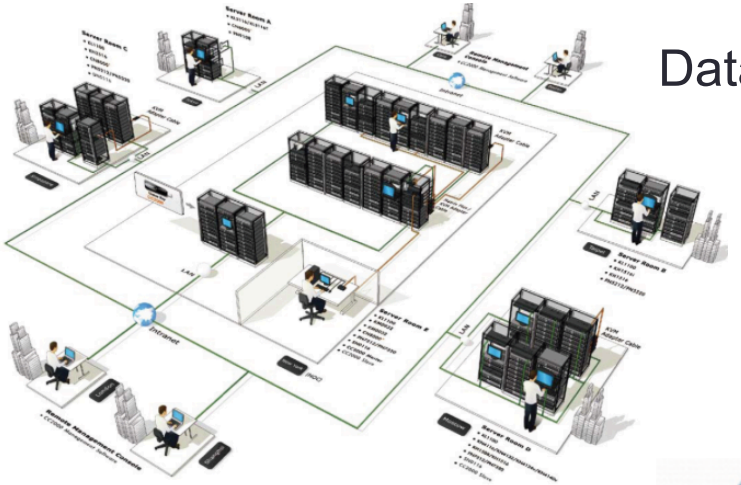
ALGORITHMIQUE DISTRIBUÉE
MIF12

INTRODUCTION & GESTION DU TEMPS

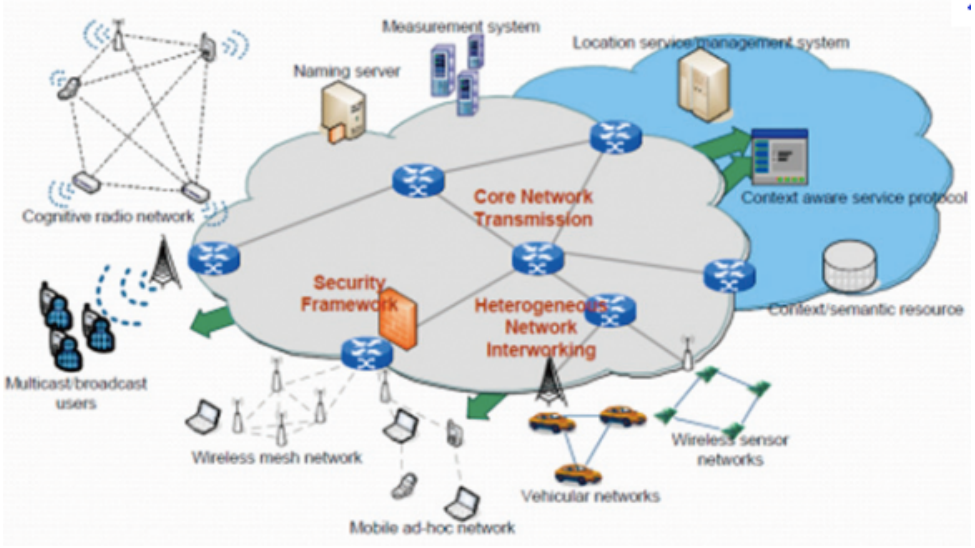
Isabelle GUERIN LASSOUS
perso.ens-lyon.fr/isabelle.guerin-lassous/index-AD.htm
isabelle.guerin-lassous@univ-lyon1.fr

Systèmes distribués

- Systèmes informatiques sont très souvent distribués



Data center



Réseaux mobiles

Pourquoi ?

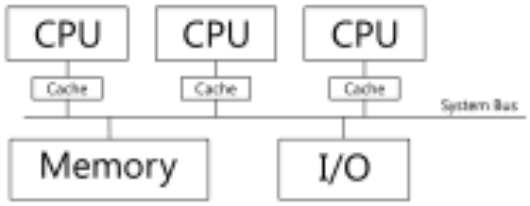
- Raisons géographique ou inhérentes au système
- Stockage
- Performances
- Fiabilité
- Disponibilité

Définition (rapide)

- Collection d'entités indépendantes qui coopèrent pour résoudre un problème qui ne pourrait pas être résolu individuellement
- Caractéristiques
  - Pas d'horloge physique commune
  - Pas de mémoire partagée
  - Autonomie
  - Hétérogénéité

Comparaison avec les systèmes parallèles

- Système multi-processeur
  - Accès direct à une mémoire partagée
    - Espace d'adressage commun



- Système multi-machine
  - Mémoire non partagée
  - Processeurs proches physiquement et souvent homogènes (hardware et logiciels)
- Objectifs des systèmes parallèles
  - Accélérer les temps de traitement
  - Traiter de gros volumes de données

Quelques problèmes

- Communications
  - Elles ne sont pas à coût nul !
- Coordination
  - Peut-on éviter le surcoût ?
- Tolérance aux pannes
- Localité
  - Est-ce toujours possible de privilégier la localité ?
- Synchronisme
  - A-t-on besoin de synchroniser les entités du système distribué ?
- Incertitude
  - Les entités du système ne savent pas où en sont les autres entités à un instant donné

Quels challenges d'un point de vue algorithmique ?

- Algorithmes de graphe distribués et dynamiques
- Algorithmes de coordination
  - Synchronisation, élection de leader, exclusion mutuelle
- Algorithmes de gestion des ressources et des données
  - Réplication, cohérence, cache, accès multiple aux ressources
- Algorithmes de tolérance aux pannes
- Algorithmes d'équilibrage de charge
- Algorithmes d'ordonnancement
- Algorithmes de routage

Contenu (prévisionnel) de ce cours

- Introduction – modèles
- Gestion du temps
- Algorithmes distribués sur les graphes
- Algorithmes d'élection de leader
  - TP
- Algorithmes de consensus et d'exclusion mutuelle
  - Nouvelle partie - Élise Jeanneau

