Mise au point d'algorithmes répartis dans un environnement fortement variable, et expérimentation dans le contexte des pico-réseaux

Corine Marchand

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : "Informatique : Systèmes et Communications"

préparée au laboratoire INFORMATIQUE ET DISTRIBUTION
dans le cadre de ***l'École Doctorale "Mathématiques, Sciences et Technologies de l'Information, Informatique"***

présentée et soutenue publiquement

par

Corine MARCHAND

le 20 décembre 2004

Mise au point d'algorithmes répartis dans un environnement fortement variable, et expérimentation dans le contexte des pico-réseaux

---

Directeur de thèse : Mme Brigitte Plateau

Co-Directeur de thèse : Mr Jean-Marc Vincent

JURY

MR JACQUES MOSSIÈRE,	Président
MR PIERRE SENS,	Rapporteur
MR MICHEL HURFIN,	Rapporteur
MR SERGE MARTIN,	Examinateur
MME BRIGITTE PLATEAU,	Examinateur (directeur de thèse)
MR JEAN-MARC VINCENT,	Examinateur (co-encadrant)

Remerciements

Je tiens en premier lieu à remercier le département MAPSAMS de France Télécom R&D pour avoir soutenu et participé à cette action spécifique sans laquelle cette thèse n'aurait pu avoir lieu.

Je voudrais également remercier les membres du jury pour avoir accepter de juger ce travail de thèse. En particulier, je tiens à remercier Jacques MOSSIÈRE, professeur à l'Institut National Polytechnique de Grenoble, pour m'avoir fait l'honneur de présider ce jury de thèse.
Merci à PierreSENS, professeur à Paris VI, et Michel HURFIN, chargé de recherche INRIA à Rennes, d'avoir accepté d'évaluer ma thèse. J'ai particulièrement apprécié vos rapports détaillés et approfondis témoignant de l'intérêt porté à mon travail.
Merci à Serge MARTIN, chercheur à France Télécom R&D, d'avoir accepter de participer à ce jury de thèse, mais également pour son écoute, son ouverture et les orientations qu'il a su me donner durant ces trois années de collaboration scientifique.

Je remercie Brigitte PLATEAU, professeur à l'Ensimag-Inpg et directrice du laboratoire ID-Imag, pour m'avoir accueillie au sein d'un laboratoire où la bonne ambiance régnait en maître. Bien que, comme tu l'as souligné lors de ma soutenance, notre relation avait commencé sur les "chapeaux de roues" par une bataille de glace pillée, j'ai pu apprécier au cours de ces années ta gentillesse et tes conseils avisés (et également une belle panoplie d'autographes indispensable à l'administration !).

Un grand merci à Jean-Marc VINCENT, maître de conférences UJF, co-encadrant officiel de ma thèse mais véritable investigateur de ce travail. Je tiens sincèrement à te remercier pour m'avoir fait confiance, et pour avoir rendu cette aventure possible. Travailler à tes côtés fut une expérience fortement enrichissante autant scientifiquement que humainement. Je me suis cependant souvent demandée quelle probabilité serait à associer à ton manque de disponibilité notoire ? Mais tout bien réfléchi, mieux vaudrait modéliser la grande patience dont tu as su faire preuve (surtout lors de mes moments de doutes et de remise en question !). Je te remercie pour avoir fait preuve jusqu'a la fin d'une grande qualité humaine.

Je ne saurai comment exprimer mes remerciements envers les membres du projet SIDRAH. Les nombreuses heures de discussion et les séances de remue-méninges m'ont appris beaucoup et surtout permis d'avoir une vision différente mais complémentaire du monde de la recherche.

Merci à Rémi, co-bureau fidèle, pour avoir partagé les bons moments comme les moins bons pendant la majeure partie de cette thèse. J'arrivais tôt (très tôt même pour la plupart des ID-istes !), tu partais tard (je pense même très tard !), notre bureau va certainement devoir s'adapter à des horaires d'ouverture plus "normaux". Comment te remercier pour avoir rendu ces 3 années de cohabitation inoubliables ?
Merci à Georges pour avoir accepté de te remplacer alors que je commençais la phase de rédaction (Quel courage ! ! !). Merci pour ta bonne humeur et tes nombreuses tentatives de re-motivations dans les moments les plus difficiles.

Merci à tous ceux qui ont participé à la réalisation de cette thèse de prés ou de loin, chacun à votre manière.

Un grand merci à ma famille pour m'avoir toujours soutenue et encouragée et surtout pour avoir toujours répondu présent quand j'en avais le plus besoin. J'ai pu puisé à travers vous l'équilibre et l'énergie nécessaires à l'aboutissement d'un tel projet.

Merci à tous mes amis pour leur indulgence face à mon manque de disponibilité et pour avoir toujours trouvé de nombreux remèdes à mes coups de blues ou mes coups de "ras-le-bol" général (soirées "on-refait-le-monde", soirées bières-pizzas, soirées barbecues, etc.....).

Comment ne pas mentionner également les nombreuses soirées passées autour d'une bière (plutôt plusieurs d'ailleurs ! ! !) ? En fin de journée, le Papaya était presque devenu la succursale du laboratoire ! ! ! Merci à tous, ID-istes ou non, pour avoir partagé ces fameuses soirées (fabuleuses je devrais dire ! ! !) ponctuées de rires, de discussions plus ou moins sérieuses sur des sujets divers et variés. Merci pour tous ces moments fortement enrichissants (tant sur le plan humain que calorifique, d'ailleurs !).

Enfin, un merci tout particulier à tous les ID-istes, amis du pinguoin pour votre aide (nombreuses installations sur machines variées, et service après vente !), développeurs fous et bidouilleurs en tout genre pour vos nombreux conseils indispensables, compagnons de pauses diverses (pauses café-thé, pauses "clopes", pause quoinche, pause Montbonnot voir St Ismier, ...), etc...
Un grand merci à vous tous pour ces trois années formidables.

Table des matières

Table des figures

Liste des tableaux

Introduction 1

L'évolution importante et la diversification du matériel "mobile" (ordinateurs portables, PDAs, téléphones cellulaires, etc ...), la généralisation de la mobilité des utilisateurs au sens nomade du terme, et l'attrait du grand public pour les nouvelles technologies dites sans fil, confortent l'intérêt général pour le concept de la mobilité informatique. Domaine aussi vaste que les travaux le concernant sont diversifiés. En effet, beaucoup de travaux de recherche dans un grand nombre de domaines différents abordent les nouvelles problématiques introduites par les environnements mobiles sous diverses approches à la fois compétitives et complémentaires.

Face à cet engouement pour la mobilité, les technologies sans fil (Bluetooth, WIFI, etc ...) et l'ensemble des domaines connexes tiennent une place importante dans les recherches actuelles. En effet, ces dernières années ont été témoin de l'émergence de plusieurs standards bas niveau offrant à la fois une liberté de mouvement ("plus de fil à la patte"), une installation facilitée (notamment dans des lieux où la mise en place de réseau filaire n'était pas envisageable), et surtout une accessibilité grand public de par leur coût modéré. En contrepartie, ces technologies sans fil sont soumises à un certain nombre de contraintes inhérentes pour la plupart à l'utilisation des ondes radio : portée limitée, risques d'interférence, risques accrus au niveau sécurité, etc ...

Aussi, le développement de ces types de technologies est encore actuellement en effervescence, mais on peut toutefois se demander quel est l'impact de leur utilisation sur les couches supérieures du modèle ISO (routage, gestion transparente des connexions/déconnexions, etc ...).

Outre le principal avantage de la mobilité des entités mises en réseau, ces technologies sans fil sont donc particulièrement adaptées à des situations où :

- le nombre d'utilisateurs simultanés est limité,
- la zone de couverture concernée est relativement restreinte,
- le besoin en ressources n'est pas primordial,
- la confidentialité des données manipulées n'est pas cruciale,
- ...

Ainsi, les environnements dits mobiles possèdent de nombreuses disparités par rapport aux environnements filaires. Par exemple, les performances escomptées dans les environnements mobiles

n'égalent pas actuellement celles des réseaux filaires. Par ailleurs, les caractéristiques principales des infrastructures basées sur des réseaux sans fil impliquent l'introduction d'un facteur d'hétérogénéité important (les appareils mis en jeu possèdent des capacités très diversifiées), la notion de variabilité (au niveau des ressources disponibles), et la dynamicité générale (essentiellement due au principe de communication par ondes radio, mais aussi amplifiée lorsque le réseau sans fil sous-jacent est de type ad-hoc).

Aussi, le développement de ces environnements basés sur des technologies sans fil sont générateurs de nouveaux défis. D'une part, des problématiques spécifiques à ces environnements mobiles devront être traitées, mais d'autre part de nombreuses problématiques similaires à celles rencontrées en environnement filaire nécessiteront des méthodes de résolution différentes.

Contexte général

Les développements des architectures de réseaux locaux sans fil tels que WIFI ou Bluetooth, permettent d'envisager de nouvelles applications sur des architectures distribuées hétérogènes. Ces applications sont caractérisées par leur topologie dynamique du fait de la mobilité et des connexions/déconnexions fréquentes de ses constituants. L'utilisateur a ou aura bientôt à sa disposition toutes sortes "d'objets communicants", tels que des téléphones cellulaires, des assistants personnels ou tout autre support futuriste, équipés de ce type de protocoles. Le support exécutif distribué doit donc permettre une reconfiguration dynamique de ses ressources afin d'offrir aux utilisateurs un ensemble de services cohérents et optimaux en terme de performances.

Dans ce contexte, l'exemple des jeux en réseau peut servir de scénario type pour illustrer le style d'environnements visés. En effet, imaginons un jeu en réseau sur des appareils à forte mobilité tels que des téléphones portables communicants par l'intermédiaire d'un réseau sans fil (par exemple Bluetooth). L'objectif commun des membres de ce réseau spontané est de pouvoir s'affronter dans un jeu de stratégie malgré le nomadisme des protagonistes. La mise en place d'un tel environnement de jeu nécessite une configuration préalable : en effet, le serveur de jeu doit être choisi parmi les machines présentes, les équipes doivent être formées, le scénario de jeu défini, etc... Le choix du serveur de jeu n'est pas une tâche triviale : c'est un problème d'élection d'une machine (de préférence la plus performante) parmi plusieurs dans un environnement totalement distribué. Une fois le serveur choisi et les différents paramètres nécessaires définis, la partie peut débuter. Cependant, les fluctuations des performances du réseau engendrée par l'utilisation des ondes radio dans le protocole de communication peuvent être à l'origine de perturbations dans le déroulement du jeu. De plus, un des participants peut être momentanément déconnecté de la partie (sortie de la zone de couverture du réseau) puis revenir sans pour autant que ses partenaires de jeu ou bien les autres participants n'en soient affectés. Aussi, toutes les sortes d'interférences possibles dans un tel environnement doivent être identifiées et prises en compte de manière à ce qu'elles ne gênent pas le bon déroulement de la partie (autrement dit, que se soit transparent au niveau des joueurs). Il faut donc pouvoir assurer la stabilité de l'environnement virtuel ainsi créé malgré la volatilité des connexions et les différentes perturbations rencontrées.

