

Systèmes et algorithmes répartis

Causalité et datation

Philippe Quéinnec, Gérard Padiou

Département Informatique et Mathématiques Appliquées
ENSEEIH

31 août 2016



plan

- 1 Problème de datation
 - Temps réel vs temps logique
 - Horloge de Lamport
 - Horloge vectorielle de Fidge-Mattern
- 2 Les protocoles de communication
 - Délivrance ordonnée
 - Protocoles ordonnés
 - Protocole causalement ordonné
 - Diffusion causalement ordonnée



Plan

- 1 Problème de datation
 - Temps réel vs temps logique
 - Horloge de Lamport
 - Horloge vectorielle de Fidge-Mattern
- 2 Les protocoles de communication
 - Délivrance ordonnée
 - Protocoles ordonnés
 - Protocole causalement ordonné
 - Diffusion causalement ordonnée



Problème de base et difficultés

Comment dater les événements d'une exécution répartie ?

- Pas d'horloge globale
- Tous les événements ne sont pas causalement liés
- Datation cohérente avec la relation causale :

$$\forall e, e' : e \prec e' \Rightarrow d_e < d_{e'}$$

Exemple

- Idée : → Utiliser les horloges matérielles de chaque site
- **Risque** : Datation incohérente de l'événement de réception d'un message : la date de réception précède la date d'émission
- **Possible** si l'horloge du site de réception est en retard (suffisamment) sur celle du site de l'émetteur

Temps réel

Difficultés

- Cohérence avec la causalité
- Datation définissant un ordre total

Solutions

Nécessite une synchronisation des horloges locales :

$$\text{invariant } \max_{i=1..N} (h_i) - \min_{i=1..N} (h_i) < \epsilon$$

Protocole de synchronisation d'horloges possible mais dans des contextes réseaux assurant une certaine qualité de service ou par l'usage d'un signal diffusé (horloge atomique)



Temps logique

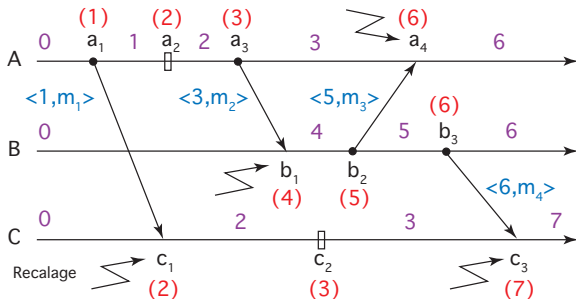
Principes de base

- Compter les événements au lieu du temps réel
- Associer à chaque site une horloge logique locale
- Surcharger les messages avec leur date d'émission
- **Recaler si nécessaire** l'horloge locale d'un site lors de chaque réception de message
- **Avantage** : vision plus abstraite d'un calcul réparti



Une tentative...

- Un compteur « horloge » sur chaque site
- Surcharge des messages et recalage de compteur



😊 $e \prec e' \Rightarrow d(e) < d(e')$

😊 Mêmes dates \Rightarrow pas causalement liés


😞 mais $d(e) < d(e') \not\Rightarrow e \prec e'$ (ex : $d(c_2) < d(b_3) \not\Rightarrow c_2 \prec b_3$)

Horloge de Lamport

Propriétés

- Introduit un ordre total entre événements
→ Dates distinctes pour tout couple d'événements
- Date = (numéro de site, compteur local)
⇒ ordre total sur les sites (par leur numéro)

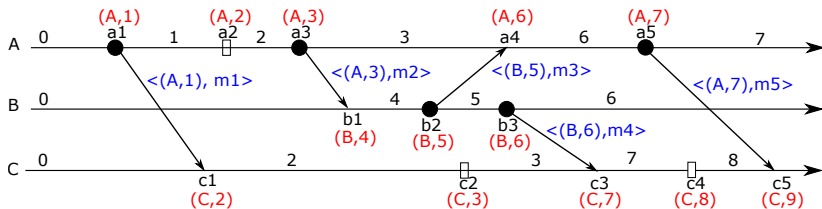
```
struct Date {  
    int s; // numéro de site  
    int cpt; // compteur d'événements  
}  
bool prec (Date d1, Date d2) {  
    return (d1.cpt < d2.cpt)  
        || ((d1.cpt == d2.cpt) && (d1.s < d2.s));  
}
```



Actions associées aux événements

L'événement est daté avec le résultat de l'action.

