N° d'ordre : 09/2003 - M/IN

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et Informatique
Département d'Informatique

THÈSE

Présentée pour l'obtention du grade

Magister

En : INFORMATIQUE

Spécialité : Programmation et Systèmes

par

BENZAID Chafika

Intitulée

Estampillage et Ordre Causal dans les Environnements Mobiles

Soutenue le 12/07/2003, devant le Jury composé de :

Mr. M. AHMED NACER	Maître de conférences, U.S.T.H.B.	Président
Mr. N. BADACHE	Maître de conférences, U.S.T.H.B.	Directeur de thèse
Mr. A. BELKHIR	Chargé de cours	Examinateur
M^{me} M. BOUKALA	Maître de conférences, U.S.T.H.B.	Examinateur

Remerciements

Je tiens à remercier le tout puissant de m'avoir donné le courage et la patience jusqu'à l'achèvement de travail.

J'exprime ma profonde reconnaissance et mes vifs remerciements à mon promoteur Mr. N. BADACHE pour m'avoir encadrer. Je le remercie également pour les lectures attentives de mes rapports et dont les critiques et suggestions ont été d'un grand apport pour la finalité de ce travail.

Je remercie Mme M. BOUKALA, Mr. A. BELKHIR, et Mr. M. AHMED NACER d'avoir accepté de juger ce travail.

Mes remerciements s'adressent aussi à ma famille, mes amis et tous ceux qui m'ont aidé et contribué de prés ou de loin par une quelconque façon, que ce soit par leur amitié, leurs conseils ou leur soutient moral ou matériel.

Table des matières

Table des figures

Estampillage et Ordre Causal dans les Environnements Mobiles

Soutenue par : BENZAID Chafika

Spécialité : Informatique

Directeur de thèse : BADACHE Nadjib, maître de conférences, USTHB

Résumé

Les évolutions techniques dans le développement des ordinateurs portables et le déploiement rapide de la technologie des réseaux sans fil fournissent la base pour un nouvel environnement de calcul, appelé *environnement mobile*.

Ces développements ont cependant un prix, les challenges techniques qui permettent de mettre en œuvre l'environnement de calcul mobile n'ont rien de trivial. Les nouvelles contraintes introduites par l'environnement mobile font qu'il est nécessaire de réviser les applications déjà conçues pour les systèmes distribués classiques. Dans notre travail, nous nous sommes intéressés à deux problèmes essentiels: *les mécanismes d'estampillage* et *l'ordre causal*.

Dans une première étape, nous avons étudié le problème d'estampillage dans l'environnement mobile, ce qui nous a permis de découvrir les nouvelles alternatives proposées pour pallier à l'inadéquation des mécanismes déjà proposés pour les systèmes distribués classiques. *Prakash* et *Singhal* ont proposé deux alternatives des horloges vectorielles, pour les systèmes mobiles, dites *les séquences de dépendance* (*dependency sequences*) et *les horloges hiérarchiques* (*hierarchical clocks*).

Et dans la deuxième étape, nous avons étudié le problème d'ordre causal des messages en environnement mobile qui est un concept approprié pour des applications qui nécessitent l'interaction entre plusieurs usagers depuis des localités différentes. Plusieurs protocoles d'ordre causal ont été proposés pour le cas des systèmes en environnement distribué. Cependant, les contributions dans ce domaine pour des systèmes en environnement mobile restent modestes. Quelques protocoles sont proposés et ils se basent généralement sur la reconstruction des protocoles déjà existants pour les adapter aux besoins d'un environnement mobile tels que la taille des informations de contrôle et les délais d'inhibition.

Ainsi, nous avons défini un nouveau protocole d'ordre causal des messages pour l'environnement mobile, *Mobi_Causal*. L'objectif majeur de notre protocole est d'étudier l'applicabilité des nouveaux mécanismes d'estampillage proposés pour les environnements mobiles (les séquences de dépendance et les horloges hiérarchiques) pour l'implémentation d'un protocole qui résout le problème d'ordre causal en environnement mobile. Notre protocole se caractérise par l'élimination des délais d'inhibition tout en maintenant une taille minimale de l'information de contrôle attachée aux messages.

SOMMAIRE

1 Introduction

Les évolutions techniques dans le développement des ordinateurs portables et le déploiement rapide de la technologie des réseaux sans fil fournissent la base pour un nouvel environnement de calcul, appelé *environnement mobile*.

Ces développements ont cependant un prix, les challenges techniques qui permettent de mettre en œuvre l'environnement de calcul mobile n'ont rien de trivial. Les nouvelles contraintes introduites par l'environnement mobile font qu'il est nécessaire de réviser les applications déjà conçues pour les systèmes distribués classiques. Dans notre travail, nous nous sommes intéressés à deux problèmes essentiels: *les mécanismes d'estampillage* et *l'ordre causal*. Ce résumé est organisé en quatre parties:

- La première partie introduit l'environnement mobile et les principaux concepts liés à ce nouvel environnement. Elle est axée essentiellement sur la définition de l'architecture des systèmes mobiles, la présentation de quelques caractéristiques de ces environnements à savoir les connexions sans fil et la mobilité, et l'impact de ces caractéristiques sur les algorithmes répartis.
- La deuxième étape présente le concept des mécanismes d'estampillage en environnement mobile. Elle commence d'abord par citer le concept du temps logique et les mécanismes d'estampillage proposés pour implémenter ce temps et par conséquent capturer les dépendances causales dans les systèmes distribués classiques: les horloges scalaires et les horloges vectorielles. Ensuite, elle présente l'inadéquation de ces mécanismes pour les systèmes mobiles et par conséquent, une présentation des alternatives existantes dans ce cadre: les séquences de dépendance et les horloges hiérarchiques.
- La troisième partie est consacrée à l'étude du problème d'ordre causal des messages en environnement mobile. Une définition de l'ordre causal est faite, suivie par une présentation des protocoles développés pour la mise en œuvre de l'ordre causal des messages dans les systèmes distribués classiques, et en dernier une présentation des protocoles proposés pour le cas des systèmes en environnement mobile.
- La dernière étape présente notre contribution pour la proposition d'un protocole d'ordre causal des messages en environnement mobile. L'objectif principal de ce protocole est d'étudier l'applicabilité des nouveaux mécanismes d'estampillage proposés pour les environnements mobiles (les séquences de dépendance et les horloges hiérarchiques) pour l'implémentation d'un protocole qui résout le problème d'ordre causal en environnement mobile. nous citons d'abord les structures de données utilisées dans notre protocole. Par la suite, nous présentons le module statique du protocole ainsi qu'un exemple illustratif. Le paragraphe suivant sera consacré à la preuve de correction de la solution; nous prouvons la sûreté et la vivacité de notre solution. Finalement, nous présentons le module *handoff* et nous terminons par une étude de la performance et de la complexité du protocole.

Nous terminons ce résumé par une conclusion sur le travail fait et les perspectives éventuelles à ce travail.

2 Caractéristiques des environnements mobiles

2.1 Modèle de système avec sites mobiles

Le terme "*mobile*" implique être capable á se déplacer tout en restant connecté au réseau [1]. Le modèle de système distribué comprenant des sites mobiles est composé de deux ensembles d'entités distincts : les sites fixes d'un réseau de communication filaire classique et des sites mobiles [2]. L'entité qui relie les deux parties, fixe et mobile, du réseau en transformant les signaux entre les deux médiums est appelée *station de support mobile (MSS – Mobile Support Station)*. Les *MSS*$_s$ sont munies d'interface de communication sans fil et la zone géographique couverte par une *MSS* est appelée *cellule*. Tout *site mobile (MH – Mobile Host)* est initialement enregistré dans une et une seule *MSS* appelée *station d'enregistrement (ou Home base)* [3]. Ceci n'empêche pas le libre déplacement de celui-ci entre les différentes cellules en qualité de 'visiteur'. Un site mobile ne peut communiquer avec d'autres entités du réseau que par l'intermédiaire d'une *MSS*.

Un site mobile peut se déplacer d'une cellule á une autre cellule. Dans un tel cas, la *MSS* de l'ancienne cellule cède les responsabilités de communication du *MH* á la *MSS* de la nouvelle cellule.

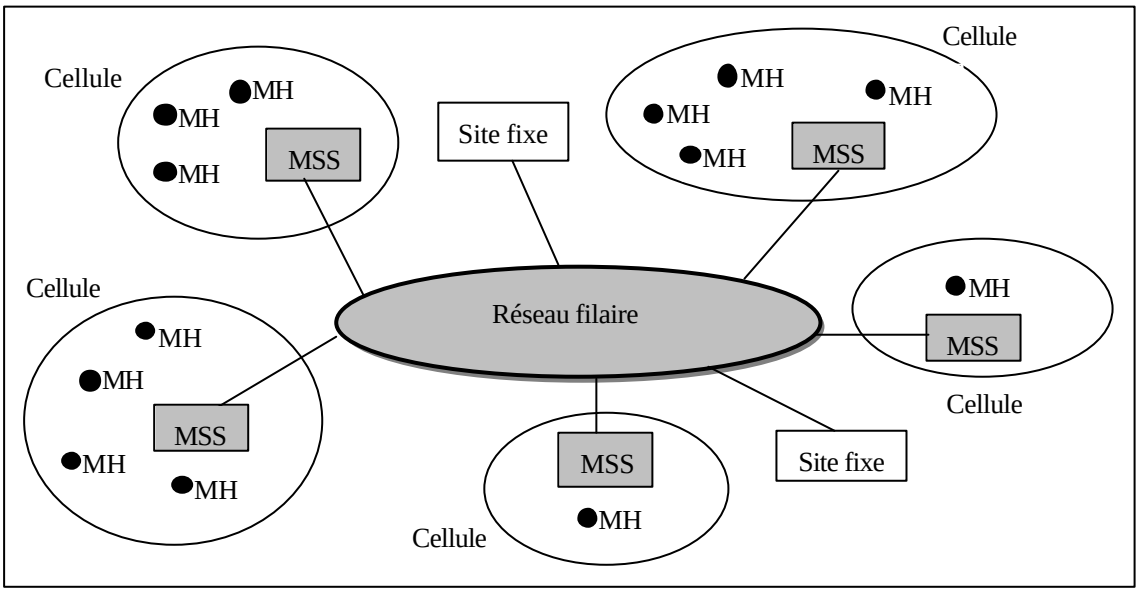


Figure 1 – *Architecture d'un système mobile*

Dans un système réparti sans ordinateur mobile, une machine ne peut travailler que dans deux modes différents, soit connectée au réseau, soit totalement déconnectée. Par contre en environnement mobile, il existe des degrés variés de déconnexion [4]. Le degré de déconnexion est relatif à la largeur de bande passante disponible allouée à la liaison sans fil.

Un site mobile dispose donc de davantage de modes de travail. Actuellement, nous dénombrons quatre modes de fonctionnement différents employés sur les sites mobiles: le mode connecté où le mobile dispose d'une connexion normale au réseau, le mode partiellement connecté qui est dû à une faible connexion, le mode veille qui est utilisé par

les mobiles pour préserver leurs ressources énergétiques, et finalement le mode déconnecté où le mobile n'est plus joignable par le réseau.

2.2 Les caractéristiques des environnements mobiles

L'environnement mobile offre aux utilisateurs la capacité de pouvoir se déplacer tout en restant connecté au réseau et d'être indépendants de toute localisation. Cependant de nouveaux problèmes peuvent apparaître causés par les nouvelles caractéristiques du système mobile. Ceci nécessite des mécanismes spécifiques pour s'adapter aux limitations qui existent, ainsi aux facteurs qui rentrent dans le jeu lors de la conception.

2.2.1 Les connexions sans fil

Les communications sans fil sont de moins bonne qualité que les communications filaires : les largeurs de bande sont plus réduites et les besoins très variables, les taux d'erreur plus élevés, les déconnexions fréquentes. Ces facteurs peuvent augmenter les temps de latence dus aux retransmissions, aux délais d'attente entre ces retransmissions, aux exécutions de protocole de contrôle d'erreur et aux courtes déconnexions. Dans un environnement sans fil, les connexions peuvent être perdues ou dégradées du fait de la mobilité et du passage de cellule en cellule, ou du fait d'interférences. A la différence des réseaux fixes, le nombre de machines connectées dans une cellule peut être très variable, et des concentrations importantes d'utilisateurs dans une même cellule peuvent surcharger le réseau [5].

L'utilisation de communications sans fil induit différents problèmes à savoir: les déconnexions fréquentes, la faible bande passante, la variabilité de la demande de bande passante. De plus, les problèmes d'hétérogénéité des réseaux et d'augmentation des risques du point de vue de la sécurité sont également liés à l'utilisation de médiums de communication sans fil.

2.2.2 La mobilité

Le but de l'informatique mobile est de fournir à l'utilisateur une complète indépendance de mouvement, tout en conservant une transparence à la localisation variable en fonction des besoins. Cette indépendance peut passer par différents concepts de mobilité, qu'il est éventuellement possible de mettre en œuvre conjointement. Trois concepts de mobilité peuvent être distingués : la mobilité du matériel qui est à l'heure actuelle la plus fréquemment citée dans les articles sur les systèmes mobiles [5, 6, 7], la mobilité des utilisateurs seuls [8] et la mobilité des interfaces des applications [9].

La mobilité du matériel et de l'utilisateur avec lui constitue pour le moment le problème le plus étudié en environnement mobile. Les problèmes posés sont d'ordre multiple et couvrent aussi bien le domaine des réseaux (adressage, protocoles de transport) que la gestion des données (accès, stockage, cohérence) ou les mécanismes systèmes (algorithmes de base, accès aux services et aux ressources).

La mobilité est un comportement ayant des implications aussi bien pour les stations qui ne bougent pas et qui constituent le réseau fixe, que pour les machines qui se déplacent avec les utilisateurs et qui constituent le réseau sans fil ou mobile [6].

Parmi les principaux problèmes introduits par la mobilité, nous citons l'adressage, la localisation des sites mobiles, la gestion des données…etc.

2.3 Algorithmes répartis en environnement mobile

L'intégration des ordinateurs mobiles avec les réseaux statiques existants introduit un nouvel ensemble de problèmes dans les systèmes répartis. Un site mobile peut se connecter au réseau à partir de différentes localisations à différents moments via un médium de communication sans fil, les mécanismes répartis (algorithmes, systèmes de gestion de fichiers, protocoles de diffusion de messages, etc.) ne peuvent donc plus supposer que les participants restent à une position fixe et universellement connue dans le réseau au cours du temps. Il devient donc nécessaire de revoir la structuration de ces mécanismes pour les adapter à l'informatique mobile.

L'étude des problèmes de synchronisation et de communication dans les systèmes répartis a été abordée pour des architectures comprenant uniquement des sites statiques. En absence de pannes de site ou de liaisons de communication, la structure topologique du réseau d'interconnexion ne varie pas. A l'inverse, dans les systèmes intégrant des sites mobiles, capables de se déplacer tout en conservant leur connexion au réseau active via des techniques de transmission sans fil, la connectique est en constante évolution. De nouveaux problèmes sont donc induits par la mobilité [10] :

1. Les mobiles se déplaçant, la connectique du réseau change. Par conséquent la structure logique, utilisée dans de nombreux algorithmes répartis, ne peut plus être définie statiquement ;
2. Pour délivrer un message à un site mobile, il est nécessaire que la destination soit d'abord localisée dans le réseau ;
3. La communication entre un site mobile et le reste du réseau se fait via un lien sans fil, possédant une faible largeur de bande par rapport aux liens filaires. De plus, la transmission et la réception des messages sur le lien sans fil entraînent la consommation d'énergie du site mobile ;
4. Les sites mobiles ont d'importantes contraintes en terme de puissance et d'énergie et opèrent souvent en mode veille ou complètement déconnecté du réseau. La déconnexion dans un environnement mobile est distinguée d'une panne : les déconnexions sont volontaires et donc, un site mobile peut informer le système d'une déconnexion imminente antérieurement à son occurrence.

Les algorithmes classiques (exclusion mutuelle, élection, terminaison, etc.), les protocoles de communication de groupe et de diffusion de messages ainsi que les systèmes de gestion de fichiers devront tenir compte de ces caractéristiques pour intégrer explicitement la mobilité, fournir des plates-formes à même de gérer efficacement les mobiles, et adaptées aux besoins des utilisateurs.

3 Mécanismes d'estampillage

3.1 Modèle d'un système distribué

Un système réparti est un ensemble fini de n processus séquentiels $\{P_1, P_2, \ldots, P_n\}$ communiquant et se synchronisant exclusivement par échange de messages. La communication entre les différents processus du calcul réparti est du type point à point et le délai de transfert d'un message entre tout couple de processus est considéré comme arbitraire mais fini. Chaque processus a une mémoire locale. Ni mémoire partagée ni

horloge globale sont disponibles. Les messages sont échangés à travers des canaux fiables non nécessairement FIFO. Les actions effectuées par un processus produisent des événements de trois types : les événements d'*émission* et de *réception* de messages (représentant la communication entre les processus) et les événements *internes* (causant un changement des variables locales).

3.2 Mécanismes d'estampillage dans les environnements distribués

Pour une compréhension exacte d'un programme distribué et de son exécution, il est important d'être capable de déterminer les relations temporelles (i.e. l'ordre des événements) entre les événements produits par les différents processus. La définition de l'ordre des événements sur un système distribué est plus problématique du fait de la difficulté de maintenir une notion de temps absolu cohérente due à l'absence d'une horloge système globale ou d'horloges locales parfaitement synchronisées.

Cependant une relation *cause et effet*, dite aussi *dépendance causale*, peut être établie entre les événements. En d'autres termes, deux événements sont contraints de se produire dans un certain ordre seulement si l'occurrence du premier peut affecter le résultat du second. Pour exprimer ces dépendances, Lamport [11] a proposé de définir pour ce type de systèmes une notion de temps logique permettant de comparer logiquement des événements : sur un site les événements locaux peuvent être ordonnés en se basant sur l'ordre de leur exécution et l'émission d'un message sur le site émetteur précède toujours sa réception sur le site récepteur. Cela a conduit à la définition de la relation *happened before* [11], dénotée par$\rightarrow$, comme suit :

1. Si *e* et *e'* sont deux événements dans le même processus, et *e* se produit avant *e'*, alors $e \rightarrow e'$.
2. Si e est l'émission d'un message par un processus et e' est la réception du même message par un autre processus, alors $e \rightarrow e'$.
3. Si $e \rightarrow e'$ et $e' \rightarrow e''$ alors $e \rightarrow e''$.

Cette relation définit un ordre partiel sur les événements. Deux événements distincts, *e* et *e'*, sont dits ***concurrents*** ou non comparables, noté $\boldsymbol{e} \parallel \boldsymbol{e'}$, si $e \not\rightarrow e'$ et $e' \not\rightarrow e$.

Une manière adéquate pour visualiser les calculs distribués est *les diagrammes de temps* (figure 2). La progression de chaque processus est symbolisée par une ligne horizontale. Le temps global est assumé de s'évoluer de gauche à droite. Les événements sont symbolisés par des points sur les lignes processus, et ceci selon leur ordre d'occurrence. Les messages sont représentés par des flèches connectant les événements d'émission avec leurs événements de réception correspondants.

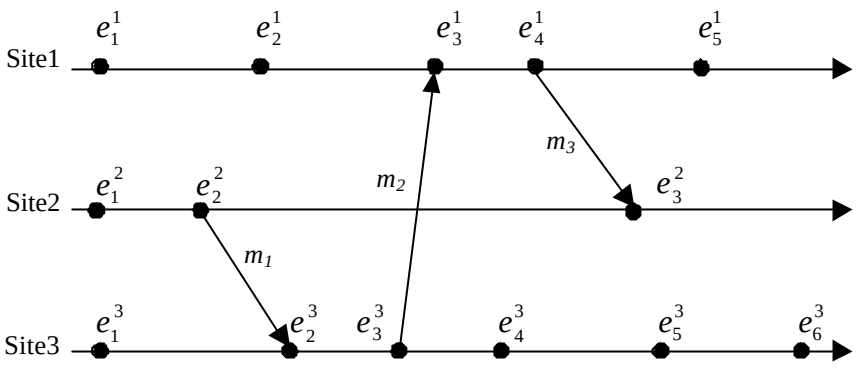


Figure 2 – *Un diagramme de temps d'un calcul distribué*

Afin de dater (estampiller) les événements, des horloges logiques de différents types ont été définies afin de rendre compte de la relation de dépendance (i.e. la relation *happened before*) entre les événements. Parmi les mécanismes d'estampillage qui ont été proposés pour les systèmes distribués, nous citons les mécanismes suivants :

3.2.1 Les horloges logiques linéaires (scalaires)

Cette représentation de temps a été proposée par Lamport en 1978 [11]. Chaque site gère un compteur dont la valeur est un entier : les valeurs des horloges sont donc comparables en tant que valeurs entières. Sur chaque site ce compteur est initialisé à 0 au lancement du système.

Si HL_i désigne l'horloge logique du site i et EL_m désigne l'estampille logique attribuée au message m lors de son envoi alors:

- Si un événement purement local se produit sur le site i, HL_i est incrémentée ($HL_i = HL_i + d$, $d>0$ *et peut prendre une valeur différente pour chaque incrémentation*) ;
- Si un événement correspond à l'envoi d'un message se produit sur le site i, HL_i est incrémentée et le message m est envoyé avec la nouvelle valeur de HL_i comme estampille ($EL_m = HL_i$) ;
- Si un événement correspondant à la réception d'un message m d'estampille EL_m se produit sur le site i, $HL_i = max\,(HL_i, EL_m) + d$.

La figure suivante (figure 3) donne la valeur des estampilles logiques scalaires associées en appliquant ce mécanisme à des événements de 3 sites entre lesquels des messages estampillés sont échangés.

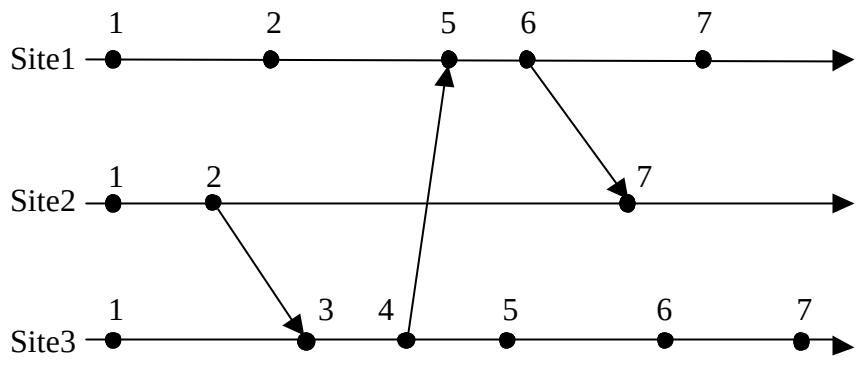


Figure 3 – *La progression des horloges linéaires*

3.2.2 Les horloges logiques vectorielles

Le système de datation par estampilles scalaires introduisait un ordre artificiel sur des événements concurrents. Il est néanmoins utiles de pouvoir déterminer la dépendance ou l'indépendance causale entre deux événements. Le mécanisme de datation par estampilles vectorielles introduit indépendamment par Fidge et Mattern en 1988 [12, 13] permet de pallier à cet inconvénient.

Les règles suivantes sont appliquées pour la gestion des horloges vectorielle : soit HV_i l'horloge vectorielle du site *i* et EV_m l'estampille vectorielle attribuée au message *m* lors de son envoi, alors :

- Si un événement purement local se produit sur le site *i*, $HV_i[i]$ est incrémentée ($HV_i[i]$ ++) ;
- Si un événement correspondant à l'envoi d'un message se produit sur le site *i*, $HV_i[i]$ est incrémentée et le message *m* est envoyé avec la nouvelle valeur de HV_i comme estampille ($EV_m = HV_i$) ;
- Si un message *m*, d'estampille EV_m, est reçu sur le site *i* :
  - $HV_i[i]$ est incrémentée ;
  - $\forall j \neq i, HV_i[j] = \max(HV_i[j], EV_m[i])$.

La figure suivante (figure 4) donne la valeur des estampilles vectorielles associées aux différents événements des 3 sites : en appliquant la méthode qu'on vient de décrire :

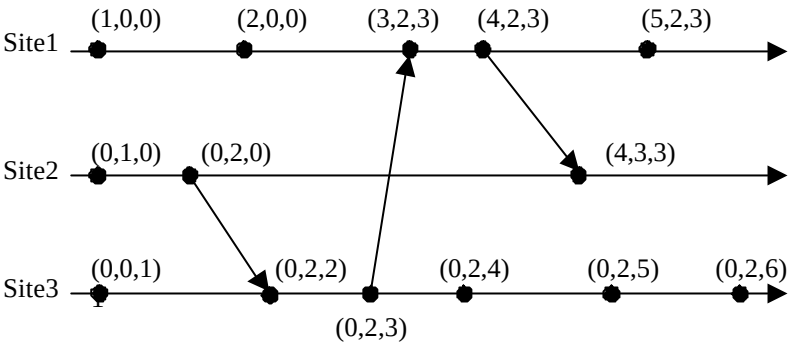


Figure 4 – *La progression des horloges vectorielles*

Le système de datation par estampilles vectorielles a la remarquable propriété de *refléter exactement la relation de précédence causale entre événements.* Soit EV_e l'estampille de e :

- $e \leq e' \Leftrightarrow EV_e \leq EV_{e'}$;
- $e \parallel e' \Leftrightarrow EV_e \parallel EV_{e'}$.

En d'autres termes, il y a un *isomorphisme* entre l'ensemble des événements partiellement ordonnés produits par le calcul distribué et leurs estampilles [14]. Donc, la dépendance causale peut être totalement caractérisée par l'utilisation des techniques d'estampillage par vecteur logique.

Ces présentations ont fourni une modélisation correcte de l'ordre partiel pour les systèmes distribués. Cependant, ces horloges ne sont pas conformes pour les systèmes mobiles, comme on va le montrer dans la section suivante, et ceci est dû au *(i)* manque d'évolution : leur taille est égale au nombre des sites, et *(ii)* leur inadéquation à la fluctuation du nombre des sites.

3.3 Insuffisance des horloges vectorielles pour un système mobile

Du fait que toute communication incluant des sites mobiles prend place à travers la *MSS* de la cellule dans laquelle ils sont présents, donc les relations de causalité peuvent être tracées en utilisant des horloges vectorielles de *n* composantes, où *n* est le nombre des MSS_s dans le système. Ceci peut amener à une réduction significative dans les taux *(overheads)* de communication, car le nombre des MSS_s est généralement une petite fraction du nombre total des nœuds (MH_s et MSS_s) dans ce type de système. Cependant, utilisant seulement *n* entiers dans une horloge vectorielle est insuffisant pour tracer toutes les dépendances causales. La preuve de cette affirmation est donnée par l'exemple suivant (figure 5):

Comme il y a deux MSS_s dans le système, l'horloge vectorielle a deux composantes. Initialement, les horloges vectorielles des deux MSS_s sont initialisées à *(0,0).* MH_b envoie un message m_1 à MH_c. Donc, MH_b envoie le message à MSS_p. MSS_p associe une estampille vectorielle de *(1,0)* avec l'événement *Send*(m_1), et retourne le message à MSS_q. Quand m_1 arrive à MSS_q, MSS_q mis à jour son horloge vectorielle à *(1,1),* associe cette estampille avec l'événement *Recv*(m_1), (*i.e.*, *V(Recv*(m_1)*) = (1,1)*) et renvoie m_1 à MH_c.

MH_d envoie un message m_2 à MH_a. Donc, MH_d envoie m_2 à MSS_q. MSS_q reçoit m_2 après la réception de m_1. D'ici, MSS_q mis à jour son horloge vectorielle à *(1,2)* et associe cette estampille avec l'événement *Send(m_2), i.e. V(Send(m_2)) = (1,2).*

Comme il n'y a pas de relation causale entre *Recv(m_1)* et *Send(m_2),* les deux événements sont concurrents. Cependant, *V(Recv(m_1)) < V(Send(m_2))* impliquant que *Send(m_2)* est causalement dépendant de *Recv(m_1)* : *une perte de la propriété d'isomorphisme des horloges vectorielles.*

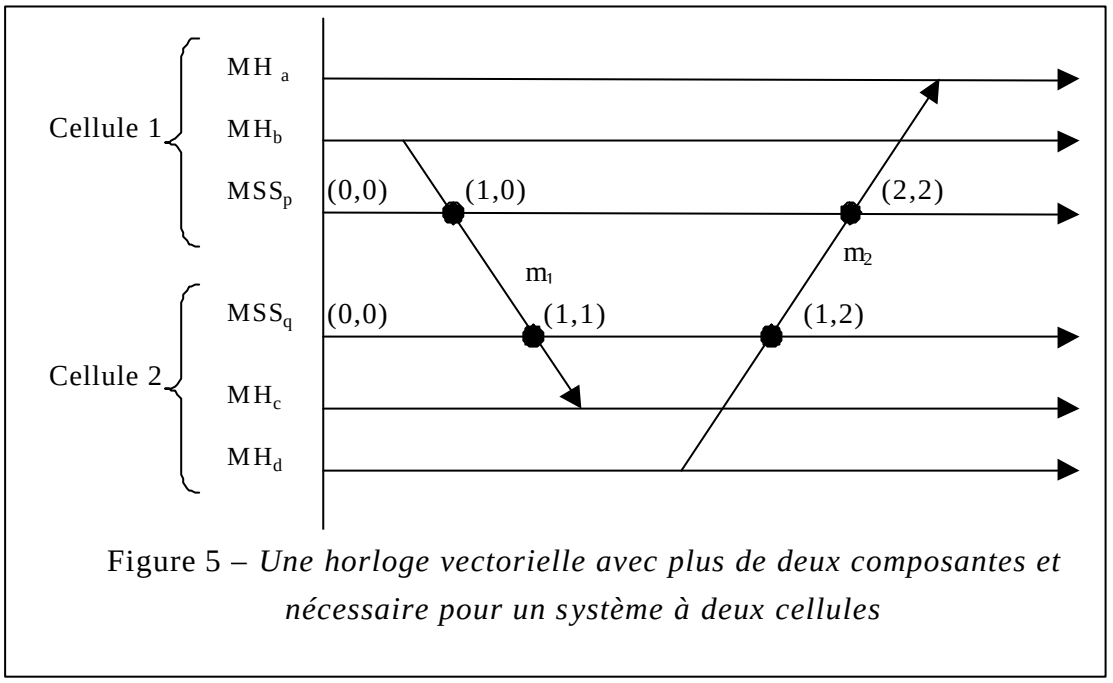


Figure 5 – *Une horloge vectorielle avec plus de deux composantes et nécessaire pour un système à deux cellules*

Puisque cette perte d'information n'est pas admissible, alors un recours à d'autres alternatives plus efficaces pour le traçage de la causalité dans les systèmes mobiles apparaît indispensable.

3.4 Les alternatives des mécanismes existants pour les systèmes mobiles

Il est clair qu'un mécanisme de traçage de la causalité qui est limité au nombre des MSS_s et qui évite une communication coûteuse sur les liens *MH-MSS* doit aboutir à une implémentation plus évolutive et efficace sur cette classe de systèmes distribués. Avec cet objectif à l'esprit, *Prakash* et *Singhal* [15] ont proposé deux alternatives pour les horloges vectorielles, pour les systèmes mobiles, dites *les séquences de dépendance (dependency sequences)* et *les horloges hiérarchiques (hierarchical clocks).*

Nous présentons dans les paragraphes suivants la définition de ces alternatives :

3.4.1 L'approche des séquences de dépendances

La *MSS* d'une cellule joue le rôle du *proxy* pour tous les MH_s présents dans sa cellule. Tous les événements causant la dépendance (*EMISSION* et *RECEPTION* de messages) observés par une *MSS*, sont assignés des nombres de séquences croissants. Initialement, le compteur séquence est initialisé à 0. Si e_i et e_j sont deux événements successifs causant la

dépendance et observés par *MSS*, alors le nombre de séquence assigné à e_j est égal au nombre de séquence assigné à e_i plus un.

Les prédécesseurs causaux d'un événement *e* sont représentés comme un ensemble de séquences de dépendance. Chaque séquence de dépendance dans l'ensemble correspond à une cellule dans le système, et se compose d'une séquence d'entiers non négatifs. Les paires de ces entiers représentent des séquences contiguës d'événements causant la dépendance dans une cellule qui sont des prédécesseurs causaux de *e*.

La figure suivante (figure 6) illustre l'approche des séquences de dépendances. Les événements d'émission et de réception de messages *0 – 4, 6 –12, 14 – 17, et 19 – 20* dans la cellule C_1, représentés par des rectangles, sont des prédécesseurs causaux de l'événement *e*. les événements *5, 13, et 18* dans la cellule C_1 ne sont pas des prédécesseurs causaux de *e*, et correspondent aux *gaps* dans la séquence. Les séquences de dépendance de *e*, pour ce système à 4 cellules sont *[{0,4,6,12,14,17,19,20}, {0,6,9,11,14,16}, {0,3,7,15,21,33}, {0,3,5,7,9,9,11,17}]*.

Figure 6 – *Les séquences de dépendance pour un événement e*

Présentation de l'algorithme

La station de base de la cellule *P*, notée par MSS_p, maintient les séquences de dépendance de tous les MH_s dans la cellule. L'ensemble des séquences de dépendance de MH_a est noté par DS_a. DS_a a *n* composantes, où *n* est le nombre des cellules dans le système : une composante pour chaque cellule. La $P^{éme}$ composante correspondant à la cellule *P*, est notée par $DS_a\ [P]$. Chaque *MSS* maintient aussi un compteur entier pour garder la trace des événements d'émission et de réception de messages observés par elle, depuis le début du calcul. $DS_a\ [P]$ est initialisée par une séquence vide, pour $0 \leq P \leq n-1$, et compteur est initialisé à *0*.

Les fonctions suivantes sont supposées être définies pour les séquences d'entiers :

last (DSa [P]) : retourne le dernier entier de la séquence DSa [P].

concatenate (seq_1, seq_2) : ajoute seq_2 à la fin de seq_1, où seq_1 et seq_2 sont toutes les deux des séquences d'entiers, et retourne la séquence résultante.

Quand MH_a, résidant dans la cellule *P* envoie un message à MH_b, le message est d'abord envoyé sur un canal sans fil à MSS_p.

Action de MSS_p en recevant le message à partir de MH_a :

```
{
  counter ¬ counter + 1 ;
  if (last(DS_a[P]) = counter – 1)
       last(DS_a[P]) ¬ counter ;
  else
      DS_a[P] ¬ concatenate(DS_a[P], {counter, counter}) ;
  send message and DS_a over the wireline network to MH_b's service station, namely MSS_q ;
}
```

Action de MSS_a en recevant le message et DS_a envoyés par MH_a pour MH_b :

```
{
  counter ¬ counter + 1 ;
  if (last(DS_b[q]) = counter – 1)
       last(DS_b[q]) ¬ counter ;
  else
      DS_b[q] ¬ concatenate(DS_b[q], {counter, counter}) ;
  for all 0 £ i< n
      DS_b[i] ¬ merge(DS_b[i], DS_a[i]) ;
}.
```

3.4.2 L'approche des horloges hiérarchiques

Les horloges hiérarchiques sont présentées comme une technique pour la caractérisation de la relation causale dans les systèmes où les processus ne nécessitent pas un ordre total. L'abstraction d'un comportement de cellule par un processus avec un ordonnancement partiel entre les événements est une violation de la définition standard d'un processus où tous les événements sont totalement ordonnés. Ainsi, la relation *happened before* ($\rightarrow$) de Lamport ne peut pas capturer complètement la relation de dépendance causale. A la place, une nouvelle relation, $\longrightarrow$ est définie sur un ensemble arbitraire d'événements e_1, e_2, et e_3 comme suit :

1. $e_1 \rightarrow e_2$ si $e_1 \xrightarrow{i} e_2$, i.e., les événements e_1 et e_2 se produisent sur le même processus abstrait et il y a une dépendance causale entre e_1 et e_2.

2. $e_1 \rightarrow e_2$ si $e_1 \xrightarrow{m} e_2$, i.e., e_1 est un événement d'émission d'un message dans un processus abstrait et e_2 est l'événement de réception correspondant dans un autre processus abstrait.

3. $e_1 \rightarrow e_2$ si $\exists\ e_3 : e_1 \rightarrow e_3$ et $e_3 \rightarrow e_2$.

Description de l'horloge

En se basant sur l'abstraction de processus décrite ci-dessus, chaque événement du processus abstrait est assigné une valeur horloge hiérarchique pour capturer la relation $\longrightarrow$ entre les événements. L'horloge hiérarchique f est maintenue dans les MSS_s, et elle a deux composantes :

1. f^i est une horloge locale représentant la relation $\xrightarrow{i}$. C'est un *vecteur bit* à longueur variable avec un bit pour chaque événement qui est produit dans le processus jusqu'à ce point. $f^i(e_{i,x})$ est de longueur *x bits* et son x^{em} bit est mis à *1*. De plus, pour chaque événement local du processus P_i qui précède causalement $e_{i,x}$, le bit correspondant est mis à 1 dans $f^i(e_{i,x})$.

2. f^m est une horloge globale représentant la relation $\xrightarrow{m}$. C'est un vecteur entier à n composants, un composant pour chaque processus abstrait (cellule) dans le système. $f^m(e_{i,x})[k]$, la $k^{éme}$ composante de l'horloge globale de l'événement $e_{i,x}$, identifie le dernier événement sur le processus k qui précède causalement $e_{i,x}$.

   Pour un événement d'émission de messages $e_{i,x}$, la *MSS* de la cellule d'émission fait l'action suivante pour mettre à jour l'horloge hiérarchique du processus abstrait correspondant : $f^m(e_{i,x})[k]$, où $k \neq i$, est mis au maximum des $k^{éme}$ composantes de tous les $e_{i,y}$ tel que $e_{i,y} \xrightarrow{i} e_{i,x}$. $f^m(e_{i,x})[i]$ est mis à x. $f^m(e_{i,x})$ est envoyée avec le message comme son estampille vectorielle.

   Pour un événement de réception de messages $e_{i,x}$, la *MSS* de la cellule réceptrice fait l'action suivante : $f^m(e_{i,x})[k]$ $(k \neq i)$ est mis au maximum des $k^{éme}$ composantes de tous les $e_{i,y}$ tel que $e_{i,y} \xrightarrow{i} e_{i,x}$ et la $k^{éme}$ composante de l'estampille vectorielle portée par le message. $f^m(e_{i,x})[i]$ est mis à x.

La figure ci-dessous (figure 7) illustre le fonctionnement des horloges hiérarchiques. L'horloge locale f^i est représentée par la séquence bit […] qui indique, pour chaque événement, le qui des autres événements est en fait dans son passé. L'horloge globale f^m est représentée par un vecteur.

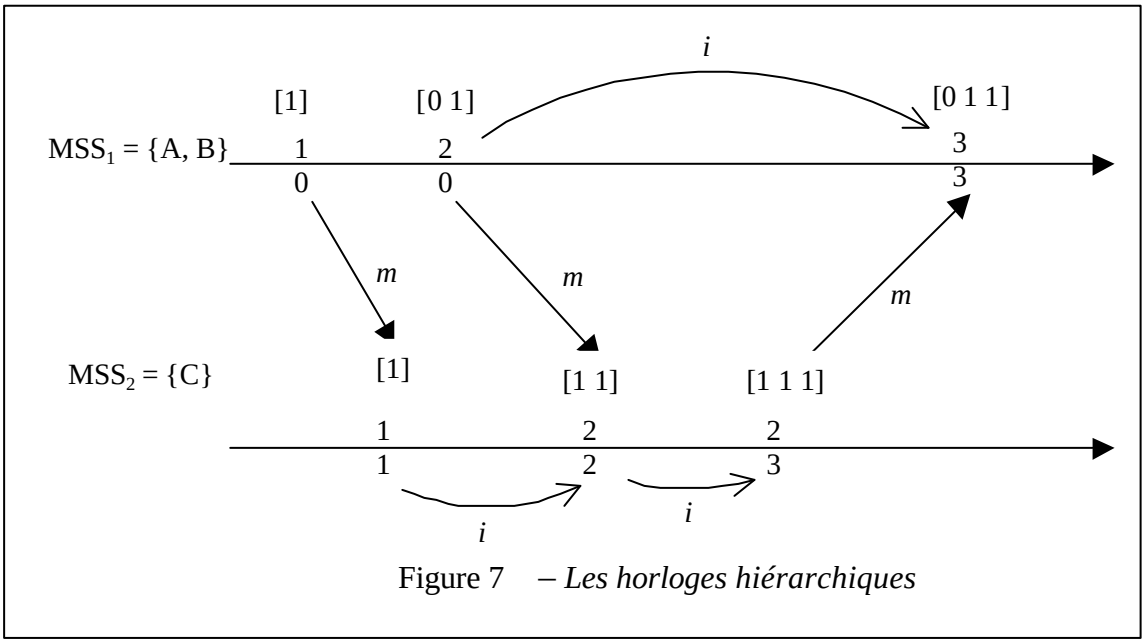


Figure 7 – *Les horloges hiérarchiques*

3.5 Comparaison entre les alternatives

Pour ces deux alternatives, la conclusion suivante est faite : l'approche séquences de dépendance a des taux de communication élevés et n'exige pas un délai de temps élevé pour déterminer le passé d'un événement. L'approche horloge hiérarchique a des taux de communication baissés, mais la détermination du passé d'un événement prend du temps.

