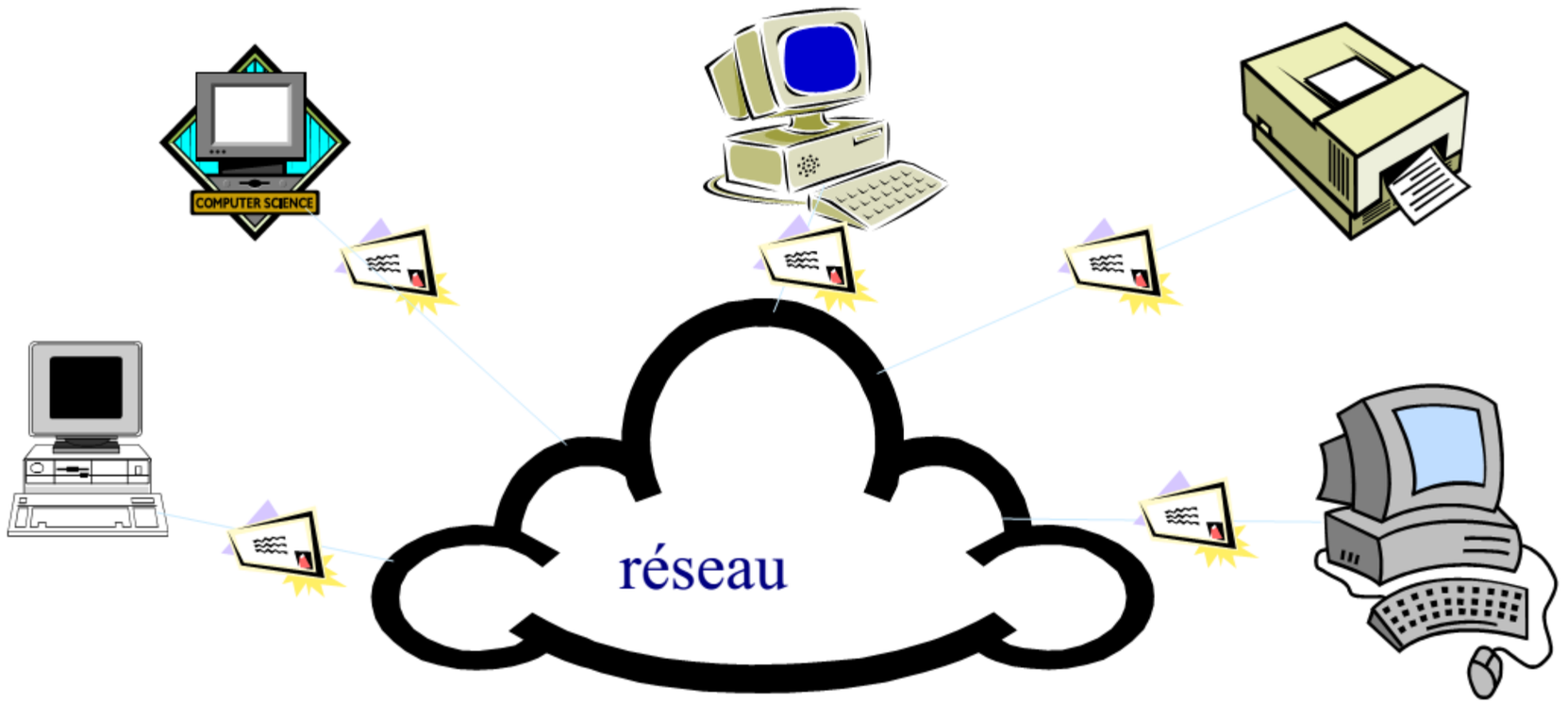
# SYSTÈMES DISTRIBUÉS

**Présenté par:**
I. Ait Abderrahim
i.aitabderrahim@univ-dbkm.dz

# GÉNÉRALITÉ

# Introduction




Machines, personnes, processus, «agents»… localisés à des endroits différents.

# Les Systèmes Distribués : définition

**Définition [Tanenbaum]:**

Un ensemble d'ordinateurs indépendants qui apparaît à un utilisateur comme un système unique et cohérent

- Les machines sont autonomes
- Les utilisateurs ont l'impression d'utiliser un seul système.

# Les Systèmes Distribués : définition

- Un système distribué est un ensemble d'entités autonomes de calcul (ordinateurs, PDA, processeurs, processus, processus léger etc.) interconnectées et qui peuvent communiquer.

