République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Tébessa

Faculté des Sciences Exactes
et Sciences de la Nature et de la Vie

Département de Mathématiques et Informatique

Mémoire
Présenté en vue de l'obtention du diplôme de MAGISTER

Option : Informatique Répartie et Mobile

Par
Lotfi TOUALBIA

Renforcement de la Robustesse des Applications Distribuées sur Réseaux Mobiles avec Infrastructure

Date de soutenance : 29/01/2015

Devant le jury

M. Farid MOKHATI	MCA	Université d'Oum El Bouaghi	Président
M. Hakim BENDJENNA	MCA	Université de Tébessa	Examinateur
M. Makhlouf DERDOUR	MCA	Université de Tébessa	Examinateur
M. Makhlouf ALIOUAT	MCA	Université de Sétif 1	Rapporteur

Dédicace

À la mémoire de mon père,

À ma mère,

À mon frère et mes sœurs,

À toute ma famille,

À tous mes amis.

TOUALBIA Lotfi

Remerciements

En premier lieu, je remercie Allah qui m'a donné la force et le courage pour accomplir ce modeste travail. C'est grâce à lui que mon chemin est éclairé pour arriver jusqu'ici.

Je remercie très vivement mon encadreur Dr. ALIOUAT Makhlouf, Maître de conférences à l'Université Ferhat Abbas de Sétif, pour avoir accepté de m'encadrer. Je tiens encore à lui exprimer ma profonde gratitude pour son aide, ses encouragements et ses précieux conseils.

Je tiens à remercier également les membres du jury pour avoir accepté d'examiner et d'évaluer ce mémoire et pour l'intérêt qu'ils ont porté à mon travail.

Mes plus vifs remerciements vont à ma famille pour m'avoir soutenu tout au long des années d'études. Qu'elle trouve ici l'expression de ma sincère gratitude.

Ma reconnaissance s'adresse aux personnes qui m'ont aidé et encouragé dans les moments difficiles. Particulièrement, à mon amie Othaila.

Enfin, merci à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Résumé

L'intérêt procuré par les réseaux mobiles de divers types n'est plus à démontrer. Cependant, un tel environnement engendre de nouvelles contraintes : des ressources de calcul et de stockage moins importantes, une bande passante faible, une source d'énergie limitée, des fréquentes déconnexions, une vulnérabilité aux dommages physiques, etc. Ces particularités rendent les protocoles de tolérance aux fautes par reprise traditionnels inadéquats. Par conséquent, ces protocoles doit être adaptés à ce contexte afin d'assurer la continuité de délivrance des services d'une application distribuée mobile en dépit de défaillances de processus.

Dans ce mémoire, nous présentons un protocole de tolérance aux fautes d'une application distribuée dans un environnement mobile basé sur un réseau mobile avec infrastructure à couverture discontinue. Notre objectif est d'assurer à une application distribuée mobile une exécution non-stop et minimiser ainsi l'overhead temporel affectant la délivrance de ses services. A cet effet, nous identifions en premier lieu les différents types de fautes matérielles transitoires et permanentes affectant un nœud mobile dans cet environnement. Ensuite, des mesures adéquates doit être prises en compte pour tolérer ces fautes.

Le protocole proposé se base sur la sauvegarde asynchrone de points de reprise et la journalisation pessimiste de messages afin de fournir un recouvrement asynchrone aux processus défaillants. La station de base fournit un support stable pour enregistrer les points de reprise d'un processus ainsi que ses messages reçus. L'exécution non stop est assurée par la migration des processus d'un nœud mobile défaillant vers la station de base ou vers un autre nœud mobile robuste. Enfin, les performances de notre protocole sont évaluées.

Mots-clés : Réseaux mobiles avec infrastructure, Applications distribuées, Handover, Couverture discontinue, Tolérance aux fautes, Recouvrement par reprise.

Abstract

The advantage provided by the mobile networks of various types is well established. However, this environment generates new constraints : smaller computing and storage resources, low bandwidth, limited energy source, frequent disconnections, vulnerability to physical damages, etc. These features make the traditional rollback recovery protocols inadequate. Therefore, these protocols should be adapted to this context to ensure the continuity of service delivery for a mobile distributed application in spite of process failures.

In this paper, we present a fault tolerance protocol for a mobile distributed application in a mobile environment based on cellular mobile network with discontinuous coverage. Our goal is to provide a not stop execution for a mobile distributed application and minimize the time overhead affecting the delivery of its services. For this purpose, we first identify the different types of transient and permanent hardware faults affecting a mobile node in this environment. Then, appropriate measures must be taken into account to tolerate these faults.

The proposed protocol is based on asynchronous checkpointing and pessimistic message logging to provide an asynchronous recovery to failed process. The base station provides a stable storage to save the checkpoints of a process and its received messages. The non-stop execution is provided by the migration of the processes of a failed mobile node to the base station or to another robust mobile node. Finally, the performance of our protocol is evaluated.

Keywords : Cellular Mobile networks, Distributed applications, Handoff, Discontinuous coverage, Fault tolerance, Rollback recovery.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'informatique mobile est un domaine en plein essor qui profite des percées technologiques dans les domaines des terminaux portables et des réseaux de communication sans fil. Cet environnement n'astreint plus l'usager à une localisation fixe, mais lui permet une libre mobilité tout en assurant sa connexion au réseau. Deux familles de réseaux sans fil et mobiles sont à même de servir de support aux services de l'informatique mobile : les réseaux mobiles avec infrastructure (cellulaire) et les réseaux mobiles sans infrastructure (ad hoc).

Nous nous intéressons dans ce mémoire aux réseaux mobiles avec infrastructure qui connaissent une très forte expansion à l'heure actuelle. Un tel réseau est constitué d'un ensemble de stations de base fixes, définissant chacune une cellule de communication. Cette architecture est utilisée dans les réseaux de téléphonie mobile comme les réseaux GSM, leurs concurrentes des réseaux sans fil comme les réseaux WiMAX et les réseaux locaux sans fil comme les réseaux WiFi. Dans les réseaux cellulaires étendus, utilisés initialement pour le support de la téléphonie mobile, les cellules proches se recouvrent partiellement pour assurer efficacement une couverture étendue et continue. À l'opposé de cette complexité, les réseaux de quatrième génération (4G) sont composés d'un ensemble de cellules haut débit, réparties de façon discontinue afin de réduire les coûts de déploiement et limiter les problèmes de planification radio.

Cet environnement mobile est caractérisé par des nouvelles contraintes affectant un terminal mobile : des ressources de calcul et de stockage moins importantes, une bande passante faible et variable, une source d'énergie limitée, des fréquentes déconnexions qui peuvent être volontaires ou involontaires, une vulnérabilité aux dommages physiques, etc.

La tolérance aux fautes dans les systèmes distribués est un besoin imposé par la répartition et c'est pourquoi plusieurs travaux de recherche ont été réalisés afin de la prendre en compte. Dans un tel système, on peut distinguer deux approches principales de tolérance aux fautes : l'approche par réplication et l'approche par reprise. La réplication nécessite un nombre important de ressources (calcul, stockage, etc.), implique un surcoût pour gérer les répliques multiples et limite le nombre de pannes tolérées. Tandis que, la reprise se base sur la redondance d'informations et semble l'approche la plus adaptée aux systèmes distribués.

Le but d'un protocole de reprise est de capturer suffisamment de données pendant une exécution distribuée sans panne et de stocker ces données sur une mémoire stable assurant leur accessibilité ainsi que leur protection contre les pannes. Lorsqu'une panne est détectée, les

données collectées sont utilisées pour redémarrer tout ou partie de l'application de sorte que l'état formé après la reprise soit cohérent. Les protocoles de reprise se décomposent en deux catégories : les protocoles basés sur la sauvegarde de points de reprise (checkpointing) et ceux basés sur la journalisation de messages (message logging). L'approche par sauvegarde vise à déterminer un ensemble de points de reprise formant un état global cohérent. L'approche par journalisation vise à enregistrer les évènements non déterministes permettant de rétablir l'état de l'application exactement tel qu'il était avant la panne.

Les particularités de l'environnement mobile et ses effets sur les mécanismes de tolérance aux fautes rendent les protocoles de reprise traditionnels inadéquats. Ainsi, différents travaux visant l'adaptation de ces protocoles à ce contexte ont été proposés dans la littérature. Cependant, ces travaux ont fait l'hypothèse que les fautes matérielles causant la défaillance d'un processus sont transitoires et leur durée de persistance se compte en millisecondes [Aliouat 2009]. De plus, l'environnement mobile considéré se base sur un réseau mobile avec infrastructure à couverture continue.

La problématique peut s'énoncer ainsi : comment assurer la continuité de délivrance des services d'une application distribuée en dépit de la défaillance d'un ou plusieurs de ses processus à cause des fautes matérielles transitoires ou permanentes affectant leurs nœuds mobiles dans un réseau mobile avec infrastructure à couverture discontinue ?

C'est ce constat qui a motivé notre travail dont l'objectif est d'apporter une contribution en matière de la résistance aux fautes dans cet environnement fragilisant applications en traitant à la fois des fautes matérielles transitoires et permanentes [Aliouat 2007]. Dans ce contexte caractérisé par des fréquentes déconnexions, des mesures adéquates doit être prises en compte afin d'assurer à une application distribuée mobile une exécution non-stop et minimiser ainsi l'overhead temporel affectant la délivrance de ses services.

Notre contribution dans ce mémoire consiste à proposer un protocole de tolérance aux fautes d'une application distribuée dans un environnement mobile avec infrastructure à couverture discontinue. Ce protocole est basé sur la sauvegarde indépendante (non coordonnée) de points de reprise et la journalisation pessimiste de messages pour offrir un recouvrement asynchrone aux processus défaillants. À cause de l'instabilité des supports de stockage des nœuds mobiles, les points de reprise d'un processus et les messages qui lui sont destinés sont enregistrés sur le support stable de la station de base. Nous nous focalisons principalement sur l'aspect de la continuité de délivrance des services (une exécution non-stop) afin de subir à l'application en cours d'exécution un minimum d'overhead temporel. À cet effet, la migration de processus est utilisée pour poursuivre l'exécution des processus d'un nœud

mobile défaillant sur la station de base ou sur un autre nœud mobile robuste choisi par un algorithme d'élection. Enfin, nous évaluons les performances de notre protocole.

Ce document est organisé en deux parties : la première présente à travers trois chapitres un état de l'art sur les domaines des réseaux mobiles avec infrastructure et de la tolérance aux fautes dans les systèmes distribués, tandis que la deuxième contient un seul chapitre consacré à notre contribution.

Le chapitre 1 introduit les réseaux sans fil et mobiles de façon générale. Il met l'accent sur les aspects essentiels liés à un réseau mobile avec infrastructure : l'architecture, les contraintes, le handover et la gestion de la mobilité. Ensuite, nous décrivons les deux technologies de réseaux sans fil : WiFi et WiMAX. Enfin, nous présentons les réseaux de téléphonie mobile, leurs différentes générations ainsi que les principales technologies y associées.

Le chapitre 2 présente le vocabulaire relatif à la sûreté de fonctionnement et introduit la tolérance aux fautes comme un moyen pour l'assurer. Nous commençons par présenter la notion de sûreté de fonctionnement, ses attributs, ses entraves, et ses moyens. Ensuite, nous étudions les différents mécanismes de tolérance aux fautes. Enfin, nous décrivons les techniques spécifiques aux systèmes distribués.

Le chapitre 3 présente la technique de tolérance aux fautes par reprise dans un système distribué. Nous décrivons tout d'abord un modèle de l'application distribuée, puis nous exposons deux problèmes liés à la reprise : le déterminisme d'exécution et la cohérence d'état global. Ensuite, nous étudions en détails un panorama des protocoles de reprise proposés dans la littérature. Enfin, une comparaison entre ces protocoles est faite selon différents critères.

Le chapitre 4 est consacré à notre contribution qui consiste en une proposition d'un protocole de tolérance aux fautes par reprise adapté au contexte d'un réseau mobile avec infrastructure à couverture discontinue. Nous présentons d'abord la problématique et les motivations de cette proposition. Ensuite, nous décrivons le modèle du système et les hypothèses y associées. Après, nous présentons l'idée de base ainsi que les détails de notre protocole. Enfin, nous évaluons les performances du protocole proposé.

Nous terminons ce document par une conclusion générale qui résume les apports essentiels de notre travail et présente quelques perspectives d'amélioration et d'extension.

Première partie

État de l’art

CHAPITRE 1

Réseaux mobiles avec infrastructure

Sommaire

1.1 Introduction

Les réseaux sans fil et mobiles ont connu un essor sans précédent ces dernières années grâce à leur grande flexibilité d'utilisation. En particulier, ils permettent la mise en réseau des sites dont le câblage serait trop onéreux à réaliser dans leur totalité, voire même impossible. On peut distinguer deux catégories de réseaux sans fil et mobiles : les réseaux avec infrastructure (cellulaire) et les réseaux sans infrastructure (ad hoc).

Les applications distribuées ciblant des terminaux mobiles s'appuient le plus souvent sur des réseaux mobiles avec infrastructure. En exploitant des technologies de la téléphonie cellulaire (GSM, GPRS, UMTS, etc.) ou plus récemment des normes comme IEEE 802.11 (WiFi) ou IEEE 802.16 (WiMAX), les terminaux mobiles de ces réseaux accèdent, via une liaison radio, à un équipement fixe (station de base ou point d'accès) servant de passerelle vers un réseau filaire. Dans ce type d'architecture, deux terminaux mobiles doivent forcément passer par l'infrastructure pour communiquer.

Dans ce chapitre, nous allons présenter les réseaux mobiles avec infrastructure. Dans un premier temps, nous allons introduire les réseaux sans fil et mobiles de façon générale. Par la suite, nous allons étudier les différents aspects de base liés à un réseau mobile avec infrastructure : son architecture, ses contraintes, la gestion des handovers (handoff) et la

gestion de la mobilité avec IP mobile. Après, nous allons décrire deux technologies de réseaux sans fil, à savoir le WiFi et le WiMAX. Enfin, nous allons aborder les réseaux de mobiles en présentant leurs concepts, les différentes générations et les principales technologies, en particulier le GSM, le GPRS et l'UMTS.

1.2 Réseaux sans fil et mobiles

1.2.1 Définition et concepts

Un système de communication, ou réseau, désigne tout ensemble d'éléments capables de véhiculer de l'information d'une source vers une destination [Agha 2001]. Un réseaux sans fil ou encore un réseau mobile est un réseau dans lequel au moins deux terminaux peuvent communiquer sans liaison filaire. La communication est assurée par des liaisons hertziennes (ondes radioélectriques) ou par la lumière infrarouge à la place des câbles habituels. Grâce aux réseaux sans fil, un utilisateur a la possibilité de rester connecté tout en se déplaçant dans une zone géographique plus ou moins étendue [Lemainque 2009].

Les termes *mobile* et *sans fil* sont souvent utilisés pour décrire les technologies existantes, tels que le GSM, UMTS, Bluetooth, IEEE 802.11, etc. Il est cependant important de distinguer les deux notions de façon à éviter toute confusion [Agha 2001].

Le concept *sans fil* est étroitement associé au support de transmission. Un système est dit sans fil s'il propose un service de communication totalement indépendant de l'emplacement des équipements informatiques composant le réseau. Ainsi, les sites peuvent être fixes mais communiquent via un support de transmission sans fil (infrarouge, ondes hertziennes, etc.).

Le concept *mobile* est étroitement associé au déplacement ou à la mobilité du terminal portable qui peut prendre deux aspects différents :

- Le déplacement de l'utilisateur dans un réseau mobile : un réseau mobile désigne un réseau sans fil dans lequel l'utilisateur reste connecté au réseau durant son déplacement et communique de façon transparente avec les autres utilisateurs.
- La portabilité du terminal et des applications ou le nomadisme : dans ce cas, l'utilisateur itinérant peut utiliser son terminal de façon autonome déconnecté du réseau. Lorsqu'il souhaite se connecter, il utilise une liaison filaire ou sans fil.

1.2.2 Classification des réseaux sans fil et mobiles

Les réseaux sans fil et mobiles peuvent être classifiés selon deux critères différents : la zone de couverture et l'infrastructure de communication (voir la figure 1.1).

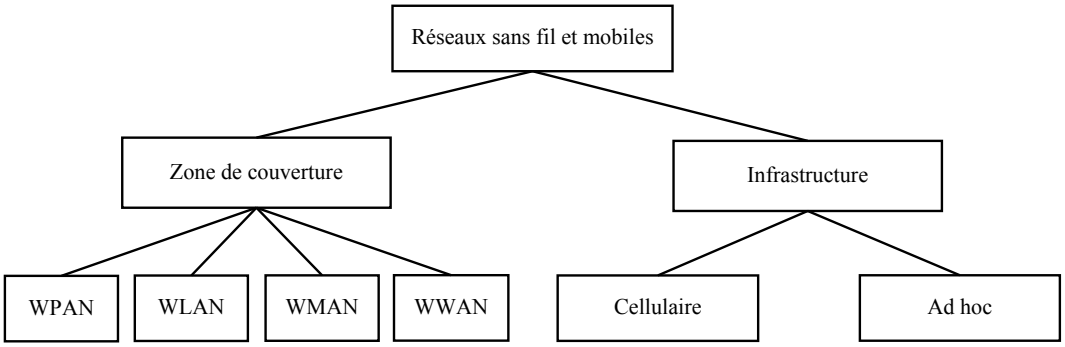

FIGURE 1.1 – Classification des réseaux sans fil et mobiles.

1.2.2.1 Classification selon la zone de couverture

Comme pour les réseaux filaires, les réseaux sans fil et mobiles sont habituellement classés en fonction de leur étendue (zone de couverture) en quatre catégories [Geier 2004] :

- Réseaux personnels (WPAN : Wirelees Personal-Area Network) : ces réseaux sont d'une faible portée, de l'ordre d'une dizaine de mètres. Par exemple la technologie Bluetooth.
- Réseaux locaux (WLAN : Wireless Local-Area Network) : d'une couverture géographique de quelques centaines de mètres. Par exemple la technologie WiFi.
- Réseaux métropolitains (WMAN : Wireless Metropolitan Area Network) : couverture dans les limites d'une ville. Par exemple la technologie WiMax.
- Réseaux étendus (WWAN : Wireless Wide Area Network) : assurent une couverture mondiale. Par exemple les technologies de réseaux de mobiles.

La figure 1.2 montre ces différentes catégories selon le périmètre géographique et les différentes technologies de communication qui y sont appliquées.

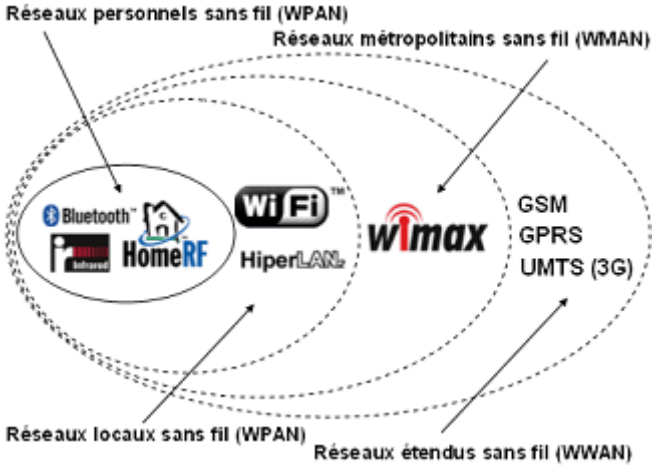

FIGURE 1.2 – Catégories des réseaux sans fil selon la zone de couverture [Lemainque 2009].

1.2.2.2 Classification selon l'infrastructure de communication

Selon l'infrastructure de communication, les réseaux sans fil et mobiles se décomposent en deux classes : les réseaux mobiles avec infrastructure (voir figure 1.3) et les réseaux mobiles sans infrastructure (voir figure 1.4) [Lemlouma 2000].

A. Réseaux mobiles avec infrastructure (Cellulaire)

Un réseau mobile avec infrastructure est également appelé réseau cellulaire. Ce réseau se compose d'un ensemble de terminaux mobiles (nœuds mobiles) communiquant via un ou plusieurs points d'accès (stations de base). Les points d'accès sont fixes et reliés entre eux par un réseau filaire qui se charge de la partie routage et des fonctions d'administration. Les terminaux mobiles se déplacent librement mais ne communiquent jamais directement les uns avec les autres. Toutes les communications se font systématiquement vers le point d'accès le plus proche qui se charge ensuite de les relayer à la destination [Conchon 2006].

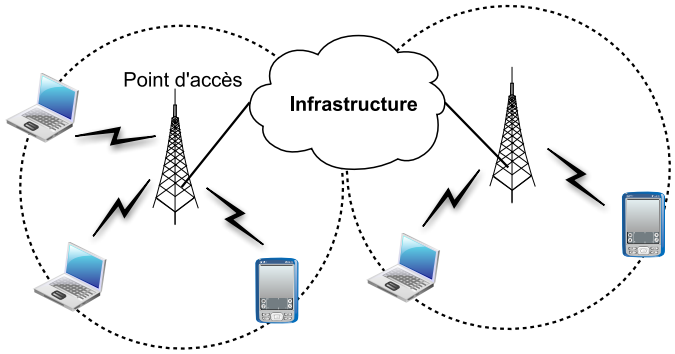

FIGURE 1.3 – Réseaux mobiles avec infrastructure (Cellulaire).

B. Réseaux mobiles sans infrastructure (Ad hoc)

Un réseau ad hoc ou MANET (Mobile Ad hoc Network) est une collection de nœuds mobiles interconnectés par une technologie sans fil, formant un réseau temporaire sans recourir à aucune infrastructure fixe ou administration centralisée. Dans ce type de réseau, les nœuds mobiles se comportent comme des hôtes et/ou des routeurs. Donc, deux nœuds qui sont dans la portée radio l'un de l'autre communiquent directement alors que ceux n'étant pas à portée radio utilisent des nœuds intermédiaires comme relais pour acheminer les paquets de données (routage multi-sauts). Ce système peut fonctionner d'une manière isolée ou s'interfacer à des réseaux fixes à travers des passerelles [Khelladi 2004][Corson 1999].

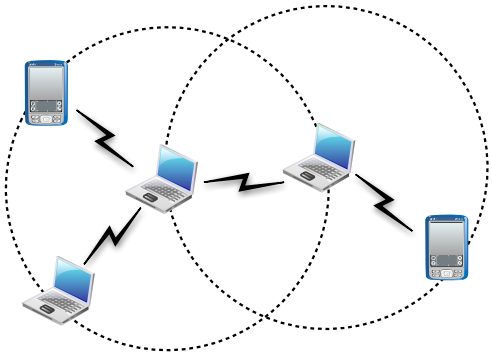

FIGURE 1.4 – Réseaux mobiles sans infrastructure (Ad hoc).

1.3 Environnement mobile

Un environnement mobile est un système composé de sites mobiles et qui permet à ses utilisateurs d'accéder à l'information indépendamment de leurs positions géographiques. On distingue deux modèles pour un environnement mobile : le modèle cellulaire basé sur un réseau mobile avec infrastructure et le modèle ad hoc basé sur un réseau mobile sans infrastructure. Cette section est consacrée aux principaux concepts liés à un réseau mobile avec infrastructure.

1.3.1 Architecture de l'environnement mobile

Un réseau mobile avec infrastructure ou réseau cellulaire est constitué de deux types de sites (unités ou nœuds) : les sites fixes (station de travail, serveur, etc.) et les sites mobiles (téléphone, PDA, PC portable, etc.). Les sites fixes sont interconnectés par un réseau filaire formant ainsi la partie fixe du réseau. Certains sites fixes appelés stations de base (SB) sont munis d'une interface de communication sans fil pour la communication directe avec les unités mobiles (UM) localisées dans une zone géographique limitée appelée cellule (voire figure 1.5) [Lemlouma 2000].

A chaque station de base correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages via des liaisons sans fil ayant une bande passante limitée. Donc, pour envoyer un message d'une unité mobile UM1 à une autre unité mobile UM2, l'UM1 doit envoyer le message à sa station de base SB1 qui le transmet à la station de base de l'unité UM2 appelée SB2, qui à son tour le transmet à UM2.

Dans ce modèle, une unité mobile ne peut être, à un instant donné, directement rattachée qu'à une seule station de base à travers laquelle elle peut communiquer avec les autres sites. Lorsqu'un site mobile se déplace et change de cellule, le cheminement de l'information doit

être modifié vers la nouvelle station de base. Cette modification s'appelle un changement intercellulaire, ou handover, ou encore handoff [Lemlouma 2000].

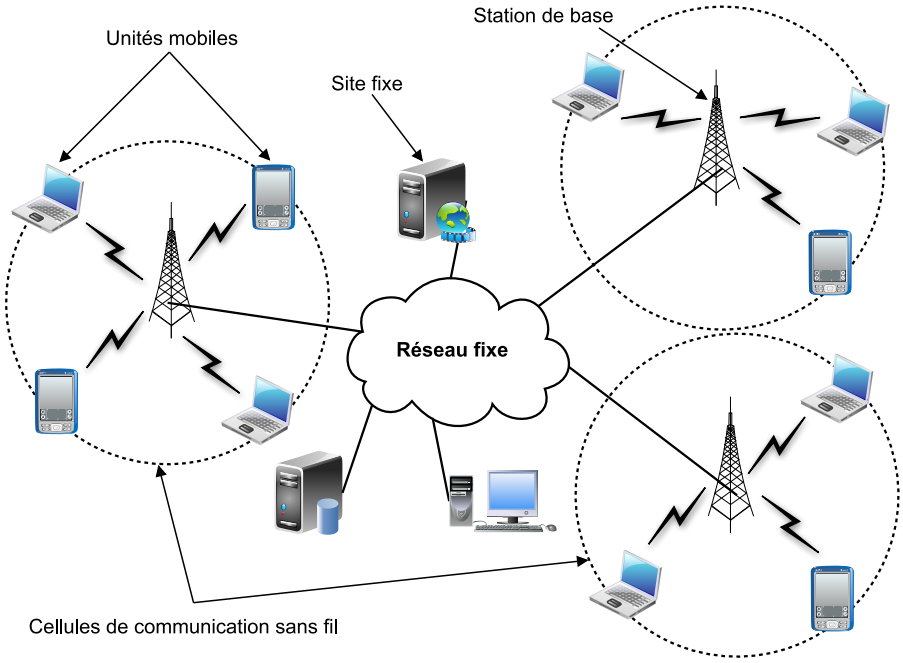

FIGURE 1.5 – Architecture d'un réseau mobile avec infrastructure (cellulaire).

