

Algorithmique Distribuée

Temps et horloges distribués

Laurent PHILIPPE

Master 2 Informatique
UFR des Sciences et Techniques

2013/2014

Sommaire

- 1 Introduction
- 2 Relations entre événements
- 3 Les horloges logiques
- 4 Synchronisation d'horloges physiques

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Synchronisation d'horloges

Systèmes Centralisés

- Accès à la date courante par un appel système
- Une seule horloge
- Pas d'ambiguïté pour dater des objets ou des événements

Système Distribué

- Chaque machine dispose d'une horloge indépendante
- Horloge à quartz peut dériver de plusieurs secondes par jour de manière aléatoire (fréquence, alimentation)
- Réglage initial différent
- Risque d'ambiguïté

⇒ Difficile pour tous les sites d'être d'accord sur l'heure

Problème en cas d'absence d'horloge globale : exemple

Programme `make` de UNIX

- code source des applications découpé en plusieurs fichiers sources
- toute modification de l'un de ces fichiers requiert la compilation de celui-ci et non pas la compilation de l'ensemble des fichiers sources
- `make` ne compile que les fichiers sources qui ont été modifiés depuis la dernière compilation
- quels fichiers sources modifiés ? comparaison entre les dates de modification des fichiers sources et objets correspondants

Programme make de UNIX

Centralisé

- Fonctionnement de `make`
- Éditeur de programme fixe la date du fichier source
- Compilateur fixe la date du fichier objet
- Date fixée par la même horloge

Distribué

- Éditeur et compilateur peuvent s'exécuter sur des sites différents
 - `fichierA.o` compilé sur le site A à la date 110
 - `fichierA.c` modifié sur le site B à la date 108

Synchronisation d'horloge

Question

Peut-on synchroniser toutes les horloges d'un système distribué pour obtenir une horloge unique, standard et non ambiguë ?

Les horloges logiques

Dans un article célèbre, Lamport^a montre que la synchronisation d'horloges est possible et propose un algorithme qui permet de la réaliser

a. Lamport, L. (1978). "Time, clocks, and the ordering of events in a distributed system". Communications of the ACM 21 (7) : 558–565

Lamport montre que la synchronisation ne doit pas forcément être *absolue*

- Dans le cas où les processus n'ont aucune interaction, il n'est pas nécessaire que leurs horloges soient synchrones
- Le problème n'est pas que tous les processus aient la même date mais qu'ils soient d'accord sur l'ordre dans lequel les événements se produisent : *horloges logiques*

Problématique

- Comment ordonner les évènements tel qu'à chaque évènement E soit associé une horloge $H(E)$ sur laquelle tous les processus soient d'accord ?
- Cette horloge doit vérifier la propriété suivante :
$$\text{si } E_1 \text{ a lieu avant } E_2 \text{ alors } H(E_1) < H(E_2)$$
- Relation de précédence

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Relation de précédence

Définition

- Introduite par Lamport : "happens-before"
- Définie par les cas suivants où E_1 précède E_2 :
 - deux événements E_1 et E_2 qui se suivent dans un même processus
 - émission d'un message E_1 par un processus et sa réception E_2 par un autre processus
- L'événement E_2 a potentiellement accès à toutes les informations de l'événement E_1

Notation

- La relation de précédence est notée : \rightarrow
- L'expression $E_1 \rightarrow E_2$, se lit l'événement E_1 précède l'événement E_2 , c'est à dire que E_1 a lieu avant E_2