- Exemples:
- – réseau physique de machines
- – Un logiciel avec plusieurs processus sur une même machine.

# Pourquoi des systèmes distribués?

- Aspects économiques (rapport prix/performance)
- Adaptation de la structure d'un système à celle des applications (géographique ou fonctionnelle)
- Besoin d'intégration (applications existantes)
- Besoin de communication et de partage d'information
- Réalisation de systèmes à haute disponibilité
- Partage de ressources (programmes, données, services)
- Réalisation de systèmes à grande capacité d'évolution

# Avantages des Systèmes distribués?

- Utiliser et partager des ressources distantes
  - Système de fichiers : utiliser ses fichiers à partir de n'importe quelle machine
  - Imprimante : partagée entre toutes les machines
- Optimiser l'utilisation des ressources disponibles
  - Calculs scientifiques distribués sur un ensemble de machines
- Système plus robuste
  - Duplication pour fiabilité : deux serveurs de fichiers dupliqués, avec sauvegarde
  - Plusieurs éléments identiques pour résister à la montée en charge ...

# Inconvénients des Systèmes distribués?

- Si problème au niveau du réseau
  - Le système marche mal ou plus du tout
- Bien souvent, un élément est central au fonctionnement du système : serveur
  - Si serveur plante : plus rien ne fonctionne
  - Goulet potentiel d'étranglement si débit d'information très important
  - Sans élément central
    - Gestion du système totalement décentralisée et distribué
    - Nécessite la mise en place d'algorithmes +/- complexes

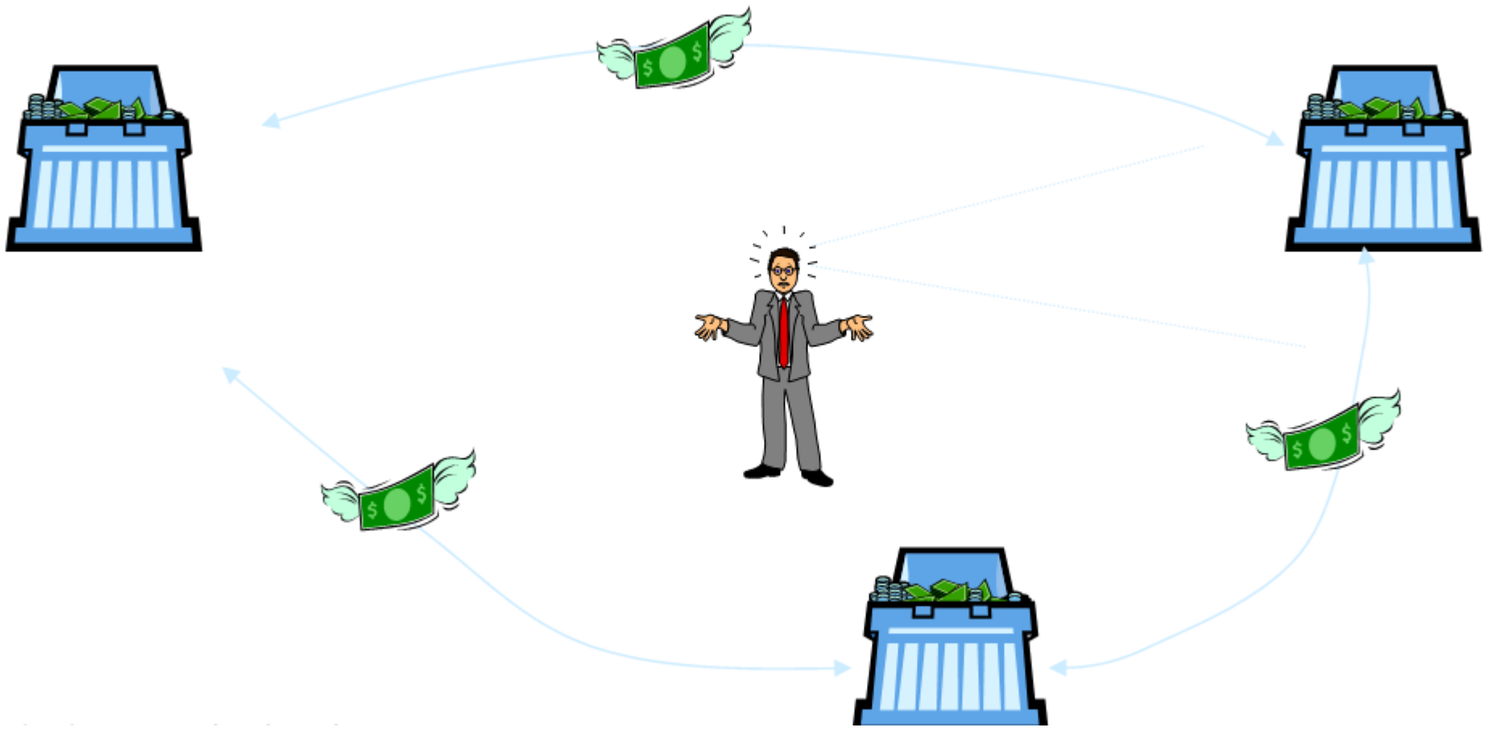
# Inconvénients des Systèmes distribués?