4 L'ordre causal dans les environnement mobile

L'exécution asynchrone des processus et les délais de communication imprévisibles confèrent aux systèmes répartis un comportement non déterministe et rend plus complexe la conception, la vérification et l'analyse des programmes répartis. Pour simplifier la conception et le développement des applications distribuées, l'idée de "l'ordre causal des messages" a été introduite par Joseph et Birman [16] dans le cadre du projet ISIS [17] développé à l'université de Cornell. L'ordre causal des messages garantit que l'ordre de livraison des messages ne viole pas la causalité dans le système. Le processus diffère la livraison d'un message donné jusqu'à l'arrivée de tous ses prédécesseurs causaux.

4.1 Définition

Si pour n'importe quels deux messages M et M', $Send(M) \rightarrow Send(M')$ et M et M' ont la même destination, alors l'ordonnancement causal assure que $Delivery(M) \rightarrow Delivery(M')$.

En d'autres termes, les messages dirigés vers la même destination sont délivrés dans un ordre consistant avec leur causalité. La causalité considérée ici est déterminée par la relation "*happened before*" de Lamport [11] mais elle est limitée aux événements d'émission et de réception de messages.

Il est important de distinguer entre l'arrivée d'un message et sa livraison. L'arrivée d'un message signifie que le réseau de communication a placé le message dans le buffer du processus récepteur. La livraison d'un message signifie que le processus a pris le message pour le traitement. Naturellement, pour un message m la relation $\mathrm{Re}\,ceive(m) \rightarrow Delivery(m)$ est toujours vraie.

4.2 L'utilité de l'ordre causal

Le concept d'ordre causal est d'un intérêt considérable pour la conception des systèmes distribués et trouve ses applications dans plusieurs domaines tels que les mises à jour des données répliquées [18], la collection d'état global [19], la mémoire distribuée partagée [20], la téléconférence [21], les systèmes multimédia [22], et l'allocation des ressources [23].

Nous citons ici l'utilité de la propriété d'ordre causal pour certains domaines:

- *La génération consistante du snapshot*: le calcul de l'état global d'un système distribué de sorte à ne pas voir un "effet" (i.e., la réception d'un message) sans voir sa "cause" (i.e., son émission) [19].

- *La gestion d'allocation des ressources distribuées*: les requêtes pour une ressource doivent être satisfaites dans l'ordre causal de leur émission par les processus [23].
- *Téléconférence*: une application de téléconférence est une application dans laquelle deux où plusieurs personnes distants peuvent mener une conversation. Dans ce cas, un participant ne doit pas recevoir un commentaire sur certaines idées avant la réception du message exprimant l'idée originale. Sinon, l'idée en retard peut être mal interprétée. [21]

4.3 Les protocoles d'ordre causal dans l'environnement distribué

La mise en œuvre de l'ordre causal dans un système asynchrone nécessite l'adjonction d'information de contrôle à chaque message. Le processus qui reçoit un message utilise cette information pour déterminer s'il y a des message non délivrés qui le précèdent causalement et auquel cas sa livraison est différée jusqu'à la livraison de ces derniers.

Plusieurs algorithmes ont été développés pour fournir un ordonnancement causal des messages:

Le premier protocole d'ordre causal a été proposé par Birman et Joseph en 1987 [16] dans le contexte du système Isis [17]. Ce protocole utilise une technique d'histoire causale pour l'ordonnancement causal dans le système Isis. L'idée est la suivante: chaque message m emporte avec lui tous les autres messages qui le précèdent causalement [16]. Si un processus reçoit un buffer de messages contenant plusieurs messages envoyés à la destination courante, alors il les traite dans l'ordre de leur apparition dans le buffer.

Due à l'information redondante, ce schéma est résistant aux défaillances de processeurs. Cependant, l'inconvénient majeur de cet algorithme consiste dans l'évolution non bornée de l'information de contrôle. Des optimisations ont été appliquées afin de réduire l'histoire causale attachée aux messages. Les versions les plus récentes de Isis sont basées sur le temps vectoriel, qui représentent une amélioration considérable due à la réduction de la surcharge [24]. L'algorithme d'ordre causal de Peterson, Buchlolz et Schlichting [25] est similaire à ISIS sauf qu'il emporte, dans l'information de contrôle, les identificateurs des messages au lieu des messages. De plus, l'information de contrôle inutile n'est pas envoyée si le site émetteur l'a déjà envoyé.

Kearns et al. [23] ont présenté un protocole pour assurer la livraison point à point d'ordre causal des messages dans un système avec un seul serveur, vers lequel les messages émis à partir de tous les autres processus (clients) sont délivrés dans l'ordre causal. Le protocole utilise des vecteurs compteurs de messages qui sont localisés sur chaque processus et qui sont attachés à chaque message. Un client j recevant un message avec un vecteur attaché V_k à partir d'un autre client k met à jour son vecteur $V_j = \max(V_j, V_k)$. Avant qu'un client i émis un message au serveur, il incrémente sa composante $V_i[i]$ dans son vecteur V_i et attache le vecteur au message. Quand le serveur reçoit un message m à partir d'un processus i, il compare d'abord son vecteur V_s avec V_m, le vecteur accompagnant le message m. si $V_s[j] < V_m[j]$ pour un certain $j \neq i$, alors il existe un message non encore reçu à partir d'un autre client, et le message m est différé jusqu'à l'arrivée du message. Sinon le serveur consomme le message après la mise à jour de son

vecteur $V_s = \max(V_s, V_m)$. Notant que $V_i[j]$ représente la connaissance du processus *i* sur le nombre de messages émis par le processus *j* au serveur.

Une autre implémentation de l'ordre causal est donnée par Schiper et al. [26]. Cette implémentation n'est qu'une généralisation de la solution proposée par Kearn où tous les processus sont considérés des serveurs. Dans cette implémentation, une information de contrôle est rajoutée aux messages. Cette information de contrôle permet au site destinataire d'un message *M* de connaître s'il y a des messages qui doivent être délivrés avant *M*, dans le but de respecter l'ordre causal: si c'est le cas, *M* n'est pas immédiatement délivré. L'information de contrôle est composée d'un nombre borné de paires (destination, temps vectoriel).

Raynal, Schiper, et Toueg [27] proposent une variante de la méthode précédente qui n'enregistre que les événements de communication. Elle a les mêmes propriétés de la méthode précédente, mais elle est marginalement plus simple à implémenter. L'algorithme proposé, dit l'algorithme RST, est basé sur le compte de messages et assume que les canaux soient fiables. L'algorithme RST attache une matrice *N* x *N* d'entiers à chaque message, où *N* est le nombre des sites dans le système.

L'algorithme RST pour l'ordonnancement causal maintient deux tableaux, $DELIV_i[N]$ et $SENT_i[N, N]$, pour chaque site P_i. $DELIV_i[j]$ dénote le nombre total des messages reçus par P_i à partir de P_j. $SENT_i[k, j]$ indique la connaissance de P_i sur le nombre de messages que P_k a envoyé à P_j. Les étapes suivantes sont exécutées par P_i pour assurer l'ordre causal: quand P_i envoie un message *m* à P_j, P_i attache sa valeur courante de $SENT_i$ au message *m*, i.e. P_i envoie (m, $SENT_i$) à P_j. P_i incrémente alors $SENT_i[i,j]$ par 1. en recevant (m, ST) à partir de P_j, l'algorithme d'ordre causal à P_i vérifie d'abord si $DELIV_i[k] \geq ST[k,i]$ pour tout *k*. si vraie, le message *m* est délivré à l'application, $DELIV_i[j]$ est incrémenté par 1, $SENT_i[j,i]$ est initialisée à $ST[j,i]$ +1, et finalement $SENT_i[j,k]$ est initialisée à $max(ST[j,k], SENT_i[j,k])$ pour tout *j*, *k*. sinon, *m* est mis en attente jusqu'à ce que $DELIV_i[k] \geq ST[k,i]$ pour tout *k*.

Ces protocoles se différencient principalement par la taille des informations de contrôle transportées par les messages. Dans le protocole [16], la taille de l'information de contrôle est non bornée. Le protocole [25] est similaire au précédent, toutefois l'information de contrôle transportée par les messages est réduite à un ensemble d'identificateurs de messages. Dans le protocole de [26], chaque message transporte *N* -1 couples (destination, vecteur de temps). Dans le protocole de [27], chaque processus maintient une matrice de taille *N* x *N* qui est transportée par chaque message émis.

4.4 Les protocoles d'ordre causal dans l'environnement mobile

L'utilité de l'ordre causal dans diverses applications a connu une activité de recherche conséquente dans les systèmes en environnement distribué. Cependant, les contributions dans ce domaine pour des systèmes en environnement mobile, restent très modestes. L'ordre causal est un concept approprié pour des applications qui nécessitent l'interaction entre plusieurs usagers depuis des localités différentes. Parmi les applications majeures des systèmes répartis en environnement mobile où l'ordre causal est essentiel, nous citons spécialement les applications coopératives, les systèmes de téléconférence, etc.

Les protocoles déjà proposés pour l'environnement distribué peuvent être exécutés par chaque site mobile avec toutes les structures de données utiles étant stockées sur les sites mobiles eux-mêmes. Cependant, en considérant les ressources limitées et la bande passante des liens sans fils disponible aux sites mobiles, il est inapproprié d'appliquer ces protocoles directement sur les systèmes mobiles. Les facteurs suivants doivent être pris en considération lors de la conception de protocoles pour les systèmes mobiles : le taux de calcul sur les sites mobiles, et la taille des messages transmis sur le medium sans fil doivent être minimaux. De plus, le protocole doit être évolutif (adaptable), et être capable de gérer l'effet des connexions et déconnexions des sites mobiles.

L'adaptation d'un protocole de mise en œuvre de l'ordre dans un système réparti avec des sites mobiles soulève deux problèmes majeurs [28]: l'impact de la mobilité (i.e. la migration des sites mobiles entre les différentes cellules du système), et l'inhibition de la livraison de messages (i.e. délai d'attente inutile pour la livraison d'un message).

Alagar et Venkatesan [29] ont proposés trois extensions de l'algorithme d'ordre causal dans [27] pour les systèmes mobiles. Leur premier algorithme (*AV1*) gère les contraintes ressources des sites mobiles. Il sauvegarde les structures de données des MH_s utiles pour l'ordre causal dans les MSS_s, et l'algorithme est exécuté par les *MSS* au compte des MH_s. Cependant, le coût de communication est proportionnel au carré du nombre des MH_s. L'avantage de cet algorithme est que la procédure pour le traitement des migrations des sites est simple. Le deuxième algorithme (*AV2*) élimine les inconvénients du premier algorithme. La propriété d'ordre causal est explicitement maintenue entre les MSS_s. La taille de l'information de contrôle attachée aux messages est proportionnelle au carré du nombre des MSS_s. Puisque cette taille ne varie pas avec le nombre des MH_s, l'algorithme est évolutif et les connexions et déconnexions des sites mobiles ne posent pas de problème mais il peut y avoir certaine inhibition dans la délivrance des messages aux MH_s. Ce problème est dû essentiellement à la mise en œuvre de l'ordre causal entre les stations fixes du système. La livraison d'un message peut être retardée car l'ordre causal peut être violé du point de vue de la station de base alors qu'il est considéré comme livrable selon l'ordre des messages tel que le perçu par les unités mobiles locales. Pour faire face aux problèmes associés à la mobilité, cet algorithme est équipé par un module compliqué pour gérer les migrations de sites. Pour réduire ces délais sans les supprimer totalement, Alagar et Venkatesan ont proposé un troisième algorithme (*AV3*) qui est un algorithme hybride des deux autres algorithmes. Chaque *MSS* est partitionnée à k MSS_s logiques. Ainsi, les messages émis ou reçus par les unités mobiles appartenant à des stations logiques différentes ne peuvent pas s'inhiber mutuellement. Mais, plus k est grand, plus la possibilité des délais inutiles diminue mais la surcharge des messages et le coût de traitement des migrations de sites augmentent.

Yen, Huang, et Hwang (*YHH*) [30] ont proposé un algorithme basé sur [27]. La taille de l'information de contrôle attachée aux messages dans cet algorithme est entre celle de *AV1* et *AV2*. En particulier, chaque *MSS* maintient une matrice de taille N_s x N_h. Le délai inutile dans cet algorithme est plus petit que celui de *AV2*. Leur module de *handoff* est aussi plus efficace que celui de *AV2*. L'inconvénient de cet algorithme est qu'il ne satisfait pas la propriété de vivacité. Prakash, Raynal et Singhal (*PSR*) [31] ont présenté un algorithme où la taille de l'information de contrôle est relativement réduite; cependant, dans le cas le plus défavorable elle peut dépassée $O(N_h^2)$. De plus, cet algorithme est inadéquat pour les systèmes où le nombre des sites mobiles change dynamiquement car la structure de l'information de contrôle dépend du nombre des processus participants. Un autre algorithme a été proposé par Skawratananond, Mittal, et Garg (*SMG*) [32]. La surcharge de

l'information de contrôle dans cet algorithme est de l'ordre de $N_s^2 + N_h$. Son module de *handoff* est plus efficace que celui de *AV2* et *AV3* puisque il ne nécessite pas que les messages échangés entre les MSS_s soient causalement ordonnés.

5 Un protocole d'ordonnancement causal en environnement Mobile (*Mobi_Causal*)

Dans cette partie nous présentons un nouveau protocole d'ordonnancement causal qui est basé sur l'utilisation des nouveaux mécanismes d'estampillage proposés pour les environnements mobiles : les horloges hiérarchiques [15] et les séquences de dépendances [15]. Donc, l'objectif de notre contribution est d'étudier l'applicabilité de ces mécanismes pour l'implémentation d'un protocole qui résout le problème d'ordre causal en environnement mobile. Notre contribution se résume comme suit : (1) Notre condition de livraison est structurée de manière hiérarchique offrant ainsi un test par rapport à une dépendance interne avant de passer à tester les dépendances externes, c'est-à-dire si la livraison d'un message dépendant d'un autre message émis par le même site mobile alors il suffit de tester la livraison de ce message au lieu de tester toutes les dépendances car ceci sera fait à l'arrivée du message duquel il dépend. Ce qui permet d'éliminer les tests inutiles, (2) Avec notre protocole, nous sommes capables d'éliminer les délais d'inhibition tout en maintenant une taille minimale de l'information de contrôle attachée aux messages, (3) Finalement, nous prouvons la correction de notre condition de livraison implémentée par le module statique.

5.1 Protocole Mobi_Causal

5.1.1 Définitions et interprétation des structures de données utilisées

Pour chaque h_i

$$h_i \longrightarrow \begin{array}{l} id_{mess_E}, id_{mess_D} \\ f^i \\ LastSend_{h_i}, LastRcv_{h_i} \\ depend_{h_i} \\ f \end{array}$$

Pour chaque S_i

$$S_i \longrightarrow \begin{array}{l} IdSend_{S_i} \\ AtFile_{S_i} \\ id_{LastSend} \end{array}$$

- id_{mess_E} : Compteur des messages émis ou reçus par un site mobile h_i. C'est un entier qui est incrémenté chaque fois qu'un message est émis ou reçu.
- id_{mess_D} : C'est l'identificateur du dernier message reçu par un site mobile h_i. C'est un entier qui prend la valeur de id_{mess_E} à la livraison du message.
- $LastSend_{h_i}$: Un ensemble de triplets (h_j, id, f^i) qui garde l'identificateur du dernier message émis par h_i aux autres h_j, ainsi que la valeur de l'horloge locale calculée f^i.

? f^i : C'est un vecteur de bits de taille variable. Sa construction est basée sur le principe des horloges hiérarchiques locales.

Construction de f^i

a. Idée

Nous distinguons trois types de messages : les messages émis par un site mobile et par conséquent par sa station de base, notés E, les messages reçus par une station de base mais pas forcement délivrés au site mobile destinataire, notés R, et les messages délivrés d'une station de base à un site mobile après avoir vérifié la condition de livraison, notés D.

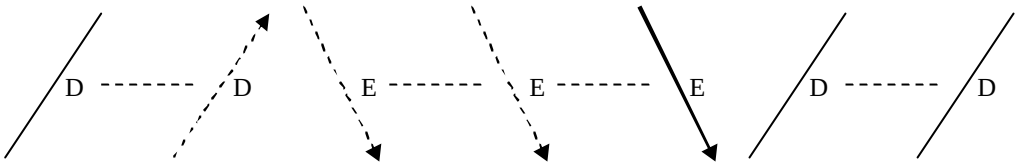


Figure 8 — *Types de messages*

Les réceptions par un site mobile des messages délivrés D sont indépendantes, car un message ne peut être délivré que si la condition de livraison est vérifiée au niveau de la station de base. Pour un message d'émission E, il dépend du dernier message D reçu et des messages E qui sont émis avant lui au même site mobile, et après le dernier message reçu D.

Comme il est montré dans la figure ci-dessus (figure 8), si le site mobile décide d'émettre le message E en gras, alors l'information de dépendance causale de ce message est construite à partir des messages E et D représentés sur la figure par des pointillés.

Nous proposons la structure suivante à f^i : $B_D C_{Int} B_E$

Tel que : B_D : Une chaîne de un bit qui est égal à 1 si $id_{mess_D} \neq 0$ sinon cette chaîne est vide.

B_E : Une chaîne de un bit qui est égal à 1. Ce bit marque le message en cours d'émission.

C_{Int} : Une chaîne de bits de longueur $id_{mess_E} - id_{mess_D} - 1$

La fonction *Construct*(f^i)

/* La longueur de f^i est $id_{mess_E} - id_{mess_D} + 1$ */

Begin

1. ***If*** $id_{mess_D} \neq 0$ ***Then*** $B_D = 1$
   Else ∄
   EndIf
2. $B_E = 1$
   /* *Construction de C_{Int}*

3. **For** $k = id_{mess_D} + 1$ **to** $(id_{mess_E} - 1)$
Do
4. **If** $\exists (h_j, k, \_) \in LastSend_{h_i}$
Then
/* $C_{Int}(val_1, val_2)$ *signifie la chaîne* C_{Int} *avec* val_1 *comme position*
/* *de départ et* val_2 *comme longueur* */
5. **If** $id_{mess_D} \neq 0$ **Then** $C_{Int}(1, k - id_{mess_D}) = f^i_{(h_j,k)}(2, k - id_{mess_D})$
Else $C_{Int}(1, k - id_{mess_D}) = f^i_{(h_j,k)}(1, k - id_{mess_D})$;
EndIf
6. $C_{Int}(k+1, id_{mess_E} - k) = 0$
Exit;
EndIf
EndDo
7. $f^i = concat(B_D, C_{Int}, B_E)$
End

b. Explication de *Construct* (f^i)

L'horloge f^i est un vecteur de bits de taille variable. La taille maximale de f^i est égale au nombre de messages émis entre deux réceptions données. Nous supposons que les sites mobiles ont un taux de coopération élevé pour empêcher l'évolution non bornée de la taille de f^i, c'est-à-dire nous supposons l'absence de sites mobiles qui ne jouent que le rôle de récepteur. Sa structure est de la forme $B_D C_{Int} B_E$. La construction de f^i est faite par les instructions suivantes de la fonction *Construct* (f^i) :

1. Cette instruction teste si le site mobile à déjà reçu un message ou pas. Dans le cas où le site mobile a reçu déjà un message ce bit, B_D, est mis à 1, sinon il n'existe pas.
2. Puisque le principe de f^i est de tracer les dépendances entre les événements et puisque un événement dépend de lui-même, alors le bit B_E est mis à 1. pour dire que le message à émettre dépend de lui-même.
3. 4. 5. 6. Ces instructions servent à construire la chaîne C_{Int}. C_{Int} est une chaîne de bits de taille égale au nombre de messages émis après le dernier message reçu par le site mobile et avant le message à émettre. Un bit de C_{Int} est égal à 1 si le message correspondant à ce bit est déjà émis au site mobile destinataire sinon le bit est mis à 0. Puisque chaque site mobile possède un ensemble de triplets $LastSend_{h_i} = \{(h_j, id, f^i), ...\}$ pour garder l'information sur le dernier message émis à un site mobile h_j. Donc, construire C_{Int} revient à chercher un triplet dans $LastSend_{h_i}$ possédant comme identificateur de site, l'identificateur du site destinataire du message (*Instr.* 5.). Ensuite compléter le reste de la chaîne par des 0 (*Instr.* 6.).
7. Cette instruction fait la concaténation de B_D, C_{Int}, B_E pour construire f^i.

- $LastRcv_{h_i}$: Un ensemble de triplets (h_j, S_j, SD_j) où $SD_j = \{id_1, id_2, id_3, id_4, ...\}$ qui garde les identificateurs des messages reçu par h_i en provenance de h_j. La troisième entrée de chaque triplet correspond à une séquence de dépendance. Donc, sa construction est faite de la même manière que dans le mécanisme des séquences de dépendance [] expliqué dans le 2em chapitre.
- $depend_{h_i}$: Un booléen dont la valeur dépend de f^i
- f : Un vecteur de couples de taille N_S (*où N_S est le nombre des stations de base*), tel que une entrée $f_m[k] = \{(h_j, id), ...\}$ dans le vecteur correspond à l'ensemble des identificateurs des messages prédécesseurs de m sur la station de base k dont la livraison n'est pas encore confirmée. Nous gardons l'identificateur et à quel site mobile ce message a été envoyé.
- $IdSend_{S_i}$: Compteur des messages émis par une station de base S_i.
- $id_{LastSend}$: C'est l'identificateur du dernier message déjà émis par un site mobile h_i au site mobile h_j avant l'émission en cours.
- $AtFile_{S_i}$: File d'attente des messages non encore délivrés.

5.1.2 Le module statique (A exécuter par la station de base)

La phase d'émission

L'émission d'un message m à partir d'un site mobile h_i de la station de base S_i vers un site mobile h_j de la station de base S_j se fait en deux étapes : tout d'abord la transmission du message à partir du site mobile émetteur h_i vers sa station de base S_i, ensuite l'émission du message à partir de S_i vers la station de base réceptrice S_j après lui avoir rattaché l'information de contrôle.

***Send* (*m*)** *de* $h_i \rightarrow S_i$
/ Quand* h_i *décide d'émettre un message m*
Begin
id_{mess_E} ++
Envoyer m à S_i *via le canal sans fil.*
End

***Emission* (*m*)** *de* $S_i \rightarrow S_j$
Begin
1. $IdSend_{S_i}$ ++
2. $Construct(f^i_{h_i})$
3. ***If*** $f^i_{h_i} = 0^*1$ ***Then*** $depend_{h_i} = false$
Else $depend_{h_i} = true$
EndIf
4. ***If*** $\exists (h_j, id, \_) \in LastSend_{h_i}$ ***Then*** $id_{LastSend} = id$

Else $id_{LastSend} = 0$

EndIf

5. *Envoyer* $(m, IdSend_{S_i}, depend_{h_i}, id_{LastSend}, f)$ *à* S_j *via le canal filaire.*
6. $f[i] = f[i] \cup (h_j, IdSend_{S_i})$
7. $LastSend_{h_i} = LastSend_{h_i} \cup (h_j, IdSend_{S_i}, f^i)$
8. $Update(LastSend_{h_i})$

End

Update (file)
Begin
/ le rôle de cette fonction est d'éliminer les triplets redondants dans une file.*

If $\exists (h_j, Id_1, f_1^i)$ *and* $(h_j, Id_2, f_2^i) \in$ *file tel que* $Id_1 < Id_2$

Then *Supprimer* (h_j, Id_1, f_1^i) *de file*

EndIf

End

- **Explication de *Emission* (*m*)**

Quand la station de base S_i du site mobile émetteur h_i reçoit le message à émettre, elle calcule l'information de contrôle à attacher au message avant son émission à la station de base réceptrice S_j. Le message à émettre doit avoir la forme suivante : $(m, IdSend_{S_i}, depend_{h_i}, id_{LastSend}, f)$

1. La réception du message *m* par la station S_i provoque l'incrémentation de son compteur $IdSend_{S_i}$, qui sera attaché au message *m*. $IdSend_{S_i}$ correspond à l'identificateur de *m* par rapport à la station de base S_i.
2. 3. La valeur de la variable booléenne $depend_{h_i}$ dépend de f^i. Donc, après la construction de l'horloge locale f^i correspondant à h_i, on vérifie si f^i est de la forme 0^*1, cela signifie que ce message ne dépend d'aucun autre message, que ce soit interne au site mobile ($C_{Int} = 0^*$) ou externe (B_D n'existe pas). Dans ce cas, on met la valeur de $depend_{h_i}$ à *fasle*, et par conséquent la livraison de ce message ne dépendra d'aucun autre message. Donc, dés son arrivée à la station de base réceptrice ce message doit être délivré immédiatement. Dans le cas contraire, $depend_{h_i}$ aura la valeur *true*, ce qui signifie que ce message soit il dépend d'un autre message interne au site mobile h_i (c'est à dire dépend d'un message émis par h_i vers h_j avant le message *m*), soit il dépend d'un message externe ($B_D = 1$, i.e. que h_i a reçu un message avant l'émission de *m*).
4. Le message doit transmettre aussi, dans l'information de contrôle, l'identificateur du dernier message déjà émis par h_i vers h_j s'il existe. Donc, si on trouve un triplet $(h_j, id, \_) \in LastSend_{h_i}$, on affecte à $id_{LastSend}$ la valeur *id*, sinon $id_{LastSend}$ aura la valeur 0 pour dire que c'est le premier message qui va être émis par h_i vers h_j.
5. 6. 7. 8. Après l'émission du message *m*, on met à jour l'horloge globale f , donc $f[i] = f[i] \cup (h_j, IdSend_{S_i})$, et on ajoute à l'ensemble $LastSend_{h_i}$ le

triplet $(h_j, IdSend_{S_i}, f^i)$. Par la suite, on élimine les triplets redondants dans l'ensemble $LastSend_{h_i}$, c'est-à-dire s'il existe deux triplets (h_j, Id_1, f_1^i) et (h_j, Id_2, f_2^i) tels que $id_1 < id_2$ alors on supprime le triplet (h_j, Id_1, f_1^i).

La phase de réception

***Délivrer* (*m*)** à $h_j(S_j)$ $(m, IdSend_m, depend_m, id_{LastSend_m}, f_m)$
Begin
1. ***If*** $depend_m = false$
   Then *Deliver*(*m*)
   Else
2. ***If*** $(id_{LastSend_m} \neq 0 \; and \; \exists (h_i, \_, SD_i) \in LastRcv_{h_j} telque \; id_{LastSend_m} \in SD_i)$
   Then *Deliver (m)*
   Else
3. ***If*** $\begin{pmatrix} id_{LastSend_m} = 0 \; and \; \forall k = \overline{1,n}, telque \; \exists (idf, h_j) \in f_m[k]: \\ \exists (\_, S_k, SD_k) \in LastRcv_{h_j} \; tel \; que \; idf \in SD_k \end{pmatrix}$
   Then *Deliver*(*m*)
   $Build(f_{h_j})$
   Délivrer tous les messages dans la file d'attente $AtFile_{S_j}$ qui sont en attente de *m.* Ensuite les supprimer de cette file.
   Else *Attente* (*m*) ***/**** *Mettre le message dans la file d'attente*
   EndIf
   EndIf
   EndIf
End

La fonction *Attente* (*m*)
/* *utiliser une file d'attente AtFile*
Begin
$AtFile = AtFile \cup (m, IdSend_m, depend_m, id_{LastSend_m}, f_m)$
End

- **Explication de *Délivrer*(*m*)**

A l'arrivée du message *m* à sa station de base réceptrice, ce message ne sera délivré au site mobile destinataire h_j que si la condition de livraison est vérifiée, c'est-à-dire que tous les messages qui précèdent causalement le message *m* sont reçus par S_j :

1. Si $depend_m = false$, donc ce message ne dépend d'aucun autre message et sa livraison est immédiate.
2. Si $depend_m = true$, on a deux possibilités : soit *m* dépend d'un autre message interne au site mobile émetteur et qui est émis au même site destinataire h_j, dans ce cas $id_{LastSend_m} \neq 0$, alors il suffit de vérifier que ce message a été reçu par h_j. La

réception d'un message par un site mobile est exprimée par un triplet (h_i, S_i, Id) dans l'ensemble $LastRcv_{h_j}$. Si c'est le cas, le message *m* est délivré.

3. Si $depend_m = true$, et $id_{LastSend_m} = 0$, alors la dépendance causale est créée non pas par un message interne à h_i mais par la réception d'un message par ce site mobile. Dans ce cas, la livraison de *m* dépend de la livraison de ses prédécesseurs immédiats sur les différentes stations de base. Ces prédécesseurs sont identifiés par l'horloge globale f, qui a pour rôle de garder la trace des prédécesseurs immédiats d'un message donné. Une entrée dans f est un ensemble de couples (*idf, hl*) qui représentent les identificateurs des messages prédécesseurs sur la station de base correspondante à cette entrée et à quel site mobile il ont été envoyés. Donc, on vérifie pour chaque entrée dans f s'il existe un prédécesseur qui est émis à h_j alors on cherche s'il existe un triplet $(\_, S_k, SD_k) \in LastRcv_{h_j}$ *tel que* $idf \in SD_k$. Si cette condition est vérifiée alors on est sûr que tous les prédécesseurs immédiats de *m* sont reçus et par conséquent *m* sera délivré à h_j, ainsi que tous les messages dans la file d'attente $AtFile_{S_j}$ qui sont en attente de *m*. la livraison de ces messages entraîne leur suppression de la file d'attente.

Dans le cas, où aucune des trois conditions précédentes n'est vérifiée, alors on dit que la condition de livraison (la condition de précédence causale) n'est pas vérifiée, et par conséquent le message *m* est mis dans le file d'attente $AtFile_{S_j}$.

La fonction *Deliver*(*m*) à h_j
Begin
HDeliver(*m*) /* *Délivrer le message m au site mobile* h_j
$IdSend_{S_j}++;$
$f_{h_j} = f_m$
$LastRcv_{h_j} = LastRcv_{h_j} \cup (h_i, S_i, (IdSend_m, IdSend_m))$
End

La fonction *HDeliver*(*m*)
/* *A l'arrivée du message m au site mobile*
Begin
$id_{mess_E}++$
$id_{mess_D} = id_{mess_E}$
End

La fonction *Build*(f_{h_j}) à h_j
/* *Supprime l'information superflue. Tous les messages qui sont délivrés vont être*
/* *supprimés de* f_{h_j}.
Begin
If $(\forall h_l \in S_j, \exists (IdSend_m, h_j) \in f_{h_l}[i])$
Then Supprimer $(IdSend_m, h_j)$ de $f_{h_l}[i]$
EndIf

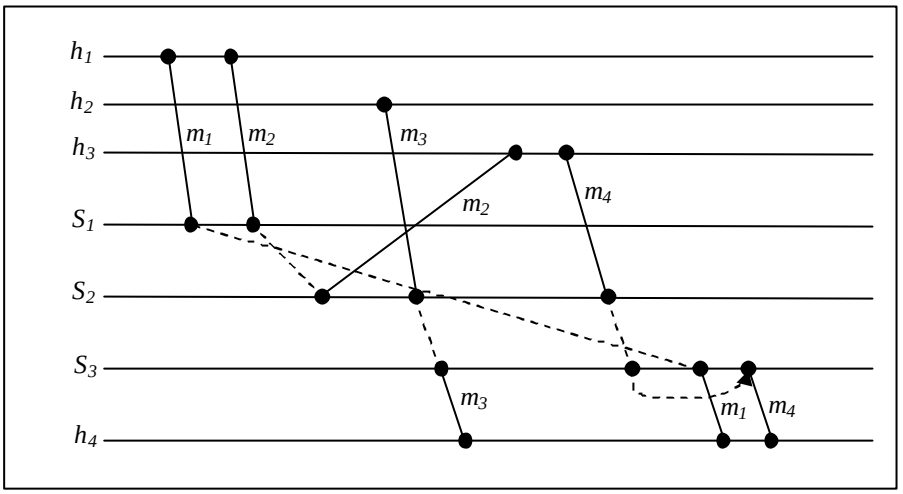
If $\left(\begin{array}{l}\forall k=\overline{1,n}, f_{h_j}[k]\neq 0, \forall h_l \in S_j \ telque \ \exists(idf, h_l)\in f_{h_j}[k]: \\ \exists(\_, S_k, SD_k)\in LastRcv_{h_l} \ tel \ que \ idf \in SD_k\end{array}\right)$

Then Supprimer (idf, h_l) de $f_{h_j}[k]$

EndIf

End

5.1.3 Exemple illustratif



- Emission de m_1 de $h_1(S_1)\rightarrow h_4(S_3)$

$h_1 : id_{mess_E} = 1,$ $\qquad S_1 : IdSend_{S_1} = 1$

$f^{i}_{h_1} = 1, depend_{h_1} = false$

$Envoyer\ (m_1, 1, false, 0, \begin{bmatrix}0\\0\\0\end{bmatrix})$

$f_{h_1} = \begin{bmatrix}(1, h_4)\\0\\0\end{bmatrix},$

$LastSend_{h_1} = \{(h_4, 1, 1)\}$

- Emission de m_2 de $h_1(S_1)\rightarrow h_3(S_2)$

$h_1 : id_{mess_E} = 2,$ $S_1 : IdSend_{S_1} = 2$

$f^{i}_{h_1} = 01, depend_{h_1} = false$

$Envoyer\,(m_2, 2, false, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix}$

$f_{h_1} = \begin{bmatrix} (1, h_4)(2, h_3) \\ 0 \\ 0 \end{bmatrix},$

$LastSend_{h_1} = \{(h_4, 1, 1), (h_3, 2, 01)\}$

? Réception de $(m_2, 2, false, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix})$ par $h_3(S_2)$

Puisque $depend_{h_1} = false$ alors le message sera délivré au site h_3, et les mises à jour suivantes seront faites :

$h_3 : id_{mess_E} = 1, id_{mess_D} = 1,$

$f_{h_3} = \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix},$

$LastRcv_{h_3} = \{(h_1, S_1, \{2,2\})\}$

? Emission de m_3 de $h_2(S_2) \rightarrow h_4(S_3)$

$h_2 : id_{mess_E} = 1,$ $S_2 : IdSend_{S_2} = 1$

$f^{i}_{h_2} = 1, depend_{h_2} = false$

$Envoyer\,(m_3, 1, false, 0, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix})$

$f_{h_2} = \begin{bmatrix} 0 \\ (1, h_4) \\ 0 \end{bmatrix},$

$LastSend_{h_2} = \{(h_4, 1, 1)\}$

? Réception de $(m_3, 1, false, 0, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix})$ par $h_4(S_3)$

Puisque $depend_{h_2} = false$ alors le message sera délivré au site h_4, et les mises à jour suivantes seront faites :

$h_4 : id_{mess_E} = 1, id_{mess_D} = 1,$

$f_{h_3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$

$LastRcv_{h_4} = \{(h_2, S_2, \{1,1\})\}$

? Emission de m_4 de $h_3(S_2) \rightarrow h_4(S_3)$

$h_3 : id_{mess_E} = 2,$ $\qquad S_2 : IdSend_{S_2} = 2$

$f^{i}_{h_3} = 11, depend_{h_3} = true$

$Envoyer\,(m_4, 2, true, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix})$

$f_{h_3} = \begin{bmatrix} (1, h_4) \\ (2, , h_4) \\ 0 \end{bmatrix},$

$LastSend_{h_3} = \{(h_4, 2, 11)\}$

? Réception de $(m_4, 2, true, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix})$ par $h_4(S_3)$

Dans ce cas $depend_{h_3} = true$ donc le message soit il dépend d'un autre message qui est interne au site mobile h_3 et qui a été envoyé avant m_4 au site mobile h_4, soit il dépend d'autres messages envoyés par les autres sites mobiles. Puisque $id_{LastSend} = 0$ alors le message m_4 est en relation causale avec d'autres messages envoyés par les autres sites mobiles avant lui au site h_4. En vérifiant f_{h_3} on trouve que m_4 dépend du premier message envoyé par S_1 par conséquent le message m_4 ne sera délivré que si h_4 a reçu le 1er message de S_1. La liste $LastRcv_{h_4} = \{(h_2, S_2, \{1,1\})\}$ ne contient pas l'information sur l'arrivée de ce message ce qui signifie que h_4 ne l'a pas encore reçu. Donc, la livraison de m_4 sera retardée jusqu'à l'arrivée du 1er message de S_1 et m4 sera mis dans la file d'attente *AtFile*.

? Réception de $(m_1, 1, false, 0, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix})$ par $h_4(S_3)$

Puisque $depend_{h_1} = false$ alors le message sera délivré au site h_4, et les mises à jour suivantes seront faites :

$h_4 : id_{mess_E} = 2, id_{mess_D} = 2,$

$f_{h_3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$

$LastRcv_{h_4} = \{(h_2, S_2, \{1,1\}), (h_1, S_1, \{1,1\})\}$

En vérifiant la file d'attente *AtFile*, on trouve qu'il existe un message (m_4) qui attend l'arrivée de m_1 donc ce message sera délivré. La livraison de m4 entraîne les mises à jour suivantes :

$$h_4 : id_{mess_E} = 3, id_{mess_D} = 3,$$

$$f_{h_4} = \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix} \longrightarrow f_{h_4} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$LastRcv_{h_4} = \{(h_2, S_2, \{1,1\}), (h_1, S_1, \{1,1\}), (h_3, S_2, \{2,2\}\}$$

L'information (1, h_4) a été supprimée de f_{h_4} car ce message est délivré et on a plus besoin de transmettre cette information avec les prochains messages.

5.1.4 Preuve de correction

- Condition de livraison

Un message m envoyé par un site mobile h_i (S_i) est délivré à un site mobile h_j (S_j) seulement si la condition suivante est vérifiée :

$$\begin{bmatrix} [depend_m = false] \\ OR \begin{bmatrix} [depend_m = true] \\ AND \begin{bmatrix} [id_{LastSend_m} \neq 0 \; and \; \exists (h_i, \_, SD_i) \in LastRcv_{h_j} \; tel \; que \; id_{LastSend_m} \in SD_i] \\ OR \begin{bmatrix} id_{LastSend_m} = 0 \; and \; \forall k = \overline{1,n}, tel \; que \; \exists (idf, h_j) \in f_m[k] \\ \exists (\_, S_k, SD_k) \in LastRcv_{h_j} \; tel \; que \; idf \in SD_k \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Soient m_1 et m_2 deux messages envoyés à h_j, respectivement par les sites mobiles h_k et h_i, donc :

$$Send(m_1) \longrightarrow Send(m_2) \Leftrightarrow (depend_{m_2} = true)$$

$$AND \begin{bmatrix} (id_{LastSend_{m_2}} = id_{m_1}) & // si \quad h_k = h_i \\ OR(\exists (id_{m_1}, h_j) \in f_{m_2}[k]) & // si \quad h_k \neq h_i \end{bmatrix}$$

- Propriété de sûreté

Sûreté : *l'ordre causal n'est jamais violé.*

Soit deux messages m_1 et m_2 tels que $m_1 \longrightarrow m_2$ et $\nexists m \mid m_1 \longrightarrow m \longrightarrow m_2$; c'est-à-dire l'émission de m_1 précède immédiatement l'émission de m_2. Deux cas sont à considérer :

? *m_1 et m_2 sont émis par le même site mobile h_i*

Soit (Id_{m_1}, h_j) et (Id_{m_2}, h_j) les identités des événements respectivement d'émission et m_1 et m_2 au niveau de h_i. Supposons que la station de base destinataire S_j où réside h_j délivre m_2 avant m_1. Ceci signifie que la condition de livraison pour m_2 est $(depend_{m_2} = true) AND (id_{LastSend_{m2}} < id_{m_1})$ et pour m_1 est $(depend_{m_1} = true) AND (id_{LastSend_{m1}} = id_{m_2})$. Cela implique que l'identificateur du message m_2 soit plus petit que l'identificateur du message m_1 et ceci ne peut être vrai

que si m_2 est envoyé avant m_1. Ce qui conduit à une contradiction avec l'hypothèse : m_2 a été délivré avant m_1.