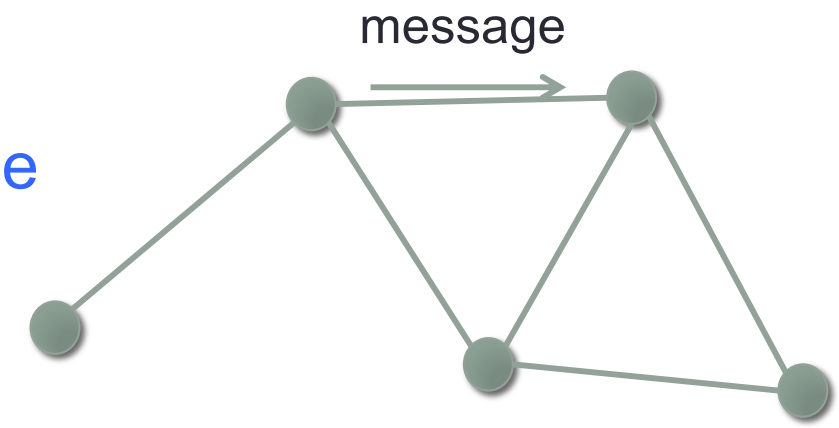
Fonctionnement du cours

- 12h CM – 9h TD – 9h TP
- TP
  - Programmation en langage D
- Note finale
  - Contrôle continu intégral
  - Idéalement : 2 CC (QCM – 25% chacun) + 1 ECA (40%) + 1 note TP (10%)
- Enseignants impliqués
  - TD : Élise Jeanneau, Laureline Pinault, Samir Si-Mohammed & moi
  - TP : Thomas Begin, Élise Jeanneau, Laureline Pinault, Samir Si-Mohammed & moi

MODÉLISATION

Système distribué

- Ensemble d'entités (indépendantes) communiquant via des messages
- Messages
  - Échangés sur un lien / canal de communication
    - lien physique : lien de communication réel
    - lien logique possible : chemin constitué de liens physiques
- Pas de
  - Mémoire globale
  - Horloge globale
- Peut être vu comme un graphe
  - Orienté / Non orienté
  - Nœuds – Arcs / Arêtes



Topologies particulières

- Anneaux unidirectionnels ou bidirectionnels
- Étoiles
- Cliques
- Hypercubes
- Arbres

- Sur une topologie connexe
  - On peut toujours définir / construire un arbre
  - Caractéristiques intéressantes

Quelques paramètres sur les graphes

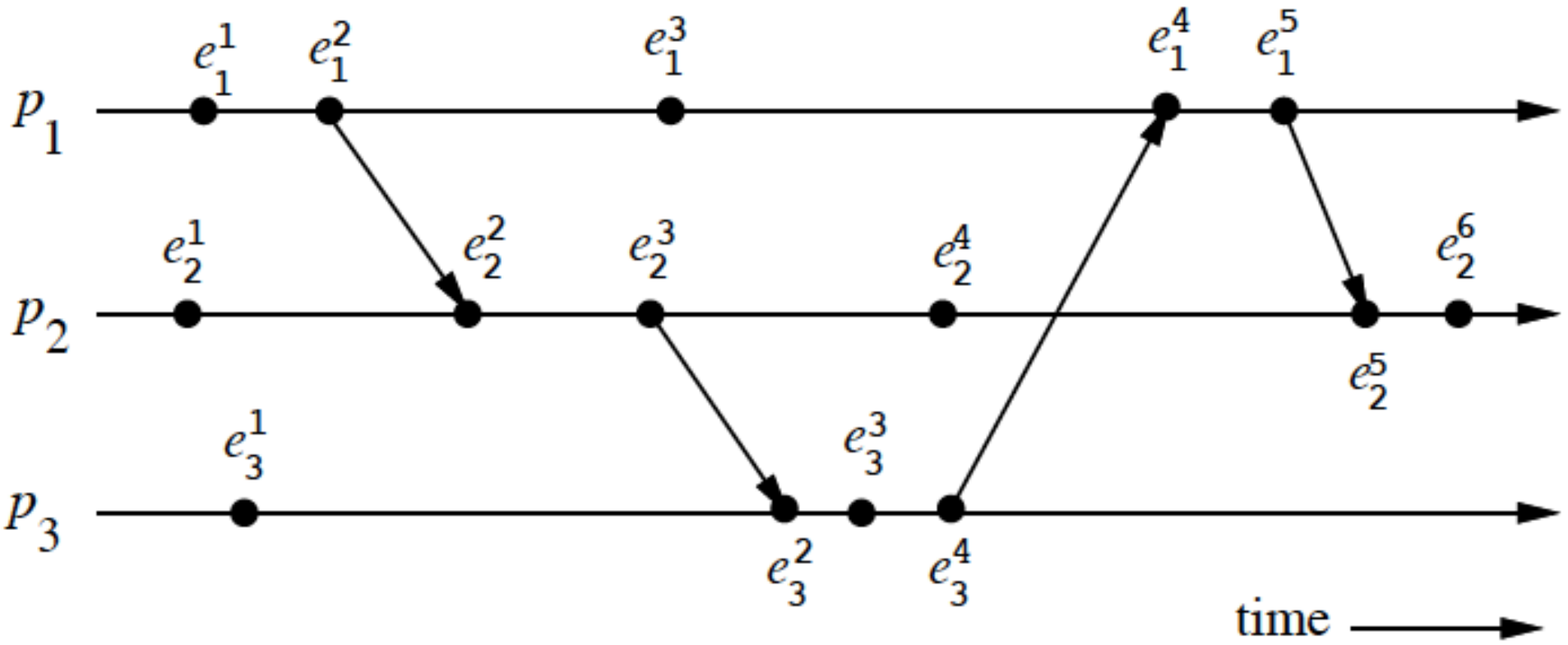
- Degré d'un nœud
  - Nombre de voisins
- Distance entre deux nœuds
  - Longueur du plus court chemin entre ces deux nœuds
- Diamètre d'un graphe
  - La plus longue distance entre deux nœuds du graphe

- Paramètres souvent utilisés pour le calcul de la complexité d'un algorithme distribué

Programme distribué

- Ensemble de processus asynchrones communiquant via des messages
- Processus exécutent des actions (événements)
  - Instruction / opération interne
  - Envoi d'un message
  - Réception d'un message
- **Nos hypothèses**
  - Les processus tournent sur des processeurs différents
  - Les actions sont atomiques
    - indivisibles

Exécution d'un programme distribué



Message

e_2^4 = 4e action/événement du processus p_2

Communications

- Modèle par passage de messages
- Ordre des messages peut être modifié
  - Possible déséquencement des messages
- Canal FIFO
  - ordre préservé sur un canal de communication
- Non-FIFO
  - ordre peut être perdu sur un canal de communication
- Fiabilité des communications
  - Message jamais reçu
  - Message reçu
    - Éventuellement suite à des retransmissions
    - En un temps borné ou
    - En un temps arbitraire mais fini
    - Pour certaines applications, le message peut arriver mais trop tard
    - Message peut arriver en erreur