1.3.2 Contraintes de l'environnement mobile

La communication sans fil, la portabilité et la mobilité des unités mobiles engendrent de nouvelles contraintes dans un environnement mobile [Forman 1994] [Satyanarayanan 1996] :

1. *Un débit de communication faible et variable* : malgré l'évolution continue des technologies des réseaux sans fil, ils restent moins performants en termes de débit que les réseaux filaires. De plus, la puissance du signal reçu par l'unité mobile diminue lorsqu'elle s'éloigne de sa station de base, et ainsi le débit se réduit en conséquence.
2. *Une fréquente déconnexion* : la mobilité impose l'alternance entre deux modes d'exécution d'une unité mobile : le mode connecté au réseau (on-line) et le mode déconnecté du réseau (off-line). La déconnexion du réseau peut être volontaire ou involontaire. La déconnexion volontaire est déclenchée par l'utilisateur pour préserver la batterie, le coût de la communication, etc. Tandis que la déconnexion involontaire est le résultat d'une sortie de la zone de couverture, d'un épuisement de la batterie, d'une faiblesse de la bande passante, etc.
3. *Des ressources limitées* : pour des raisons de taille et de poids imposés par la portabilité, les terminaux mobiles disposent de ressources moins importantes par rapport à celles

qu'offrent des stations fixes (traitement, stockage, interface utilisateur, etc.). De plus, l'utilisation de ces ressources est limitée dans le temps puisqu'elle dépend d'une source d'énergie limitée, la batterie.

4. *Un réseau hétérogène et dynamique* : le réseau mobile avec infrastructure est constitué des unités fixes et d'autres mobiles. Les unités fixes possèdent relativement plus de ressources que les unités mobiles qui eux-mêmes se diffèrent en termes de capacité selon leur type (téléphone mobile, PDA, PC portable, etc.). Par ailleurs, la mobilité libre d'un terminal mobile d'une cellule à une autre cellule voisine et l'alternance entre les modes d'exécution connecté et déconnecté définissent un dynamisme dans cet environnement mobile.
5. *Une vulnérabilité des terminaux mobiles* : étant basés sur les communications sans fil, les terminaux mobiles sont plus sensibles aux attaques qui menacent les données transmises. De plus, leur mobilité augmente le risque de dommages physiques (pannes), l'accès non autorisé, la perte et le vol.

1.3.3 Le handover (handoff)

Dans un réseau cellulaire, la station de base constitue le point d'accès de toutes les communications. Elle joue le rôle de serveur pour les clients se trouvant dans sa cellule. Lorsqu'un utilisateur mobile quitte sa cellule pour entrer dans une autre, il se voit contraint de changer de point d'accès pour poursuivre sa communication. Cet utilisateur effectue alors ce que l'on appelle un *transfert intercellulaire*, ou *handover* ou encore *handoff*, qui consiste à demander à un gestionnaire du réseau, ou commutateur, de mettre en place la signalisation nécessaire au transfert [Agha 2001].

Selon le scénario de mobilité, on distingue deux classes de handovers : un handover horizontal et un handover vertical. Le handover horizontal est effectué entre deux cellules de la même technologie dans un réseau homogène. Dans ce cas, le handover est déclenché quand le point d'accès devient indisponible due au mouvement du terminal mobile (raisons de connectivité). Par contre, le handover vertical se déclenche entre deux cellules de deux technologies différentes dans un réseau hétérogène. De plus, il est initié pour des raisons de qualité de service et des préférences des utilisateurs [Kassar 2008].

Il exite différents types de handovers [Montavont 2001] [Komarova 2008]. La configuration du nœud mobile et le nombre des interfaces réseaux avec lesquelles il est équipé déterminent deux types de handover :

- Hard handover (dur) : utilise la technologie *break before make*. Dans ce cas, le terminal mobile est connecté à un seul point d'accès à un moment donné.
- Soft handover (doux) : utilise la technologie *make before break*. Ainsi, le terminal mobile est connecté à deux points d'accès au même temps.

Selon la métrique de performance à améliorer (la latence d'exécution, le nombre de paquets perdus ou la charge de la signalisation), le handover peut être :

- Fast handover (rapide) : est un handover qui a pour but principal de minimiser la latence d'exécution (les délais), sans conditions sur le nombre de paquets perdus.
- Smooth handover (souple) : est un handover qui a pour but principal de minimiser la perte de paquets, sans conditions sur le délai de transmission des paquets.
- Seamless handover (transparent) : est un handover où il n'y a pas de changement dans la capacité, la sécurité ou la qualité du service (aucune dégradation de service perceptible). Ainsi, il minimise à la fois la latence d'exécution et le nombre de paquets perdus.

Le processus du handover peut être divisé en trois étapes : initiation, décision et exécution. La phase d'initiation du handover est utilisée pour collecter les informations requises pour détecter la nécessité d'un handover et par conséquent le déclencher [Kassar 2008]. Deux approches pour la détection d'un handover sont possibles : une approche ascendante basée sur des mesures dans des couches inférieures comme la qualité du signal et une autre descendante basée sur des mesures de qualité de service dans la couche application [Komarova 2008].

La phase de décision permet de sélectionner le nouveau point d'accès le plus approprié. Les critères de décision peuvent être liés au réseau (zone de couverture, bande passante, latence, qualité du lien, niveau de sécurité, etc.), liés au terminal mobile (vitesse, puissance de la batterie, informations de localisation, etc.), liés à l'utilisateur (profil et préférences) ou liés au service (capacités de service, qualité de service, etc.) [Kassar 2008]. La politique traditionnelle de décision est basée uniquement sur la puissance du signal reçu (RSS : Received Signal Strength). Le choix d'un nouveau point d'accès est déterminé par des conditions spécifiques qui incluent d'autres paramètres : un seuil (threshold), une hystérésis (hysteresis), etc. [Pollini 1996]. Dans les réseaux hétérogènes, la politique de décision d'un handover vertical doit évaluer d'autres critères. La combinaison de tous ces critères et la dynamicité de certains d'entre eux va augmenter de manière significative la complexité du processus de décision. Les différentes stratégies proposées dans la littérature sont classifiées en cinq catégories : basée sur une fonction de décision (DF : Decision Function-based), centrée sur l'utilisateur (UC : User-Centric), à attributs multiples (MAD : Multiple Attribute Decision), basée sur la logique floue et les réseaux de neurones (FL/NN : Fuzzy Logic and Neural

Networks based), sensible au contexte (CA : Context-Aware).

L'exécution d'un handover exige un échange de signalisation pour rétablir la communication et ré-acheminer les paquets de données via le nouveau point d'accès. Les procédures qui constituent cette phase dépendent du type du handover effectué [Kassar 2008].

Le processus du handover peut être contrôlé soit par le terminal mobile ou par une entité du réseau (un seul routeur d'accès ou un groupe de points d'accès en collaboration) [Zdarsky 2004]. Dans le handover contrôlé par le réseau (NCHO : Network-Controlled HandOver), l'entité du réseau possède le contrôle total sur le handover. Dans le handover contrôlé par le terminal mobile (MCHO : Mobile-Controlled HandOver), le terminal mobile doit décider selon ses propres mesures. Lorsque des informations et des mesures du terminal mobile sont utilisées par l'entité du réseau pour décider, on parle d'un handover assisté par le terminal mobile (MAHO : Mobile-Assisted HandOver). Le handover assisté par le réseau (NAHO : Network-Assisted HandOver) se base sur des mesures passées par le réseau au terminal mobile qui les utilise pour décider.

Le standard IEEE 802.21 (MIH : Media Independent Handover) [Lampropoulos 2008] [Taniuchi 2009] consiste en l'élaboration d'une architecture qui facilite le handover et permet la continuité de service entre deux réseaux d'accès sans fil hétérogènes de manière transparente. La fonction de MIH (MIHF : MIH Function), représentant l'entité cœur de cette fonctionnalité, est une couche localisée à la fois dans la pile protocolaire de gestion de mobilité du nœud mobile et dans celles des entités du réseau, qui fournit trois types de services :

- Media Independent Event Service (MIES) : ce type de service est fourni des couches inférieures aux couches supérieures. Il est responsable de la détection des évènements des interfaces locales et distantes.
- Media Independent Command Service (MICS) : fournit des commandes concernant le handover aux couches supérieures pour contrôler les couches inférieures.
- Media Independent Information Service (MIIS) : fournit un mécanisme pour la recherche des informations utiles pour la phase de décision du handover.

Le standard IEEE 802.21 ne fait qu'aider les couches supérieures pour initialiser et préparer les handovers, et ne gère en aucun cas l'exécution de ceux-ci, ni définit leur politique de décision.

1.3.4 Gestion de la mobilité : IP Mobile

Le protocole IP Mobile [Perkins 2002] est basé sur la distinction de l'identification du nœud mobile et son attachement physique à un réseau d'accès IP. Dans cette optique, le

nœud mobile possède deux adresses : son adresse permanente (*Home Address*) liée à son réseau mère ou d'abonnement (*Home Network*) est utilisée pour identifier les connexions des couches les plus élevées (par exemple TCP, UDP), tandis qu'une adresse temporaire (*COA : Care-Of-Address*) liée au réseau visité (*Foreign Network*) permet de router les paquets jusqu'à lui via son point d'accès actuel. L'adresse temporaire change chaque fois que le nœud mobile change de sous-réseau. IP Mobile définit deux entités réseau chargées de gérer la mobilité (agents de mobilité) :

- Agent mère (HA : Home Agent) : routeur dans le réseau d'abonnement du nœud mobile. Il met à jour une information de localisation du nœud mobile et envoie par un tunnel les paquets destinés à ce dernier, lorsqu'il est en dehors de son réseau d'abonnement.
- Agent visité (FA : Foreign Agent) : routeur situé dans le réseau visité par le nœud mobile. Il coopère avec l'agent mère pour décapsuler et relayer les paquets destinés au nœud mobile visiteur.

Le processus de gestion de la mobilité d'IP mobile passe par trois étapes [Akyildiz 2004] :

- Découverte des agents de mobilité et obtention d'une adresse temporaire (COA) pour les nœuds mobiles permettant leur localisation.
- Enregistrement de cette adresse auprès de l'agent mère (HA) qui l'associe avec l'adresse permanente du nœud mobile et établit un tunnel vers celui-ci.
- Encapsulation des paquets arrivant au réseau d'abonnement et destinés au nœud mobile via le tunnel.

À chaque changement de point d'accès (handoff) le nœud mobile détecte son déplacement dans le nouveau sous-réseau, reçoit une COA et doit s'enregistrer auprès de son agent mère (HA). Durant cette période, soit les paquets sont perdus parce que le lien entre le précédent point d'accès et le nœud mobile n'est plus valide, soit les paquets sont transmis par le point d'accès précédent au nouveau point d'accès (smooth handoff) comme proposé dans [Cáceres 1996].

Le routage des paquets vers la nouvelle localisation du nœud mobile est peu optimal du fait du passage des paquets par le réseau d'abonnement du nœud mobile (le problème du routage triangulaire). Pour réduire la signalisation engendrée par IP Mobile, une optimisation de la route est proposée dans [Perkins 2001].

Dans un réseau micro-cellulaire et pico-cellulaire où chaque point d'accès couvre une zone géographique relativement petite, les nœuds mobiles exécutent très souvent des Handoffs IP Mobile. Ceci engendre une énorme charge de signalisation et une augmentation de la latence, ce qui implique une dégradation des performances du réseau et de la qualité du

service. Pour pallier ces problèmes, la gestion de la mobilité est divisée en macro-mobilité et micro-mobilité. La macro-mobilité désigne le déplacement inter-domaine, c'est-à-dire d'un domaine administratif IP à un autre (par exemple, entre réseaux d'accès sans fil étendus) par contre la micro-mobilité représente le déplacement intra-domaine, c'est-à-dire à l'intérieur d'un même domaine administratif IP (par exemple, entre points d'accès d'un réseau d'accès sans fil). IP Mobile est bien adapté pour la gestion de la macro-mobilité. Pour faire face aux problèmes liés à la micro-mobilité, plusieurs protocoles ont été proposés. Ils peuvent être classés en trois catégories [Badis 2006] :

- Protocoles basés sur une hiérarchie des agents [Gustafsson 2004] : dans ces protocoles, le domaine visité est structuré en une hiérarchie d'agents. Ils sont caractérisés par l'utilisation de l'encapsulation IP pour la communication entre les différentes parties de la hiérarchie. Le nœud mobile utilise une care-of-address locale au domaine visité.
- Protocoles basés sur un marquage de route : ces protocoles utilisent un acheminement IP par marquage de route dans le réseau d'accès. L'établissement de la route se fait par la transmission de proche en proche de paquets spécifiques d'un nœud mobile vers la passerelle. Cette transmission permet aux nœuds intermédiaires de retenir le chemin de manière à l'utiliser en sens inverse pour joindre le nœud mobile. En général, le nœud mobile garde sa Care-of-Address enregistrée auprès de son Home Agent. Parmi ces protocoles on trouve le IP cellulaire [Campbell 2000], HAWAII [Ramjee 1999] et TeleMIP [Das 2000].
- Protocoles basés sur les réseaux MANET (Mobile Ad hoc NETwork) : Ces protocoles sont conçus pour les réseaux d'accès ad hoc, où les hôtes et les routeurs sont mobiles. Le routage est multi-sauts et s'adapte au fur et à mesure que le nœud mobile se déplace et que la topologie change.

1.4 Technologies de réseaux sans fil

1.4.1 Le WiFi (IEEE 802.11)

L'IEEE a initié la spécification 802.11, norme régissant les réseaux locaux sans fil, en 1990 et l'avait finalisée en 1997. Le nom WiFi (Wireless Fidelity), correspond initialement au nom donné à la certification délivrée par la WECA (Wireless Ethernet Compatibility Alliance), l'organisme chargé de maintenir l'interopérabilité entre les matériels répondant à la norme 802.11.

1.4.1.1 Les modes de fonctionnement de 802.11

La norme IEEE 802.11 définit deux modes de fonctionnement : le mode infrastructure et le mode ad-hoc [Sayah 2009].

- **Mode infrastructure :** en mode infrastructure, le réseau sans fil consiste au minimum en un point d'accès (AP : Access Point) connecté éventuellement à l'infrastructure du réseau filaire et un ensemble de stations. L'ensemble formé par le point d'accès et les stations situées dans sa zone de couverture est appelé ensemble de services de base (BSS : Basic Service Set) et constitue une cellule. Un ensemble de services étendu (ESS : Extended Service Set) est un ensemble d'au moins deux BSS formant un seul sous-réseau, et reliés entre eux par une liaison appelée système de distribution (DS : Distribution System).
- **Mode ad hoc :** le mode ad-hoc, point-à-point, ou ensemble de services de base indépendants IBSS (Independent Basic Service Set), représente simplement un ensemble de stations qui communiquent directement entre elles sans point d'accès ni connexion à un réseau filaire. Un IBSS est constitué d'un seul BSS.

1.4.1.2 La pile protocolaire

Comme tous les standards IEEE 802, le standard 802.11 se concentre sur les deux couches inférieures du modèle IEEE, la couche physique (PHY) et la couche MAC (Medium Access Control). La figure 1.6 illustre ces deux couches.

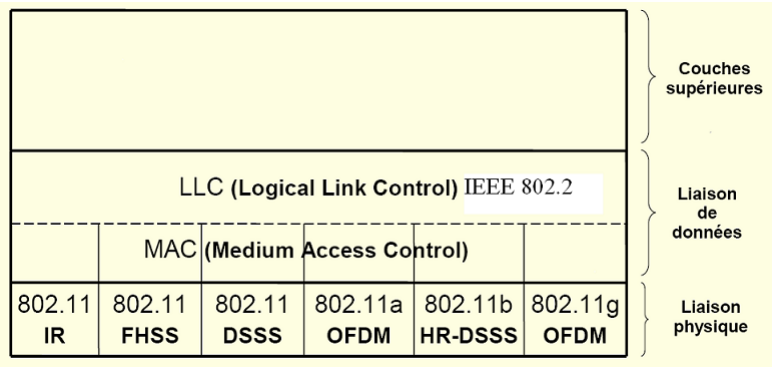

FIGURE 1.6 – Les couches PHY et MAC de 802.11.

Initialement, le standard IEEE 802.11 définit trois couches physiques avec un débit de 1 ou 2 Mbps : l'une sur infrarouge (IR) et les deux autres sur les ondes radio en utilisant les deux techniques d'étalement du spectre, par saut de fréquence (FHSS) et par séquence directe (DSSS). Au fil des années, des améliorations importantes ont été apportées au standard

802.11. Les principales améliorations concernant les couches physiques sont [Pujolle 2007] [Géron 2009] :

- La norme 802.11b utilise une technique d'étalement de spectre à haut débit par séquence directe (HR-DSSS) pour offrir un débit de 11 Mbps.
- Les normes 802.11a et 802.11g offrent un débit allant jusqu'à 54 Mbps en utilisant la modulation OFDM.
- La norme 802.11n permet d'atteindre un débit de 540 Mbps en utilisant la technologie multi-antennes MIMO. Elle est compatible avec les normes 802.11a et 802.11b/g.

La couche MAC de 802.11 permet à plusieurs utilisateurs d'exister sur un support partagé (accès multiple). Elle définit deux méthodes d'accès différentes [Dhoutaut 2003] :

- **DCF (Distributed Coordination Function) :** cette technique peut être utilisée par toutes les stations pour permettre un accès équitable au canal radio sans aucune centralisation de la gestion de l'accès (totalement distribuée). Elle est utilisée en mode ad-hoc ou infrastructure pour la transmission de données asynchrones.
- **PCF (Point Coordination Function) :** est une méthode dans laquelle les points d'accès prendre en charge la gestion de l'accès au canal dans leur zone de couverture pour les stations qui leur sont rattachées. Elle est utilisée seulement en mode infrastructure pour implémenter des services temps réel, comme la transmission de voix ou de vidéo.

1.4.1.3 Les évolutions de 802.11

Des extensions ont été apportées à la norme originale 802.11 afin de lui ajouter des améliorations et des modes de fonctionnement plus performants. Voici les principales extensions et leurs significations [Van Den Bossche 2007] [Sayah 2009] :

- 802.11b : propose un débit de 11 Mbps dans la bande des 2.4 GHz avec 3 canaux radio disponibles et une portée pouvant aller jusqu'à 300 mètres dans un environnement dégagé.
- 802.11g : offre un débit de 54 Mb/s sur la bande de fréquence des 2.4 GHz et elle a une compatibilité ascendante avec la norme 802.11b.
- 802.11a : permet d'obtenir un débit de 54 Mbps dans la bande des 5 GHz et spécifie 8 canaux radio.
- 802.11e : vise à améliorer la qualité de service au niveau de la couche liaison de données.
- 802.11f : désigne le protocole IAPP (Inter-Access Point Protocol) comme protocole de référence permettant à un utilisateur itinérant de changer de point d'accès d'une façon

transparente lors d'un déplacement.

- 802.11i : son but est d'améliorer la sécurité des transmissions (gestion et distribution des clés, chiffrement et authentification) pour les technologies 802.11a, 802.11b et 802.11g.
- 802.11n : permet d'atteindre un débit de 540 Mbps dans les deux bandes des 2.4 GHz et 5 GHz, grâce à l'utilisation conjointe des techniques MIMO et OFDM.

1.4.2 Le WiMAX (IEEE 802.16)

Le réseau WiMAX (Worldwide Interoperability for Microwaves Access) désigne dans le langage courant un ensemble de standards et techniques du monde des réseaux métropolitains sans fil (WMAN). Le standard IEEE 802.16, ou WiMAX permet le raccordement sans fil d'entreprises ou de particuliers sur de longues distances à haut débit. Cette technologie vise donc à introduire une solution complémentaire au DSL (Digital Subscriber Line) et aux réseaux câblés d'une part, et à interconnecter des hotspots WiFi d'autre part. WiMAX est principalement fondé sur une topologie en étoile bien que la topologie maillée soit possible. La communication peut être réalisée en ligne de vue (LOS : Line Of Sight) ou non (NLOS) [Salhani 2008].

1.4.2.1 Standard de l'interface air IEEE 802.16

Plusieurs standards IEEE 802.16 ont été définis : IEEE 802.16 (2001) dédié aux systèmes LOS fonctionnant sur la bande des 10 à 66 GHz, IEEE 802.16c (2002) introduit des profils système WiMAX, IEEE 802.16a pour les systèmes NLOS sur la bande des 2 à 11 GHz et IEEE 802.16d (2004) permet de réviser les standards antérieurs pour les rassembler en un seul. Les deux derniers standards retenus sont IEEE 802.16 (2004) pour les terminaux fixes (WiMAX fixe) et IEEE 802.16e (2005) pour les terminaux mobiles (WiMAX mobile). La dernière mouture du standard est la version IEEE 802.16m (2011) qui permet de qualifier WiMAX comme une technologie candidate à la quatrième génération (4G) [Salhani 2008] [Pujolle 2007].

1.4.2.2 Pile protocolaire

La pile protocolaire du standard IEEE 802.16 est focalisée sur les couches PHY et MAC comme on peut le voir sur la figure 1.7. Elle contient une couche physique et trois sous-couches MAC.

La couche physique de WiMAX utilise différentes techniques radio : une seule porteuse

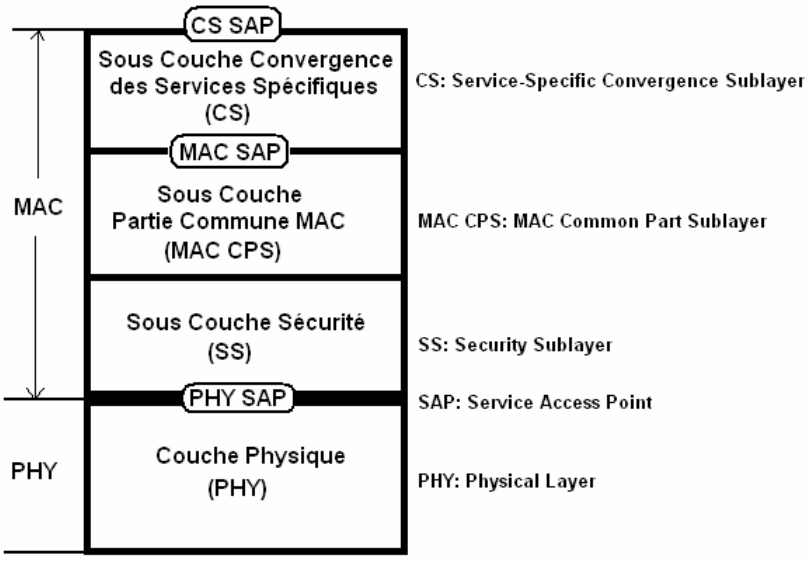

FIGURE 1.7 – Pile protocolaire de WiMAX [Bchini 2010].

SC (Single Carrier), OFDM (Orthogonal Frequency Division Multiplexing) divise le canal en plusieurs sous-canaux alloués à un seul utilisateur à un instant donné, OFDMA (Orthogonal Frequency Division Multiple Access) partage les sous-canaux entre plusieurs utilisateurs et S-OFDMA (Scalable OFDMA) permet le passage à l'échelle en utilisant de façon adaptative la bande passante de 1,25 à 20 MHz. Pour gérer le partage des porteuses sur les voies montantes et descendantes, deux techniques de multiplexage sont utilisées : TDD (Time Division Duplex) et FDD (Frequency Division Duplex). WiMAX utilise d'autres fonctionnalités avancées de la couche physique : la modulation adaptative AMC (Adaptive Modulation and Coding) en fonction de la qualité du signal, la technique multi-antennes MIMO, etc. [Gray 2006].

La couche MAC est composée de trois sous-couches [Bchini 2010] :

- **Sous-couche de convergence spécifique au service** (Service-Specific Convergence Sublayer) : joue le rôle d'interface avec les couches supérieures ou bien avec les systèmes externes. Elle a entre autre la charge de classer les paquets selon leur provenance et leur destination afin de les répartir sur la bonne connexion MAC.
- **Sous-couche de partie commune MAC** (MAC Common Part Sublayer) : contient les fonctions clés de la couche MAC (gestion de l'accès au système, allocation des ressources, établissement et maintenance des connexions, contrôle de la qualité de service, etc.).
- **Sous-couche de sécurité** (Security Sublayer) : fournit des mécanismes de sécurité pour l'authentification, le chiffrement, l'échange des clés, etc.

WiMAX mobile utilise efficacement les bandes de fréquences disponibles. Il met en

place un certain nombre de fonctionnalités permettant d'offrir une solution pour le passage à l'échelle, et un accès haut débit tout en tenant compte des besoins de l'utilisateur et des applications en terme de qualité de service (QoS), de mobilité et de sécurité [Gray 2006].

1.5 Réseaux de mobiles

1.5.1 Définition et principes

Un réseau de mobiles fait partie de la famille des réseaux cellulaires, il peut se définir par la fourniture à l'utilisateur d'au moins un des deux services caractéristiques de la mobilité : lui permettre de se déplacer à travers le réseau en conservant une même adresse et lui proposer un accès sans fil à l'information. Cette diversité n'est possible que par l'entremise d'une architecture spécifique, comportant des éléments originaux au regard des réseaux fixes [Agha 2001].

Le concept cellulaire est introduit dans les années 1970 par les Bell Labs. Les systèmes cellulaires sont conçus pour augmenter la mobilité des terminaux. Dans tels systèmes, le territoire couvert, ou la zone de couverture, est découpé en petites surfaces géographiquement limitées sous forme hexagonale appelées cellules. Ces cellules se superposent partiellement pour assurer une couverture complète du territoire. Ainsi, le terminal mobile peut changer de cellule sans coupure de la communication (handover).

Chaque cellule dispose d'une station de base ou BTS (Base Transceiver Station) qui assure la couverture radio. Les stations de base sont reliées à des contrôleurs de station de base ou BSC (Base Station Controller) qui gèrent les ressources radio. Le BSC et l'ensemble des BTS qui lui sont raccordés constituent un sous-système radio ou BSS (Base Station Subsystem). Les BSC sont tous raccordés à des commutateurs du service mobile appelés MSC (Mobile services Switching Center). Les MSC gèrent la mobilité en utilisant deux bases de données pour la localisation des utilisateurs : le registre de localisation nominale HLR (Home Location Register) qui contient les données de l'abonné et le registre de localisation des visiteurs VLR (Visitor Location Register) qui gère le client dans la cellule où il se trouve [Pierre 2003] [Pujolle 2007].

1.5.2 Générations de réseaux de mobiles

Les trois premières générations de réseaux de mobiles se distinguent par la nature de la communication transportée. Dans la première génération, la communication est analogique. La communication dans la deuxième génération est numérique sous forme de circuit. La

troisième génération intègre des applications multimédias sous forme de paquet. Enfin, la quatrième génération permet un débit de plus de 10 Mbps avec la possibilité de se connecter à plusieurs réseaux simultanément [Pujolle 2007].