- Pas d’horloge globale
- Pas d'état global immédiat accessible à un site
- Fiabilité relative : distribution permet d'introduire de la tolérance aux fautes
- Sécurité relative : architecture distribuée plus difficile à protéger (plusieurs p oints d'accès aux ressources, évolution dynamique de l'architecture)

# Exmples de SD

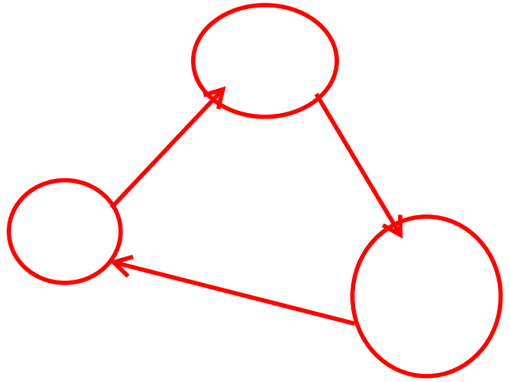
- **Serveur de fichiers**
  - Accès aux fichiers de l'utilisateur quelque soit la machine utilisé
  - Machines du réseau
    - Clients : scinfeXXX
    - Un serveur de fichier
  - Sur toutes les machines : /home/Med, est le « home directory » de l'utilisateur *Med*
  - *Physiquement* : fichiers se trouvent uniquement sur le serveur
  - *Virtuellement* : accès à ces fichiers à partir de n'importe quelle machine cliente en faisant « ***croire*** » que ces fichiers sont stockés localement

# Exmples de SD



# Ce qui est distribué et les problèmes associé

- ***Distribution des données*** : duplication et/ou partitionnement
  - problème de cohérence des données à un instant.
  - problème d'accès (exclusion mutuelle répartie).
  - problème de définition optimale des répartitions.



- ***Distribution des calculs*** :
  - problème de blocage « inter-blocage ».
  - problème de terminaison d'un calcul.
  - Problème de coordination (rendez-vous).

# Ce qui est distribué et les problèmes associé

- ***Distribution des algorithmes de mise en œuvre de la distribution***
  - gestion de l'exclusion mutuelle sans mémoire commune.
  - gestion de la cohérence des données dans le temps.
  - détection de la terminaison et de l'interblocage.

# Les différents types de systèmes distribués

- **Les systèmes de calculs distribués**
  - ***Les clusters :***
    - Ensemble de nœuds identiques ayant le même OS connectés par un réseau local high speed (ex : InfiniBand)
    - Architecture caractérisée par une certaine homogénéité
    - Utilisés pour la programmation parallèle : un même programme est exécuté en parallèle sur plusieurs machines. (SPMD)

# Les différents types de systèmes distribués

- **Les systèmes de calculs distribués**
  - ***Les grilles :***
    - Système profite d'un ensemble de machines qui sont ou ne sont pas dédiées
    - Machines peuvent être éloignées
    - Architectures hétérogènes : au niveau hardware, système ....
    - Calcul, données, applications, etc.
    - Types de grilles :
      - Desktop grid
      - Resource grid
      - Service Grid

# Les différents types de systèmes distribués

- **Les systèmes d'information distribués**
  - ***Systèmes orientés données***
    - Les systèmes à base de transactions : systèmes de gestion de bases de données distribués.
    - Les EAI (Enterprise Application Integration) : coordination de différents systèmes
    - Systèmes coopératifs
    - Midleware : gestion coordonnées de systèmes
    - WEB Services : accès à des serveurs distants