Communications

- **Nos hypothèses**
  - Message toujours reçu
    - Mais il peut arriver après un temps très long
  - Message ne subit pas d'erreur
  - Canal FIFO ou non FIFO

NOTION ET GESTION DU TEMPS

Temps dans un système distribué

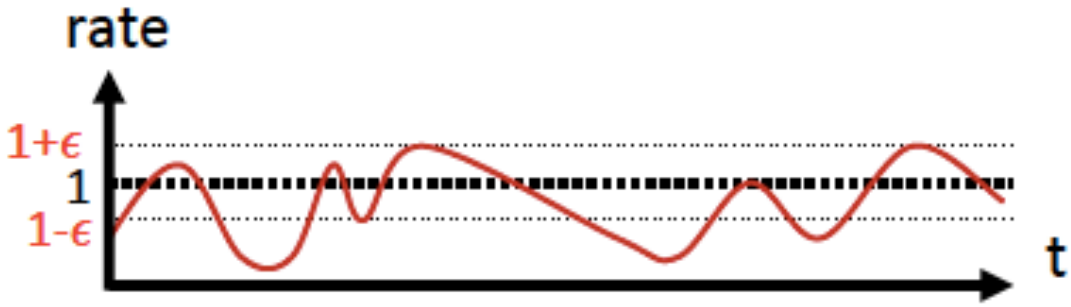
- Paramètre important
  - Pour beaucoup d'applications distribuées
    - Pour leur fonctionnement
    - Mais aussi pour pouvoir comprendre ce qu'il s'est passé au sein du système
- Rappel de nos hypothèses
  - Pas d'horloge commune dans les systèmes distribués

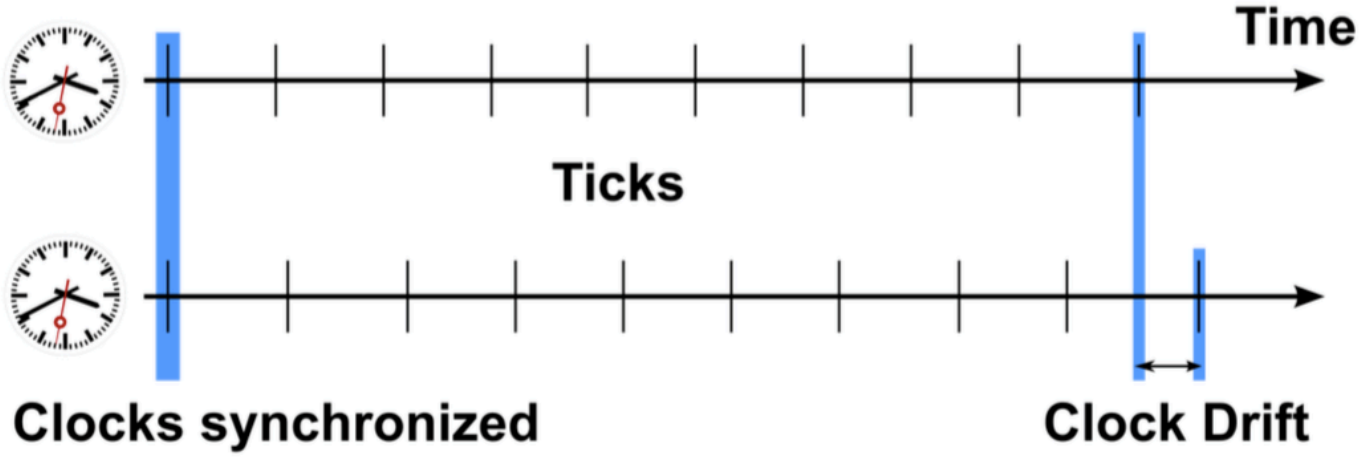
Horloges physiques

- 2 parties
  - Partie oscillante
  - Compteur
    - Compte les cycles de la partie oscillante
    - Fournit le temps à partir de ce comptage
- Pendule
  - Oscillant à une fréquence de 1 Hz
  - Un cycle par seconde
- Horloge à quartz
  - Quartz (au sein d'un circuit électrique) vibre / oscille à une fréquence de 32768 Hz
- Horloge atomique
  - Basée sur les transitions orbitales des photons au sein d'un atome
  - Pour l'atome de Césium 133
    - 1 s = 9 192 631 770 transitions
    - Adopté comme mesure du SI (Système International) en 1967
  - Coûteux

Dérive d'horloge

- Fluctuation aléatoire en fréquences
  - Températures
  - Altitude
  - Mobilité
- Exprimée en ppm
  - Partie par million
  - Horloge à quartz
    - 20 ppm ?





Gestion du temps dans un système distribué

- Horloges physiques des entités peuvent avoir des temps différents
- Deux possibilités
  - Temps commun non nécessaire
    - Il suffit (seulement) de connaître l'ordre des actions
  - Temps commun nécessaire
    - Nécessite que les horloges des entités soient

Pause vidéo

- https://www.youtube.com/watch?v=Aaxw4zbULMs

Synchronisation des horloges

- Externe vs. Interne
  - Synchronisation externe
    - Les entités se synchronisent sur une horloge source externe
  - Synchronisation interne
    - Les entités se synchronisent entre elles sur un temps commun
- Continue vs. Ponctuelle
  - Les entités ont besoin d'être en permanence synchronisées
  - Les entités peuvent se synchroniser de temps en temps
- Online vs. Offline
  - Offline : reconstruction du temps quand on en a besoin
- Globale vs. Locale
- Précision vs. Convergence

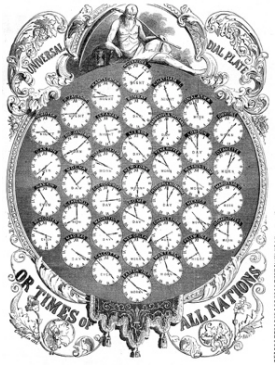
Échelle de temps

- On se synchronise sur quel temps ?
- Temps solaire
  - Basé sur la rotation de la terre
  - Jour solaire
    - Variable et non prédictible
    - Valeur moyenne (très légèrement) > 86400 s

La rotation de la Terre a accéléré en 2020, entraînant des journées de moins de 24 heures