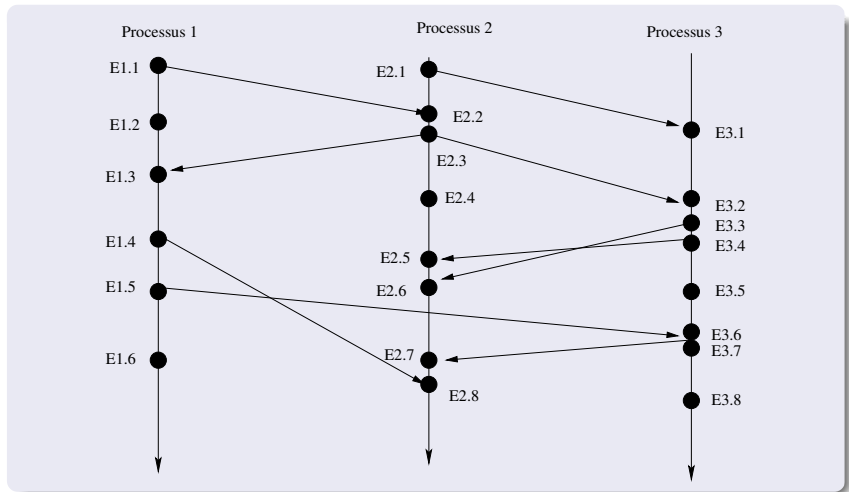
Propriétés de la relation de précédence

Relation d'ordre

La précédence définit une relation d'ordre car elle satisfait les propriétés suivantes :

- Transitivité :
Si $E_1 \rightarrow E_2$ et $E_2 \rightarrow E_3$ alors $E_1 \rightarrow E_3$
- Irréflexibilité :
on n'a pas $E_1 \rightarrow E_1$
- Antisymétrie :
si $(E_1 \rightarrow E_2)$ alors $\neg(E_2 \rightarrow E_1)$
-

Exemple de relation de précedence



Exemple de relation de précédence

Précédence entre les événements

- Au sein de P_1 , on a : $E_{1.1} \rightarrow E_{1.2} \rightarrow E_{1.3}$
- Au sein de P_2 , on a : $E_{2.1} \rightarrow E_{2.2}$
- De manière transitive on a, au sein du processus P_3 :
 $E_{3.2} \rightarrow E_{3.4}$ et $E_{3.2} \rightarrow E_{3.5}, \dots$
- Et de manière distribuée :
 - Entre deux processus P_1 et P_2 : $E_{1.1} \rightarrow E_{2.2}$ mais aussi, par transitivité $E_{1.1} \rightarrow E_{2.2}$
 - Et par rebond de transitivité entre P_1 , P_1 et P_3 : $E_{1.1} \rightarrow E_{3.2}$,
 $E_{1.5} \rightarrow E_{3.8}$, $E_{3.3} \rightarrow E_{2.7}$

Propriétés de la relation de précédence

Remarques

- La relation de précédence détermine un ordre partiel entre les événements : il n'est pas toujours possible de mettre en relation de précédence deux événements. Par exemple $\neg(E_{1.1} \rightarrow E_{2.1})$ et $\neg(E_{2.1} \rightarrow E_{1.1})$
- $E_1 \nrightarrow E_2$ signifie que les événements E_1 et E_2 ne dépendent ni directement ni transitivement l'un de l'autre.
- Dans ce cas E_2 n'est pas affecté par E_1 ni aucun autre événement qui a lieu après E_1 dans le même processus
- Sur la figure : $E_{1.4} \nrightarrow E_{2.5}$
- On peut noter :
 - $\forall E_1, E_2 : E_1 \nrightarrow E_2 \nRightarrow E_2 \nrightarrow E_1$
 - $\forall E_1, E_2 : E_1 \rightarrow E_2 \Rightarrow E_2 \nrightarrow E_1$

Relation de concurrence

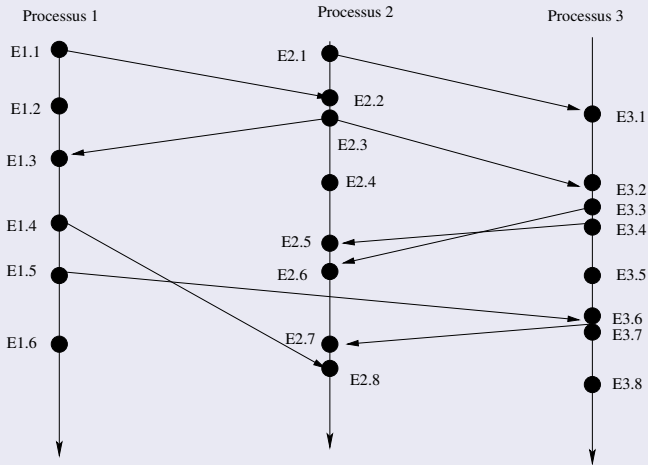
Définition

Deux évènements E_1 et E_2 sont concurrents si et seulement si ils n'ont pas de relation de précédence : $\neg(E_1 \rightarrow E_2) \wedge \neg(E_2 \rightarrow E_1)$, ce qui est noté $E_1 \parallel E_2$