Dans le cadre du projet industriel Sidrah, projet décrit par la suite, un autre scénario caractéristique a été envisagé. Dans ce scénario beaucoup plus sérieux, le thème d'une réunion de travail est abordé afin de permettre l'illustration des avantages offerts par l'environnement Sidrah. Le domaine d'application concerne un environnement de salle de réunion où les appareils peuvent partager les ressources disponibles dans la pièce et fournies par l'infrastructure locale. Dans cet exemple, un utilisateur, un des participant de la réunion de travail, équipé d'un assistant personnel (PDA) rejoint le réseau spontané formé le temps de la réunion. Le PDA, qui possède une capacité d'amorçage fournie préalablement, s'intègre à la communauté d'objets. Via son PDA, l'utilisateur peut profiter des services présents, par exemple le service fournit par un scanner. Ainsi, cet utilisateur a la possibilité de scanner un texte manuscrit et d'obtenir ainsi un fichier graphique de son texte. En utilisant un autre des services présents dans l'infrastructure, ce même utilisateur peut ensuite transformer le fichier graphique ainsi créé en un fichier texte et récupérer le texte sur son PDA. Sur la base de ce scénario très simple, diverses possibilités d'utilisation des services de l'infrastructure sont illustrées : insertion spontanée du PDA, accès à un service local (scanner), médiation à travers un portable et reconnaissance de texte comme service sur un système distant. A partir de ce scénario simpliste, de nombreuses variantes peuvent alors être élaborées, spécialement pour montrer les aspects de résistance aux aléas :

- Aléas au niveau de l'infrastructure de services : on peut montrer, par exemple, que le répertoire de service, s'exécutant initialement sur une machine, peut redémarrer spontanément sur une autre machine dans le cas où la première machine "disparaît".
- Aléas au niveau services : même si le PDA quitte momentanément la communauté, il pourra reprendre automatiquement ses connections aux services au moment ou il réintégrera la communauté.
- Aléas au niveau applicatif : la possibilité pour une application distribuée de continuer à fonctionner en mode dégradé pendant une déconnexion temporaire.

Problématique

Le développement rapide des protocoles de communication sans fil a permis la généralisation de nouveaux "réseaux locaux ad-hoc" principalement caractérisés par leur topologie dynamique. Aussi, les infrastructures logicielles reposant sur de tels environnements - réseaux sans fil en mode ad-hoc composés d'entités hétérogènes - doivent pouvoir s'adapter et gérer cette dynamicité (connexion / déconnexion des entités) ainsi que la forte variabilité des communications (latence, débit, pertes) spécifiques à ces réseaux ambiants. L'objectif général est donc de concevoir des algorithmes permettant de maintenir la cohérence d'un groupe d'entités hétérogènes partageant des services sur de telles architectures fortement dynamiques. La problématique concerne donc la prise de décision en environnement distribué en considérant les particularités du réseau sous-jacent.

Cependant, dans ce contexte (système asynchrone où les processus peuvent subir des défaillances définitives), nous sommes confrontés au résultat d'impossibilité de réalisation du consensus de Fischer, Lynch et Paterson [40]. Aussi, parmi les différentes stratégies proposées afin de pallier ce problème fondamental, l'utilisation d'un mécanisme de détection de défaillances parait adaptée. En effet, un tel mécanisme permet d'une part de garantir la correction des différents ser-

vices de décision répartie et d'autre part, son implantation semble assez flexible pour permettre une mise au point pour l'optimisation des performances de l'intergiciel. De plus, la mise en place d'un module de détection de défaillances permet de concentrer toute l'analyse de la variabilité de l'environnement dans un seul composant. En effet, il est absolument nécessaire de prendre en compte les particularités de l'environnement considéré, et notamment de pouvoir s'adapter aux dégradations éventuelles engendrées par l'utilisation d'un réseau sans fil.

La problématique abordée dans ces travaux est donc double. Elle concerne d'une part la caractérisation des performances envisageables dans un environnement totalement distribué dont le réseau sous-jacent s'appuie sur un protocole de communication sans fil, ainsi que l'identification des différentes perturbations susceptibles d'interférer dans le bon déroulement des applications. Et d'autre part, elle porte sur les problèmes de décision en algorithmique répartie tolérante aux pannes, mais dans un environnement aux caractéristiques particulières.

Contexte Industriel et Scientifique de la thèse

Le projet SIDRAH :

Le projet RNRT SIDRAH [94, 33] (**S**ervices d'**I**nfrastructure **D**ynamique sur **R**éseau local **A**d **H**oc) est un projet exploratoire basé sur la collaboration entre plusieurs partenaires : HP-Labs, France Télécom R&D, Kelua et le laboratoire Informatique et Distribution (ID-IMAG).

L'objectif de ce projet consiste en la mise en œuvre d'une infrastructure permettant d'une part la collaboration spontanée entre différents appareils connectés entre eux par un ou plusieurs réseaux sans fil, l'intégration de l'hétérogénéité des "technologies" et assurant d'autre part la résilience des services offerts. Par résilience, nous entendons la capacité à survivre, éventuellement en mode dégradé, aux aléas.

Plus précisément, ce projet a pour but de permettre à des objets communicants de s'assembler spontanément, ou de s'agréger à une infrastructure existante, et de permettre aux utilisateurs de bénéficier des services présents dans l'environnement virtuel ainsi créé. Pour être utilisable, un tel environnement doit être réellement simple : tous les aspects d'hétérogénéité et de distribution des services doivent être masqués à l'utilisateur. L'infrastructure Sidrah vise à offrir une architecture orientée services où les composants découvrent et publient spontanément des services indépendamment des protocoles de transport et des environnements hétérogènes de déploiement. En outre, l'infrastructure Sidrah offre également des solutions permettant aux objets communicants et aux services offerts de supporter les problèmes liés à la volatilité des connexions.

La résolution du problème de la distribution de services s'est appuyée sur le développement d'un "intergiciel" dédié. Par contre, l'approche retenue face au problème de l'hétérogénéité est basée sur des adaptations des solutions actuelles, en les enrichissant vers les aspects de connexion spontanée et de support de connexions volatiles. Enfin, une solution complète a été implémentée dans un démonstrateur.

Une description plus détaillée est présentée en annexe A, ainsi que la présentation de l'organisation de ce projet exploratoire en sous-projets.

Le projet MIRRA :

Le projet MIRRA (Maintien d'Infrastructure Répartie sur des Réseaux Ambiants), projet de financement de cette thèse, s'appuie sur un contrat de collaboration entre l'INRIA et France Télécom R&D s'inscrivant dans le cadre des activités de développement de briques innovantes du programme *Cybermonde* de France Télécom R&D. L'ambition du programme *Cybermonde* est de mener des recherches technologiques exploratoires dans le domaine du virtuel. Les principaux axes sont : les technologies d'accès (interfaces, objets communicants, identification), la mobilité (immersion, localisation) et l'échange (représentation, codage) entre les objets et utilisateurs de ces mondes virtuels, réels ou augmentés c'est à dire issus d'un mélange réel/virtuel.

Dans ce cadre, le projet MIRRA, en s'appuyant sur des applications existantes, a pour principal objectif de décrire et d'expérimenter des modèles d'architecture pour des environnements faiblement connectés, c'est à dire connectés de temps en temps à l'image des PCs portables et de leur utilisation actuelle. Ces modèles visent à décrire notamment la découverte des postes clients et leur disparition, la définition d'un domaine de recherche d'une ressource ou d'un service, la notion de distance entre plusieurs services afin de trier des réponses trop nombreuses à une requête.

Les appareils connectés en communauté déclenchent des événements : connexion d'un appareil, déconnexion, reconnexion, disparition d'un service distribué, etc... La communauté ainsi constituée doit pouvoir survivre à ces événements. Aussi, ce projet intervient à plusieurs niveaux :

- Au niveau transport puisqu'une communication interrompue doit pouvoir reprendre même si le support physique est différent.
- Au niveau service car un service d'infrastructure doit pouvoir apparaître et disparaître spontanément.
- Au niveau applicatif puisqu'une application distribuée qui subit une déconnexion doit pouvoir continuer à fonctionner en fonction de son niveau de criticité, éventuellement en mode dégradé.

Pour être utilisable, le réseau d'objets doit garantir un niveau de fonctionnalité acceptable sous des conditions d'environnement spécifiées.

Le projet MIRRA, projet prospectif, vise donc à fournir une panoplie de solutions en vue de rendre le comportement de la communauté d'objets stable et fiable.
Dans ce projet, différents rapports techniques d'avancement ont été élaborés [77, 79, 78, 80].

Le projet académique DECORE :

La collaboration entre le laboratoire ID-IMAG et le laboratoire LAG-ELESA réalisée dans le cadre du projet IMAG DECORE (Dimensionnement Et COmmande de REseaux de communication) porte sur le développement d'applications de dimensionnement des ressources et de contrôle des systèmes de communication évoluant en environnement aléatoire.

Deux domaines d'applications sont principalement visés : les protocoles de communication de type TCP et les communications sans fil (réseaux locaux ou ad-hoc sans fil, réseaux de téléphones portables). L'approche méthodologique envisagée est basée sur les méthodes d'évaluation des performances de systèmes à événements discrets évoluant en environnement aléatoire. Les recherches menées dans le cadre de ce projet se situent autant au niveau du développement des modèles adéquats aux systèmes de communication qu'au niveau de la synthèse des algorithmes

d'évaluation des performances.
Plus de précisions, et notamment l'organisation interne du ce projet, sont données en annexe A.

Contributions

Les contributions apportées par ce travail de recherche s'insèrent dans les différents projets présentés précédemment, et interviennent à différents niveaux :

Méthode d'évaluation de l'environnement :

La prise en compte des caractéristiques spécifiques de l'environnement considéré nécessite préalablement une étude qualitative et quantitative précise des particularités de cet environnement. En effet, la réalisation de cette analyse comportementale permet une meilleure connaissance du système mais aussi et surtout permet d'envisager une meilleure adaptation face aux spécificités de l'environnement dans les futurs travaux de développement. L'environnement considéré étant basé sur un réseau sans fil de machines, l'étude comportementale réalisée a porté sur l'expérimentation du protocole 802.11b (WIFI). Dans ce cadre, l'approche a consisté dans un premier temps à la caractérisation du comportement général du réseau, puis à l'évaluation des performances de l'environnement par la mise en œuvre de nombreuses expériences particulières. Bien que réalisée spécifiquement sur le protocole 802.11b, la méthodologie d'approche proposée peut être reproduite et complétée sur d'autres protocoles réseau.