1.5.2.1 La première génération (1G)

Les systèmes cellulaires de première génération sont caractérisés par des terminaux analogiques dotés d'une mobilité restreinte et de services limités. Ils ont été les premiers à permettre à un utilisateur mobile d'utiliser un téléphone de façon continue, n'importe où dans la zone de service d'un opérateur. Le succès de cette génération est resté très faible en raison du coût des équipements qui n'ont pas connu de miniaturisation. Les plus importants systèmes sont AMPS (Advanced Mobile Phone System), NMT (Nordic Mobile Telephone), et TACS (Total Access Communication System).

1.5.2.2 La deuxième génération (2G)

Les réseaux cellulaires de deuxième génération, tels IS-95 et GSM, sont caractérisés par l'introduction de la technologie numérique. Ils favorisent la mise au point d'un terminal portable à un coût raisonnable et doté d'une autonomie acceptable grâce aux progrès réalisés dans la technologie des composants. Plusieurs technologies numériques cellulaires sont apparues au débit des années 1990 : le GSM (Global System for Mobile communications) en Europe, le D-AMPS une version numérique de l'AMPS aux Etats-Unis et le PDC (Pacific Digital Cellular) au Japon.

1.5.2.3 La deuxième génération et demie (2,5G)

La mise en place de la troisième génération va demander un laps de temps assez long, de l'ordre d'une dizaine d'années. Les raisons à cela sont d'une part l'installation d'une infrastructure totalement nouvelle et d'autre part un manque de capitaux de la part des opérateurs de mobiles. Ce laps de temps est mis à profit pour améliorer la deuxième génération.

Les réseaux de 2,5G se caractérisent par un double réseau cœur, un réseau circuit pour le transfert de la voix et un réseau paquet pour le transfert des données. Par ailleurs, les terminaux sont capables de gérer à la fois les voies téléphoniques, comme dans le GSM et les voies de données, beaucoup plus sporadiques. En Europe, l'amélioration de la technique GSM est représentée par le GPRS (General Packet Radio Service).

1.5.2.4 La troisième génération (3G)

Les systèmes de mobiles de troisième génération se présentent comme des concurrents des infrastructures de deuxième génération déjà déployées. La troisième génération améliore la précédente par une qualité du service rendue au moins comparable à celle fournie par les réseaux fixes. De plus, les normes de 3G, comme l'UMTS, cherchent à fournir de nouvelles avancées significatives incluant l'itinérance mondiale, une large gamme de services, à haut débit ou non, des services audiovisuels et l'utilisation d'un seul terminal dans différents environnements radio. Les services seront disponibles dans une variété d'environnements dans lesquelles se pourra trouver l'utilisateur, dans les bureaux, à l'intérieur ou à l'extérieur des zones urbaines denses, dans les zones reculées, suburbaines et rurales.

1.5.2.5 Les générations 3,5G et 4G

La 3,5G correspond aux hauts débits de données, c'est-à-dire de plus de 1 Mbps. Cette valeur est obtenue par la technologie HSDPA dans le sens descendant et par son successeur HSUPA dans le sens montant. Le HSDPA (High-Speed Downlink Packet Access) est un protocole pour la téléphonie mobile parfois appelé 3G+ qui offre des performances approximativement dix fois supérieures à la 3G. Le HSUPA (High-Speed Uplink Packet Access) s'intéresse à la voix montante qui devrait atteindre à terme 5,76 Mbps.

Pour entrer dans la quatrième génération, les débits doivent dépasser la dizaine de mégabits par seconde. La technologie HSOPA (High-Speed OFDM Packet Access) appelée parfois le super 3G marque l'entrée dans la 4G. L'interface radio est totalement modifiée pour passer à l'OFDMA qui est incompatible avec les deux techniques précédentes (changement de génération). Les débits sont de 100 Mbps dans le sens descendant et 50 Mbps dans le sens montant.

La 4G permet aussi d'utiliser plusieurs technologies simultanément, afin de former un réseau multi-homés possédant plusieurs réseaux de base. Les applications s'exécutant sur un terminal peuvent ainsi choisir le meilleur réseau par rapport à leurs contraintes de qualité de service, de sécurité, de disponibilité et de gestion de la mobilité. Les deux normes LTE (Long Term Evolution) et UMB (Ultra Mobile Broadband) sont des technologies candidates à la 4G.

1.5.3 Technologies de réseaux de mobiles

Dans cette section, nous allons présenter les évolutions des réseaux de mobiles à travers trois technologies complémentaires : le GSM, le GPRS et l'UMTS. La figure 1.8 illustre les architectures de ces réseaux [Girodon 2002].

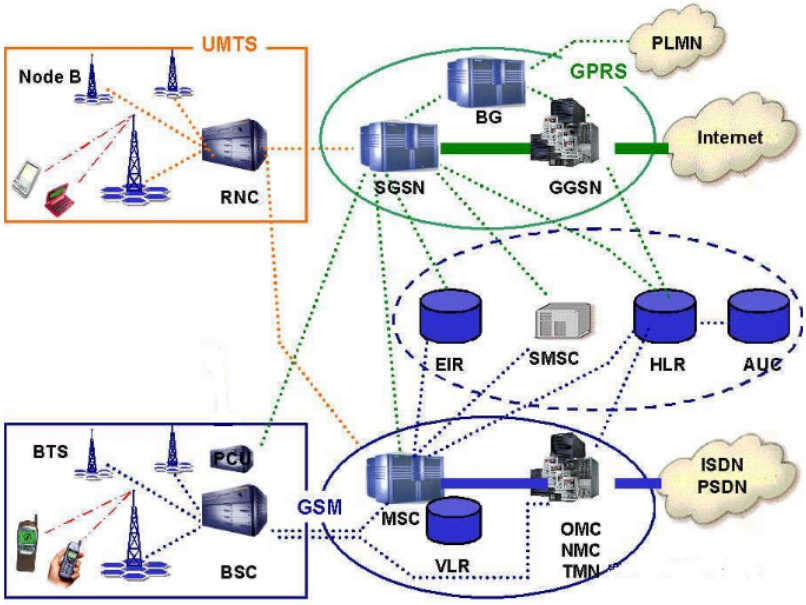

FIGURE 1.8 – Réseaux GSM, GPRS et UMTS [Girodon 2002].

1.5.3.1 Le GSM

Le GSM (Global System for Mobile communications) est le système de téléphonie mobile de deuxième génération le plus populaire. Il fournit des services de type voix en mode circuit. L'architecture d'un réseau GSM est composée de trois sous-systèmes [Demoulin 2004] :

- Le sous-système radio : gère la transmission radio. Il est constitué de plusieurs entités fonctionnelles, la station mobile, la station de base (BTS) et le contrôleur de station de base (BSC).
- Le sous-système réseau : comprend l'ensemble des fonctions nécessaires pour le contrôle des appels, la gestion de la mobilité et la commutation. Il est constitué des entités fonctionnelles suivantes : le centre de commutation de service mobile (MSC), le registre de localisation des visiteurs (VLR), le registre de localisation nominale (HLR), le registre d'identité des équipements (EIR) et le centre d'authentification (AuC).
- Le sous-système d'exploitation et de maintenance : regroupe trois activités principales de gestion : la gestion administrative, la gestion commerciale et la gestion technique. Il consiste principalement en des centres d'opération et de maintenance (OMC).

1.5.3.2 Le GPRS

L'évolution du réseau GSM vers le réseau UMTS passe par une étape importante : le réseau GPRS (General Packet Radio Service). Le GPRS introduit des services de commutation de paquets dans le réseau cœur du GSM pour permettre un accès direct à des réseaux de

données en mode paquets (PDNs). Pour assurer le transfert de paquets, des nouvelles entités fonctionnelles sont ajoutées à l'infrastructure du réseau GSM [De Vriendt 2002] :

- L'unité de contrôle de paquets (PCU : Packet Control Unit) : est situé dans le sous-système radio de GSM. Il gère la segmentation du paquet, l'accès au canal radio, la retransmission automatique et le contrôle de puissance.
- Le nœud de service (SGSN : Serving GPRS Support Node) : se trouve au même niveau hiérarchique que le MSC/VLR. Il effectue les fonctions de routage, de gestion de mobilité et de contrôle d'accès aux services de données.
- Le nœud de passerelle (GGSN : Gateway GPRS Support Node) : joue le rôle de routeur acheminant le trafic vers des réseaux de données GPRS ou externes.

1.5.3.3 L'UMTS

L'UMTS (Universal Mobile Telecommunications System) est un système de télécommunications mobiles de troisième génération, capable d'offrir au grand public des services de type multimédia à débit élevé. L'architecture d'un réseau UMTS est composée de trois domaines : l'équipement utilisateur, le réseau d'accès radio ou l'UTRAN et le réseau cœur [Pujolle 2007].

- L'équipement utilisateur : se compose d'un terminal capable de gérer l'interface radio et d'une carte à puce (USIM : UMTS Subscriber Identity Module) qui contient les caractéristiques de l'utilisateur et de son abonnement.
- Le réseau d'accès radio (UTRAN : UMTS Terrestrial Radio Access Network) : regroupe les stations de base ou les Node B qui correspondent aux BTS en GSM et les contrôleurs de stations de base ou RNC (Radio Network Controller) correspondants aux BSC en GSM.
- Le réseau cœur : est constitué d'un réseau cœur de type circuit et d'un réseau cœur de type paquet. Le réseau cœur orienté circuit est composé, à l'image de celui du GSM, de commutateurs circuits. Le réseau cœur orienté paquet est semblable à celui du GPRS. Il est composé de commutateurs paquets reliés par un réseau IP.

1.6 Conclusion

À travers ce chapitre, nous avons tiré les concepts de base sur les réseaux sans fil et mobiles. Nous avons étudié en particulier les réseaux mobiles avec infrastructure en présentant leur architecture ainsi que les nouvelles contraintes engendrées par la mobilité. Nous avons vu les deux opérations qui sont au cœur de fonctionnement de tel environnement, à savoir la gestion

des handovers et la gestion de la mobilité. Ensuite, nous avons présenté deux technologies de réseaux sans fil : WiFi et WiMAX. Nous avons présenté dans la dernière partie, les différentes générations des réseaux de mobiles et les architectures de leurs technologies GSM, GPRS et UMTS.

Nous avons pu constater qu'il existe deux catégories de réseaux mobiles avec infrastructure. Il s'agit d'une part de réseaux de mobiles provenant du monde des télécommunications et essentiellement dédiés à la téléphonie (2G) puis plus orientés vers le multimédia et le transfert de données (3G). D'autre part, des réseaux sans fil conçus au départ pour l'échange de données puis étendus pour les applications temps réel (parole, vidéo, etc.) sont déployés au travers de standards phares tels que WiFi, WiMAX, etc. Les réseaux 4G vise la convergence entre ces deux catégories en permettant l'intégration des diverses technologies et leur interopérabilité.

Les applications distribuées ciblant des terminaux mobiles s'appuient le plus souvent sur des réseaux mobiles avec infrastructure. Dans un tel contexte mobile, la tolérance aux fautes est un besoin qui doit être pris en compte. Dans le chapitre suivant, nous allons présenter la notion de sûreté de fonctionnement d'un système informatique, les différentes techniques de tolérance aux fautes ainsi que les techniques spécifiques aux applications distribuées.

CHAPITRE 2

Sûreté de fonctionnement et tolérance aux fautes

Sommaire

2.1 Introduction

La sûreté de fonctionnement informatique est centrée sur la notion de faute, comme cause potentielle de mauvais fonctionnement d'un système informatique, et propose divers moyens pour y faire face. Parmi ceux-ci, la tolérance aux fautes représente le moyen le plus utilisé. Elle repose sur deux principes : la redondance et la diversité. La redondance consiste à multiplier les sources d'informations du système. La diversité consiste à s'assurer de l'indépendance des défaillances des services dupliqués.

L'objectif de ce chapitre est de présenter le vocabulaire relatif à la sûreté de fonctionnement et d'introduire la tolérance aux fautes. Nous allons commencer par la présentation de la notion de sûreté de fonctionnement, ses attributs, ses entraves, et les moyens pour l'assurer. Ensuite, nous allons étudier en détails la tolérance aux fautes et les techniques pour sa mise en œuvre, ainsi que les approches pour leur validation. Enfin, nous allons aborder les techniques de tolérance aux fautes spécifiques aux systèmes répartis.

2.2 Sûreté de fonctionnement

2.2.1 Définition

La sûreté de fonctionnement d'un système informatique est la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans la qualité du service qu'il leur délivre. Le service délivré par un système est le comportement tel que perçu par ses utilisateurs. Un utilisateur est un autre système, éventuellement humain, qui interagit avec le système considéré. La sûreté de fonctionnement peut aussi être définie comme l'aptitude à éviter des défaillances du service plus fréquentes ou plus graves que ce qui est acceptable [Laprie 1996] [Avižienis 2004].

La sûreté de fonctionnement englobe trois différentes notions : les attributs, les entraves et les moyens. Celles-ci peuvent être résumées par l'arbre de la sûreté de fonctionnement, comme indiqué par la figure 2.1 [Arlat 2006].

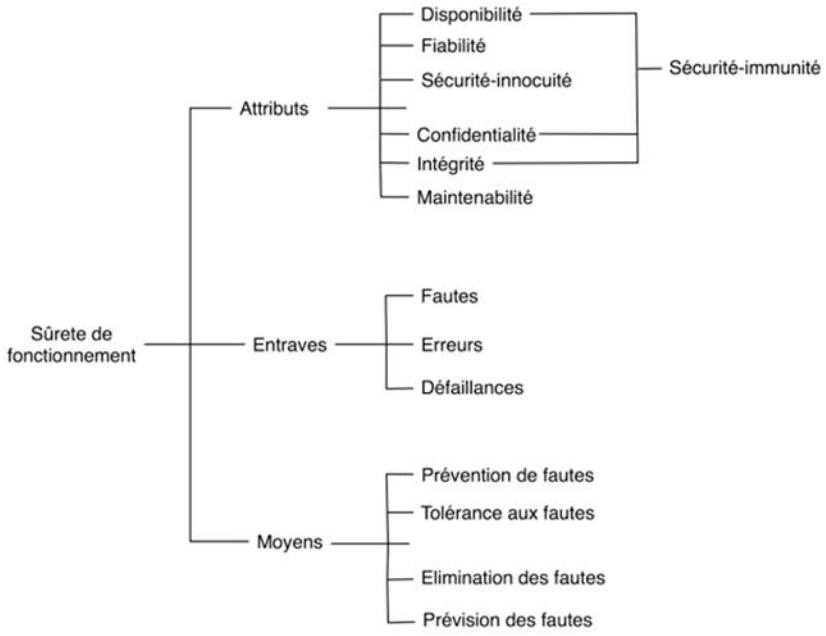

FIGURE 2.1 – L'arbre de la sûreté de fonctionnement [Laarouchi 2009].

2.2.2 Attributs de la sûreté de fonctionnement

La sûreté de fonctionnement d'un système peut être perçue selon des propriétés différentes, mais complémentaires. Ces propriétés sont appelées les *attributs* de la sûreté de fonctionnement. Les attributs à considérer dépendent des applications auxquelles le systèmes est destiné. Les attributs pouvant être dégagés sont les suivants [Arlat 2006] [Laarouchi 2009] :

- **La disponibilité** (availability) : c'est la capacité du système à être prêt à l'utilisation.
- **La fiabilité** (reliability) : c'est la continuité du service. Elle est définie par une fonction qui représente la probabilité que le système délivre le service attendu pendant une durée donnée.
- **La sécurité-innocuité** (safety) : définie par la non-occurrence de conséquences catasirophiques pour l'environnement.
- **La confidentialité** (confidentiality) : c'est l'absence d'accès et de divulgations non autorisées de l'information.
- **L'intégrité** (integrity) : définie par la non-occurence d'altérations inappropriées de l'information.
- **La maintenabilité** (maintenability) : définie par l'aptitude du système aux réparations et aux évolutions.

L'association, à la confidentialité, de l'intégrité et de la disponibilité vis-à-vis des actions autorisées, conduit à la sécurité-immunité.

2.2.3 Entraves à la sûreté de fonctionnement

Les entraves sont les circonstances indésirables qui représentent les causes ou les résultats de la non-sûreté de fonctionnement du système. Nous en distinguons trois sortes : les défaillances, les erreurs et les fautes [Laarouchi 2009].

Mettre en œuvre la sûreté de fonctionnement d'un système consiste à empêcher sa défaillance. La *défaillance* (failure) d'un système survient lorsque le service qu'il délivre diffère du service attendu. Pour lutter contre ces défaillances, on doit déterminer leurs causes. La sûreté de fonctionnement introduit deux notions : les fautes et les erreurs [Avižienis 2004] [Taïani 2004].

Une *faute* (fault) est la cause d'une erreur. Elle représente un défaut d'un composant physique ou logiciel d'un système. L'activation d'une faute provoque la propagation d'erreurs dans le système. Une faute active peut être soit une faute interne activée par le processus de traitement et qui a été précédemment dormante, soit une faute externe. Une faute interne peut repasser de l'état actif à l'état dormant.

Une *erreur* (error) est la partie de l'état du système qui est susceptible d'entraîner une défaillance. Le système peut continuer à délivrer un service correct malgré la présence des erreurs qui affectent son état (un résultat incorrect ou imprécis). Une défaillance se produit lorsque l'erreur atteint l'interface du service fournit et le modifie. La détection d'une erreur se fait soit par l'analyse de l'état du système, soit par l'effet sur le service (défaillance).

La défaillance d'un système est la conséquence d'une erreur qui est elle-même la conséquence d'une faute activée. Ainsi, faute, erreur et défaillance forment une chaîne de causalité. Si un système est considéré comme un ensemble de sous-systèmes, la défaillance d'un sous-système peut provoquer la création et/ou l'activation d'une faute dans un autre sous-système ou dans le système lui-même. Donc, la chaîne de causalité : faure, erreur, défaillance devient récursive. La figure 2.2 illustre la relation entre faute, erreur et défaillance [Kalla 2004].

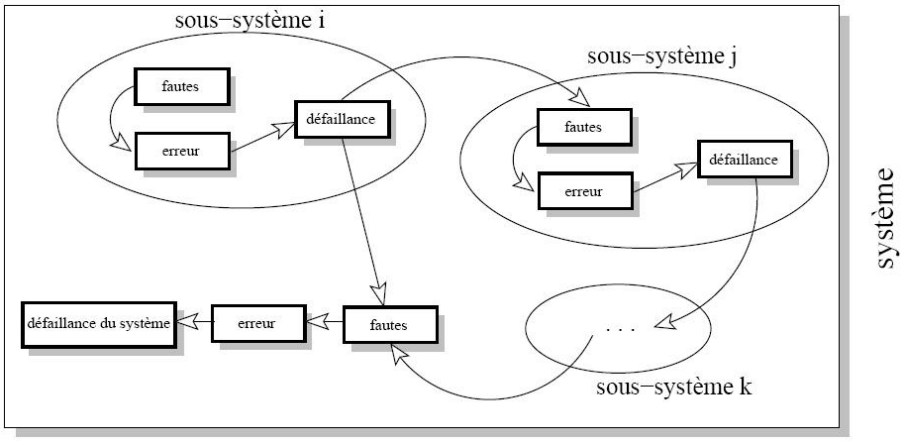

FIGURE 2.2 – Relation entre faute, erreur et défaillance [Kalla 2004].

2.2.4 Moyens pour assurer la sûreté de fonctionnement

Les moyens sont les méthodes et les techniques qui cherchent à rendre un système capable d'accomplir correctement sa fonction, ce qui donne une confiance dans cette aptitude à l'utilisateur. Ils sont classés en quatre moyens suivant l'objectif visé : [Avižienis 2004] [Besseron 2010] :

- **La prévention des fautes :** vise à empêcher l'apparition ou l'introduction des fautes dans le système. Elle repose sur des règles de développement.
- **L'élimination des fautes :** s'attache à réduire la présence (nombre, sévérité) des fautes. Cette méthode opère à la fois lors du développement ou lors de l'utilisation (maintenance).
- **La prévision des fautes :** cherche à estimer la présence des fautes courantes et futures et leurs conséquences. Elle est réalisée par la modélisation et l'évaluation de systèmes.
- **La tolérance aux fautes :** consiste à continuer de délivrer un service correct en dépit des fautes. Elle est mise en œuvre par la détection des erreurs et le rétablissement du système.

La prévention et l'élimination des fautes peuvent se regrouper sous le concept d'*évitement des fautes* : on cherche à concevoir un système exempt de fautes. La tolérance aux fautes et la prévision des fautes peuvent se regrouper sous le concept d'*acceptation des fautes* : il y a toujours des fautes qu'on ne peut pas éviter, on essaye de réduire la sévérité des défaillances qu'elles peuvent causer. Ces deux concepts sont des approches complémentaires pour concevoir un système sûr de fonctionnement [Lussier 2007].

La tolérance aux fautes est l'unique méthode qui peut être mise en place durant la vie opérationnelle du système. Nous allons la présenter en détails dans la section suivante.

2.3 Tolérance aux fautes

2.3.1 Définition

La tolérance aux fautes permet à un système de continuer à délivrer un service conforme à sa spécification en dépit de la présence ou de l'occurrence de fautes. Elle peut être obtenue par l'utilisation de solutions logicielles, matérielles ou une combinaison des deux. L'approche matérielle consiste à introduire un ensemble de redondances physiques dans le système, tels que des processeurs, des médias de communication et des capteurs. L'approche logicielle consiste à introduire un ensemble de redondances logicielles dans le système, tels que des voteurs logiciels. À la différence des solutions matérielles qui ne peuvent tolérer que des fautes physiques, les solutions logicielles peuvent être utilisées pour tolérer des fautes logicielles et matérielles [Kalla 2004].

2.3.2 Mécanismes de tolérance aux fautes

La tolérance aux fautes vise à éviter les défaillances et est mise en œuvre par la *détection d'erreur* et le *rétablissement du système* [Arlat 2006].

2.3.2.1 La détection d'erreur

La détection d'erreur permet au système d'identifier un état erroné au cours de son fonctionnement [Taïani 2004]. Les objectifs consistent à prévenir, si possible, l'occurrence d'une défaillance, à éviter la propagation de l'erreur et à faciliter l'identification ultérieure de la faute en vue de son élimination ou de sa prévention. La réussite d'une telle technique de détection des erreurs dépend de deux paramètres qui sont la *latence* (délai entre la production et la détection de l'erreur), et le *taux de couverture* (pourcentage d'erreurs détectées) [Jallouli 2009]. Les formes les plus courantes de détection d'erreur sont les suivantes [Arlat 2006] [Lussier 2007] :

- **Les codes détecteurs d'erreur :** ciblent plus particulièrement les erreurs induites par les fautes physiques. La détection est basée sur une redondance dans la représentation de l'information, soit en ajoutant des bits de contrôle aux données, soit en utilisant une nouvelle forme de représentation incluant la redondance.
- **La duplication et comparaison :** consiste à comparer les résultats fournis par au moins deux unités fournissant le même service et indépendantes vis-à-vis des fautes que l'on souhaite tolérer. On utilise la redondance des composants matériels pour des fautes physiques, et la diversification des composants logiciels pour des fautes de conception.
- **Le contrôle temporel par chien de garde :** est le moyen de détection le plus couramment utilisé pour la détection d'erreur en ligne. Il peut être utilisé pour détecter la défaillance d'un périphérique en vérifiant que son temps de réponse ne dépasse pas une valeur maximale (time-out) ou pour surveiller l'activité des processeurs.
- **Le contrôle de vraisemblance :** permet de détecter des erreurs induites par un large spectre de fautes, mais la couverture associée peut parfois être limitée. Il peut être mis en œuvre soit par du matériel pour détecter par exemple des adresses mémoire erronées, soit par du logiciel pour vérifier la conformité des entrées, des sorties ou des variables internes du système par rapport à des invariants.
- **Le contrôle de données structurées :** consiste à vérifier soit l'intégrité des données d'un point de vue sémantique, soit l'intégrité structurelle de la structure de données.

2.3.2.2 Le rétablissement du système

Le rétablissement du système peut être assuré par deux méthodes complémentaires : le *traitement d'erreur*, qui vise à corriger l'état erroné du système, et le *traitement de faute*, qui vise à empêcher la faute à l'origine de l'erreur d'être à nouveau activée [Lussier 2007].

A. Traitement d'erreur

On distingue trois techniques de traitement d'erreur : la reprise, la poursuite et la compensation d'erreur [Deswarte 1990] [Arlat 2006].

- **Reprise :** est la technique la plus couramment utilisée. Elle consiste en la sauvegarde périodique de l'état du système dans un point de reprise. Lorsqu'une erreur est détectée, le système est ramené dans un état antérieur à l'occurrence de l'erreur. La sauvegarde périodique de l'état du système doit s'effectuer au moyen d'un mécanisme de mémorisation, ou « support stable » qui protège les données contre les effets des fautes. La sauvegarde peut être facilitée par des mécanismes matériels ou logiciels permettant de

sauvegarder automatiquement les données modifiées entre deux points de reprise.

– **Poursuite :** constitue une approche alternative ou complémentaire à la reprise. Elle consiste à trouver un nouvel état acceptable à partir duquel le système pourra fonctionner (éventuellement en mode dégradé). En fonction de l'application, cela peut être fait par une réinitialisation du système et l'acquisition d'un nouveau contexte d'exécution auprès de l'environnement. Une autre approche est celle des traitements d'exceptions : les programmes d'application sont conçus pour prendre en compte des signaux d'erreur et passer d'un traitement normal à un traitement exceptionnel.
– **Compensation :** consiste à utiliser des redondances présentes dans le système pour fournir un service correct en dépit des erreurs. Avec la compensation, il n'est pas nécessaire de ré-exécuter une partie de l'application (reprise) ou d'exécuter une procédure dédiée (poursuite) pour permettre de continuer le traitement fonctionnel. Elle peut être soit consécutive à une détection d'erreur (détection et compensation), soit systématique (masquage).
  - **Détection et compensation d'erreur :** la compensation est déclenchée sur détection d'erreur. L'état du système est transformé par la commutation d'un composant défaillant vers un composant non-défaillant. Un exemple typique de la détection et compensation d'erreur est l'utilisation des composants auto-testables exécutant en redondance active le même traitement ; en cas de défaillance de l'un d'entre eux, il est déconnecté et le traitement se poursuit sans interruption sur les autres.
  - **Masquage d'erreur :** dans cette méthode la compensation est effectuée de manière systématique, sans détection préalable d'erreur. L'erreur est masquée et éventuellement détectée par la suite pour effectuer le traitement de faute. Un exemple typique est celui du vote majoritaire : les traitements sont exécutés par au moins trois composants identiques dont les sorties sont votées ; les résultats majoritaires sont transmis, les résultats minoritaires (supposés erronés) sont éliminés.