# Les différents types de systèmes distribués

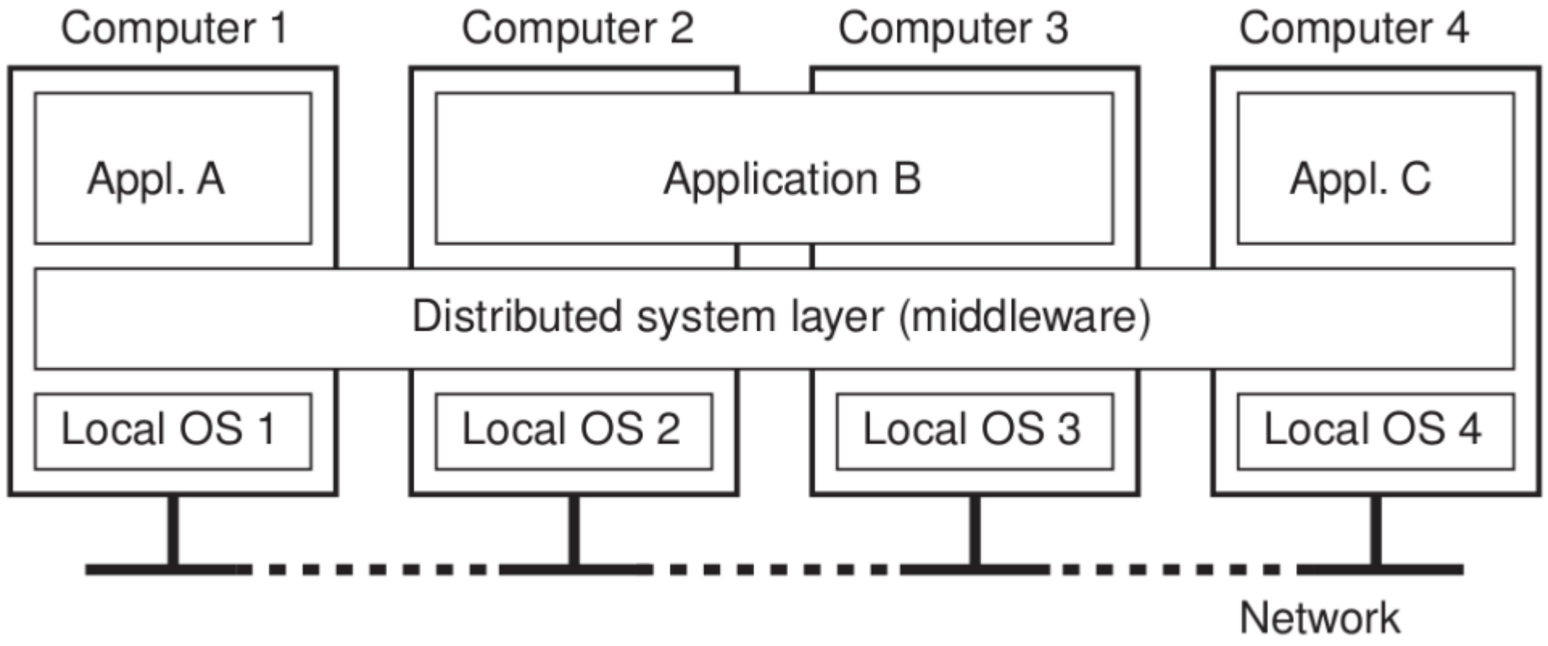
- **Les systèmes mobiles et « pervasifs »**
- ***Exemples***
  - Ordinateurs portables, téléphones, etc.
  - Robots, drônes, etc
  - Réseaux de capteurs : collecte dans un environnement, surveillance médicale, ...
  - Réseaux diffus : domotique, réseaux had-hoc, ...
- ***Instabilité***
  - Nœuds peuvent apparaître/disparaître
  - Pas toujours de connexion réseau
  - Mobilité des nœuds
  - Systèmes peu traités dans ce cours

# Les Middleware

- **Définition**
  Couche d'exécution distribuée entre les systèmes d'exploitation locaux et les applications

- **Objectifs**
  Offrir une interface permettant de :
  - rendre indépendantes les applications par rapport aux plates-formes
  - cacher la distribution aux utilisateurs : offrir une interface la plus proche possible du "centralisé"

# Les Middleware



# Les Middleware

- **Services facilitant l'exploitation des ressources**
  - Datation
  - Synchronisation
  - Nommage/Localisation
  - Courtier
  - Sécurité
  - Messaging
  - Transactions distribuées

    ...