Type d'événement sur un site s	Action sur le site s
Événement interne sur s	$H_s \leftarrow H_s + 1;$
Émission sur s de m	$H_s \leftarrow H_s + 1;$ envoi de $\langle H_s, m \rangle;$
Réception sur s de $\langle dm, m \rangle$	$H_s \leftarrow \max(H_s, dm) + 1;$



Horloge vectorielle de Fidge-Mattern (1988)

Propriétés

- datation isomorphe à l'ordre causal
- utilisation de vecteurs de dimension égale au nombre de sites
- coût plus élevé : surcharge des messages par un vecteur
- Pour un événement e , $HV(e)[i] =$ nombre d'événements du passé de e sur p_i (y compris e)

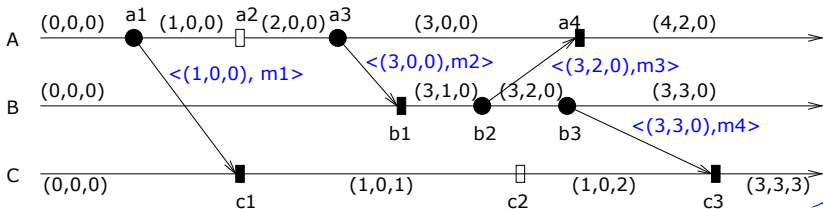
Expression des relations entre dates

$$\begin{aligned} D \leq D' &\triangleq \forall i : D[i] \leq D'[i] \\ D < D' &\triangleq D \leq D' \wedge \exists k : D[k] < D'[k] \\ D \parallel D' &\triangleq \neg(D < D') \wedge \neg(D' < D) \end{aligned}$$

Actions associées aux événements

L'événement est daté avec le résultat de l'action.

Type d'événement sur un site s	Action sur le site s
Événement interne sur s	$H_s[s] \leftarrow H_s[s] + 1$
Émission sur s de m	$H_s[s] \leftarrow H_s[s] + 1$ envoi de $\langle H_s, m \rangle$
Réception sur s de $\langle dm, m \rangle$	$H_s[s] \leftarrow H_s[s] + 1$ $H_s[s'] \leftarrow \max(H_s[s'], dm[s']), \forall s' \neq s$



Horloge vectorielle de Fidge-Mattern

Datation isomorphe à l'ordre causal

$$e \prec e' \Leftrightarrow D(e) \leq D(e') \quad \text{et} \quad e \parallel e' \Leftrightarrow D(e) \parallel D(e')$$

Coupure

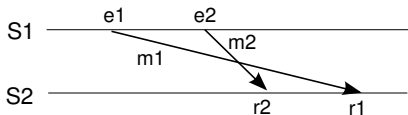
- Date d'une coupure $C = (c_1, \dots, c_n)$:
 $HV(C) \triangleq \sup(HV(c_1), \dots, HV(c_n))$
- $C_2 \subset C_1 \Leftrightarrow HV(C_2) < HV(C_1)$
- C cohérente $\Leftrightarrow HV(C) = \langle HV(c_1)[1], \dots, HV(c_n)[n] \rangle$
- Soit C cohérente. Alors $\forall i, j : HV(c_i)[i] \geq HV(c_j)[i]$.
L'incrémement de $HV(c_i)[i]$ ne peut venir que d'un événement local ou résulter d'un message provenant du passé.
- Soit C non cohérente.
 \Rightarrow il existe un événement e_i hors coupe qui est dans le passé causal de C
 \Rightarrow Sur le site $i : HV(c_i)[i] < HV(e_i)[i]$ et $HV(e_i)[i] \leq HV(C)[i]$
 $\Rightarrow HV(C)[i] > HV(c_i)[i]$, donc $HV(C) \neq (HV(c_1)[1], \dots, HV(c_n)[n])$

Plan

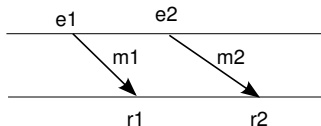
- 1 Problème de datation
 - Temps réel vs temps logique
 - Horloge de Lamport
 - Horloge vectorielle de Fidge-Mattern
- 2 Les protocoles de communication
 - Délivrance ordonnée
 - Protocoles ordonnés
 - Protocole causalement ordonné
 - Diffusion causalement ordonnée



Protocole FIFO

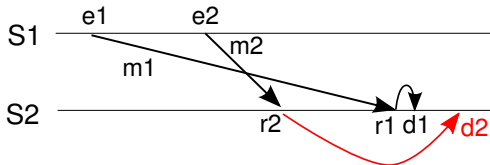


Non FIFO



FIFO

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

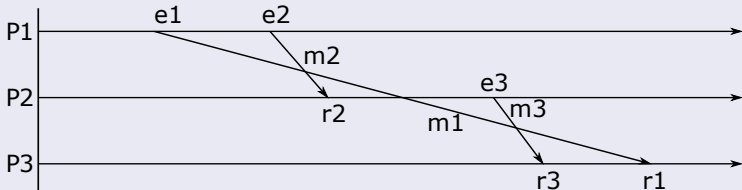


délivrance \neq réception

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

Protocole FIFO

Objectif

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

⇒ 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 :

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

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 :