Algorithmique distribuée :

Basée sur la constitution d'une communauté hétérogène de machines, l'infrastructure considérée est totalement distribuée et les entités la constituant peuvent à tout moment subir des défaillances. Aussi, dans ce contexte, des algorithmes permettant de maintenir la cohérence au sein de cette communauté dans laquelle les différentes entités partagent divers services, doivent être utilisés. Cependant, les caractéristiques particulières de l'environnement considéré, environnement basé sur un réseau sans fil, doivent être prises en compte, et notamment les problèmes liés aux perturbations impliquées par le réseau sous-jacent. Aussi, inspiré d'un algorithme existant en algorithmique répartie tolérante aux pannes, un algorithme de consensus spécifiquement adapté à la dynamicité de l'environnement considéré, a été proposé.

Technicité :

Dans le cadre du projet Sidrah, un module de détection de défaillances et un module de service de consensus adapté aux caractéristiques de l'environnement ont été développés et implémentés au niveau middleware. Faisant partie intégrante de l'infrastructure mise en œuvre, ces deux modules, bien qu'étant en constante interaction, ont été développés séparément afin de préserver la flexibilité architecturale. Par exemple, le module de détection de défaillances a été implanté de telle sorte qu'il puisse être utilisé par d'autres modules présents dans l'infrastructure indépendamment du module de service de consensus.

Évaluation de performance :

Une première campagne d'évaluation de performances a été menée lors de l'étude expérimentale du protocole 802.11b. De nombreux tests concernant le paramétrage des détecteurs de défaillances ont également été réalisés, et l'impact de la mise en place du module de détection de défaillances sur l'infrastructure a par ailleurs été évalué. Enfin, l'infrastructure globale telle qu'elle a été implantée (avec les modules de détection de défaillances et de consensus) a fait l'objet de nombreuses expérimentations et évaluations qualitatives et quantitatives.

Organisation du manuscrit

Ce document est divisé en trois parties principales.

Une première partie (partie I) permet d'introduire l'environnement fortement dynamique dans lequel nous nous plaçons. Pour ce faire, une caractérisation relativement générale de l'environnement est proposée au sein des chapitres 2, 3 et 4, permettant ainsi de positionner notre approche contextuelle vis à vis des travaux de recherche gravitant autour du concept "d'objets communicants". Nous présenterons ensuite une étude comportementale de l'environnement considéré dans nos travaux, et ce afin de caractériser précisément ses spécificités (chapitre 5).

La seconde partie (partie II) est principalement consacrée au développement d'algorithmes répartis nécessaires au maintien de la cohérence au sein d'une communauté dans laquelle les entités distribuées partagent des services. Aussi, après avoir défini les contraintes environnementales à considérées du point de vue algorithmique (chapitre 7) et présenté les fondements de l'algorithme de consensus (chapitre 11), nous exposerons le principe du mécanisme de détections de défaillances ainsi que son implémentation (chapitre 9). Une étude permettant de qualifier la qualité de service associée au mécanisme de détections de défaillances est ensuite proposée (chapitre 10). Enfin, le chapitre 11 est consacré à la présentation de l'implémentation d'un algorithme de consensus spécifiquement adapté à la dynamicité de l'environnement considéré.

La dernière partie (partie III) s'attache à présenter les différentes expérimentations et évaluations qualitatives et quantitatives réalisées sur l'infrastructure globale implémentée (le chapitre 13 fait référence au module de détections de défaillances et le chapitre 14 au module de consensus).

I

Environnements Distribués

L'introduction du concept de mobilité au sein des systèmes existants est générateur de nouvelles problématiques. Après un rapide tour d'horizon des nouveaux thèmes de recherche développés dans le domaine de la mobilité au sens large du terme, cette première partie est essentiellement consacrée à la définition des types d'environnements envisagés dans notre problématique. La caractérisation des spécificités d'un environnement totalement distribué et dont le réseau sous-jacent utilise un protocole de communication sans fil, nécessite à la fois une analyse comportementale du système et une évaluation des performances envisageables dans un tel environnement. Un chapitre est donc consacré à la description de l'étude environnementale réalisée et permet ainsi d'illustrer la méthodologie d'expérimentation ayant été mise en œuvre.

Systèmes Répartis 2

2.1 Généralités

Quel meilleur exemple que celui d'Internet pour décrire un système distribué ?
Il s'agit d'une collection d'entités autonomes ("objets communicants") qui communiquent et coopèrent, permettant ainsi l'existence d'une communauté aux caractéristiques particulières.

Les exemples de systèmes distribués sont omniprésents : Internet, mais aussi tout réseaux physiques de machines tels que les réseaux locaux, réseaux d'entités mobiles (téléphones, PDAs, ordinateurs, etc ...), forment un ensemble communément appelé système distribué ou réparti.

Domaine de recherche à part entière mais aussi à la périphérie de nombreux domaines connexes, les systèmes distribués se sont vus proposés différentes définitions dans la littérature :

- "A distributed system is a collection of independent computers that appears to its users as a single coherent system." A. Tanenbaum [102].
- "A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages." G. Coulouris et al [25].
- "We shall use the term distributed system to mean an interconnected collection of autonomous computers, processes, or processors." G. Tel [104].
- "A distributed system is one on which I cannot get any work done because some machine I have never heard of has crashed." L. Lamport.
- ...

Dès lors que les progrès techniques à la fois de l'informatique et des télécommunications ont eu un fort impact sur la structuration même des systèmes, le développement des systèmes distribués [101, 25, 84] a connu un essor important. En effet, motivé tout d'abord par un besoin d'échanger de l'information puis par celui de partager les ressources disponibles, le concept des systèmes répartis est devenu très attractif et est désormais le plus répandu. Toutefois, l'agrégation

de ressources n'est pas le seul intérêt de ces systèmes, leur popularité s'explique aussi par les nombreux avantages offerts :

- **Un accès distant facilité** : la possibilité d'accéder à une ressource sans forcément connaître sa localisation physique ; en fait utiliser une même interface que l'accès soit local ou distant.
- **Une meilleure fiabilité et une disponibilité augmentées** : d'une part la nature du système permet d'assurer une certaine sûreté de fonctionnement (tolérance aux pannes) puisque les entités matérielles sont autonomes, et d'autre part, le principe de la redondance permet d'accroître la disponibilité des services.
- **Des performances accrues** : notamment grâce à la distribution des calculs et des données, mais aussi grâce à la redondance qui offre la possibilité de choisir le service le mieux adapté parmi des services équivalents.
- **Un passage à l'échelle possible** : Internet semble être la meilleure illustration de la grande capacité d'évolution des systèmes distribués puisque le nombre de machines le composant connaît une augmentation impressionnante (d'une centaine d'ordinateurs avant 1980 à plus de 56 millions en 1999).
- **Un coût financier modéré** : en effet, les petits ordinateurs ont un bien meilleur rapport prix/performances que les gros. De plus le coût technique de mise en place est aussi limité.

Cependant, bien que ces systèmes distribués soient pourvus de nombreux atouts, il n'en reste pas moins que leur construction et leur utilisation sont sources de nombreuses difficultés.

2.2 Caractérisation des systèmes distribués

Les divers systèmes distribués ont en commun un certain nombre de spécificités. Ces particularités fondamentales caractérisent à la fois les atouts des systèmes distribués mais correspondent aussi aux difficultés techniques à gérer. L'absence de mémoire partagée et d'horloge globale, due à l'utilisation d'entités distinctes et autonomes, sont en grande partie à l'origine de la complexité du problème de gestion et de maintien de la cohérence entre ces diverses entités. En effet, chaque entité ne peut avoir qu'une vision restreinte de l'état global du système considéré, vision qui en plus d'être limitée peut être aussi quelque fois erronée. La collaboration de plusieurs entités à une action commune peut donc très vite devenir une tâche délicate à réaliser.

Un système distribué se caractérise d'autre part par la dynamicité de sa structure (certains parlent de système "ouvert"). Un tel système est en effet évolutif au cours du temps : de nouvelles entités peuvent se joindre au système existant, d'autres au contraire peuvent le quitter, ... Cette dynamicité de la structure sous-jacente doit être transparente à l'utilisateur, l'insertion et le départ spontanés des entités doivent donc être gérés par les couches intermédiaires afin d'éviter d'engendrer de quelconques perturbations dans les couches supérieures.

De plus, la nature même d'un système distribué détermine une de ces nombreuses spécificités à savoir son hétérogénéité. En effet, chaque entité le composant possède ses propres caractéristiques, la puissance de chacune des entités pouvant être d'ordre de grandeur totalement différent. Ces disparités entre les entités composant le système distribué : variabilité de la puissance des machines (vitesse des processeurs, etc...), différence de capacité (en mémoire, en stockage, en énergie, etc...) doivent être gérées de façon à ce que le système puisse s'adapter à l'entité la moins performante mais aussi dans le même temps bénéficier des avantages fournis par les entités les

plus performantes.

De manière générale, ces disparités entre les entités constituant entre elles un système distribué, viennent conforter le principe d'indépendance des pannes au sein d'un système réparti. En effet, chaque entité, voir même chacun des composants de cette entité, peut à tout instant subir une défaillance quelconque sans que celle-ci ne soit provoquée par une autre entité du système ni même que cette défaillance n'implique celle d'une autre entité ou d'un de ses composants.

2.3 Bilan

Globalement, notre problématique s'inscrit dans le vaste domaine des systèmes distribués et de leurs applications. Cependant, comme nous le verrons dans la suite de ce document, les systèmes distribués basés sur des technologies sans fil sont des systèmes distribués bien particuliers. En effet, bien que partageant les mêmes principes de base que les systèmes distribués dits classiques (communications filaires), les systèmes distribués construits grâce à une ou plusieurs technologies sans fil présentent d'autres particularités. Ainsi, en plus de devoir prendre en compte les différents aspects de "mobilité", ces *nouveaux* systèmes distribués doivent également s'adapter aux contraintes introduites par l'utilisation d'un protocole de communication sans fil.

La dynamicité dans les systèmes distribués 3

Ces dernières années ont été témoins de l'intérêt grandissant pour le nouveau concept de l'informatique mobile. En effet, motivé essentiellement par le développement important et la multiplication des équipements mobiles (PDAs, ordinateurs portables, téléphones, etc...), et principalement leur miniaturisation, ce besoin de mobilité a déclenché l'évolution des réseaux et permis aux technologies sans fil de voir le jour. Aussi de nombreux travaux de recherche s'orientent vers la gestion de la mobilité dans leur environnement ou se focalisent sur le traitement des spécificités des environnement mobiles et sans fil.

