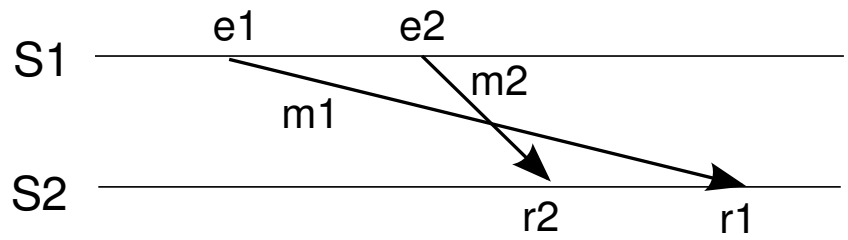
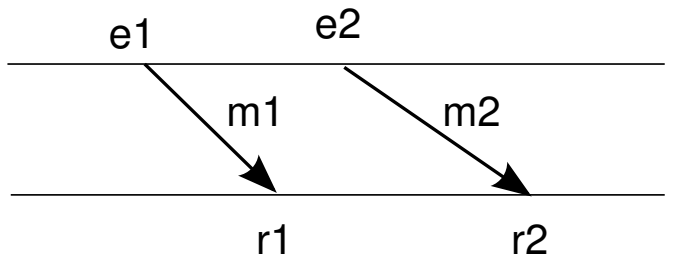
Plan

Protocole FIFO

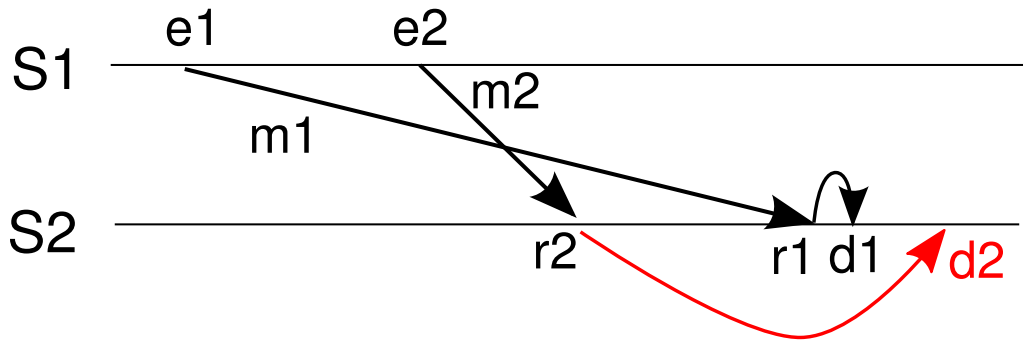


Non FIFO



FIFO

$$\forall m, m' : s_1 \xrightarrow{m} s_2 \wedge s_1 \xrightarrow{m'} s_2 \wedge e(m) \prec e(m') \Rightarrow r(m) \prec r(m')$$