Parmi les avantages des techniques de reprise et de poursuite, il faut compter la faible redondance structurelle nécessaire, en particulier, si la détection repose sur des contrôles de vraisemblance. De plus, si de multiples contrôles de vraisemblance sont mis en œuvre, de larges classes de fautes peuvent être tolérées. Les principaux inconvénients de la reprise sont la taille des points de reprise et le surcoût temporel nécessaire à leur établissement. A cela s'ajoute, aussi bien pour la poursuite que pour la reprise, un surcoût spécifique pour le traitement d'erreur. De plus, les deux techniques imposent souvent des contraintes structurelles qui doivent être prises en compte dans le développement de l'application, avec un support

spécifique du système d'exploitation.

L'inconvénient majeur des techniques de compensation par rapport à celles de reprise et de poursuite est la redondance nécessairement plus élevée. Le premier avantage de ces techniques est que la durée du traitement d'erreur est plus faible. Dans le cas du masquage, cette durée est même constante qu'il y ait ou non une erreur. Ceci peut être très utile pour les systèmes temps-réels. Le second avantage est la transparence des mécanismes de traitement d'erreur vis-à-vis de l'application [Deswarte 1990] [Arlat 2006].

B. Traitement de faute

Le traitement de faute est composé de quatre phases successives [Avižienis 2004] [Lussier 2007] :

- **Le diagnostic :** cherche à identifier la faute responsable de l'état erroné du système en termes de localisation et de nature.
- **L'isolement (passivation) :** consiste à exclure la participation des composants fautifs dans la délivrance du service, par moyen physique ou logiciel. En d'autres termes, il empêche une nouvelle activation de la faute.
- **La reconfiguration :** est effectuée si le système ne peut plus délivrer le même service qu'auparavant. Elle cherche à compenser l'isolement du composant défaillant, soit en basculant vers des composants redondants, soit en réassignant ces tâches à d'autres composants.
- **La réinitialisation :** vérifie, met à jour et enregistre la nouvelle configuration, et met à jour les tables et enregistrements du système.

Les fautes intermittentes ne nécessitent ni passivation, ni reconfiguration ; identifier si une faute est intermittente ou non peut être effectué par le traitement d'erreur (la récurrence d'une erreur indique que la faute n'est pas intermittente) ou par le diagnostic de faute.

2.3.3 Validation des mécanismes de tolérance aux fautes

La conception et la réalisation des mécanismes de tolérance aux fautes sont des activités de développement aussi faillibles que les autres, et peuvent introduire de nouvelles fautes dans le système. Il est donc important de procéder à l'évaluation et la validation de ces mécanismes qui peuvent être réalisées suivant deux approches complémentaires : la *vérification formelle* et l'*injection de fautes* [Lussier 2007].

2.3.3.1 Vérification formelle

Les méthodes de vérification formelle consistent à utiliser des techniques formelles comme l'analyse statique, la preuve mathématique, ou l'analyse de comportement, pour obtenir des certitudes sur le comportement du système. Ces méthodes permettent également de valider le comportement d'un système en présence de fautes, en utilisant de modèles formels décrivant les fautes et leurs conséquences.

2.3.3.2 Injection de fautes

L'injection de fautes est une méthode de test ou d'évaluation des mécanismes de tolérance aux fautes, qui consiste à introduire des fautes dans un système ou dans un modèle du système, puis à observer son comportement. Les fautes physiques ou logicielles peuvent être simulées par altération du contenu des composants mémoires du système, ou par modification des entrées et des sorties d'un composant. L'injection de mutations est une technique spécifique à la simulation de fautes logicielles. On crée un ensemble de programmes appelés mutants qui diffèrent du programme original par une seule modification élémentaire syntaxiquement correcte. Ensuite, le comportement des mécanismes de tolérance aux fautes logicielles est observé en appliquant à ces mutants des entrées de test cherchant à activer la faute injectée.

2.4 Tolérance aux fautes dans les systèmes répartis

2.4.1 Types de pannes

Les pannes qui peuvent survenir durant une exécution répartie peuvent être classées en quatre catégories [Delbé 2007] :

- **Les pannes franches** (crash, fail-stop) : sont appelées aussi arrêt sur défaillance. C'est le cas le plus simple : on considère qu'un processus peut être dans deux états, soit il fonctionne et donne le résultat correct, soit il ne fait plus rien. Dans le second cas, le processus est considéré comme *définitivement* défaillant.
- **Les pannes par omission** (transient, omission failures) : dans ce cas, on considère que le système peut perdre des messages. Ce modèle peut servir à représenter des défaillances du réseau plutôt que des processus.
- **Les pannes de temporisation** (timing, performance failures) : ce sont les comportements anormaux par rapport à un temps, comme par exemple l'expiration d'un délai de garde.

– **Les pannes arbitraires, ou byzantines** (malicious, byzantine failures) : cette classe représente toutes les autres pannes : le processus peut alors faire "n'importe quoi", y compris avoir un comportement malveillant.

Le cas le plus simple est bien sûr le cas des pannes franches, et on essaie toujours de s'y ramener, par exemple en tuant un processus en cas de comportement imprévu. La plupart des protocoles de tolérance aux pannes pour les systèmes répartis ne considèrent que ce type de pannes.

2.4.2 Détection des pannes

Le problème de la détection des pannes est résolu différemment selon le modèle de communication du système : synchrone ou asynchrone. Dans le cas des modèles synchrones, le temps de transmission d'un message est borné. On suppose qu'un envoi de message provoque la mise en attente de l'émetteur d'une confirmation de réception de la part du récepteur. La détection des pannes franches et de temporisation peut alors être réalisée à l'aide de délais de garde lors des communications : lorsqu'un processus communique avec un autre et ne reçoit pas de confirmation après ce délai de garde, il peut considérer que le processus cible est défaillant [Delbé 2007].

Le problème devient plus complexe dans le cas des modèles asynchrones : le temps de transmission d'un message n'est pas borné. Les communications entre les nœuds ne peuvent donc plus être utilisées pour détecter les pannes car il est impossible de prédire le moment où le message est effectivement reçu par le récepteur. On utilise alors un détecteur (ou suspecteur) de pannes, qui informe les processus sur l'état du système. On distingue deux modèles différents :

– Le modèle *push*, dans lequel chaque processus du système doit régulièrement informer les détecteurs de pannes de son état. Si ce détecteur n'a pas reçu de message de type "je suis vivant" de la part d'un processus depuis un temps donné, ce processus est suspecté d'être défaillant.
– Le modèle *pull*, dans lequel ce sont les détecteurs de pannes qui envoient régulièrement des requêtes de type "es-tu vivant ?" aux processus du système. Un processus qui ne répond pas dans un temps donné est suspecté d'être défaillant.

2.4.3 Solutions de tolérance aux fautes pour les systèmes répartis

Dans un système réparti, les techniques de tolérance aux fautes peuvent être séparées en deux classes : les techniques basées sur la *réplication* et les techniques basées sur une

mémoire stable.

2.4.3.1 Tolérance aux fautes par réplication

La tolérance aux fautes par réplication consiste à utiliser des copies multiples d'un même composant ou processus sur des processeurs différents. De cette manière, la défaillance d'un composant peut être masquée par l'une de ses copies. La principale difficulté de cette approche est de conserver une cohérence forte entre les copies. On distingue trois stratégies principales pour réaliser la réplication : la réplication active, passive et semi-active [Besseron 2010].

- **La réplication active :** désigne les stratégies dans lesquelles toutes les copies jouent un rôle identique. Toutes les copies reçoivent la même séquence ordonnée de requêtes, qui sont toutes traitées dans le même ordre. Cette stratégie nécessite un mécanisme de diffusion atomique et requiert que l'exécution des requêtes soit déterministe pour garantir la cohérence.
- **La réplication passive :** on distingue la copie primaire et les copies secondaires. La copie primaire est la seule qui reçoit les requêtes et qui effectue toutes les opérations. Pour assurer la cohérence, la copie primaire diffuse son nouvel état aux copies secondaires après chaque modification. Cet état sert de point de reprise en cas de défaillance.
- **La réplication semi-active :** est une amélioration de la réplication active. À la différence de celle-ci, les copies secondaires attendent une notification de la copie primaire avant de traiter la requête. Cette notification comporte les informations nécessaires qui permettent de résoudre le problème de l'indéterminisme du traitement des requêtes.

La tolérance aux pannes par réplication est alors réalisée par masquage d'erreur. Le principal désavantage de cette méthode est qu'elle nécessite de nombreuses ressources : pour tolérer p défaillances, il est nécessaire d'avoir $p + 1$ composants identiques. Cette méthode n'est donc pas adaptée aux calculs parallèles en termes de performances et de ressources de calculs nécessaires à sa mise en œuvre [Jafar 2006] [Besseron 2010].

2.4.3.2 Tolérance aux fautes par mémoire stable

La mémoire stable représente un support persistant de stockage. Son rôle principal est d'assurer une accessibilité et une protection aux données contre les défaillances pouvant affecter le système. Le principe de la tolérance aux fautes par mémoire stable est, en cas de défaillance, de rétablir l'application dans un état cohérent ayant été stocké antérieurement à cette défaillance sur la mémoire stable [Jafar 2006] [Besseron 2010].

La mise en œuvre d'une mémoire stable dépend essentiellement des types de défaillances

auxquels on souhaite faire face.

- Pour un système qui ne tolère qu'une seule défaillance (respectivement p défaillances), la mémoire stable peut correspondre à la mémoire volatile ou au disque d'un autre processus (respectivement de p autres processus).
- Dans un système qui ne souhaite tolérer que les défaillances transitoires, la mémoire stable peut correspondre au disque dur local du processus.
- Pour un système qui tolère un nombre quelconque de défaillances permanentes, la mémoire stable est un support de stockage persistant situé en dehors des nœuds exécutant les processus de calcul et reste accessible à tout moment.

2.5 Conclusion

Dans ce chapitre, nous avons présenté la notion de sûreté de fonctionnement d'un système informatique. Nous avons vu que la tolérance aux fautes n'est qu'un moyen pour assurer la sûreté de fonctionnement et que sa mise en œuvre passe par la détection d'erreur et le rétablissement du système, qui nécessitent tous deux de la redondance dans le système. Il est à noter qu'il n'existe pas de méthode de tolérance aux fautes qui soit valable dans l'absolu. Seules existent des méthodes adaptées à des hypothèses particulières d'occurrence de fautes.

Les techniques de tolérance aux fautes spécifiques aux systèmes répartis sont basées sur la réplication ou sur l'utilisation d'une mémoire stable (la reprise). La réplication n'est pas adaptée au domaine du calcul haute performance puisqu'elle nécessite un nombre important de ressources qui pourraient être utilisées pour accélérer le calcul. De plus, elle implique un surcoût important lié à la gestion des copies multiples et limite le nombre de pannes tolérées. Par conséquent, la tolérance aux fautes par reprise semble la méthode la plus adaptée aux systèmes parallèles.

Dans le chapitre suivant, nous allons présenter les différents problèmes liés à la tolérance aux fautes par reprise et les diverses solutions proposées dans la littérature.

CHAPITRE 3

Tolérance aux fautes par reprise et applications distribuées

Sommaire

3.1 Introduction

La tolérance aux fautes par reprise est basée sur la redondance d'informations. Ces informations correspondent à la sauvegarde de l'état des processus ou bien à la journalisation d'évènements. Ceci nécessite d'une part, l'existence d'une mémoire stable de sorte que les données soient toujours accessibles et, d'autre part, que l'état du système ou du calcul soit cohérent après une reprise. Il existe deux grandes familles de protocoles de reprise : les protocoles basés sur la sauvegarde de points de reprise et les protocoles basés sur la journalisation de messages.

Dans ce chapitre, nous allons présenter la tolérance aux fautes par reprise dans une application distribuée. Nous allons commencer par l'introduction d'une modélisation de l'application distribuée, puis nous allons aborder deux problèmes liés à la reprise : le déterminisme d'exécution et la cohérence d'état global. Par la suite, nous allons étudier les deux familles de protocoles de tolérance aux fautes par reprise, à savoir les protocoles basés sur la sauvegarde de points de reprise et ceux basés sur la journalisation de messages. Enfin, une analyse comparative entre ces protocoles de reprise selon différents critères sera présentée.

3.2 Application distribuée

Une application distribuée est modélisée comme un ensemble de processus qui communiquent en échangeant des messages via un réseau de communication. L'état de l'application est l'ensemble des états locaux de tous les processus participants et de tous les canaux de communication (tous les messages en transit). Cependant, un processus ne peut stocker que son état local et ses messages entrants et sortants. Aussi, les messages en transit ne peuvent être contrôlés par aucun des processus de l'application distribuée [Jafar 2006].

Par conséquent, le raisonnement sur le système ou l'application (état, ordre des événements) pose un certain nombre de problèmes liés à la répartition.

- L'absence d'une **mémoire commune** qui peut supporter l'état d'une application.
- L'absence d'une **horloge commune** permettant la définition d'un ordre sur les événements du système.
- L'**asynchronisme des communications** entre les processus.
- L'**hétérogénéité** des sites d'exécution.

Dans [Lamport 1978], Lamport a proposé de définir pour les systèmes répartis une notion de temps logique permettant d'ordonner les événements du système. Sur un site, les événements locaux peuvent être ordonnés en se basant sur l'ordre de leur exécution. L'émission d'un message sur le site émetteur précède toujours sa réception sur le site récepteur. Ceci correspond à la notion de précédence causale.

Il existe plusieurs modèles temporels caractérisant les canaux de communication [Besseron 2010].

- Le modèle synchrone indique que la durée de transfert des messages est bornée. Cependant ce modèle n'est pas suffisamment proche de la réalité puisqu'il nécessite que la borne de temps soit toujours respectée.
- Le modèle asynchrone garantit qu'un message émis sera délivré au destinataire, cependant il n'est pas possible de borner la durée de transfert du message. En présence de défaillances, ce modèle ne permet pas de résoudre certains problèmes fondamentaux comme le consensus.
- Le modèle asynchrone avec détecteur de défaillance permet de résoudre le problème du consensus en présence de défaillances tout en conservant des hypothèses réalistes vis-à-vis des communications.

3.3 Les problèmes de la reprise

Deux problèmes interviennent pour effectuer la tolérance aux fautes par reprise : le déterminisme d'exécution et la cohérence d'état global de l'application distribuée.

3.3.1 Exécution déterministe

Le déterminisme de l'exécution d'un programme P est le fait de pouvoir répéter l'exécution de P plusieurs fois et de la même façon. Cette propriété d'exécution permet de déterminer si la reprise peut être faite par une technique de journalisation de l'exécution ou non. En effet, le stockage de l'histoire de l'exécution de P dans une mémoire stable est inutile si la réexécution de P n'est pas équivalente à l'exécution de P [Jafar 2006].

Afin d'obtenir une exécution déterministe, un processus peut être modélisé par une séquence d'intervalles d'états [Strom 1985] chacun débutant par un événement non déterministe pouvant être identifié. L'exécution durant chaque intervalle d'état est déterministe. Ce concept d'intervalle d'état est appelé hypothèse de déterminisme par morceaux (PWD : Piecewise deterministic assumption). L'hypothèse PWD permet de capturer suffisamment d'informations concernant les événements non déterministes qui initialisent les intervalles d'états.

3.3.2 État global cohérent d'une application distribuée

Un état global d'une application distribuée est une collection des états locaux de tous les processus participant au calcul et les états des canaux de communication entre les processus. Un état global cohérent est simplement l'un des états globaux qui peut se produire durant une exécution correcte (sans fautes). Autrement dit, dans un état global cohérent, si l'état d'un processus reflète la réception d'un message alors l'envoi du message est dans l'état de son émetteur [Chandy 1985] [Elnozahy 2002].

L'état global C_1, C_2, C_3 illustré dans la figure 3.1 est cohérent. Les messages m_2 et m_3 ont été envoyés mais pas encore reçus, ce qui est possible durant une exécution correcte si les canaux sont asynchrones. Ces messages sont appelés messages ***en transit*** et ils font partie de l'état global du système. Les messages en transit n'introduisent aucune incohérence dans l'état global. Cependant, la garantie de la délivrance des messages en transit doit être assurée, soit par l'hypothèse d'un réseau de communication fiable, soit par le protocole de reprise lui-même.

Par contre, l'état global C_1, C_2, C_3 illustré dans la figure 3.2, n'est pas cohérent. En effet, le message m_2 n'a pas été envoyé par le processus P_1 alors que m_2 est reçu par le processus P_2.

Cette situation ne peut jamais se produire dans une exécution correcte. Dans un état global, un message reçu sans être envoyé (par exemple, le message m_2 dans la figure 3.2) est appelé message ***orphelin***. Un message orphelin cause l'incohérence de l'état global qui le contient.

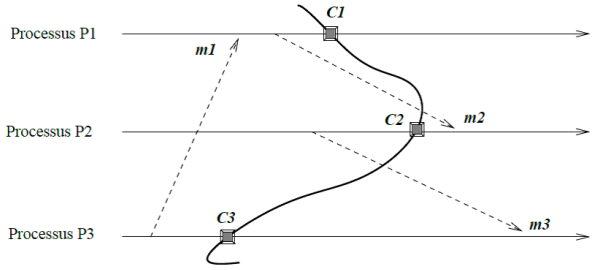

FIGURE 3.1 – État global cohérent.

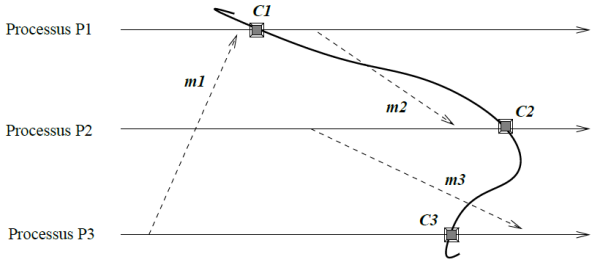

FIGURE 3.2 – État global incohérent.

3.4 Reprise par sauvegarde

Les protocoles de reprise par sauvegarde sont les plus utilisés. Ils consistent à sauvegarder périodiquement l'état des processus sur un support stable (support qui tolère les défaillances). Lorsqu'une défaillance se produit, ces protocoles restaurent l'état du système ou de l'application à partir du plus récent ensemble cohérent de points de reprise (un point de reprise par processus). Cet ensemble cohérent de points de reprise est appelé *ligne de recouvrement* [Randell 1975].

Ces approches basées sur la sauvegarde (*checkpointing*) supposent un modèle d'exécution non déterministe. De plus, elles n'ont pas besoin de stocker et de rejouer les événements non déterministes. De ce fait, après une défaillance, il n'y a aucun moyen de garantir le rejeu à l'identique des évènements. [Randell 1975]

Les protocoles de reprise par sauvegarde peuvent être classifiés en trois catégories selon le mode de construction de la ligne de recouvrement : sauvegarde non coordonnée, sauvegarde coordonnée et sauvegarde induite par les communications. [Elnozahy 2002]

3.4.1 Sauvegarde non coordonnée

La sauvegarde non coordonnée [Elnozahy 2002] évite la coordination et laisse à chaque processus la décision de sauvegarder son état. Le principal avantage est que chaque processus peut décider de sauvegarder son état quand ça lui convient le mieux. Par exemple, un processus peut procéder à la sauvegarde de son état quand la taille de celui-ci est minimale, réduisant ainsi le surcoût en terme de quantité d'informations à sauvegarder [Wang 1995].

Cependant cette approche comporte plusieurs inconvénients. Tout d'abord, lors de la reprise, il y a un risque d'*effet domino* [Randell 1975] : en forçant le retour arrière des processus ayant reçu un message orphelin, les processus peuvent être amenés à faire des retours arrière en cascade jusqu'à atteindre l'état initial. Ce phénomène peut causer la perte d'une grande partie du travail déjà effectué avant la panne. De plus, certains points de reprise ne feront jamais partie d'un état global cohérent. Ceux-ci induisent un surcoût pendant l'exécution sans panne mais ne servent pas à la reprise. Enfin, la sauvegarde non coordonnée oblige chaque processus à maintenir plusieurs points de reprise et nécessite l'utilisation périodique d'un ramasse-miette pour supprimer les points de reprise inutiles.

Durant l'exécution sans panne, les dépendances entre les points de reprise causées par l'échange de messages entre les processus sont enregistrées en mémoire stable. Lorsqu'une défaillance survient, le protocole de reprise recherche une ligne de recouvrement parmi tous les états globaux formés. Cette recherche s'effectue généralement par des techniques fondées sur les graphes de dépendance entre les points de reprise [Bhargava 1988] [Wang 1997].

3.4.2 Sauvegarde coordonnée

Dans la sauvegarde coordonnée [Elnozahy 2002], les points de reprise sont coordonnés sur tous les processus de manière à assurer que l'état global résultant soit cohérent. Cette approche permet d'éviter l'effet domino lors de la reprise puisque tous les états globaux crées sont forcément cohérents. De plus, seul le dernier point de reprise de chaque processus est nécessaire, ce qui réduit le surcoût de stockage et de suppression de points de reprise. Le principal inconvénient de cette technique est le surcoût introduit par la synchronisation des processus.

Une approche simple pour réaliser la coordination est de bloquer les processus pendant la phase de la sauvegarde [Tamir 1984] [Plank 1993]. Le processus initiateur sauvegarde un point de reprise et diffuse une requête vers tous les autres processus. Lorsqu'un processus reçoit une telle requête, il arrête son exécution, prend un premier point de reprise et envoie

une réponse au processus initiateur. Quand celui-ci reçoit toutes les réponses, il diffuse un autre message pour sauvegarder le nouveau point de reprise et supprimer le précédent. Enfin, le processus reprend son exécution.

Cependant, si la taille des points de reprise à sauvegarder est grande, le surcoût introduit sur le temps d'exécution en bloquant les processus est important [Elnozahy 1992]. Afin d'améliorer les performances de la sauvegarde coordonnée, plusieurs techniques ont été proposées :

- La sauvegarde coordonnée **non bloquante** [Chandy 1985] permet d'identifier l'état des canaux de communication grâce à des messages *marqueurs*. Tout d'abord, le processus initiateur sauvegarde son état local et diffuse un marqueur sur tous les canaux sortants. Lorsqu'un processus reçoit un tel message pour la première fois, il prend son état local et diffuse immédiatement le marqueur. Ensuite, il sauvegarde l'état de chaque canal entrant dès la réception d'un marqueur. Cette approche suppose que les canaux sont FIFO. Dans le cas où les canaux ne sont pas FIFO, les marqueurs sont attachés aux messages émis après la prise de l'état [Lai 1987].
- La sauvegarde coordonnée avec **horloges synchronisées** [Cristian 1991] [Tong 1992] est fondée sur le temps pour coordonner les processus. Elle suppose que les horloges sont approximativement synchronisées. Chaque processus sauvegarde son état lorsque l'horloge locale atteint le moment de sauvegarde du point de reprise.
- La sauvegarde coordonnée **minimale** [Koo 1987] ne coordonne que les processus ayant communiqué depuis le dernier point de reprise, afin de limiter le nombre de processus impliqués dans la coordination. De plus, lorsqu'une défaillance se produit, seuls les processus ayant communiqué effectuent un retour arrière.

3.4.3 Sauvegarde induite par les communications

La sauvegarde induite par les communications [Baldoni 1997] est un compromis entre la sauvegarde coordonnée et la sauvegarde non coordonnée. Elle utilise deux types de points de reprise : local et forcé. Les points de reprise locaux sont effectués de manière indépendante, tandis que les points de reprise forcés doivent être effectués pour éviter l'effet domino.

Dans cette approche, des informations supplémentaires sont attachées aux messages afin de permettre à un processus de décider s'il doit prendre un point de reprise forcé. Cette décision permet d'empêcher la création des points de reprise inutiles, et de garantir donc la progression de la ligne de recouvrement. Ce principe a été proposé formellement dans [Netzer 1995], par la définition des notions de Z-chemin (zigzag path) et de Z-cycle qui

formalisent les conditions de cohérence d'un état global.

On distingue deux familles de protocoles de sauvegarde induite par les communications :

- Les protocoles *model-based* [Hélary 2000] : les communications et les sauvegardes de points de reprise doivent respecter un certain motif. Par exemple, la prise d'un point de reprise immédiatement après chaque envoi de message permet d'éviter l'effet domino. Ces protocoles sont généralement basés sur les notions de Z-chemin et de Z-cycle.
- Les protocoles *index-based* [Briatico 1984] [Manivannan 1996b] : les points de reprise sont indexés et chaque message porte l'index du dernier point de l'émetteur. Lorsqu'un processus reçoit un message indiquant un index supérieur au sien, il doit prendre un point de reprise avant le traitement du message pour éviter l'incohérence entre les points de reprise ayant le même index. Ces protocoles sont donc basés sur la suppression des dépendances causales entre les points de reprise des processus qui ont des communications directes.

3.5 Reprise par journalisation

Les protocoles de reprise par journalisation reposent sur l'hypothèse de *déterminisme par morceaux* (PWD : piecewise determinism) [Strom 1985]. Sous cette hypothèse, l'exécution d'un processus peut être modélisée par une séquence d'intervalles d'états déterministes commençant par des événements non déterministes. Un événement non déterministe peut être la réception d'un message ou un événement interne au processus.

Le principe de la reprise basée sur la journalisation (*logging*) est de sauvegarder l'histoire d'exécution de l'application. Durant l'exécution sans panne, chaque processus sauvegarde en mémoire stable les informations correspondant à tous les événements non déterministes qu'il observe, et effectue, de façon totalement indépendante, des sauvegardes périodiques de son état. Après une panne, le processus défaillant reconstruit son état d'avant la panne à partir de son point de reprise le plus récent et en rejouant les événements non déterministes sauvegardés.

Traditionnellement, les réceptions de messages sont considérées comme les seuls évènements non déterministes et le mécanisme de journalisation est donc appelé *journalisation de messages*. La journalisation des messages se fait soit par l'émetteur soit par le récepteur. Dans les protocoles basés sur le récepteur, les processus sauvegardent chaque message qu'ils reçoivent sur support stable [Sistla 1989]. Dans les protocoles basés sur l'émetteur, chaque processus enregistre les messages dans sa mémoire. Ces données sont enregistrées sur support

stable lorsque le processus sauvegarde son point de reprise [Johnson 1987].

Les protocoles de reprise basés sur la journalisation doivent assurer la condition de « **non-orphelinité** » suivante : lors de la reprise des processus défaillants, le système ou l'application ne doit contenir aucun *processus orphelin*. Un processus orphelin est un processus dont l'état dépend d'un événement non déterministe qui ne peut pas être reproduit à la reprise. Une description formelle de cette condition a été introduite dans [Elnozahy 2002].