? *m_1 et m_2 sont émis par deux sites mobiles distincts $h_k(S_k)$ et $h_i(S_i)$ respectivement*

Soient (Id_{m_1}, h_j) et (Id_{m_2}, h_j) les identités des événements d'émission de m_1 et m_2. Supposons que m_2 ait été délivré avant m_1 par la station de base destinataire S_j au site mobile h_j. Donc, $m_1 \longrightarrow m_2 \Rightarrow h_i(S_i)$ a sûrement reçu un message m tel que $(Id_{m_1}, h_j) \in f_m[k]$ avant l'émission de $m_2 \Rightarrow (Id_{m_1}, h_j) \in f_{m_2}[k]$.

La livraison de m_2 par $S_j(h_j) \Leftrightarrow \exists(\_, S_k, Id_{m_1}) \in LastRcv_{h_j}$ et ceci ne peut être vérifié que si S_j a délivré m_1 avant m_2. Ce qui conduit aussi à une contradiction.

- Propriété de vivacité

Vivacité : *Chaque message est délivré au bout d'un temps fini.*

? Si $m_1 \longrightarrow m_2$ alors on a deux possibilités : soit $id_{LastSend_{m2}} = Id_{m_1}$ si m_1 et m_2 sont émis par le même site mobile, soit $(Id_{m_1}, h_j) \in f_{m_1}[k]$ si m_1 et m_2 sont émis par deux sites différents.

Donc la livraison de m_2 ne peut être différée que par d'autres messages qui ne sont pas encore délivrés à h_j. Pour prouver que ce délai d'attente est fini, nous procédons comme suit : un message m_2 reçu par une station de base S_j ne peut être délivré au site mobile destinataire h_j que si la condition de livraison est vérifiée. Nous considérons tous les messages qui ne sont pas encore délivrés à h_j de S_j. La relation de *happened before* peut être utilisée pour définir un ordre partiel sur les événements d'émission de ces messages non encore délivrés. Soit *m'* un de ces messages dans l'ordre partiel dont l'émission n'a pas de prédécesseur. Puisque *m'* n'a pas été délivré à h_j à sa réception, alors la condition suivante doit être vraie $depend_{m'} = true$ et ceci n'est valable que si *m'* est causalement lié à un autre message qui a été envoyé avant l'émission de *m'* soit par le même site ou un site différent. Ce qui conduit à une contradiction avec l'hypothèse que *m'* n'a pas de prédécesseur.

5.1.5 Module *handoff*

Pour assurer une gestion fiable de l'ordre causal des messages en environnement mobile certaines étapes doivent être prises en charge par le module "*handoff*".

Nous supposons qu'un site mobile h_i résidant dans la cellule de la station de base S_i se déplace vers la cellule de la station de base S_j. De plus, nous assumons que chaque station de base utilise un tableau, noté *cellule*, avec une entrée pour chaque site mobile. La $k^{éme}$ entrée de *cellule*, $cellule_i[k]$ représente la connaissance de S_i sur la localisation courante du site mobile h_k.

La première étape établie par h_i est d'envoyer le message *register*(*hi*, *Si*) à S_j pour l'informer de sa présence et de l'identité de son ancienne station de base. En recevant ce message, S_j envoie *handoff_begin*(*hi*) à S_i pour marquer le déclenchement de la procédure de handoff. La station de base S_i, en recevant *handoff_begin*(*hi*), transmet les structures de données correspondantes à h_i vers S_j et ceci est exprimé par l'envoi du message $Data(h_i, id_{mess_E}, id_{mess_D}, LastSend_{h_i}, LastRcv_{h_i}, f_{h_i}, AtFile_{h_i})$. Ensuite, S_i diffuse le message

NewLocation(h_i, S_j) à toutes les stations de base à l'exception de S_i et S_j pour les mettre au courant du nouvel emplacement de h_i.

Après la transmission des structures de données de hi vers S_j, si un message destiné à h_i est reçu par S_i, il sera redirigé vers S_j sans le traiter. Sa livraison sera à la charge de S_j après avoir reçu le message $Data(h_i, id_{mess_E}, id_{mess_D}, LastSend_{h_i}, LastRcv_{h_i}, f_{h_i}, AtFile_{h_i})$ à partir de S_i.

La réception du message *NewLocation*(h_i, S_j) par les autres stations de base entraîne la mise à jour de leurs tableaux *cellule* pour exprimer le nouvel emplacement de h_i, et les nouveaux messages destinés à hi seront envoyés vers sa nouvelle station de base S_j.

La procédure du handoff se termine quand Sj reçoit toutes les données transmises par Si. La terminaison de la procédure du handoff sera exprimée par l'envoi du message *handoff_over*(*hi*) par S_i à S_j, et dés ce moment S_j se charge de maintenir l'ordre causal des messages destinés à h_i.

5.1.6 Performance et complexité

Dans ce protocole, toutes les structures de données sont localisées sur les stations de base. Ces structures de données sont indépendantes du nombre de sites mobiles du système, ce qui rend le protocole évolutif relativement au nombre des sites mobiles. L'exécution du protocole est entièrement à la charge des stations de base alors que le rôle des sites mobiles est réduit à l'émission et la réception des messages. Les messages transmis sur les liaisons sans fil ne transportent aucune information de contrôle ce qui contribue à minimiser l'utilisation de la bande passante des liaisons sans fil.

Le coût de communication du protocole est de l'ordre de $\sum_{i=1}^{n_s} l_i$, où n_s est le nombre des stations de base dans le système, et l_i est le nombre de messages dans l'entrée $f[i]$. l_i représente le nombre de messages émis par S_i et dont la livraison n'est pas encore assurée. Malgré que ce nombre n'est pas borné mais puisque la réception de chaque message entraîne la mise à jour de cette horloge par la fonction *BUILD* alors nous pouvons assumer que ce nombre reste réduit et que $\sum_{i=1}^{n_s} l_i << n_s \times n_h$, où n_h est le nombre des sites mobiles dans le système.

La taille maximale que peut atteindre $LastSend_{h_i}$ est égale à $n_h - 1$ et ceci sera le cas quand h_i émet au moins un message à chaque h_j. Par contre, l'inconvénient de cet algorithme est l'évolution non bornée de $LastRcv_{h_i}$ qui découle des caractéristiques des séquences de dépendance. Mais nous pouvons pallier à ce problème en rajoutant une procédure pour éliminer les identificateurs des messages dont l'information sur leur livraison n'est plus nécessaire.

Chaque message émis par un site mobile est immédiatement délivré si la condition de livraison et vérifiée. L'information de contrôle transportée par chaque message émis reflète la relation de causalité effective entre les messages, telle que perçue par leurs sites mobiles. Ainsi, la livraison des messages n'est jamais inhibée comme dans les autres protocoles.

6 Conclusion

La caractérisation des dépendances causales entres les activités distribuées joue un rôle important dans l'analyse des systèmes distribués. Plusieurs mécanismes ont été proposés pour tracer cette causalité dans les systèmes distribués classiques. Le mécanisme le plus utilisé était l'estampillage par horloges vectorielles car il offre la possibilité de tracer l'ordre partiel entre les événements tout en gardant l'information de concurrence entre ces derniers. Cependant, dans les systèmes distribués où le passage efficace des messages avec les sémantiques appropriées est de grande importance, la violation de l'ordre de réception des messages n'est pas permise. Mais se baser seulement sur les mécanismes d'estampillage pour déterminer cette violation est impossible. Par conséquent, le concept d'ordre causal a été introduit et plusieurs protocoles de mise en œuvre de ce concept ont été développés pour les environnements distribués.

Mais l'apparition des systèmes mobiles avec de nouvelles caractéristiques tels la mobilité et la communication sans fil fait qu'il est nécessaire de réviser les mécanismes et les protocoles des systèmes distribués classiques. Notre travail a été consacré aux problèmes des mécanismes d'estampillage et des protocoles d'ordre causal des messages. Nous avons fait une étude sur les mécanismes d'estampillage proposés pour ces systèmes: les séquences de dépendances et les horloges hiérarchiques, et nous avons essayé de bénéficier des avantages de ces deux mécanismes pour développer un protocole d'ordre causal des messages pour l'environnement mobile. Le protocole proposé est caractérisé par l'élimination de l'effet d'inhibition dans la livraison des messages tout en maintenant une taille minimale de l'information de contrôle transmise avec chaque message. Cependant, l'inconvénient de notre protocole est l'évolution non bornée de la file $LastRcv_{h_i}$ qui découle des caractéristiques des séquences de dépendances.

Les perspectives possibles à ce travail sont : de rajouter une procédure pour éliminer les identificateurs des messages à partir de $LastRcv_{h_i}$ dont l'information sur leur livraison n'est plus nécessaire dans le but de minimiser sa taille; et l'adaptation de ce protocole pour les problèmes du *Multicast* et du *Broadcast* avec gestion de la vue de groupe.

7 Références

[1] J. Ioannidis, D. Duchamp, et G. Q. Maguire. *Ipbased Protocols for Mobile Internetworking*. In Proc. of ACM SIGCOMM Symposium on Communication, Architectures and Protocols, pages 235–245, September 1991.

[2] N. Badache. *La Mobilité dans les Systèmes Répartis.* Publication Interne IRISA n° 962, Octobre 1995. ftp://ftp.irisa.fr /techreports/1995/PI–962.ps.gz.

[3] N. Aleb. *Spécification des Systèmes en Environnement Mobile à l'aide du Pi-calcul.* Thèse de Magister, USTHB, Octobre 2000.

[4] E. Pitoura, B. Bhargava. *Dealing with Mobility: Issues and Research Challenges*. Technical Report TR-93-070, Department of Computer Science, Purdue University, November 1993.

[5] E. Pitoura, B. Bhargava. *Dealing with Mobility: Issues and Research Challenges*. Technical Report TR-93-070, Department of Computer Science, Purdue University, November 1993.

[6] T. Imielinski et B. R. Badrinath. *Mobile Wireless Computing: Solutions and Challenges in Data Management.* CACM, vol. 37, n° 10, pages 18-28, October 1994.

[7] E. Pitoura, B. Bhargava. *Building Information Systems for Mobile Environments*. Department of Computer Sciences, Purdue University, Third International Conference on Information and Knowledge Management, pages 371-378, November 1994.

[8] M. Weiser. *Les Réseaux Informatiques de l'An 200*0. Pour la Science, n° 169, pages 72-84, novembre 1991.
[9] F. Bennett, T. Richardson, A. Harter. *Teleporting Applications Mobil*e. Olivetti Research Laboratory, IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US, December 1994.
[10] B. R. Badrinath, A. Acharya, T. Imielinski. *Impact of Mobility on Distributed Computations*. Department of Computer Science, Rutgers University, ACM Operating System Review, vol. 27, n° 2, pages 15-20, April 1993.
[11] L. Lamport. *Time, Clocks and the Ordering of Events in a Distributed System*. Communications of the ACM, 21(7) : 558-565, July 1978.
[12] Colin Fidge. *Logical Time in Distributed Computing Systems*. IEEE Computer, 24(8) :28-33, August 1991.
[13] R. Schwarz and F. Mattern. *Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail*. Distributed Computing, 3(7) :149-174, 1994.
[14] Michel Raynal. *About Logical Clocks for Distributed Systems*. Publication Interne IRISA n° 607, Octobre 1991.
[15] Ravi Prakash and Mukesh Singhal. *Dependency Sequences and Hierarchical Clocks: Efficient Alternatives to Vector Clocks for Mobile Computing Systems*. Wireless Networks, (3): 349-360, 1997. Also in Mobicom96.
[16] K. Birman, T. Joseph. *Reliable Communication in the Presence of Failures*. ACM Transactions on Computer Systems, 5(1): 47-76, February 1987.
[17] K. Birman, R. (eds.) Van Renesse. *Reliable Distributed Computing with the ISIS Toolkit*. IEEE Computer Society Press, 1994.
[18] K. Birman, T. Joseph. *Exploiting replication in Distributed Systems*. In S. Mullender, ed., Distributed Systems, (ACM, New York, 1990).
[19] A. Alagar, S. Venkatesan. *An Optimal Algorithm for Distributed Snapshots with Causal Message Ordering.* Information Processing Letters, 50, 311-316, 1994.
[20] M. Ahamad, P. Hutto, R. John. *Implementing and Programming Causal Distributed Memory*. Proceedings of the 11th Intl. Conf. on Distributed Computing Systems, 274-281, 1991.
[21] K. Ravindran, B. Prasad. *Communication Structures and Paradigms for Distributed Conferencing Applications*. Proc. 12th Intl. Conf. on Distributed Computing Systems, May 1992.
[22] F. Adelstein, M. Singhal. *Real-Time Causal Message Ordering in Multimedia Systems,* Proc. 15th Intl. Conf. on Distributed Computing Systems, May -june1995.
[23] P. Kearns, B. Koodalattupuram. *Immediate Ordered Service in Distributed Systems*. Proc. 9th Intl. Conf. on Distributed Computing Systems, Newport Beach, California, 611-618, June 1989.
[24] K. Birman, A. Schiper, P. Stephenson. *Lightweight Causal and Atomic Multicast*. ACM Transactions on Computer Systems, 9(3): 272-314, August 1991.
[25] L. L. Peterson, N. C. Buchholz, R. D. Schlichting. *Preserving and Using Context Information in Interprocess Communication*. ACM Trans. on Computer Systems, 1989.
[26] A. Schiper, J. Eggli, A. Sandoz. *A New Algorithm to Implement Causal Ordering.* Proc 3rd Int. Workshop on Distributed Algorithms, 219-232, 1989, in Lecture Notes in Computer Science 392, Springer-Verlag.
[27] M. Raynal, A. Sciper, S. Toueg. *The Causal Ordering Abstraction and a Simple Way to Implement It*. Information Processing Letters, 39(6): 343-350, 1991.
[28] N. Badache. *Ordre Causal et Tolérance aux Défaillances en Environnement Mobile*. Thèse de Doctorat d'état, Octobre 1998.
[29] S. Alagar, S. Venkatesan. *Causally Ordered Message Delivery in Mobile Systems*. In: Proc. Workshop on Mobile Computing Systems and Applications, 169-174, December 1994.
[30] L. H. Yen, T. L. Huang, S. Y. Hwang. *A Protocol for Causally Ordered Message Delivery in Mobile Computing Systems*. Mobile Networks and Applications, 1997.
[31] R. Prakash, M. Raynal, M. Singhal. *An Efficient Causal Ordering Algorithm for Mobile Computing Environments*. In Proc. Of the 16th Intl. Conf. on Distributed Computing Systems, 1996.

[32] C. Skawratananond, N. Mittal, V. K. Garg. *A Lightweight Algorithm for Causal Message Ordering in Mobile Computing Systems*. 1999.

Introduction

Un calcul distribué se compose d'un ensemble de processus qui coopèrent entre eux pour accomplir un but commun. Une caractéristique principale de ces calculs est que les processus ne partagent pas une mémoire globale commune et qu'ils communiquent seulement par échange de messages via un réseau de communication. De plus, les délais de transfert des messages sont finis mais non bornés. Ce modèle de calcul définit ce qui est appelé *système distribué asynchrone.* L'exécution asynchrone des processus et le caractère fini et non borné des délais de communication confèrent aux systèmes répartis un comportement *non déterministe.*

Pour une compréhension appropriée d'un programme distribué et de son exécution, il est important de déterminer la relation causale et temporelle entre les événements qui se produisent durant le calcul. Par exemple, il est souvent le cas que deux événements concurrents ou causalement indépendants peuvent se produire dans n'importe quel ordre, donnant probablement des résultats différents dans chaque cas. Ceci indique que le non déterminisme est strictement lié à la concurrence. En fait, les effets de concurrence et du non déterminisme jouent un rôle important dans le processus d'analyse, de monitoring, de debugging, et de visualisation du comportement d'un système distribué. Mais la caractéristique d'absence d'un référentiel de temps global dans un système distribué rend plus difficile l'identification des activités concurrentes dans les calculs distribués et par conséquent plus complexe la conception, la vérification et l'analyse des programmes répartis.

Dans un système distribué et en absence d'un référentiel de temps global, les événements d'un calcul distribué ne peuvent être ordonnés que sur la base de la relation de précédence causale (*happened before*), notée ? , définie par Lamport. En d'autres termes, deux événements sont contraints de se produire dans un certain ordre seulement si l'occurrence du premier peut affecter celle du second. Afin de dater (estampiller) les événements, des horloges logiques de différents types ont été définies afin de rendre compte de la relation de dépendance (i.e. la relation *happened before*) entre les événements. Parmi ces mécanismes, on trouve : les *horloges scalaires* de Lamport dont le temps est représenté par des entiers positifs ; les *horloges vectorielles* où le temps est représenté par un vecteur d'entiers positifs.

Les évolutions techniques dans le développement des ordinateurs portables et le déploiement rapide de la technologie des réseaux sans fil fournissent la base pour un nouvel environnement de calcul, appelé *environnement mobile* ou *nomade*.

En utilisant conjointement ces ordinateurs portables et les médiums de communication sans fil, (ondes radiofréquence, ondes lumineuses), il devient possible de rester connecté á son réseau habituel et de communiquer avec les autres machines mobiles tout en se déplaçant. Les utilisateurs sont á même de communiquer, d'accéder á l'information n'importe où et n'importe quand.

Ces développements ont cependant un prix, les challenges techniques qui permettront de mettre en œuvre l'environnement de calcul mobile n'ont rien de trivial. Les nouvelles contraintes introduites par l'environnement mobile font qu'il est nécessaire de réviser les mécanismes et les algorithmes des systèmes répartis classiques (exclusion mutuelle, ordonnancement causal, …).

Les solutions aux problèmes de communication et de synchronisation dans les systèmes distribués conventionnels sont souvent basées sur des structures logiques telles que l'anneau, l'arbre, le graphe complet,…Malheureusement ces structures ne sont plus adaptées à des topologie physiques caractérisées par de fréquents déplacements ainsi que des connexions/déconnexions de sites. Les capacités de calcul et de stockage relativement faibles des unités mobiles, la faible bande passante des liaisons sans fil et leurs contraintes en énergie exigent une plus grande maîtrise des coûts de calcul et de communication induits par l'exécution d'algorithmes répartis.

Un des problèmes fondamentaux dans les systèmes répartis est le contrôle de l'indéterminisme dû à la nature asynchrone des sites et du médium de communication. Dans le but de réduire l'indéterminisme du médium de communication, Birman et Joseph ont proposé le concept de livraison des messages selon *l'ordre causal*.

L'ordre causal des messages garantit que l'ordre de livraison des messages ne viole pas la causalité dans le système. Le processus diffère la livraison d'un message donné jusqu'à l'arrivée de tous ses prédécesseurs causaux. L'ordre causal est une propriété importante de la communication dans les applications où les interactions humaines sont fréquentes. Ces applications telles que les services d'accès à l'information, le commerce et les systèmes de téléconférence sont caractéristiques des systèmes répartis en environnement mobile.

Dans cette thèse, nous nous sommes intéressés à deux problèmes essentiels pour faciliter la construction des applications réparties en environnement mobile: les mécanismes d'estampillage et l'ordre causal des messages. Cette thèse est composée de quatre chapitres:

Le chapitre 1 introduit l'environnement mobile, et les principaux concepts liés á ce nouvel environnement. Du fait que les systèmes mobiles ont émergé comme un cas distinctif des systèmes distribués classiques, ce chapitre présente d'abord une brève définition d'un système distribué classique. Ensuite, nous introduisons l'architecture des systèmes distribués avec sites mobiles et leurs différents modes de fonctionnement, ainsi que quelques caractéristiques de ces environnements. En dernier lieu, nous essayons de caractériser l'impact de la mobilité et des caractéristiques physiques des unités support de calcul mobile sur les applications, algorithmes et protocoles des systèmes répartis.

Le chapitre 2 présente les mécanismes d'estampillage en environnement mobile. Il commence d'abord par une définition du concept de temps logique et da la relation de dépendance causale dans un système distribué classique. Nous passons en revue les mécanismes d'estampillage proposés pour implémenter ce temps et par conséquent capturer les dépendances causales dans les systèmes distribués classiques: les horloges scalaires et les horloges vectorielles. Une présentation de l'inadéquation de ces mécanismes pour les systèmes mobiles est faite et par conséquent, une présentation des alternatives existantes dans ce cadre: les séquences de dépendance et les horloges hiérarchiques, est aussi faite.

Le chapitre 3 présente le problème d'ordre causal des messages et les protocoles proposés pour résoudre ce problème en environnement mobile.

L'idée de "*l'ordre causal des messages*" a été introduite par Joseph et Birman dans le cadre du projet ISIS et permet d'astreindre la réception des messages au même ordre que celui de leur émission. Plusieurs protocoles sont proposés pour mettre en œuvre l'ordre causal des messages dans les systèmes distribués classiques. Cependant, les contributions pour l'adaptation de ces protocoles pour l'environnement mobile restent modestes.

La construction d'algorithmes répartis pour les systèmes en environnement mobile nécessite de reporter les coûts de calcul, de communication et de stockage sur le réseau statique. Les unités mobiles et leurs liens de communication sans fil doivent être sollicités le moins possible. Toutefois cette approche, adoptée principalement pour pallier aux faibles capacités de calcul des unités mobiles et de la bande passante des liens de communication sans fil, peut engendrer un délai d'inhibition des messages. L'exécution d'un protocole par des stations fixes pour le compte des unités mobiles conduit à une vision différente sur l'ordre des événements entre ces dernières et les stations fixes du système. Une interprétation inexacte de l'ordre des événements peut introduire un délai d'inhibition dans les protocoles d'ordre causal des messages en environnement mobile. La livraison d'un message par une station de base peut être retardée parce que de son point de vue ce message n'est pas délivrable alors qu'il peut l'être d'un point de vue des unités mobiles locales.

Le chapitre 4 présente notre contribution pour la proposition d'un protocole d'ordre causal des messages en environnement mobile. L'objectif principal de ce protocole est d'étudier l'applicabilité des nouveaux mécanismes d'estampillage proposés pour les environnements mobiles (les séquences de dépendance et les horloges hiérarchiques) pour l'implémentation d'un protocole qui résout le problème d'ordre causal en environnement mobile. Nous citons d'abord les structures de données utilisées dans notre protocole. Par la suite, nous présentons le module statique du protocole ainsi que des exemples illustratifs. Le paragraphe suivant sera consacré à la preuve de correction de la solution ; nous prouvons la sûreté et la vivacité de notre solution. Finalement, nous présentons le module *handoff* et nous terminons par une étude de la performance et de la complexité du protocole. Le protocole que nous avons proposé se caractérise par l'élimination de l'effet d'inhibition.

Chapitre 1

Généralités sur les environnements mobiles

1.1 Introduction

Les évolutions techniques dans le développement des ordinateurs portables et le déploiement rapide de la technologie des réseaux sans fil fournissent la base pour un nouvel environnement de calcul, appelé *environnement mobile* ou *nomade*.

En utilisant conjointement ces ordinateurs portables et les médiums de communication sans fil, (ondes radiofréquence, ondes lumineuses), il devient possible de rester connecté á son réseau habituel et de communiquer avec les autres machines mobiles tout en se déplaçant soi-même. Les utilisateurs sont á même de communiquer, d'accéder á l'information n'importe où et n'importe quand.

Ces développements ont cependant un prix, les challenges techniques qui permettront de mettre en œuvre l'environnement de calcul mobile n'ont rien de trivial [10]. Les nouvelles contraintes introduites par l'environnement mobile font qu'il est nécessaire de réviser les algorithmes des systèmes répartis classiques (exclusion mutuelle, ordonnancement causal, …).

Ces contraintes découlent d'une part de la mobilité, qui induit que le travail ne s'exécute plus seulement sur un réseau statique mais aussi avec des machines qui se déplacent, ce qui complique considérablement les exécutions réparties. Par exemple, les structures logiques (anneau, arbre,…), exploitées traditionnellement par les algorithmes répartis [1] ne peuvent plus être utilisées efficacement dans un environnement où les sites changent de place fréquemment [2, 3, 4]. D'autre part, l'architecture des systèmes supportant des ordinateurs mobiles est différente de celle des réseaux fixes. Il peut en effet être nécessaire de supporter la notion cellule de communication sans fil (zone au sein de laquelle la communication sans fil est possible), de station support (station du réseau fixe qui a la charge d'une cellule de communication sans fil), etc. [5]

De plus, les communications sans fil créent des problèmes de déconnexion, de faible largeur de la bande passante ou de besoins de communication extrêmement variables. Enfin, la nécessité de portabilité du matériel induit de nombreuses limitations et oblige á gérer la consommation d'énergie, les problèmes de stockage, etc. L'impact du nouvel environnement de calcul est donc important, tant au niveau du réseau que des applications et des systèmes.

Ce chapitre a pour but de présenter l'environnement mobile, et les principaux concepts liés á ce nouvel environnement. Du fait que les systèmes mobiles sont émergés comme un cas distinctif des systèmes distribués classiques, le chapitre va présenter

d'abord une brève définition d'un système distribué classique. Ensuite, nous introduisons l'architecture des systèmes distribués avec sites mobiles et leurs différents modes de fonctionnement, ainsi que quelques caractéristiques de ces environnements. En dernier lieu, nous essayons de caractériser l'impact de la mobilité et des caractéristiques physiques des unités support de calcul mobile sur les applications, algorithmes et protocoles des systèmes répartis.

1.2 Modèle de système distribué standard

Un système distribué est composé d'un ensemble fini de sites autonomes géographiquement dispersés qui sont interconnectés á travers un réseau de communication filaire. Chaque site possède une mémoire locale et un emplacement de stockage stable et exécute un ou plusieurs processus. Sans perte de généralité, on assume qu'il n'y a qu'un seul processus par site. Les états locaux de tous les processus sont assumés d'être disjoints, ce qui signifie que les processus ne partagent pas de mémoire commune, et ils communiquent seulement par échange de messages á travers les canaux du réseau de communication. C'est la multiplicité des sites connectés á travers un réseau qui rend le *système distribué*.

Il n'y a aucune supposition sur la vitesse relative des processus et sur les délais de transfert des messages qui sont finis mais imprévisibles. Cette absence des limites temporelles rend le système distribué *asynchrone*.

Dans un système distribué, le comportement de chaque processus se compose des changements de l'état local, et des émissions de messages aux autres processus. Ces actions sont complètement déterminées par un algorithme local, qui détermine aussi la réaction envers les messages reçus. L'exécution concurrente et coordonnée de tous les algorithmes distribués forme un *calcul distribué*. De plus, une propriété essentielle des systèmes distribués est l'absence d'une horloge globale ou d'horloges locales parfaitement synchronisées.

Les occurrences d'actions réalisées par les algorithmes locaux sont dites *événements*. D'un point de vue abstrait, un calcul distribué peut être décrit par les types et l'ordre relatif des événements produits dans chaque processus. Dans un système distribué, deux activités générales peuvent prendre place : les *activités locales* qui sont réalisées de manière indépendante par chaque processus et les *activités de synchronisation* durant lesquelles deux ou plusieurs processus agissent entre eux et échangent des messages.

Les deux activités générales peuvent être décomposées en trois types d'événements : les *événements d'émission*, les *événements de réception*, et les *événements internes*. Un événement d'émission reflète le fait qu'un message ait été émis ; un événement de réception dénote la réception d'un message en même temps que l'état local change suivant le contenu du message. Les événements internes affectent seulement l'état local du processus. Un événement interne á un processus est causalement dépendant de tous les autres événements internes antérieurs.

Le médium de communication ne délivre pas les messages dans le même ordre dans tous les sites, car les messages des différents sites se déplacent á travers différents chemins. Donc, il est nécessaire pour chaque site d'imposer certain genre d'ordre consistant sur les messages afin que les applications dans les sites puissent avoir les messages délivrés dans le même ordre, sans se soucier de l'ordre dans lequel les messages arrivent au site.

1.3 Modèle de système avec sites mobiles

Le terme *"mobile"* implique être capable á se déplacer tout en restant connecté au réseau [6]. Le modèle de système distribué comprenant des sites mobiles est composé de deux ensembles d'entités distincts : les sites fixes d'un réseau de communication filaire classique et des sites mobiles [7]. L'entité qui relie les deux parties, fixe et mobile, du réseau en transformant les signaux entre les deux médiums est appelée *station de support mobile (MSS – Mobile Support Station)*. Les *MSS*$_s$ sont munies d'interface de communication sans fil et la zone géographique couverte par une *MSS* est appelée *cellule*. Tout *site mobile (MH – Mobile Host)* est initialement enregistré dans une et une seule *MSS* appelée *station d'enregistrement (ou Home base)* [8]. Ceci n'empêche pas le libre déplacement de celui-ci entre les différentes cellules en qualité de 'visiteur'. Un site mobile ne peut communiquer avec d'autres entités du réseau que par l'intermédiaire d'une *MSS*.

Un site mobile peut se déplacer d'une cellule á une autre cellule. Dans un tel cas, la *MSS* de l'ancienne cellule cède les responsabilités de communication du *MH* á la *MSS* de la nouvelle cellule.

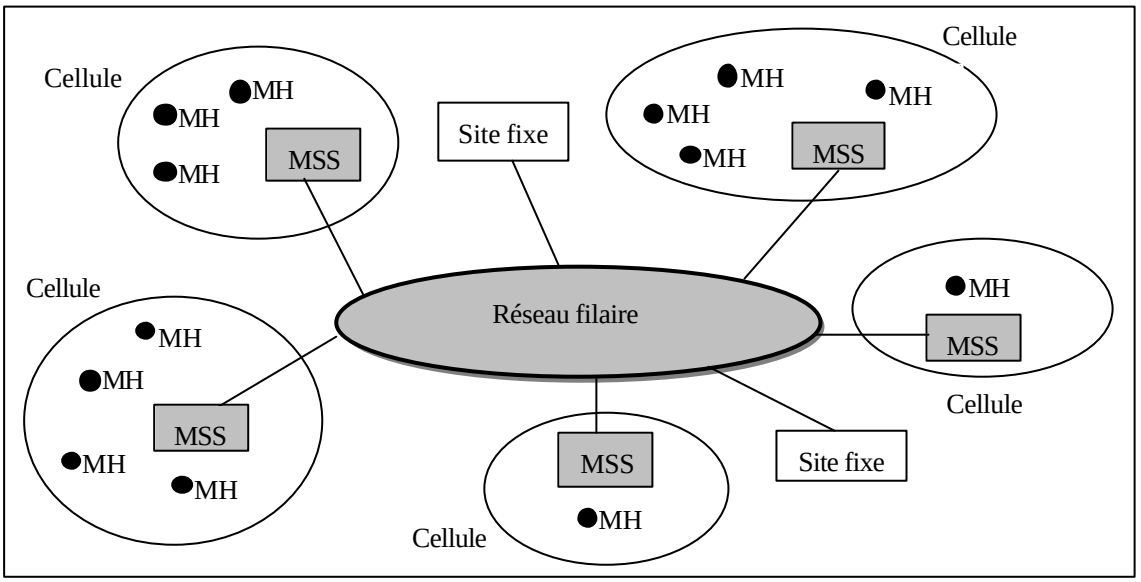


Figure 1.1 – *Architecture d'un système mobile*

1.4 Modes de fonctionnement des mobiles

Dans un système réparti sans ordinateur mobile, une machine ne peut travailler que dans deux modes différents, soit connectée au réseau, soit totalement déconnectée. Par contre en environnement mobile, il existe des degrés variés de déconnexion [9]. Le degré de déconnexion est relatif à la largeur de bande passante disponible allouée à la liaison sans fil.

Un site mobile dispose donc de davantage de modes de travail. Actuellement, nous dénombrons quatre modes de fonctionnement différents employés sur les sites mobiles, nous les définissons dans ce qui suit :

1.4.1 Mode connecté

Dans ce cas de figure, le mobile dispose d'une connexion normale au réseau, à la manière d'une station classique. La connexion est réalisée par une interface de communication sans fil, qui fournit des débits plus faibles qu'une liaison câblée.

1.4.2 Mode partiellement connecté

Le mobile ne dispose plus pour communiquer avec le réseau que d'un lien à faible largeur de bande (connexion faible ou déconnexion partielle). Cette perte de capacité de la bande passante peut être due à des perturbations, à des surcharges de la station de base qui gère les communications des mobiles se trouvant dans sa cellule.

1.4.3 Mode veille

Ce mode est utilisé par les mobiles pour préserver leurs ressources énergétiques. La vitesse de l'horloge est alors réduite et les exécutions des applications de l'utilisateur sont suspendues. La liaison avec le réseau est malgré tout maintenue, le mobile n'envoie plus de messages, mais peut encore en recevoir et repasser ainsi en mode connecté.

1.4.4 Mode déconnecté

Un site mobile peut bien sûr se trouver totalement déconnecté du réseau, à la fois parce qu'il n'y est plus physiquement relié ou parce qu'il est impossible de maintenir une connexion sans fil (volontairement, du fait de fortes interférences ou de surcharges momentanées, par exemple).

Les déconnexions totales ou partielles étant très fréquentes pour un mobile connecté via une liaison sans fil, celles-ci ne doivent pas être traitées comme des pannes. Une machine mobile doit être capable de continuer ses exécutions même avec une connexion au réseau ou plus de connexion du tout.

Il est á noter que les déconnexions dans un environnement mobile, contrairement á un environnement réparti classique, peuvent le plus souvent être détectées et un protocole spécifique peut donc être mis en œuvre pour les prendre en compte. Le protocole de déconnexion doit être exécuté avant que la machine mobile ne soit physiquement détachée du réseau fixe. Ce protocole est tenu d'assurer que suffisamment d'informations sont présentes localement pour garantir au mobile son autonomie durant la déconnexion.

Le protocole de déconnexion partielle a pour charge de préparer le mobile á exécuter des opérations dans un mode où toutes les communications avec le réseau fixe seront aussi réduites que possible. Finalement, les protocoles de *hand-off* permet á un mobile de sortir des bornes d'une cellule pour entrer dans une autre, tout en conservant la connexion et en mettant á jour des informations sur le réseau fixe (la localisation du mobile, par exemple) et sur le mobile (le nom de la station support locale). Des informations d'état se référant au mobile peuvent, á ce moment, être transférées vers la station support de la nouvelle cellule.

La figure suivante résume les différents modes d'opération supportés par un mobile ainsi que les différentes transitions possibles entre ces états :

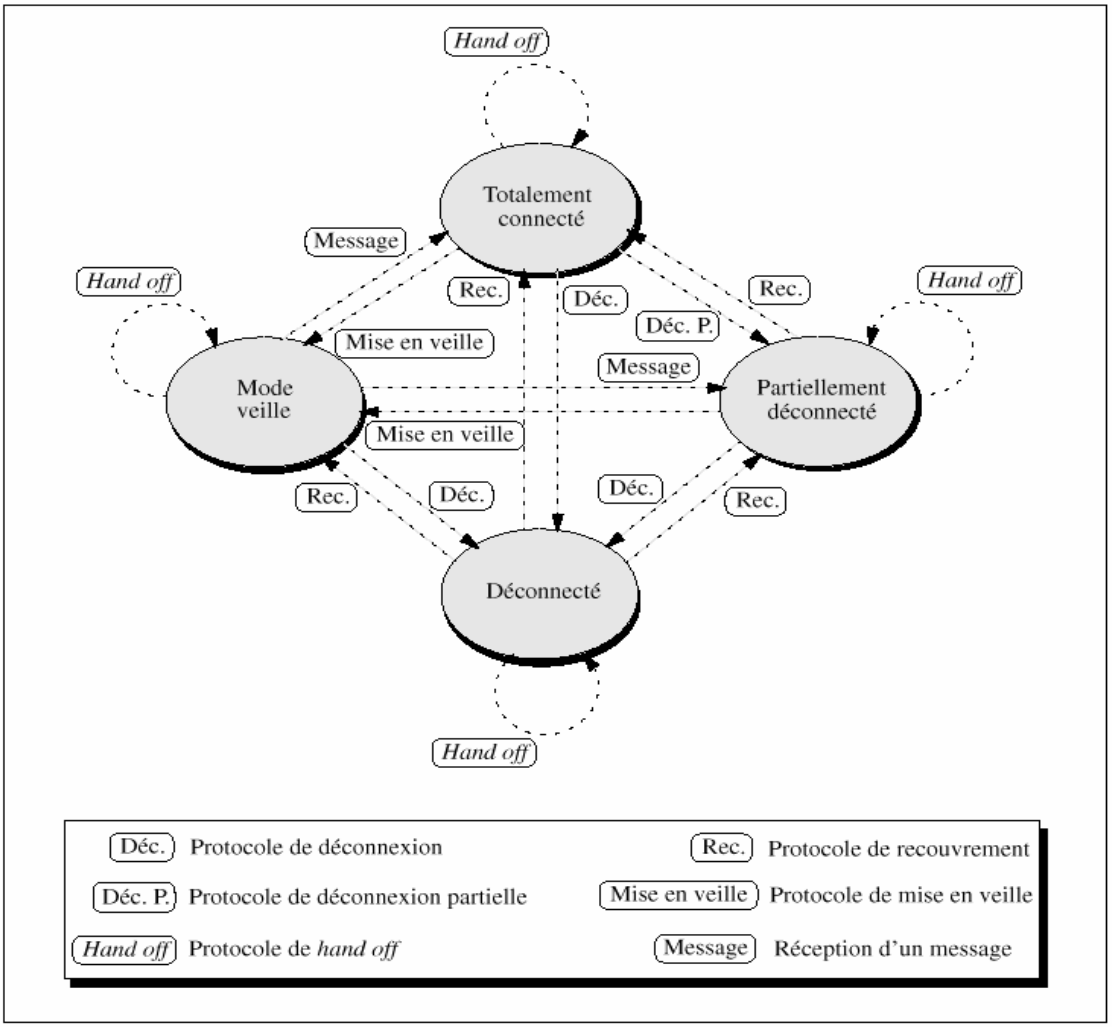


Figure 1.2 – *Modes de fonctionnement d'un site mobile*

1.5 Les caractéristiques des environnements mobiles

L'environnement mobile offre aux utilisateurs la capacité de pouvoir se déplacer tout en restant connecté au réseau et d'être indépendants de toute localisation. Pour permettre aux utilisateurs d'un tel environnement d'avoir un accès continu aux services et aux ressources du réseau, il est nécessaire de disposer d'interfaces de communication sans fil, á la fois sur certaines stations fixes du réseau et sur les mobiles. C'est alors seulement que la mobilité acquiert tout son intérêt, les utilisateurs étant enfin à même de se déplacer tout en conservant active leur connexion au réseau.

Cependant de nouveaux problèmes peuvent apparaître causés par les nouvelles caractéristiques du système mobile. Ceci nécessite des mécanismes spécifiques pour s'adapter aux limitations qui existent, ainsi aux facteurs qui rentrent dans le jeu lors de la conception.

1.5.1 Les connexions sans fil

Les mécanismes permettant de mettre en œuvre les connexions sans fil sont multiples, ils impliquent la prise en compte de l'environnement qui interagit avec le signal, le bloque, ou introduit du bruit, de l'écho [5]. Les communications sans fil sont donc de moins bonne qualité que les communications filaires : les largeurs de bande sont plus réduites et les besoins très variables, les taux d'erreur plus élevés, les déconnexions fréquentes. Ces facteurs peuvent augmenter les temps de latence dus aux retransmissions, aux délais d'attente entre ces retransmissions, aux exécutions de protocole de contrôle d'erreur et aux courtes déconnexions. Dans un environnement sans fil, les connexions peuvent être perdues ou dégradées du fait de la mobilité et du passage de cellule en cellule, ou du fait d'interférences. A la différence des réseaux fixes, le nombre de machines connectées dans une cellule peut être très variable, et des concentrations importantes d'utilisateurs dans une même cellule peuvent surcharger le réseau [10].