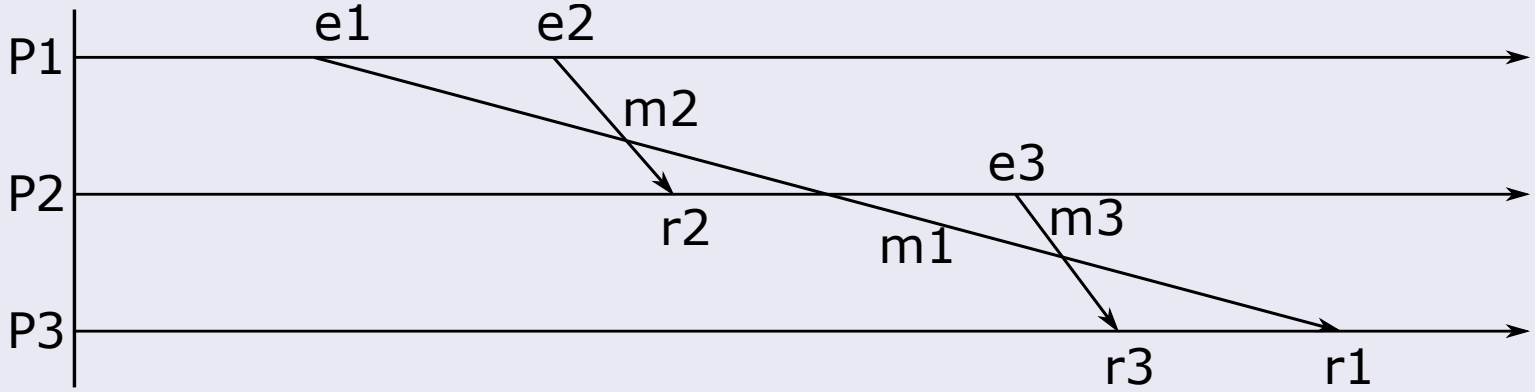


délivrance $\neq$ réception

Protocole causalement ordonné

Objectif : Mettre de l'ordre

Réceptions incohérentes par rapport aux émissions



$r_3 \prec r_1$ alors que $e_1 \prec e_3$

$\Rightarrow$ délivrance $\neq$ réception

1. *Reliable communication in the presence of failures*, Kenneth P. Birman and Thomas A. Joseph. ACM Transactions on Computer Systems, January 1987.

Protocole FIFO

♪♪♪

Objectif

Garantir la cohérence des réceptions sur un même site par rapport à leur éventuelle émission depuis un même site

$\Rightarrow$ réordonner les messages reçus sur un site

- Trois événements au lieu de deux par message :
  - l'émission e,
  - la réception r,
  - la délivrance d.
  - Causalité : $e \prec r \prec d$
- S'exprime par la propriété :

$$\forall s_1, s_2, m, m' \ : \ s_1 \xrightarrow{m} s_2 \ \wedge \ s_1 \xrightarrow{m'} s_2 \ \wedge \ e(m) \prec e(m') \Rightarrow d(m) \prec d(m')$$

Protocole FIFO

Réalisation : il suffit de numéroter les messages pour chaque canal (couple site d'émission, site de réception)
Récepteur pour un canal :

```
type Message = ⟨contenu, numéro⟩;
int prochain = 0;
SortedSet<Message> enAttente;  // trié par numéro
while (true) {
    recevoir m;
    enAttente.add(m);
    while (enAttente.first().numéro == prochain) {
        m ← enAttente.removeFirst();
        délivrer m.contenu
        prochain++;
    }
}
```

Protocole causalement ordonné

Objectif

Garantir la cohérence des réceptions sur un même site par rapport à leur causalité éventuelle en émission

$\Rightarrow$ réordonner les messages reçus sur un site

- Trois événements au lieu de deux par message :
  - l'émission e,
  - la réception r,
  - la délivrance d.
  - Causalité : $e \prec r \prec d$
- S'exprime par la propriété :

$$\forall s, m, m' \ : \ \_ \xrightarrow{m} s \ \wedge \ \_ \xrightarrow{m'} s \ \wedge \ e(m) \prec e(m') \Rightarrow d(m) \prec d(m')$$

[Précis 3.2 pp.47–50]

Histoire causale

♪♪♪

Histoire causale d'un message $H_c(m)$

L'histoire causale $H_c(m)$ d'un message m est l'ensemble des messages qui ont leurs événements d'émission précédant causalement l'émission de m :