Remarques

- La relation n'est pas transitive $E_1 \parallel E_2$ et $E_2 \parallel E_3 \not\Rightarrow E_1 \parallel E_3$
- $\forall E_1, E_2$ nous avons $E_1 \rightarrow E_2$ ou $E_2 \rightarrow E_1$ ou $E_1 \parallel E_2$
- Sur la figure : $E_{1.2} \parallel E_{2.3}$ et $E_{2.4} \parallel E_{3.2}$
- Des suites d'exécutions peuvent être concurrentes :
 $E_{11} \rightarrow E_{12} \rightarrow E_{13} \parallel E_{21} \rightarrow E_{22}$
- La concurrence "logique" ne signifie pas que les événements sont exécutés en même temps contrairement à la concurrence "physique"

Exemple de relation de précedence



Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Relation de causalité

Définition

Un événement E_1 produit un événement E_2 si et seulement si E_2 est une conséquence de E_1 . Par exemple, l'émission d'un message produit la réception de celui-ci.

- La relation de causalité est notée \nearrow
- L'expression $E_1 \nearrow E_2$, se lit l'événement E_1 produit l'événement E_2 , c'est à dire que E_2 est la conséquence de E_1
- Liée à l'algorithme : deux événements ayant une relation de précédence n'ont pas forcément de relation de causalité
- Exemple : un envoi de message peut suivre une exécution sans contenir de variable affectée par l'exécution
- Précédence et causalité sont souvent assimilées car si on ne connaît pas le contenu exact d'une exécution alors nous sommes obligés de faire l'hypothèse d'une relation de causalité.

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Proposition de Lamport

Proposition

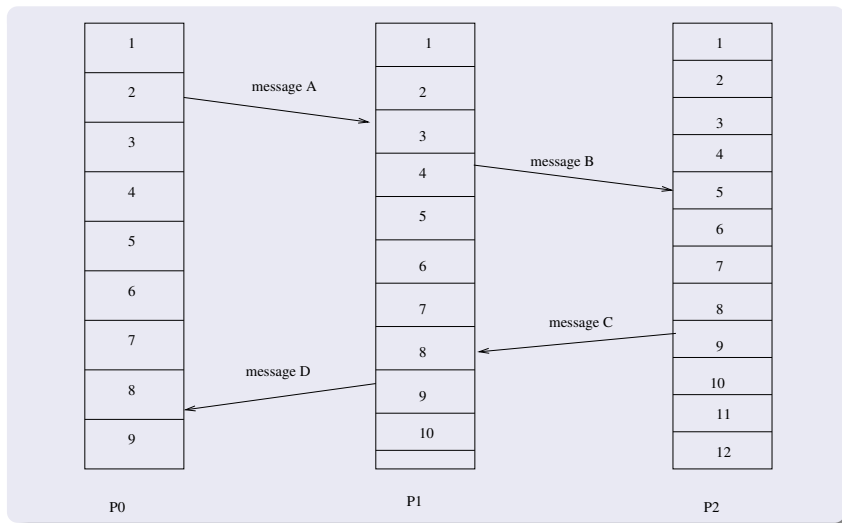
- Gestion d'une horloge logique ou estampille par processus
- Entre deux évènements locaux, ou suite à un évènement local, l'estampille est augmentée de 1
- Suite à un envoi, l'estampille est augmentée de un top
- Suite à une réception, l'estampille est fixée à :
$$\text{Max}(H_{\text{locale}}, H_{\text{re cue}}) + 1$$

Comportement sans l'algorithme de Lamport

Exemple

- Processus s'exécutent sur des machines différentes
- Chaque machine i a :
 - sa propre horloge H_i
 - sa propre fréquence d'horloge
- La fréquence d'horloge de P_0 plus basse que celle de P_1 , elle même plus basse que celle de P_2 .