Les points qui suivent passent en revue les différents problèmes induits par l'utilisation de communications sans fil : déconnexions fréquentes, faible bande passante, variabilité de la demande de bande passante. Nous verrons aussi les problèmes d'hétérogénéité des réseaux et d'augmentation des risques du point de vue de la sécurité, qui sont également liés aux médiums de communication sans fil.

a. Les déconnexions

Les pannes réseau concernent encore plus l'informatique mobile que l'informatique classique, puisqu'elle est sujette à de fréquentes déconnexions lors des transmissions sans fil [10]. Deux possibilités se présentent alors : soit consommer plus de ressources sur le réseau pour essayer d'éviter ces déconnexions, soit consommer ces ressources pour permettre aux systèmes de mieux gérer ces déconnexions et de continuer à fonctionner malgré les inconvénients qu'elles induisent.

Donc, plus un ordinateur sera indépendant, (puissance, données), mieux il pourra supporter les déconnexions du réseau [10].

Le système Coda fournit un bon exemple pour la gestion des déconnexions, bien qu'il soit prévu pour les *notebooks* actuels où les déconnexions sont beaucoup moins fréquentes, plus prévisibles et plus longues qu'en environnement mobile [11]. Des informations sur le profil utilisateur sont conservées pour optimiser les mécanismes de cache de fichiers en choisissant les fichiers les plus appropriés.

b. La faible largeur de la bande passante des médiums sans fil

L'environnement de calcul mobile est davantage concerné par les problèmes de limitation et de consommation de largeur de bande passante que l'environnement fixe ; les réseaux sans fil ayant des largeurs de bande plus petites que les réseaux filaires. Les débits crête pour les communications sans fil sur portables atteignent seulement un Mega bit par seconde pour l'infrarouge, deux Mega bits par seconde pour la radio, neuf à quatorze Mega bits par seconde pour la téléphonie cellulaire, tandis qu'un réseau Ethernet fournit dix Mega bits par seconde, FDDI cent Mega bits par seconde et ATM cent cinquante cinq Mega bits par seconde, et même un réseau sans fil non portable comme le Motorola Altair fournit 5,7 Mega bits par seconde [10].

En plus de ces limitations déjà existantes, il faut se souvenir que, dans une cellule, la largeur de bande est partagée entre les différents utilisateurs. La largeur de bande

disponible pour chaque utilisateur offre donc un meilleur aperçu de la capacité du réseau que la largeur de bande totale. Cette mesure dépend en effet du nombre et de la distribution des mobiles.

Pour augmenter la capacité du réseau, il est possible d'installer davantage de cellules de communication sans fil sur une zone donnée. Il existe deux techniques: soit l'on fait se recouvrir des cellules fonctionnant sur des longueurs d'onde différentes, soit l'on réduit les zones de couverture de chaque cellule. [5]

c. La variabilité des besoins utilisateur

La mobilité des utilisateurs est également synonyme de plus grandes variations dans les besoins de bande passante qu'en environnement statique. Les besoins en largeur de bande passante peuvent se voir multipliés par quatre dans un réseau sans fil. Les fluctuations de trafic sur réseaux fixes atteignent rarement de telles variations [5].

Une application peut supporter œtte variabilité de trois manières différentes [10]. Elle peut :

- Supposer toujours disposer de connexions á large bande et n'opérer que via des liens de communication physiques ;
- Supposer disposer de connexions á faible bande et ne pas tirer avantage des plus grandes largeurs disponibles ;
- S'adapter dynamiquement aux ressources disponibles fournissant ainsi à l'utilisateur un niveau de qualité variable, mais qui tire parti au maximum des capacités réelles du réseau.

d. L'hétérogénéité des réseaux

Contrairement aux systèmes répartis classiques (fixes) où les machines sont connectées une fois pour toute à un réseau donné, dans un environnement mobile, se rencontre non seulement une multitude de types de sites mobiles mais également un grand nombre de réseaux à la fois avec ou sans fil. Les mobiles se retrouvent donc à naviguer dans un environnement hautement hétérogène et de même les réseaux risquent de voir passer un grand nombre de machines de tout ordre et d'utiliser un nombre important de protocoles d'accès différents.

Les interfaces de communication sans fil risquent également de changer lors de déplacements entre l'intérieur et l'extérieur. Par exemple, les infrarouges relativement sensibles aux rayonnements solaires sont plus facilement utilisés à l'intérieur. Et même si une interface radio est conservée, les protocoles sont susceptibles de changer lorsque nous passons d'une couverture cellulaire à une couverture satellite. La gestion de cette hétérogénéité implique des traitements plus complexes en environnement mobile q'en environnement traditionnelle (fixe) [10].

e. Les risques de la sécurité

Précisément puisqu'il est facile de se connecter à un lien sans fil, la sécurité des communications sans fil peut être compromise bien plus facilement que celle des communications avec fils, tout spécialement lorsque les transmissions se font sur de très grandes distances (interception , génération de messages). Il est donc nécessaire d'inclure des mécanismes sécuritaires aux réseaux sans fil.

La sécurité sur ce type de communications est d'autant plus complexe si les utilisateurs sont autorisés à traverser des domaines de sécurité de différents pays. Plus localement, par exemple, s'ils traversent différents services dans un hôpital, ou les bornes de divers réseaux.

Les problèmes rencontrés ont donc trait à l'usurpation d'identité, au refus de service, à l'écoute ou encore à la surveillance des déplacements des mobiles [12]. A la fois les mobiles et les réseaux fixes qu'ils visitent, doivent donc être protégés contre tous ces problèmes. Cette protection passe à l'heure actuelle par deux principes, l'authentification et le respect de l'anonymat (confidentialité des informations).

1.5.2 La mobilité

Le but de l'informatique mobile est de fournir à l'utilisateur une complète indépendance de mouvement, tout en conservant une transparence à la localisation variable en fonction des besoins. Cette indépendance peut passer par différents concepts de mobilité, qu'il est éventuellement possible de mettre en œuvre conjointement. Trois concepts de mobilité peuvent être distingués : la mobilité du matériel qui est à l'heure actuelle la plus fréquemment citée dans les articles sur les systèmes mobiles [10, 13, 14], la mobilité des utilisateurs seuls [15] et la mobilité des interfaces des applications [16].

La mobilité du matériel et de l'utilisateur avec lui constitue pour le moment le problème le plus étudié en environnement mobile. Les problèmes posés sont d'ordre multiple et couvrent aussi bien le domaine des réseaux (adressage, protocoles de transport) que la gestion des données (accès, stockage, cohérence) ou les mécanismes systèmes (algorithmes de base, accès aux services et aux ressources).

La mobilité est un comportement ayant des implications aussi bien pour les stations qui ne bougent pas et qui constituent le réseau fixe, que pour les machines qui se déplacent avec les utilisateurs et qui constituent le réseau sans fil ou mobile [13].

Dans cette partie, nous allons présenter les principaux problèmes introduits par la mobilité, tels que l'adressage, la localisation des sites mobiles, la gestion des données…etc.

a. Le nommage de sites mobiles et le routage

Les utilisateurs se déplaçant, leur ordinateur mobile utilise différents points d'accès réseau, autrement dit, des "adresses". Actuellement, les protocoles réseau ne permettent pas des changements d'adresses dynamiques et les connexions réseau actives ne peuvent pas être migrées vers de nouvelles adresses.

Les adresses tendent à désigner des machines précises, et non pas seulement une interface de connexion. Une fois une adresse est connue dans un système, pour un certain nom de machine, elle est conservée dans un cache pour un temps relativement long et il n'existe aucun moyen d'invalider les entrées obsolètes. Dans *Internet Protocol* (IP) par exemple, un nom de machine IP est lié à son adresse réseau et à sa localisation. Se déplacer vers une autre position implique l'acquisition d'une nouvelle adresse IP, et donc en général, une intervention humaine pour coordonner l'utilisation des adresses [10].

Lorsqu'un site mobile franchit les limites de son réseau, il doit donc obtenir un nom qui permettra aux messages ou données qui lui sont destinées d'être routées jusqu'au nouveau réseau, ou d'être redirigées à partir de l'ancien domaine. La plupart des

protocoles de routage sont basés sur l'hypothèse que les ordinateurs se déplacent rarement. Dans *Internet Protocol* (IP) une adresse IP d'un site dépend du réseau auquel il est connecté, et les paquets de données sont routés en se basant sur le numéro du réseau. Un tel schéma est inadéquat pour les sites mobiles, puisque les sites mobiles ne sont pas connectés de manière permanente au même réseau.

La recherche dans le domaine des schémas d'adressage et le routage se concentre : (a) sur comment assigner les adresses aux sites mobiles de telle sorte que les protocoles basés sur IP peuvent être utilisés pour le routage des messages, et (b) sur comment optimiser le coût de recherche pour la localisation d'un site mobile. La localisation d'un site mobile se reporte à l'adresse de la station de base à laquelle il est actuellement connecté.

b. La localisation de ressources et de sites mobiles

Les ordinateurs classiques ne se déplaçant pas, les informations dépendantes de la localisation sont définies statiquement, comme le serveur de nom le plus proche, les imprimantes disponibles et le fuseau horaire. Un des challenges de l'informatique mobile est de gérer intelligemment l'information et de fournir des mécanismes capables d'obtenir la configuration appropriée à la localisation courante [10].

A côté de ces problèmes de configuration dynamique, les sites mobiles ont besoin d'accéder à des informations relatives à la localisation, plus pertinentes que n'en disposent actuellement les ordinateurs fixes, et ce principalement s'ils doivent servir de guides dans des lieux peu ou pas connus de leur utilisateur. Ils doivent donc être capables, grâce à ces informations, de répondre à des questions du type "où est la salle de conférence sur l'informatique mobile ?" ou "où est la prochaine station service en allant vers le nord ?". Ces requêtes concernent des informations sur l'environnement dont la localisation est statique. Il est aussi possible d'envisager l'extension de ces mécanismes à des informations dynamiques, se référant à d'autres objets mobiles, par exemple, pour déterminer où se trouve le taxi libre le plus proche [17].

En environnement mobile, les mouvements fréquents des machines impliquent à la fois une gestion appropriée des informations de localisation et des mécanismes de recherche de sites mobiles, de manière à mettre à jour leur position [18]. Il est cependant nécessaire de faire la part des choses entre le maintien des localisations et les besoins d'information des mobiles.

Pour localiser les mobiles, il est possible de partitionner l'environnement en utilisant une structure hiérarchique [19, 20] organisée en stations support (ou station de base) et en différents niveaux de serveurs de localisation. Dans un tel système, les mobiles sont chargés d'envoyer des notifications lors de sorties de cellules aux serveurs de localisation, qui sont eux-mêmes chargés de gérer toutes les mises à jour, d'où des surcharges en messages et une nécessité de partitionnement de l'information [21, 22].

Le but d'une telle structure est de gérer les mises à jour simultanément pour éviter le goulot d'étranglement d'un système de mise à jour centralisé [12]. Elle correspond aux méthodes utilisées dans les architectures cellulaires actuelles [23] ou celle proposée pour Internet dans [24].

c. La gestion des données

Du point de vue de la gestion de données, l'informatique mobile engendre des problèmes de récupération de données, dépendance à la localisation, diffusion de

données, gestion des déconnexions et accès efficaces aux données (en prenant en considération les problèmes d'énergie). [5]

Globalement, les objectifs de recherche se classent, en fonction du fait qu'ils sont dépendants de la mobilité, des déconnexions, des modes d'accès aux données ou du dimensionnement du réseau mobile [17].

Lorsque les utilisateurs posent des requêtes aux différentes bases de données auxquelles ils ont accès, il est nécessaire de connaître leur position, non seulement pour leur envoyer la réponse, mais aussi pour assurer des réponses correctes aux requêtes dépendantes de la localisation comme des questions telles que ``Quelle est la plus proche station de métro ?'' ou ``Quelle est la route la plus courte pour aller à l'hôpital le plus proche ?''. Si la localisation contenue dans la base est incomplète, il va être nécessaire de procéder à la fois à une recherche dans la base de données et à une acquisition de données pendant le temps d'exécution de la requête. Un nouveau modèle de réponse aux requêtes devra être développé pour inclure les méthodes d'acquisition d'information [20, 19].

D'autre part, la mobilité des utilisateurs implique souvent de répliquer les données couramment utilisées. La réplication est en effet le moyen le plus simple pour assurer la transparence aux utilisateurs mobiles. Un utilisateur qui s'est déplacé et qui utilisait auparavant certains fichiers, applications ou services doit pouvoir continuer à y accéder à sa nouvelle position.

D'une manière générale, les données doivent suivre d'assez près les mouvements des lecteurs les plus actifs, alors qu'elles s'éloigneront des moins actifs dont les mouvements font peu de différences dans l'usage des données [25]. Elles pourront même être placées sur les mobiles par des méthodes de cache ou de prefetching dont le contenu sera dépendant de la localisation [26, 27].

Les sites mobiles étant souvent déconnectés, volontairement ou non, et à des degrés différents, il est nécessaire de gérer sur les mobiles des mécanismes de cache et de prefetching, ainsi que la cohérence des données contenues dans ces caches et dans le système (ensemble des serveurs, et éventuellement mobiles, où sont répliquées les données) [28]. Il est généralement nécessaire d'adapter les mécanismes de requêtes ou de transactions pour prendre en compte ces éventuelles déconnexions des mobiles et ne pas laisser le système dans un état incohérent [17, 9, 29].

Les degrés de déconnexion étant variables et fonction de la largeur de bande passante disponible, il est donc irréaliste de vouloir maintenir parfaitement les données répliquées sur le mobile [9].

Pour ce qui concerne les requêtes et transactions, il va être nécessaire de les adapter de manière à ne pas laisser les bases de données ou le système dans un état inconsistant. Pour le cas des transactions, il faut prévoir des mécanismes permettant aux mobiles de terminer les transactions en cours [17]. Les déconnexions étant d'une manière générale prévisibles, il est possible de décharger sur le site mobile les informations nécessaires pour qu'il puisse opérer de façon autonome. Toutefois, dans le cas général, la mémoire du mobile ne peut être considérée comme étant fiable, il est donc à éviter d'y stocker les journaux. Il sera préférable de les conserver sur le réseau fixe [9, 17]. La localisation du mobile, si elle est connue doit elle aussi être sauvée dans le journal, ou au pire, il faut signaler que le mobile s'est déplacé [17].

Du point de vue de l'exécution des demandes des utilisateurs, il est nécessaire de gérer à la fois de nouveaux types de requêtes, et d'introduire de nouveaux paramètres

d'optimisation. Les résultats de ces requêtes peuvent être dépendants de la localisation du mobile et éventuellement, même, de la direction qu'il suit. Cette localisation peut de plus évoluer pendant l'évaluation de la requête, ou être imprécise du fait des coûts de maintien de ce type d'information.

D'une manière générale, les nouveaux paramètres à prendre en compte dans l'optimisation de requêtes incluent la minimisation du coût des transactions et du coût de recherche des informations de localisation, la préservation de la largeur de bande et la limitation de la consommation d'énergie [9, 30].

Nous avons donc donné un aperçu des problèmes liés aux caractéristiques des environnements mobiles, à savoir les problèmes liés à la communication sans fil et à la mobilité. Nous ne donnerons pas davantage de détails sur ce sujet, nous allons plutôt nous attacher dés à présent aux problèmes des algorithmes système.

1.6 Algorithmes répartis en environnement mobile

L'intégration des ordinateurs mobiles avec les réseaux statiques existants introduit un nouvel ensemble de problèmes dans les systèmes répartis. Un site mobile peut se connecter au réseau à partir de différentes localisations à différents moments via un médium de communication sans fil, les mécanismes répartis (algorithmes, systèmes de gestion de fichiers, protocoles de diffusion de messages, etc.) ne peuvent donc plus supposer que les participants restent à une position fixe et universellement connue dans le réseau au cours du temps. Il devient donc nécessaire de revoir la structuration de ces mécanismes pour les adapter à l'informatique mobile.

L'étude des problèmes de synchronisation et de communication dans les systèmes répartis a été abordée pour des architectures comprenant uniquement des sites statiques. En absence de pannes de site ou de liaisons de communication, la structure topologique du réseau d'interconnexion ne varie pas. A l'inverse, dans les systèmes intégrant des sites mobiles, capables de se déplacer tout en conservant leur connexion au réseau active via des techniques de transmission sans fil, la connectique est en constante évolution. De nouveaux problèmes sont donc induits par la mobilité [31] :

1. Les mobiles se déplaçant, la connectique du réseau change. Par conséquent la structure logique, utilisée dans de nombreux algorithmes répartis, ne peut plus être définie statiquement ;
2. Pour délivrer un message à un site mobile, il est nécessaire que la destination soit d'abord localisée dans le réseau ;
3. La communication entre un site mobile et le reste du réseau se fait via un lien sans fil, possédant une faible largeur de bande par rapport aux liens filaires. De plus, la transmission et la réception des messages sur le lien sans fil entraînent la consommation d'énergie du site mobile ;
4. Les sites mobiles ont d'importantes contraintes en terme de puissance et d'énergie et opèrent souvent en mode veille ou complètement déconnecté du réseau. La déconnexion dans un environnement mobile est distinguée d'une panne : les déconnexions sont volontaires et donc, un site mobile peut informer le système d'une déconnexion imminente antérieurement à son occurrence.

Les algorithmes classiques (exclusion mutuelle, élection, terminaison, etc.), les protocoles de communication de groupe et de diffusion de messages ainsi que les systèmes de gestion de fichiers devront tenir compte de ces caractéristiques pour intégrer explicitement la mobilité, fournir des plates-formes à même de gérer efficacement les mobiles, et adaptées aux besoins des utilisateurs.

1.6.1 Les structures logiques

La structure topologique du réseau d'interconnexion dans les systèmes répartis est généralement modélisée par un graphe non orienté où les sommets symbolisent les unités de calcul (sites) et les arêtes les liaisons physiques entre les sites [32]. C'est ainsi que les algorithmes répartis, développés jusque-là, sont souvent basés sur une topologie de type arbre, graphe complet ou anneau, où chaque lien logique correspond à un chemin dans le réseau physique sous-jacent [1]. Un site peut envoyer un message à n'importe quel autre tant qu'il existe une suite d'arcs permettant d'atteindre le destinataire.

La mobilité d'une machine implique que sa localisation par rapport au reste du réseau évolue au cours du temps, la connectique du réseau tout entier est modifiée, et donc la structure logique qui en découle [31]. En conséquence, un lien logique entre deux mobiles ne peut plus correspondre à une séquence de liens physiques du réseau.

La plupart des algorithmes distribués, exp. [33, 42], se basent sur une structure logique crée entre les participants pour assurer la bonne transmission des messages. Le but principal d'une telle structure est de fournir un certain degré d'ordre et de prédictibilité dans les communications entre les participants [34, 35]. Les messages échangés en utilisant de telles structures suivent uniquement les chemins logiques prédéfinis.

Pour un site mobile *MH* qui va pouvoir changer sa localisation dans le réseau, l'implémentation d'un lien logique nécessite d'être mise à jour à chaque déplacement des sites mobiles. Pour illustrer ce fait, considérons le cas d'un anneau logique entre trois sites mobiles MH_1, MH_2, MH_3 (Figure 1.3 (a)). Dans cet exemple, un seul lien logique dans l'anneau, exp. entre MH_1 et MH_2, se compose de: (1) la connexion sans fil entre MH_1 et sa *MSS* locale courante, MSS_1, (2) une connexion point-à-point entre MSS_1 et MSS_2, la *MSS* locale de MH_2 et (3) une connexion sans fil entre MH_2 et MSS_2. Si un mobile change de position, sa station support change également, l'anneau logique doit donc être reconfiguré (Figure 1.3 (b)), de même si le mobile se déconnecte ou même passe en mode veille (Figure 1.3 (c)).

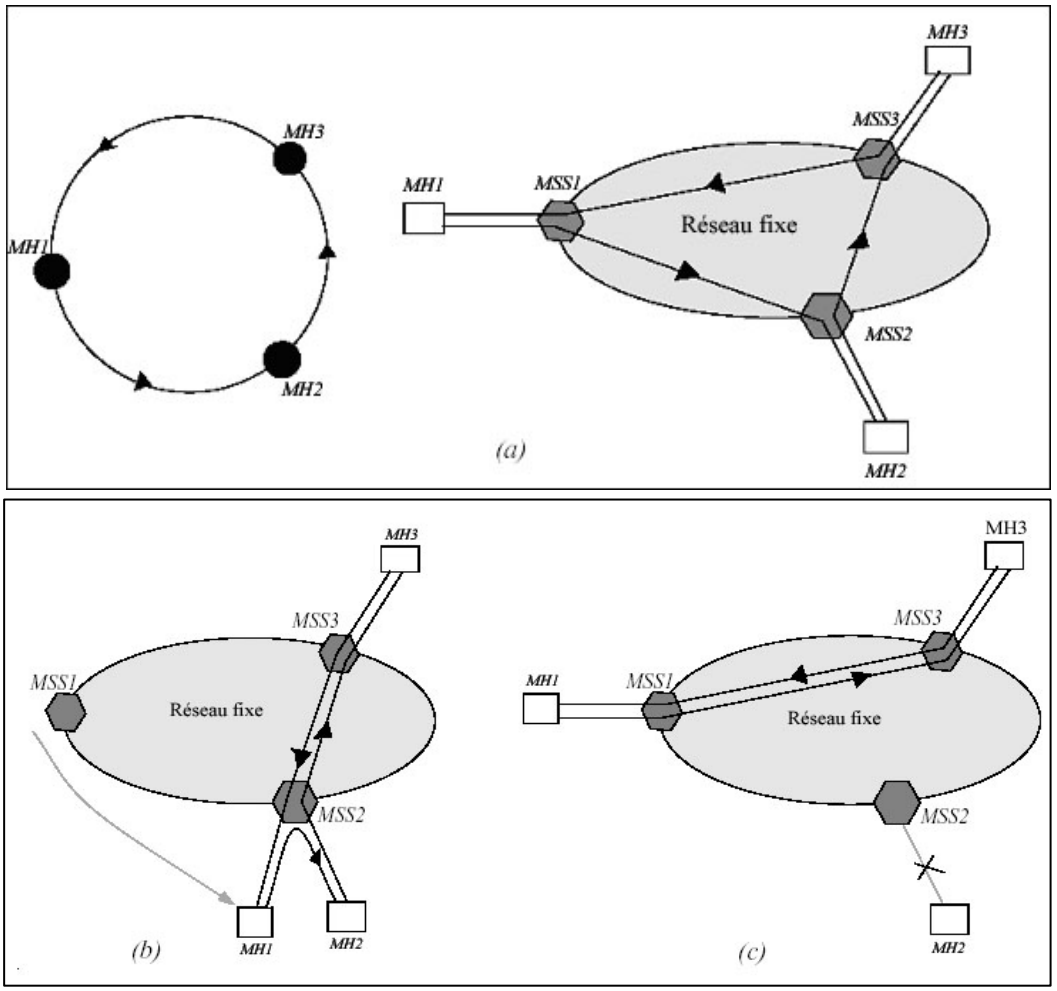


Figure 1.3 – *Effets de la mobilité sur un anneau logique unidirectionnel*

L'impact de la mobilité sur les structures logiques se traduit par des reconfigurations à chaque déplacement d'une machine. Globalement, pour autoriser la conservation d'une structure logique dans un algorithme, les bénéfices apportés par cette structure doivent justifier les coûts de maintien induits par les déplacements des sites mobiles.

En implémentant la structure logique uniquement entre les stations support [31] (Figure 1.4), il est possible d'obtenir les bénéfices escomptés par une structure logique, sans pour autant être confrontés aux inconvénients dus aux sites mobiles et aux besoins de reconfigurations fréquentes de la structure.

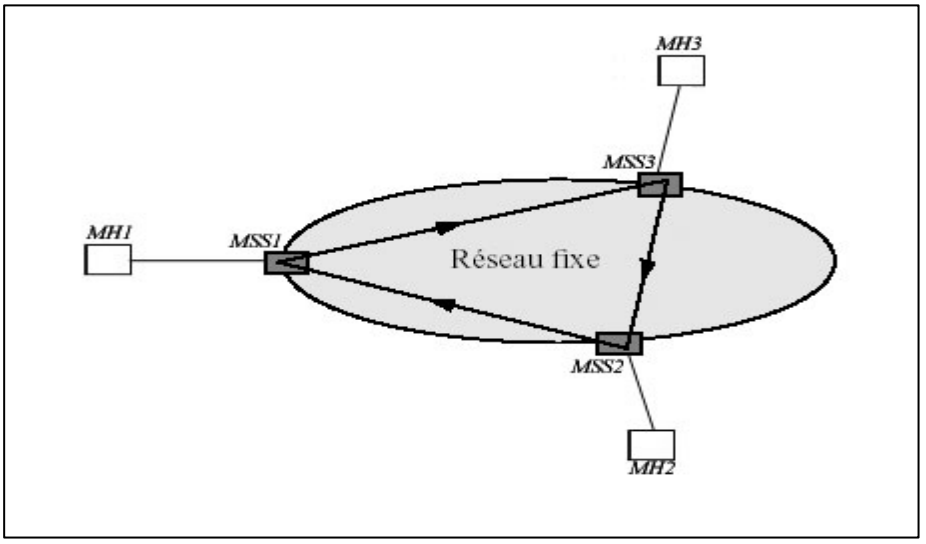


Figure 1.4 – *Anneau logique sur stations support*

1.6.2 Localisation des sites mobiles

Pour envoyer un message à partir d'un site mobile MH_1 vers un autre site mobile MH_2, il est tout d'abord nécessaire que MH_1 transmette le message à sa station support locale à travers le réseau sans fil. Si les deux mobiles se trouvent dans la même cellule, la station retransmet directement le message au site destinataire, MH_2, [34, 35]. À l'inverse, si les deux sites mobiles ne se trouvent pas dans la même cellule, il va être nécessaire de localiser le mobile destinataire pour pouvoir faire suivre le message à la station support appropriée [34, 35]. La station distante ainsi trouvée sera alors chargée de la retransmission au mobile destinataire.

Le coût de communication induit dépend du coût lié à la distance à parcourir sur le réseau fixe pour atteindre le destinataire, du coût de communication sans fil et du coût de recherche des mobiles impliqués [34, 35].

1.6.3 Connexion sans fil

Un inconvénient des communications sans fil est que la largeur de bande passante est relativement réduite par rapport à celle disponible pour les liaisons filaires. Par contre, les communications sans fil supportent physiquement la diffusion à l'intérieur d'une cellule, une station de base peut émettre un message à toutes les unités mobiles localisées dans sa cellule, en une seule opération de transfert. L'exploitation de ce mode de transmission permettra de diminuer les coûts des envois de données.

Il est cependant important de tenir compte des problèmes classiques liés à la diffusion de messages dans des groupes de machines ou de processus. En l'occurrence, il faut assurer, la réception des messages par tous les membres du groupe, et ceux-ci exclusivement. Puis, selon les besoins, il est nécessaire de respecter un ordre au sein du groupe de communication (par exemple, réception selon un ordre quelconque identique pour tous les membres du groupe, selon l'ordre d'émission, l'ordre causal, etc.). Ces problèmes sont bien connus en environnement distribué, les résultats déjà obtenus peuvent donc être exploités avec bénéfice dans le cas de l'environnement mobile.

Les délais de transfert des messages entre les sites fixes étant arbitraires, un message émis par une même unité mobile peut être reçu à des instants différents par des stations de base différentes. Il est donc possible qu'un message diffusé dans une cellule puisse ne

pas être reçu par une unité mobile destinataire, car son entrée dans la cellule a eu lieu après la diffusion du message ou bien sa sortie de celle-ci a eu lieu avant la diffusion [31]. En revanche, une unité peut recevoir plusieurs copies du même message, issues de stations de base différentes. Dans la figure 1.5, MH_1 se déplace de MSS_3 vers MSS_1, en perdant le message dans les deux cellules, cependant MH_2 reçoit deux copies du message dans MSS_4 et MSS_3. On trouve dans [36] une approche de mise en œuvre de la communication multi-destinataire dans un environnement mobile qui garantit la réception de exactement une copie du même message par tous les destinataires.

Notons par ailleurs que les transmissions de messages à partir de sites mobiles consomment davantage de puissance que les réceptions pour un message de taille identique [36]. La communication au sein d'une cellule doit de préférence être asymétrique de manière à exploiter au mieux la capacité de diffusion des communications sans fil et de réduire la consommation d'énergie sur les mobiles.

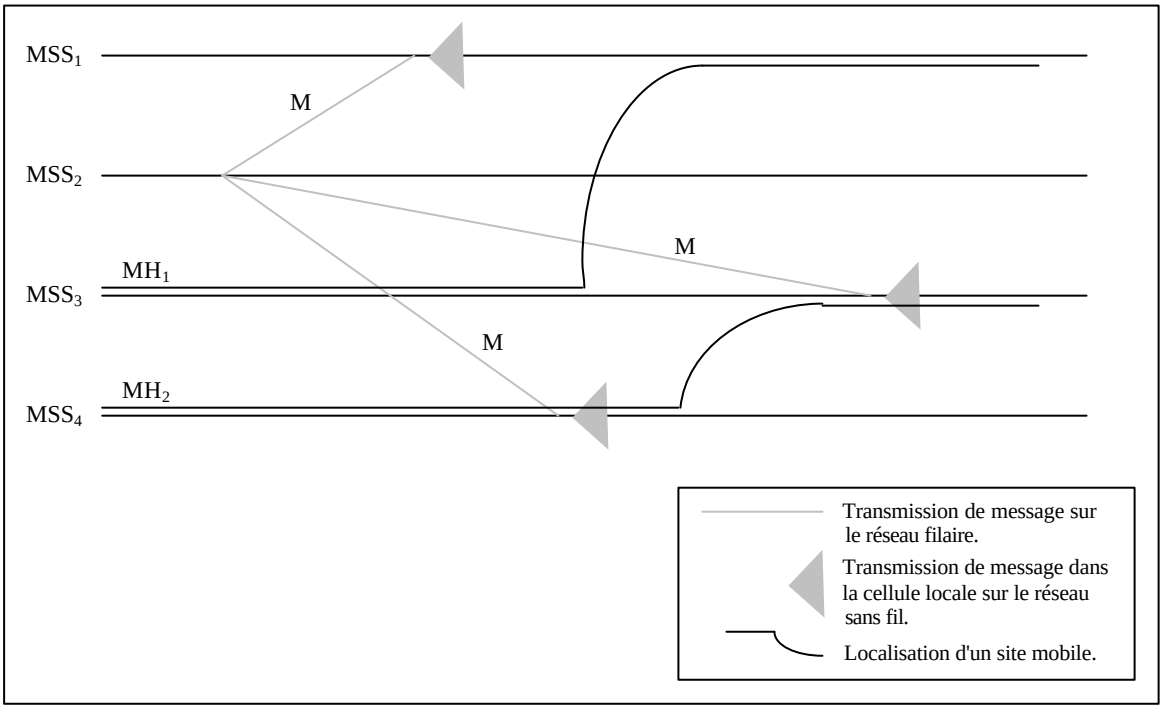


Figure 1.5 – *la communication sans fil et la mobilité*

1.6.4 Déconnexion et mode veille

Les sites mobiles sont souvent déconnectés du reste du réseau. Un site mobile déconnecté ne peut ni envoyer ni recevoir de messages et par conséquent, un site mobile qui se déconnecte au milieu d'une exécution d'un algorithme risque de bloquer toute l'exécution jusqu'à sa reconnexion. La déconnexion dans l'environnement mobile est distinguée d'une panne: les déconnexions sont volontaires par nature et donc, un site mobile doit pouvoir anticiper une possible déconnexion en exécutant un protocole de déconnexion qui informe le système de l'imminence de sa sortie du réseau avant qu'il soit physiquement détaché du réseau. Ceci a deux implications. Premièrement, le participant détaché peut s’assurer de son fonctionnement normal, indépendamment du

reste des participants, en téléchargeant à son niveau les données dont il a besoin, et pouvoir réintégrer ultérieurement. Deuxièmement, le participant détaché peut aussi transmettre au réseau fixe toute information d'état ou des données que pourrait nécessiter la poursuite de l'exécution du programme en son absence. [34, 35]

Un nouveau mode à prendre en compte correspond au mode veille dans lequel la vitesse de l'horloge est réduite et aucune exécution utilisateur n'est réalisée. Le site mobile attend alors simplement la réception d'un message pour reprendre son mode de fonctionnement normal. Comme dans le cas des déconnexions, ce mode de fonctionnement est une opération volontaire, cependant les implications sont différentes. Dans le mode veille, un site mobile peut encore être atteint par le reste du système et donc peut être amener par le système à reprendre son mode de fonctionnement normal, alors que les déconnexions et reconnexions ne peuvent être initialisées que par le mobile lui-même.

Le mode veille est mal supporté par les algorithmes totalement décentralisés qui exigent la participation de chaque site mobile; de tels algorithmes empêchent ces sites mobiles d'opérer en mode veille même s'ils ne sont pas impliqués dans le calcul. Par conséquent les tentatives de conservation d'énergie dans ces sites mobiles en opérant en mode veille sont complètement contrariées.

Un mécanisme réparti qui comprend des sites mobiles ne doit pas les contraindre à participer à chaque exécution, mais les autoriser à passer en mode veille ou en mode déconnecté. Si l'algorithme est exécuté directement sur des mobiles, il doit donc tenir compte des éventuelles déconnexions et devra fournir aux autres participants les informations nécessaires à la poursuite correcte de l'exécution. Ceci passe par l'adaptation des différents algorithmes et applications réparties, c'est à dire par ajout de protocoles de déconnexion, de chargement et déchargement de données, messages d'information, etc. [5]

Dans [34, 35] Badrinath et al. Analysent l'impact de la mobilité sur un ensemble d'algorithmes répartis classiques d'exclusion mutuelle [37, 33]. L'idée qui sous-tend cette analyse est d'arriver à un découplage entre la mobilité des sites et la construction des algorithmes pour des environnements mobiles. Pour cela, les algorithmes répartis sont structurés de manière à ce que la majeure partie des communications et du calcul induit par l'algorithme soit prise en charge par le réseau statique. La même démarche est suivie par [38] pour adapter l'algorithme de [39] pour la mise en œuvre de l'ordre causal dans un environnement mobile.

Pour illustrer cette approche, nous reprenons ci-dessous les deux variantes de l'algorithme réparti de la circulation d'un jeton dans un anneau logique telles que présentées dans [2]. La première variante est une mise en œuvre de l'algorithme sans tenir compte des particularités de l'environnement mobile et la deuxième variante est une restructuration de l'algorithme pour l'adapter à ce type d'environnement.

1.6.5 Etude de cas : Algorithme de circulation de jeton sur un anneau logique [2]

Un algorithme fondamental dans les systèmes distribués consiste dans la circulation d'un jeton entre les participants d'un anneau logique. Chaque participant se comporte comme suit :

- Attendre la réception du jeton à partir de son prédécesseur dans l'anneau ;
- Entrer dans <la section critique>, si désirer ;
- Envoyer le jeton à son successeur dans l'anneau.

L'algorithme satisfait deux propriétés importantes :

1. L'exclusion mutuelle est trivialement garantie pour le propriétaire courant du jeton, et
2. Il permet un accès équitable au jeton en permettant à chaque participant d'accéder au jeton au plus une fois pour une circulation donnée du jeton.

Cet algorithme a été utilisé à diverses fins telles que la détection de la terminaison [40], l'exclusion mutuelle [33], les protocoles d'ordonnancement de messages [41] dans les systèmes fixes. Dans un environnement mobile, le jeton peut être utilisé comme un mécanisme pour l'accès distant à une ressource partagée.

a. Algorithme R-MH

L'algorithme R-MH est une implémentation directe de l'anneau logique entre les sites mobiles, et représente le cas extrême pour l'exécution d'un algorithme existant sans se soucier de la mobilité des sites et des contraintes ressources associées. Malgré que la correction de l'algorithme soit assurée par cette approche, les contraintes ressources spécifiques à l'environnement mobile ne sont pas prises en considération :

- *coût de recherche élevé*

  Tous les messages de l'algorithme doivent être émis à chaque site mobile, ce qui fait entraîner un coût de recherche. Puisque l'algorithme fait circuler le jeton entre tous les MH_s, le coût de recherche total entraîné par l'algorithme est proportionnel au nombre des MH_s.

- *usage excessif des liens sans fil*

  L'émetteur ainsi que le récepteur de chaque message est un *MH* ; le message est donc transmis à travers les liens sans fil entre les deux MH_s émetteur et récepteur, et leurs MSSs locales respectives.

- *Consommation d'énergie dans les MH_s*

  Chaque message dans cet algorithme consomme de l'énergie dans l'émetteur ; pour le transmettre à sa *MSS* locale, ainsi que dans le récepteur ; pour recevoir le message à partir de sa *MSS* locale.

- *Les modes veille et déconnecté*

  L'algorithme R-MH exige la participation de chaque MH pour maintenir la structure de l'anneau logique et par conséquent ne permet à aucun MH de se déconnecter. D'autre part, l'algorithme R-MH ne permet pas à un site mobile de se mettre à l'état veille sans être interrompu même s'il n'a pas besoin d'accéder à la ressource partagée (représentée par le jeton) ; il doit assurer la réception et l'expédition du jeton pour permettre aux autres MH_s d'accéder à la ressource.

Il est important de souligner ici que les inconvénients cités ci-dessus ne sont pas intrinsèques à l'algorithme, mais découlent plutôt d'une application inadéquate de l'algorithme, i.e. l'anneau logique est établit entre les sites mobiles qui sont caractérisés,

contrairement aux sites fixes, par une connectivité physique qui nécessite d'être redéfinie suite à chaque déplacement d'un site mobile. De plus, les sites fixes ne souffrent pas des contraintes de consommation d'énergie et des connexions sans fil à bande passante limitée, par conséquent une solution à ces inconvénients est d'appliquer l'algorithme à l'environnement mobile selon le principe deux-tiers.

b. Restructuration de l'anneau logique selon le principe deux tiers

Le principe deux-tiers suppose que l'anneau logique soit établi sur le réseau fixe entre toutes les MSS_s avec le jeton visitant chaque *MSS* dans une séquence prédéfinie. Un site mobile qui désire accéder au jeton a besoin de soumettre une requête à sa *MSS* locale. Quand le jeton visite cette *MSS*, toutes les requêtes en attente sont servies. Cependant, un *MH* peut changer sa localisation après l'émission de sa requête, et donc la localisation d'un tel *MH* exige d'être explicitement gérée. Deux stratégies de gestion de la localisation ont été proposées, la stratégie *Chercher* et la stratégie *Informer*.

Stratégie *Chercher*

Les actions exécutées par une MSS M

- A la réception d'une requête pour l'obtention du jeton à partir d'un *MH* local, *M* insère la requête à la queue de sa file *request*.
- Quand *M* reçoit le jeton à partir de son prédécesseur dans l'anneau logique, elle exécute les étapes suivantes:

1. fait un transfert des requêtes de la file *request* vers la file *grant*.
2. *Répéter*
   - Défiler la requête de la tête de file *grant*;
   - Si le *MH* qui a établi la requête est local à *M*, alors délivrer le jeton à *MH* via le lien sans fil;
   - Sinon, chercher et délivrer le jeton à *MH* dans sa cellule courante;
   - Attendre le retour du jeton à partir de *MH.*

   Jusqu'à ce que grant soit vide
3. rediriger le jeton vers le successeur de *M* dans l'anneau logique

Les actions exécutées par un MH h

- Quand *h* nécessite l'accès au jeton, il émet une requête à sa *MSS* courante, dite *M*, et
- La MSS où *h* émet sa requête envoiera éventuellement le jeton à *h*. Après que *h* accède à la section critique, il retourne le jeton à la même *MSS*.

Cet algorithme assume qu'un *MH* n'émet une deuxième requête que si sa dernière requête est servie. De plus, quand un *MH* reçoit le jeton, il doit le remettre à la *MSS* émettrice après avoir accéder à la section critique, i.e. il ne peut pas se déconnecter de manière permanente après la réception du jeton.

Stratégie *Informer*

Une alternative de la stratégie *Chercher* pour localiser un *MH* migrant est d'exiger au site mobile de signaler à la *MSS* (d'où il émet sa requête) chaque changement dans sa localisation jusqu'à la réception du jeton.