- Temps Atomique International - TAI
  - Moyenne (pondérée) des temps obtenus sur plus de 500 horloges atomiques sur plus de 70 sites/laboratoires
  - Pondération
    - Fonction de la stabilité et de la précision des horloges
    - Comparaison des temps entre les horloges

Échelle de temps

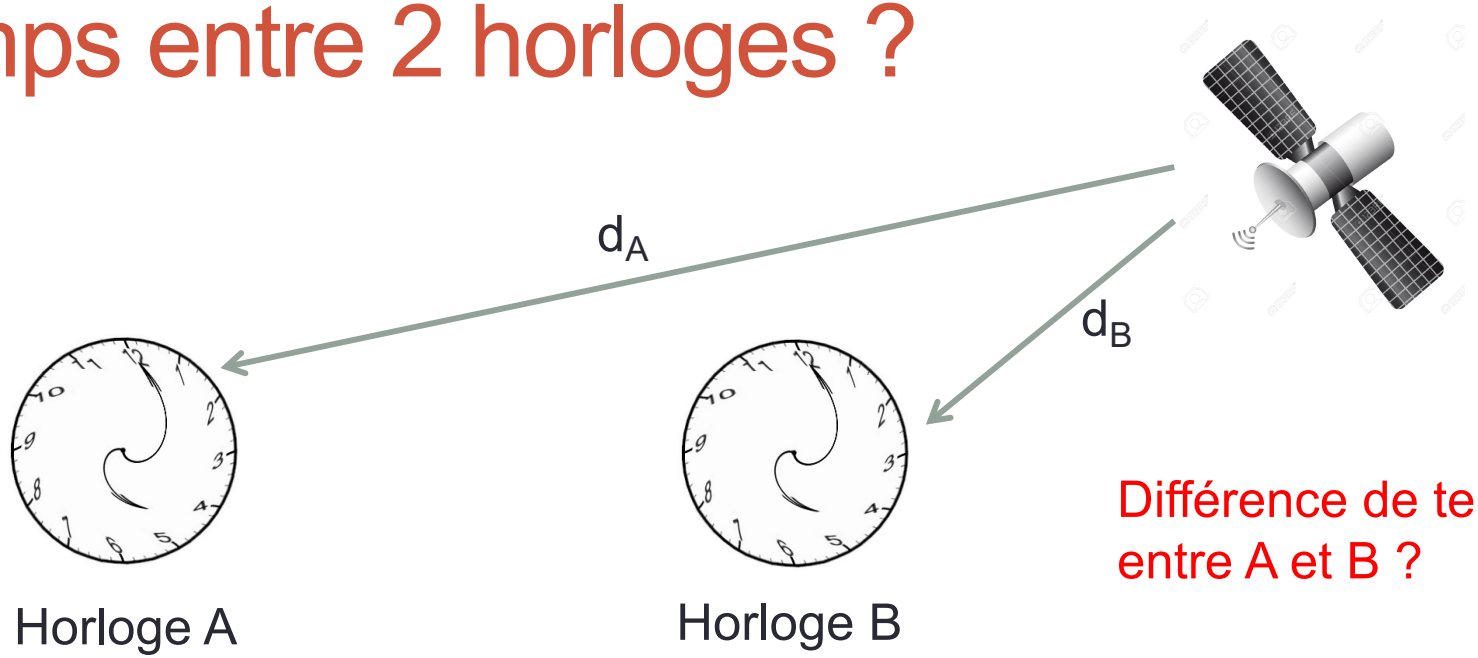


- Temps atomique indépendant du jour solaire
- Temps Universel Coordonné – UTC
  - Basé sur le temps atomique et le jour solaire
    - Stabilité du TAI et |UTC – temps solaire| ≤ 0,9 s
    - Ajustement du temps atomique sur le jour solaire avec l'ajout ou le retrait d'1s
      - Seconde intercalaire (leap second)
  - Actuellement, UTC en retard de 37 s sur TAI
  - Fourni par le Bureau International des Poids et des Mesures
    - ~ 50 centres font une estimation du temps UTC en temps réel
  - Erreur de ± 1 s / 10 millions d'années
  - Utilisé par plusieurs applications réseaux (e.g. WWW, NTP)

Global Positioning System

- GPS
  - Utilisation du temps pour localiser
- Satellites déployés autour de la terre
  - Possèdent chacun une horloge atomique
    - Synchronisée avec des horloges au sol
- Sur un point donné du globe
  - Au moins 4 satellites sont en vue directe
- Récepteur GPS
  - A une horloge précise mais pas une horloge atomique
  - Signaux reçus des satellites permettent de déterminer la position du récepteur et le temps GPS
    - => cf TD1
- Temps GPS
  - A été synchronisé sur le temps UTC en 1980
  - Mais n'a pas pris ensuite en compte les secondes intercalaires
  - En avance de 18 s sur UTC
  - Précision inférieure à 40 ns (dans la plupart des cas)

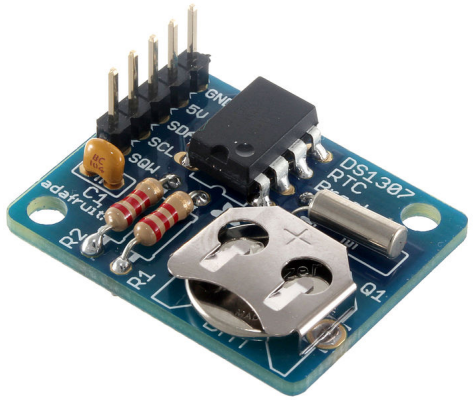
Comment connaître la différence de temps entre 2 horloges ?