3.1 Généralités

La mobilité informatique est un terme générique souvent employé pour désigner le nomadisme des utilisateurs. Cependant, sous ce terme de mobilité, différents concepts sont englobés. En effet, de manière générale, la mobilité regroupe à la fois la possibilité qu'un utilisateur se connecte en différent lieux géographiques en utilisant son environnement personnel (nomadisme), la capacité pour ce même utilisateur de bénéficier des ressources et services présents, les avantages liés aux technologies sans fil, etc...

Dans la littérature, un environnement mobile correspond dans la majorité des cas à un environnement préexistant (stations fixes reliées entre elle par un réseau filaire) sur lequel viennent se greffer des entités mobiles grâce à la mise en place d'un réseau sans fil. De notre point de vue, un environnement mobile est bien plus général puisqu'il regroupe à la fois les environnements cités précédemment et les environnement formés exclusivement d'entités mobiles connectées entre elles par un ou plusieurs réseaux de communication sans fil. Ce deuxième type d'environnements mobiles peut être qualifié d'environnement mobile ad-hoc c'est à dire où aucune infrastructure fixe n'est présente. Outre l'absence d'entités fixes permettant une certaine centralisation, les réseaux ad-hoc mobiles présentent certains inconvénients, ou tout du moins difficultés techniques, à savoir notamment leur topologie non prédictible.

Quelque soit l'environnement mobile considéré, un certain nombre de caractéristiques leur sont communes. En effet, mis à part les problèmes engendrés par la gestion de la nomadicité au sens large du terme, les environnements mobiles sont soumis à de nombreuses contraintes différentes de celles rencontrées dans les réseaux filaires. A titre d'exemples, l'accès aux ressources et services présents quelque soit le lieu géographique implique forcément la nécessité de gérer les changements d'adressage. Ou bien pour autre exemple, on peut citer la gestion délicate de la pertinence d'un service en fonction de la localisation géographique de l'entité mobile.

L'utilisation d'un réseau sans fil : Les principales contraintes des technologies sans fil concernent leur bande passante disponible qui est relativement faible et partagée entre les entités du réseau, ainsi qu'une portée de transmission limitée. De plus, les communications étant réalisées exclusivement par ondes radio, de nombreuses interférences peuvent affecter les signaux transmis ce qui a pour effet de soumettre la bande passante à de nombreuses variations. Aussi, l'environnement construit au-dessus d'un réseau sans fil peut être caractérisé essentiellement par sa variabilité et sa sensibilité au contexte extérieur.

L'utilisation de terminaux mobiles : De manière générale, la capacité des terminaux mobiles (PDAs, téléphones, ...) est souvent limitée. En effet, malgré les progrès technologiques en matière de miniaturisation, et bien que les capacités d'un ordinateur portable soit désormais équivalentes ou presque à celles d'une station fixe, les ressources mémoire ou la puissance de calcul d'un PDA par exemple reste relativement restreintes. A cela viennent s'ajouter les problèmes d'autonomie de ces entités mobiles, leur capacité limitée en énergie représentant une contrainte non négligeable.

Le déplacement des entités : Introduite par la terminologie des réseaux sans fil, la notion de zone de couverture permet de délimiter le périmètre dans lequel une entité a la possibilité de communiquer directement avec une autre. Autrement dit, une entité ne peut pas communiquer avec une autre entité extérieure à sa zone de couverture (modulo bien sûr l'existence d'une troisième entité dont la zone de couverture couvre les deux premières entités et la mise en place de mécanismes de routage). Ainsi, cette notion permet de se rendre compte de toute l'importance du mouvement des entités les unes par rapport aux autres et de ses conséquences. Une des conséquences de ces mouvements même minimes des entités est la dynamicité à laquelle est soumis le système considéré.

De notre point de vue, l'environnement mobile considéré se restreint à un réseau ad-hoc sans fil composé d'entités mobiles hétérogènes, mais où la mobilité se limite à l'éloignement ou le rapprochement d'une ou de plusieurs entités par rapport aux autres. En effet, on ne cherche pas à gérer la mobilité au sens nomade du terme, mais simplement à s'adapter aux contraintes générée par la forte variabilité introduite dans ce type d'environnement mobile.

3.2 État de l'art

Ces dernières années, un grand nombre de travaux traitant de problématiques diverses et variées en rapport avec le concept de mobilité ont été développés. L'émergence de nombreux projets dans des domaines différents, projets focalisés essentiellement sur l'étude des spécificités des environnements mobiles et leur gestion, a favorisé le développement de solutions basées sur des approches différentes et à divers niveaux dans le modèle de référence OSI. Cependant, deux approches principales peuvent être distinguées [60] : d'une part les approches qui consistent à tout mettre en œuvre pour masquer la mobilité aux couches supérieures, et d'autre part les approches qui, au contraire, cherchent à adapter les applications aux phénomènes induits par la mobilité.

La première approche générale qui vise à rendre les déplacements des entités mobiles les plus transparents possibles, a fait l'objet de nombreux travaux. Ces travaux ont notamment été menés sur les problèmes d'adressage IP (MobileIP [92], IPv6 [28, 36], Mobile-IPv6 [64, 93]), les difficultés liées au routage dans les environnements mobiles (groupe de travail MANET [63], le groupe de recherche CReWMan [27], un des thèmes de recherche du laboratoire WAM (Wireless Adaptive Mobility) [73]), etc...
De plus, le principe du masquage de la mobilité a été abordé au sein de nombreux projets et les solutions proposées s'appliquent à différents niveaux. Ainsi, par exemple le système de fichiers Coda [69, 85, 68] propose de dissimuler les déconnexions au moyen de mécanismes de cache afin de ne pas avoir à modifier les applications existantes. Dans le domaine du support réseau, le projet Monarch [61, 62] vise quant à lui à permettre aux hôtes mobiles de communiquer de manière transparente entre eux ainsi qu'avec des hôtes statiques ayant des connexions filaires, tout en privilégiant l'efficacité dans l'utilisation de la connectique.

Toute aussi prisée, la seconde approche qui consiste en une adaptation en fonction des caractéristiques introduites par la mobilité, a été abordée selon différents points de vue. En effet, cette seconde approche peut être divisée en plusieurs types d'adaptations selon que l'on cherche à s'accommoder de la variabilité de l'environnement mobile (latences plus ou moins importantes, phénomènes de déconnexions momentannées, ...), ou à gérer les changements d'environnements (adaptation en fonction des services proposés, localisation des ressources, ...), ou encore à s'adapter à l'hétérogénéité des entités mises en jeu.

Le projet Odyssey [88, 87, 89] par exemple s'inscrit dans ce cadre. Basé sur une collaboration étroite entre le système et les applications, l'objectif principal du projet Odyssey est d'avoir une gestion fine des ressources (bande passante, espace mémoire, CPU, ...) afin de permettre aux applications de pouvoir s'adapter en fonction de leurs besoins. Pour ce faire, la plateforme logicielle Odyssey propose entre autres aux applications un système de notification leur permettant de prendre en compte les informations liées à l'état de l'environnement d'exécution et de s'adapter dynamiquement à ses changements.

Le projet Timely (The Illinois Mobile Environments Laboratory) [9, 15] qui s'inscrit lui aussi dans le domaine de la mobilité, vise à la création d'algorithmes ainsi qu'à la construction d'une infrastructure permettant de fournir une qualité de service dans les environnements de réseaux sans fil hétérogènes. Les problèmes de définition de services, de réservation de ressources, d'adaptation en fonction des ressources sont ainsi abordés. L'architecture de gestion adaptée des ressources, proposée en environnement mobile intervient à différents niveaux, de la couche MAC à la couche

transport dans le modèle OSI [83, 67].

Le projet Bayou [30] quant à lui, propose de fournir aux applications une adaptation spécifique afin de pallier aux problèmes de déconnexions. Les applications visées par ce projet correspondent essentiellement aux bases de données partagées, mais aussi à toutes applications ayant des données partagées [35]. Ces applications ne nécessitant pas une connexion continue, des mécanismes de réplication cohérente des données et de résolution de conflits ont été mis en œuvre [105].

Intervenant aux niveaux des couches basses à réseau, le projet MosquitoNet [113, 114] vise à optimiser et adapter les envois de paquets en fonction des besoins applicatifs et de l'utilisation du réseau [21]. Parallèlement, dans ce projet, des travaux ont été menés sur l'analyse de traces dans des réseaux sans fil de type LAN et MAN (Local Area Network et Metropolitan Area Network) [103] afin de mettre en évidence les comportement des utilisateurs face à un environnement mobile.

Beaucoup d'autres projets ayant développés des problématiques liées à la mobilité pourrait être cités. Cependant, devant l'essor qu'à connu ce sujet de recherche ces dernières années et face à l'amplitude des domaines de recherche concernés, une présentation exhaustives des projets réalisés et en cours ne saurait être faite. Néanmoins, une présentation plus précise a été développée dans [72].

3.3 État de l'art au niveau middleware

Dans un souci d'adéquation avec les objectifs fixés en collaboration avec les partenaires industriels (projets Sidrah, MIRRA), l'approche considérée s'est progressivement orientée au niveau middleware. Le positionnement de la problématique au middleware a été principalement motivé par les caractéristiques spécifiques d'un tel environnement d'exécution. En effet, un middleware possède des propriétés particulières de portabilité et de généricité. De plus, il permet de masquer l'hétérogénéité des machines et des systèmes, ce qui paraît être un atout pour la gestion de l'architecture évolutive que l'on s'est fixée.
Le middleware est un "support logiciel permettant aux utilisateurs d'exécuter des applications distribuées sur un ensemble de machines de façon transparente, indépendamment des réseaux sous-jacents et des architectures matérielles ou logicielles des différentes machines impliquées".

La notion de transparence est primordiale au sein des middlewares :

- La transparence au niveau de la localisation, permettant l'accès et l'invocation des opérations sur des objets quelque soit le site auquel ils appartiennent.
- La transparence par rapport à l'abstraction de programmation qui permet d'implanter les fonctionnalités d'un objet avec le langage de son choix.
- La transparence vis à vis de l'hétérogénéité sous-jacente (des matériels utilisés, des protocoles de communication, des systèmes d'exploitation, etc...).

Ainsi, le niveau d'abstraction offert par la couche middleware est particulièrement attractif pour la gestion des problématiques liées à la mobilité. En effet, la couche middleware paraît la plus adaptée à supporter une connectivité instable et à gérer les changements dynamiques de contextes. Aussi, de nombreux projets se sont focalisés sur le concept de mobilité et de maintien de connexion et ont proposé des solutions au niveau middleware.