Les actions exécutées par une MSS M

- A la réception d'une requête pour l'obtention du jeton à partir d'un *MH* local *h*, *M* insère la requête <*h*, *M*> à la queue de sa file *request*.
- A la réception d'un message *inform*(*h*, *M'*), la valeur courante de *locn*(*h*) est remplacée par *M'* dans l'entrée <*h*, *locn*(*h*)> dans la file *request* de *M*.
- Quand *M* reçoit le jeton à partir de son prédécesseur dans l'anneau logique, elle exécute les étapes suivantes:

1. fait un transfert des requêtes de la file *request* vers la file *grant*.
2. *Répéter*
   - Défiler la requête <*h*, *locn*(*h*)> de la tête de file *grant*;
   - Si *locn*(*h*) == *M*, alors délivrer le jeton à *MH* via le lien sans fil;
   - Sinon, rediriger le jeton à *locn*(*h*), i.e. à la *MSS* courante de *h* qui transmettra le jeton à *h*;

   Jusqu'à ce que grant soit vide
3. Rediriger le jeton vers le successeur de *M* dans l'anneau logique

Les actions exécutées par un MH h

- Quand *h* nécessite l'accès au jeton
  - il émet une requête à sa *MSS* courante, dite *M*, et
  - sauvegarde l'identité de *M* dans la variable local *req_locn*.
- Quand *h* reçoit le jeton à partir de la *MSS req_locn*, il accède à la section critique, retourne le jeton à la même *MSS* et remet alors *req_locn* à ⊥.
- Après chaque déplacement, *h* émet le message *join*(*h*, *req_locn*) à sa nouvelle *MSS M'*.
  - Si *req_locn* reçu avec le message *join*() est différent de ⊥, alors *M'* envoie un message *inform*(*h*, *M'*) à la *MSS req_locn*.

Si un site mobile *h* change de cellules moins souvent après l'émission de sa requête, alors il est préférable pour *h* d'informer *S* de chaque changement dans sa localisation au lieu que *S* recherche *h*.

1.7 Conclusion

Ce chapitre a été principalement axé sur la définition de l'environnement mobile, ses caractéristiques et ses problèmes.

Tout d'abord, nous avons pu constater que, du fait des communications sans fil, les sites mobiles disposent de davantage de modes de fonctionnement par rapport aux stations fixes. Donc, en plus des modes connecté et déconnecté, un site mobile pourra également économiser de l'énergie en passant en mode veille, ou encore subir une chute de la capacité de la bande passante et ne plus opérer qu'en étant partiellement connecté.

Nous avons ensuite passé en revue les problèmes causés par les nouvelles caractéristiques de l'environnement mobile; principalement les problèmes liés à la connexion sans fil et à la mobilité.

Enfin, nous avons présenté l'impact de ces caractéristiques sur les mécanismes systèmes existants : les implications sur les algorithmes répartis qui nécessitent une restructuration pour tenir compte des caractéristiques de l'environnement mobile.

Ce chapitre donne donc un aperçu des problèmes des environnements mobiles et des communications sans fil, problèmes qui se répercutent bien évidemment sur tout développement au sein de ces environnements, aussi bien sur les applications que sur les différents mécanismes système. Nous allons s'intéresser dans le prochain chapitre aux mécanismes d'estampillage.

Chapitre 2

Temps logique et mécanismes d'estampillage

2.1 Introduction

Un calcul distribué se compose d'un ensemble de processus qui coopèrent entre eux pour accomplir un but commun. Une caractéristique principale de ces calculs est que les processus ne partagent pas une mémoire globale commune et qu'ils communiquent seulement par échange de messages via un réseau de communication. De plus les délais de transfert des messages sont finis cependant non bornés. Ce modèle de calcul définit ce qui est appelé *système distribué asynchrone*. L'exécution asynchrone des processus et le caractère fini et non borné des délais de communication confèrent aux systèmes répartis un comportement *non déterministe.*

Pour une compréhension appropriée d'un programme distribué et de son exécution, il est important de déterminer la relation causale et temporelle entre les événements qui se produisent durant le calcul. Par exemple, il est souvent le cas que deux événements concurrents ou causalement indépendants peuvent se produire dans n'importe quel ordre, donnant probablement des résultats différents dans chaque cas. Ceci indique que le non déterminisme est strictement lié à la concurrence. En fait, les effets de concurrence et du non déterminisme jouent un rôle important dans le processus d'analyse, de monitoring, de debugging, et de visualisation du comportement d'un système distribué. Mais la caractéristique d'absence d'un référentiel de temps global dans un système distribué rend plus difficile l'identification des activités concurrentes dans les calculs distribués et par conséquent plus complexe la conception, la vérification et l'analyse des programmes répartis.

Le défi consiste dans la définition d'une notion abstraite du temps adaptée pour les systèmes distribués qui, d'un côté, facilement réalisable sans utilisation d'horloges physiques mais, d'un autre côté, a suffisamment de propriétés intéressantes qui justifient le nom « temps » [47]. Idéalement, le concept de temps logique doit fonctionner comme une substitution du temps réel et doit être pratiquement utile à cet égard. Par exemple, une personne préfère être capable d'assigner des valeurs temps aux événements de sorte qu'il soit possible d'inférer la causalité entre ces événements ou d'exclure l'influence causale dans le sens qu'un évènement « récent » ne peut pas affecter un évènement « antérieur ». Des mécanismes d'estampillage de différents types ont été définis dont le rôle est d'assigner des dates logiques (des estampilles) aux évènements, afin de rendre compte de la relation de dépendance entre les évènements. Parmi ces mécanismes, on trouve : les horloges scalaires de Lamport dont le temps est représenté par des entiers

positifs ; les horloges vectorielles dont le temps est représenté par un vecteur d'entiers positifs ; les horloges matricielles dont le temps est représenté par une matrice d'entiers positifs.

Récemment, les systèmes mobiles sont émergés comme un cas distinctif des systèmes distribués augmentant ainsi la demande pour l'adaptation de la théorie et des techniques des systèmes distribués. En comparaison aux nœuds stationnaires servis par un réseau local filaire, les nœuds mobiles se caractérisent par différents modèles de communication et de disponibilité, souvent soumis à de longues périodes de déconnexion. La mobilité en elle-même peut conduire à des changements dans la cardinalité et l'identité des activités incluses dans les calculs distribués. Pris ensemble, ces facteurs influenceront la détermination de la causalité dans les environnements mobiles, augmentant ainsi le besoin pour une adaptation des techniques existantes.

Les modèles existants pour l'estampillage causal sont basés sur un ensemble d'entités préétabli (i.e. les processus où les évènements peuvent se produire). L'arrivée des systèmes de calcul mobiles, qui favorisent la coopération entre des groupes arbitraires de nœuds, empêche l'utilisation d'un ensemble préétabli d'entités. Dans ce chapitre, nous allons faire un tour d'horizon sur les modèles d'estampillage proposés, à ce jour, pour capter les relations de causalité dans ces environnements.

Le chapitre est organisé comme suit : nous présentons d'abord un modèle d'un système distribué, ensuite nous passons en revue les mécanismes d'estampillage déjà existants pour les systèmes distribués : les horloges linéaires, vectorielles et matricielles. Nous parlons par la suite de leur inadéquation pour les systèmes mobiles et par conséquent, nous présentons les alternatives existantes dans ce cadre. Et nous terminons par une conclusion générale de ce chapitre.

2.2 Modèle d'un système distribué

Un système réparti est un ensemble fini de n processus séquentiels $\{P_1, P_2, \ldots, P_n\}$ communiquant et se synchronisant exclusivement par échange de messages. Le comportement de chaque processus est déterminé par un algorithme local. Il définit la réaction du processus lors de la réception de messages émis depuis d'autres processus. Cela consiste en la modification de son état local et éventuellement en l'émission d'un ou plusieurs messages vers d'autres processus. Nous appelons exécution répartie l'exécution simultanée de tous ces algorithmes locaux [46]. La communication entre les différents processus du calcul réparti est du type point à point et le délai de transfert d'un message entre tout couple de processus est considéré comme arbitraire mais fini. Chaque processus a une mémoire locale. Ni mémoire partagée ni horloge globale sont disponibles. Les messages sont échangés à travers des canaux fiables non nécessairement FIFO. Les actions effectuées par un processus produisent des évènements de trois types : les évènements d'*émission* et de *réception* de messages (représentant la communication entre les processus) et les évènements *internes* (causant un changement des variables locales).

2.3 Mécanismes d'estampillage dans les environnements distribués

Pour une compréhension exacte d'un programme distribué et de son exécution, il est important d'être capable de déterminer les relations temporelles (i.e. l'ordre des évènements) entre les évènements produits par les différents processus. La définition de l'ordre des évènements sur un système distribué est plus problématique du fait de la difficulté de maintenir une notion de temps absolu cohérente due à l'absence d'une horloge système globale ou d'horloges locales parfaitement synchronisées.

Cependant une relation *cause et effet*, dite aussi *dépendance causale,* peut être établie entre les événements. En d'autres termes, deux événements sont contraints de se produire dans un certain ordre seulement si l'occurrence du premier peut affecter le résultat du second. Pour exprimer ces dépendances, Lamport [43] a proposé de définir pour ce type de systèmes une notion de temps logique permettant de comparer logiquement des événements : sur un site les événements locaux peuvent être ordonnés en se basant sur l'ordre de leur exécution et l'émission d'un message sur le site émetteur précède toujours sa réception sur le site récepteur. Cela a conduit à la définition de la relation *happened before* [43], dénotée par $\rightarrow$, comme suit :

1. Si *e* et *e'* sont deux événements dans le même processus, et *e* se produit avant *e'*, alors $e \rightarrow e'$.
2. Si e est l'émission d'un message par un processus et e' est la réception du même message par un autre processus, alors $e \rightarrow e'$.
3. Si $e \rightarrow e'$ et $e' \rightarrow e''$ alors $e \rightarrow e''$.

Cette relation définit un ordre partiel sur les événements. Deux événements distincts, *e* et *e'*, sont dits ***concurrents*** ou non comparables, noté ***e*** || ***e'***, si $e \not\rightarrow e'$ et $e' \not\rightarrow e$. On peut alors associer à un événement *e* trois ensembles d'événements :

- ***Passé(e)***: ensemble des événements antérieurs à *e* dans l'ordre causal (*e* appartient à cet ensemble);
- ***Futur(e)***: ensemble des événements postérieurs à *e* dans l'ordre causal (*e* appartient à cet ensemble);
- ***Concurrent(e)*****:** ensemble des événements concurrents avec *e*.

Une manière adéquate pour visualiser les calculs distribués est *les diagrammes de temps*. La figure 2.1 montre un exemple pour un calcul comprenant trois processus (un processus par site), où la progression de chaque processus est symbolisée par une ligne horizontale. Le temps global est assumé de s'évoluer de gauche à droite. Les événements sont symbolisés par des points sur les lignes processus, et ceci selon leur ordre d'occurrence. Les messages sont représentés par des flèches connectant les événements d'émission avec leurs événements de réception correspondants.

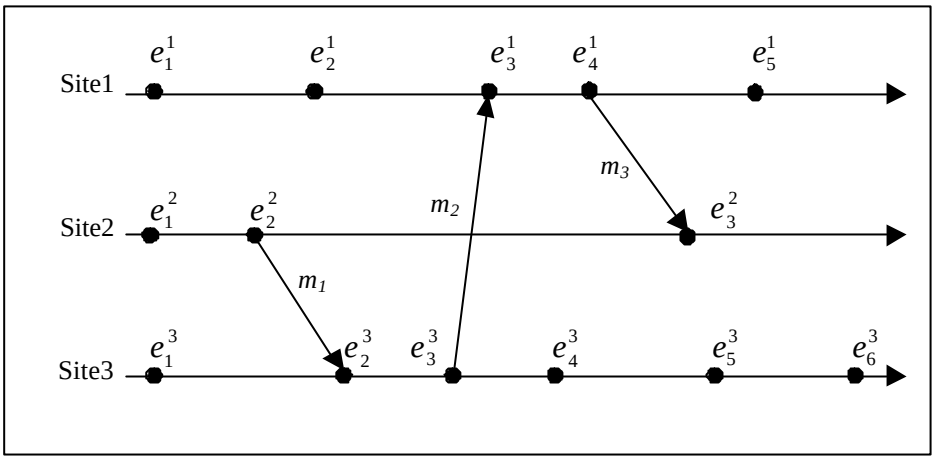


Figure 2.1 – *Un diagramme de temps d'un calcul distribué*

L'application de la relation *happened before* sur cet exemple conduit à tirer les résultats suivants :

- En appliquant la première règle, on déduit les relations suivantes entre les événements internes à chaque processus :
  - Sur le site 1 : $e_1^1 \longrightarrow e_2^1 \longrightarrow e_3^1 \longrightarrow e_4^1 \longrightarrow e_5^1$
  - Sur le site 2 : $e_1^2 \longrightarrow e_2^2 \longrightarrow e_3^2$
  - Sur le site 3 : $e_1^3 \longrightarrow e_2^3 \longrightarrow e_3^3 \longrightarrow e_4^3 \longrightarrow e_5^3 \longrightarrow e_6^3$
- L'application de la deuxième règle introduit les relations suivantes entre les événements d'émission et de réception :
  - Pour le message m_1 : $e_2^2 \longrightarrow e_2^3$
  - Pour le message m_2 : $e_3^3 \longrightarrow e_3^1$
  - Pour le message m_3 : $e_4^1 \longrightarrow e_3^2$
- L'ordre partiel définit sur les événements de ce calcul distribué est :

$$e_1^2 \longrightarrow e_2^2 \longrightarrow e_2^3 \longrightarrow e_3^3 \longrightarrow e_3^1 \longrightarrow e_4^1 \longrightarrow e_3^2$$

Afin de dater (estampiller) les événements, des horloges logiques de différents types ont été définies afin de rendre compte de la relation de dépendance (i.e. la relation *happened before*) entre les événements.

2.3.1 Principe des horloges logiques

Dans une exécution d'un système distribué, où toutes les actions sont modélisées par des événements, rien ne peut se produire entre deux événements successifs. D'ici, le temps a besoin seulement d'être avancé avec l'occurrence d'un événement et est donc

« discret ». Puisque les évènements sont supposés être non durables et se produisent à des instants spécifiques dans le temps, une fonction $C : E? T$ doit être trouvée dont le rôle est d'assigner une estampille $C(e)$ d'un « domaine temps T à chaque événement e de l'ensemble d'événements E [47]. La comparaison des différents événements doit permettre de tirer certaines conclusions sur la relation entre les événements. Il semble d'être plausible qu'au moins les notions « antérieur » et « ultérieur » doivent exister dans n'importe quelle notion de temps. D'ici, le temps logique doit être un ordre partiel (T, ?). D'un point de vu abstrait, la fonction C peut être appelée « horloge logique ». Un besoin raisonnable sur C est quelle soit conforme à la relation de causalité :

$$\forall e, e' \in E : e \rightarrow e' \Rightarrow C(e) < C(e').$$

Cette importante propriété est souvent dite la condition de l'horloge (*Clock condition*). Dite verbalement : « un événement e doit avoir une estampille plus petite qu'un événement e' si e peut causalement affecter e' ». Comme conséquence de la condition de l'horloge, on trouve les propriétés suivantes :

1. Pour chaque processus, le temps est incrémenté de manière monotone,
2. Le temps logique d'un événement d'émission est toujours antérieur au temps logique de l'événement de réception correspondant.

Parmi les mécanismes d'estampillage qui ont été proposés pour les systèmes distribués, on cite les mécanismes suivants :

2.3.2 Les horloges logiques linéaires (scalaires)

Cette représentation de temps a été proposée par Lamport en 1978 [43]. Chaque site gère un compteur dont la valeur est un entier : les valeurs des horloges sont donc comparables en tant que valeurs entières. Sur chaque site ce compteur est initialisé à 0 au lancement du système. La valeur de l'horloge logique d'un site est incrémentée chaque fois qu'un événement local s'y produit : un tel événement est soit une opération purement locale, soit l'envoi d'un message. Dans ce dernier cas la valeur courante (après incrémentation) de l'horloge de l'émetteur est attachée au message ("piggybacking") et sert à l'estampiller.

La réception d'un message permet de synchroniser l'horloge du récepteur avec celle de l'émetteur du message qui est transportée par le message. Le principe est simple : il consiste à attribuer à l'horloge du récepteur une valeur supérieure à la fois à la valeur courante de l'horloge du site et à celle de l'estampille du message reçu.

En résumé, si HL_i désigne l'horloge logique du site i et EL_m désigne l'estampille logique attribuée au message m lors de son envoi alors:

- Si un événement purement local se produit sur le site i, HL_i est incrémentée ($HL_i = HL_i + d, d > 0$ *et peut prendre une valeur différente pour chaque incrémentation*) ;
- Si un événement correspondant à l'envoi d'un message se produit sur le site i, HL_i est incrémentée et le message m est envoyé avec la nouvelle valeur de HL_i comme estampille ($EL_m = HL_i$) ;

- Si un événement correspondant à la réception d'un message *m* d'estampille EL_m se produit sur le site *i*, $HL_i = max\ (HL_i, EL_m\ ) + d$.

La figure suivante donne la valeur des estampilles logiques scalaires associées en appliquant ce mécanisme à des événements de 3 sites entre lesquels des messages estampillés sont échangés.

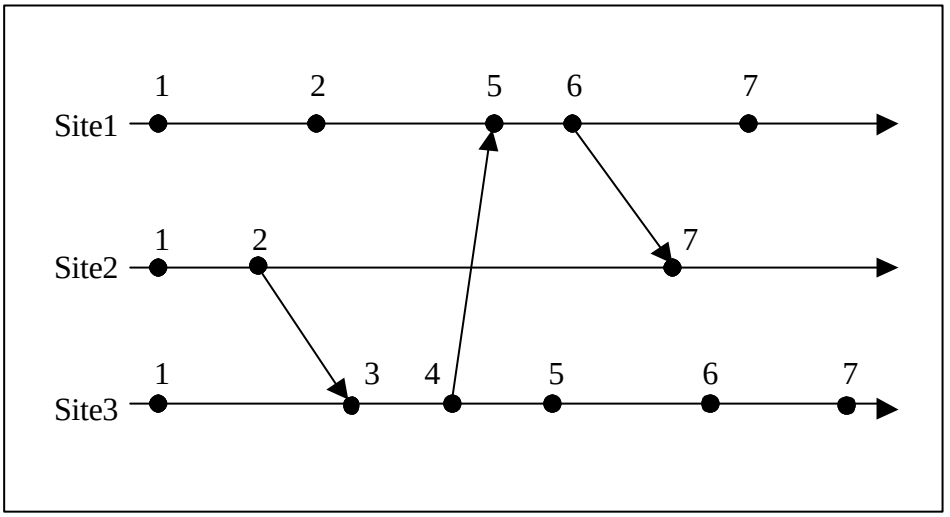


Figure 2.2 – *La progression des horloges linéaires*

a. Propriétés

En plus de la propriété de monotonie, il est possible d'utiliser ce mécanisme d'estampillage pour construire un ordre total "*t-before*" sur l'ensemble des événements. L'estampille d'un événement est alors composée de sa date d'occurrence et de l'identité du site qui l'a produit (par exemple un entier attribué artificiellement ou une adresse IP ou physique). La règle de comparaison des dates de deux événements *e* et *e'* estampillés respectivement par (HL_i, i) et (HL_j, j) est alors :

$$e\ t\text{-}before\ e' \Leftrightarrow (HL_i < HL_j \text{ ou } (HL_i = HL_j \text{ et } i < j))$$

Si on considère que la valeur d'incrémentation *d* est égale à 1 et si *e* est un événement du site *i* estampillé par Hl_i, alors $Hl_i - 1$ est la durée logique minimale nécessaire avant la production de l'événement *e* [44].

2.3.3 Les horloges logiques vectorielles

On vient de voir que le système de datation par estampilles scalaires introduisait un ordre artificiel sur des événements concurrents. Il est néanmoins utile de pouvoir déterminer la dépendance ou l'indépendance causale entre deux événements :

- Mise au point de programme répartis. La dépendance entre deux événements est une notion centrale pour la recherche des causes potentielles d'une erreur.

- Placement d'une application répartie. La mesure du degré d'indépendance des calculs est nécessaire pour évaluer les gains liés à la mise en œuvre d'un calcul sur différents processeurs.
- Réalisation efficace d'algorithme ne nécessitant pas un ordre total.

Le mécanisme de datation par estampilles vectorielles introduit indépendamment par Fidge et Mattern en 1988 [45, 46, 49] permet de pallier à cet inconvénient.

Chaque site gère une horloge vectorielle constituée de n entiers, notée $HV_i[1..n]$ (n est le nombre de composants du système) où $HV_i[i]$ est l'horloge locale du site i et $HV_i[j]$ représente la dernière connaissance du site i sur le temps local du site j. Une telle horloge permet de dater les événements d'un site et est mise à jour lors de l'occurrence des évènements. Comme pour les horloges scalaires, les messages envoyés par un site sont estampillés en utilisant la valeur courante de l'horloge vectorielle du site émetteur et la réception d'un message permet au site récepteur de synchroniser son horloge vectorielle avec celle du site émetteur du message.

De manière plus précise, les règles suivantes sont appliquées pour la gestion des horloges vectorielles : soit HV_i l'horloge vectorielle du site i et EV_m l'estampille vectorielle attribuée au message m lors de son envoi, alors :

- Si un événement purement local se produit sur le site i, $HV_i[i]$ est incrémentée ($HV_i[i]++$) ;
- Si un événement correspondant à l'envoi d'un message se produit sur le site i, $HV_i[i]$ est incrémentée et le message m est envoyé avec la nouvelle valeur de HV_i comme estampille ($EV_m = HV_i$) ;
- Si un message m, d'estampille EV_m, est reçu sur le site i :
  - $HV_i[i]$ est incrémentée ;
  - $\forall j \neq i, HV_i[j] = max(HV_i[j], EV_m[i])$.

La figure suivante donne la valeur des estampilles vectorielles associées aux différents événements des 3 sites : en appliquant la méthode qu'on vient de décrire :

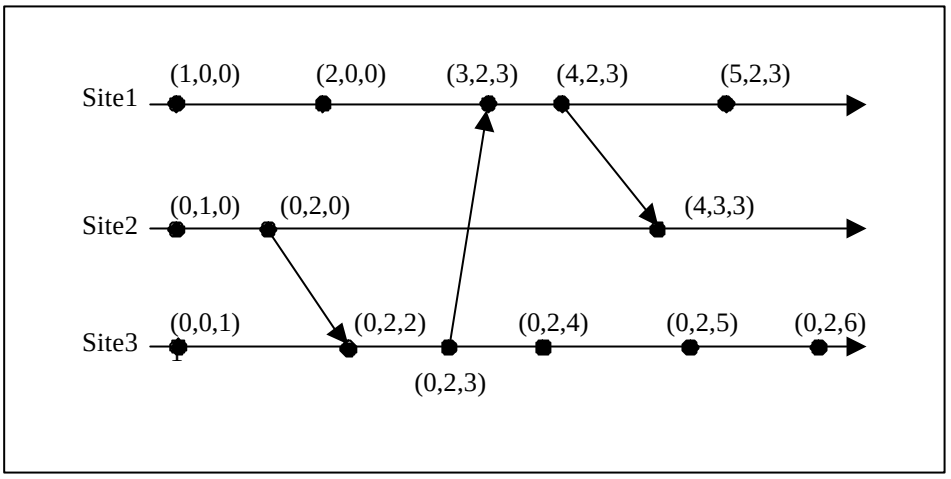


Figure 2.3 – *La progression des horloges vectorielles*

a. Propriétés

- La propriété fondamentale que possèdent les horloges vectorielles est que : pour EV_e l'estampille vectorielle de e on a : $\forall\ i,\ EV_e[i] = Card(\{é / é\ \hat{I} S_i \text{ et } é \rightarrow e\})$, c'est à dire que la valeur de la $i^{éme}$ composante de EV_e est exactement le nombre d'événements du site S_i qui appartiennent au passé de e [44].
- La relation d'ordre suivante peut par ailleurs être définie sur les estampilles vectorielles:
  - $EV_e \leq EV_{e'} \Leftrightarrow \forall\ i,\ EV_e[i] \leq EV_{e'}[i]$,
  - $EV_e < EV_{e'} \Leftrightarrow EV_e \leq EV_{e'}$ et $\exists\ i,\ EV_e[i] < EV_{e'}[i]$,
  - $EV_e \parallel EV_{e'} \Leftrightarrow non(EV_e < EV_{e'})$ *et* $non(EV_{e'} < EV_e)$

  Muni de cette relation d'ordre, le système de datation par estampilles vectorielles a la remarquable propriété de refléter exactement la relation de précédence causale entre événements. Soit EV_e l'estampille de e :
  - $e \leq e' \Leftrightarrow EV_e \leq EV_{e'}$;
  - $e \parallel e' \Leftrightarrow EV_e \parallel EV_{e'}$.

  En d'autres termes, il y a un *isomorphisme* entre l'ensemble des événements partiellement ordonnés produits par le calcul distribué et leurs estampilles [44].

2.3.4 Les horloges logiques matricielles

Cet autre mécanisme de datation des événements va permettre de détecter l'arrivée de messages dont la délivrance violerait le respect de la causalité et permettra donc d'en différer la délivrance.

Avec ce système de datation, dans un système de n sites, l'horloge d'un site i et les estampilles des événements (et des messages échangés entre les sites) sont des matrices carrées d'ordre n. Soit HM_i l'horloge matricielle du site S_i, EM_m l'estampille matricielle

du message *m* et EM_e l'estampille matricielle d'un événement *e*. Sur le site S_i, la matrice HM_i va :

- Mémoriser le nombre de messages que le site S_i a envoyés aux différents autres sites: cette information est fournie par la $i^{ème}$ ligne de la matrice. L'élément diagonal compte les événements sur le site S_i ;
- Mémoriser, pour chacun des autres sites *j*, le nombre de messages émis par ce site dont le site *i* a connaissance (c'est-à-dire dont le site S_i sait qu'ils ont été envoyés). Ainsi sur le site *i*, la valeur $EM_i[j, k]$ donne le nombre de messages en provenance du site S_j délivrables sur le site S_k dont le site S_i a connaissance (c'est-à-dire dont l'envoi est causalement antérieur à tout ce qui arrivera dorénavant sur S_i).

En fait la rangée HMi[i, .] n'est que l'horloge vectorielle HVi[.] ; donc cette rangée hérite les propriétés des horloges vectorielles.

L'avancement des horloges est réalisé de la manière suivante :

- Lorsqu'un événement local se produit sur le site S_i, $HM_i[i, i]$ est incrémentée ;
- Lorsqu'un message *m*, d'estampille EM_m, en provenance du site S_j est reçu sur le site S_i :
  - $1 \le k \le n : HM_i[i, k] = max (HM_i[i, k], EM_m[j, k])$;
  - $1 \le k, l \le n : HM_i[k, l] = max (HM_i[k, l], EM_m[k, l])$;

La figure suivante modélise la progression de ces horloges dans un système distribué constitué de 3 sites :

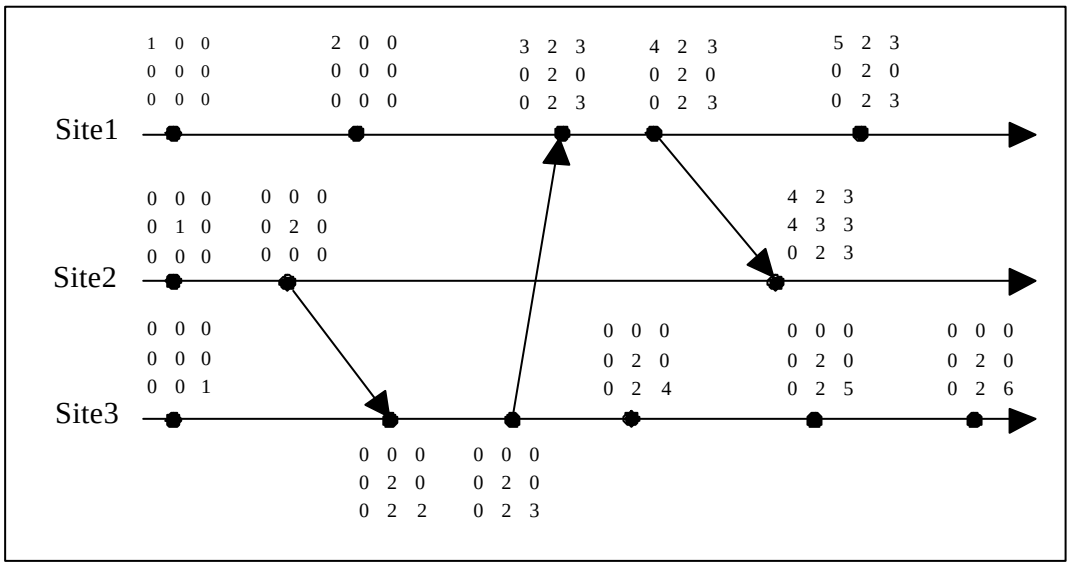


Figure 2.4 – *La progression des horloges matricielles*

Comme nous avons a déjà mentionné, le traçage des relations causales entre les activités concurrentes est un mécanisme de base pour une variété d'application telles que la gestion des données répliquées, l'observation d'un système distribué, l'allocation de ressources, systèmes multimédia et téléconférence. Et suivant la définition de la causalité comme un ordre partiel sur les événements, il est bien établi que dans un système distribué la dépendance causale peut être totalement caractérisée par l'utilisation des techniques d'estampillage par vecteur logique.

Ces présentations ont fourni une modélisation correcte de l'ordre partiel pour les systèmes distribués. Cependant, ces horloges ne sont pas conformes pour les systèmes mobiles, comme on va le montrer dans la section suivante, et ceci est dû au *(i)* manque d'évolution : leur taille est égale au nombre de sites, et *(ii)* leur inadéquation à la fluctuation du nombre des sites.

2.4 Insuffisance des horloges vectorielles pour un système mobile

Avoir des horloges vectorielles avec une entrée pour chaque *MH* peut fournir une information suffisante pour caractériser la relation causale. Cependant, la taille du vecteur deviendra probablement importante à émettre même sur un réseau filaire à bande passante élevée. De plus, la population des sites mobiles fluctue avec le temps ; les sites mobiles joignent le réseau, restent connectés pour une période de temps, et puis se déconnectent, soit temporairement ou de manière permanente. Ces fluctuations ne peuvent pas être traitées par les horloges logiques déjà proposées du fait qu'elles exigent que le nombre de sites dans le système reste inchangé.

Du fait que toute communication incluant des sites mobiles prend place à travers la *MSS* de la cellule dans laquelle ils sont présents, donc les relations de causalité peuvent être tracées en utilisant des horloges vectorielles de n composantes, où n est le nombre des MSS_s dans le système. Ceci peut amener à une réduction significative dans les taux *(overheads)* de communication, car le nombre des MSS_s est généralement une petite fraction du nombre total des nœuds (MH_s et MSS_s) dans ce type de système. Cependant, utilisant seulement n entiers dans une horloge vectorielle est insuffisant pour tracer toutes les dépendances causales. La preuve de cette affirmation est donnée par l'exemple suivant (Figure 2.5):

Comme il y a deux MSS_s dans le système, l'horloge vectorielle a deux composantes. Initialement, les horloges vectorielles des deux MSS_s sont initialisées à *(0,0)*. MH_b envoie un message m_1 à MH_c. Donc, MH_b envoie le message à MSS_p. MSS_p associe une estampille vectorielle de *(1,0)* avec l'événement $Send(m_1)$, et retourne le message à MSS_q. Quand m_1 arrive à MSS_q, MSS_q met à jour son horloge vectorielle à *(1,1)*, associe cette estampille avec l'événement $Recv(m_1)$, (*i.e.*, $V(Recv(m_1)) = (1,1)$) et renvoie m_1 à MH_c.

MH_d envoie un message m_2 à MH_a. Donc, MH_d envoie m_2 à MSS_q. MSS_q reçoit m_2 après la réception de m_1. D'ici, MSS_q mis à jour son horloge vectorielle à *(1,2)* et associe cette estampille avec l'événement $Send(m_2)$, *i.e.* $V(Send(m_2)) = (1,2)$.

Comme il n'y a pas de relation causale entre $Recv(m_1)$ et $Send(m_2)$, les deux événements sont concurrents. Cependant, $V(Recv(m_1)) < V(Send(m_2))$ impliquant que $Send(m_2)$ est causalement dépendant de $Recv(m_1)$: une perte de la propriété d'isomorphisme des horloges vectorielles.

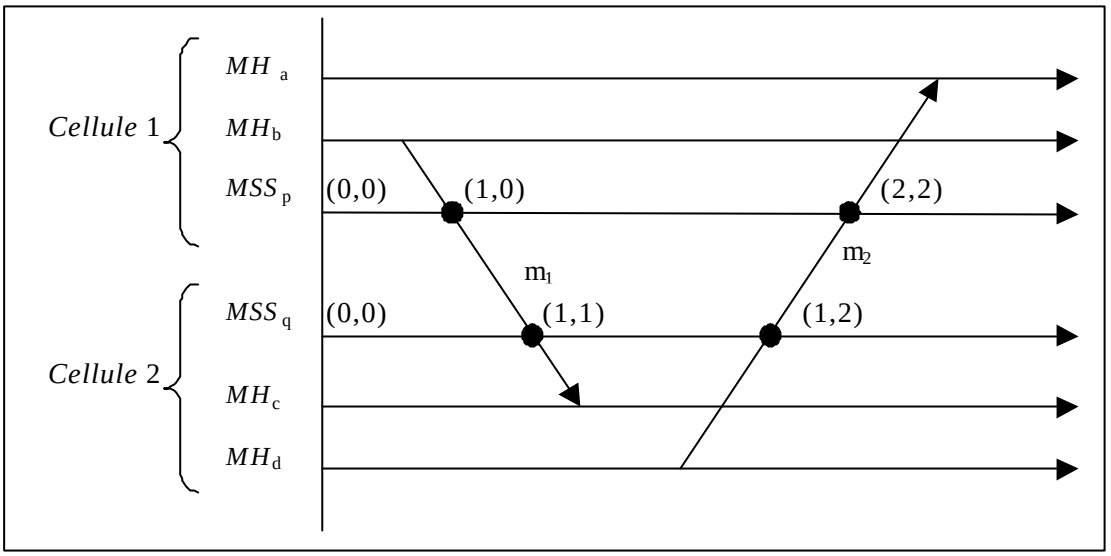


Figure 2.5 – *Une horloge vectorielle avec plus de deux entrées est nécessaire pour un système à deux cellules*

D'après cet exemple, on peut conclure que lorsqu'un entier est utilisé par cellule pour représenter la causalité, la *MSS* d'une cellule augmente sa composante de l'horloge vectorielle pour chaque événement d'*EMISSION* et de *RECEPTION* qu'elle observe même si ces événements sont mutuellement concurrents. Donc, au niveau de la cellule, il y a une sérialisation implicite des événements. La *MSS* est incapable de maintenir l'information sur la concurrence mutuelle des événements produits dans les différents MH_s de la cellule quand aucune dépendance causale n'existe entre eux.

Puisque cette perte d'information n'est pas admissible, alors un recours à d'autres alternatives plus efficaces pour le traçage de la causalité dans les systèmes mobiles apparaît indispensable.

2.5 Les alternatives des mécanismes existants pour les systèmes mobiles

L'utilisation d'une horloge vectorielle avec un nombre de composants *n* égal au nombre des MSS_s est insuffisante pour capturer les dépendances causales entre les MH_s. Cependant, la présence de la *MSS* dans chaque cellule pour séquentialiser les émissions et les réceptions des MH_s dans la cellule indique qu'une horloge vectorielle de taille *N (nombre total des nœuds, mobiles et fixes)* n'est pas nécessaire pour capturer les dépendances causales.

Il est clair qu'un mécanisme de traçage de la causalité qui est limité au nombre des MSS_s et qui évite une communication coûteuse sur les liens *MH-MSS* doit aboutir à une implémentation plus évolutive et efficace sur cette classe de systèmes distribués. Avec cet objectif à l'esprit, *Prakash* et *Singhal* [50] ont proposé deux alternatives pour les horloges vectorielles, pour les systèmes mobiles, dites *les séquences de dépendance (dependency sequences)*et *les horloges hiérarchiques (hierarchical clocks)*. Une autre alternative a été proposée par la suite par *Baquero* et *Moura* [52] comme une

amélioration de l'approche *horloges hiérarchiques*, nommée *les horloges entrelacées (interleaving clocks).*

Nous présentons dans les paragraphes suivants la définition de ces trois alternatives, et leur traitement des déconnexions *(handling Handoffs)* :

2.5.1 L'approche des séquences de dépendances

C'est une représentation compressée des histoires causales.

a. Définition (histoire causale)

Soit $E=E_1 \cup E_2 \cup \ldots \cup E_N$ l'ensemble des événements dans le calcul distribué, où N est le nombre des MSS_s. E_i est l'ensemble des événements qui sont sérialisés par une MSS_i et se produisent dans les MH_s qu'elle supporte. Pour un événement $e \in E$ se produisant dans le calcul distribué, l'histoire causale de e, dénotée $C(e)$, est définie par [46]:

$$C(e) = \{e' \in E / e' \rightarrow e\} \cup \{e\}$$

La projection de $C(e)$ sur E_i, notée $C(e)[i]$, est définie par :

$$C(e)[i] = C(e) \cap E_i.$$

Une modélisation de l'histoire causale est crée en associant à chaque événement N ensembles des événements causalement précédents, où chaque ensemble rassemble les événements qui sont enregistrés dans une *MSS* particulière. Dans cette représentation, les événements dans chaque ensemble peuvent être représentés par leur index d'ordonnancement (un entier) puisqu'ils appartiennent tous à la même *MSS*. La présentation de Schwarz et Mattern des histoires causales prouve que l'histoire causale de deux événements est suffisante pour caractériser la relation de causalité entre eux. Elle fournit aussi une procédure pour créer l'histoire causale de chaque événement dans le calcul. De manière Informelle, l'algorithme fait apparaître que les événements internes et les événements d'émission héritent l'histoire causale de leur prédécesseur immédiat, et les événements de réception héritent les histories causales de leurs deux prédécesseurs. La Figure 2.6 illustre les histoires causales dans un système mobile avec deux *MSSs* et trois *MHs* : *A* et *B* dans la MSS_1 et *C* dans la MSS_2. Les relations *i* et *m* représentent la causalité introduite par les événements internes et les messages, respectivement.

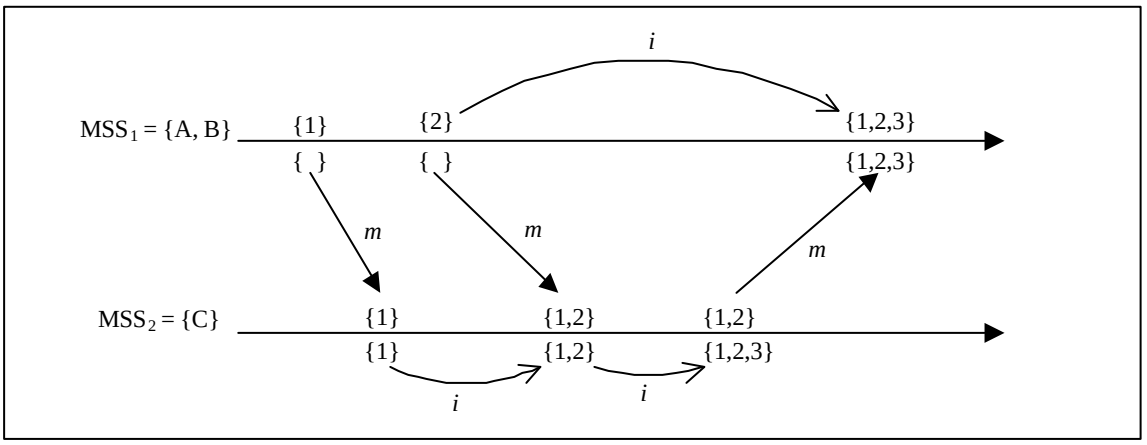


Figure 2.6 – *Les histoires causales dans un système mobile*

b. Les séquences de dépendance

La *MSS* d'une cellule joue le rôle du *proxy* pour tous les MH_s présents dans sa cellule. Tous les événements causant la dépendance (*EMISSION* et *RECEPTION* de messages) observés par une *MSS*, sont assignés des nombres de séquences croissants. Initialement, le compteur séquence est initialisé à 0. Si e_i et e_j sont deux événements successifs causant la dépendance et observés par *MSS*, alors le nombre de séquence assigné à e_j est égal au nombre de séquence assigné à e_i plus un.