$$H_c(m) = \{m' : e(m') \prec e(m)\}$$

Critère de délivrance d'un message

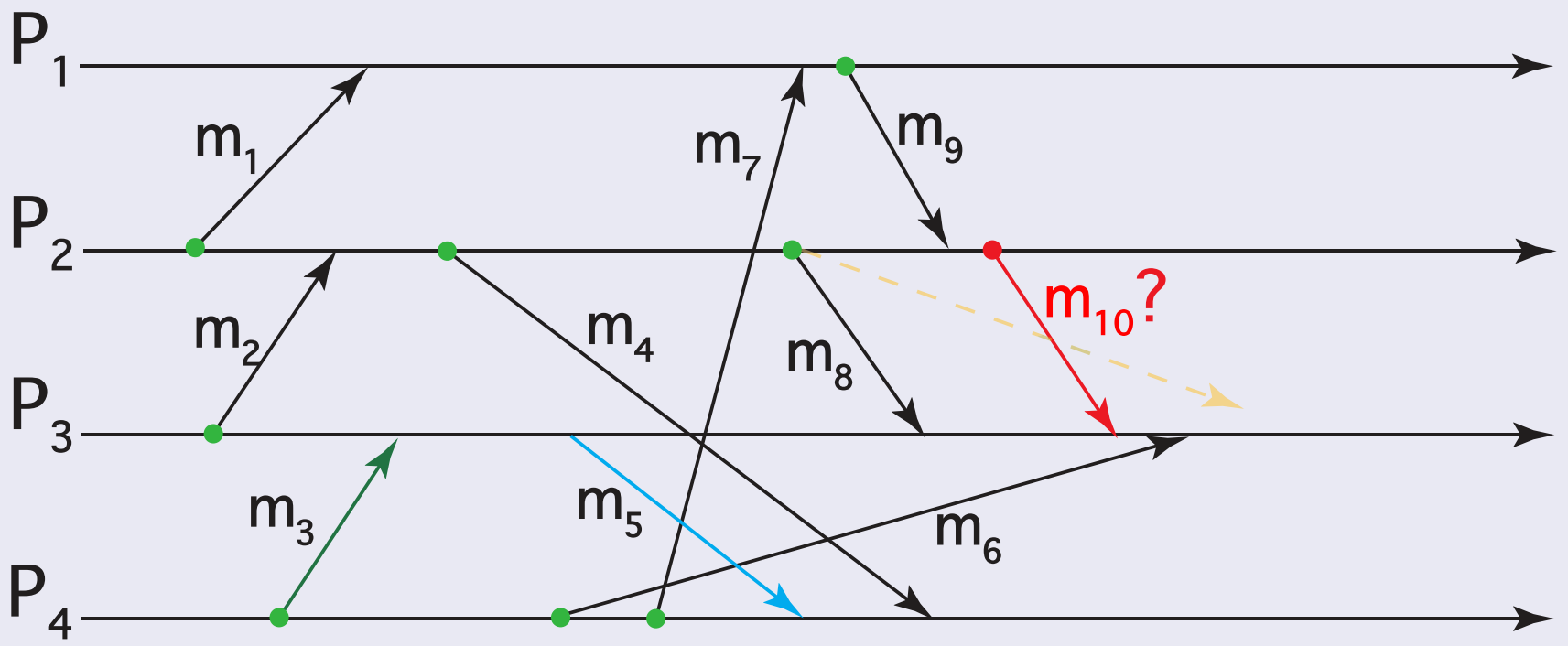
Un message m est délivré sur un site s ssi tous les messages de son histoire causale ayant aussi s comme site de destination ont été déjà délivrés :

$$\forall s, m' \in H_c(m) : \_ \xrightarrow{m} s \ \wedge \ \_ \xrightarrow{m'} s \wedge \ \Rightarrow d(m') \prec d(m)$$

Histoire causale : exemple

♪♪♪

Exemple



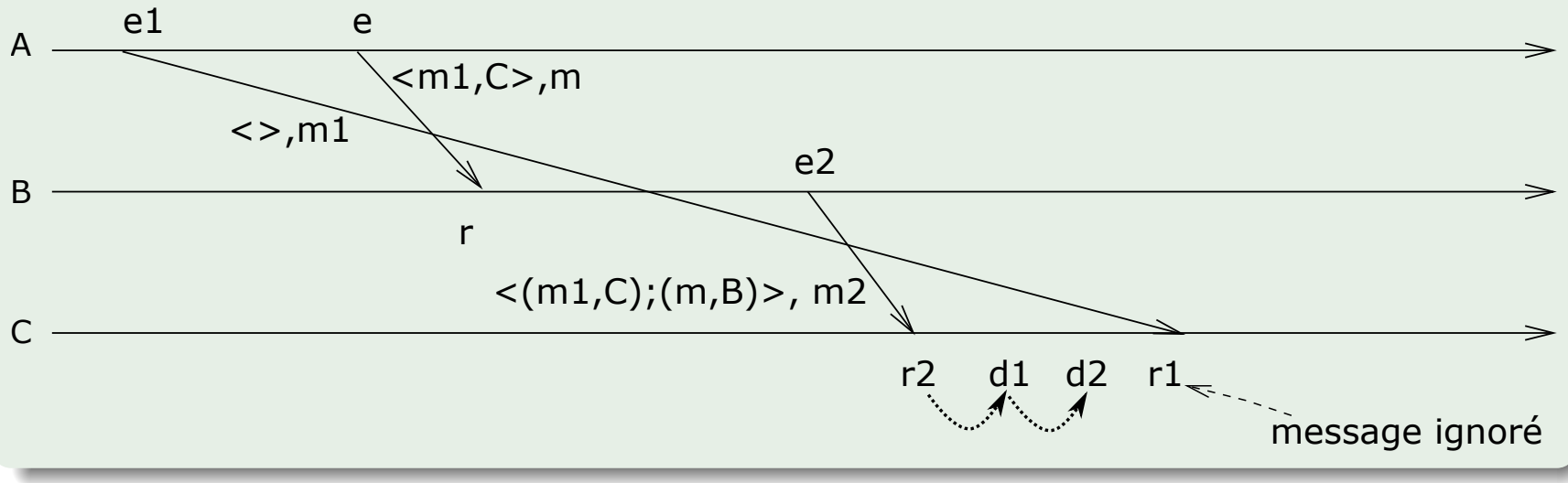
$H_c(m_{10}) = \{m_8, m_4, m_9, m_7, m_1, m_6, m_3, m_2\}$
$H_c(m_6) = \{m_3\}$
$H_c(m_8) = \{m_4, m_1, m_2\}$