La mise en œuvre et les solutions apportées par les projets cités dans la suite de ce document dépendent largement du contexte applicatif et des technologies middleware utilisées. Cependant, le choix de présenter certains projets est totalement arbitraire et ne dénote en aucun cas des retombées scientifiques de ces différents projets.

Projet VIVIAN :

Regroupant différents partenaires (Nokia, Philips, INRIA, INT, ...), le projet ITEA Vivian a comme objectif général "l'ouverture des plates-formes mobiles pour le développement d'applications à base de composants" [23]. Aussi, les principales motivations de ce projet concourent à proposer des services middleware pour différents domaines d'application et à fournir une plate-forme adaptée aux divers supports mobiles. De plus, une part importante de ce projet est essentiellement consacrée au problème de la gestion des déconnexions volontaires et involontaires pouvant interférer dans le bon déroulement des applications. L'idée maîtresse est d'une part de fournir une interface simple facilitant les déconnexions et reconnexions volontaires, et d'autre part, de rendre les déconnexions involontaires transparentes à l'utilisateur.

Ainsi, afin de permettre aux applications distribuées d'avoir une continuité de service malgré les déconnexions pouvant survenir, l'adaptation concerne l'introduction de deux mécanismes CORBA[1] : l'utilisation du passage d'objets par valeur (pour la gestion des déconnexions volontaires) et l'utilisation d'intercepteurs portables (pour la gestion des déconnexions involontaires). L'illustration de ces mécanismes de fonctionnement mis en place sur une application de messagerie électronique est donnée dans [24].

Projet CESURE :

Axé sur "la modélisation et l'exploitation de la notion d'application de service aux usagers" éventuellement mobiles du réseau, le projet CESURE (Configuration et Exécution de Services pour les Usagers mobiles des Réseaux Étendus) fait intervenir divers partenaires : Gemplus, INRIA, LIFL, INT [8]. Orientant son approche du côté de l'utilisateur mobile, ce projet a pour objectif principal de permettre aux usagers mobiles de se connecter à une application répartie offrant divers services en tous lieux et depuis des terminaux aux caractéristiques diverses (ordinateurs, téléphones, PDAs, ...). Le principe général de ce projet repose sur l'utilisation d'une carte à puce contenant les informations relatives à l'usager et agissant comme point central pour l'application. Ainsi, l'utilisateur mobile via la carte à puce devient porteur de son environnement d'accès aux services sur le réseau puisque la carte offre la possibilité de stocker à la fois la description de la configuration des services et d'en gérer les accès [95]. L'architecture de la plateforme ainsi développée décrit l'interconnexion entre une carte à puce et un terminal mobile, le lien entre le portail de déploiement contenu sur le terminal et une machine de déploiement (susceptible de ne pas être installée sur le terminal), ainsi que les accès, via la machine de déploiement, aux serveurs d'exécution, au service de recherche et aux étagères électroniques (ou bibliothèques de composants) [11]. Deux aspects importants ont notamment été abordés : la répartition de charge qui permet d'une part d'optimiser les ressources disponibles sur le réseau et d'autre part d'obtenir un déploiement plus performant, et la gestion des déconnexions (volontaires et involontaires) via un service de duplication des composants.

[1] Common Object Request Broker Architecture

Projet ALICE :

Le projet ALICE (Architecture for Location Independent CORBA Environment) propose une architecture permettant aux applications CORBA[2] d'être supportées dans un environnement mobile [75], ainsi que dans d'autres infrastructures telles que SOAP, DCOM et Java RMI. L'architecture ainsi présentée dans ce projet tient compte des mouvements des entités et assure que les connexions client-serveur restent établies de manière transparente pour l'utilisateur. La gestion de la mobilité intervient à différents niveaux : (1) gestion de la connectivité entre un hôte mobile et une station de base [47], (2) la gestion de la localisation des serveurs sur les hôtes mobiles et (3) la gestion des opérations de déconnexions d'un hôte mobile [48]. Grâce à l'ajout d'une nouvelle couche dans la hiérarchie, la couche D/IIOP (Disconnected Internet Inter-ORB Protocol), les opérations de déconnexions peuvent être supportées.

Un grand nombre d'autres projets font également référence aux environnements mobiles, et ceci au niveau middleware.
Ce rapport n'a pas pour but de donner une liste exhaustive de tous les projets intervenant dans ces domaines de recherche, mais l'on peut cependant en citer quelques uns : DOLMEN [109], MosquitoNet [7], Odyssey [1, 88, 87], Monarch [6, 61], BARWAN [66], MOWGLI [10], Rover [65], etc...

De plus, toujours au niveau middleware, les travaux concernant les problèmes liés au nommage, à la découverte de ressources ainsi qu'à la localisation des ressources tels que Jini [99, 100], UPnP[3] [110, 82], SLP[4] [46], Salutation [96], etc... proposent de nouvelles stratégies permettant une adaptation aux environnements mobiles.

3.4 Conclusion

La diversité des travaux présentés dans ce chapitre illustre bien l'ampleur du concept d'informatique mobile dans de nombreux domaines de recherche. Cependant, devant le grand nombre de problématiques nouvelles soulevées par ce concept, il semble indispensable de spécifier précisément le contexte dans lequel nous nous plaçons ainsi que les contraintes qui lui sont associées. En effet, nos travaux de recherche s'inscrivant en collaboration avec des partenaires industriels (projet Sidrah et projet Mirra, chapitre 1), les problématiques retenues se situent essentiellement dans les couches hautes du système et s'intéressent principalement au problème de la gestion des déconnexions fréquentes, lesquelles dénotent directement le comportement typique des réseaux mobiles.

[2] Common Object Request Broker Architecture
[3] Universal Plug and Play
[4] Service Location Protocol

Supports sans fil 4

"En fait, la communication numérique sans fil n'a rien d'une idée neuve. En 1901, déjà, le physicien Guglielmo Marconi fit sur un bateau la démonstration d'une liaison télégraphique sans fil avec la côte en utilisant le code Morse (les points et les traits forment bien un système binaire). Les systèmes numériques d'aujourd'hui ont certes de meilleures performances, mais l'idée reste la même [42, 91]" A. Tanenbaum [101].

4.1 Généralités

Aujourd'hui, un grand nombre de "nouvelles" technologies portant sur le principe des communications sans fil sont apparues et connaissent une popularité grandissante. Pour ne donner qu'un bref aperçu des différents types de supports sans fil, on peut citer par exemple les réseaux basés sur les ondes infrarouges (standard IrDA [58]), les réseaux cellulaires (GPRS[1] [2], GSM[2] [3], UMTS[3] [53]), les réseaux satellites, les réseaux locaux sans fil tels que les WPAN[4] (Bluetooth [106, 49], HomeRF [54]) ou les WLAN[5] (WIFI [5, 57], HiperLAN [4]).

Bien qu'ayant au moins en commun le principe de communication sans contrainte filaire, ces différentes technologies divergent cependant tant au point de vue de leur concept initial (des motivations différentes sont à l'origine du développement de ces diverses technologies) que sur leurs caractéristiques spécifiques et les problématiques qu'elles engendrent. Dans le cadre de nos collaborations industrielles, les problématiques considérées ont porté exclusivement sur les réseaux locaux sans fil, et plus précisément ceux basés sur les technologies Bluetooth et WIFI (Annexe B et Annexe C).

[1] Global System for Mobile communication
[2] General Packet Radio Service
[3] Universal Mobile Telecommunication System
[4] Wireless Personal Area Network
[5] Wireless Local Area Network

4.2 Des systèmes distribués particuliers

L'émergence de ces technologies sans fil a permis le développement de nouveaux types de réseaux basés sur l'utilisation des protocoles sans fil. Ces nouveaux réseaux diffèrent en plusieurs points des réseaux filaires classiques et possèdent chacun leurs propres spécificités.

Réseaux sans fil complémentaires :

On regroupe sous cette appellation les réseaux sans fil venant s'inscrire en complément d'une infrastructure filaire préexistante, ou ceux nécessitant une certaine infrastructure statique telle qu'un réseau 802.11 dans lequel la gestion des communications s'effectue par le biais d'une ou plusieurs bornes d'accès (voir Annexe B). Ce type de réseaux est souvent déployé dans un lieu précis afin d'offrir à une population de passage une possibilité d'accès à diverses informations. L'infrastructure statique sous-jacente dans ce type de réseau facilite quelque peu la résolution de problématiques liées aux particularités des réseaux sans fil.

Réseaux sans fil ad-hoc :

Également qualifiés de réseaux spontanés, les réseaux ad-hoc sans fil consistent en une collection d'entités éventuellement hétérogènes et mobiles qui peuvent communiquer entre elles via une ou plusieurs interfaces réseau. Les différentes entités composant un réseau ad-hoc sont toutes indépendantes entre elles, et aucune n'a pour vocation la gestion du réseau lui-même. A savoir l'existence du réseau ad-hoc n'est soumis à aucune forme de centralisation. L'étude de ces réseaux a donné lieu à de nombreux travaux dont notamment ceux menés par le groupe de recherche MANET (Mobile Ad-hoc Networks) de l'IETF (Internet Engineering Task Force).
Un exemple type de ces réseaux est donné par le protocole sans fil 802.11 qui propose deux modes de fonctionnement dont un mode de type ad-hoc (voir Annexe B).

Réseaux de capteurs :

Un réseau de capteurs est par définition un système composé d'un grand nombre d'entités (souvent plusieurs milliers) nommées nœuds et dédiées à une application. Les caractéristiques spécifiques de ce type de réseaux se situent essentiellement au niveau des particularités des nœuds le composant. En effet, chaque nœud est caractérisé entre autres par sa zone de couverture réduite, sa capacité énergétique limitée tout comme sa capacité de calcul ou sa mémoire, ainsi que par sa non fiabilité puisqu'il arrive fréquemment qu'un nœud soit enclin à une défaillance. D'autre part, un nœud n'est pas nécessairement identifié globalement. Par ailleurs, la topologie d'un réseau de capteurs est dans la grande majorité des cas beaucoup plus changeante que celle des réseaux ad-hoc.
Globalement, on retiendra essentiellement que contrairement aux réseaux ad-hoc où chaque entité évolue indépendamment des autres, les nœuds d'un réseau de capteurs collaborent à un objectif commun consistant la plupart du temps à la collecte et au traitement des informations provenant de l'environnement. De plus, la principale problématique dans ces réseaux de capteurs concerne le principe de limitation des dépenses énergétiques.