Les prédécesseurs causaux d'un événement *e* sont représentés comme un ensemble de séquences de dépendance. Chaque séquence de dépendance dans l'ensemble correspond à une cellule dans le système, et se compose d'une séquence d'entiers non négatifs. Les paires de ces entiers représentent des séquences contiguës d'événements causant la dépendance dans une cellule qui sont des prédécesseurs causaux de *e*.

Comme les événements internes des sites mobiles ne créent pas des dépendances causales, ils ne sont pas considérés. La cardinalité de l'ensemble est égale au nombre des cellules dans le système. Initialement, chaque séquence dans l'ensemble est vide. Il n'y a pas de borne supérieure pour la longueur d'une séquence. Cependant, la longueur de chaque séquence est généralement paire, et les entiers dans la séquence sont rangés par ordre croissant.

Le premier entier dans la séquence de dépendance de *e*, par rapport à une cellule, dénote le premier événement causant la dépendance observé par cette *MSS* qui est dans le *Past(e) (i.e. le passé de e et défini par Past(e) = {e' / e'→e}).* Tous les événements causant la dépendance de la cellule dont les nombres de séquence sont supérieurs ou égaux au premier entier et inférieurs ou égaux au second entier dans la séquence sont dans le passé de *e*. Tous les événements causant la dépendance de la cellule dont les nombres de séquence sont supérieurs ou égaux au second entier et inférieurs ou égaux au troisième entier ne sont pas dans le passé de *e*, et représentent la *dependency gap*. De manière similaire, tous les événements causant la dépendance avec des nombres de séquence entre le troisième et le quatrième entier dans la séquence sont dans le passé de *e* et ainsi de suite.

La figure suivante *(Figure 2.7)* illustre l'approche des séquences de dépendances. Les événements d'émission et de réception de messages *0 – 4, 6 –12, 14 – 17, et 19 – 20* dans la cellule C_1, représentés par des rectangles, sont des prédécesseurs causaux de l'événement *e*. les événements *5, 13, et 18* dans la cellule C_1 ne sont pas des prédécesseurs causaux de *e*, et correspondent aux *gaps* dans la séquence. Les séquences de dépendance de *e*, pour ce système à 4 cellules sont *[{0,4,6,12,14,17,19,20}, {0,6,9,11,14,16}, {0,3,7,15,21,33}, {0,3,5,7,9,9,11,17}]*.

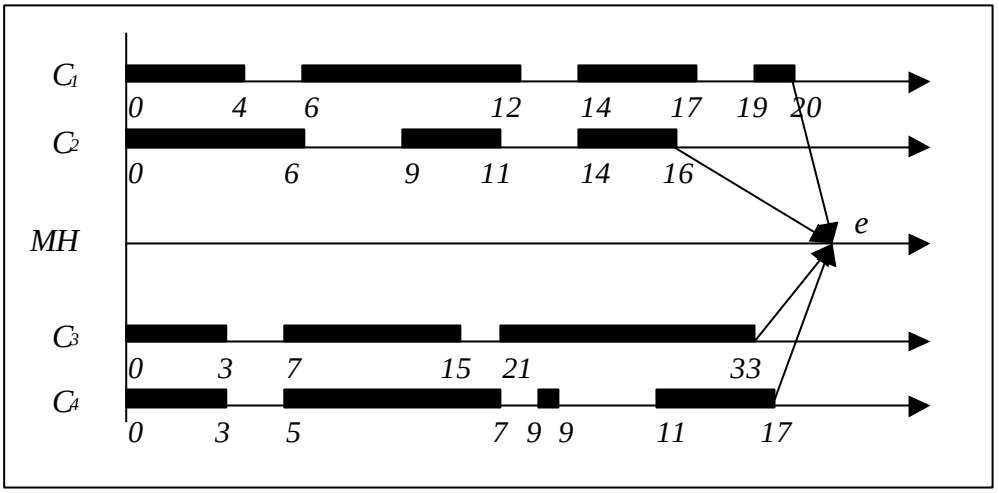


Figure 2.7 – *Les séquences de dépendances pour un événement e*

c. La gestion des séquences de dépendance

L'algorithme pour la gestion des séquences de dépendance assure que les événements d'émission héritent la séquence de dépendance du dernier événement à partir de ce *MH*, et c'est cette nouvelle *DS* (dependency sequence) qui est envoyée avec le message. Les événements de réception héritent la *DS* du dernier événement à partir du *MH* récepteur et la fusionnent avec la *DS* qui est reçue avec le message, assurant ainsi que les deux dépendances sont combinées.

La station de base de la cellule *P*, notée par MSS_p, maintient les séquences de dépendance de tous les MH_s dans la cellule. L'ensemble des séquences de dépendance de MH_a est noté par DS_a. DS_a a *n* composantes, où *n* est le nombre des cellules dans le système : une composante pour chaque cellule. La $P^{éme}$ composante correspondant à la cellule *P*, est notée par $DS_a\ [P]$. Chaque *MSS* maintient aussi un compteur entier pour garder la trace des événements d'émission et de réception de messages observés par elle, depuis le début du calcul. $DS_a\ [P]$ est initialisée par une séquence vide, pour $0 \leq P \leq n-1$, et compteur est initialisé à *0*.

Les fonctions suivantes sont supposées être définies pour les séquences d'entiers :

last (DS_a [*P*]) : retourne le dernier entier de la séquence DS_a [*P*].

concatenate (seq_1, seq_2) : ajoute seq_2 à la fin de seq_1, où seq_1 et seq_2 sont toutes les deux des séquences d'entiers, et retourne la séquence résultante.

Quand MH_a, résidant dans la cellule *P* envoie un message à MH_b, le message est d'abord envoyé sur un canal sans fil à MSS_p.

Action de MSS_p en recevant le message à partir de MH_a :

```
{
  counter ¬ counter + 1 ;
  if (last(DS_a[P]) = counter – 1)
       last(DS_a[P]) ¬ counter ;
  else
       DS_a[P] ¬ concatenate(DS_a[P], {counter, counter}) ;
```

send message and DS_a over the wireline network to MH_b's service station, namely MSS_q ;
}

Action de MSS_q en recevant le message et DS_a envoyés par MH_a pour MH_b :

```
{
  counter ¬ counter + 1 ;
  if (last(DS_b[q]) = counter – 1)
       last(DS_b[q]) ¬ counter ;
  else
       DS_b[q] ¬ concatenate(DS_b[q], {counter, counter}) ;
  for all 0 £ i< n
       DS_b[i] ¬ merge(DS_b[i], DS_a[i]) ;
}.
```

d. Traitement des déconnexions

Quand un site mobile se déplace de la cellule *p* vers la cellule *q*, la responsabilité pour maintenir ses séquences de dépendances causales se déplace de MSS_p vers MSS_q. Ceci est traité comme suit. Durant la déconnexion, MH_a s'enregistre avec MSS_q et se désenregistre avec MSS_q. Les séquences de dépendance pour MH_a, notées DS_a, sont alors transférées à travers le réseau filaire de MSS_p vers MSS_q. Comme résultat, MSS_q a maintenant une information complète sur tous les prédécesseurs causaux de l'état courant de MH_a. A partir de ce point, si MH_a nécessite d'envoyer un message, il communique le message à MSS_q qu'elle le diffuse vers la destination. De plus, si un nœud dans le système veut envoyer un message à MH_a, le message est routé à MSS_q. En recevant le message, MSS_q utilise les séquences de dépendance accompagnantes pour mettre à jour DS_a, et diffuse le message à MH_a à travers un canal sans fil.

Ainsi, les déconnexions sont traitées de manière directe. Les séquences de dépendance sont transférées à travers le réseau filaire à bande passante relativement élevée. Les canaux sans fil ne sont pas saturés par le trafic de déconnexion. Les fluctuations dans le nombre des sites mobiles dans le système n'ont aucun impact sur la maintenance des séquences de dépendance car le nombre de chaque séquence ne dépend que du nombre des MSS_s, que l'on suppose fixé.

2.5.2 L'approche des horloges hiérarchiques

La quantité importante d'informations stockée avec les événements, par l'approche des séquences de dépendance, conduit à l'échange d'un taux considérable de données quand le système est tracé. Les horloges hiérarchiques sont présentées comme une alternative qui fournit un volume d'échange de données réduit lors de la construction des traces. Ceci veut dire que la donnée nécessaire pour comparer deux événements est distribuée sur les autres événements. Ceci implique, dans le cas général, une communication entre les MSS_s quand les événements sont comparés.

Les horloges hiérarchiques sont présentées comme une technique pour la caractérisation de la relation causale dans les systèmes où les processus ne nécessitent

pas un ordre total [51]. L'abstraction d'un comportement de cellule par un processus avec un ordonnancement partiel entre les événements est une violation de la définition standard d'un processus où tous les événements sont totalement ordonnés. Ainsi, la relation *happened before (→)* de Lamport ne peut pas capturer complètement la relation de dépendance causale. A la place, une nouvelle relation, $\longrightarrow$ est définie sur un ensemble arbitraire d'événements e_1, e_2, et e_3 comme suit :

1. $e_1 \rightarrow e_2$ si $e_1 \xrightarrow{i} e_2$, i.e., les événements e_1 et e_2 se produisent sur le même processus abstrait et il y a une dépendance causale entre e_1 et e_2.
2. $e_1 \rightarrow e_2$ si $e_1 \xrightarrow{m} e_2$, i.e., e_1 est un événement d'émission d'un message dans un processus abstrait et e_2 est l'événement de réception correspondant dans un autre processus abstrait.
3. $e_1 \rightarrow e_2$ si $\exists\, e_3 : e_1 \rightarrow e_3$ et $e_3 \rightarrow e_2$.

a. Description de l'horloge

En se basant sur l'abstraction de processus décrite ci-dessus, à chaque événement du processus abstrait est assigné une valeur horloge hiérarchique pour capturer la relation $\longrightarrow$ entre les événements. L'horloge hiérarchique f est maintenue dans les MSS_s, et a deux composantes :

1. f^i est une horloge locale représentant la relation $\xrightarrow{i}$. C'est un *vecteur bit* à longueur variable avec un bit pour chaque événement qui est produit dans le processus jusqu'à ce point. A chaque événement (émission ou réception de messages) observé par le processus abstrait est assigné un nombre de séquence unique croissant. Soient $e_{i,x}$ et $e_{i,y}$ le x^{em} et le y^{em} événements du processus abstrait P_i, respectivement. $f^i(e_{i,x})$ est de longueur *x bits* et son x^{em} bit est mis à *1*. De plus, $f^i(e_{i,x})$ est généré par l'application d'un *OU logique* entre les vecteurs $f^i(e_{i,y})$ pour tous les événements $e_{i,y}$ tel que $e_{i,y} \xrightarrow{i} e_{i,x}$. Par conséquent, pour chaque événement local du processus P_i qui précède causalement $e_{i,x}$, le bit correspondant est mis à 1 dans $f^i(e_{i,x})$.
2. f^m est une horloge globale représentant la relation $\xrightarrow{m}$. C'est un vecteur entier à n composantes, une composante pour chaque processus abstrait (cellule) dans le système. $f^m(e_{i,x})[k]$, la $k^{éme}$ composante de l'horloge globale de l'événement $e_{i,x}$, identifie le dernier événement sur le processus k qui précède causalement $e_{i,x}$.

Pour un événement d'émission de messages $e_{i,x}$, la *MSS* de la cellule d'émission fait l'action suivante pour mettre à jour l'horloge hiérarchique du processus abstrait correspondant : $f^m(e_{i,x})[k]$, où $k \neq i$, est mis au maximum des $k^{éme}$ composantes de tous les $e_{i,y}$ tel que $e_{i,y} \xrightarrow{i} e_{i,x}$. $f^m(e_{i,x})[i]$ est mis à x. $f^m(e_{i,x})$ est envoyée avec le message comme son estampille vectorielle. Pour un événement de réception de messages $e_{i,x}$, la *MSS* de la cellule réceptrice fait l'action suivante : $f^m(e_{i,x})[k]$ $(k \neq i)$ est mis au maximum des $k^{éme}$ composantes de tous les $e_{i,y}$ tel que $e_{i,y} \xrightarrow{i} e_{i,x}$ et la $k^{éme}$ composante de l'estampille vectorielle portée par le message. $f^m(e_{i,x})[i]$ est mis à x.

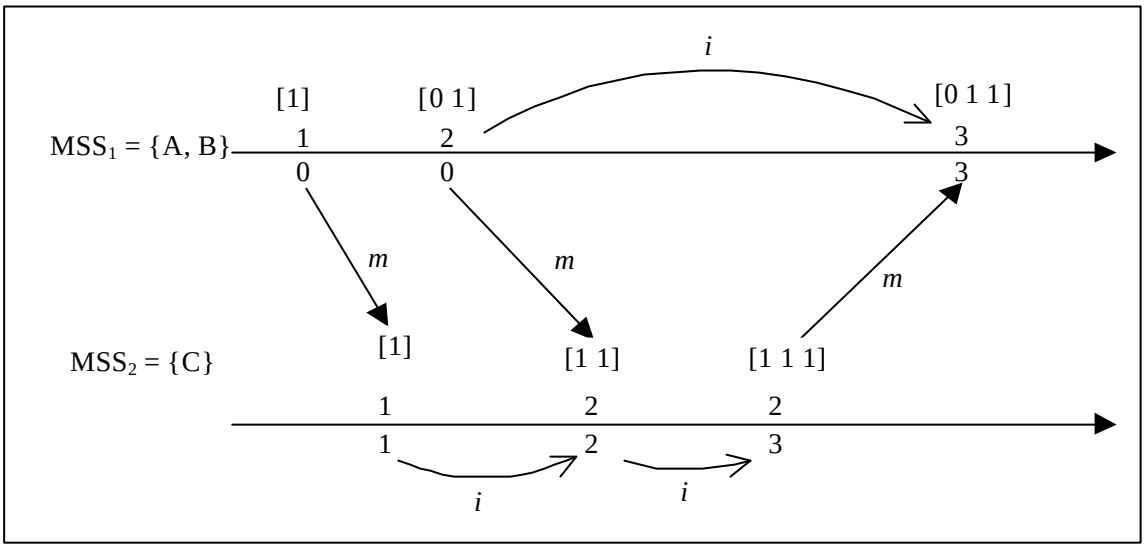


Figure 2.8 – *Les horloges hiérarchiques*

La figure ci-dessus (Figure 2.8) illustre le fonctionnement des horloges hiérarchiques. L'horloge locale f^i est représentée par la séquence bit *[…]* qui indique, pour chaque événement, lequel des autres événements est en fait dans son passé. L'horloge globale f^m est représentée par un vecteur.

b. Traitement des déconnexions

Quand un site mobile MH_a se déplace de la cellule C_i vers la cellule C_j, le processus de déconnexion est exécuté. MH_a se désenregistre de MSS_i en envoyant un message à MSS_i qui comporte l'information sur MSS_j, sa nouvelle *MSS*. En recevant le message de désenregistrement, MSS_i envoie un message d'enregistrement à MSS_j à travers le réseau filaire. Le message de registration comporte la valeur courante de f^m dans MSS_i. En réponse à l'événement d'arrivée du message de registration, les actions suivantes sont exécutées dans MSS_j :

1. $f^i(e_{i,y})$ est mis à jour pour refléter l'occurrence de $e_{i,y}$.

2. $f^m(e_{j,y})$ est calculé en prenant le maximum entre les composantes de f^m dans MSS_j avant l'arrivé du message de registration, et f^m reçue dans le message de registration.

Comme dans l'approche de séquences de dépendance, les fluctuations dans le nombre des MHs n'a aucun impact sur l'approche Horloge Hiérarchique et ceci pour la

même raison : les dépendances causales, dues aux communications, sont abstraites au niveau cellule au lieu du niveau Mh, et le nombre des cellules reste fixé.

2.5.3 L'approche des horloges entrelacées

La considération d'un système dont les événements d'un même processus sont totalement ordonnés, entraîne que les horloges hiérarchiques ne sont pas optimales pour une représentation efficace d'un système. Ceci est une conséquence naturelle de l'origine des horloges hiérarchiques, du fait qu'elles étaient conçues pour décrire un système qui offre un ordre partiel non restreint. Un ordre partiel non restreint est un ordre partiel qui n'est pas sujet à d'autres invariants qui restreignent de plus la définition de l'ordre partiel [52]. Quand un ordre partiel non restreint est considéré, il est nécessaire de prendre en compte des situations telles que avoir un événement qui dépend causalement de deux événements non causalement reliés.

Les MSS_s assignent un ordre total non réel aux événements de leurs MH_s, mais les événements de chaque *MH* sont eux-mêmes totalement ordonnés. Ce que la *MSS* fait est de centraliser l'affectation d'index croissants aux événements observés, regroupant ainsi les ordres totaux non reliés. Ceci peut être accompli en stockant avec chaque événement l'index du dernier événement dans son *MH*, ou *nil* pour le premier événement. On nommera cette approche *les horloges entrelacées* (Figure 2.9). On va décrire dans ce qui reste comment ce type d'horloge ajoute les informations perdues aux horloges vectorielles des MSS_s.

L'algorithme de construction, décrit ci-dessous, utilise un index temporaire *L()* qui identifie dans chaque *MSS* le dernier événement à partir de chacun de ses MH_s. Par exemple dans une *MSS* donnée et à un instant donné, *L(x)* indiquera l'index dans cette *MSS* du dernier événement qui concerne MH_x.

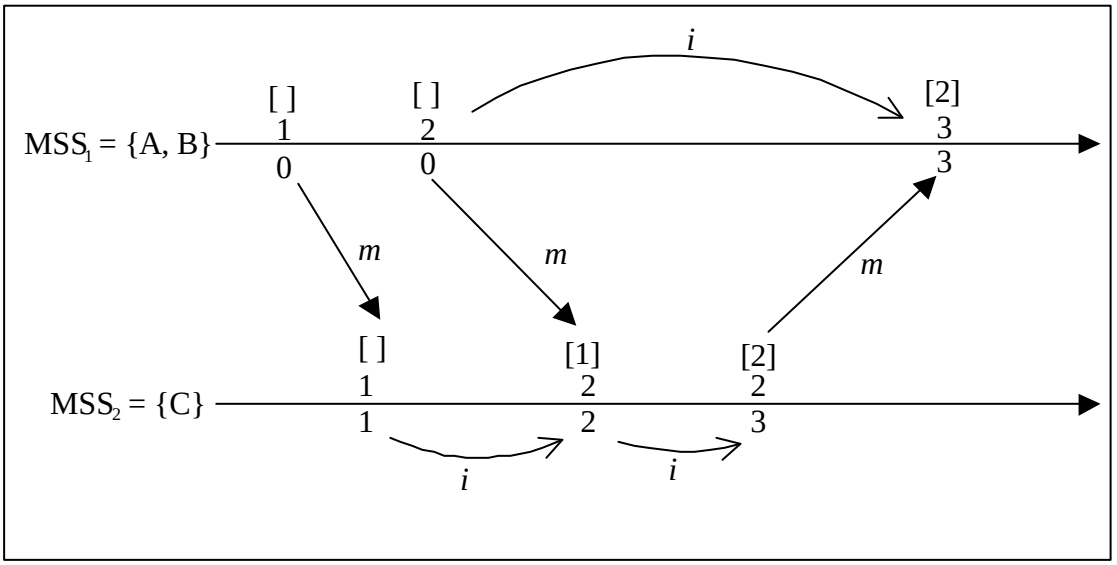


Figure 2.9 – *Les horloges entrelacées*

Par contre, les autres variables, *V()* et *P*, sont reliées aux événements. Elles représentent l'état tracé (comme il est représenté dans la Figure 2.9) tel que, dans la notation $\frac{[N]}{v}$ où *v* est un vecteur d'entiers, *P* est représenté par [*N*] et le vecteur *V()* par

v. L'algorithme est basé sur la dérivation de l'état du nouvel événement à partir du dernier événement tracé et de l'index temporaire *L().*

a. Présentation de l'algorithme

Soient *V()* une horloge vectorielle qui classe les événements sur un groupe de *N* MSS_s, et *P* un naturel qui indique le prédécesseur local d'un événement donné. Soit *L()* qui indique le dernier index événement pour un *MH* donné.

- Initialement,
  - *V(k) := 0* pour *k = 1, …,N*, et
  - *L()* est initialisée à *nil.*
- Quand une requête d'émission de message est reçue à partir de MH_i, dans une MSS_k,
  - *V(k) := V(k) + 1*,
  - *P := L(i)*, et
  - *L(i) := V(k).*

  Le message est émis avec le nouveau vecteur *V().*
- Quand un message est reçu avec $V_m()$ dirigé à MH_j par MSS_l,
  - *V(l) := V(l) + 1*,
  - *P := V(l)*,
  - *Ensuite, V() est mis à jour en prenant le maximum entre chaque paire du V() local et du Vm() reçu.*

2.6 Comparaison entre les alternatives

Les séquences de dépendance et les horloges hiérarchiques sont deux alternatives des horloges vectorielles pour la représentation des dépendances causales dans un environnement de calcul mobile. Les deux alternatives dépendent des MSS_s pour maintenir les horloges, et elles sont adaptées aux fluctuations dans le nombre des sites mobiles dans le système. Les déconnexions sont facilement traitées dans les deux cas.

La quantité d'informations de dépendance qui est envoyée avec chaque message, sur le réseau filaire, dans l'approche séquences de dépendance peut être grande, spécialement s'il y a un grand nombre de *gaps de dépendance* pour l'événement d'émission de message. Dans l'approche horloge hiérarchique, chaque message à besoin d'emporter un vecteur de *n* entiers seulement. Cependant, avec l'approche d'horloge hiérarchique une relation de récurrence doit être évaluée pour déterminer les prédécesseurs causaux d'un événement dans tout le système. Ceci peut conduire à une perte de temps quand les chaînes de dépendance vont être traversées. Avec l'approche séquences de dépendance, l'ensemble des séquences de dépendance indique généralement l'ensemble des prédécesseurs d'un événement.

Pour ces deux alternatives, la conclusion suivante est faite : l'approche séquences de dépendance a des taux de communication élevés et n'exige pas un délai de temps élevé pour déterminer le passé d'un événement. L'approche horloge hiérarchique a des taux de communication baissés, mais la détermination du passé d'un événement prend du temps.

Pour les horloges entrelacées qui sont une alternative des horloges hiérarchiques, quand on considère un système qui offre un ordre total entre les événements d'un même processus, la première différence qu'on peut faire entre elles et les horloges hiérarchiques est que ces dernières gardent sous forme de vecteur bits l'ensemble des prédécesseurs causaux dans le même *MH*, tandis que les horloges entrelacées gardent seulement le dernier prédécesseur causal local et atteignent de manière récursive les autres prédécesseurs locaux. Cette simplification est possible puisque l'ordre partiel entre les événements vus par une *MSS* est un regroupement des ordres totaux, des événements des MH_s, plutôt qu'un ordre total non restreint. Le calcul de l'ensemble des prédécesseurs locaux, qui peuvent être vus si les horloges hiérarchiques sont utilisées, peut être déterminé à partir de l'état local de la *MSS*. Une utilisation élevée de la récursivité dans le calcul local, avec des pénalités de temps associées pour certaines traces, est le coût pour tracer une représentation plus compacte. Malgré ça, si le calcul local est fait en premier, alors les deux approches peuvent entraîner le même taux de communication entre les MSS_s, qui est la portion la plus coûteuse, mais un espace considérable va être gagné en utilisant des horloges entrelacées.

2.7 Conclusion

La caractérisation des dépendances causales entres les activités distribuées joue un rôle important dans l'analyse des systèmes distribués. Plusieurs mécanismes d'estampillage ont été proposés pour tracer cette causalité dans les systèmes de calcul distribués. Le mécanisme le plus utilisé était l'estampillage par horloges vectorielles car il offre la possibilité de tracer l'ordre partiel entre les événements tout en gardant l'information de concurrence entre ces derniers. Cette information est cruciale pour l'analyse de certaines propriétés dans les systèmes distribués, telles que le degré de parallélisme d'une application afin d'avoir une bonne répartition des tâches sur les différents processeurs.

Mais on a vu que l'utilisation des horloges vectorielles pour capturer les dépendances causales dans les systèmes de calcul mobile n'est pas faisable et ceci pour deux raisons. La première est que la taille du vecteur peut être importante quand un nombre de sites mobiles participent dans le calcul distribué. La deuxième est que le nombre de sites mobiles participant peut changer avec le temps, tandis que les horloges vectorielles exigent que le nombre des nœuds reste fixé.

Par conséquent, il est apparu un besoin pour une représentation efficace des dépendances causales dans les systèmes mobiles. Deux alternatives principales ont été proposées, nommées, l'ensemble des séquences de dépendance et les horloges hiérarchiques. Les séquences de dépendance contiennent l'information complète sur tous les prédécesseurs causaux d'un événement, mais peut avoir des taux de communication élevés. Les horloges hiérarchiques ont des taux de communication faibles, mais la détermination de tous les prédécesseurs causaux d'un événement peut entraîner une consommation de temps car l'information se trouve dispersée entre les différentes stations de base.

Chapitre 3

L'ordre causal en environnement mobile

3.1 Introduction

L'exécution asynchrone des processus et le caractère fini et non borné des délais de communication confèrent aux systèmes répartis un comportement non déterministe et rend plus complexe la conception, la vérification et l'analyse des programmes répartis. Et du fait que l'émission et la réception de messages sont le seul moyen permettant aux processus de coopérer, échanger des données, et avoir des connaissances sur l'état des autres composants du système, alors le passage efficace des messages avec les sémantiques appropriées est de grande importance. Pendant que les estampilles causales nous permettent de déterminer l'ordre relatif entre n'importe quelle paire d'événements, elles ne peuvent pas être utilisées pour déterminer l'existence d'événements intermédiaires. Par exemple, si un serveur de fichier recevant des requêtes en provenance des clients a besoin de servir les requêtes dans l'ordre d'émission, les estampilles causales sont insuffisantes. Car il est impossible de déterminer s'il existe des requêtes non encore reçues qui précèdent causalement la requête courante. Pour satisfaire les besoins du serveur, l'ordre causal des messages est requis.

L'idée de "*l'ordre causal des messages*" a été introduite par Joseph et Birman [53] dans le cadre du projet ISIS [54] développé à l'université de Cornell. L'ordre causal des messages garantit que l'ordre de livraison des messages ne viole pas la causalité dans le système. Le processus diffère la livraison d'un message donné jusqu'à l'arrivée de tous ses prédécesseurs causaux. Le concept d'ordre causal est d'un intérêt considérable pour la conception des systèmes distribués et trouve ses applications dans plusieurs domaines tels que les mises à jour des données répliquées, le calcul d'état global, la mémoire distribuée partagée, la téléconférence, les systèmes multimédia, et l'allocation des ressources partagées. La mise en œuvre de l'ordre causal dans un système asynchrone nécessite l'adjonction d'information de contrôle à chaque message. Le processus qui reçoit un message utilise cette information pour déterminer s'il y a des messages non délivrés qui le précèdent causalement et auquel cas sa livraison est différée jusqu'à la livraison de ces derniers. Plusieurs protocoles de mise en œuvre de l'ordre causal existent pour les environnements distribués [53,62,63,64]. Ils se différencient principalement par la taille des informations de contrôle transportées par les messages.

Mais l'apparition des systèmes mobiles avec de nouvelles caractéristiques telles la mobilité et la communication sans fil fait qu'il est nécessaire de réviser les algorithmes déjà conçus pour les systèmes distribués classiques afin de les adapter aux environnements mobiles. La construction d'algorithmes répartis pour les systèmes

répartis en environnement mobile nécessite de reporter les coûts de calcul, de communication et de stockage sur le réseau statique. Les unités mobiles et leurs liens de communication sans fil doivent être sollicités le moins possible. Cependant, l'exécution d'un protocole par les stations de base pour le compte des unités mobiles conduit à une vision différente sur l'ordre des événements entre ces dernières et les stations de base du système, ce qui engendre un délai d'*inhibition* dans la livraison des messages par un protocole d'ordre causal des messages en environnement mobile. Certaines contributions ont été faites pour l'adaptation des protocoles d'ordre causal des messages pour l'environnement mobile mais elles restent modestes. L'objectif principal de ces contributions était de minimiser la taille de l'information de contrôle transportée avec chaque message ainsi que les délais d'inhibition et d'assurer une bonne gestion des migrations des sites mobiles d'une cellule à une autre.

Nous allons présenter dans ce chapitre la définition du concept d'ordre causal des messages, ainsi que les protocoles proposés pour mettre en œuvre ce concept dans l'environnement distribué classique. Et enfin, nous citons les contributions proposées pour l'adaptation de ces protocoles pour les systèmes en environnement mobile.

3.2 Définition (Ordre causal)

Si pour n'importe quels deux messages M et M', $Send(M) \rightarrow Send(M')$ et M et M' ont la même destination, alors l'ordonnancement causal assure que $Delivery(M) \rightarrow Delivery(M')$.

En d'autres termes, les messages dirigés vers la même destination sont délivrés dans un ordre consistant avec leur causalité. La causalité considérée ici est déterminée par la relation "*happened before*" de Lamport [43] mais elle est limitée aux événements d'émission et de réception de messages.

Il est important de distinguer entre l'arrivée d'un message et sa livraison. L'arrivée d'un message signifie que le réseau de communication a placé le message dans le buffer du processus récepteur. La livraison d'un message signifie que le processus a pris le message pour son traitement. Naturellement, pour un message M la relation $Receive(M) \rightarrow Delivery(M)$ est toujours vraie.

D'après la définition de l'ordre causal, si M' est reçu par le processus destinataire avant M, la livraison de M' au processus est différée par le protocole d'ordre causal jusqu'à ce que M soit reçu et délivré. La violation de l'ordre causal peut être illustrée par l'exemple classique présenté par la figure ci-dessous (figure 3.1). Le processus P_1 émet M_1 avant l'émission de M_2. P_2 émet M_3 après que M_2 soit délivré. D'ici, par la relation happened before, $Send(M_1) \rightarrow Send(M_3)$. Donc, M_1 doit être délivré à P_3 avant M_3. Cependant, M_3 est délivré avant M_1, ce qui représente une violation de l'ordre causal des messages.

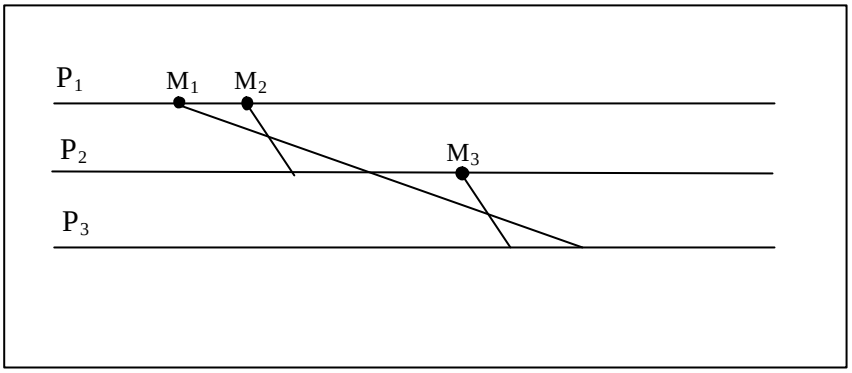


Figure 3.1 – *Un exemple de la violation de l'ordre causal*

3.3 L'utilité de l'ordre causal

Le concept d'ordre causal est d'un intérêt considérable pour la conception des systèmes distribués et trouve ses applications dans plusieurs domaines tels que les mises à jour des données répliquées [55], la collection d'état global [56], la mémoire distribuée partagée [57], la téléconférence [58], les systèmes multimédia [59], et l'allocation des ressources [60].

Nous citons ici l'utilité de la propriété d'ordre causal pour certains domaines:

1. *Gestion des données dupliquées* : Soit une donnée *X* répliquée sur plusieurs sites. Dans le but d'assurer une consistance mutuelle des différentes copies x_1, x_2,…, les mises à jour de ces copies doivent avoir lieu dans le même ordre. En particulier, ceci introduit la nécessité d'une exclusion mutuelle sur les mises à jour faites par deux sites différents. Ceci peut être résolu en utilisant un jeton; à la possession du jeton, un site peut mettre à jour sa copie de *X* et diffuse la mise à jour à tous les autres sites possédant une copie de *X*. L'ordre causal assure que tous les sites recevront toutes les mises à jour dans le même ordre (la mise à jour W_i ne doit jamais être délivrée avant W_j, pour $i < j$), ce qui assure la consistance mutuelle des copies. Le jeton assure l'ordonnancement total des mises à jour sur *X* et l'ordonnancement causal assure que toutes les copies x_i sont mises à jour dans ce même ordre.

2. *Monitoring d'un système distribué* : Une observation consistante d'un système distribué est un autre exemple de l'utilité de l'ordre causal. Soit un nombre de sites contrôlant les événements qui se produisent localement, et supposant que toutes ces informations intéressent un site observateur *OS*. Si les informations de contrôle collectées par les sites sont envoyées à *OS* en respectant l'ordre causal, l'information sera reçue dans un ordre cohérent (i.e. dans un ordre qui ne viole pas l'ordre de précédence entre les événements).

3. *Allocation de ressources partagées* : Ce problème peut être résolu par un allocateur de ressources résidant sur un site, qui reçoit les requêtes d'allocation dans une file *FIFO.* Avec ce schéma, les requêtes sont satisfaites dans l'ordre de réception, ceci peut être cependant considéré insuffisant dans certains cas: il peut être souhaitable de satisfaire les requêtes non pas dans leur ordre de réception, mais dans leur ordre d'émission, i.e. si deux requêtes R_1 et R_2 sont telles que R_1? R_2, alors R_1 doit être satisfaite avant R_2.

4. *Téléconférence*: une application de téléconférence est une application dans laquelle deux où plusieurs personnes distantes peuvent mener une conversation. Nous assumons que chaque participant dans la conversation peut émettre des messages et répondre à d'autres à n'importe quel moment, et qu'un message original et ses réponses peuvent être envoyés à un nombre de participants. Supposons maintenant que l'utilisateur *A* envoie un message et après la réception de ce message par l'utilisateur *B*, il émet une réponse. La propriété de livraison causalement ordonnée des messages assure qu'aucun participant ne peut recevoir la réponse à partir de *B* avant la réception du message original à partir de *A*. Sans la propriété d'ordre causal un participant peut recevoir un commentaire sur certaines idées avant la réception du message exprimant l'idée originale. Dans ce cas, l'idée en retard peut être mal interprétée.

3.4 Les protocoles d'ordre causal dans l'environnement distribué

La mise en œuvre de l'ordre causal dans un système asynchrone nécessite l'adjonction d'information de contrôle à chaque message. Le processus qui reçoit un message utilise cette information pour déterminer s'il y a des messages non délivrés qui le précèdent causalement et auquel cas sa livraison est différée jusqu'à la livraison de ces derniers.

Plusieurs algorithmes ont été développés pour fournir un ordonnancement causal des messages:

Le premier protocole d'ordre causal a été proposé par Birman et Joseph en 1987 [53] dans le contexte du système Isis [54]. Ce protocole utilise une technique d'histoire causale pour l'ordonnancement causal dans le système Isis. L'idée est la suivante: chaque message m emporte avec lui tous les autres messages qui le précèdent causalement [53]. Si un processus reçoit un buffer de messages contenant plusieurs messages envoyés à la destination courante, alors il les traite dans l'ordre de leur apparition dans le buffer.

A cause de l'information redondante, ce schéma est résistant aux défaillances de processeurs. Cependant, l'inconvénient majeur de cet algorithme consiste dans l'évolution non bornée de l'information de contrôle. Des optimisations ont été appliquées afin de réduire l'histoire causale attachée aux messages. Les versions les plus récentes de Isis sont basées sur le temps vectoriel, et constituent une amélioration considérable due à la réduction de la surcharge [61]. L'algorithme d'ordre causal de Peterson, Buchlolz et Schlichting [62] est similaire à celui de ISIS sauf qu'il emporte, dans l'information de contrôle, les identificateurs des messages au lieu des messages. De plus, l'information de contrôle inutile n'est pas envoyée si le site émetteur l'a déjà envoyé.

Kearns et al. [60] ont présenté un protocole pour assurer la livraison point à point d'ordre causal des messages dans un système avec un seul serveur, vers lequel les messages émis à partir de tous les autres processus (clients) sont délivrés dans l'ordre causal. Le protocole utilise des vecteurs compteurs de messages qui sont localisés sur chaque processus et qui sont attachés à chaque message. Un client j recevant un message avec un vecteur attaché V_k à partir d'un autre client k met à jour son vecteur $V_j = \max(V_j, V_k)$. Avant qu'un client i n'émette un message au serveur, il

incrémente sa composante $V_i[i]$ dans son vecteur V_i et attache le vecteur au message. Quand le serveur reçoit un message m à partir d'un processus *i*, il compare d'abord son vecteur V_s avec V_m, le vecteur accompagnant le message m. Si $V_s[j] < V_m[j]$ pour un certain $j \neq i$, alors il existe un message non encore reçu à partir d'un autre client, et le message m est différé jusqu'à l'arrivée du message. Sinon le serveur consomme le message après la mise à jour de son vecteur $V_s = \max(V_s, V_m)$. Notons que $V_i[j]$ représente la connaissance du processus i sur le nombre de messages émis par le processus j au serveur.

Une autre implémentation de l'ordre causal est donnée par Schiper et al. [63]. Cette implémentation n'est qu'une généralisation de la solution proposée par Kearn où tous les processus sont considérés comme serveurs. Dans cette implémentation, une information de contrôle est rajoutée aux messages. Cette information de contrôle permet au site destinataire d'un message M de connaître s'il y a des messages qui doivent être délivrés avant M, dans le but de respecter l'ordre causal: si c'est le cas, M n'est pas immédiatement délivré. L'information de contrôle est composée d'un nombre borné de paires (destination, temps vectoriel).

Raynal, Schiper, et Toueg [64] proposent une variante de la méthode précédente qui n'enregistre que les événements de communication. Elle a les mêmes propriétés de la méthode précédente, mais elle est marginalement plus simple à implémenter. L'algorithme proposé, dit l'algorithme RST, est basé sur le compte de messages et assume que les canaux soient fiables. L'algorithme RST attache une matrice N x N d'entiers à chaque message, où N est le nombre des sites dans le système.

L'algorithme RST pour l'ordonnancement causal maintient deux tableaux, $DELIV_i[N]$ et $SENT_i[N, N]$, pour chaque site P_i. $DELIV_i[j]$ dénote le nombre total des messages reçus par P_i à partir de P_j. $SENT_i[k, j]$ indique la connaissance de P_i sur le nombre de messages que P_k a envoyé à P_j. Les étapes suivantes sont exécutées par P_i pour assurer l'ordre causal: quand P_i envoie un message m à P_j, P_i attache sa valeur courante de $SENT_i$ au message m, i.e. P_i envoie $(m, SENT_i)$ à P_j. P_i incrémente alors $SENT_i[i,j]$ par 1. en recevant (m, ST) à partir de P_j, l'algorithme d'ordre causal à P_i vérifie d'abord si $DELIV_i[k] \geq ST[k,i]$ pour tout k. si cette condition est vérifiée, le message m est délivré à l'application, $DELIV_i[j]$ est incrémenté par 1, $SENT_i[j,i]$ est initialisée à $ST[j,i]$ +1, et finalement $SENT_i[j,k]$ est initialisée à $max(ST[j,k], SENT_i[j,k])$ pour tout j, k. sinon, m est mis en attente jusqu'à ce que $DELIV_i[k] \geq ST[k,i]$ pour tout k.

Ces protocoles se différencient principalement par la taille des informations de contrôle transportées par les messages. Dans le protocole [53], la taille de l'information de contrôle est non bornée. Le protocole [62] est similaire au précédent, toutefois l'information de contrôle transportée par les messages est réduite à un ensemble d'identificateurs de messages. Dans le protocole de [63], chaque message transporte N - 1 couples (destination, vecteur de temps). Dans le protocole de [64], chaque processus maintient une matrice de taille N x N qui est transportée par chaque message émis.

3.5 Les protocoles d'ordre causal dans l'environnement mobile

L'utilité de l'ordre causal dans diverses applications a connu une activité de recherche conséquente dans les systèmes en environnement distribué. Cependant, les

contributions dans ce domaine pour des systèmes en environnement mobile, restent très modestes. L'ordre causal est un concept approprié pour des applications qui nécessitent l'interaction entre plusieurs usagers depuis des localités différentes. Parmi les applications majeures des systèmes répartis en environnement mobile où l'ordre causal est essentiel, nous citons spécialement les applications coopératives, les systèmes de téléconférence, etc.