En utilisant le système GPS

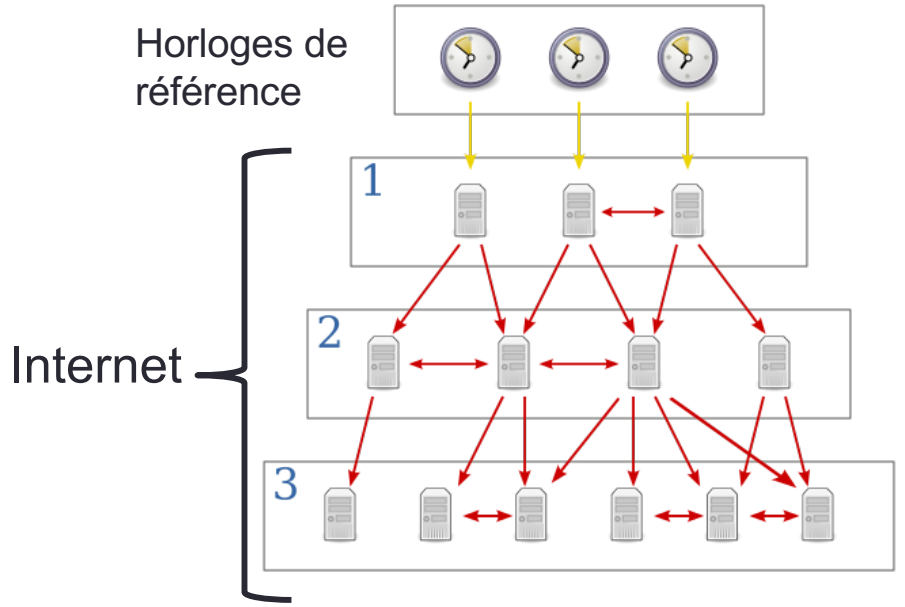
Horloges dans les ordinateurs

- Real-time clock
  - Ne pas confondre avec l'horloge hardware
  - Oscillateur à 32 768 Hz
  - Circuit avec une batterie propre
    - Ne repart pas de zéro quand l'ordinateur s'arrête
- High Precision Event Timer
  - HPET
  - Oscillateur entre 10 et 100 MHz
    - Résolution jusqu'à 10 ns
  - Tâches
    - Ordonnancement de threads
    - Synchro de flux multimédia
    - Retard à la lecture des applications de streaming
  - Souvent intégré au southbridge de la carte mère

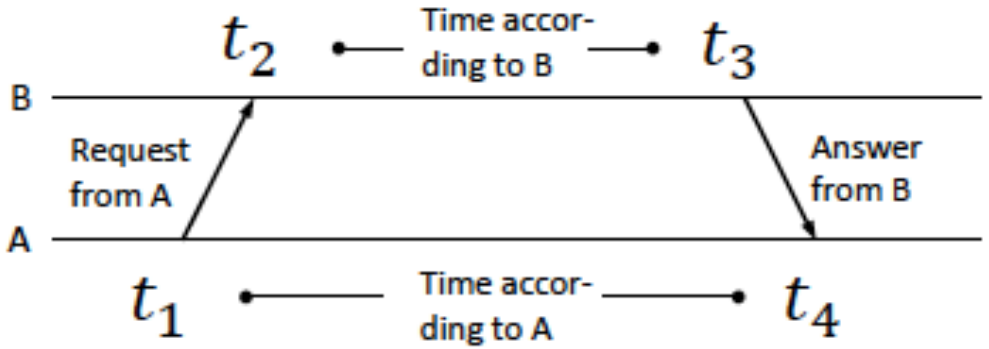


Synchronisation dans les réseaux informatiques

- Synchronisation en utilisant un réseau informatique
- Network Time Protocol (NTP)
  - Standard IETF – v4 en 2010
  - Basé sur des horloges de référence UTC
- Architecture arborescente
  - Stratum
    - Serveurs primaires = stratum 1
    - Serveurs secondaires
- Diffusion de l'heure
  - Diffusion verticale en mode C/S
  - Diffusion latérale en mode symétrique
  - Diffusion locale en mode broadcast
- UDP
  - Port 123
- Timestamp NTP
  - Horloge de la machine
  - 64 bits
    - 32 bits indiquant le nombre de secondes écoulées depuis le 1er janvier 1900
    - 32 bits représentant la partie fractionnaire



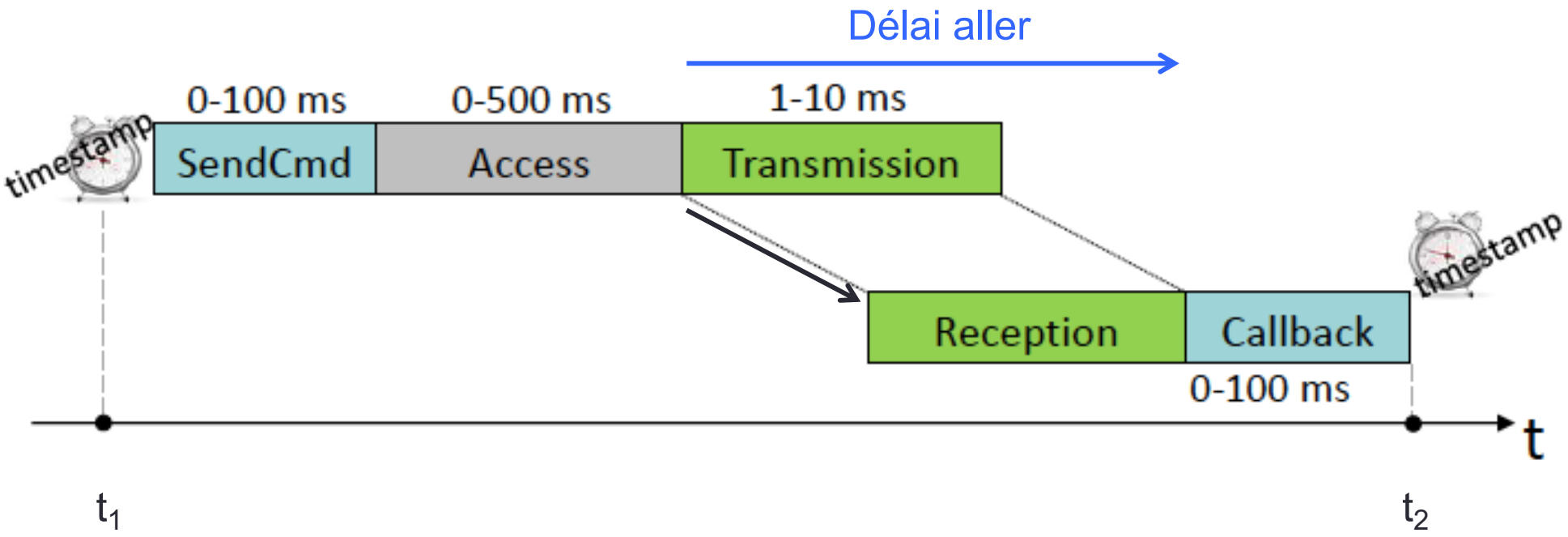
NTP : diffusion verticale en mode Client / Serveur