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

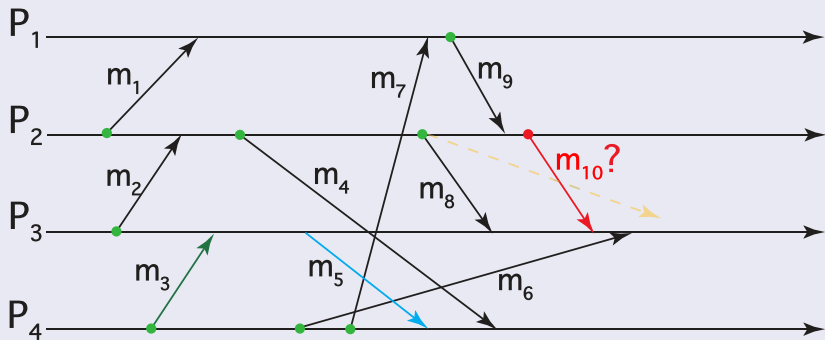
Critère de délivrance d'un message

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



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

$$H_c(m_6) = \{m_3\}$$

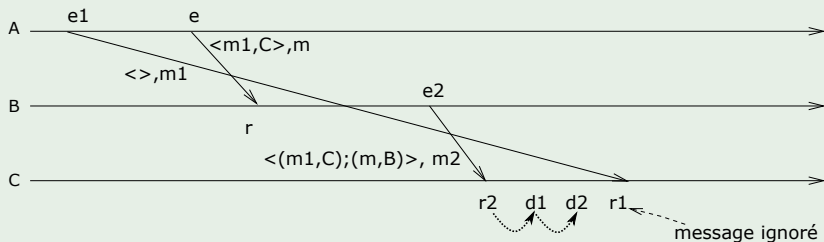
$$H_c(m_8) = \{m_4, m_1, m_2\}$$

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

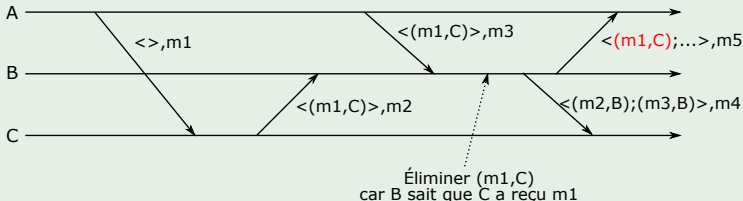


Approche par surcharge (piggybacking)

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 ?

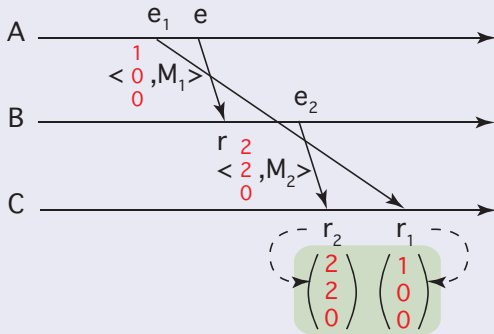
Exemple



Datation, premier essai...

Datation causale (horloge vectorielle)

Permet la détection des anomalies en réception MAIS pas leur prévention



27

Approche par compteurs (Raynal)

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

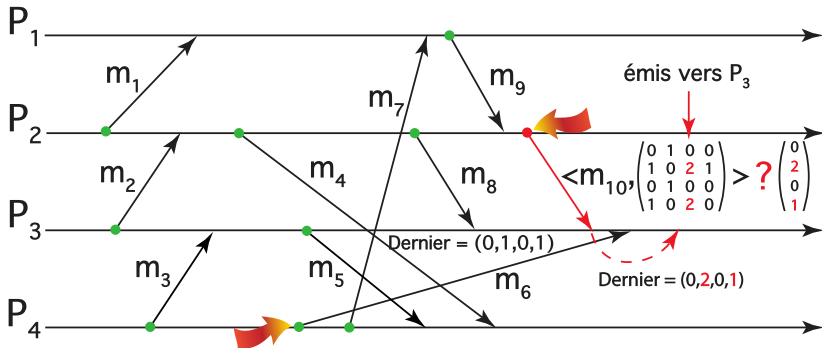
Actions associées aux événements

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'	Délivvable(M) \triangleq $Dernier_s[s'] = MP[s', s]$ $\wedge \forall i \neq s : Dernier_s[i] \geq MP[i, s]$

M Délivvable \triangleq 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

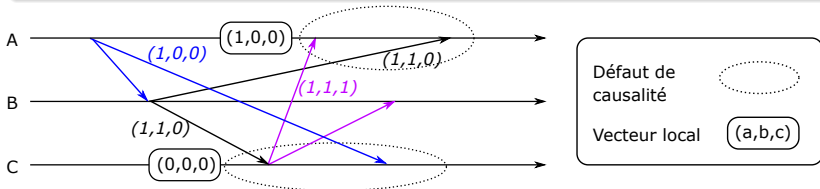


Diffusion causalement ordonnée

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 lignes sont identiques : un processus envoie le même nombre de messages à tous



Conclusion

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