Approche par surcharge (piggybacking) ♪♪♪

Principe

- Surcharger chaque message avec l'histoire des messages qui le précèdent causalement
- Dans le contexte du courrier électronique : Approche similaire du « réexpédier » avec copie de ce que l'on a reçu

Exemple



Approche par surcharge (piggybacking) ♪♪♪

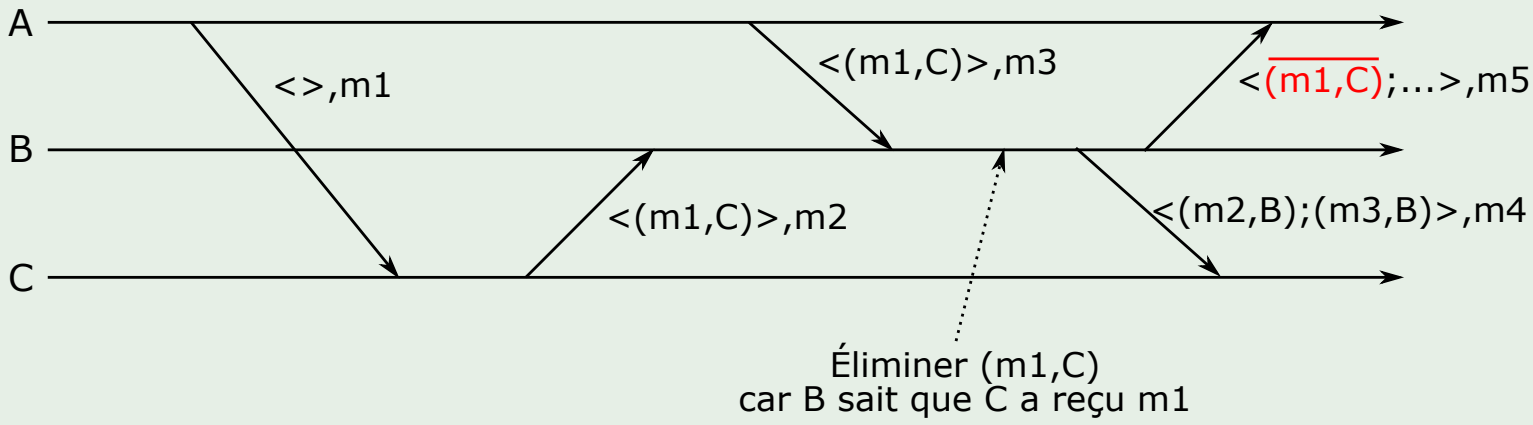
Mise en œuvre

☺ Simple et apport d'une certaine tolérance aux pertes de messages par redondance

☹ Messages de + en + longs

⇒ Quand, comment réduire les histoires ?