Les protocoles déjà proposés pour l'environnement distribué peuvent être exécutés par chaque site mobile avec toutes les structures de données utiles stockées sur les sites mobiles eux-mêmes. Cependant, en considérant les ressources limitées et la bande passante des liens sans fils disponible aux sites mobiles, il est inapproprié d'appliquer ces protocoles directement sur les systèmes mobiles. Les facteurs suivants doivent être pris en considération lors de la conception de protocoles pour les systèmes mobiles : le taux de calcul sur les sites mobiles, et la taille des messages transmis sur le medium sans fil doivent être minimaux. De plus, le protocole doit être évolutif (adaptable), et être capable de gérer l'effet des connexions et déconnexions des sites mobiles.

L'adaptation d'un protocole de mise en œuvre de l'ordre dans un système réparti avec des sites mobiles soulève deux problèmes majeurs [32]: *l'impact de la mobilité* qui peut induire une violation de l'ordre causal à cause de la migration des sites mobiles entre les différentes cellules du système. Et *l'inhibition de la livraison des messages* : La mise en œuvre d'un algorithme réparti dans un environnement mobile conduit intuitivement à associer un processus à chaque unité mobile. Malheureusement cette approche est impraticable à cause des restrictions en énergie, puissance CPU et espace disque des unités mobiles ainsi que la bande passante limitée des liaisons de communication sans fil. C'est ainsi que les coûts de calcul et de communication induits par l'exécution d'un algorithme réparti doivent être assurés par les stations fixes du système. Malheureusement, l'exécution d'un algorithme sur les stations de base peut introduire un délai d'inhibition de la livraison des messages aux unités mobiles. Une station de base ne peut pas avoir une vue exacte des relations de dépendance causale entre les messages qu'elle reçoit depuis les unités mobiles localisées dans sa cellule. Dans la figure 3.2, les événements d'émissions de m_1 et m_2 chez S_1 sont tels que $Send(m_1) \rightarrow Send(m_2)$ et donc m_2 aura à transporter l'identité de m_1 comme une de ses contraintes de livraison causale. A cause de la latence du réseau statique, m_2 peut être reçu par la station de base S_2 avant m_1. La livraison de m_2 sera alors différée jusqu'à la réception de m_1 alors que $Send(m_1)$ et $Send(m_2)$ sont deux événements indépendants selon les unités mobiles source.

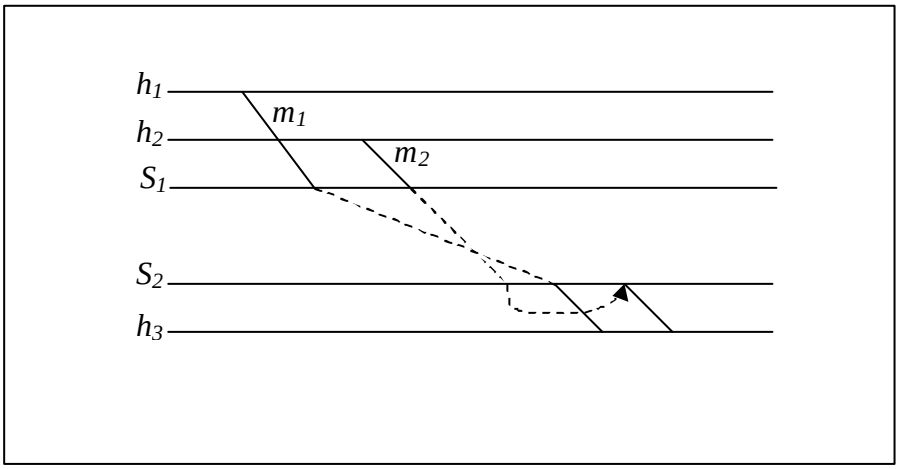


Figure 3.2 – *Effet d'inhibition de la livraison des messages*

Alagar et Venkatesan [65] ont proposé trois extensions de l'algorithme d'ordre causal dans [64] pour les systèmes mobiles. Leur premier algorithme (*AV1*) gère les contraintes ressources des sites mobiles. Il sauvegarde les structures de données des MH_s utiles pour l'ordre causal dans les MSS_s, et l'algorithme est exécuté par les *MSS* pour le compte des MH_s. Cependant, le coût de communication est proportionnel au carré du nombre des MH_s. L'avantage de cet algorithme est que la procédure pour le traitement des migrations des sites est simple. Le deuxième algorithme (*AV2*) élimine les inconvénients du premier algorithme. La propriété d'ordre causal est explicitement maintenue entre les MSS_s. La taille de l'information de contrôle attachée aux messages est proportionnelle au carré du nombre des MSS_s. Puisque cette taille ne varie pas avec le nombre des MH_s, l'algorithme est évolutif et les connexions et déconnexions des sites mobiles ne posent pas de problème mais il peut y avoir certaine inhibition dans la délivrance des messages aux MH_s. Ce problème est dû essentiellement à la mise en œuvre de l'ordre causal entre les stations fixes du système. La livraison d'un message peut être retardée car l'ordre causal peut être violé du point de vue de la station de base alors qu'il est considéré comme livrable selon l'ordre des messages tel que perçu par les unités mobiles locales. Pour faire face aux problèmes associés à la mobilité, cet algorithme comporte un module particulier pour gérer les migrations de sites. Pour réduire ces délais sans les supprimer totalement, Alagar et Venkatesan ont proposé un troisième algorithme (*AV3*) qui est un algorithme hybride des deux autres algorithmes. Chaque *MSS* est partitionnée en k MSS_s logiques. Ainsi, les messages émis ou reçus par les unités mobiles appartenant à des stations logiques différentes ne peuvent pas s'inhiber mutuellement. Mais, plus k est grand, plus la possibilité des délais inutiles diminue mais la surcharge des messages et le coût de traitement des migrations de sites augmentent.

Yen, Huang, et Hwang (*YHH*) [66] ont proposé un algorithme basé sur [64]. Chaque station de base S_i maintient une matrice MSS_SENT_i à N_s x N_h entrées où la $j^{éme}$ rangée dénote la connaissance de S_i sur le nombre de message émis par S_j vers chaque site mobile h_k. Chaque message m transmis par h_i est reçu d'abord par sa station de base S_i. S_i attache sa matrice MSS_SENT_i au message m et dirige m vers sa destination h_j. Ensuite, S_i incrémente par 1 l'entrée $MSS\_SENT_i[i,j]$. A chaque site mobile h_i dans le système est associé un vecteur à n_s entrées, MH_DELIV_i dont la $j^{éme}$ entrée de MH_DELIV_i dénote le nombre de messages émis par la station de base S_j qui sont délivrés à h_i. Ce vecteur est stocké sur la station de base de h_i. Quand le message m destiné à h_j est reçu par la station de base de h_j, soit S_l, cette dernière compare le contenu de MH_DELIV_j avec le contenu de la matrice attachée au message m, $m.S$, pour décider si m peut être délivré ou non. Un message m est délivré si $MH\_DELIV_j[k] \geq m.S[k,j]$ pour tout $k \in \{1,...,N_s\}$, sinon le message m est mis dans la file d'attente $PEND_j$ jusqu'à ce que la condition de livraison pour ce message soit satisfaite. Dans le cas où le message m est délivrable, S_l l'insère dans la file $WAIT\_ACK_j$ qui sert à garder le message délivré au site mobile h_j jusqu'à la réception d'un accusé de réception de la part du site destinataire h_j. A l'arrivée de cet accusé de réception, S_l exécute les actions suivantes: $MSS\_SENT_l[i,j]++$; $MSS\_SENT_l = \max(MSS\_SENT_l, m.S)$ et finalement supprime m de $WAIT\_ACK_j$.

La taille de l'information de contrôle attachée aux messages dans cet algorithme est entre celle de *AV1* et *AV2*. En particulier, chaque *MSS* maintient une matrice de taille N_s x N_h. Le délai d'inhibition dans cet algorithme est plus petit que celui de *AV2*. Leur module de *handoff* est aussi plus efficace que celui de *AV2*. L'inconvénient de cet algorithme est qu'il ne satisfait pas la propriété de vivacité :

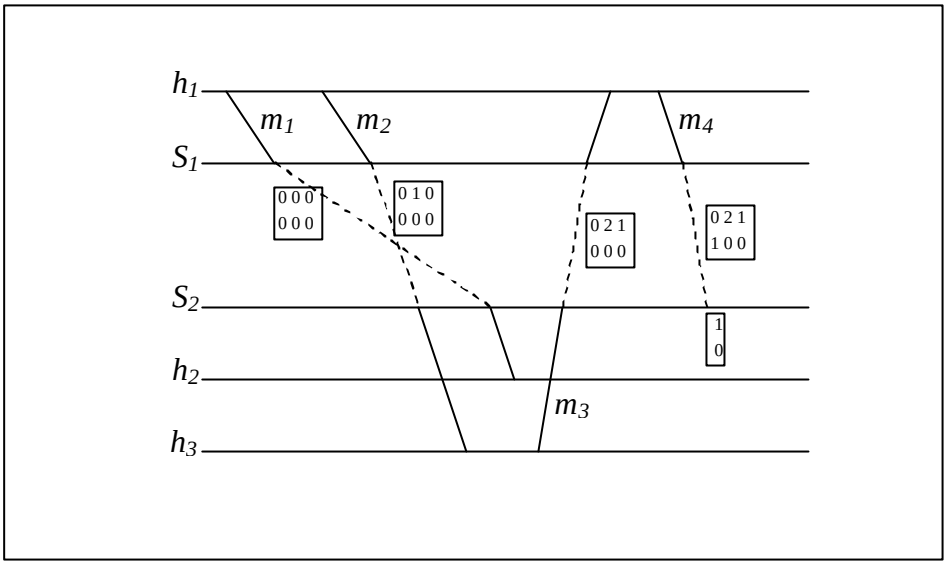


Figure 3.3 – *Problème de vivacité dans le protocole YHH*

Dans la figure 3.3, nous présentons un scénario où le protocole *YHH* ne satisfait pas la propriété de vivacité. Selon le protocole *YHH*, la livraison du message m_4 va être différée car $m_4.M[1,2] > MH\_DELIV_2[1]$. Et puisque du moment où m_4 arrive à S_2, aucun autre message n'est transmis, la livraison de m_4 est différée indéfiniment.

Prakash, Raynal et Singhal (*PSR*) [67] ont présenté un algorithme où la taille de l'information de contrôle est relativement réduite; cependant, dans le cas le plus défavorable elle peut dépasser $O(N_h^2)$.

Un autre algorithme a été proposé par Skawratananond, Mittal, et Garg (*SMG*) [68]. Le protocole maintient pour chaque site mobile h_l une matrice $M_l[N_s, N_s]$, où $M_l[i, j]$ dénote la connaissance de h_l sur le nombre de message transmis par S_i à S_j. De plus, chaque station de base S_i, maintient deux tableaux, $lastsent_i[N_s]$ et $lastrcvd_i[N_s]$ où $lastsent_i[j]$ dénote le nombre de messages émis par S_i à S_j et $lastrcvd_i[j]$ dénote le nombre de messages émis par S_j qui sont délivrés à S_i. Toutes ces structures de données sont stockées sur les stations de base afin de réduire le coût de stockage sur les sites mobiles. Pour émettre un message *m* à un autre site mobile h_d localisé dans la cellule de la station de base $\underline{S_j}$, h_s envoie d'abord le message à sa station de base S_i. S_i incrémente alors $lastsent_i[j]$ par 1 et émet $(m, M_s, lastsent_i[j])$ à S_j. Ensuite, S_i initialise $M_s[i,j]$ par $lastsent_i[j]$. En recevant le message $(m, M, seqno)$ par S_j, cette dernière vérifie si *m* est délivrable. Le message *m* est délivrable si $(lastrcvd_j[k] \geq M[k, j]$ pour tout k) et $(\nexists(m', M', seqno')$ envoyé par S_k à h_d et non encore délivré tel que $seqno' \leq M[k, j])$. Si la condition est satisfaite le message est délivré à h_d, sinon le message est inséré dans $rcvQ_j$ jusqu'à ce qu'il devient délivrable. Similairement à *YHH*, ce protocole prévoit une file $ackQ_d$ qui sert à garder le message délivré jusqu'à la réception d'un acquittement affirmant sa réception par le site destinataire h_d. Dans ce cas, $M_d[i, j] = \max(M_d[i, j], m.seqno)$ et $M_d[k,h] = \max(M_d[k,h], m.M)$ pour tout k, h. La surcharge de l'information de contrôle dans cet algorithme est de l'ordre de $N_s^2 + N_h$. Son module de *handoff* est plus efficace que celui de *AV2* et *AV3* puisqu'il ne nécessite pas que les messages échangés entre les MSS_s soient causalement ordonnés.

3.6 Conclusion

Des travaux récents ont été consacrés au problème de structuration des algorithmes distribués pour les environnements mobiles. Les algorithmes distribués doivent être conçus de telle manière que leurs coûts de calcul et de communication soient essentiellement supportés par le réseau statique plutôt que par les sites mobiles eux-mêmes. Toutefois, cette approche introduit dans le comportement des protocoles certains effets indésirables tels qu'un délai d'inhibition de la livraison des messages.

Parmi les problèmes qui ont connu des contributions reportées vers cette direction de recherche, nous trouvons le problème d'ordre causal des messages. Certaines propositions ont été faites pour mettre en œuvre le concept d'ordre causal en environnement mobile, mais elles restent modestes. L'inconvénient majeur de ces protocoles est qu'ils peuvent entraîner des délais d'inhibition de la livraison de messages même si ces derniers ne violent pas l'ordre causal, cependant le taux de ces délais varie d'un protocole à l'autre ainsi que la taille de l'information de contrôle. L'approche suivie par les auteurs pour restructurer les protocoles d'ordre causal et les adapter à un environnement mobile consiste à exécuter ces protocoles par les stations de base au compte des sites mobiles résidant dans leurs cellules respectives.

Chapitre 4

Un protocole d'ordonnancement causal en environnement Mobile (*Mobi_Causal*)

4.1 Introduction

Dans ce chapitre nous présentons un nouveau protocole d'ordonnancement causal qui est basé sur l'utilisation des nouveaux mécanismes d'estampillage proposés pour les environnements mobiles : les horloges hiérarchiques [50] et les séquences de dépendances [50]. Donc, l'objectif de notre contribution est d'étudier l'applicabilité de ces mécanismes pour l'implémentation d'un protocole qui résout le problème d'ordre causal en environnement mobile. Notre contribution se résume comme suit : (1) Notre condition de livraison est structurée de manière hiérarchique offrant ainsi un test par rapport à une dépendance interne avant de passer à tester les dépendances externes, c'est-à-dire si la livraison d'un message dépendant d'un notre message émis par le même site mobile alors il suffit de tester la livraison de ce message au lieu de tester toutes les dépendances car ceci sera fait à l'arrivée du message duquel il dépend. Ce qui permet d'éliminer les tests inutiles. (2) Avec notre protocole, nous sommes capables d'éliminer les délais d'inhibition tout en maintenant une taille minimale de l'information de contrôle attachée aux messages. (3) Finalement, nous prouvons la correction de notre condition de livraison implémentée par le module statique.

Le reste de ce chapitre est organisé comme suit : nous citons d'abord les structures de données utilisées dans notre protocole. Par la suite, nous présentons le module statique du protocole ainsi que des exemples illustratifs. Le paragraphe suivant sera consacré à la preuve de correction de la solution ; nous prouvons la sûreté et la vivacité de notre solution. Finalement, nous présentons le module *handoff* et nous terminons par une conclusion.

4.2 Protocole Mobi_Causal

4.2.1 Structures de données utilisées

Pour chaque h_i

$h_i \longrightarrow id_{mess_E}, id_{mess_D}$
f^i
$LastSend_{h_i}, LastRcv_{h_i}$
$depend_{h_i}$
f

Pour chaque S_i

$S_i \longrightarrow IdSend_{S_i}$
$AtFile_{S_i}$
$id_{LastSend}$

4.2.2 Interprétation des différentes structures de données

- id_{mess_E} : Compteur des messages émis ou reçus par un site mobile h_i. Il est incrémenté chaque fois qu'un message est émis ou reçu.
- id_{mess_D} : C'est l'identificateur du dernier message reçu par un site mobile h_i. Il prend la valeur de id_{mess_E} à la livraison du message.
- $LastSend_{h_i}$: Un ensemble de triplets (h_j, id, f^i) qui garde l'identificateur du dernier message émis par h_i aux autres h_j, ainsi que la valeur de l'horloge locale calculée f^i.
- f^i : C'est un vecteur de bit de taille variable. Sa construction est basée sur le principe des horloges hiérarchiques locales.

Construction de f^i

a. Idée

Nous distinguons trois types de messages : les messages émis par un site mobile et par conséquent par sa station de base, notés E, les messages reçus par une station de base mais pas forcement délivrés au site mobile destinataire, notés R, et les messages délivrés d'une station de base à un site mobile après avoir vérifié la condition de livraison, notés D.

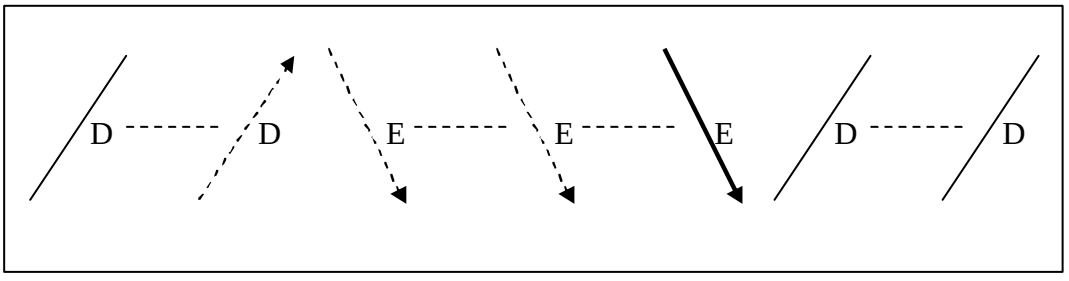


Figure 4.1 – *Types de messages au niveau d'un site mobile*

Les réceptions par un site mobile des messages délivrés D sont indépendantes, car un message ne peut être délivré que si la condition de livraison est vérifiée au niveau de la station de base. Pour un message d'émission E, il dépend du dernier message D reçu et des messages E qui sont émis avant lui au même site mobile, et après le dernier message reçu D.

Comme il est montré dans la figure ci-dessus (Figure 4.1), si le site mobile décide d'émettre le message E en gras, alors l'information de dépendance causale de ce message est construite à partir des messages E et D représentés sur la figure par des pointillés.

Nous proposons la structure suivante à f^i : $B_D C_{Int} B_E$

Tel que : B_D : Une chaîne de un bit qui est égal à 1 si $id_{mess_D} \neq 0$ sinon cette chaîne est vide.

B_E : Une chaîne de un bit qui est égal à 1. Ce bit marque le message en cours d'émission.

C_{Int}: Une chaîne de bits de longueur $id_{mess_E} - id_{mess_D} - 1$

La fonction *Construct*(f^i)

/* La longueur de f^i est $id_{mess_E} - id_{mess_D} + 1$ */

Begin

1. ***If*** $id_{mess_D} \neq 0$ ***Then*** $B_D = 1$
   Else ∄
   EndIf
2. $B_E = 1$
   /* *Construction de* C_{Int}
3. ***For*** $k = id_{mess_D} + 1$ ***to*** $(id_{mess_E} - 1)$
   Do
4. ***If*** $\exists (h_j, k, \_) \in LastSend_{h_i}$
   Then
   /* $C_{Int}(val_1, val_2)$ *signifie la chaîne* C_{Int} *avec* val_1 *comme position*
   /* *de départ et* val_2 *comme longueur* */
5. ***If*** $id_{mess_D} \neq 0$ ***Then*** $C_{Int}(1, k - id_{mess_D}) = f^i_{(h_j,k)}(2, k - id_{mess_D})$
   Else $C_{Int}(1, k - id_{mess_D}) = f^i_{(h_j,k)}(1, k - id_{mess_D})$;
   EndIf
6. $C_{Int}(k+1, id_{mess_E} - k) = 0$
   Exit;
   EndIf
   EndDo
7. $f^i = concat(B_D, C_{Int}, B_E)$

End

b. Explication de *Construct* (f^i)

Le but de l'horloge locale f^i est de capturer la relation de causalité interne au site mobile, c'est-à-dire, calculer les dépendances de causalité par rapport au site mobile non pas par rapport à sa station de base et ceci afin de réduire le taux d'inhibition. Donc, le principe de cet algorithme est d'être capable de tracer les dépendances entres les deux sites mobiles qui échangent le message et les autres station de base. La figure ci-dessous (Figure 4.2) explique ce principe :

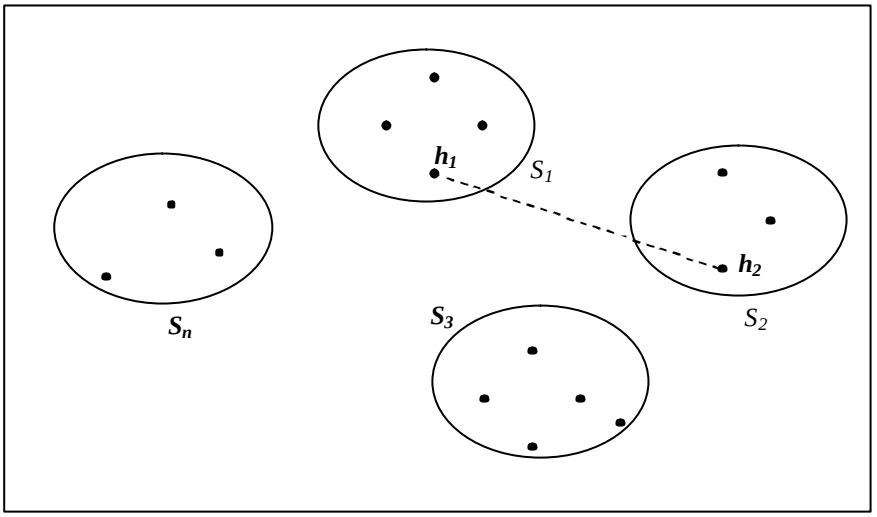


Figure 4.2 – *La causalité est tracée entre h_1, h_2 et S_3,…,S_n*

Si h_1 et h_2 désirent échanger un message, alors les dépendances causales doivent être tracées entre h_1, h_2 et les autres stations de bases S_3,…,S_n.

L'horloge f^i est un vecteur de bits de taille variable. La taille maximale de f^i est égale au nombre de messages émis entre deux réceptions données. Nous supposons que les sites mobiles ont un taux de coopération élevé pour empêcher l'évolution infinie de la taille de f^i, c'est-à-dire nous supposons l'absence de sites mobiles qui ne jouent que le rôle de récepteur. Sa structure est de la forme $B_D C_{Int} B_E$. La construction de f^i est faite par les instructions suivantes de la fonction *Construct* (f^i) :

1. Cette instruction teste si le site mobile a déjà reçu un message ou pas. Dans le cas où le site mobile a reçu déjà un message ce bit, B_D, est mis à 1, sinon il n'existe pas.

2. Puisque le principe de f^i est de tracer les dépendances entre les événements et puisque un événement dépend de lui-même, alors le bit B_E est mis à 1. pour dire que le message à émettre dépend de lui-même.

3. 4. 5. 6. Ces instructions servent à construire la chaîne C_{Int}. C_{Int} est une chaîne de bits de taille égale au nombre de messages émis après le dernier message reçu par le site mobile et avant le message à émettre. Un bit de C_{Int} est égal à 1 si le message correspondant à ce bit est déjà émis au site mobile destinataire sinon le bit est mis à 0. Puisque chaque site mobile possède un ensemble de triplets $LastSend_{h_i} = \{(h_j, id, f^i), ...\}$ pour garder l'information sur le dernier message émis à un site mobile h_j.

Donc, construire C_{Int} revient à chercher un triplet dans $LastSend_{h_i}$ possédant comme identificateur de site, l'identificateur du site destinataire du message (*Instr.* 5.). Ensuite compléter le reste de la chaîne par des 0 (*Instr.* 6.).

7. Cette instruction fait la concaténation de B_D, C_{Int}, B_E pour construire f^i.

c. Exemple

Soient les émissions suivantes (Figure 4.3) au niveau du site mobile h_1 qui communique avec deux autres sites mobiles h_2, h_3.

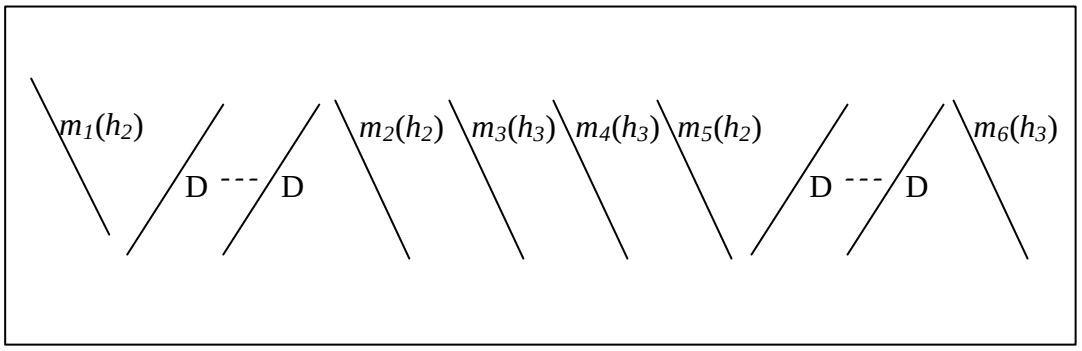


Figure 4.3 – *Exemple d'émissions et de réceptions au niveau d'un site h_1*

1. A l'émission de m_1 vers h_2

$$\left.\begin{array}{c} Puisque\ id_{mess_D} = 0 \Rightarrow B_D\ n'existe\ pas \\ B_E = 1 \\ C_{Int}\ est\ vide \end{array}\right\} \Rightarrow f^i = 1$$

2. A l'émission de m_2 vers h_2

$$\left.\begin{array}{c} Puisque\ id_{mess_D} \neq 0 \Rightarrow B_D = 1 \\ B_E = 1 \\ C_{Int}\ est\ vide \end{array}\right\} \Rightarrow f^i = 11$$

3. A l'émission de m_3 vers h_3

$$\left.\begin{array}{c} Puisque\ id_{mess_D} \neq 0 \Rightarrow B_D = 1 \\ B_E = 1 \\ C_{Int} = 0 \end{array}\right\} \Rightarrow f^i = 101$$

4. A l'émission de m_4 vers h_3

$$\left.\begin{array}{c} Puisque\ id_{mess_D} \neq 0 \Rightarrow B_D = 1 \\ B_E = 1 \\ C_{Int} = 01 \end{array}\right\} \Rightarrow f^i = 1011$$

5. A l'émission de m_5 vers h_2

$$\left.\begin{array}{c} Puisque\ id_{mess_D} \neq 0 \Rightarrow B_D = 1 \\ B_E = 1 \\ C_{Int} = 100 \end{array}\right\} \Rightarrow f^i = 11001$$

6. A l'émission de m_6 vers h_3

$$\left.\begin{array}{c} Puisque\ id_{mess_D} \neq 0 \Rightarrow B_D = 1 \\ B_E = 1 \\ C_{Int}\ est\ vide \end{array}\right\} \Rightarrow f^i = 11$$

- $LastRcv_{h_i}$: Un ensemble de triplets (h_j, S_j, SD_j) où $SD_j = \{id_1, id_2, id_3, id_4, ...\}$ qui garde les identificateurs des messages reçu par h_i en provenance de h_j. La troisième entrée de chaque triplet correspond à une séquence de dépendance. Donc, sa construction est faite de la même manière que dans le mécanisme des séquences de dépendance [50] expliqué dans le 2^{em} chapitre.
- $depend_{h_i}$: Un booléen dont sa valeur dépend de f^i
- f : Un vecteur de couples de taille N_S (*où N_S est le nombre des MSS_s*), tel que une entrée $f_m[k] = \{(h_j, id), ...\}$ dans le vecteur correspond à l'ensemble des identificateurs des messages prédécesseurs de *m* sur la station de base *k* dont la livraison n'est pas encore confirmée. nous gardons l'identificateur et à quel site mobile ce message a été envoyé.
- $IdSend_{S_i}$: Compteur des messages émis par une station de base S_i.
- $id_{LastSend}$: C'est l'identificateur du dernier message déjà émis par un site mobile h_i au site mobile h_j avant l'émission en cours.
- $AtFile_{S_i}$: File d'attente des messages non encore délivrés.

4.2.3 Le module statique

a. La phase d'émission

L'émission d'un message *m* à partir d'un site mobile h_i de la station de base S_i vers un site mobile h_j de la station de base S_j se fait en deux étapes : tout d'abord la transmission du message à partir du site mobile émetteur h_i vers sa station de base S_i, ensuite l'émission du message à partir de S_i vers la station de base réceptrice S_j après lui avoir rattaché l'information de contrôle.

Send (m) *de* $h_i \rightarrow S_i$
/* *Quand* h_i *décide d'émettre un message m*
Begin
Envoyer m à S_i *via le canal sans fil.*
id_{mess_E} ++
End

Emission **(*m*)** *de* $S_i \rightarrow S_j$

Begin

1. $IdSend_{S_i}$ ++
2. $Construct(f^i_{h_i})$
3. ***If*** $f^i_{h_i} = 0^*1$ ***Then*** $depend_{h_i} = false$
   Else $depend_{h_i} = true$
   EndIf
4. ***If*** $\exists(h_j, id, \_) \in LastSend_{h_i}$ ***Then*** $id_{LastSend} = id$
   Else $id_{LastSend} = 0$
   EndIf
5. *Envoyer* $(m, IdSend_{S_i}, depend_{h_i}, id_{LastSend}, f)$ *à* S_j *via le canal filaire.*
6. $f[i] = f[i] \cup (h_j, IdSend_{S_i})$
7. $LastSend_{h_i} = LastSend_{h_i} \cup (h_j, IdSend_{S_i}, f^i)$
8. $Update(LastSend_{h_i})$

End

Update **(*file*)**

Begin

/* *le rôle de cette fonction est d'éliminer les triplets redondants dans une file.*

If $\exists(h_j, Id_1, f^i_1)$ *and* $(h_j, Id_2, f^i_2) \in$ *file tel que* $Id_1 < Id_2$

Then *Supprimer* (h_j, Id_1, f^i_1) *de file*

EndIf

End

- **Explication de *Emission*(*m*)**

Quand la station de base S_i du site mobile émetteur h_i reçoit le message à émettre, elle procède à calculer l'information de contrôle à attacher au message avant son émission à la station de base réceptrice S_j. Le message à émettre doit avoir la forme suivante : $(m, IdSend_{S_i}, depend_{h_i}, id_{LastSend}, f)$

1. La réception du message m par la station S_i provoque l'incrémentation de son compteur $IdSend_{S_i}$, qui sera attaché au message m. $IdSend_{S_i}$ correspond à l'identificateur de m par rapport à la station de base S_i.
2. 3. La valeur de la variable booléenne $depend_{h_i}$ dépend de f^i. Donc, après la construction de l'horloge locale f^i de h_i, on vérifie si f^i est de la forme 0^*1, cela signifie que ce message ne dépend d'aucun autre message, que ce soit interne au site mobile ($C_{Int} = 0^*$) ou externe (B_D n'existe pas). Dans ce cas, on met la valeur de $depend_{h_i}$ à *fasle*, et par conséquent la livraison de ce message ne dépendra d'aucun autre message. Donc, dés son arrivée à la station de base réceptrice ce message doit être délivré immédiatement. Dans le cas contraire, $depend_{h_i}$ aura la

valeur *true*, ce qui signifie que ce message soit il dépend d'un autre message interne au site mobile h_i (c'est à dire dépend d'un message émis par h_i vers h_j avant le message *m*), soit il dépend d'un message externe (B_D = 1, i.e. que h_i a reçu un message avant l'émission de *m*).

4. Le message doit transmettre aussi, dans l'information de contrôle, l'identificateur du dernier message déjà émis par h_i vers h_j s'il existe. Donc, si on trouve un triplet $(h_j, id, \_) \in LastSend_{h_i}$, on affecte à $id_{LastSend}$ la valeur *id*, sinon $id_{LastSend}$ aura la valeur 0 pour dire que c'est le premier message qui va être émis par h_i vers h_j.
5. 6. 7. 8. Après l'émission du message *m*, on mis à jour l'horloge globale f, donc $f[i] = f[i] \cup (h_j, IdSend_{S_i})$, et on ajoute à l'ensemble $LastSend_{h_i}$ le triplet $(h_j, IdSend_{S_i}, f^i)$. Par la suite, on élimine les triplets redondants dans l'ensemble $LastSend_{h_i}$, c'est à dire s'il existe deux triplets (h_j, Id_1, f_1^i) et (h_j, Id_2, f_2^i) tel que $id_1 < id_2$ alors on supprime le triplet (h_j, Id_1, f_1^i).

b. La phase de réception

***Délivrer* (*m*)** à $h_j(S_j)$ $(m, IdSend_m, depend_m, id_{LastSend_m}, f_m)$

Begin

1. ***If*** $depend_m = false$
   Then *Deliver*(*m*)
   Else
2. ***If*** $(id_{LastSend_m} \neq 0 \; and \; \exists (h_i, \_, SD_i) \in LastRcv_{h_j} \; telque \; id_{LastSend_m} \in SD_i)$
   Then *Deliver (m)*
   Else
3. ***If*** $\left(\begin{array}{l} id_{LastSend_m} = 0 \; and \; \forall k = \overline{1,n}, telque \; \exists (idf, h_j) \in f_m[k]: \\ \exists (\_, S_k, SD_k) \in LastRcv_{h_j} \; tel \; que \; idf \in SD_k \end{array} \right)$
   Then *Deliver*(*m*)
   Build(f_{h_j})
   Délivrer tous les messages dans la file d'attente $AtFile_{S_j}$ qui sont en attente de *m*. Ensuite les supprimer de cette file.
   Else *Attente* (*m*) ***/**** *Mettre le message dans la file d'attente*
   EndIf
   EndIf
   EndIf

End

La fonction *Attente* (*m*)
/* *utiliser une file d'attente AtFile*
Begin
$AtFile = AtFile \cup (m, IdSend_m, depend_m, id_{LastSend_m}, f_m)$
End

- **Explication de *Délivrer*(*m*)**

A l'arrivée du message *m* à sa station de base réceptrice, ce message ne sera délivré au site mobile destinataire h_j que si la condition de livraison est vérifiée, c'est-à-dire que tous les messages qui précèdent causalement le message *m* sont reçus par S_j :

1. Si $depend_m = false$, donc ce message ne dépend d'aucun autre message et sa livraison est immédiate.
2. Si $depend_m = true$, on a deux possibilités : soit *m* dépend d'un autre message interne au site mobile émetteur et qui est émis au même site destinataire h_j, dans ce cas $id_{LastSend_m} \neq 0$, alors il suffit de vérifier que ce message a été reçu par h_j. La réception d'un message par un site mobile est exprimée par un triplet (h_i, S_i, Id) dans l'ensemble $LastRcv_{h_j}$. Si c'est le cas, le message *m* est délivré.
3. Si $depend_m = true$, et $id_{LastSend_m} = 0$, alors la dépendance causale est créée non pas par un message interne à h_i mais par la réception d'un message par ce site mobile. Dans ce cas, la livraison de *m* dépend de la livraison de ses prédécesseurs immédiats sur les différentes stations de base. Ces prédécesseurs sont identifiés par l'horloge globale f, qui a pour rôle de garder la trace des prédécesseurs immédiats d'un message donné. Une entrée dans f est un ensemble de couples (*idf, hl*) qui représentent les identificateurs des messages prédécesseurs sur la station de base correspondante à cette entrée et à quel site mobile il ont été envoyés. Donc, on vérifie pour chaque entrée dans f s'il existe un prédécesseur qui est émis à h_j alors on cherche s'il existe un triplet $(\_, S_k, SD_k) \in LastRcv_{h_j}$ *tel que* $idf \in SD_k$. Si cette condition est vérifiée alors on est sûre que tous les prédécesseurs immédiats de *m* sont reçus et par conséquent *m* sera délivré à h_j, ainsi que tous les messages dans la file d'attente $AtFile_{S_j}$ qui sont en attente de *m*. la livraison de ces messages entraîne leur suppression de la file d'attente.

Dans le cas, où aucune des trois conditions précédentes n'est vérifiée, alors on dit que la condition de livraison (la condition de précédence causale) n'est pas vérifiée, et par conséquent le message *m* est mis dans le file d'attente $AtFile_{S_j}$.

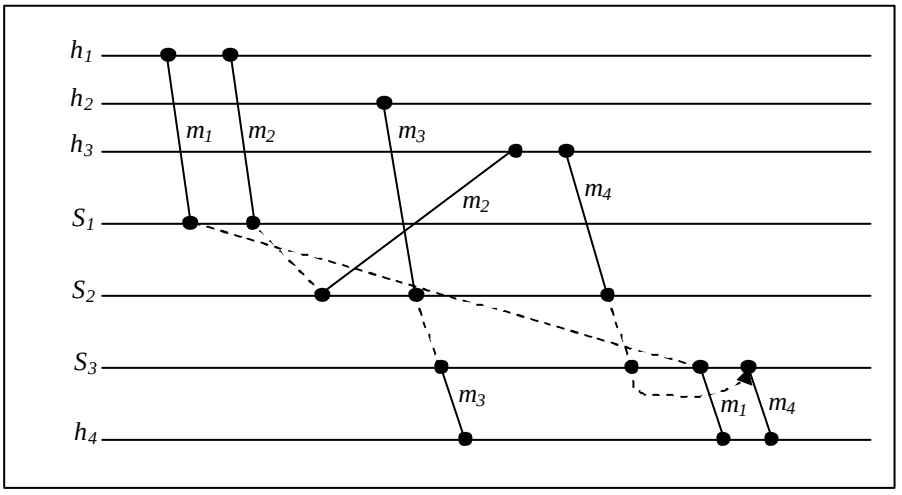
La fonction *Deliver*(*m*) à h_j
Begin
HDeliver(*m*) /* *Délivrer le message m au site mobile* h_j
$IdSend_{S_j}++;$
$f_{h_j} = f_m$
$LastRcv_{h_j} = LastRcv_{h_j} \cup (h_i, S_i, (IdSend_m, IdSend_m))$
End

La fonction *HDeliver(m)*
***** *A l'arrivée du message m au site mobile*
Begin
id_{mess_E} ++
$id_{mess_D} = id_{mess_E}$
End

La fonction *Build*(f_{h_j}) à h_j
***** *Supprime l'information superflue. Tous les messages qui sont délivrés vont être*
***** *supprimés de* f_{h_j} .
Begin
If $\left(\forall h_l \in S_j, \exists (IdSend_m, h_j) \in f_{h_l}[i]\right)$
Then Supprimer $(IdSend_m, h_j)$ de $f_{h_l}[i]$
EndIf
If $\begin{pmatrix} \forall k = \overline{1,n}, f_{h_j}[k] \neq 0, \forall h_l \in S_j \ telque \exists (idf, h_l) \in f_{h_j}[k]: \\ \exists (\_, S_k, SD_k) \in LastRcv_{h_l} tel\ que\ idf \in SD_k \end{pmatrix}$
Then Supprimer (idf, h_l) de $f_{h_j}[k]$
EndIf
End

4.3 Exemples illustratifs

4.3.1 Exemple 1



? Emission de m_1 de $h_1(S_1) \rightarrow h_4(S_3)$

$h_1 : id_{mess_E} = 1,$ $S_1 : IdSend_{S_1} = 1$

$f^{i}_{h_1} = 1, depend_{h_1} = false$

$Envoyer\,(m_1, 1, false, 0, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix})$

$f_{h_1} = \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix},$

$LastSend_{h_1} = \{(h_4, 1, 1)\}$

? Emission de m_2 de $h_1(S_1) \rightarrow h_3(S_2)$

$h_1 : id_{mess_E} = 2,$ $S_1 : IdSend_{S_1} = 2$

$f^{i}_{h_1} = 01, depend_{h_1} = false$

$Envoyer\,(m_2, 2, false, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix})$

$f_{h_1} = \begin{bmatrix} (1, h_4)(2, h_3) \\ 0 \\ 0 \end{bmatrix},$

$LastSend_{h_1} = \{(h_4, 1, 1), (h_3, 2, 01)\}$

? Réception de $(m_2, 2, false, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix})$ par $h_3(S_2)$

Puisque $depend_{h_1} = false$ alors le message sera délivré au site h_3, et les mises à jour suivantes seront faites :

$h_3 : id_{mess_E} = 1, id_{mess_D} = 1,$

$f_{h_3} = \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix},$

$LastRcv_{h_3} = \{(h_1, S_1, \{2,2\})\}$

? Emission de m_3 de $h_2(S_2) \rightarrow h_4(S_3)$

$h_2 : id_{mess_E} = 1,$ $S_2 : IdSend_{S_2} = 1$

$f^{i}_{h_2} = 1, depend_{h_2} = false$

$Envoyer\,(m_3, 1, false, 0, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix})$

$f_{h_2} = \begin{bmatrix} 0 \\ (1, h_4) \\ 0 \end{bmatrix},$

$LastSend_{h_2} = \{(h_4, 1, 1)\}$

? Réception de $(m_3, 1, false, 0, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix})$ par $h_4(S_3)$

Puisque $depend_{h_2} = false$ alors le message sera délivré au site h_4, et les mises à jour suivantes seront faites :

h_4 $id_{mess_E} = 1, id_{mess_D} = 1,$

$f_{h_4} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$

$LastRcv_{h_4} = \{(h_2, S_2, \{1,1\})\}$

? Emission de m_4 de $h_3(S_2) \rightarrow h_4(S_3)$

$h_3 : id_{mess_E} = 2,$ $S_2 : IdSend_{S_2} = 2$

$f^{i}_{h_3} = 11, depend_{h_3} = true$

$Envoyer\,(m_4, 2, true, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix})$

$f_{h_3} = \begin{bmatrix} (1, h_4) \\ (2, , h_4) \\ 0 \end{bmatrix},$

$LastSend_{h_3} = \{(h_4, 2, 11)\}$

? Réception de $(m_4, 2, true, 0, \begin{bmatrix} (1, h_4) \\ 0 \\ 0 \end{bmatrix})$ par $h_4(S_3)$

Dans ce cas $depend_{h_3} = true$ donc le message soit il dépend d'un autre message qui est interne au site mobile h_3 et qui a été envoyé avant m_4 au site mobile h_4, soit il dépend d'autres messages envoyés par les autres sites mobiles. Puisque $id_{LastSend} = 0$ alors le message m_4 est en relation causal avec d'autres

messages envoyés par les autres sites mobiles avant lui au site h_4. En vérifiant f_{h_3} on trouve que m_4 dépend du premier message envoyé par S_1 par conséquent le message m_4 ne sera délivré que si h_4 a reçu le 1er message de S_1. La liste $LastRcv_{h_4} = \{(h_2, S_2, \{1,1\})\}$ ne contient pas l'information sur l'arrivée de ce message ce qui signifie que h_4 ne l'a pas encore reçu. Donc, la livraison de m_4 sera retardée jusqu'à l'arrivée du 1er message de S_1 et m4 sera mis dans la file d'attente *AtFile*.