Il existe trois familles de protocoles de reprise par journalisation : journalisation pessimiste, journalisation optimiste et journalisation causale. Ils diffèrent par leur manière d'assurer et d'implanter la condition de « non-orphelinité ».

3.5.1 Journalisation pessimiste

Les protocoles pessimistes [Johnson 1987] [Elnozahy 1994a] reposent sur l'hypothèse qu'une défaillance peut se produire immédiatement après un événement non déterministe. Le principe de ces protocoles est donc de ne pas permettre à un processus de dépendre d'un évènement non déterministe tant que celui-ci n'a pas été stocké sur un support stable. Dans le cas où les événements non déterministes sont uniquement les réceptions de messages, un processus doit sauvegarder tous les messages reçus avant d'émettre un message à un autre processus [Alvisi 1998]. Les sauvegardes effectuées doivent donc être réalisées de manière synchrone. De plus, chaque processus effectue des points de reprise périodiques pour limiter la quantité de travail qui doit être répété lors du recouvrement. Lorsqu'une panne se produit, seul le processus défaillant effectue un retour arrière vers son dernier point de reprise.

Ces protocoles ont plusieurs avantages. D'une part, seuls les processus fautifs redémarrent après une défaillance puisqu'aucun processus orphelin n'est créé. D'autre part, étant donné qu'un processus fautif effectue un retour arrière vers son dernier point de reprise, il n'est pas nécessaire de garder les informations de recouvrement qui ont été sauvegardées avant ce point de reprise, facilitant ainsi leur suppression. Cependant, ces protocoles introduisent un surcoût important durant l'exécution sans panne. Ceci est dû à l'enregistrement synchrone des messages [Elnozahy 2002].

3.5.2 Journalisation optimiste

Les protocoles optimistes [Strom 1985] [Sistla 1989] [Johnson 1990] tentent d'améliorer les performances en exécution sans panne en faisant l'hypothèse (optimiste) qu'une panne ne se produira pas avant la sauvegarde de l'évènement non déterministe. La sauvegarde des messages sur support stable se fait de façon asynchrone : les messages sont gardés en mémoire

volatile et sont périodiquement sauvegardés sur support stable. Ainsi, un processus n'a pas besoin de se bloquer pendant cette sauvegarde.

Lorsqu'une défaillance se produit, le processus fautif effectue un retour arrière vers son dernier point de reprise, perdant ainsi l'ensemble des messages enregistrés en mémoire volatile. Ceci peut donc amener à la création de messages orphelins. Les processus ayant reçu de tels messages deviennent des processus orphelins et doivent effectuer également un retour arrière. Cependant, dans [Johnson 1990], les auteurs ont montré que ces protocoles ne provoquent pas d'effet domino.

Durant l'exécution sans panne, les processus doivent alors maintenir des informations sur les dépendances causales entre eux. Lors de la reprise, ces informations sont utilisées pour déterminer les processus qui doivent redémarrer pour que l'état global reste cohérent. Ainsi, ces protocoles introduisent un surcoût important dû au calcul de l'état global cohérent.

Ces protocoles ont également pour inconvénients de compliquer la suppression des données obsolètes car plusieurs processus peuvent être forcés à redémarrer d'un point de reprise qui n'est pas le plus récent après la défaillance d'un seul processus.

3.5.3 Journalisation causale

La journalisation causale [Elnozahy 1994b] [Alvisi 1996] combine les avantages de la journalisation pessimiste pour la reprise et les avantages de la journalisation optimiste en ce qui concerne les performances à l'exécution. Elle évite d'une part, l'accès synchrone au support stable et, d'autre part, la création de processus orphelins. Cela permet la reprise de n'importe quel processus défaillant à partir de son dernier point de reprise. L'inconvénient de cette technique est la complexité du protocole de recouvrement suite à une défaillance.

Le principe de la journalisation causale est d'assurer que le *déterminant* de chaque événement non déterministe (c'est-à-dire les informations permettant de rejouer un évènement non déterministe) qui précède causalement l'état d'un processus se trouve soit sur support stable, soit est disponible localement pour ce processus. Ces informations protègent ce processus des défaillances des autres processus et permettent de garantir la condition de non-orphelinité. L'implantation de cette condition est faite par l'ajout des déterminants se trouvant dans la mémoire volatile d'un processus aux messages envoyés vers d'autres processus [Elnozahy 2002].

Lorsqu'une défaillance se produit, seul le processus fautif effectue un retour arrière et rejoue les messages dans l'ordre grâce aux déterminants sauvegardés soit sur support stable, soit au niveau des autres processus. Si tous les processus dépendant d'un même évènement

non déterministe subissent une défaillance, tous les déterminants dont ils ont besoin sont perdus. Cependant, étant donné qu'ils redémarrent tous, aucun message orphelin n'est créé et ainsi l'état qu'ils atteindront sera cohérent [Elnozahy 1994b].

3.6 Analyse comparative

Le tableau 3.1 récapitule les avantages et les inconvénients des différentes techniques de tolérance aux fautes par reprise. Les critères utilisés pour la comparaison sont les suivants [Besseron 2010] [Elnozahy 2002].

- **Hypothèse PWD** : indique si cette technique repose sur l'hypothèse PWD. On remarque que seuls les protocoles de reprise par journalisation font cette hypothèse. Si l'exécution ne respecte pas cette hypothèse, la reprise par sauvegarde doit être utilisée.
- **Processus orphelin** : indique si le dernier état global sauvegardé peut contenir des processus orphelins. La sauvegarde coordonnée, la journalisation pessimiste et la journalisation causale ne créent aucun processus orphelin. Cela permet de reprendre un processus défaillant à partir de son dernier point de reprise. Plusieurs points de reprise par processus sont utilisés pour éviter les processus orphelins dans la sauvegarde non coordonnée, la sauvegarde induite par les communications et la journalisation optimiste.
- **Effet domino** : indique s'il y a un risque d'effet domino au moment de la reprise. La sauvegarde non coordonnée est la seule technique sensible à l'effet domino. Elle oblige chaque processus à conserver plusieurs points de reprise afin d'éviter la perte d'une grande quantité des calculs effectués avant la panne.
- **Point de reprise par processus** : donne le nombre de points de reprise par processus à conserver pour permettre de construire un état global cohérent lors de la reprise. Ceci est la conséquence de la possibilité d'apparition de processus orphelins et de l'effet domino.

3.7 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur la tolérance aux fautes par reprise dans un système distribué. Nous avons vu dans un premier temps que la reprise pose deux problèmes : le déterminisme d'exécution et la cohérence de l'état global. Dans le cas où l'exécution respecte l'hypothèse PWD, une des techniques de journalisation des événements non déterministes peut être utilisée. Autrement, l'approche de reprise par sauvegarde doit être utilisée. La cohérence d'état global peut être assurée soit au moment de la sauvegarde

Protocoles	Hypothèse PWD	Processus orphelin	Effet domino	Point de reprise par processus
Sauvegarde non coordonnée	Non	Possible	Possible	Plusieurs
Sauvegarde coordonnée	Non	Non	Non	Une
Sauvegarde induite par les communications	Non	Possible	Non	Plusieurs
Journalisation pessimiste	Oui	Non	Non	Une
Journalisation optimiste	Oui	Possible	Non	Plusieurs
Journalisation causale	Oui	Non	Non	Une

TABLEAU 3.1 – Comparaison des différentes techniques de tolérance aux fautes par reprise.

des points de reprise, soit durant la reprise. Le protocole de reprise détermine si après une panne, la reprise globale du système est imposée ou si seuls les processus défaillants doivent redémarrer.

Nous avons présenté les deux familles de protocoles de tolérance aux fautes par reprise : les protocoles basés sur la sauvegarde et les protocoles basés sur la journalisation. Les protocoles basés sur la sauvegarde visent à construire un état global cohérent en cherchant à déterminer un ensemble de points de reprise qui forme un état exempt de messages orphelins. Les protocoles basés sur la journalisation visent à rétablir l'état de l'application exactement tel qu'il était avant la panne en rejouant les évènements non déterministes qui ont amené les processus à cet état.

Nous avons montré que chaque protocole possède ses propres caractéristiques et qu'il n'existe pas de solution unique qui soit adaptée à toutes situations. Le choix d'un protocole de tolérance aux fautes doit être guidé par les propriétés de l'application, par les propriétés de l'infrastructure physique utilisée et par la fréquence d'apparition des pannes.

Dans le chapitre suivant, nous allons présenter un protocole de tolérance aux fautes par reprise adapté au contexte d'un réseau mobile avec infrastructure à couverture discontinue.

Deuxième partie

Contribution

CHAPITRE 4

Protocole proposé et évaluation

Sommaire

4.1 Introduction

Nous avons étudié dans le chapitre précédent la technique de tolérance aux fautes par reprise dans les systèmes distribués. Nous avons vu deux familles de protocoles de reprise : les protocoles basés sur la sauvegarde de points de reprise et les protocoles basés sur la journalisation de messages. Compte tenu des nouvelles contraintes caractérisant un environnement mobile, de nouveaux protocoles adaptés à ce contexte ont été proposés dans la littérature.

Cependant, ces protocoles supposent que les fautes matérielles causant la défaillance d'un processus sont transitoires et ne prennent pas en considération leur durée de persistance. En outre, l'environnement mobile considéré se base sur une infrastructure à couverture continue. Par conséquent, dans ce contexte, des mesures adéquates doit être prises en compte afin d'assurer la continuité de délivrance des services d'une application distribuée mobile en traitant à la fois des fautes matérielles transitoires et permanentes.

Dans ce chapitre, nous allons présenter notre protocole proposé pour la tolérance aux fautes d'une application distribuée dans un environnement mobile basé sur un réseau mobile avec infrastructure à couverture discontinue. Nous allons commencer d'abord par présenter la problématique et les motivations de cette proposition. Ensuite, nous allons décrire le modèle du système et les hypothèses considérés. Après, nous allons présenter l'idée de base ainsi que

la description détaillée de notre protocole. Enfin, nous allons étudier le protocole proposé par l'évaluation de ses performances.

4.2 Problématique et motivation

De nombreux algorithmes supportant des services distribués sont aujourd'hui étendus pour continuer leurs services dans l'environnement mobile [Badrinath 1994]. La technique de reprise est l'un des services distribués permettant d'assurer la tolérance aux pannes dans le système. Ainsi, plusieurs protocoles de tolérance aux fautes par reprise ont été proposés dans la littérature pour les systèmes distribués. Cependant ces protocoles ne peuvent pas être utilisés directement dans l'environnement mobile, comme la plupart des autres services distribués. L'existence de nœuds mobiles dans un tel environnement introduit de nouvelles contraintes qui nécessitent une manipulation appropriée lors de la conception de ces protocoles. Ces contraintes sont la mobilité, les déconnexions, la bande passante faible, la source d'énergie limitée, la vulnérabilité aux dommages physiques, le manque de support stable, etc.

Les protocoles de reprise qui requièrent la synchronisation entre les processus pendant la sauvegarde des points de reprise ou lors du recouvrement après panne, ainsi que ceux exigeant de prendre plusieurs points de reprise par processus, ne sont pas appropriés pour l'environnement mobile. En effet, le grand nombre de messages de contrôle échangés entre les nœuds mobiles introduit un surcoût en termes de bande passante et de consommation d'énergie, et aussi les nœuds mobiles déconnectés temporairement du réseau ne peuvent pas participer à la coordination. De plus, la sauvegarde fréquente des points de reprise génère un surcoût de stockage et une consommation éventuelle en bande passante.

Dans le chapitre 3, les deux familles de protocoles de reprise sont étudiées : les protocoles basés sur la sauvegarde de points de reprise et les protocoles basés sur la journalisation de messages. La sauvegarde de points de reprise est effectuée de trois manières : non coordonnée, coordonnée et induite par les communications. La sauvegarde non coordonnée souffre de l'effet domino et complique le recouvrement. La sauvegarde coordonnée nécessite la coordination des processus pour former un état global cohérent et simplifie le recouvrement. La sauvegarde induite par les communications évite l'effet domino et la coordination, mais génère un nombre important de points de reprise avec une complexité de recouvrement. Plusieurs travaux étendant ces approches pour l'environnement mobile ont été proposés. Parmi ceux-ci, nous citons : [Acharya 1994], [Manivannan 1996a], [Prakash 1996], [Cao 1998], [Cao 2001], [Kumar 2008] et [Khunteta 2011].

Les protocoles de reprise par journalisation peuvent être classifiés en trois catégories : pessimiste, optimiste et causale. La journalisation pessimiste introduit un surcoût important durant l'exécution sans panne dû à l'accès synchrone au support stable, cependant elle simplifie le recouvrement. La journalisation optimiste est plus performante durant l'exécution sans panne, mais lors du recouvrement, plusieurs processus peuvent être amenés à redémarrer. La journalisation causale combine les avantages des deux protocoles précédents, toutefois elle implique un surcôut en termes de stockage et de communication dû aux informations de dépendances échangées entre processus. Parmi les travaux qui adaptent ces protocoles de journalisation pour l'environnement mobile, nous citons : [Park 2000], [Ahn 2004], [Park 2003], [George 2006] et [Imran 2007].

Ces différents travaux de tolérance aux fautes par reprise dans l'environnement mobile supposent que les fautes matérielles (pannes) sont transitoires et ne prennent pas en considération leur durée de persistance. Ainsi, dans le cas des fautes matérielles permanentes, si aucune mesure n'est prise en compte pour tolérer cette faute, l'application distribuée mobile n'arrivera jamais à sa terminaison normale [Aliouat 2007] [Aliouat 2009].

De plus, l'environnement mobile considéré dans ces travaux est basé sur une infrastructure à couverture continue déployée d'une manière planifiée afin d'offrir en permanence aux nœuds mobiles un lien radio vers une station de base. Dans un réseau mobile avec infrastructure à couverture discontinue, un nœud mobile subit des fréquentes déconnexions durant sa mobilité. Dans pareille situation, une stratégie adéquate doit être proposée afin d'assurer aux applications distribuées une exécution non-stop en dépit de la discontinuité de la couverture.

L'objectif premier de ce travail est d'assurer à une application distribuée mobile une exécution non-stop [Aliouat 2009] en traitant des différents types de fautes matérielles transitoires et permanentes, et de minimiser l'overhead temporel qui affecte la délivrance de ses services (résultats) dans le respect des temps impartis. Pour atteindre cet objectif dans cet environnent mobile caractérisé par des fréquentes déconnexions, un processus défaillant nécessite un recouvrement asynchrone (indépendant) qui n'affecte pas les autres processus de l'application distribuée. Par conséquent, la stratégie de tolérance aux fautes proposée se base sur la sauvegarde asynchrone de points de reprise et la journalisation pessimiste de messages.

4.3 Modèle du système

Dans cette section, nous allons présenter notre modèle du système qui inclut les modèles du réseau, de l'application distribuée et des fautes (pannes).

4.3.1 Modèles du réseau et de l'application distribuée

Dans notre étude, nous considérons un réseau mobile avec infrastructure à couverture discontinue (voir figure 4.1). Il se compose d'un ensemble de nœuds ou sites mobiles (MH : Mobile Host) qui se déplacent librement et d'un ensemble de stations de base (BS : Base Station) fixes interconnectées par un réseau filaire. Un site mobile est relié à une station de base via un lien sans fil. Une station de base gère toutes les communications depuis et vers les sites mobiles se trouvant dans sa zone de couverture appelée cellule et déterminée par une portée de transmission sans fil. Le déploiement des stations de base est effectué de manière non planifiée, formant ainsi un ensemble des îlots de connectivité. Chacun d'entre eux est composé d'une ou plusieurs cellules adjacentes qui se recouvrent partiellement. Lorsqu'un site mobile se déplace entre deux cellules d'un même îlot de connectivité, un handover est effectué, sinon il va subir une déconnexion due à l'absence de couverture.

Les protocoles des réseaux filaire et sans fil sont supposés être fiables et assurant la délivrance des messages dans un ordre FIFO aux processus. Chaque site mobile peut connaître son niveau d'énergie ainsi que sa position absolue par un récepteur GPS. Certains sites mobiles sont disponibles dans le système pour remplacer ceux défaillants en cas de faute permanente (sites de support ou de reprise). Les ressources de la station de base sont supposées être largement suffisantes (calcul, stockage, énergie, bande passante, etc.).

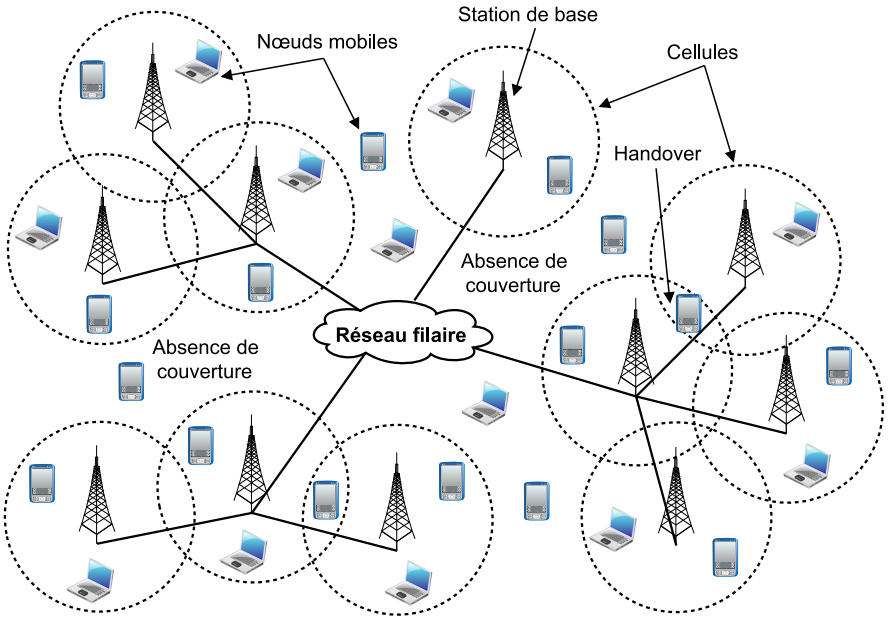

FIGURE 4.1 – Le modèle du réseau.

Une application distribuée est constituée de N processus séquentiels désignés par P_0, P_1, P_2, ... , et P_{n-1} s'exécutant simultanément sur des sites mobiles dans un réseau mobile avec

infrastructure à couverture discontinue. Les processus ne partagent pas une mémoire commune ou une horloge physique globale. Le passage de messages est le seul moyen de communication entre les processus. L'exécution est asynchrone : chaque processus évolue à son propre rythme et les messages sont transmis par le biais des canaux de communication dont les délais de transmission sont finis mais arbitraires. L'exécution d'un processus respecte l'hypothèse de déterminisme par morceaux (PWD). Les messages générés par l'exécution distribuée sont appelés *messages de données ou d'application*. Les messages explicites générés par le protocole de tolérance aux fautes sont appelés *messages de contrôle*.

4.3.2 Modèle de fautes

Les processus s'exécutant sur les sites mobiles suivent le mode de défaillance fail-stop (arrêt sur défaillance). Donc, un processus peut être dans l'un des deux états, soit il fonctionne et donne le résultat correct, soit il ne fait plus rien (c'est-à-dire définitivement défaillant). La défaillance d'un processus est causée par une faute matérielle (panne) transitoire ou permanente. Dans notre modèle, on distingue trois types de fautes :

- **Fautes dues aux dommages physiques :** est le résultat de phénomènes physiques permanents (court-circuit, circuit ouvert, etc.) causant des dommages irréversibles. Un site mobile endommagé se voit dans l'incapacité de poursuivre l'exécution de ses processus. Cette faute est permanente et son occurrence est imprévue.
- **Fautes d'épuisement de batterie :** un site mobile est affecté par un défaut d'énergie si le temps global d'exécution d'une application dépasse la durée de vie restante de sa batterie. Cette faute peut être considérée transitoire (remplacement de la batterie par une autre suffisamment chargée) ou permanente et son occurrence peut être prévue.
- **Fautes de déconnexion :** la déconnexion d'un site mobile du réseau est une faute transitoire et fréquente. Elle peut être volontaire ou involontaire.
  - **Déconnexion volontaire :** est déclenchée par l'utilisateur pour éteindre le site mobile, préserver la batterie, le coût de la communication, etc. Ce type de déconnexion est planifié pour une durée déterminée par l'utilisateur après laquelle le site mobile se reconnecte au réseau.
  - **Déconnexion involontaire :** est le résultat d'une sortie de la zone de couverture, d'un échec de handover, de perturbations environnementales, d'une faible bande passante, etc. La déconnexion à cause d'une sortie de la zone de couverture peut être prévue. Dans les autres cas, la déconnexion est imprévue.

4.4 Protocole proposé

Dans cette section, nous allons présenter notre protocole de tolérance aux fautes d'une application distribuée dans un réseau mobile avec infrastructure à couverture discontinue.

4.4.1 Idée de base

Le manque de support stable au niveau des sites mobiles en raison de leur vulnérabilité à la perte, le vol et les dommages physiques. De plus, un site déconnecté n'est pas accessible depuis le reste du réseau et les informations de recouvrement y sauvegardées deviennent non disponibles. Par conséquent, un support stable alternatif doit être utilisé. Compte tenu des ressources abondantes et de la stabilité des stations de base, celles-ci sont utilisées pour la sauvegarde de points de reprise et la journalisation de messages. Elles peuvent aussi tenir le rôle de sites de reprise (support) poursuivant les activités des sites défaillants.

Dans notre protocole, chaque station de base maintient un journal de messages sur son support stable pour les sites mobiles se trouvant dans sa cellule de couverture. Puisqu'un message délivré vers un site mobile est acheminé par sa station de base, la journalisation de messages n'implique aucun overhead de communication et n'affecte pas les performances de ses processus. De plus, un site mobile prend trois types de points de reprise : périodique, de migration et local. Chacun d'eux est destiné pour le recouvrement d'une catégorie de fautes.

Dans le cas d'une faute planifiée ou prédictible (déconnexion volontaire, déconnexion due à la couverture discontinue et épuisement de la batterie), un site mobile sauvegarde un point de reprise de migration à la détection de l'évènement qui précède l'occurrence de la faute, puis se déconnecte du réseau. Ensuite, ce point de reprise est utilisé pour la migration des processus du site mobile susceptible défaillant vers sa station de base courante afin de poursuivre ses activités. Pendant la durée de persistance d'une telle faute (généralement entre 180 et 300 secondes [Sadok 1999]), le site mobile peut récupérer ses processus de la station de base de déconnexion. À l'expiration de cette période, si le site mobile ne se reconnecte pas, la station de base considère que cette faute est permanente et procède à l'élection d'un site de reprise pour remplacer le site susceptible défaillant. Après sa reconnexion, le site mobile devient un site de reprise disponible.

La panne due aux dommages physiques (permanente) et la déconnexion non planifiée (transitoire) toutes les deux sont des évènements imprévus. La durée de persistance d'une déconnexion transitoire non planifiée (généralement entre 10 et 15 secondes [Sadok 1999]) est utilisée par la station de base pour la distinction entre ces deux fautes. Ainsi, lorsqu'un site

défaillant ne se reconnecte pas pendant cette période, il est affecté par une faute permanente.

Afin de permettre le recouvrement d'une faute permanente, un site mobile prend périodiquement un point de reprise et le sauvegarde sur le support stable de sa station de base. Lors de la reprise d'un site défaillant, la station de base collecte ses données de recouvrement qui peuvent être distribuées sur plusieurs stations de base, puis reprend son exécution jusqu'à la reconstruction de son état d'avant la panne. Après, elle sauvegarde un point de reprise et élit un site de support pour poursuivre les activités du site défaillant.

Dans le cas d'une faute de déconnexion non planifiée (transitoire), un site mobile sauvegarde un point de reprise sur son support stable local avant chaque handoff. Lors du recouvrement d'une telle faute, le site mobile utilise ce point de reprise local et son journal de messages enregistré sur la station de base pour reprendre ses activités. Ce type de points de reprise permet de minimiser le surcoût de recouvrement.

4.4.2 Structures de données et notations

Voici les structures de données et les notations utilisées dans notre protocole.

A. Dans un site mobile MH_i

- $LocalChkpt\_space_i$: l'espace de stockage sur le support stable local du site MH_i qui est utilisé pour la sauvegarde locale d'un point de reprise et les données y associées.
- $Chkpt_i$: le point de reprise pris par le site MH_i et contenant l'état du processus P_i.
- $Chkpt\_sn_i$: le numéro de séquence du dernier point de reprise pris par le site MH_i. Il est incrémenté chaque fois que le site MH_i prend un nouveau point de reprise.
- $RcvMsg\_sn_i$: le numéro de séquence du dernier message de données reçu par le site MH_i. Il est incrémenté chaque fois que le site MH_i reçoit un nouveau message.
- $Chkpt\_timer_i$: le temporisateur (timer) utilisé pour déclencher la sauvegarde d'un point de reprise périodique pour le site MH_i.
- $EnergyThreshold_i$: le seuil critique d'énergie (valeur prédéfinie) du site MH_i.
- $EnergyLevel_i$: le niveau d'énergie de la batterie du site MH_i.
- $AcceptEnergyLevel_i$: le niveau d'énergie au-delà duquel le site MH_i accepte sa candidature à une élection d'un site de support.
- $HandoffThreshold_i$: la distance au-delà de laquelle un handoff est déclenché. Elle est fournie au site MH_i par sa station de base courante.
- $Distance_i$: la distance entre le site MH_i et sa station de base courante BS_j. Elle est calculée en fonction de sa position (x_i, y_i) déterminée par le récepteur GPS et la position

de sa station de base courante (x_j, y_j) fournie au site MH_i lors de sa connexion.

- $AvailableBS\_set_i$: l'ensemble des nouvelles stations de base pouvant servir le site MH_i lors d'un handoff.
- $OldBS_i$: l'ancienne station de base à laquelle le site MH_i a été connecté.
- $NewBS_i$: la station de base courante du site MH_i.