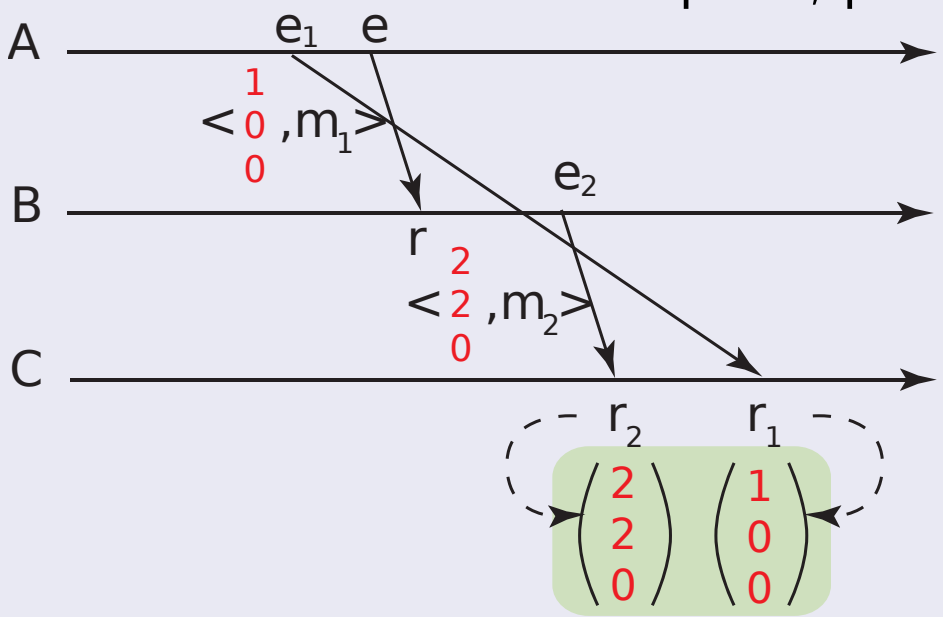
Exemple



Datation, premier essai...

Datation causale (horloge vectorielle)

Permet la détection des anomalies en réception, pas leur prévention



- Lors de r_2, C sait qu'il y a 2 événements sur A et 2 sur B (dont e_2) dans le passé de m_2, mais le concernent-ils ?
- Lors de r_1, C découvre qu'un événement de A, causalement antérieur à r_2, le concerne.

Approche par matrice

♪♪♪

Structures de données

Représenter l'histoire causale de chaque message

Chaque site S_s gère :

- MP_s : une matrice de précédence causale
  $MP_s[i, j]$ = nombre de messages émis de S_i vers S_j, connu de S_s
- $Dernier_s$: un vecteur de compteurs des messages reçus de chaque site
  $Dernier_s[i]$ = nombre de messages reçus du site S_i sur S_s
- Tout message est surchargée par une copie de la matrice MP du site émetteur

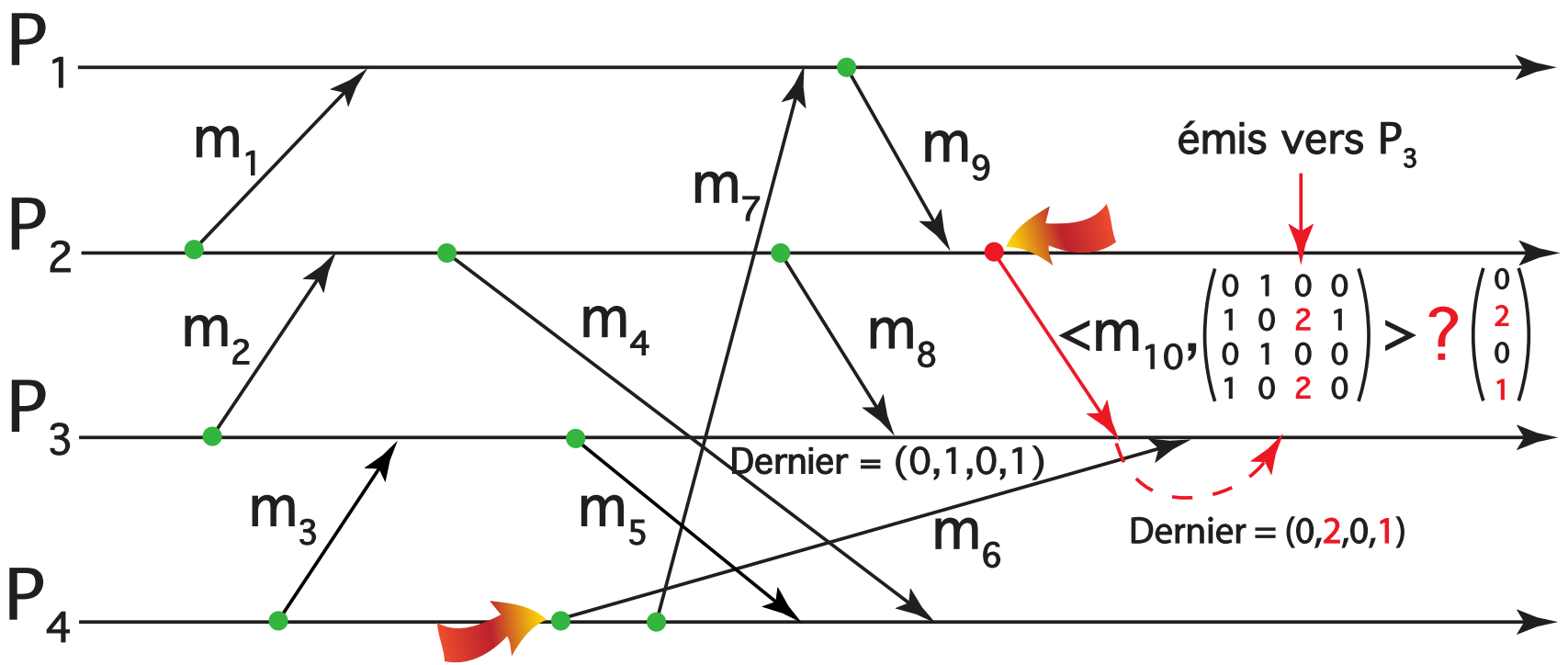
1. *The Causal Ordering Abstraction and a Simple Way to Implement it*, Michel Raynal, André Schiper and Sam Toueg. Information Processing Letters, 1991.

