évènement.~~
~~relative~~
l'**ordre** des évènements.
? **?**

Qu'est-ce qu'un évènement ?

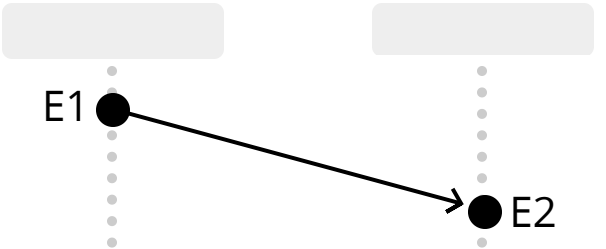
- Une action sur une machine *(exécution d'une ou plusieurs instructions)*
- Emission d'un message
- Réception d'un message

Qu'est-ce qu'un ordre d'évènements ? *Quand est-ce que E1 → E2 ?*

- Si E1 et E2 sont sur la même machine et E1 arrive avant E2.



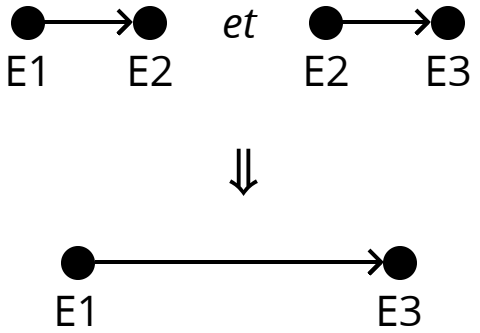
- Si E1 est l'émission d'un message et E2 sa réception.



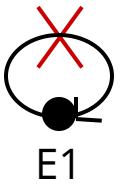
Une relation d'ordre

Propriétés

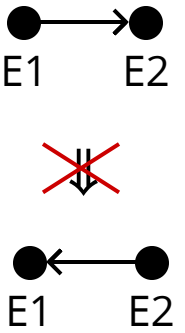
Transitif



Anti-réflexif



Anti-symétrique



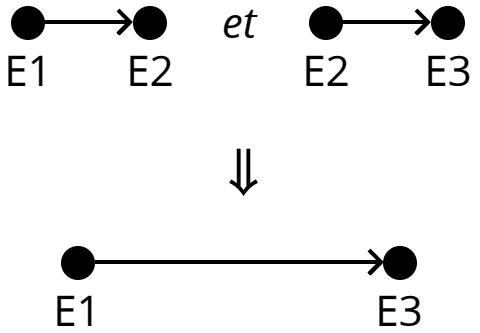
Ordre Partiel ou Total ?

(Partiel s'il existe deux évènements E1 et E2 tels que ni E1 → E2, ni E2 → E1.)

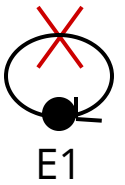
Une relation d'ordre

Propriétés

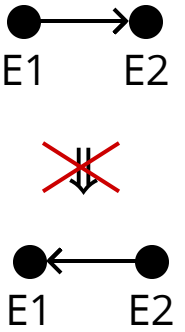
Transitif



Anti-réflexif

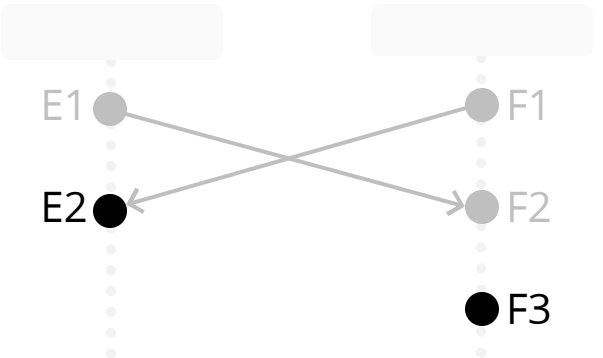


Anti-symétrique



Ordre Partiel

(i.e. il existe deux évènements E1 et E2 tels que ni E1 → E2, ni E2 → E1.)



E2 → F3 ?

F3 → E2 ?

Les deux sont possibles.
E2 et F3 n'ont pas d'ordre.
E1 et F1, E2 et F2, non plus.

↓ Horloge logique

Une horloge *logique*

Représenter l'ordre partiel des évènements.