# Distribué ou centralisé?

- **Centralisé souvent plus facile mais...**
  - Concentration en un p oint conduit à la surcharge
  - Fragilité face à la tolérance aux pannes
  - Pas toujours adapté aux besoins (ex : traitements locaux)

- **Distribué souvent plus résistant mais...**
  - Difficulté de mise en œuvre
  - N'évite pas tous les risques de surcharge (ex : diffusion systématique de messages)
  - Preuve complexe

# MODÉLISATION DES SYSTÈMES DISTRIBUÉS

# Modélisation conceptuelle des SD

- **Les éléments de modélisation**

  - Processus, sites, nœuds : notion de site logique (processus)
    Communication : liens, topologie, protocoles (point à point, diffusion), ...
    Connaissance partielle de chaque site logique

# Site logique ou processus

- **Définitions**
  Elément logiciel effectuant une tâche donnée.
  - *tâche* : ensemble d'instructions
  - *instruction* : correspond à un évènement local du processus
  - *évènement local* :
    - changement d'état du processeur,
    - émission de message
    - réception de message

# Site logique ou processus

- **Possède**
  - Un identifiant unique
  - Un état local
  - Une mémoire locale
  - Une horloge locale
- **Connaît (hypothèse)**
  - Les processus avec qui il p eut communiquer et leur nombre (voisins)
  - Les procédures de communication
  - Eventuellement l'état approximatif des autres processus

# La communication

- **Topologie**
  - La topologie d'un réseau décrit la façon dont les processus peuvent communiquer entre eux.

- **Exemple de topologies**
  - anneau uni ou bi-directionnel
  - réseau complet
  - Arbre
  - Grille

- **Protocoles de communication**
  - point à point
  - diffusion

# La communication Synchrone/Asynchrone

- **Communication synchrone**
  - Entre les processus $P_i$ et $P_j$ se fait par rendez-vous : le premier processus qui atteint un point de communication attend l'autre.
  - Synchronisation entre l'émetteur et le récepteur

- **Communication asynchrone**
  - Le processus $P_i$ envoie un message et continue son exécution sans se préoccuper si $P_j$ l'a reçu
  - Délais de communication arbitraires mais finis
  - On ne sait pas quand un message émis sera reçu
  - Cas particulier de la perte de message : algorithmes tolérants aux pannes

# Modèles de système distribué

- **Graphe de processus**
  - Représenté par un graphe de sommets et d'arc entre les sommets

- **Chronogramme**
  - Représentation graphique
  - Ordre des évènements
  - Déroulement chronologique sur chaque site
  - Représentation simple mais limitée

info.jsmi2022@univ-dbkm.dz

- **Evènement**
  - Emission de message
  - Réception de message
  - Evènement interne

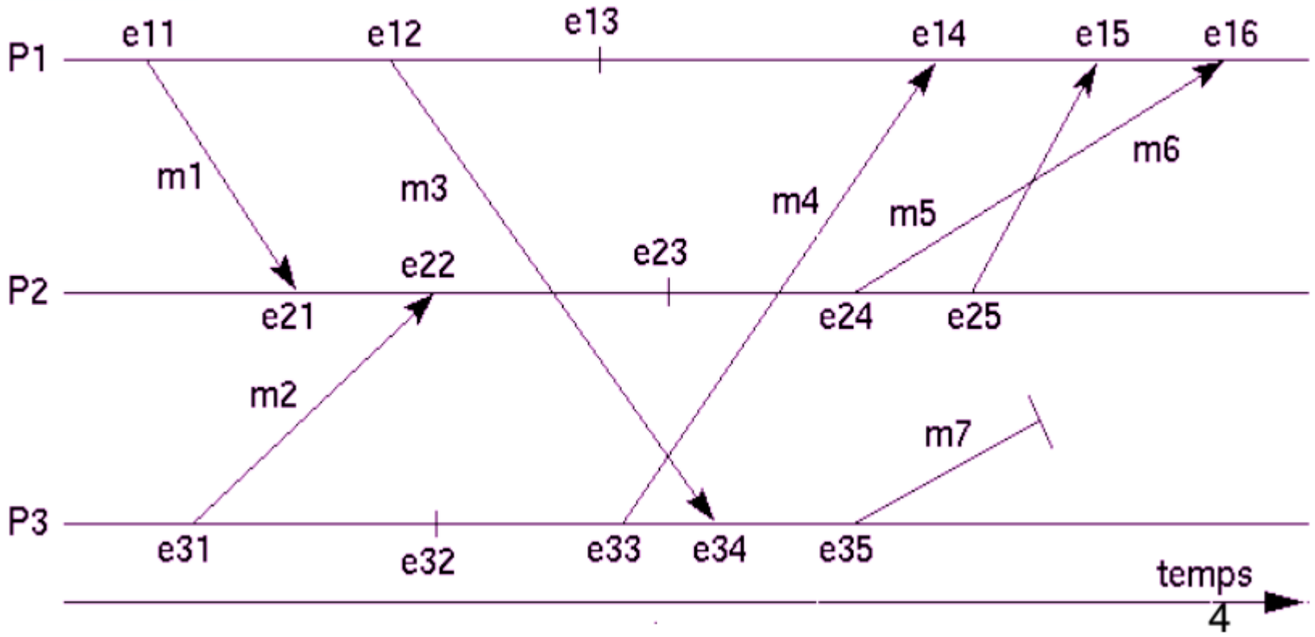
# GESTION DU TEMPS & ÉTAT GLOBAL DANS UN SYSTÈME DISTRIBUÉ