Actions associées aux événements

Type d'événement sur un site s	Action sur le site s
Émission sur s de m vers s'	$MP_s[s, s'] + +$ envoi de $\langle MP_s, M \rangle$
Réception sur s de $\langle MP, m \rangle$ issu de s'	$MP_s \leftarrow max(MP, MP_s)$ $Dernier_s[s'] + +$
Délivrance sur s de $\langle MP, m \rangle$ issu de s'	Délivrable(m) $\stackrel{\Delta}{=}$ $Dernier_s[s'] = MP[s', s]$ $\wedge \forall i \neq s : Dernier_s[i] \geq MP[i, s]$

m délivrable $\stackrel{\Delta}{=}$ FIFO entre s' et s et il ne précède pas les messages dont l'émission le précède causalement.

Contrôle des délivrances



Horloges de plus en plus précises

Horloges de Lamport

Passé de l'ensemble du système, connu de s, réduit à la longueur de la plus longue chaîne causale aboutissant à l'événement.

Horloges vectorielles

Connaissance de premier ordre : passé de s' que s connaît, connaissance par s que s' a eu un certain nombre d'événements.

Horloges matricielles

Connaissance de second ordre : connaissance par s de la connaissance par s' du passé de s''.

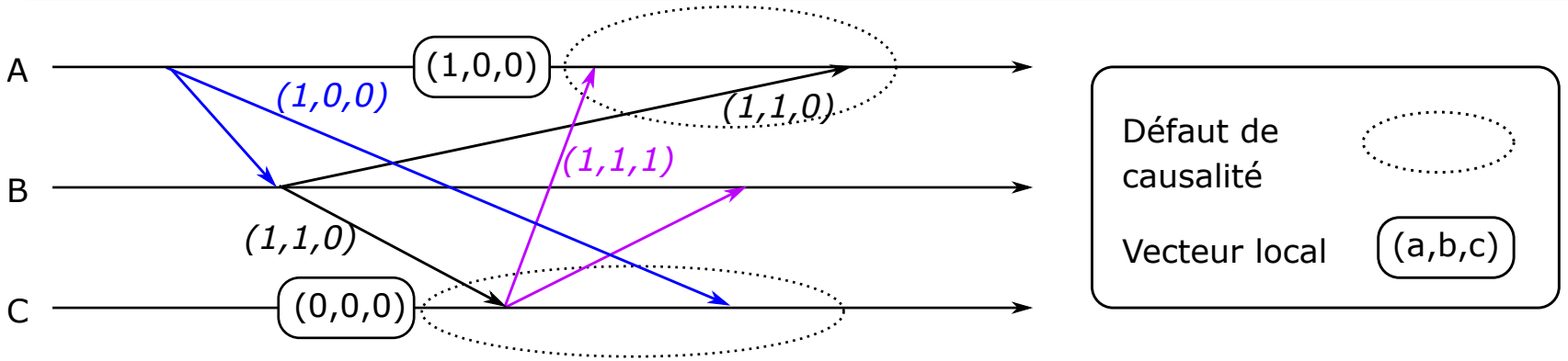
Par exemple, permet de savoir que *tous les sites* connaissent un événement donné.

Diffusion causalement ordonnée

La diffusion ordonnée est plus simple que la communication point-à-point !

Approche par matrice causale

- Il suffit de gérer un vecteur d'émission au lieu d'une matrice
- Toutes les colonnes sont identiques : un processus envoie le même nombre de messages à tous



[Précis 3.2.2 pp.50–51]

1. *Lightweight Causal and Atomic Group Multicast*, Kenneth P. Birman, André Schiper and Pat Stephenson. ACM Trans. on Computer Systems, 1991.

Conclusion

- Datation logique des événements
- Protocoles de communication ordonnés
- Distinction réception / délivrance