Suivant le type de réseaux sous-jacents, le système développé doit prendre en compte différentes contraintes impliquées et savoir tirer partie des apports divers.
Dans notre contexte, l'environnement considéré s'appuie précisément sur un réseau local ad-hoc mettant en œuvre un protocole de communication sans fil tel que 802.11. Ce type d'environnements apparaît comme très attractif car il est à la fois facilement déployable, peu coûteux et extensible à souhait. Cependant, outre les problèmes directement liés au mode de communication sans fil, la topologie non prédictible de ces environnements, l'absence de point de centralisation et l'hétérogénéité des entités mise en jeu ainsi que leurs capacités limitées, donnent un aperçu des difficultés à considérer.

4.3 Les difficultés

Nous ne considérons pas ici les problèmes engendrés par les contraintes typiques des réseaux ad-hoc, mais seulement ceux rencontrés dans les réseaux ad-hoc sans fil.
En effet, outre les difficultés liées aux réseaux ad-hoc, l'introduction du sans fil comme protocole de communication n'est pas sans conséquence puisqu'elle implique inexorablement l'existence de nombreux facteurs susceptibles d'affecter le bon déroulement du système ainsi créé.

La non fiabilité des communications :

Dire que le mode de communication s'effectue grâce à la propagation d'ondes radio dénote implicitement les particularités des réseaux ad-hoc considérés. En effet, ce mode de communication affiche une forte sensibilité au contexte environnemental et est donc directement soumis à de nombreuses interférences (ou perturbations extérieures) venant gêner la propagation des ondes. L'impact de ces phénomènes sur notre environnement distribué sera développé dans le chapitre suivant (chapitre 5). Cependant, on peut déjà noter que ces perturbations extérieures ne font qu'ajouter à l'instabilité générale de l'environnement considéré. Le modèle de communication envisagé doit donc prendre en compte les possibles pertes de messages. Aussi, la non fiabilité du médium de communication étant une caractéristique spécifique des réseaux ad-hoc sans fil, les traitements apportés par la suite devront donc s'adapter à cette contrainte.

Les performances restreintes :

Comme nous l'illustrerons plus en détail au cours du chapitre suivant (chapitre 5), le partage de la bande passante ainsi que la forte variabilité des latences (due à différents facteurs : types d'entités, distance entre les entités, les modes d'économie d'énergie, la charge du réseau, etc...) peuvent être à l'origine de délais importants dans les communications. Aussi, l'infrastructure développée doit être capable de s'accommoder de cette forte variabilité des temps de communication.

Le problème des déconnexions :

De par sa définition même, un réseau ad-hoc possède une topologie dynamique : l'arrivée ou le départ d'une entité sont des opérations classiques dans ce type d'environnements. Cependant, la technologie sans fil considérée dans notre environnement est génératrice d'une tout autre dynamicité. En effet, comme nous l'avons précédemment noté, la mobilité des entités mises en jeu peut engendrée des déconnexions intempestives (ou involontaires) correspondant à des sorties de zone

de couverture. Ces déconnexions intempestives étant relativement fréquentes, leur prise en considération et leur gestion sont indispensables. Par ailleurs, à ces déconnexions intempestives des entités fait suite la plupart du temps les reconnexions des mêmes entités. Aussi, l'approche choisie devra également pouvoir gérer ou tout du moins s'adapter à ces déconnexions/reconnexions afin que celles-ci soient totalement transparentes aux couches supérieures.

4.4 Conclusion

Aussi, l'environnement considéré correspond à un système totalement distribué basé sur une architecture évolutive de type ad-hoc et sur une technologie de communication sans fil. Comme nous l'avons vu au cours du chapitre précédent (chapitre 3), le traitement retenu se situe au niveau middleware. Le principal problème à gérer concerne les déconnexions intempestives. En effet, face à une architecture évolutive au cours du temps et la forte variabilité introduite par le protocole de communication sous-jacent, le challenge à relever consiste à s'adapter à la dynamicité ambiante de façon à ce que les fréquentes déconnexions soient autant que possible transparentes au niveau de l'applicatif.
Plusieurs types d'évolution de l'architecture peuvent être envisagés :

- **évolution de l'ensemble des ressources de calcul et de mémoire impliquées dans l'application :** par exemple un nouveau serveur se connecte à l'application, un portable se déconnecte, un PDA se met en veille,...
- **évolution de l'ensemble des ressources de communications utilisées par l'application :** le réseau sature, des perturbations externes influent sur les latences d'un réseau sans fil,...
- **évolution des ressources d'entrées/sorties :** dispositifs d'interaction avec les utilisateurs, accès à des média de stockage,...

Dans la suite, nous ne prendrons pas en compte les évolutions liées aux entrées/sorties et nous considérerons principalement des architectures avec connexions/déconnexions de machines et des dégradations de la qualité de service du réseau d'interconnexion (modifications des latences et des débits). La difficulté est donc de prendre en compte cette évolution et de faire en sorte que le middleware fournisse, au niveau applicatif, la meilleure qualité de service possible. Il parait donc nécessaire d'anticiper au mieux l'évolution et de mettre en place des mécanismes efficaces permettant la reconfiguration, que ce soient des techniques de récupération à partir de points de reprise, des anticipations par duplication lorsque le réseau devient critique, l'usage de caches, etc...

Dans le contexte des applications distribuées à base d'objets, nous dirons qu'il y a déconnexion lorsque l'on n'obtient pas de réponse à l'invocation d'un objet distant. Il est clair qu'avec cette définition aucune propriété de déconnexion ne pourra être établie à moins d'obtenir la confirmation de la destruction de l'objet. Par exemple, la rupture d'une connexion TCP ne peut pas être interprétée directement comme une déconnexion au niveau middleware, puisqu'en effet celle-ci peut être causée par une perturbation du réseau, une mise en veille momentanée d'un PDA, etc...

Le middleware est donc obligé de faire une approximation de l'état global du système (l'objet est-il accessible ou non), information à laquelle il ne peut avoir réellement accès.
Aussi, une première approche consiste à restreindre la déconnexion à une fenêtre temporelle. Cela

revient à armer une temporisation à chaque demande d'accès à un objet distant et à décider que la liaison est rompue si la réponse n'arrive pas dans un délai imparti.
Une autre approche base la décision sur des informations issues des couches plus basses du système, qui changent des variables d'état du middleware (par exemple des estimations de latence/débits sur le lien utilisé pour l'accès à distance).
Dans ces deux approches, la réelle difficulté est la mise au point des paramètres algorithmiques (réglages des temporisations, analyse des informations provenant des niveaux inférieurs, etc...).

Méthodologie d'expérimentation : Caractérisation quantitative de l'environnement distribué

5

L'objectif de ce chapitre est d'exhiber et d'analyser les comportements complexes de systèmes distribués dans un environnement sans fil hétérogène. Pour ce faire, l'étude des performances du protocole de communication est alors une étape indispensable. En effet, l'analyse des latences de communication, de la gigue, des taux de perte et les débits offerts par un protocole permet d'adapter les applications et d'améliorer significativement la qualité de service offerte par les couches supérieures. Afin d'offrir de meilleures performances, tout système doit être adapté à son environnement, de plus la bonne maîtrise des communications réseau est un atout majeur pour le développement de middlewares distribués.

Cependant, l'identification et l'analyse des phénomènes introduits par les technologies sans fil n'est pas une tâche triviale. En effet, étant donné le grand nombre de facteurs pouvant influencer le comportement général du réseau sans fil ainsi mis en place (nombre de terminaux présents, types de terminaux, distance entre les différents terminaux présents, ...), cette étude peut très vite s'avérée complexe. Aussi, l'idée maîtresse de cette démarche était de proposer une méthodologie qui permettrait par la suite de renouveler facilement cette analyse quelque soit la technologie sous-jacente ou les paramètres du contexte expérimental pris en compte.

5.1 Préliminaires

Dans le cadre de cette étude expérimentale, un réseau ambiant hétérogène s'appuyant sur la technologie WIFI a été déployé. Afin d'appréhender le comportement du protocole 802.11b et sa qualité de service, une plate-forme hétérogène sans fil basée sur le protocole 802.11b a donc été mise en place en vue d'évaluer ses performances. Pour respecter la contrainte d'hétérogénéité, cette plate-forme d'évaluation ainsi établie s'est composée de différents types d'appareils commu-

nicant par voie hertzienne.

Dans cette approche, nous avons choisi d'étudier les latences de communications ainsi que les taux de pertes en vue de la nécessité ultérieure d'ajustement de temporisations. De plus, l'étude des débits nous permettra de connaître la taille des paquets la mieux adaptée aux types de connexions utilisées (UDP ou TCP) ainsi qu'aux distances entre les stations. Suivant le type d'application, le type de connexion et/ou la taille des paquets pourront varier. Cependant, la gigue ne sera pas étudiée puisque sa connaissance ne présente que peu d'intérêt pour le projet (qualité de service importante surtout pour le trafic audio et vidéo).

5.1.1 Le contexte expérimental

Architecture

La technologie WIFI définit deux types d'architectures réseaux [108] : le mode infrastructure et le mode ad-hoc.

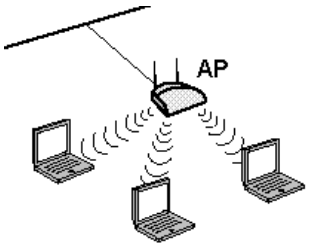


Figure 5.1 *Le mode Infrastructure*

Ce mode est basé sur une architecture cellulaire (le système est subdivisé en cellules) semblable à celle employée pour les téléphones portables, et où chaque cellule (appelée Basic Service Set ou BSS dans la nomenclature 802.11) est contrôlée par une station de base (appelée Access Point ou AP, Point d'Accès en français). Les points d'accès sont chargés des services d'authentification et d'association. Le réseau local sans fil peut être formé par une cellule unique, avec un seul Point d'Accès.

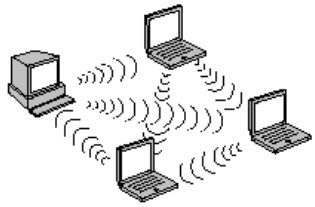


Figure 5.2 *Le mode Ad-Hoc*

Pour ce mode, aucune infrastructure supplémentaire n'est nécessaire. Les stations communiquent entre elles de façon autonome afin d'assurer leur connectivité et celle des autres stations par voie hertzienne. La rapidité de déploiement et la mobilité physique des stations permettent une grande souplesse d'utilisation (pas de raccordement, pas de câbles,...). Les stations doivent être capables d'effectuer toutes les opérations nécessaires à l'établissement et au maintien du réseau comme les procédures d'authentification et d'association. Le réseau ad-hoc minimal est constitué de deux stations dans la zone de couverture radio l'une de l'autre. Cette architecture est aussi appelée IBBS (Independent Basic Service SET).

Pour davantage de détails concernant ces deux types d'architecture, voir l'annexe B. Ces deux modes de fonctionnement étant conceptuellement différents, il est à présager que les performances mesurées dans chacune de ces architectures soient quelque peu divergentes.