- A peut déterminer le décalage avec l'horloge serveur (B)
  - si on suppose que
    - les délais allers des messages sont symétriques (d(A,B) = d(B,A))
    - le décalage entre les 2 horloges est constant durant toute la durée de l'opération
- Plus le délai aller-retour est petit,
  - plus le décalage est déterminé avec précision
- Interrogation de plusieurs serveurs
  - Suite d'algorithmes pour sélectionner les meilleurs serveurs et pour diminuer l'impact du délai aller-retour

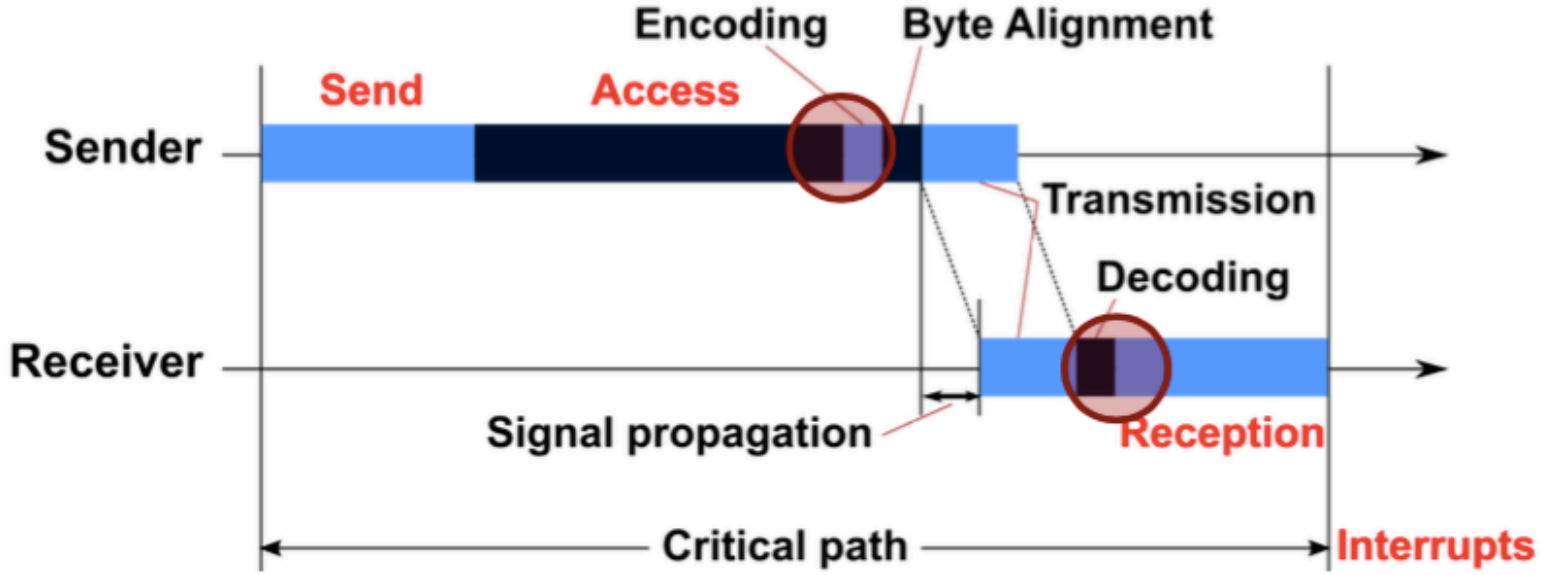
Problème de la gigue

- Messages subissent des délais variables
  - Délais déterministes
  - Délais non déterministes



Contournement de la gigue

- Mettre le timestamp le plus bas possible dans les couches
  - MAC timestamp
  - Hardware timestamp



- Nécessite des cartes réseaux coûteuses
- Technique efficace dans des réseaux locaux
  - Pas d'impact sur la variabilité des délais sur les chemins empruntés

Notion de temps logique

- On n'a pas toujours besoin d'une échelle de temps commune
  - Synchronisation des horloges non nécessaire
- On veut juste pouvoir ordonner les actions
  - Savoir si une action s'est passée avant une autre
  - Dépendance causale
- On notera « Action A s'est passée avant / précède l'action B »
  - A $\longrightarrow$ B

Dépendance causale : définition

$$\forall e_i^x,\ \forall e_j^y \in H,\ \ e_i^x \rightarrow e_j^y \quad \Leftrightarrow \quad \begin{cases} e_i^x \rightarrow_i e_j^y \ \ i.e., (i = j) \wedge (x < y) \\ or \\ e_i^x \rightarrow_{msg} e_j^y \\ or \\ \exists e_k^z \in H : e_i^x \rightarrow e_k^z \ \wedge \ e_k^z \rightarrow e_j^y \end{cases}$$