# Chronogramme

- Décrit l'ordonnancement temporel des événements des processus et des échanges de messages
- Chaque processus est représenté par une ligne
- Trois types d'événements signalés sur une ligne
  - Émission d'un message à destination d'un autre processus
  - Réception d'un message venant d'un autre processus
  - Événement interne dans l'évolution du processus
- Les messages échangés doivent respecter la topologie de liaison des processus via les canaux

# Chronogramme

- **Exemple**
  - Trois processus tous reliés entre-eux par des canaux
  - Temps de propagation des messages quelconques et
    possibilité de perte de message




**Exemples d'événements**

- ***Processus P1***
  - e1 1 : événement d'émission du message m1 à destination du
    processus P2
  - e1 3 : événement interne au processus
  - e1 4 : réception du message m4 venant du processus P3
- ***Processus P2*** : message m5 envoyé avant m6 mais m6 reçu avant m5
- ***Processus P3*** : le message m7 est perdu par le canal de communication

# Dépendance causale

- ***Relation de dépendance causale***
  - Il y a une dépendance causale entre 2 événements si un événement doit avoir lieu avant l'autre
  - Notation : e → e‘
    - *e* doit se dérouler avant *e‘*
  - Si e → e', alors une des trois conditions suivantes doit être vérifiée pour *e* et *e‘*
  - Si *e* et *e'* sont des événements d'un même processus, *e* précède localement *e‘*
  - Si *e* est l'émission d'un message, *e'* est la réception de ce message
  - Il existe un événement *f* tel que e → f et f → e'

# Dépendance causale

- ***Ordonnancement des événements***
  - Les dépendances causales définissent des ordres partiels pour des ensembles d'événements du système

# Temps logique: Dépendance causale

- **Sur exemple précédent**
  - Quelques dépendances causales autour de e12
  - Localement : e11 ➔ e12, e12 ➔ e13
  - Sur message : e12 ➔ e34
  - Par transitivité : e12 ➔ e35 (car e34 ➔ e35) et e11 ➔ e13
  - Dépendance causale entre e12 et e32 ?
  - A priori non : absence de dépendance causale
  - Des événements non liés causalement se déroulent en parallèle
  - Relation de parallélisme : ||
  - e || e' ➔ ¬((e ➔ e') v (e' ➔ e))
  - Parallélisme logique : ne signifie pas que les 2 événements se déroulent simultanément mais qu'il peuvent se dérouler dans n'importe quel ordre.

# Dépendance causale

- Ordonnancement des événements
  - Les dépendances causales définissent des ordres partiels pour des ensembles d'événements
- Buts d'une horloge logique (selon le type d'horloge)
  - Créer un ordre total global sur tous les événements de tous les processus
  - Déterminer si un événement a eu lieu avant un autre ou s'il n'y a pas de dépendances causales entre eux
  - Vérifier que des propriétés d'ordre sur l'arrivée de messages sont respectées
- Horloge logique
  - Fonction H(e) : associe une date à chaque événement
  - Respect des dépendances causales

# *Merci pour votre Attention*