- Réception de $(m_1, 1, false, 0, \begin{bmatrix}0\\0\\0\end{bmatrix})$ par $h_4(S_3)$

Puisque $depend_{h_1} = false$ alors le message sera délivré au site h_4, et les mises à jour suivantes seront faites :

h_4 $id_{mess_E} = 2, id_{mess_D} = 2,$

$$f_{h_4} = \begin{bmatrix}0\\0\\0\end{bmatrix},$$

$$LastRcv_{h_4} = \{(h_2, S_2, \{1,1\}), (h_1, S_1, \{1,1\})\}$$

En vérifiant la file d'attente *AtFile*, on trouve qu'il existe un message (m_4) qui attend l'arrivée de m_1 donc ce message sera délivré. La livraison de m4 entraîne les mises à jour suivantes :

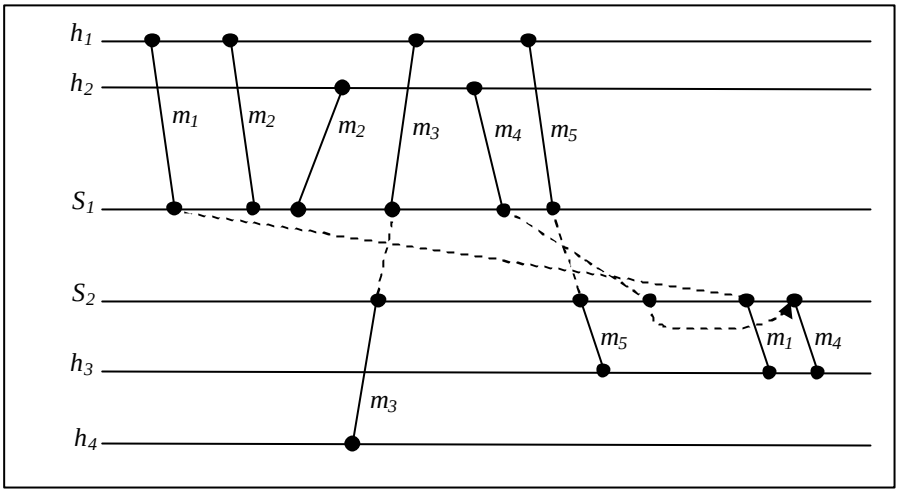
$h_4 : id_{mess_E} = 3, id_{mess_D} = 3,$

$$f_{h_4} = \begin{bmatrix}(1, h_4)\\0\\0\end{bmatrix} \longrightarrow f_{h_4} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$LastRcv_{h_4} = \{(h_2, S_2, \{1,1\}), (h_1, S_1, \{1,1\}), (h_3, S_2, \{2,2\})\}$$

L'information (1, h_4) a été supprimée de f_{h_4} car ce message est délivré et on a plus besoin de transmettre cette information avec les prochains messages.

4.3.2 Exemple 2



? Emission de m_1 de $h_1(S_1) \rightarrow h_3(S_2)$

$h_1 : id_{mess_E} = 1,$ $\qquad S_1 : IdSend_{S_1} = 1$

$f^{i}_{h_1} = 1, depend_{h_1} = false$

$Envoyer\,(m_1, 1, false, 0, \begin{bmatrix} 0 \\ 0 \end{bmatrix})$

$f_{h_1} = \begin{bmatrix} (1, h_3) \\ 0 \end{bmatrix},$

$LastSend_{h_1} = \{(h_3, 1, 1)\}$

? Emission de m_2 de $h_1(S_1) \rightarrow h_2(S_1)$

$h_1 : id_{mess_E} = 2,$ $\qquad S_1 : IdSend_{S_1} = 2$

$f^{i}_{h_1} = 01, depend_{h_1} = false$

$Envoyer\,(m_2, 2, false, 0, \begin{bmatrix} (1, h_3) \\ 0 \end{bmatrix})$

$f_{h_1} = \begin{bmatrix} (1, h_3)(2, h_2) \\ 0 \end{bmatrix},$

$LastSend_{h_1} = \{(h_3, 1, 1), (h_2, 2, 01)\}$

? Réception de $(m_2, 2, false, 0, \begin{bmatrix} (1, h_3) \\ 0 \end{bmatrix})$ par $h_2(S_1)$

Puisque $depend_{h_1} = false$ alors le message sera délivré au site h_2, et les mises à jour suivantes seront faites :

$h_2 : id_{mess_E} = 1, id_{mess_D} = 1,$

$f_{h_2} = \begin{bmatrix} (1, h_3) \\ 0 \end{bmatrix},$

$LastRcv_{h_2} = \{(h_1, S_1, \{2,2\})\}$

? Emission de m_3 de $h_4(S_2) \rightarrow h_1(S_1)$

$h_4 : id_{mess_E} = 1,$ $\qquad S_2 : IdSend_{S_2} = 1$

$f^i_{h_4} = 1, depend_{h_4} = false$

$Envoyer(m_3, 1, false, 0, \begin{bmatrix} 0 \\ 0 \end{bmatrix})$

$f_{h_4} = \begin{bmatrix} 0 \\ (1, h_1) \end{bmatrix},$

$LastSend_{h_4} = \{(h_1, 1, 1)\}$

? Réception de $(m_3, 1, false, 0, \begin{bmatrix} 0 \\ 0 \end{bmatrix})$ par $h_1(S_1)$

Puisque $depend_{h_4} = false$ alors le message sera délivré au site h_1, et les mises à jour suivantes seront faites :

$h_1 : id_{mess_E} = 3, id_{mess_D} = 3,$

$f_{h_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$

$LastRcv_{h_1} = \{(h_4, S_2, \{1,1\})\}$

? Emission de m_4 de $h_2(S_1) \rightarrow h_3(S_2)$

$h_2 : id_{mess_E} = 2,$ $\qquad S_1 : IdSend_{S_1} = 3$

$f^i_{h_2} = 11, depend_{h_2} = true$

$Envoyer(m_4, 2, true, 0, \begin{bmatrix} (1, h_3) \\ 0 \end{bmatrix})$

$f_{h_2} = \begin{bmatrix} (3, h_3) \\ 0 \end{bmatrix},$

$LastSend_{h_2} = \{(h_3, 3, 11)\}$

? Emission de m_5 de $h_1(S_1) \rightarrow h_3(S_2)$

$h_1 : id_{mess_E} = 4,$ $S_1 : IdSend_{S_1} = 4$

$f_{h_1}^{i} = 11, depend_{h_1} = true$

$Envoyer\ (m_5, 4, true, 0, \begin{bmatrix} 0 \\ 0 \end{bmatrix})$

$f_{h_1} = \begin{bmatrix} (4, h_3) \\ 0 \end{bmatrix},$

$LastSend_{h_1} = \{(h_3,1,1), (h_2,2,01), (h_3,4,11)\}$

$= \{(h_2,2,01), (h_3,4,11)\}$

? Réception de $(m_5, 4, true, 0, \begin{bmatrix} 0 \\ 0 \end{bmatrix})$ par $h_3(S_2)$

Puisque la valeur de $depend_{h_1}$ est égale à *true* donc le message soit il dépend d'un message interne au site mobile h_1 qui a été envoyé avant m_5 au site mobile h_3, soit il dépend d'autres messages envoyés par les autres sites mobiles. Puisque $id_{LastSend} = 0$ alors le message m_5 est en relation causal avec d'autres messages envoyés par les autres sites mobiles avant lui au site h_3. En vérifiant f_{h_1} on constate que tous les messages desquels m_5 dépend ont été livrés dans l'ordre causal de leurs émissions, car $f_{h_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, et par conséquent le message m_5 sera délivré à *h3* et les mises à jour suivantes seront faites :

$h_3 : id_{mess_E} = 1, id_{mess_D} = 1,$

$f_{h_3} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$

$LastRcv_{h_3} = \{(h_1, S_1, \{4,4\})\}$

? Réception de $(m_4, 3, true, 0, \begin{bmatrix} (1, h_3) \\ 0 \end{bmatrix})$ par $h_3(S_2)$

De même $depend_{h_2} = true$ donc le message soit il dépend d'un message interne au site mobile h_2 envoyé avant m_4 au site mobile h_3, soit il dépend des messages envoyés par les autres sites mobiles. Puisque $id_{LastSend} = 0$ alors le message m_4 est en relation causal avec des messages envoyés par les autres sites mobiles avant lui au site h_3. En vérifiant f_{h_2} on trouve que m_4 dépend du premier message envoyé par S_1 par conséquent le message m_4 ne sera délivré que si h_3 a reçu le 1er message de S_1. La liste $LastRcv_{h_3} = \{(h_1, S_1, \{4,4\})\}$ ne contient pas l'information sur l'arrivée de ce message ce qui signifie que h_3 ne l'a pas encore reçu. Donc, la livraison de m_4 sera retardée jusqu'à l'arrivée du 1er message de S_1 et m_4 sera mis dans la file d'attente *AtFile*.

? Réception de $(m_1, 1, false, 0, \begin{bmatrix} 0 \\ 0 \end{bmatrix})$ par $h_3(S_2)$

Puisque $depend_{h_1} = false$ alors le message sera délivré au site h_3, et les mises à jour suivantes seront faites :

$$h_3 : id_{mess_E} = 2, id_{mess_D} = 2,$$

$$f_{h_3} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$LastRcv_{h_3} = \{(h_1, S_1, \{1,1,4,4\})\}$$

En vérifiant la file d'attente *AtFile*, on trouve qu'il existe un message (m_4) qui attend l'arrivée de m_1 donc ce message sera délivré. La livraison de m_4 entraîne les mises à jour suivantes :

$$h_3 : id_{mess_E} = 3, id_{mess_D} = 3,$$

$$f_{h_3} = \begin{bmatrix} (1, h_3) \\ 0 \end{bmatrix} \longrightarrow f_{h_3} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$LastRcv_{h_3} = \{(h_1, S_1, \{1,2,4,4\})\}$$

L'information (1, h_3) a été supprimée de f_{h_3} car ce message est délivré et on a plus besoin de transmettre cette information avec les prochains messages.

4.4 Preuve de correction

4.4.1 Condition de livraison

Un message m envoyé par un site mobile h_i (S_i) est délivré à un site mobile h_j (S_j) seulement si la condition suivante est vérifiée :

$$\begin{bmatrix} [depend_m = false] \\ OR \begin{bmatrix} [depend_m = true] \\ AND \begin{bmatrix} [id_{LastSend_m} \neq 0 \; and \; \exists (h_i, \_, SD_i) \in LastRcv_{h_j} \; tel \; que \; id_{LastSend_m} \in SD_i] \\ OR \begin{bmatrix} id_{LastSend_m} = 0 \; and \; \forall k = \overline{1,n}, tel \; que \; \exists (idf, h_j) \in f_m[k] \\ \exists (\_, S_k, SD_k) \in LastRcv_{h_j} \; tel \; que \; idf \in SD_k \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Soient m_1 et m_2 deux messages envoyés à h_j, respectivement par les sites mobiles h_k et h_i, donc :

$$Send(m_1) \longrightarrow Send(m_2) \Leftrightarrow (depend_{m_2} = true)$$

$$AND \begin{bmatrix} (id_{LastSend_{m_2}} = id_{m_1}) & // \; si \quad h_k = h_i \\ OR(\exists (id_{m_1}, h_j) \in f_{m_2}[k]) & // \; si \quad h_k \neq h_i \end{bmatrix}$$

4.4.2 Propriété de sûreté

Sûreté : *l'ordre causal n'est jamais violé.*

Soient deux messages m_1 et m_2 tel que $m_1 \longrightarrow m_2$ et $\nexists m \mid m_1 \longrightarrow m \longrightarrow m_2$; c'est-à-dire l'émission de m_1 précède immédiatement l'émission de m_2. Deux cas sont à considérer :

- *m_1 et m_2 sont émis par le même site mobile h_i*

  Soient (Id_{m_1}, h_j) et (Id_{m_2}, h_j) les identités des événements respectivement d'émission et m_1 et m_2 au niveau de h_i. Supposons que la station de base destinataire S_j où il réside h_j délivre m_2 avant m_1. Ce qui signifie que la condition de livraison pour m_2 est $(depend_{m_2} = true) AND (id_{LastSend_{m2}} < id_{m_1})$ et pour m_1 est $(depend_{m_1} = true) AND (id_{LastSend_{m1}} = id_{m_2})$. Cela implique que l'identificateur du message m_2 est plus petit que l'identificateur du message m_1 et ceci ne peut être vrai que si m_2 est envoyé avant m_1. Ce qui conduit à une contradiction avec l'hypothèse : m_2 a été délivré avant m_1.

- *m_1 et m_2 sont émis par deux sites mobiles distincts $h_k(S_k)$ et $h_i(S_i)$ respectivement*

  Soient (Id_{m_1}, h_j) et (Id_{m_2}, h_j) les identités des événements d'émission de m_1 et m_2. Supposons que m_2 ait été délivré avant m_1 par la station de base destinataire S_j au site mobile h_j. Donc, $m_1 \longrightarrow m_2 \Rightarrow h_i(S_i)$ a sûrement reçu un message m tel que $(Id_{m_1}, h_j) \in f_m[k]$ avant l'émission de $m_2 \Rightarrow (Id_{m_1}, h_j) \in f_{m_2}[k]$.

  La livraison de m_2 par $S_j(h_j) \Leftrightarrow \exists(\_, S_k, Id_{m_1}) \in LastRcv_{h_j}$ et ceci ne peut être vérifié que si S_j a délivré m_1 avant m_2. Ce qui conduit aussi à une contradiction.

4.4.3 Propriété de vivacité

Vivacité : *Chaque message est délivré au bout d'un temps fini.*

- Si $m_1 \longrightarrow m_2$ alors on a deux possibilités : soit $id_{LastSend_{m_2}} = Id_{m_1}$ si m_1 et m_2 sont émis par le même site mobile, soit $(Id_{m_1}, h_j) \in f_{m_1}[k]$ si m_1 et m_2 sont émis par deux sites différents.

  Donc la livraison de m_2 ne peut être différée que par d'autres messages qui ne sont pas encore délivrés à h_j. Pour prouver que ce délai d'attente est fini, nous procédons comme suit : un message m_2 reçu par une station de base S_j ne peut être délivré au site mobile destinataire h_j que si la condition de livraison est vérifiée. Considérons tous les messages qui ne sont pas encore délivrés à h_j de S_j. La relation de *happened before* peut être utilisée pour définir un ordre partiel sur les événements d'émission de ces messages non encore délivrés. Soit m' un de ces messages dans l'ordre partiel dont l'émission n'a pas de prédécesseur. Puisque m' n'a pas été délivré à h_j à sa réception, alors la condition suivante doit être vraie $depend_{m'} = true$ et ceci n'est valable que si m' est causalement lié à un autre message qui a été envoyé avant l'émission de m' soit par le même site ou un site différent. Ce qui conduit à une contradiction avec l'hypothèse que m' n'a pas de prédécesseur.

4.5 Module *handoff*

Pour assurer une gestion fiable de l'ordre causal des messages en environnement mobile, certaines étapes doivent être prises en charge par le module "*handoff*".

Nous supposons qu'un site mobile h_i résidant dans la cellule de la station de base S_i se déplace vers la cellule de la station de base S_j. De plus, nous assumons que chaque station de base utilise un tableau, noté *cellule*, avec une entrée pour chaque site mobile. La $k^{éme}$ entrée de *cellule*, $cellule_i[k]$ représente la connaissance de S_i sur la localisation courante du site mobile h_k.

La première étape établie par h_i est d'envoyer le message *register*(*hi*, *Si*) à S_j pour l'informer de sa présence et de l'identité de son ancienne station de base. En recevant ce message, S_j envoie *handoff_begin*(*hi*) à S_i pour marquer le déclenchement de la procédure de handoff et met $cellule_j[i]$ à S_j. La station de base S_i, en recevant *handoff_begin*(*hi*), transmet les structures de données correspondantes à h_i vers S_j et ceci est exprimé par l'envoi du message $Data(h_i, id_{mess_E}, id_{mess_D}, LastSend_{h_i}, LastRcv_{h_i}, f_{h_i}, AtFile_{h_i})$ et met à jour $cellule_i$[i] ($cellule_i$[i] = $\underline{S_j}$). Ensuite, S_i diffuse le message *NewLocation*(h_i, S_j) à toutes les stations de base à l'exception de S_i et S_j pour les mettre au courant du nouvel emplacement de h_i.

Après la transmission des structures de données de hi vers S_j, si un message destiné à h_i est reçu par S_i, il sera redirigé vers S_j sans le traiter. Sa livraison sera à la charge de S_j après avoir reçu le message $Data(h_i, id_{mess_E}, id_{mess_D}, LastSend_{h_i}, LastRcv_{h_i}, f_{h_i}, AtFile_{h_i})$ à partir de S_i.

La réception du message *NewLocation*(h_i, S_j) par les autres stations de base entraîne la mise à jour de leurs tableaux *cellule* pour exprimer le nouvel emplacement de h_i, et les nouveaux messages destinés à h_i seront envoyés vers sa nouvelle station de base S_j.

La procédure du handoff se termine quand S_j reçoit toutes les données transmises par Si. La terminaison de la procédure du handoff sera exprimée par l'envoi du message *handoff_over*(*hi*) par S_i à S_j, et dés ce moment S_j se charge de maintenir l'ordre causal des messages destinés à h_i.

4.6 Performance et complexité

Dans ce protocole, toutes les structures de données sont localisées sur les stations de base. Ces structures de données sont indépendantes du nombre de sites mobiles du système, ce qui rend le protocole évolutif relativement au nombre des sites mobiles. L'exécution du protocole est entièrement à la charge des stations de base alors que le rôle des sites mobiles est réduit à l'émission et la réception des messages. Les messages transmis sur les liaisons sans fil ne transportent aucune information de contrôle ce qui contribue à minimiser l'utilisation de la bande passante des liaisons sans fil.

Le coût de communication du protocole est de l'ordre de $\sum_{i=1}^{n_s} l_i$, où n_s est le nombre des stations de base dans le système, et l_i est le nombre de messages dans l'entrée $f[i]$. l_i représente le nombre de messages émis par S_i et dont la livraison n'est pas encore assurée. Malgré que ce nombre n'est pas borné mais puisque la réception de chaque message entraîne la mise à jour de cette horloge par la fonction *BUILD* alors nous

pouvons assumer que ce nombre reste réduit et que $\sum_{i=1}^{n_s} l_i << n_s \times n_h$, où n_h est le nombre des sites mobiles dans le système.

La taille maximale que peut atteindre $LastSend_{h_i}$ est égale à $n_h - 1$ et ceci sera le cas quand h_i émet au moins un message à chaque h_j. Par contre, l'inconvénient de cet algorithme est l'évolution non bornée de $LastRcv_{h_i}$ qui découle des caractéristiques des séquences de dépendance. Mais nous pouvons pallier à ce problème en rajoutant une procédure pour éliminer les identificateurs des messages dont l'information sur leur livraison n'est plus nécessaire.

Chaque message émis par un site mobile est immédiatement délivré si la condition de livraison et vérifiée. L'information de contrôle transportée par chaque message émis reflète la relation de causalité effective entre les messages, telle que perçue par leurs sites mobiles. Ainsi, la livraison des messages n'est jamais inhibée comme dans les autres protocoles.

4.7 Conclusion

Dans le but de résoudre le problème d'ordre causal des messages en environnement mobile, nous avons proposé un protocole basé sur l'utilisation des horloges hiérarchiques et des séquences de dépendance. Le protocole proposé se caractérise par l'élimination des délais d'inhibition tout en maintenant une taille minimale de l'information de contrôle. Dans ce protocole, toutes les structures de données sont localisées sur les stations de base. Ces structures de données sont indépendantes du nombre de sites mobiles du système, ce qui rend le protocole évolutif relativement au nombre des sites mobiles. Le module de handoff de notre protocole est simple. Le coût de communication est relativement réduit cependant l'inconvénient de cet algorithme est l'évolution non bornée de $LastRcv_{h_i}$ qui découle des caractéristiques des séquences de dépendance.

Conclusion

L'apparition des systèmes distribués en environnement mobile, avec de nouveaux problèmes induits par les caractéristiques liées au medium de communication sans fil et aux sites mobiles, ouvre un nouveau domaine de recherche qui reste à ses débuts. Le défi majeur de ces recherches est de savoir dans quelle mesure les solutions déjà connues pour les problèmes en environnement statique peuvent être adaptées aux systèmes en environnement mobile. La construction d'algorithmes pour les systèmes répartis et leur coût de communication sont basés sur l'hypothèse implicite que la localisation des participants dans le réseau fixe et la connectivité du réseau ne change pas en absence de défaillances.

Dans cette thèse, nous avons étudié deux problèmes : les mécanismes d'estampillage et l'ordre causal des messages. Ces deux problèmes ont été peu abordés dans les systèmes en environnement mobile.

Du fait que toute communication incluant des sites mobiles prend place à travers la *MSS* de la cellule dans laquelle ils sont présents, donc les relations de causalité peuvent être tracées en utilisant des horloges vectorielles de *n* composantes, où *n* est le nombre des MSS_s dans le système. Ceci peut amener à une réduction significative dans les coûts de communication, car le nombre des MSS_s est généralement une petite fraction du nombre total des nœuds (MH_s et MSS_s) dans ce type de système. Cependant, l'utilisation de seulement *n* entiers dans une horloge vectorielle est insuffisante pour tracer toutes les dépendances causales et conduit à une perte de l'information de concurrence entre les éventements.

Puisque cette perte d'information n'est pas admissible, alors le recours à d'autres alternatives plus efficaces pour le traçage de la causalité dans les systèmes mobiles apparaît indispensable. Avec cet objectif à l'esprit, *Prakash* et *Singhal* [50] ont proposé deux alternatives pour les horloges vectorielles, pour les systèmes mobiles, dites *les séquences de dépendance (dependency sequences)*et *les horloges hiérarchiques (hierarchical clocks).*

Dans une première étape, nous avons fait une étude de ces deux altérnatives et nous avons pu tirer la conclusion suivante : l'approche séquences de dépendance a des taux de communication élevés et n'exige pas un délai de temps élevé pour déterminer le passé d'un événement. L'approche horloge hiérarchique a des taux de communication bas, mais la détermination du passé d'un événement prend du temps.

La première proposition pour l'adaptation des protocoles de mise en œuvre de l'ordre causal des messages pour l'environnement mobile a été faite par Alagar et venkatesan [65]. D'autres contributions ont été faites par la suite, mais elles restent modestes. L'inconvénient majeur de ces protocoles est qu'ils peuvent entraîner des délais

d'inhibition lors de la livraison de messages même si ces derniers ne violent pas l'ordre causal. Cependant le taux de ces délais varie d'un protocole à l'autre. L'approche suivie par les auteurs pour restructurer les protocoles d'ordre causal et les adapter à un environnement mobile consiste à exécuter ces protocoles par les stations de base pour le compte des sites mobiles résidant dans leurs cellules respectives. Bien que cette approche soit assez utilisée dans la plupart des travaux relatifs aux algorithmes répartis en environnement mobile, elle peut cependant aboutir à une perception différente entre les sites mobiles et leurs stations de base respectives, des relations de dépendance causale qui lient les messages échangés.

Dans la deuxième étape de notre travail, nous avons développé un protocole d'ordre causal des messages en environnement mobile, *Mobi_Causal*. L'objectif majeur de notre protocole est d'étudier l'applicabilité des nouveaux mécanismes d'estampillage proposés pour les environnements mobiles (les séquences de dépendance et les horloges hiérarchiques) pour l'implémentation d'un protocole qui résout le problème d'ordre causal en environnement mobile. Nous avons essayé de bénéficier des avantages de ces deux mécanismes pour développer un protocole d'ordre causal des messages pour l'environnement mobile. Notre protocole se caractérise par l'élimination de l'effet d'inhibition dans la livraison des messages tout en maintenant une taille minimale de l'information de contrôle transmise avec chaque message. Cependant, l'inconvénient de notre protocole est l'évolution non bornée de la file $LastRcv_{h_i}$ qui découle des caractéristiques des séquences de dépendances.

Les perspectives possibles à ce travail sont : de rajouter une procédure pour éliminer les identificateurs des messages à partir de $LastRcv_{h_i}$ dont l'information sur leur livraison n'est plus nécessaire dans le but de minimiser sa taille; et l'adaptation de ce protocole pour les problèmes du *Multicast* et du *Broadcast* avec gestion de la vue de groupe.

Le développement des algorithmes et plus généralement des applications réparties en environnement mobile soulève plusieurs problèmes tels que les problèmes liés à la mobilité des sites et au medium de communication sans fil. Ce sont ces problèmes qui rendent difficile l'analyse des performances et l'efficacité des algorithmes répartis en environnement mobile. La mobilité des sites introduit de nouvelles considérations aux problèmes rencontrés dans les systèmes répartis statiques et par conséquent un coût de calcul et de communication supplémentaire dans la complexité des algorithmes répartis. La recherche dans le domaine des systèmes répartis en environnement mobile reste un domaine largement ouvert.

Bibliographie

[1] M. Raynal. Distributed Algorithms and Protocols. Wiley and Sons, 1988.

[2] B. R. Badrinath, A. Acharya et T. Imielinski. *Designing Distributed Algorithms for Mobile Computing NetWorks.* Rapport technique, Departement of Computer Sciences, University of Rutgers, US, 1994.

[3] B. R. Badrinath, A. Acharya et T. Imielinski. *Structuring Distributed Algorithms for Mobile Hosts.* In 14th International Conference on Distributed Computing Systems, Poznan, Poland, pages 21-28, June 1994.

[4] B. R. Badrinath, A. Acharya et T. Imielinski. *Impact of Mobility on Distributed Computations.* ACM Operating Systems Review, vol. 27, n° 2, April 1993.

[5] Baggio. *Environnements Mobiles : Etude et Synthèse Bibliographique.* DEA de Systèmes Informatiques, au Laboratoire MASI, Université Pierre et Marie Curie, 1995.

[6] J. Ioannidis, D. Duchamp, et G. Q. Maguire. *Ipbased Protocols forMobile Internetworking.* In Proc. of ACM SIGCOMM Symposium on Communication, Architectures and Protocols, pages 235–245, September 1991.

[7] N. Badache. *La Mobilité dans les Systèmes Répartis.* Publication Interne IRISA n° 962, Octobre 1995. ftp://ftp.irisa.fr /techreports/1995/PI–962.ps.gz.

[8] N. Aleb. Spécification des Systèmes en Environnement Mobile à l'aide du Pi-calcul. Thèse de Magister, USTHB, Octobre 2000.

[9] E. Pitoura, B. Bhargava. *Dealing with Mobility: Issues and Research Challenges.* Technical Report TR-93-070, Department of Computer Science, Purdue University, November 1993.

[10] G. Forman, J. Zahorjan. *The Challenges of Mobile Computing.* Computer Science & Engineering, University of Washington, US, IEEE Computer, pages 39-47, April 1994.

[11] J. J. Kistler, M. Satyanarayanan. *Disconnected Operation in the Coda File System.* Thirteenth ACM Symposium on Operating Systems Principles, Asilomar Conference Center, Pacific Grove, US, vol. 25, pages 213-225, 1991.

[12] B. Marsh, F. Douglis, R. Cáceres. *System Issues in Mobile Computing.* Matsushita Information Technology Laboratory, Technical Report MITL-TR-50-93, February 1993.

[13] T. Imielinski et B. R. Badrinath. *Mobile Wireless Computing: Solutions and Challenges in Data Management.* CACM, vol. 37, n° 10, pages 18-28, October 1994.

[14] E. Pitoura, B. Bhargava. *Building Information Systems for Mobile Environnement*s. Department of Computer Sciences, Purdue University, Third International Conference on Information and Knowledge Management, pages 371-378, November 1994.

[15] M. Weiser. *Les Réseaux Informatiques de l'An 200*0. Pour la Science, n° 169, pages 72-84, novembre 1991.

[16] F. Bennett, T. Richardson, A. Harter. *Teleporting Applications Mobil*e. Olivetti Research Laboratory, IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US, December 1994.

[17] T. Imielinski, B. R. Badrinath. *Data Management for Mobile Computing.* Department of Computer Science, Rutgers University 2, SIGMOD RECORD, vol. 22, n° 1, pages 34-39, March 1993.

[18] T. Imielinski, B. R. Badrinath. *Mobile Wireless Computing: Challenges in Data Management*. Department of Computer Science, Rutgers University.

[19] B. R. Badrinath, T. Imilelinski, A. Virmani. *Locating Strategies for Personal Communication Networks.* Department of Computer Science, Rutgers University, IEEE Globecom 92 Workshop on Networking of Personal Communications Applications, Orlando, US, December 1992.

[20] T. Imielinski, B. R. Badrinath. *Querying in Highly Mobile Distributed Environments.* Department of Computer Science, Rutgers University, 18th Very Large Data Bases, pages 41-52, August 1992.

[21] K. S. Meier-Hellstern, E. Alonso, D. O'Neil. *The Use of the SS7 and GSM to Support High Density Personal Communications*. Third WinLab Workshop on Third Generation Wireless Information Networks, pages 49-57, April 1992.

[22] L. J. Ng et al. Distributed Architectures and Databases for Intelligent Personal Communication Networks. ICWC, pages 300-304, June 1992.

[23] D. J. Goodman. *Trends in Cellular and Cordless Communications*. IEEE Communication Magazine, June 1991.

[24] J. Ioannidis, G. Q. Maguire Jr. *The Design and Implementation of Mobile Internetworking Architecture*. Columbia University, Usenix Winter 1993 Conference, January 1993.

[25] B. R. Badrinath, T. Imielinski. *Replication and Mobility.* Department of Computer Science, Rutgers University, Second IEEE Workshop on Management of Replicated Data, pages 9-12, November 1992.

[26] G. Liu. Exploitation of Location-dependent Caching and Prefetching Techniques for Supporting Mobile Computing and Communications. Ericsson Radio Systems AB, 6th International Conference on Wireless Communications, July 1994.

[27] M. R. Ebling, L. B. Mummert, D. C. Steere. *Overcoming the Network Bottleneck in Mobile Computing*. School of Computer Science, Carnegie Mellon University, IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US, December 1994.

[28] M. Ahamad, F. J. Torres-Rojas, R. Kordale, J. Singh, S. Smith. *Detecting Mutual Consistency of Shared Objects.* College of Computing, Georgia Institute of Technology, Department of Computer Science, Rice University, IEEE Workshop

on Mobile Computing Systems and Applications, Santa Cruz, CA, US, December 1994.

[29] R. Gruber, F. Kaashoek, B. Liskov, L. Shrira. *Disconnected Operation in the Thor Object-Oriented Database System*. Laboratory for Computer Science, Massachusetts Institute of Technology, IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US, December 1994.

[30] R. Alonso, H. F. Korth. *Database System Issues in Nomadic Computing.* Matsushita Information Technology Laboratory, International Conference on Management of Data, ACM SIGMOD RECORD, vol. 22, pages 388-392, may 1993.

[31] B. R. Badrinath, A. Acharya, T. Imielinski. *Impact of Mobility on Distributed Computations*. Department of Computer Science, Rutgers University, ACM Operating System Review, vol. 27, n° 2, pages 15-20, April 1993.

[32] N. Badache. *Ordre Causal et Tolérance aux Défaillances en Environnement Mobile.* Thèse de Doctorat, Département: Architecture et Systèmes, Laboratoire: Systèmes d'Exploitation et Réseaux. USTHB, Octobre 1998.

[33] G. Le lann. *Distributed Systems: Towards a Formal Approach.* In: IFIP Congress, pages 155-160, Toronto, August 1977.

[34] B. R. Badrinath, A. Acharya, T. Imielinski. *Structuring Distributed Algorithms for Mobile Hosts.* Department of Computer Science, Rutgers University, 14th International Conference on Distributed Computing Systems, Poznan, Poland, may 1994.

[35] B. R. Badrinath, A. Acharya, T. Imilelinski. *Designing Distributed Algorithms for Mobile Computing Networks.* Department of Computer Science, Rutgers University, 1994.

[36] Acharya et B. R. Badrinath. *A Framework for Delivering Multicast Messages in Networks with Mobile Hosts.* Rapport Technique n° DCS-TR-310, Department of Computer Science, University of Rutgers, 1994.

[37] L. Lamport. Time, Clocks and the Ordering of Events in Distributed Systems. CACM, vol. 21, n° 7, pages 558-565, 1978.

[38] S. Alagar et S. Venkatesan. *Causally ordered Message Delivery in Mobile Systems.* IEEE Transactions on Computers, vol. 46, n° 3, pages 353-361, March 1997.

[39] M. Raynal, A. Schiper et S. Toueg. *The Causal Ordering and a Simple Way to Implement it.* Information Processing Letters, 1991.

[40] E. W. Dijkstra et al. Derivation of a Termination Detection Algorithm for Distributed Computation. In Information Processing Letters, June 1983.

[41] Y. Amir et al. *Fast Message Ordering and Membership Using a Logical Token-Passing Ring.* In Proc. Of the 13th Intl. Conf. on Distributed Computing Systems, May 1993.

[42] K. Raymond. A tree-based algorithm for distributed mutual exclusion. ACM Transactions on Computer Systems, 7(1), February 1989.

[43] L. Lamport. *Time, Clocks and the Ordering of Events in a Distributed System*. Communications of the ACM, 21(7) : 558-565, July 1978.

[44] Michel Raynal. *About Logical Clocks for Distributed Systems.* Publication Interne IRISA n° 607, Octobre 1991.

[45] Colin Fidge. *Logical Time in Distributed Computing Systems.* IEEE Computer, 24(8) :28-33, August 1991.

[46] R. Schwarz and F. Mattern. *Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail*. Distributed Computing, 3(7) :149-174, 1994.

[47] F. Mattern. *Logical Time.*

[48] M.Raynal, M.Singhal. *Logical Time: A Way to Capture Causality in Distributed Systems.* INRIA. N° 2472, Mars 1995.

[49] R. Baldoni, M. Raynal. *Fundamentals of Distributed Computing: A Practical Tour of Vector Clock Systems.* IEEE Distributed Systems ONLINE, February 2002.

[50] Ravi Prakash and Mukesh Singhal. *Dependency Sequences and Hierarchical Clocks : Efficient Alternatives to Vector Clocks for Mobile Computing Systems.* Wireless Networks, (3) : 349-360,1997. Also in Mobicom96.

[51] Gahlot and M. Singhal. *Hierarchical Clocks.* Technical Report OSU-CISRC-93-TR19, The Ohio State University, Computer and Information Science Research Center, 1993.

[52] Carlos Baquero and Francisco Moura. *Improving Causality Logging in Mobile Computing Networks.* ACM Mobile Computing and Communications Review, 2(4) :62-66, October 1998.

[53] K. Birman, T. Joseph. *Reliable Communication in the Presence of Failures*. ACM Transactions on Computer Systems, 5(1): 47-76, February 1987.

[54] K. Birman, R. (eds.) Van Renesse. *Reliable Distributed Computing with the ISIS Toolkit*. IEEE Computer Society Press, 1994.

[55] K. Birman, T. Joseph. *Exploiting replication in Distributed Systems.* In S. Mullender, ed., Distributed Systems, (ACM, New York, 1990).

[56] Alagar, S. Venkatesan. *An Optimal Algorithm for Distributed Snapshots with Causal Message Ordering*. Information Processing Letters, 50, 311-316, 1994.

[57] M. Ahamad, P. Hutto, R. John. *Implementing and Programming Causal Distributed Memory*. Proceedings of the 11th Intl. Conf. on Distributed Computing Systems, 274-281, 1991.

[58] K. Ravindran, B. Prasad. *Communication Structures and Paradigms for Distributed Conferencing Applications*. Proc. 12th Intl. Conf. on Distributed Computing Systems, May 1992.

[59] F. Adelstein, M. Singhal. *Real-Time Causal Message Ordering in Multimedia Systems*, Proc. 15th Intl. Conf. on Distributed Computing Systems, May -june1995.

[60] P. Kearns, B. Koodalattupuram. *Immediate Ordered Service in Distributed Systems.* Proc. 9th Intl. Conf. on Distributed Computing Systems, Newport Beach, California, 611-618, June 1989.

[61] K. Birman, A. Schiper, P. Stephenson. *Lightweight Causal and Atomic Multicast*. ACM Transactions on Computer Systems, 9(3): 272-314, August 1991.

[62] L. L. Peterson, N. C. Buchholz, R. D. Schlichting. *Preserving and Using Context Information in Interprocess Communication*. ACM Trans. on Computer Systems, 1989.

[63] Schiper, J. Eggli, A. Sandoz. *A New Algorithm to Implement Causal Ordering*. Proc 3rd Int. Workshop on Distributed Algorithms, 219-232, 1989, in Lecture Notes in Computer Science 392, Springer-Verlag.

[64] M. Raynal, A. Sciper, S. Toueg. *The Causal Ordering Abstraction and a Simple Way to Implement It*. Information Processing Letters, 39(6): 343-350, 1991.

[65] S. Alagar, S. Venkatesan. *Causally Ordered Message Delivery in Mobile Systems.* In: Proc. Workshop on Mobile Computing Systems and Applications, 169-174, December 1994.

[66] L. H. Yen, T. L. Huang, S. Y. Hwang. *A Protocol for Causally Ordered Message Delivery in Mobile Computing Systems.* Mobile Networks and Applications, 1997.

[67] R. Prakash, M. Raynal, M. Singhal. *An Efficient Causal Ordering Algorithm for Mobile Computing Environments.* In Proc. Of the 16th Intl. Conf. on Distributed Computing Systems, 1996.

[68] C. Skawratananond, N. Mittal, V. K. Garg. A Lightweight Algorithm for Causal Message Ordering in Mobile Computing Systems. 1999.