Comportement sans l'algorithme de Lamport

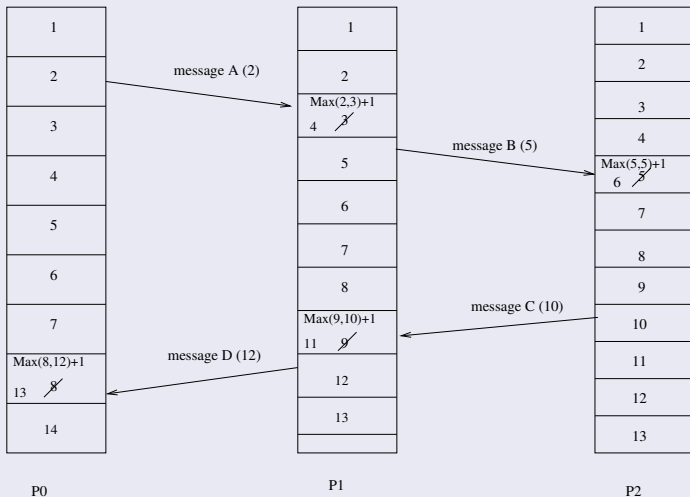


Proposition de Lamport

Exemple

- Comportement avec l'algorithme de Lamport
 - chaque message transporte l'estampille d'émission du point de vue de l'émetteur
 - à la réception d'un message, le récepteur synchronise son horloge (au moins horloge reçue + 1)

Exemple d'estampille de Lamport



Proposition de Lamport

Conclusion

L'ordonnancement proposé par Lamport permet d'attribuer une *horloge logique* ou *estampille* à tous les événements d'un système distribué.

Cette horloge logique est conforme aux conditions suivantes :

- Si un événement E_1 précède un événement E_2 dans le même processus, alors $H(E_1) < H(E_2)$.
- Si E_1 et E_2 représentent respectivement l'envoi et la réception d'un message, alors $H(E_1) < H(E_2)$.
- pour tous événements, E_1 et E_2 , $H(E_1)$ est différente de $H(E_2)$

Les horloges de Lamport permettent d'ordonner les événements d'un système distribué

Estampilles et ordre total

Rappel : ordre total

- Tous les événements peuvent être comparés deux à deux par la relation d'ordre R
- $\forall E_1, E_2 : E_1 R E_2 \vee E_2 R E_1$

Horloges à estampilles

- La comparaison entre horloge est possible si on a une relation de précedence : $E_1 \rightarrow E_2$ alors $H(E_1) < H(E_2)$
- Deux, ou plus, événements concurrents peuvent avoir la même valeur d'horloge
- Ne peut servir de support à un problème de décision

Estampilles et ordre total

Ordre total pour horloges à estampilles

- Il est nécessaire d'avoir un mécanisme supplémentaire pour mettre en place un ordre total
- Chaque processus dispose d'un identificateur unique
- Les identificateurs de processus sont associés aux estampilles qui deviennent des couples : (H, P_i) , H est l'horloge locale, P_i l'identificateur du processus
- En cas d'égalité d'estampille, l'horloge la plus petite est celle de site ayant le plus petit identificateur.
- Pour deux événements E_1 et E_2 , d'estampille H_1 et H_2 ayant lieu sur les processus P_i et P_j on a :
$$H(E_1) < H(E_2) \Leftrightarrow H_1 < H_2 \vee (H_1 = H_2 \wedge i < j)$$

Utilisation des estampilles

En algorithmique distribuée

- Algorithmes de mise en œuvre de files d'attente virtuelles réparties :
 - exclusion mutuelle répartie
 - mise à jour de copies cohérentes
 - diffusion cohérente
- Détermination de l'événement le plus récent dans un ensemble d'événements :
 - gestion de la cohérence de caches
 - mise en œuvre de la mémoire virtuelle partagée
 - datation des transactions réparties
- Génération de noms uniques
- Synchronisation d'horloges physiques

Avantages des estampilles

Trois principaux avantages :

- Première datation répartie introduite
- Économiques : la datation est réalisée par un seul nombre et non par un vecteur
- Causalité des messages est respectée par remise à l'heure du récepteur