Plate-forme d'évaluation

La plate-forme d'évaluation mise en place se compose de plusieurs terminaux mobiles, ordinateurs portables et assistants personnels (PDAs), reliés entre eux par un réseau sans fil 802.11b. Les différents tests réalisés ont été effectués d'une part sur un réseau de type ad-hoc (sans borne d'accès) et d'autre part sur un mode d'architecture infrastructure (avec un point d'accès de type Compaq WL410).

	PDA 1	Ordinateur portable	PDA 2
Référence	HP Jornada 568	Compaq armada M700	Compaq iPAQ H3800
Processeur	ARM 206 MHz	PIII 1Go	StrongARM 206 MHz
Mémoire	64 MB RAM 32 MB ROM	128 MO	64 MB SDRAM 32 MB FlashROM
Système d'exploitation	Windows CE v3	Linux (kernel 2.4.18)	Linux Debian
Cartes sans fil 802.11b	CompactFlash SocketCom	PCMCIA Compaq WL110	PCMCIA Compaq WL110

Figure 5.3 *Fiche Technique des stations utilisées*

5.1.2 Premières observations

Suite à la réalisation d'expérimentations préliminaires, quelques mesures de performances telles que la latence moyenne entre deux terminaux ou le débit possible suivant le mode d'architecture (ad-hoc ou infrastructure) ont été mises en valeur. Ainsi, dans des conditions optimales[1] d'utilisation, les débits relevés vont de 1.6 Mbits/s pour les PDAs de type Jornada jusqu'à 5.5 Mbits/s pour les ordinateurs portables ou les PDAs nouvelle génération, et les latences sont inférieures à 10ms [2]. Cependant, les performances se dégradent rapidement lorsque la distance entre les stations augmente ou lorsque le réseau est surchargé.

D'autre part, les taux de pertes observés peuvent être élevées et de faibles débits ont pu être constatés. En effet, en mode ad-hoc, les ordinateurs portables perdent la connectivité vers les PDAs lorsque ceux-ci sont en mode d'économie d'énergie.

Bien que ces premières expérimentations aient permis une identification rapide des caractéristiques significatives de ces réseaux ainsi qu'une analyse empirique, d'une manière générale, elles ont essentiellement tendu à illustrer la difficulté à réaliser une évaluation des performances complète dans un environnement soumis à de nombreuses influences. Une étude complémentaire plus structurée doit être menée afin de mettre en évidence les principaux facteurs ainsi que leurs interactions pouvant avoir une influence significative sur les performances du protocole 802.11b.

[1] dans une même pièce, sans économie d'énergie

[2] la première sollicitation est souvent supérieure à 10ms

5.2 Méthode

Cette section a pour objet la présentation de la démarche expérimentale suivie. Afin de généraliser l'approche à effectuer, la méthodologie proposée vise à permettre d'une part la mise au point et la réalisation d'expériences, et d'autre part, l'analyse des phénomènes observés ainsi que l'identification des paramètres intéressants.

Afin d'une part de pouvoir quantifier les performances du protocole 802.11b sur une plateforme hétérogène et d'autre part de mettre en évidence les paramètres pouvant influencer fortement sur ce système, des plans d'expériences ont été élaborés.

5.2.1 Plans d'expériences

Face à un système dont on ne connaît pas le fonctionnement, mais que l'on souhaite maîtriser voire optimiser, une solution s'impose : définir une campagne d'essais. Pour notre étude, nos objectifs étaient de mieux connaître le processus afin de déterminer les facteurs influents sur le système.

Les plans d'expériences [111] ainsi élaborés, ont été construits suivant deux démarches expérimentales différentes :

- une première étude avec un plan complet qui est une approche sûre et fine mais dont l'inconvénient majeur est le temps de réalisation,
- et une deuxième étude avec un plan fractionnaire qui permet de diminuer le nombre d'expériences tout en gardant une bonne précision dans la détermination des résultats.

Compte tenu du nombre important de facteurs susceptibles d'avoir une influence sur un système sans fil, la démarche retenue s'appuiera essentiellement sur des plans fractionnaires.

La méthodologie ainsi proposée peut être reproduite dans un contexte différent, c'est à dire où les paramètres considérés comme essentiels et donc retenus pour les expériences seraient différents.

Plan complet

On appelle plan complet ou méthode cartésienne l'étude qui consiste à fixer tous les facteurs sauf un, pour connaître son effet sur la réponse.

Plusieurs mesures ont été effectuées avec un plan complet pour tester l'influence de quelques paramètres particuliers dans des conditions qualifiées d'optimales. Ce type de mesures permet donc d'évaluer les dégradations engendrées par la variation d'un paramètre particulier.

Cependant, il n'est pas envisageable d'étudier l'influence de chacun des paramètres retenus en utilisant cette méthode car ceci impliquerait un nombre bien trop important d'expériences à réaliser. En effet, prenons à titre d'exemple le cas où l'on cherche à déterminer l'influence de 7 paramètres différents (ce qui, on le verra plus loin sera notre cas), qui peuvent chacun varier sur deux niveaux (deux valeurs possibles pour chacun des paramètres). Aussi pour connaître l'influence de chacun des paramètres, la mise en place de 128 expériences différentes ($= 2^7$ expériences) sera nécessaire. De plus, une analyse significative des résultats de chacune des expériences réalisées ne sera possible que si l'échantillon de données obtenu est suffisamment grand, ce qui signifie

que les expériences doivent être renouvelées un grand nombre de fois. Aussi, la réalisation d'un tel nombre d'expérimentations n'est absolument pas envisageable, même si dans le cas présent le nombre de niveaux considérés est extrêmement restreint.

Plan fractionnaire

On appelle plan fractionnaire, le principe consistant à étudier la réponse en faisant varier tous les facteurs étudiés à chaque essai.

Une méthode mathématique bien connue dans le milieu des statistiques de la qualité, la méthode de Taguchi permet la mise en œuvre de ces plans d'expérience fractionnaires et leurs interprétations. Dans cette méthode, le plan n'est pas choisi au hasard mais répond à la condition d'orthogonalité qui permet d'étudier l'effet d'un facteur indépendamment des autres facteurs. L'utilisation de cette méthode permet ainsi d'obtenir l'influence des différents paramètres et leurs interactions sur les performances générales.

5.2.2 Facteurs/Paramètres

L'étude des variations de performances de la plate-forme passe par la maîtrise de l'ensemble des paramètres pouvant influer sur ces performances. Aussi, une liste non exhaustive des paramètres pouvant intervenir dans le système a été dressée : la distance, le nombre d'obstacles, le nombre d'appareils, le mode d'architecture (ad-hoc ou infrastructure), le type d'émetteur, le type de récepteur, la charge réseau, l'économie d'énergie (PSP[3] ou CAM[4]), la veille de l'émetteur, le type de connexion (UDP TCP), la version des drivers, le lieu, etc...

Cependant, dans un soucis de simplification, certains paramètres ont été fixés. En effet l'influence des drivers utilisés n'a pas été testée et le contexte géographique a, quant à lui, été figé (pas de mobilité des entités).

Ainsi, pour cette étude, les paramètres à deux niveaux présentés dans la figure 5.4 ont été retenus.

Comme indiqué dans le tableau 5.4, lors de la réalisation des expérimentations présentées dans cette section, les paramètres correspondants aux types d'entités émettrices ou réceptrices ont été étudiés sur deux niveaux (PDA1 (type Jornada) ou portable). Cependant, de nouveaux types de matériels ayant été mis à notre disposition par la suite, d'autres campagnes d'expérimentations ont été réalisées afin de compléter cette étude préliminaire par la prise en compte d'une nouvelle génération de PDAs (PDA2, type iPAQ, présenté dans le tableau 5.3). Aussi, bien que les résultats présentés dans ce chapitre portent essentiellement sur la comparaison entre les types d'entités PDA1 et portable, quelques résultats annexes illustrant les performances obtenues avec des entités de type PDA2 (type iPAQ) seront introduits de manière complémentaire.

[3]Power Saving mode
[4]Constantly Awake Mode

	Paramètres	Niveau 1	Niveau 2	Informations
1	Charge réseau (a)	oui	non	charge de 1.6 Mbits/s
2	Distance	1 mètre	5 mètres	taille d'une salle de réunion
3	Type d'émetteur (b)	PDA	portable	pour l'hétérogénéité
4	Type de récepteur	PDA	portable	pour l'hétérogénéité
5	Nombre de composants (c)	3	5	réseau de taille suffisante
6	Nombre d'obstacles (d)	rien	un mur	pour la confidentialité
7	Économie d'énergie (e)	PSP	CAM	PSP niveau 1 sur 5

Remarques :

(a) La charge réseau augmente le nombre de collisions et ainsi dégrade les performances.

(b) Les différents terminaux utilisés n'ont pas les mêmes capacités.

(c) Plus le nombre de composants présents est important, plus les risques de collisions sont grands et plus les performances risque d'être dégradées.

(d) La fréquence radio utilisée (2.4 Ghz) pour les liaisons sans fil en 802.11b est sensible aux obstacles.

(e) Dans le mode CAM, l'adaptateur scrute le réseau en continu sans engendrer de latence. Alors que dans le mode PSP (Power Saving mode), l'adaptateur coupe ses transmissions radio momentanément pour économiser son énergie.

Figure 5.4 *Les différents paramètres retenus*

5.3 Résultats - Analyse

Trois critères ont été retenus pour l'évaluation des performances de la plate-forme : la latence, le taux de perte et le débit. La variabilité de ces trois critères a été illustrée grâce à la mise en œuvre des différents plans d'expériences.

5.3.1 Latence

L'évaluation des latences s'est faite avec deux plans d'expériences identiques séparant les deux modes d'architectures possibles (voir Annexe B). Pour chaque essai, 500 mesures espacées d'un intervalle de temps d'une seconde[5] ont été réalisées.

D'autre part, de manière complémentaire, quelques tests en dehors du plan d'expérience ont été réalisés afin d'explorer des comportements particuliers. Ces premiers tests ont fait apparaître deux types de répartitions (FIG 5.5) différentes suivant le mode d'économie d'énergie :

- Si le mode "économie d'énergie" est activé, il apparaît que les occurrences sont localisées en "pic", espacés d'environ 80 ms, variant selon la distance et le mode d'architecture.
- Si le mode "économie d'énergie" n'est pas activé, la répartition semble de type "exponentiel" avec un léger décalage suivant le mode d'infrastructure ou le type d'appareil.