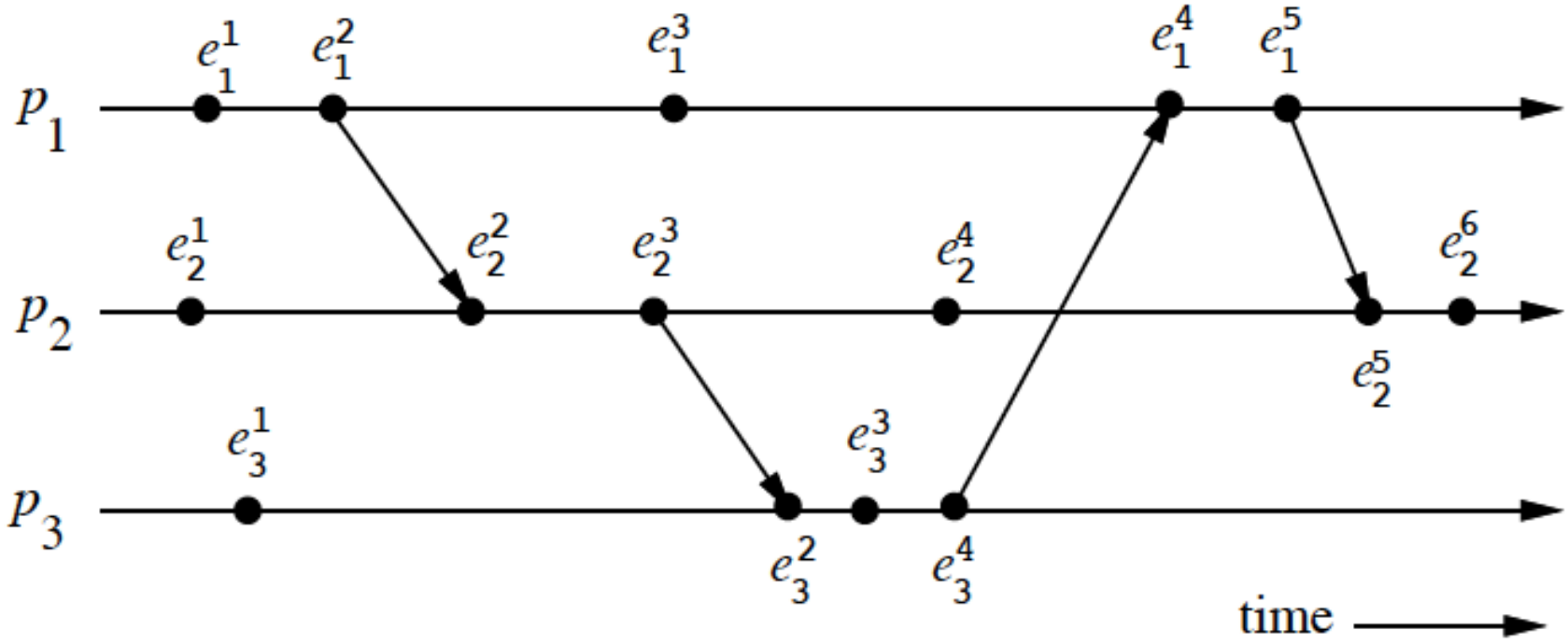
Avec H = ensemble des actions produites par le programme distribué

e_j^y = événement / action numéro y sur le processus j

S'il n'existe pas de relation de dépendance causale entre 2 actions alors les 2 actions sont dites concurrentes (noté par ||)

Définit un ordre causal, qui est un ordre partiel

Dépendance causale : exemple



Horloges logiques

- Trouver un système de datation des événements qui respectent l'ordre causal d'un système distribué
- Système d'horloges logiques
  - Échelle de temps T
  - Ensemble des actions H
  - Horloge logique C
    - Fonction qui à une action associe un temps
  - Vérifie la propriété de cohérence d'horloges
    - Si $A \longrightarrow B$, alors $C(A) < C(B)$
- Certaines horloges logiques peuvent vérifier la propriété de cohérence forte
  - $A \longrightarrow B$ si et seulement si $C(A) < C(B)$

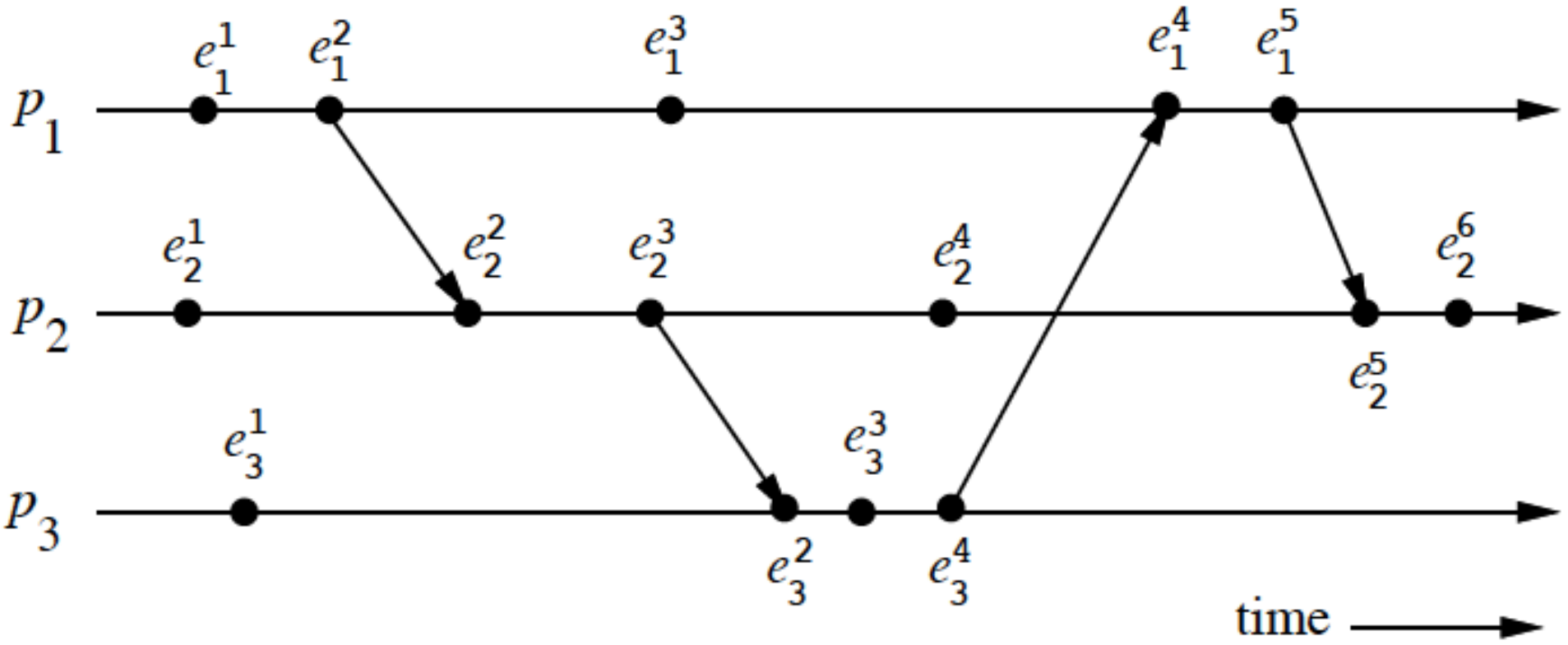
Implémentation du temps logique

- Construire un système d'horloges logiques
- Chaque processus maintient
  - Une horloge logique locale
    - pour mesurer son avancement
  - Une horloge logique globale
    - qui représente une vue locale du temps logique global
- Protocole de mise à jour de ces 2 variables reposent sur 2 règles
  - R1 : La mise à jour de l'horloge logique locale est réalisée au moment d'une action
  - R2 : La mise à jour de l'horloge logique globale dépend d'informations véhiculées dans les messages reçus
- Il existe plusieurs systèmes d'horloges logiques