B. Dans une station de base BS_j

- $Chkpt\_space_i^j$: l'espace de stockage sur le support stable de la station de base BS_j qui est utilisé pour la sauvegarde d'un point de reprise stable pris par le site MH_i pour le processus P_i ainsi que les données y associées ($Chkpt_i^j$, $Chkpt\_sn_i^j$, $RcvMsg\_sn_i^j$).
- $Log\_space_i^j$: l'espace de stockage sur le support stable de la station de base BS_j qui est utilisé pour la sauvegarde du journal de messages de données destinés vers le site MH_i ainsi que les données y associées.
- $Chkpt\_loc_i^j$: l'identificateur de la station de base sauvegardant le dernier point de reprise stable pris par le site MH_i (une information de localisation).
- $MsgLog_i^j$: le journal de messages de données destinés vers le site MH_i. Ces messages sont enregistrés sur le support stable de la station de base BS_j.
- $LogMsg\_sn_i^j$: le numéro de séquence du dernier message journalisé pour le site MH_i sur la station de base BS_j.
- $Log\_set_i^j$: l'ensemble des identificateurs des stations de base contenant les journaux de messages pour le site MH_i (une information de localisation).
- $TransmRange_j$: la portée de transmission de la station de base BS_j.
- $Cand\_list_i^j$: la liste des candidats reçus par BS_j pour remplacer le site MH_i.
- $ElectedCand_i^j$: le site de support élu par BS_j pour remplacer le site MH_i.
- $ActiveMH\_set_j$: l'ensemble des sites actifs connectés à la station de base BS_j et participent à l'exécution de l'application distribuée.
- $SupportMH\_set_j$: l'ensemble des sites de support (reprise) connectés à la station de base BS_j et sont disponibles pour remplacer si nécessaire un site défaillant.
- $Planned\_timer_i^j$: le temporisateur utilisé par BS_j pour la migration du processus P_i du site MH_i vers un site de support après sa déconnexion planifiée.
- $Unplanned\_timer_i^j$: le temporisateur utilisé par BS_j pour savoir qu'une faute permanente affecte le site MH_i après la détection de sa déconnexion imprévue (non planifiée).
- $Election\_timer_i^j$: le temporisateur utilisé par la station de base BS_j pour déclencher le choix d'un site de support pour le site MH_i parmi les candidatures parvenues.

4.4.3 Description du protocole

Cette section décrit le protocole de tolérance aux fautes que nous avons proposé.

4.4.3.1 Sauvegarde de points de reprise périodiques

Chaque site mobile MH_i prend périodiquement un point de reprise indépendamment des autres et un numéro de séquence unique est attribué à chaque point de reprise. Ainsi, à l'expiration du temporisateur (timer) $Chkpt\_timer_i$, le site MH_i incrémente le numéro de séquence $Chkpt\_sn_i$ et sauvegarde son état sous forme d'un point de reprise $Chkpt_i$. Il sauvegarde aussi sur le support stable local le point de reprise $Chkpt_i$ avec son numéro de séquence $Chkpt\_sn_i$ et le numéro de séquence $RcvMsg\_sn_i$ du dernier message reçu par MH_i avant de prendre le point de reprise. Ensuite, MH_i envoie une requête *SaveChkptReq* contenant ces informations vers sa station de base courante BS_j. À la réception d'une telle requête, BS_j sauvegarde les informations de reprise sur son support stable.

Afin de minimiser l'overhead de stockage, les données de recouvrement obsolètes du site MH_i doit être supprimées. Pour ce faire, BS_j envoie une requête *ChkptDeleteReq* vers la station de base indiquée par $Chkpt\_loc_i^j$ et qui contient le dernier point de reprise. Elle envoie aussi une requête *LogDeleteReq* vers les stations de base qui se trouvent dans $Log\_set_i^j$ et contenant le journal de messages. Lorsqu'une station de base reçoit l'une des deux requêtes, elle supprime les données de recouvrement correspondantes. La valeur de $RcvMsg\_sn_i$ est utilisée pour déterminer la positon correcte du point de reprise par rapport aux messages journalisés dans $MsgLog_i^j$. Ainsi, BS_j supprime aussi tous les messages de données dont le numéro de séquence est inférieur ou égal à $RcvMsg\_sn_i$. Après la suppression des données de recouvrement obsolètes, BS_j met à jour l'information de localisation du dernier point de reprise $Chkpt\_loc_i^j$ ainsi que celle de messages journalisés $Log\_set_i^j$.

Voici le pseudo-code décrivant la sauvegarde de points de reprise périodiques.

Rôle du site mobile MH_i :

- Lorsque $Chkpt\_timer_i$ expire :
  - $Chkpt\_sn_i = Chkpt\_sn_i + 1$;
  - Prendre un point de reprise stable $Chkpt_i$;
  - Enregistrer ($Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) dans $LocalChkpt\_space_i$;
  - Envoyer *SaveChkptReq* (MH_i , $Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) à BS_j ;
  - Réinitialiser $Chkpt\_timer_i$;

Rôle de la station de base BS_j :

• Lorsque BS_j reçoit *SaveChkptReq* (MH_i , $Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) :
Enregistrer ($Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) dans $Chkpt\_space_i^j$;
Supprimer les données de recouvrement obsolètes de MH_i ;

• Lorsque BS_j supprime les données de recouvrement obsolètes de MH_i :
Si ($Chkpt\_loc_i^j \neq BS_j$) Alors
Envoyer *ChkptDeleteReq* (BS_j , MH_i) à $Chkpt\_loc_i^j$;
FinSi
Pour chaque ($BS_k \in Log\_set_i^j$ et $BS_k \neq BS_j$) Faire
Envoyer *LogDeleteReq* (BS_j , MH_i) à BS_k ;
FinPour
Si ($BS_j \in Log\_set_i^j$) Alors
Supprimer chaque $(\alpha , M) \in MsgLog_i^j$ avec $\alpha \leq RcvMsg\_sn_i$;
FinSi
Si ($MsgLog_i^j = NULL$) Alors $Log\_set_i^j = \varnothing$;
Sinon $Log\_set_i^j = \{BS_j\}$;
FinSi
$Chkpt\_loc_i^j = BS_j$;

Rôle de la station de base BS_k contenant des données de recouvrement du site MH_i :

• Lorsque BS_k reçoit *ChkptDeleteReq* (BS_j , MH_i) :
Supprimer $Chkpt\_space_i^k$ du support stable de BS_k ;

• Lorsque BS_k reçoit *LogDeleteReq* (BS_j , MH_i) :
Supprimer $Log\_space_i^k$ du support stable de BS_k ;

4.4.3.2 Journalisation de messages

Chaque station de base effectue la journalisation de messages des sites mobiles se trouvant dans sa cellule de couverture. Ainsi, lors de la délivrance d'un message de données destiné vers MH_i, sa station de base BS_j incrémente le numéro de séquence du dernier message journalisé $LogMsg\_sn_i^j$, puis sauvegarde le message avec son numéro de séquence dans le journal de messages $MsgLog_i^j$ pour le site MH_i. À la journalisation du premier message destiné à MH_i, l'identificateur de BS_j est ajouté à l'ensemble $Log\_set_i^j$.

À la réception d'un message de données de BS_j, le site MH_i incrémente le numéro de séquence du dernier message reçu $RcvMsg\_sn_i$ et délivre le message au processus P_i s'exécutant sur MH_i. Chaque message reçu par MH_i est considéré comme un message délivré au processus P_i. Donc, $LogMsg\_sn_i^j$ et $RcvMsg\_sn_i$ possèdent toujours la même valeur.

Voici le pseudo-code décrivant la journalisation de messages.

Rôle de la station de base BS_j :

• Lorsque BS_j délivre un message de données M vers MH_i :

$LogMsg\_sn_i^j = LogMsg\_sn_i^j + 1$;

Enregistrer ($LogMsg\_sn_i^j$, M) dans $MsgLog_i^j$;

Si ($BS_j \notin Log\_set_i^j$) Alors

$Log\_set_i^j = Log\_set_i^j \cup \{BS_j\}$;

FinSi

Délivrer M au site MH_i ;

Rôle du site mobile MH_i :

• Lorsque MH_i reçoit un message de données M :

$RcvMsg\_sn_i = RcvMsg\_sn_i + 1$;

Délivrer M au processus P_i ;

4.4.3.3 Traitement du handoff

Lorsque la distance entre le site MH_i et sa station de base courante BS_j notée $Distance_i$ dépasse le seuil du handoff $HandoffThreshold_i$ et qu'il existe d'autres stations de base pouvant servir le site MH_i ($AvailableBS\_set_i \neq \emptyset$), celui-ci va effectuer un handoff de l'ancienne BS_j vers la nouvelle BS_k. $Distance_i$ est calculée par la formule $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ tels que (x_i, y_i) et (x_j, y_j) sont les positions de MH_i et BS_j respectivement. Lors d'un handoff, le site MH_i prend un point de reprise local ($Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) et le sauvegarde sur son support stable local $LocalChkpt\_space_i$. Ensuite, MH_i envoie une requête *LeaveReq* à BS_j et une requête *JoinReq* à BS_k. À la réception d'une requête *LeaveReq* de MH_i, BS_j arrête la délivrance des messages de données vers MH_i, sauvegarde le message (*"leave"*) dans $MsgLog_i^j$ et supprime l'identificateur de MH_i de l'ensemble des sites actifs $ActiveMH\_set_j$ sur lesquels s'exécute l'application distribuée.

À la réception d'une requête *JoinReq* de MH_i, BS_k envoie une requête de handoff *HandoffReq* à BS_j. Lorsque BS_j reçoit une telle requête de BS_K, elle répond par un message *MHInfo* contenant les informations de localisation des données de recouvrement $Chkpt\_loc_i^j$ et $Log\_set_i^j$, ainsi que le numéro de séquence $LogMsg\_sn_i^j$ du dernier message journalisé dans $MsgLog_i^j$. Ensuite, elle met à jour la nouvelle localisation de MH_i et envoie ses messages de données non encore délivrés vers BS_k.

À la réception d'un message *MHInfo* de BS_j, BS_k sauvegarde $Chkpt\_loc_i^j$, $Log\_set_i^j$ et $LogMsg\_sn_i^j$ dans $Chkpt\_loc_i^k$, $Log\_set_i^k$ et $LogMsg\_sn_i^k$ respectivement, et ajoute l'identificateur de MH_i à l'ensemble des sites actifs $ActiveMH\_set_j$. Ensuite, elle envoie une confirmation

de handoff *HandoffConfirm* à MH_i. Lorsque MH_i reçoit une telle confirmation, il met BS_j comme ancienne station de base $OldBS_i$ et BS_k comme nouvelle station de base $NewBS_i$.

Voici le pseudo-code décrivant le traitement du handoff.

Rôle du site mobile MH_i :

- Lorsque ($Distance_i > HandoffThreshold_i$) et ($AvailableBS\_set_i \neq \emptyset$) :
  $Chkpt\_sn_i = Chkpt\_sn_i + 1$;
  Prendre un point de reprise local $Chkpt_i$;
  Enregistrer ($Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) dans $LocalChkpt\_space_i$;
  Envoyer *LeaveReq* (MH_i) à BS_j ;
  Envoyer *JoinReq* (MH_i , BS_j) à BS_k ;
- Lorsque MH_i reçoit *HandoffConfirm* (BS_k) :
  $OldBS_i = NewBS_i$; $NewBS_i = BS_k$;

Rôle de l'ancienne station de base BS_j :

- Lorsque BS_j reçoit *LeaveReq* (MH_i) :
  Arrêter de délivrer les messages de données vers MH_i ;
  Enregistrer ($LogMsg\_sn_i^j$, *"leave"*) dans $MsgLog_i^j$;
  $ActiveMH\_set_j = ActiveMH\_set_j$ - $\{MH_i\}$;
- Lorsque BS_j reçoit *HandoffReq* (BS_k , MH_i) :
  Envoyer *MHInfo* (BS_j , MH_i , $Chkpt\_loc_i^j$, $Log\_set_i^j$, $LogMsg\_sn_i^j$) à BS_k ;
  Mettre à jour la nouvelle localisation de MH_i ;
  Envoyer les messages de données non encore délivrés à MH_i vers BS_k ;

Rôle de la nouvelle station de base BS_k :

- Lorsque BS_k reçoit *JoinReq* (MH_i , BS_j) :
  Envoyer *HandoffReq* (BS_k , MH_i) à BS_j ;
- Lorsque BS_k reçoit *MHInfo* (BS_j , MH_i , $Chkpt\_loc_i^j$, $MsgLog\_set_i^j$, $LogMsg\_sn_i^j$) :
  $Chkpt\_loc_i^k = Chkpt\_loc_i^j$;
  $Log\_set_i^k = Log\_set_i^j$;
  $LogMsg\_sn_i^k = LogMsg\_sn_i^j$;
  $ActiveMH\_set_k = ActiveMH\_set_k \cup \{MH_i\}$;
  Envoyer *HandoffConfirm* (BS_k) à MH_i ;

4.4.3.4 Recouvrement par migration de processus

Chaque site mobile sauvegarde un point de reprise de migration à la détection des événements précédant l'occurrence des fautes suivantes : l'épuisement de la batterie, la déconnexion due à la couverture discontinue et la déconnexion volontaire. La batterie va s'épuiser, si le

niveau d'énergie $EnergyLevel_i$ du site MH_i atteint le seuil critique $EnergyThreshold_i$. Le site MH_i va sortir de la zone de couverture (cellule) de BS_j, si la distance entre le site MH_i et sa station de base courante BS_j (notée $Distance_i$) dépasse le seuil du handoff $HandoffThreshold_i$ et qu'il n'existe aucune autre station de base pouvant servir MH_i ($AvailableBS\_set_i = \varnothing$). La déconnexion volontaire du site MH_i est initialisée par son utilisateur.

À la détection de tels événements, le site MH_i prend un point de reprise de migration ($Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) et l'envoie via une requête de migration de processus *ProcMigrReq* à sa station de base courante BS_j. À la réception d'une requête *ProcMigrReq* du site MH_i, BS_j restaure le point de reprise $Chkpt_i$, poursuivre l'exécution du processus P_i et supprime l'identificateur de MH_i de l'ensemble des sites actifs $ActiveMH\_set_j$. Ensuite, elle déclenche le temporisateur $Planned\_timer_i^j$ et attend la reconnexion du site MH_i.

Lors de sa reconnexion à la station de base BS_j, le site MH_i lui envoie une requête de recouvrement *RecoveryReq*. S'il se reconnecte à une nouvelle station de base BS_k, il lui envoie la requête avec l'identificateur de l'ancienne station de base $OldBS_i$ (BS_j). À la réception d'une telle requête, BS_k envoie une requête *RecoveryInfoReq* à BS_j pour obtenir les informations de recouvrement. On distingue deux cas lors de la réception d'une requête *RecoveryInfoReq* (ou une requête *RecoveryReq*) par BS_j.

A. Avant l'expiration de $Planned\_timer_i^j$:

Lorsque BS_j reçoit une requête *RecoveryReq* de MH_i, elle arrête $Planned\_timer_i^j$, exécute la procédure permettant de prendre un point de reprise stable et ajoute l'identificateur de MH_i à l'ensemble des sites actifs $ActiveMH\_set_j$. Ensuite, elle répond à MH_i par un message de confirmation de recouvrement *RecoveryConfirm*.

Lorsque BS_j reçoit une requête *RecoveryInfoReq* de BS_k, elle arrête $Planned\_timer_i^j$, exécute la procédure permettant de prendre un point de reprise stable et répond à BS_k par un message *RecoveryInfo* contenant ce point de reprise ($Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$). Après, elle met à jour la nouvelle localisation de MH_i et envoie les messages de données non encore délivrés vers BS_k. À la réception des informations de recouvrement *RecoveryInfo*, BS_k met à jour les informations de localisation des données de recouvrement, ajoute l'identificateur de MH_i à l'ensemble des sites actifs $ActiveMH\_set_k$ et envoie une confirmation de recouvrement *RecoveryConfirm* à MH_i. À la réception d'une telle confirmation de recouvrement de BS_j (ou BS_k), MH_i sauvegarde le point de reprise sur son support stable local, réinitialise le temporisateur $Chkpt\_timer_i$, restaure $Chkpt_i$ et poursuit l'exécution du processus P_i.

B. Après l'expiration de $Planned\_timer_i^j$:

À l'expiration de $Planned\_timer_i^j$ pour MH_i, BS_j exécute la procédure *MigrateToNewMH* pour élire un site de support MH_R et lui envoyer un message *RecoveryInfo* contenant les informations de recouvrement. Lorsque BS_j reçoit une requête *RecoveryReq* de MH_i (resp. *RecoveryInfoReq* de BS_k), elle ajoute l'identificateur de MH_i à l'ensemble des sites de support $SupportMH\_set_j$ (resp. elle répond par un message *ProcMigrated* informant BS_k que le processus du site MH_i est migré vers un autre site mobile). À la réception du message *ProcMigrated*, BS_k ajoute l'identificateur de MH_i à l'ensemble des sites de support $SupportMH\_set_k$.

Voici le pseudo-code qui décrit le recouvrement par migration de processus.

Rôle du site mobile MH_i susceptible défaillant :

- Lorsque ($EnergyLevel_i \leq EnergyThreshold_i$) :
  $Chkpt\_sn_i = Chkpt\_sn_i + 1$;
  Prendre un point de reprise de migration $Chkpt_i$;
  Envoyer *ProcMigrReq* (MH_i , $Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) à BS_j ;
  Signaler l'épuisement de batterie à l'utilisateur via un message sonore ;
  Arrêter le système ;

- Lorsque ($Distance_i > HandoffThreshold_i$) et ($AvailableBS\_set_i = \emptyset$) :
  $Chkpt\_sn_i = Chkpt\_sn_i + 1$;
  Prendre un point de reprise de migration $Chkpt_i$;
  Envoyer *ProcMigrReq* (MH_i , $Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) à BS_j ;
  Signaler la sortie de la zone de couverture à l'utilisateur via un message sonore ;

- Avant une déconnexion volontaire de MH_i :
  $Chkpt\_sn_i = Chkpt\_sn_i + 1$;
  Prendre un point de reprise de migration $Chkpt_i$;
  Envoyer *ProcMigrReq* (MH_i , $Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) à BS_j ;
  Signaler la déconnexion volontaire du réseau ;

- Lorsque MH_i se reconnecte à BS_k :
  $OldBS_i = NewBS_i$; $NewBS_i = BS_k$;
  Si ($OldBS_i = NewBS_i$) Alors Envoyer *RecoveryReq* (MH_i) à BS_k ;
  Sinon Envoyer *RecoveryReq* (MH_i , $OldBS_i$) à BS_k ;
  FinSi

- Lorsque MH_i (ou MH_R) reçoit *RecoveryConfirm* (BS_k , $Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$) :
  Enregistrer ($Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$) dans $LocalChkpt\_space_i$;
  Réinitialiser $Chkpt\_timer_i$;
  Créer un processus P_i ;
  Restaurer le point de reprise $Chkpt_i$;
  Poursuivre l'exécution de P_i ;

Rôle de la station de base BS_j de déconnexion :

• Lorsque BS_j reçoit *ProcMigrReq* (MH_i , $Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) :

Créer un processus P_i ;

Restaurer le point de reprise de migration $Chkpt_i$;

Poursuivre l'exécution de P_i ;

$ActiveMH\_set_k = ActiveMH\_set_k$ - $\{MH_i\}$;

Déclencher $Planned\_timer_i^j$ pour MH_i ;

Attendre la reconnexion de MH_i ;

• Lorsque BS_j reçoit *RecoveryReq* (MH_i) avant expiration de $Planned\_timer_i^j$:

Arrêter $Planned\_timer_i^j$ pour MH_i ;

TakeStableChkpt (MH_i , BS_j) ;

$ActiveMH\_set_k = ActiveMH\_set_k \cup \{MH_i\}$;

Envoyer *RecoveryConfirm* (BS_j , $Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$) à MH_i ;

• Lorsque BS_j reçoit *RecoveryReq* (MH_i) après expiration de $Rec\_timer_i^j$:

$SupportMH\_set_j = SupportMH\_set_j \cup \{MH_i\}$;

• Lorsque BS_j reçoit *RecoveryInfoReq* (BS_k , MH_i) avant expiration de $Planned\_timer_i^j$:

Arrêter $Planned\_timer_i^j$ pour MH_i ;

TakeStableChkpt (MH_i , BS_j) ;

Envoyer *RecoveryInfo* (BS_j , MH_i , $Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$) à BS_K ;

Mettre à jour la nouvelle localisation de MH_i ;

Envoyer les messages de données non encore délivrés à MH_i vers BS_k ;

• Lorsque BS_j reçoit *RecoveryInfoReq* (BS_k , MH_i) après expiration de $Planned\_timer_i^j$:

Envoyer *ProcMigrated* (BS_j , MH_i) à BS_k ;

• Lorsque $Planned\_timer_i^j$ pour MH_i expire :

MigrateToNewMH (MH_i , BS_j) ;

Procedure *TakeStableChkpt* (MH_i , BS_j)

$Chkpt\_sn_i = Chkpt\_sn_i + 1$;

Prendre un point de reprise stable $Chkpt_i$ pour le processus P_i ;

Enregistrer ($Chkpt_i$, $Chkpt\_sn_i$, $RcvMsg\_sn_i$) dans $Chkpt\_space_i^j$;

Supprimer les données de recouvrement obsolètes de MH_i ;

Procedure *MigrateToNewMH* (MH_i , BS_j)

Élire un site de reprise MH_R ;

TakeStableChkpt (MH_i , BS_j) ;

Envoyer *RecoveryInfo* (BS_j , $Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$) à MH_R ;

Diffuser *ChangeIDReq* (MH_i , MH_R) à tous les sites MH_p ;

Délivrer les messages de données destinés à MH_i vers MH_R ;

Rôle de la station de base BS_k de reconnexion :

- Lorsque BS_k reçoit *RecoveryReq* (MH_i , BS_j) :
  Envoyer *RecoveryInfoReq* (BS_k , MH_i , BS_j) à BS_j ;
- Lorsque BS_k reçoit *RecoveryInfo* (BS_j , MH_i , $Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$) :
  $Chkpt\_loc_i^k = BS_j$;
  $Log\_set_i^k = \varnothing$;
  $LogMsg\_sn_i^k = LogMsg\_sn_i^j$;
  $ActiveMH\_set_k = ActiveMH\_set_k \cup \{MH_i\}$;
  Envoyer *RecoveryConfirm* (BS_j , $Chkpt_i^j$, $Chkpt\_sn_i^j$, $LogMsg\_sn_i^j$) à MH_i ;
- Lorsque BS_k reçoit *ProcMigrated* (BS_j , MH_i) :
  $SupportMH\_set_k = SupportMH\_set_k \cup \{MH_i\}$;

4.4.3.5 Recouvrement d'un site défaillant

Quand le site MH_i se déconnecte d'une manière imprévue (non planifiée), sa station de base courante déclenche le temporisateur $Unplanned\_timer_i^j$ et attend sa reconnexion. Si MH_i se reconnecte avant l'expiration de $Unplanned\_timer_i^j$, il subit une faute de déconnexion transitoire, sinon il est affecté par une faute permanente due aux dommages physiques.

A. Recouvrement d'une faute de déconnexion transitoire

Lors de la reconnexion du site MH_i à BS_j, il lui envoie une requête de recouvrement *LocalRecoveryReq*. S'il se reconnecte à une nouvelle station de base BS_k, il lui envoie la requête avec l'identificateur de l'ancienne station de base $OldBS_i$ (BS_j) ainsi que le numéro de séquence du dernier message reçu *RcvMsg_sn* qui correspond au point de reprise sauvegardé sur le support stable local. À la réception d'une telle requête, BS_k envoie une requête *LocalRecoveryInfoReq* à BS_j pour obtenir les informations de recouvrement.

Lorsque BS_j reçoit une requête de recouvrement local *LocalRecoveryReq* de MH_i, elle arrête $Unplanned\_timer_i^j$, ajoute l'identificateur du site MH_i à l'ensemble des sites actifs $ActiveMH\_set_j$ et envoie une confirmation de recouvrement local *LocalRecoveryConfirm* à MH_i. Ensuite, elle lui délivre dans l'ordre tous les messages journalisés dans $MsgLog_i^j$.

Lorsque BS_j reçoit une requête de recouvrement local *LocalRecoveryInfoReq* de BS_k, elle répond d'abord par un message *MHInfo* contenant les informations de localisation des données de recouvrement $Chkpt\_loc_i^j$ et $Log\_set_i^j$, ainsi que le numéro de séquence $LogMsg\_sn_i^j$ du dernier message journalisé dans $MsgLog_i^j$. À la réception d'un tel message, BS_k sauvegarde $Chkpt\_loc_i^j$, $Log\_set_i^j$ et $LogMsg\_sn_i^j$ dans $Chkpt\_loc_i^k$, $Log\_set_i^k$ et $LogMsg\_sn_i^k$ respectivement, et ajoute l'identificateur de MH_i à l'ensemble des sites actifs $ActiveMH\_set_j$.

Par la suite, BS_j envoie à BS_k un message *LocalRecoveryInfo* contenant tous les messages de données journalisés dont le numéro de séquence est supérieur à $RcvMsg\_sn_i$. Après, elle met à jour la nouvelle localisation de MH_i et envoie ses messages de données non encore délivrés vers BS_k. Lorsque BS_k reçoit le message *LocalRecoveryInfo* de BS_j, elle envoie une confirmation de recouvrement local *LocalRecoveryConfirm* à MH_i, puis elle lui délivre dans l'ordre les messages journalisés. À la réception d'une telle confirmation de BS_j (ou BS_k), MH_i restaure son point de reprise local $Chkpt_i$ et reprendre l'exécution du processus P_i.

B. Recouvrement d'une faute due aux dommages physiques (permanente)

Lorsque le temporisateur $Unplanned\_timer_i^j$ expire, le MH_i est affecté par une faute permanente. Par conséquent, sa station de base courante BS_j doit effectuer le recouvrement de son processus P_i, puis un site de support doit être utilisé pour jouer le rôle de MH_i en poursuivant l'exécution de son processus P_i.

Ainsi, à l'expiration de $Unplanned\_timer_i^j$, BS_j collecte les informations de recouvrement de MH_i en envoyant une requête *ChkptRetrieveReq* vers la station de base $Chkpt\_loc_i^j$ contenant le point de reprise et une autre requête *LogRetrieveReq* vers chaque station de base appartient à $Log\_set_i^j$. À la réception de telles requêtes par BS_p contenant des données de recouvrement de MH_i, elle envoie ces données à BS_j, puis les supprime de son support stable.

À la réception et la sauvegarde de toutes les informations de recouvrement de MH_i, BS_j met à jour les informations de localisation, restaure le point de reprise $Chkpt_i^j$, puis effectue le recouvrement de P_i en rejouant dans l'ordre tous les messages qui se trouvent dans $MsgLog_i^j$. Par la suite, BS_j exécute la procédure *MigrateToNewMH* pour élire un site de support MH_R et lui envoyer un message *RecoveryInfo* contenant les informations de recouvrement.

Voici le pseudo-code qui décrit le recouvrement d'un site défaillant.