On veut : une fonction H telle que :

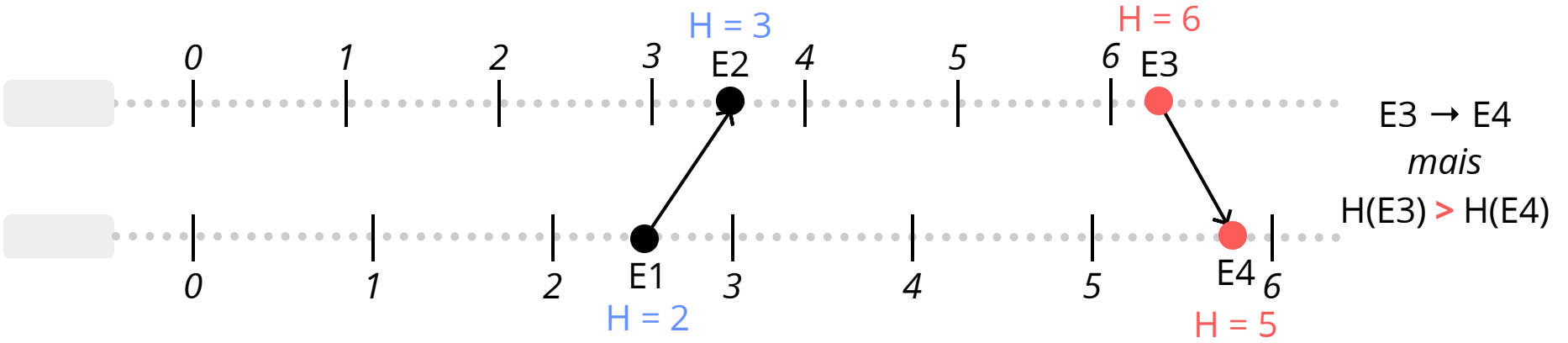
$$E1 \rightarrow E2$$

$$\Downarrow$$

$$H(E1) < H(E2)$$

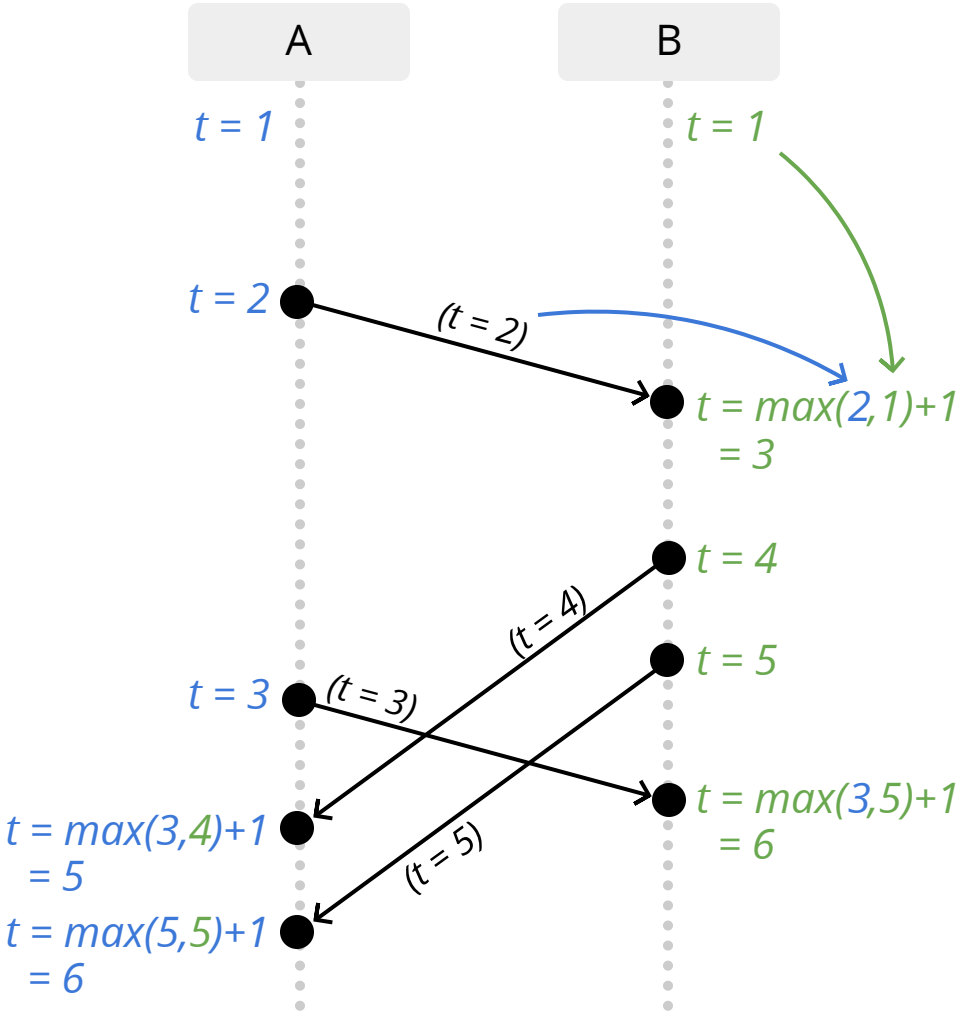
Idée 1 : horloge logique = horloge locale ?