Limitation de la datation par estampille

- $E_1 - > E_2$ implique $H(E_1) < H(E_2)$ est une condition faible car on ne peut rien affirmer si $H(E_1) < H(E_2)$
 - soit les évènements E_1 et E_2 sont concurrents
 - soit les évènements E_1 et E_2 sont ordonnés
- Quand deux évènements sont concurrents, on ne peut rien conclure quant à leurs horloges logiques respectives
- Seule certitude :
Si $H(E_1) = H(E_2)$ Alors E_1 et E_2 sont concurrents.
- Les horloges logiques ne sont pas denses :
soient E_1 et E_2 tels que $H(E_1) < H(E_2)$, on ne peut pas savoir s'il existe E_3 tel que $E_1 - > E_3$ et/ou $E_3 - > E_2$

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - **Horloges vectorielles**
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Horloges vectorielles

Propriété de consistance

- Les estampilles ne conservent pas la trace des chemins de causalité : réciproque de la dépendance causale
- Non constantes = pas de lien avec la précédence
- Nécessité de conserver l'état de chaque estampille pour déterminer localement la précédence entre deux événements
- Vecteur d'horloges

A voir plus tard

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - **Horloges matricielles**
- 4 Synchronisation d'horloges physiques

Horloges matricielles

Snapshot

- Les horloges vectorielles ne permettent que de valider les dépendances entre événements.
- L'ordre de réception des messages peut avoir une incidence sur de déroulement de l'algorithme du processus P_i
- Mémorisation dans une matrice :
 - Ligne (i, i) : état du processus i
 - Ligne (i, j) : nombre de messages échangés entre P_i et P_k

Sommaire

- 1 Introduction
- 2 Relations entre événements
 - La relation de précédence
 - La relation de causalité
- 3 Les horloges logiques
 - Estampilles
 - Horloges vectorielles
 - Horloges matricielles
- 4 Synchronisation d'horloges physiques

Horloges physiques

Remarques

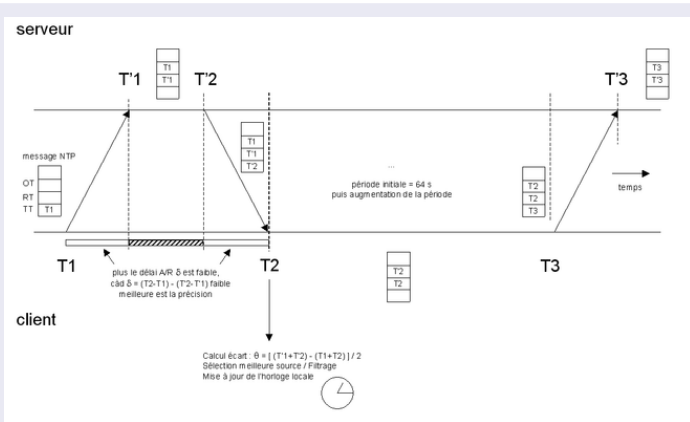
- Algorithme de Lamport propose une solution pour ordonner les évènements de façon non ambiguë
- Mais horloges logiques associées aux évènements ne correspondent pas forcément à la date réelle à laquelle ils se produisent
- Dans certains systèmes (trafic aérien, commerce, log, transactions) , le temps réel est important : un certain nombre d'algorithmes permettant de gérer ce problème existent (problématique non abordée dans ce cours)

Horloges physiques

Synchronisation

- Synchroniser les horloges physiques sur une horloge fixe, UCT (Universal Coordinated Time)
- Les horloges peuvent avoir des fréquences différentes
- Protocole de gestion du temps : Network Time Protocol (NTP)
- Repose sur une hiérarchie :
 - Serveurs racines (primaires) qui se synchronisent sur UCT
 - Serveurs intermédiaires et sauvegarde des serveurs primaires
 - Sous-réseaux de clients

NTP



Horloges logiques

Exercice

Utiliser les horloges logiques pour gérer l'accès au parking

Références

- Distributed Systems, Principle and Paradigms, A.Tanenbaum, Pearson Ed.
- M.Raynal, Une introduction aux principes des systèmes répartis (3 tomes), EyrollesEd. (1991)
- Cours de C.Kaiser du Cnam.