Rôle du site mobile MH_i défaillant :

- Lorsque MH_i se reconnecte à BS_k :

  $OldBS_i = NewBS_i$; $NewBS_i = BS_k$;

  Si ($OldBS_i = NewBS_i$) Alors Envoyer *LocalRecoveryReq* (MH_i) à BS_k ;

  Sinon Envoyer *LocalRecoveryReq* (MH_i , $RcvMsg\_sn_i$, $OldBS_i$) à BS_k ;

  FinSi

- Lorsque MH_i reçoit *LocalRecoveryConfirm* (BS_k) :

  Créer un processus P_i ;

  Restaurer le point de reprise local $Chkpt_i$;

  Reprendre l'exécution de P_i ;

Rôle de la station de base BS_j de déconnexion :

• Lorsque BS_j détecte une déconnexion imprévue de MH_i :

$ActiveMH\_set_k = ActiveMH\_set_k$ - $\{MH_i\}$;

Déclencher $Unplanned\_timer_i^j$ pour MH_i ;

Attendre la reconnexion de MH_i ;

• Lorsque BS_j reçoit *LocalRecoveryReq* (MH_i) avant expiration de $Unplanned\_timer_i^j$:

Arrêter $Unplanned\_timer_i^j$ pour MH_i ;

$ActiveMH\_set_k = ActiveMH\_set_k \cup \{MH_i\}$;

Envoyer *LocalRecoveryConfirm* (BS_j) à MH_i ;

Envoyer chaque message $M \in MsgLog_i^j$ à MH_i ;

• Lorsque BS_j reçoit *LocalRecoveryInfoReq* (BS_k , MH_i) avant expiration de $Unplanned\_timer_i^j$:

Arrêter $Unplanned\_timer_i^j$ pour MH_i ;

Envoyer *MHInfo* (BS_j , MH_i , $Chkpt\_loc_i^j$, $Log\_set_i^j$, $LogMsg\_sn_i^j$) à BS_K ;

Si ($RcvMsg\_sn_i < LogMsg\_sn_i^j$) Alors

  Envoyer *LocalRecoveryInfo* (BS_j , MH_i , $MsgLog_i^j$) à BS_K ;

Sinon

  Envoyer *LocalRecoveryInfo* (BS_j , MH_i , $\emptyset$) à BS_K ;

FinSi

Mettre à jour la nouvelle localisation de MH_i ;

Envoyer les messages de données non encore délivrés à MH_i vers BS_k ;

• Lorsque $Unplanned\_timer_i^j$ pour MH_i expire :

Si ($Chkpt\_loc_i^j \neq BS_j$) Alors

  Envoyer *ChkptRetrieveReq* (BS_j , MH_i) à $Chkpt\_loc_i^j$;

FinSi

Pour chaque ($BS_p \in Log\_set_i^j$ et $BS_p \neq BS_j$) Faire

  Envoyer *LogRetrieveReq* (BS_j , MH_i) à BS_p ;

FinPour

• Lorsque BS_j reçoit *MHChkpt* (BS_m , MH_i , $Chkpt_i^m$, $Chkpt\_sn_i^m$, $RcvMsg\_sn_i^m$) :

Enregistrer ($Chkpt_i^m$, $Chkpt\_sn_i^m$, $RcvMsg\_sn_i^m$) dans $Chkpt\_space_i^j$;

• Lorsque BS_j reçoit *MHMsgLog* (BS_p , MH_i , $MsgLog_i^p$) :

$MsgLog_i^j = MsgLog_i^j \cup MsgLog_i^p$;

• Lorsque BS_j collecte toutes les informations de recouvrement de MH_i :

$Chkpt\_loc_i^k = BS_j$; $Log\_set_i^k = \{BS_j\}$;

Créer un processus P_i ;

Restaurer le point de reprise stable $Chkpt_i^j$;

Reprendre l'exécution de P_i en rejouant tous les messages $M \in MsgLog_i^j$;

MigrateToNewMH (MH_i , BS_j) ;

Rôle de la station de base BS_k de reconnexion :

- Lorsque BS_k reçoit *RecoveryReq* (MH_i , $RcvMsg\_sn_i$, BS_j) :
  Envoyer *LocalRecoveryInfoReq* (BS_k , MH_i , $RcvMsg\_sn_i$) à BS_j ;
- Lorsque BS_k reçoit *MHInfo* (BS_j , MH_i , $Chkpt\_loc_i^j$, $Log\_set_i^j$, $LogMsg\_sn_i^j$) :
  $Chkpt\_loc_i^k = Chkpt\_loc_i^j$;
  $Log\_set_i^k = Log\_set_i^j$;
  $LogMsg\_sn_i^k = LogMsg\_sn_i^j$;
  $ActiveMH\_set_k = ActiveMH\_set_k \cup \{MH_i\}$;
- Lorsque BS_k reçoit *LocalRecoveryInfo* (BS_j , MH_i , $MsgLog_i^j$) :
  Envoyer *LocalRecoveryConfirm* (BS_k) à MH_i ;
  Envoyer chaque message $M \in MsgLog_i^j$ à MH_i ;
- Lorsque BS_k reçoit *LocalRecoveryInfo* (BS_j , MH_i , $\emptyset$) :
  Envoyer *LocalRecoveryConfirm* (BS_k) à MH_i ;

Rôle de la station de base BS_p contenant des données de recouvrement du site MH_i :

- Lorsque BS_p reçoit *ChkptRetrieveReq* (BS_j , MH_i) :
  Envoyer *MHChkpt* (BS_p , MH_i , $Chkpt_i^p$, $Chkpt\_sn_i^p$, $LogMsg\_sn_i^p$) à BS_j ;
  Supprimer $Chkpt\_space_i^p$ du support stable de BS_p ;
- Lorsque BS_p reçoit *LogRetrieveReq* (BS_j , MH_i) :
  Envoyer *MHMsgLog* (BS_p , MH_i , $MsgLog_i^p$) à BS_j ;
  Supprimer $Log\_space_i^p$ du support stable de BS_p ;

4.4.3.6 Élection d'un site de reprise (support)

Quand un site MH_i est affecté par une faute permanente ou effectue une déconnexion planifiée et dépasse le délai de déconnexion défini, sa station de base BS_j lance une élection d'un site de reprise (support) MH_R pour remplacer le site MH_i. Lors de l'initialisation d'une élection, BS_j diffuse une requête d'élection *ElectionReq* à chaque site de support MH_k appartient à $SupportMH\_set_j$. Ensuite, elle déclenche le temporisateur $Election\_timer_i^j$ et attend l'arrivée des candidatures de MH_k.

À la réception d'une telle requête, le site MH_k vérifie si son niveau d'énergie $EnergyLevel_k$ est acceptable, c-à-d il est supérieur à $AcceptEnergyLevel_k$. Si c'est le cas, il calcule la distance $Distance_k$ qui le sépare de BS_j par la formule $\sqrt{(x_k - x_j)^2 + (y_k - y_j)^2}$, puis répond par un message *CandAccept* contenant son niveau d'énergie et cette distance.

Le choix d'un candidat s'effectue selon deux critères : le niveau d'énergie et la stabilité du site mobile. Plus qu'un site est proche de sa station de base, plus que son niveau de stabilité augmente. Ainsi, le niveau de la stabilité d'un site MH_k connecté à BS_j est calculé par la

formule $TransmRange_j$ - $Distance_k$ / $TransmRange_j$. Les deux critères sont combinés par la fonction $Func = \alpha * EnergyLevel_k + \beta * (TransmRange_j - Distance_k / TransmRange_j)$, tel que α et β sont les facteurs de pondération pour le niveau d'énergie et le niveau de stabilité respectivement. Lorsque BS_j reçoit une acceptation de candidature *CandAccept*, elle calcule la valeur *Func* correspondante et ajoute (MH_k , $EnergyLevel_k$, $Distance_k$, *Func*) à la liste des candidats $Cand\_list_i^j$ et incrémente le nombre de candidatures reçues $Cand\_nb_i^j$.

À l'expiration de $Election\_timer_i^j$, si aucune candidature n'est parvenue à BS_j, elle réinitialise une nouvelle élection. Dans le cas de la réception d'une seule candidature de MH_k, celui-ci est sélectionné comme un site de reprise. En revanche, si plusieurs candidatures sont parvenues, la procédure *SelectCand* permettant de sélectionner le meilleur candidat parmi ceux se trouvant dans $Cand\_list_i^j$ est exécutée.

Voici le pseudo-code qui décrit l'élection d'un site de reprise.

Rôle de la station de base BS_j initialisant l'élection d'un site de reprise :

• Lorsque BS_j initialise une élection d'un site de reprise MH_R :

Diffuser *ElectionReq* (BS_j) à chaque $MH_k \in SupportMH\_set_j$;
Déclencher $Election\_timer_i^j$ pour BS_j ;
Attendre l'arrivée des candidatures de MH_k ;

• Lorsque BS_j reçoit *CandAccept* (MH_k , $EnergyLevel_k$, $Distance_k$) :

$Func = \alpha * EnergyLevel_k + \beta * (TransmRange_j - Distance_k / TransmRange_j)$;
Enregistrer (MH_k , $EnergyLevel_k$, $Distance_k$, *Func*) dans $Cand\_list_i^j$ [$Cand\_nb_i^j$] ;
$Cand\_nb_i^j = Cand\_nb_i^j + 1$;

• Lorsque $Election\_timer_i^j$ expire :

Si ($Cand\_nb_i^j = 0$) Alors
  Réinitialiser une nouvelle élection d'un site de reprise MH_R ;
Sinon
  Si ($Cand\_nb_i^j = 1$) Alors $ElectedCand_i^j = Cand\_list_i^j$ [0] ;
  Sinon *SelectCand* ($Cand\_list_i^j$) ;
  FinSi
Finsi

Procedure *SelectCand* ($Cand\_list_i^j$)

$ElectedCand_i^j = Cand\_list_i^j$ [0] ;
Pour n = 1 jusqu'à n = $Cand\_nb_i^j$-1 Faire
  Si ($Cand\_list_i^j$ [n].*Func* > $ElectedCand_i^j$.*Func*) Alors $ElectedCand_i^j = Cand\_list_i^j$ [n] ;
  FinSi
FinPour

Rôle du site mobile de reprise MH_k qui reçoit une requête d'élection :

• Lorsque MH_k reçoit *ElectionReq* (BS_j) :

Si ($EnergyLevel_k \geq AcceptEnergyLevel_k$) Alors

$Distance_k = \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2}$;

Envoyer *CandAccept* (MH_k , $EnergyLevel_k$, $Distance_k$) à BS_j ;

FinSi

4.5 Analyse des performances

Notre protocole se base sur la reprise par journalisation pessimiste de messages et utilise le support stable de la station de base pour sauvegarder le journal de messages ainsi que les points de reprise périodiques. Par conséquent, il fournit les avantages suivants :

- Un seul point de reprise doit être sauvegardé pour chaque site. Donc, lors du recouvrement le site défaillant redémarre son exécution à partir de ce point de reprise, ce qui lui permet d'éviter le problème de l'effet domino.
- La défaillance d'un site n'affecte pas les processus des autres sites. Ainsi, seul le site défaillant doit effectuer son recouvrement (recouvrement indépendant).
- La sauvegarde d'un point de reprise n'implique aucune forme de coordination, ce qui permet d'éviter la surcharge du réseau avec des messages de coordination ainsi que l'overhead temporel associé à leur traitement par un site mobile.
- La journalisation de messages s'effectue au niveau de la station de base et n'implique aucun overhead de communication. De plus, un site mobile évite l'overhead temporel induit par l'accès synchrone au support stable local.

L'analyse des performances du protocole proposé (Proposed Protocol) est effectuée en évaluant le surcoût (overhead) induit par celui-ci lors du recouvrement ainsi que pendant l'exécution sans faute. Les résultats de notre protocole sont comparés avec ceux d'un protocole de reprise par journalisation pessimiste de messages appelé protocole de base (Basic Protocol) et qui utilise seulement des points de reprise périodiques. On suppose qu'un site mobile ne soit pas affecté par des fautes durant son recouvrement.

Les différents paramètres utilisés dans l'évaluation sont :

- $MTBF$ (Mean Time Between Failures) : le temps moyen entre deux fautes (pannes) consécutives de n'importe quel type.
- $T_{execution}$: la durée d'exécution globale de l'application distribuée ou d'un site mobile.
- T_{chkpt} : l'intervalle de temps pour la sauvegarde d'un point de reprise périodique.

- Log_{rate} : la fréquence de journalisation de messages.
- H_{rate} : la fréquence d'occurrence d'un handoff.
- $T_{unplanDis}$: la durée de persistance moyenne d'une faute de déconnexion non planifiée (transitoire).
- $T_{unplanDisMax}$: la durée de persistance maximale d'une faute de déconnexion non planifiée (transitoire).
- F_{perm} : le taux d'occurrence d'une faute permanente.
- F_{temp} : le taux d'occurrence d'une faute transitoire planifiée ou non planifiée (imprévue).
- F_{plan} : le taux d'occurrence d'une faute transitoire planifiée.
- F_{unplan} : le taux d'occurrence d'une faute de déconnexion imprévue (transitoire).
- $F_{unplanNotH}$: le taux d'occurrence d'une faute de déconnexion imprévue avant ou après un handoff.
- $F_{unplanH}$: le taux d'occurrence d'une faute de déconnexion imprévue lors d'un handoff.

4.5.1 Le surcoût de recouvrement

Deux métriques sont calculées pour déterminer l'overhead induit par un site mobile lors de son recouvrement : l'overhead temporel (temps d'exécution) et l'overhead de communication (nombre de messages).

4.5.1.1 Le surcoût de temps d'exécution

Lorsqu'un site est affecté par une faute, il doit reprendre son exécution à partir de son dernier point de reprise. Donc, l'overhead de temps d'exécution engendré lors du recouvrement dépend du type de la faute ou du point de reprise utilisé.

L'overhead induit par une faute permanente est égal au temps de persistance maximal d'une faute de déconnexion non planifiée plus la moitié de l'intervalle de temps pour la sauvegarde d'un point de reprise périodique :

$Overhead_{perm} = T_{unplanDisMax} + \frac{T_{chkpt}}{2}$

L'overhead impliqué par une faute de déconnexion imprévue dépend du moment de son occurrence. Lorsque la faute se produit avant ou après un handoff, l'overhead est égal au temps moyen d'une déconnexion imprévue plus la moitié de la durée moyenne entre deux handoff : $T_{unplanDis} + \frac{1}{2 \cdot H_{rate}}$. Quand la faute se produit lors d'un handoff, l'overhead est égal au temps moyen d'une déconnexion imprévue : $T_{unplanDis}$. Donc, l'overhead induit par une déconnexion imprévue est donné par la formule suivante :

$Overhead_{unplanDis} = F_{unplanH} \cdot T_{unplanDis} + F_{unplanNotH} \cdot (T_{unplanDis} + \frac{1}{2 \cdot H_{rate}})$

L'overhead induit par une faute planifiée est nul ($Overhead_{plannedDis} = 0$), parce que le site mobile susceptible défaillant effectue la migration de ses processus vers sa station de base qui poursuit leur traitement (une exécution non-stop). Le temps de migration est négligé.

Par conséquent, l'overhead total de temps d'exécution induit par l'occurrence d'une faute de n'importe quel type est calculé par la formule suivante :

$Overhead_{total}^{Proposed} = (F_{perm} \cdot Overhead_{perm} + F_{temp} \cdot F_{unplan} \cdot Overhead_{unplanDis}) \cdot \frac{T_{execution}}{MTBF}$

Cet overhead exprime le temps d'exécution additionnel induit par les différentes fautes. Il peut aussi être exprimé par un taux représentant le pourcentage de ce temps additionnel par rapport au temps d'exécution global. Donc, le taux de l'overhead est calculé comme suit :

$Overhead_{rate}^{Proposed} = (F_{perm} \cdot Overhead_{perm} + F_{temp} \cdot F_{unplan} \cdot Overhead_{unplanDis}) \cdot \frac{1}{MTBF}$

Le protocole de base utilisant seulement des points de reprise périodiques traite tous les types de fautes de la même manière. Ainsi, le taux de l'overhead est donné comme suit :

$Overhead_{rate}^{Basic} = \frac{T_{chkpt}}{2 \cdot MTBF}$

Pour la comparaison du surcoût temporel de notre protocole (Proposed Protocol) avec celui du protocole de base (Basic Protocol), on utilise les valeurs de paramètres suivantes :

- $T_{execution}$ = 2880 minutes (48 heures)
- $MTBF$ = 30 minutes , T_{chkpt} = 30 minutes
- Log_{rate} = 4 msgs / minute , H_{rate} = 0,2 handoff / minute
- $T_{unplanDis}$ = 9 secondes (0,15 minutes) , $T_{unplanDisMax}$ = 15 secondes (0,25 minutes)

Cet overhead est calculé dans différents cas (modèles) d'occurrence de fautes pouvant caractériser un environnement mobile. On utilise les quatre cas suivants :

- Cas 1 : $F_{unplan} = 25\%$, $F_{unplanH} = 75\%$, $F_{perm} = 1\%$
- Cas 2 : $F_{unplan} = 75\%$, $F_{unplanH} = 75\%$, $F_{perm} = 1\%$
- Cas 3 : $F_{unplan} = 25\%$, $F_{unplanH} = 25\%$, $F_{perm} = 1\%$
- Cas 4 : $F_{unplan} = 25\%$, $F_{unplanH} = 75\%$, $F_{perm} = 10\%$

La figure 4.2 représente l'overhead temporel affectant la durée d'exécution globale de l'application distribuée. Différentes combinaisons de plusieurs types de fautes sont utilisées afin de déterminer l'effet de chaque faute sur l'overhead temporel dans notre protocole. On constate que notre protocole proposé donne un surcoût très réduit par rapport à celui du protocole de base dans les différents cas. Cette amélioration est justifiée par l'utilisation d'un point de reprise local pour traiter une faute de déconnexion transitoire et d'un point de reprise de migration pour traiter une faute planifiée. De plus, ces deux types de fautes sont plus fréquents dans un environnement mobile. Dans notre protocole, le recouvrement des fautes permanentes impliquant relativement un surcoût plus élevé est effectué par la station de base.

En conséquence, la probabilité de recouvrement d'un site défaillant augmente.

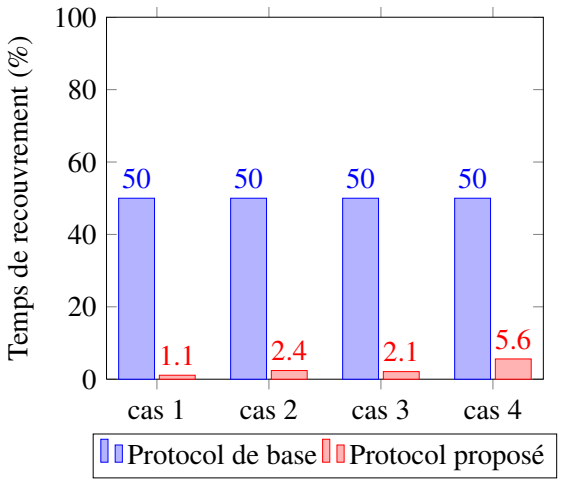

FIGURE 4.2 – L'overhead de temps d'exécution (le taux du temps de recouvrement).

4.5.1.2 Le surcoût de communication

L'overhead de communication à évaluer représente le nombre de messages transférés entre un site mobile et sa station de base lors du recouvrement. On distingue deux types de messages : les messages de données rejoués (renvoyés) lors du recouvrement et les messages contenant les points de reprise de recouvrement.

A. Le surcoût des messages de données rejoués

L'overhead de messages de données induit par une faute permanente est nul (égal à 0) parce que la reprise du site défaillant est prise en charge par la station de base.

L'overhead impliqué par une faute planifiée est aussi nul (égal à 0) parce que le site susceptible défaillant poursuit ses traitements sur sa station de base.

L'overhead induit par une faute de déconnexion imprévue (transitoire) dépend du moment de son occurrence. Lorsque la faute se produit avant ou après un handoff, l'overhead est égal au nombre de messages journalisés pendant la moitié de la durée moyenne entre deux handoff : $\frac{Log_{rate}}{2 \cdot H_{rate}}$. Quand la faute se produit lors d'un handoff, le site mobile effectue le recouvrement à partir de son point de reprise local pris juste avant le handoff. Dans ce cas, il n'existe aucun message journalisé à rejouer après ce point de reprise (overhead égal à 0).

Donc, l'overhead total de communication en termes de messages de données impliqué par notre protocole est calculé par la formule suivante :

$$hoverhead_{total}^{Proposed} = \frac{F_{temp} \cdot F_{unplan} \cdot F_{unplanNotH} \cdot Log_{rate} \cdot T_{execution}}{2 \cdot H_{rate} \cdot MTBF}$$

L'overhead de messages de données du protocole de base (Based Protocol) qui utilise seulement des points de reprise périodiques est donné par la formule suivante :

$$hoverhead_{total}^{Basic} = \frac{Log_{rate} \cdot T_{chkpt} \cdot T_{execution}}{2 \cdot MTBF}$$

Notre protocole est comparé avec le protocole de base en utilisant les mêmes valeurs de paramètres et les quatre cas d'occurrence de fautes utilisés pour évaluer le surcoût temporel.

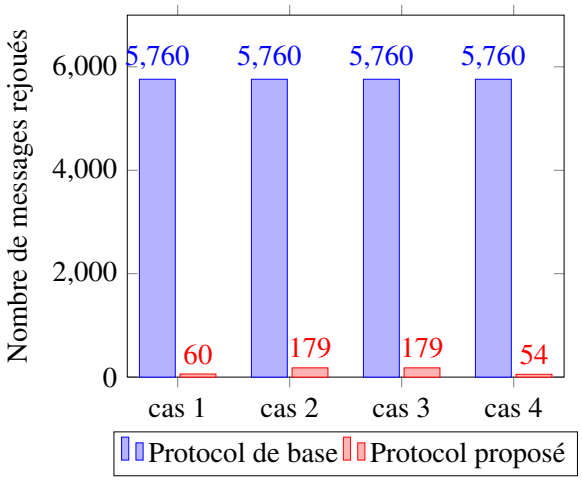

FIGURE 4.3 – L'overhead de messages de données rejoués.

La figure 4.3 représente l'overhead de communication en termes du nombre de messages rejoués lors du recouvrement. On constate que notre protocole donne un surcoût très réduit par rapport au protocole de base. Cette amélioration est justifiée par l'utilisation d'un point de reprise local et d'un point de reprise de migration pour traiter respectivement une faute de déconnexion imprévue et une faute planifiée.

B. Le surcoût des points de reprise

L'overhead en termes de points de reprise induit par une faute permanente est égal à 1.

L'overhead induit par une faute planifiée est égal à 2. Un point de reprise est utilisé pour la migration du processus susceptible défaillant vers sa station de base et un autre est utilisé pour la récupération de ce processus par son site mobile ou un site de reprise.

L'overhead impliqué par une faute de déconnexion imprévue est nul (égal à 0) parce que le site défaillant utilise un point de reprise local.

Donc, l'overhead total en termes de points de reprise induit par notre protocole est calculé comme suit :

$$hoverhead_{total}^{Proposed} = (F_{perm} + 2 \cdot F_{temp} \cdot F_{plan}) \cdot \frac{T_{execution}}{MTBF}$$

L'overhead total en termes de points de reprise du protocole de base qui utilise seulement des points de reprise périodiques est donné par la formule suivante :

$hoverhead_{total}^{Basic} = \frac{T_{execution}}{MTBF}$

Notre protocole est comparé avec le protocole de base en utilisant les mêmes valeurs de paramètres utilisées pour évaluer le surcoût temporel. Quatre cas (modèles) d'occurrence de fautes planifiées et permanentes sont utilisés :

- Cas 1 : $F_{plan} = 75\%$, $F_{perm} = 1\%$
- Cas 2 : $F_{plan} = 50\%$, $F_{perm} = 1\%$
- Cas 3 : $F_{plan} = 25\%$, $F_{perm} = 1\%$
- Cas 4 : $F_{plan} = 75\%$, $F_{perm} = 10\%$

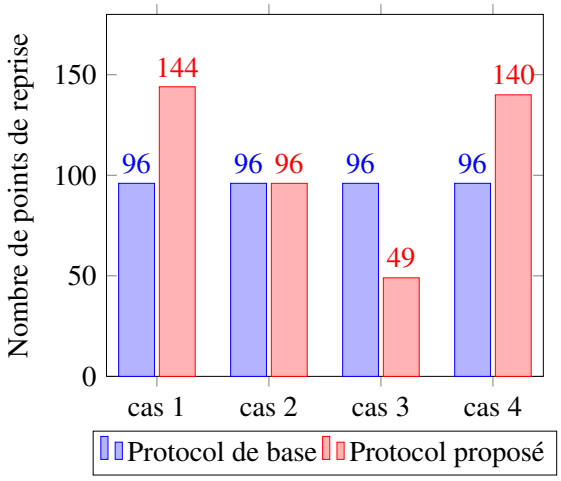

FIGURE 4.4 – L'overhead de points de reprise transférés.

La figure 4.4 représente l'overhead de communication en termes du nombre de points de reprise transférés entre le site mobile et sa station de base durant le recouvrement. On constate que le nombre de points de reprise change en fonction du taux d'occurrence d'une faute planifiée. Lorsque le nombre de fautes planifiées est plus élevé, notre protocole implique un overhead supérieur à celui du protocole de base. En revanche, lorsque ce nombre est plus réduit, notre protocole induit un overhead inférieur à celui du protocole de base. On remarque aussi que les deux protocoles induisent presque le même overhead pour un certain taux de fautes planifiées. Ceci est justifié par l'équilibrage entre l'overhead évité en utilisant des points de reprise locaux et celui impliqué en utilisant des points de reprise de migration.

4.5.2 Le surcoût pendant l'exécution sans faute

Pendant l'exécution sans faute, un site mobile effectue deux types de points de reprise : périodique et local. Le point de reprise de migration est utilisé pour le recouvrement d'une faute planifiée. Ainsi, l'overhead induit par celui-ci est introduit dans le calcul du surcoût de recouvrement (la migration est utilisée comme un mécanisme de recouvrement).

Lors de la migration d'un processus son exécution se poursuit sur la station de base et temporisateur (timer) utilisé pour la sauvegarde de points de reprise périodiques est réinitialisé après la récupération du processus par son site mobile ou un site de reprise. Donc, les points de reprise de migration diminuent le nombre de points de reprise périodiques dans notre protocole. Par conséquent, le nombre de ponts de reprise périodiques dans le protocole proposé dépend du nombre de fautes planifiés et leurs moments d'occurrence.

Le protocole proposé implique un overhead additionnel pour prendre et sauvegarder un point de reprise local. Ces points de reprise locaux affectent les performances du site mobile et n'ont aucun effet sur la communication entre le site et sa station de base. Leur nombre est égal au nombre de handoff effectués pendant la durée de l'exécution globale de l'application (ou du site mobile) : $H_{rate} \cdot T_{execution}$.