(i.e. H(E) = heure sur la machine sur laquelle l'évènement a eu lieu)



L'horloge logique *de Lamport*

L'idée : les timestamps *(ou estampilles)*



- Chaque site maintient un numéro.
- À chaque événement *local*, le timestamp est incrémenté.
- Le timestamp est envoyé avec chaque nouveau message.
- À la réception, le timestamp le plus grand entre local et reçu est utilisé *et incrémenté*.

L'horloge logique *de Lamport*

Vérification des besoins

On voulait : une fonction H telle que :

$$E1 \rightarrow E2$$
$$\Downarrow$$
$$H(E1) < H(E2)$$

On avait défini $E1 \rightarrow E2$:

- Si E1 et E2 sont sur la même machine et E1 arrive avant E2.

$$\Downarrow$$
$$H(E1) < H(E2)$$



- Si E1 est l'émission d'un message et E2 sa réception.

$$\Downarrow$$
$$H(E1) < H(E2)$$



L'horloge logique *de Lamport*

Vérification des besoins - Un ordre total ?

On voulait : une fonction H telle que :

$$E1 \rightarrow E2$$
$$\Downarrow$$
$$H(E1) < H(E2)$$

Remarque :

$$E1 \rightarrow E2$$
$$\not\Uparrow$$
$$H(E1) < H(E2)$$

*On projette un **ordre partiel** sur un **ordre total**.*
Autrement dit, si E1 et E2 n'ont pas de relation, H leur en donne une.

L'horloge logique *de Lamport*

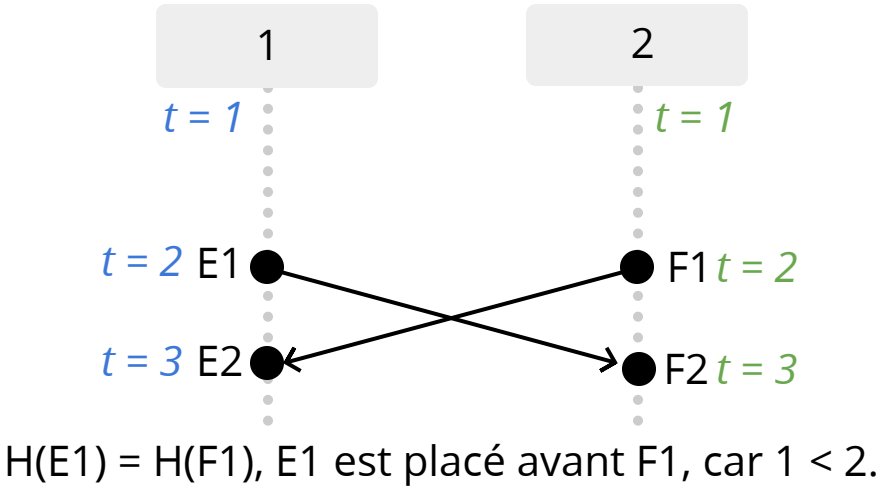
Vérification des besoins - Un ordre total *strict ?*

Remarque : On peut avoir H(E1) = H(E2).
*Ce n'est pas un ordre **strict**.*

Compliqué pour prendre des décisions, affecter des priorités, etc.

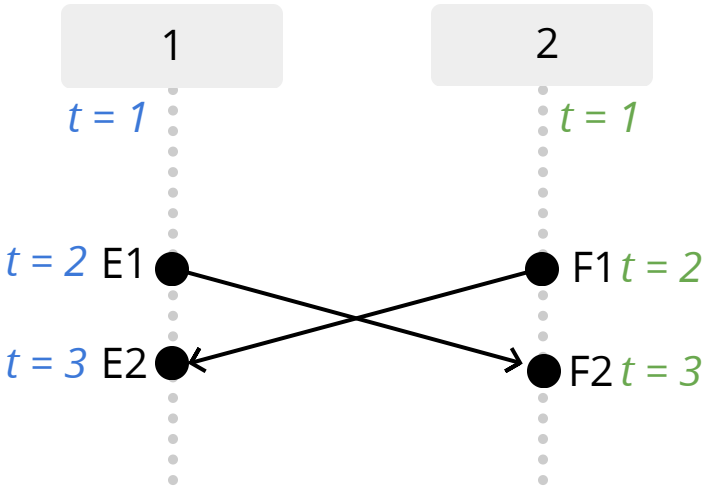
Solution : le rendre strict, arbitrairement :
priorité à la machine à l'ID le plus petit.

Si E1 arrive sur machine `i`,
E2 arrive sur machine `j`,
tels que `i` < `j`
et H(E1) = H(E2),
Alors E1 est arrivé avant E2.



L'horloge logique *de Lamport*

Ordre total rendu strict ; Exemple



Quel est l'ordre ?

E1, F1, E2, F2.