Horloge scalaire

- Horloge de Lamport
- Une seule variable pour représenter les 2 horloges
  - C_i pour le processus p_i
  - $C_i = 0$ à l'initialisation
- Lors d'une action interne
  - $C_i := C_i + 1$
- À l'envoi d'un message
  - $C_i := C_i + 1$
  - Ci est incluse dans le message à envoyer
- À la réception d'un message
  - Récupération du message
  - $C_i := \max(C_i, C_{messg}) + 1$
    - C_{messg} = horloge reçue dans le message

Horloge scalaire : exemple



Indiquer l'évolution des horloges (initialement $C_i = 0$)

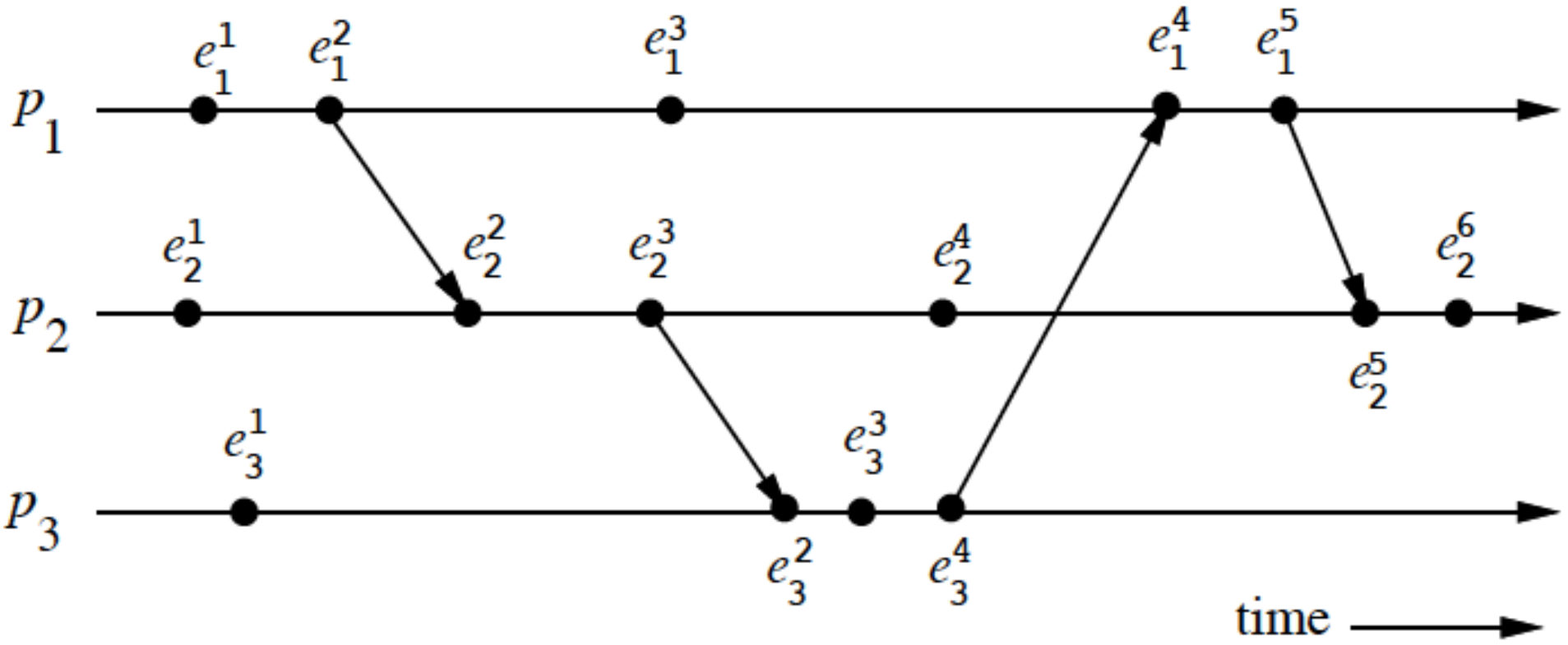
Horloge scalaire : propriétés

- Cohérence d'horloge ?
- Cohérence forte ?
- Comptage d'événements

Horloge vectorielle

- Chaque processus p_i
  - Possède un vecteur $vt_i[]$ à n entrées (si n est le nombre de processus)
  - $vt_i[i]$ = horloge logique locale du processus i
  - $vt_i[j]$ = dernière connaissance, par le processus i, de l'horloge logique locale du processus j
  - $vt_i[] = [0 ; 0 ; ... ; 0]$ à l'initialisation
- Lors d'une action interne
  - $vt_i[i] := vt_i[i] + 1$
- À l'envoi d'un message
  - $vt_i[i] := vt_i[i] + 1$
  - $vt_i[]$ est inclus dans le message à envoyer
- À la réception d'un message avec vt[] horloge reçue dans le message
  - $vt_i[k] := \max(vt_i[k], vt[k])$ pour tout k entre 1 et n et k <> i
  - $vt_i[i] := vt_i[i] + 1$

Horloge vectorielle : exemple



Indiquer l'évolution des horloges (initialement les vecteurs sont nuls)

Horloge vectorielle : propriétés

- Isomorphisme
  - Cohérence d'horloge
  - Les horloges de 2 événements concurrents ne peuvent pas être comparées
- Cohérence forte
- Comptage d'évènements
- Comparaison scalaire – vectorielle
  - Scalaire : horloge locale C_i donne une connaissance globale
  - Vectorielle
    - $vt_i[j]$ indique ce que le processus i connait du processus j
    - Nécessite plus d'informations à stocker et à envoyer
      - On ne peut pas faire moins localement pour avoir une cohérence forte

Ce qu'il faut retenir

- Définition d'un système distribué et ses principales caractéristiques
- Principales caractéristiques d'un programme distribué
- Principe de fonctionnement d'une horloge et la notion de dérive d'horloge
- Notion d'échelle de temps et les différents référentiels
- Horloges dans les ordinateurs
- Principes généraux du protocole NTP
- Problème de la gigue dans la synchronisation des réseaux informatiques
- MAC/Hardware timestamping
- Notion d'horloges logiques et les implémentations scalaire et vectorielle