Lors de la journalisation de messages, notre protocole n'implique aucun overhead de communication et c'est le même cas pour le protocole de base. De plus, les deux protocoles impliquent le même surcoût de stockage des journaux de messages au niveau de la station de base.

4.6 Conclusion

Dans ce chapitre, nous avons proposé un protocole de tolérance aux fautes d'une application distribuée s'exécutant sur un environnement mobile. Cet environnement s'appuie sur un réseau mobile avec infrastructure à couverture discontinue. Le protocole proposé est basé sur la sauvegarde asynchrone (indépendante) de points de reprise et la journalisation pessimiste de messages. Il traite l'aspect de la continuité de délivrance des services (une exécution non-stop) d'une application distribuée en présence de différentes fautes matérielles transitoires et permanentes. Nous avons mis l'accent en particulier sur les fautes de déconnexion du réseau qui sont plus fréquentes dans ce contexte. Trois types de points de reprise sont utilisés : un point de reprise périodique pour le recouvrement des fautes permanentes, un point de reprise local permet la reprise d'une faute de déconnexion non planifiée et un point de reprise de migration pour le recouvrement d'une faute planifiée. Notre protocole permet de minimiser le surcoût de recouvrement ainsi que l'overhead temporel affectant la délivrance des services d'une application distribuée mobile.

Conclusion générale

Les applications distribuées ciblant des terminaux mobiles s'appuient le plus souvent sur des réseaux mobiles avec infrastructure. Cet environnement mobile présente des nouvelles contraintes telles que : les ressources de calcul et de stockage moins importantes, la bande passante faible et variable, la source d'énergie limitée, les fréquentes déconnexions qui peuvent être volontaires ou involontaires, la vulnérabilité aux dommages physiques, etc.

La tolérance aux fautes dans les systèmes distribués est un besoin imposé par la répartition et l'approche par reprise semble la plus adaptée pour sa réalisation. Compte tenu des particularités de l'environnement mobile, de nouveaux protocoles adaptés à ce contexte ont été proposés dans la littérature. Cependant, ces protocoles supposent que les fautes matérielles causant la défaillance d'un processus sont transitoires et ne prennent pas en considération leur durée de persistance. De plus, l'environnement mobile considéré se base sur une infrastructure à couverture continue. Ainsi, l'objectif de notre travail consiste à assurer à une application distribuée mobile une exécution non-stop en traitant des différents types de fautes matérielles transitoires et permanentes, et de minimiser l'overhead temporel affectant la délivrance de ses services (résultats).

Dans ce mémoire, nous avons proposé un protocole de tolérance aux fautes par reprise d'une application distribuée s'exécutant sur un réseau mobile avec infrastructure à couverture discontinue. Notre stratégie proposée se base sur la sauvegarde asynchrone (indépendante) de points de reprise et la journalisation pessimiste de messages afin d'assurer aux sites défaillants un recouvrement indépendant. Chaque station de base fournit un support stable pour sauvegarder les points de reprise et les journaux de messages des sites mobiles se trouvant dans sa zone de couverture. Elle peut aussi jouer le rôle d'un site de reprise qui poursuit les activités des sites défaillants. Nous avons utilisé trois types de points de reprise, chacun destiné pour le recouvrement d'une catégorie de fautes. Le point de reprise périodique sauvegardé dans la station de base permet le recouvrement d'une faute permanente. Le point de reprise local sauvegardé dans le site mobile sert à la reprise d'une faute de déconnexion non planifiée (transitoire). Le point de reprise de migration est utilisé pour le recouvrement d'une faute planifiée. Un mécanisme d'élection est utilisé afin de choisir un site de reprise remplaçant le site défaillant dans le cas d'une faute permanente. Le choix s'effectue selon deux critères : le niveau d'énergie et la stabilité du site de support.

Notre protocole fournit les avantages que présente l'utilisation de la reprise par journalisa-

tion pessimiste de messages, à savoir un seul point de reprise est nécessaire par processus (pas d'effet domino), la défaillance d'un site n'affecte pas les processus des autres sites (recouvrement indépendant), etc. L'analyse des performances de ce protocole nous a montré que celui-ci permet de minimiser le surcoût (overhead) de recouvrement en termes de temps d'exécution et de communication. De plus, la journalisation de messages n'implique aucun overhead de communication. Cependant, ce protocole génère un surcoût induit par la sauvegarde de points de reprise locaux et le transfert de points de reprise de migration (migration de processus).

Les perspectives d'extension et d'amélioration de notre travail sont nombreuses, parmi lesquelles on peut citer :

- L'implémentation du protocole proposé dans une plateforme de simulation afin de permettre d'évaluer au mieux ses performances à travers différents modèles de mobilité, d'applications réparties, etc.
- Extension de notre protocole pour prendre en considération les fautes au niveau de la station de base.
- Adaptation du protocole aux réseaux mobiles ad hoc (MANET).
- Étude de l'intégration de notre protocole dans les grilles de calcul mobiles (Mobile Grid Computing) ainsi que dans les nuages de calcul mobiles (Mobile Cloud Computing) afin de montrer sa faisabilité dans le domaine du calcul intensif.

Bibliographie

[Acharya 1994] Arup Acharya et BR Badrinath. *Checkpointing distributed applications on mobile computers*. In Proceedings of the Third International Conference on Parallel and Distributed Information Systems, 1994, pages 73–80. IEEE, 1994. 51

[Agha 2001] Khaldoun Al Agha, Guy Pujolle et Guillaume Vivier. Réseaux de mobiles & réseaux sans fil. Eyrolles, 2001. 6, 11, 20

[Ahn 2004] JinHo Ahn, Sung-Gi Min et Chong-Sun Hwang. *A causal message logging protocol for mobile nodes in mobile computing systems*. Future Generation Computer Systems, vol. 20, no. 4, pages 663–686, 2004. 52

[Akyildiz 2004] I. F. Akyildiz, Jiang Xie et S. Mohanty. *A Survey of Mobility Management in Next-generation all-IP-based Wireless Systems*. Wireless Commun., vol. 11, no. 4, pages 16–28, August 2004. 14

[Aliouat 2007] M Aliouat et Z Aliouat. *Recovery in Distributed Systems from Transient and Permanent Faults*. Journal of Computer Science, vol. 3, no. 8, page 617, 2007. 2, 52

[Aliouat 2009] Makhlouf Aliouat et Zibouda Aliouat. *Applications Distribuées Tolérantes aux Pannes de Batteries des Nœuds dans les Systèmes Autonomes*. 2009. 2, 52

[Alvisi 1996] Lorenzo Alvisi. *Understanding the message logging paradigm for masking process crashes*. Rapport technique, Cornell University, 1996. 46

[Alvisi 1998] Lorenzo Alvisi et Keith Marzullo. *Message logging : Pessimistic, optimistic, causal, and optimal*. IEEE Transactions on Software Engineering, vol. 24, no. 2, pages 149–159, 1998. 45

[Arlat 2006] Jean Arlat, Yves Crouzet, Yves Deswarte, Jean-Charles Fabre, Jean-Claude Laprie et David Powell. In Encyclopédie de l'Informatique et des systèmes d'information, chapitre Tolérance aux fautes, page 241–270. Vuibert, 2006. 27, 30, 31, 33

[Avižienis 2004] Algirdas Avižienis, Jean-Claude Laprie, Brian Randell et Carl Landwehr. *Basic Concepts and Taxonomy of Dependable and Secure Computing*. IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pages 11–33, 2004. 27, 28, 29, 33

[Badis 2006] Hakim Badis. *Etude et conception d'algorithmes pour les réseaux mobiles et ad hoc*. Thèse de doctorat, Université Paris-Sud, Orsay, France, 2006. 15

[Badrinath 1994] BR Badrinath, Arup Acharya et Tomasz Imielinski. *Structuring distributed algorithms for mobile hosts*. In Proceedings of the 14th International Conference on Distributed Computing Systems, 1994, pages 21–28. IEEE, 1994. 51

[Baldoni 1997] Roberto Baldoni, J-M Helary, Achour Mostefaoui et Michel Raynal. *A communication-induced checkpointing protocol that ensures rollback-dependency trackability*. In Digest of Papers of Twenty-Seventh Annual International Symposium on Fault-Tolerant Computing, FTCS-27, 1997, pages 68–77. IEEE, 1997. 43

[Bchini 2010] Tarek Bchini. *Gestion de la Mobilité, de la Qualité de Service et Interconnexion de Réseaux Mobiles de Nouvelle Génération*. Thèse de doctorat, Université de Toulouse, Toulouse, France, Juin 2010. 19

[Besseron 2010] Xavier Besseron. *Tolérance aux fautes et reconfiguration dynamique pour les applications distribuées à grande échelle*. Thèse de doctorat, Université de Grenoble, Grenoble, France, Avril 2010. 29, 36, 39, 47

[Bhargava 1988] B. Bhargava et S. R. Lian. *Independent checkpointing and concurrent rollback for recovery in distributed systems-an optimistic approach*. In Proceedings of the Seventh Symposium on Reliable Distributed Systems, pages 3–12, Oct 1988. 42

[Briatico 1984] Daniele Briatico, Augusto Ciuffoletti et Luca Simoncini. *A Distributed Domino-Effect free recovery Algorithm.* In Symposium on Reliability in Distributed Software and Database Systems, volume 84, pages 207–215, 1984. 44

[Cáceres 1996] Ramón Cáceres et Venkata N. Padmanabhan. *Fast and Scalable Handoffs for Wireless Internetworks*. In Proceedings of the 2Nd Annual International Conference on Mobile Computing and Networking, MobiCom '96, pages 56–66, New York, NY, USA, 1996. ACM. 14

[Campbell 2000] A. T. Campbell, J. Gomez, S. Kim, Valko A. G. et Chieh-Yih Wan. *Design, Implementation, and Evaluation of Cellular IP*. IEEE Personal Communications, vol. 7, no. 4, pages 42–49, August 2000. 15

[Cao 1998] Guohong Cao et Mukesh Singhal. *On the impossibility of min-process non-blocking checkpointing and an efficient checkpointing algorithm for mobile computing systems*. In Proceedings of International Conference on Parallel Processing, 1998, pages 37–44. IEEE, 1998. 51

[Cao 2001] Guohong Cao et Mukesh Singhal. *Mutable checkpoints : a new checkpointing approach for mobile computing systems*. IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 2, pages 157–172, 2001. 51

[Chandy 1985] K Mani Chandy et Leslie Lamport. *Distributed snapshots : determining global states of distributed systems*. ACM Transactions on Computer Systems (TOCS), vol. 3, no. 1, pages 63–75, 1985. 40, 43

[Conchon 2006] Emmanuel Conchon. *Définition et Mise en Oeuvre d'une Solution d'Émulation de Réseaux Sans fil*. Thèse de doctorat, Institut National Polytechnique de Toulouse, Toulouse, France, Octobre 2006. 8

[Corson 1999] S. Corson et J. Macker. *Mobile Ad hoc Networking (MANET) : Routing Protocol Performance Issues and Evaluation Considerations*. RFC 2501 (Informational), January 1999. 8

[Cristian 1991] Flaviu Cristian et Farnam Jahanian. *A timestamp-based checkpointing protocol for long-lived distributed computations*. In Proceedings of Tenth Symposium on Reliable Distributed Systems, 1991, pages 12–20. IEEE, 1991. 43

[Das 2000] S. Das, A. Misra et P. Agrawal. *TeleMIP : Telecommunications-Enhanced Mobile IP Architecture for Fast Intradomain Mobility*. IEEE Personal Communications, vol. 7, no. 4, pages 50–58, August 2000. 15

[De Vriendt 2002] J. De Vriendt, P. Laine, C. Lerouge et Xiaofeng Xu. *Mobile network evolution : a revolution on the move*. Communications Magazine, IEEE, vol. 40, no. 4, pages 104–111, Apr 2002. 24

[Delbé 2007] Christian Delbé. *Tolérance aux pannes pour objets actifs asynchrones : protocole, modèle et expérimentation*. Thèse de doctorat, Université de Nice - Sophia Antipolis, France, Janvier 2007. 34, 35

[Demoulin 2004] C. Demoulin et M. Van Droogenbroeck. *Principes de base du fonctionnement du réseau GSM*. Revue de l'AIM, no. 4, pages 3–18, 2004. 23

[Deswarte 1990] Y. Deswarte. Tolérance aux fautes, sécurité et protection dans les systèmes répartis. Rapport LAAS. Laboratoire d'automatique et d'analyse des systèmes du CNRS, 1990. 31, 33

[Dhoutaut 2003] Dominique Dhoutaut. *Etude du standard IEE 802.11 dans le cadre des réseaux ad hoc : de la simulation à l'expérimentation*. Thèse de doctorat, Institut National des Sciences Appliquées de Lyon, Décembre 2003. 17

[Elnozahy 1992] Elmootazbellah Nabil Elnozahy, David B Johnson et Willy Zwaenepoel. *The performance of consistent checkpointing*. In Proceedings of 11th Symposium on Reliable Distributed Systems, 1992, pages 39–47. IEEE, 1992. 43

[Elnozahy 1994a] Elmootazbellah N Elnozahy et Willy Zwaenepoel. *On the use and implementation of message logging*. In Digest of Papers of Twenty-Fourth International Symposium on Fault-Tolerant Computing, FTCS-24, 1994, pages 298–307. IEEE, 1994. 45

[Elnozahy 1994b] Elmootazbellah Nabil Elnozahy et Willy Zwaenepoel. *Manetho : fault tolerance in distributed systems using rollback-recovery and process replication*. Rice University, Houston, TX, 1994. 46, 47

[Elnozahy 2002] E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang et David B. Johnson. *A Survey of Rollback-recovery Protocols in Message-passing Systems*. ACM Comput. Surv., vol. 34, no. 3, pages 375–408, September 2002. 40, 41, 42, 45, 46, 47

[Forman 1994] George H. Forman et John Zahorjan. *The Challenges of Mobile Computing*. Computer, vol. 27, no. 4, pages 38–47, April 1994. 10

[Geier 2004] Jim Geier. Wireless networks first-step. Cisco Press, 2004. 7

[George 2006] Sapna E George, Ing-Ray Chen et Ying Jin. *Movement-based checkpointing and logging for recovery in mobile computing systems*. In Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access, pages 51–58. ACM, 2006. 52

[Géron 2009] A. Géron. Wifi professionnel- 3e édition - : La norme 802.11, le déploiement, la sécurité. Réseaux et télécoms. Dunod, 2009. 17

[Girodon 2002] Stéphane Girodon. *Réseaux GSM, GPRS, UMTS. Architecture évolutive pour une stratégie services*. Rapport de stage, DESS MTI, IAE Aix en Provence, Juin 2002. 22, 23

[Gray 2006] Doug Gray. *Mobile WiMAX – Part I : A Technical Overview and Performance Evaluation*. WiMAX Forum, 2006. 19, 20

[Gustafsson 2004] E. Gustafsson, A. Jonsson et C. E. Perkins. *Mobile IP Regional Registration*. Internet draft, IETF : The Internet Engineering Task Force, draft-ietf-mobileip-reg-tunnel-09.text, June 2004. work in progress. 15

[Hélary 2000] J-M Hélary, Achour Mostefaoui, Robert HB Netzer et Michel Raynal. *Communication-based prevention of useless checkpoints in distributed computations*. Distributed Computing, vol. 13, no. 1, pages 29–43, 2000. 44

[Imran 2007] Nomica Imran, Imran Rao, Young-Koo Lee et Sungyoung Lee. *A proxy-based uncoordinated checkpointing scheme with pessimistic message logging for mobile grid systems*. In Proceedings of the 16th international symposium on High performance distributed computing, pages 237–238. ACM, 2007. 52

[Jafar 2006] Samir Jafar. *Programmation des systèmes parallèles distribuées : tolérance aux pannes, résilience et adaptabilité*. Thèse de doctorat, Institut National Polytechnique de Grenoble (INPG), Grenoble, France, Juin 2006. 36, 39, 40

[Jallouli 2009] Mehdi Jallouli. *Méthodologie de conception d'architectures de processeur sûres de fonctionnement pour les applications mécatroniques*. Thèse de doctorat, Université Paul Verlaine – Metz, Juin 2009. 30

[Johnson 1987] David B. Johnson et Willy Zwaenepoel. *Sender-based message logging*. In The 7th annual international symposium on fault-tolerant computing. IEEE Computer Society, 1987. 45

[Johnson 1990] David B Johnson et Willy Zwaenepoel. *Recovery in distributed systems using optimistic message logging and checkpointing*. Journal of algorithms, vol. 11, no. 3, pages 462–491, 1990. 45, 46

[Kalla 2004] Hamoudi Kalla. *Génération automatique de distributions/ordonnancements temps réel, fiables et tolérants aux fautes*. Thèse de doctorat, Institut National Polytechnique de Grenoble (INPG), Greboble, France, Décembre 2004. 29, 30

[Kassar 2008] Meriem Kassar, Brigitte Kervella et Guy Pujolle. *An overview of vertical handover decision strategies in heterogeneous wireless networks*. Computer Communications, vol. 31, no. 10, pages 2607–2620, June 2008. 11, 12, 13

[Khelladi 2004] Lyes Khelladi et Nadjib Badache. *Les réseaux de capteurs : état de l'art*. Rapport de recherche LSI-TR0304, USTHB, Faculté électronique et informatique, Bab Ezzouar - Algérie, Février 2004. 8

[Khunteta 2011] A Khunteta, P Sharma et R Garg. *New & efficient low overheads algorithm for mobile distributed systems*. In Proceedings of the International Conference & Workshop on Emerging Trends in Technology, pages 447–450. ACM, 2011. 51

[Komarova 2008] Maryna Komarova. *Authentification rapide et contrôle d'accès basé sur la confiance dans les réseaux sans fil hétérogènes*. Thèse de doctorat, Telecom-ParisTech, Paris, France, Mai 2008. 11, 12

[Koo 1987] Richard Koo et Sam Toueg. *Checkpointing and rollback-recovery for distributed systems*. IEEE Transactions on Software Engineering, no. 1, pages 23–31, 1987. 43

[Kumar 2008] Parveen Kumar. *A low-cost hybrid coordinated checkpointing protocol for mobile distributed systems*. Mobile Information Systems, vol. 4, no. 1, pages 13–32, 2008. 51

[Laarouchi 2009] Youssef Laarouchi. *Sécurités (immunité et innocuité) des architectures ouvertes à niveaux de criticité multiples : application en avionique*. Thèse de doctorat, Institut National des Sciences Appliquées - Toulouse (INSA), Toulouse, France, Novembre 2009. 27, 28

[Lai 1987] Ten H Lai et Tao H Yang. *On distributed snapshots*. Information Processing Letters, vol. 25, no. 3, pages 153–158, 1987. 43

[Lamport 1978] Leslie Lamport. *Time, clocks, and the ordering of events in a distributed system*. Communications of the ACM, vol. 21, no. 7, pages 558–565, 1978. 39

[Lampropoulos 2008] G. Lampropoulos, A. K. Salkintzis et N. Passas. *Media-independent handover for seamless service provision in heterogeneous networks*. Communications Magazine, IEEE, vol. 46, no. 1, pages 64–71, 2008. 13

[Laprie 1996] J.C. Laprie, J. Arlat et Laboratoire d'ingénierie de la sûreté de fonctionnement. Guide de la sûreté de fonctionnement. Cépaduès-Editions, 1996. 27

[Lemainque 2009] Fabrice. Lemainque. Tout sur les réseaux sans fil. Dunod, 2009. 6, 7

[Lemlouma 2000] Tayeb Lemlouma et Nadjib Badache. Le routage dans les réseaux mobiles ad hoc. Mini projet, Université des Sciences et de la Technologie Houari Boumèdiene, Institut d'Informatique, Septembre 2000. 8, 9, 10

[Lussier 2007] Benjamin Lussier. *Tolérance aux fautes dans les systèmes autonomes*. Thèse de doctorat, Institut National Polytechnique de Grenoble (INPG), Grenoble, France, Avril 2007. 30, 31, 33

[Manivannan 1996a] D Manivannan et M Singhal. *Failure recovery based on quasi-synchronous checkpointing in mobile computing systems*. The Ohio State University, Department of Computer and Information Science, Technical Report No. OSU-CISRC-7/96-TR36, 1996. 51

[Manivannan 1996b] D Manivannan et Mukesh Singhal. *A low-overhead recovery technique using quasi-synchronous checkpointing*. In Proceedings of the 16th International Conference on Distributed Computing Systems, 1996, pages 100–107. IEEE, 1996. 44

[Montavont 2001] Nicolas Montavont. La Mobilité dans les Réseaux IP. Rapport de D.E.A. Informatique, Université Louis Pasteur de Strasbourg, 2001. 11

[Netzer 1995] Robert HB Netzer et Jian Xu. *Necessary and sufficient conditions for consistent global snapshots*. IEEE Transactions on Parallel and distributed Systems, vol. 6, no. 2, pages 165–169, 1995. 43

[Park 2000] Taesoon Park et Heon Young Yeom. *An asynchronous recovery scheme based on optimistic message logging for mobile computing systems*. In Proceedings of 20th International Conference on Distributed Computing Systems, 2000, pages 436–443. IEEE, 2000. 52

[Park 2003] Taesoon Park, Namyoon Woo et Heon Y Yeom. *An efficient recovery scheme for fault-tolerant mobile computing systems*. Future Generation Computer Systems, vol. 19, no. 1, pages 37–53, 2003. 52

[Perkins 2001] C. E. Perkins et D. B. Johnson. *Route Optimization in Mobile IP*. Internet draft, IETF : The Internet Engineering Task Force, draft-ietf-mobileip-optim-11.text, September 2001. work in progress. 14

[Perkins 2002] C. E. Perkins. *Mobile IP*. Communications Magazine, IEEE, vol. 40, no. 5, pages 66–82, 2002. 13

[Pierre 2003] Samuel Pierre. Réseaux et systèmes informatiques mobiles : Fondements, architectures et applications. Presses internationales Polytechnique, 2003. 20

[Plank 1993] James Steven Plank. *Efficient checkpointing on MIMD architectures*. PhD thesis, Princeton University Princeton, 1993. 42

[Pollini 1996] G. P. Pollini. *Trends in handover design*. Communications Magazine, IEEE, vol. 34, no. 3, pages 82–90, March 1996. 12

[Prakash 1996] Ravi Prakash et Mukesh Singhal. *Low-cost checkpointing and failure recovery in mobile computing systems*. IEEE Transactions on Parallel and Distributed Systems, vol. 7, no. 10, pages 1035–1048, 1996. 51

[Pujolle 2007] Guy Pujolle, Olivier Collaborateur. Salvatori et Jacques Collaborateur. Nozick. Les réseaux : édition 2008. Eyrolles, Paris, 2007. 17, 18, 20, 21, 24

[Ramjee 1999] R. Ramjee, T. La Port, S. Thuel, K. Varadhan et S. Y. Wang. *HAWAII : A Domain-based Approach for Supporting Mobility in Wide-area Wireless Networks*. In Proceedings of the IEEE International Conference on Network Protocols, 1999. 15

[Randell 1975] B. Randell. *System Structure for Software Fault Tolerance*. In Proceedings of the International Conference on Reliable Software, pages 437–449, New York, NY, USA, 1975. ACM. 41, 42

[Sadok 1999] Djamel H Sadok, Judith Kelner et Carlos de Morais Cordeiro. *Disconnection protocol support in mobile access*. Journal of the Brazilian Computer Society, vol. 5, no. 3, pages 00–00, 1999. 55

[Salhani 2008] Mohamed Salhani. *Modélisation et Simulation des Réseaux Mobiles de 4ème Génération*. Thèse de doctorat, Université de Toulouse, Toulouse, France, Octobre 2008. 18

[Satyanarayanan 1996] M. Satyanarayanan. *Fundamental challenges in mobile computing*. In Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing, PODC '96, pages 1–7, New York, NY, USA, 1996. ACM. 10

[Sayah 2009] Jinane Sayah. *Contribution à la modélisation, à la simulation et à l'évaluation d'applications nomades à intelligence répartie – Application à l'assistance aux voyageurs aveugles dans les transports publics et les pôles d'échanges*. Thèse de doctorat, Université Paris-Est, Paris, France, Décembre 2009. 16, 17

[Sistla 1989] A Prasad Sistla et Jennifer L Welch. *Efficient distributed recovery using message logging*. In Proceedings of the eighth annual ACM Symposium on Principles of distributed computing, pages 223–238. ACM, 1989. 44, 45

[Strom 1985] Rob Strom et Shaula Yemini. *Optimistic recovery in distributed systems*. ACM Transactions on Computer Systems (TOCS), vol. 3, no. 3, pages 204–226, 1985. 40, 44, 45

[Taïani 2004] François Taïani. *La Réflexivité dans les architectures multi-niveaux : application aux systèmes tolérant les fautes*. Thèse de doctorat, Université Paul Sabatier de Toulouse, Toulouse, France, Janvier 2004. 28, 30

[Tamir 1984] Yuval Tamir et Carlo H Sequin. *Error recovery in multicomputers using global checkpoints*. In 1984 International Conference on Parallel Processing. Citeseer, 1984. 42

[Taniuchi 2009] Kenichi Taniuchi, Yoshihiro Ohba, Victor Fajardo, Subir Das, Miriam Tauil, Yuu-Heng Cheng, Ashutosh Dutta, Donald Baker, Maya Yajnik et David Famolari. *IEEE 802.21 : media independent handover : features, applicability, and realization*. Communications Magazine, IEEE, vol. 47, no. 1, pages 112–120, January 2009. 13

[Tong 1992] Zhijun Tong, Richard Y. Kain et WT Tsai. *Rollback recovery in distributed systems using loosely synchronized clocks*. IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 2, pages 246–251, 1992. 43

[Van Den Bossche 2007] Adrien Van Den Bossche. *Proposition d'une nouvelle méthode d'accès déterministe pour un réseau personnel sans fil à fortes contraintes temporelles*. Thèse de doctorat, Université de Toulouse II, Juillet 2007. 17

[Wang 1995] Yi-Min Wang, Pi-Yu Chung, In-Jen Lin et W. Kent Fuchs. *Checkpoint Space Reclamation for Uncoordinated Checkpointing in Message-Passing Systems.* IEEE Trans. Parallel Distrib. Syst., vol. 6, no. 5, pages 546–554, May 1995. 42

[Wang 1997] Yi-Min Wang. *Consistent global checkpoints that contain a given set of local checkpoints*. IEEE Transactions on Computers, vol. 46, no. 4, pages 456–468, 1997. 42

[Zdarsky 2004] Frank A. Zdarsky et Jens B. Schmitt. *Handover in Mobile Communication Networks : Who is in Control Anyway ?* In Proceedings of the 30th EUROMICRO Conference, EUROMICRO '04, pages 205–212, Washington, DC, USA, 2004. IEEE Computer Society. 13