[5] permet d'obtenir des mesures indépendantes. En effet, une mesure est inférieure à une seconde

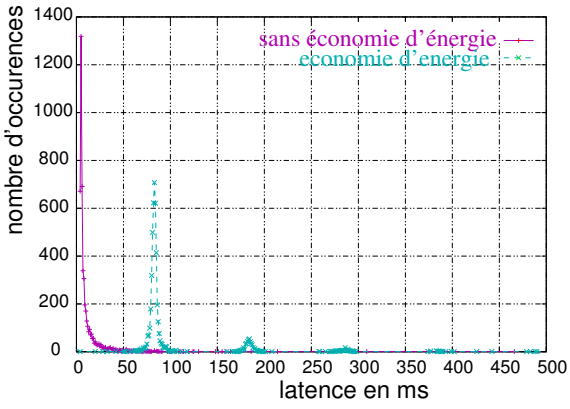


Figure 5.5 *PDA de type Jornada en mode ad-hoc*

Sur cette expérience mettant en jeu plusieurs PDAs de type Jornada, on constate une forte variabilité des latences en fonction par exemple du mode de gestion d'énergie. En mode sans économie d'énergie, la latence est bien distribuée autour de la valeur moyenne et les temporisations des détecteurs de défaillances peuvent se baser sur cette estimation. Par contre, en mode "économie d'énergie", la variabilité est très forte : 1 % de la distribution se situe au delà de 50 fois la valeur moyenne sans économie d'énergie. Le basculement de mode induit alors des ruptures de comportements dont il faut tenir compte.

Suite à la réalisation du plan d'expérience fractionnaire pour le mode ad-hoc, les résultats obtenus (figure 5.6[6]) ont mis en évidence l'influence des facteurs retenus sur le système ainsi que leurs interactions. Cependant, compte tenu des taux d'échec élevés pour les expériences utilisant un ordinateur portable et un PDA de type Jornada en mode d'économie d'énergie, ces résultats restent plutôt de nature qualitative.

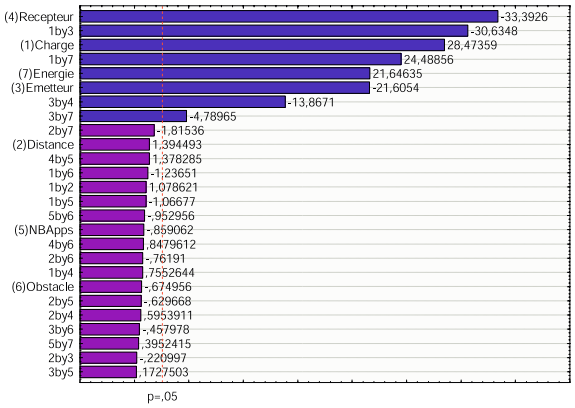
D'après ces résultats, une constatation s'impose : une distance de 1 à 5 mètres, un nombre de composants compris entre 2 et 5 et la présence d'un mur entres les stations ont une influence négligeable sur les latences. Cependant, le type de récepteur, la charge, l'énergie, et le type d'émetteur sont, quant à eux, des paramètres influents. En effet, leurs variations modifient les performances en changeant la moyenne des latences.

Au vue de ces premiers résultats, il serait donc intéressant de mettre en place un nouveau plan d'expérience contenant seulement les paramètres influant afin d'affiner l'analyse.

En ce qui concerne les moyennes des latences, des disparités entre les PDAs de type Jornada, les PDAs de type iPAQ et les ordinateurs portables ont été constatées. Outre l'influence de la charge réseau ou du mode d'économie d'énergie, les ordinateurs portables ont des latences plus faibles (de l'ordre de 3ms) que celles des PDAs (pour les PDAs de type Jornada ces latences sont de l'ordre de 6ms).

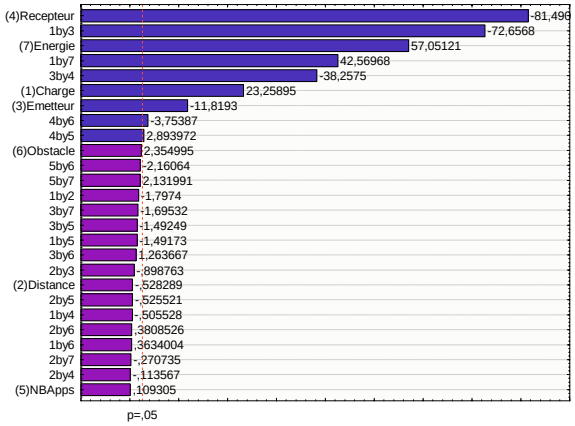
Par ailleurs, le même plan d'expérience fractionnaire a été réalisé pour le mode infrastructure. La figure 5.7, résultant de la même analyse que pour le mode ad-hoc, montre l'influence des paramètres et de leurs interactions sur la moyenne. A la différence du mode ad-hoc, seulement 2 paramètres (énergie et récepteur) et une interaction influencent significativement le système. Ces résultats confirment la meilleure robustesse du mode infrastructure comparé au mode ad-hoc. De plus, l'influence de l'interaction entre les facteurs *récepteur* et *émetteur* montre les incidences de l'hétérogénéité sur les performances du réseau. En effet, les latences ont tendance à être plus

[6]Le diagramme de PARETO est un outil d'analyse quantitatif permettant de représenter l'importance relative de différents paramètres d'un système. Il s'appuie sur la loi du 80/20.



Seuls les facteurs et les interactions dépassant "p=0.5" ont une influence significative. Les interactions sont notées 'XbyY' avec X et Y les numéros des facteurs.

Figure 5.6 *pareto sur la moyenne en mode ad-hoc*



Les résultats obtenus sont très proches de ceux obtenus pour le mode ad-hoc.

Figure 5.7 *pareto sur la moyenne en mode infrastructure*

faibles entre les cartes réseaux de même type.

Une disparité existe encore pour les deux modes d'économie d'énergie. Sans économie d'énergie, sur l'ensemble des tests, les latences s'étalent de 3ms à 30ms[7] avec une moyenne à 7ms.

5.3.2 Taux de perte

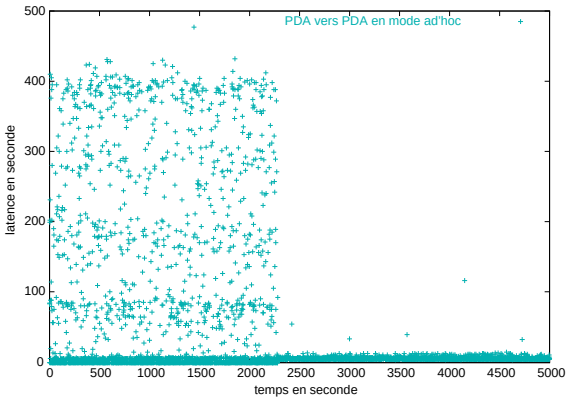
Suite à l'ensemble de des tests effectués, il apparaît que des pertes ne sont relevées que dans certaines configurations. En effet, dans des conditions optimales d'utilisation[8] le taux de perte est inférieur à 1 %. Cependant, en mode ad-hoc, lorsqu'un portable émet vers un PDA en mode économie d'énergie (mode PSP), le taux de perte avoisine les 80 %. Ces erreurs sont probablement

[7] valeur maximum sur 99% des 500 mesures

[8] Dans une même pièce, sans économie d'énergie

dues à une désynchronisation des stations n'ayant pas lieu en mode infrastructure (dans lequel les communications sont centralisées au niveau du point d'accès) puisque aucune perte de données n'a alors été détectée.

L'évaluation du taux de perte en mode ad-hoc à partir d'un test expérimental comprenant 5000 échantillons de mesures, a mis en évidence l'apparition possible de régimes chaotiques. Le changement brutal pouvant ainsi survenir semble encore une fois être dû à un problème de désynchronisation.



Cette expérience se divise en deux phases. Tout d'abord un régime chaotique de durée significative (de l'ordre d'une heure) avec de nombreuses pertes puis un régime stable avec des latences autour de 5ms.

Figure 5.8 *5000 pings espacés d'1 seconde*

Notons également que pour des distances élevées, le récepteur est beaucoup plus sensible aux perturbations telles que : le passage de personnes, les obstacles passagers proches des stations, ceci pouvant même provoquer des échecs de transmissions.

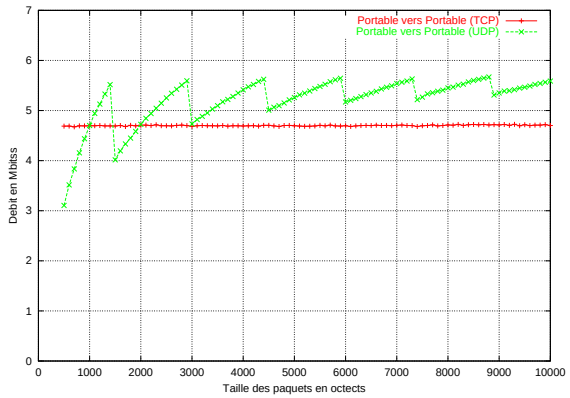
5.3.3 Débits

Les mesures de débit ont été effectuées (via l'utilitaire 'netperf') suivant diverses configurations : mode d'architecture ad-hoc ou mode infrastructure, protocole TCP ou protocole UDP, différents types de terminaux émetteurs, variation de la taille des paquets émis, et variation de la distance entre les terminaux.

Dans une première campagne de tests en mode ad-hoc, dans des conditions normales d'utilisation, les débits obtenus entre deux ordinateurs portables s'échelonnent entre 3 Mbits/s et 5.6 Mbits/s en UDP et de 3 Mbits/s à 4.7 Mbits/s en TCP (FIG 5.9). Ces débits sont conformes à ceux espérés puisque théoriquement 802.11b propose un débit maximal de 11Mbits/s. Cependant, pour les PDAs (FIG 5.10), le débit relevé en mode UDP est de 1.82 Mbits/s au maximum pour une taille de paquets de 8 ko.

En comparaison des résultats illustrés dans la figure 5.10, les expériences réalisées par la suite sur la nouvelle génération de PDAs (type iPAQ) ont montré de bien meilleures performances. En effet, sur ce type de PDAs, le protocole UDP permet d'atteindre des débits moyens de 5.52 Mb/sec là où les PDAs de type Jornada ne dépassaient pas les 1.82 Mb/sec.

Pour les courtes et moyennes distances le débit croit avec l'augmentation de la taille du paquet car le sur-débit devient de plus en plus négligeable par rapport à la charge utile lors des trans-



Pour la courbe en mode UDP, des décrochages sont visibles tous les 1470 octets dûs au MTU (Maximum Transmission Unit). Le coût de l'utilisation de TCP est bien mis en évidence. En effet, le débit chute de 0.5 Mbits/s par rapport à UDP.

Figure 5.9 *Débits moyen pour les ordinateurs portables*

Mode	Taille des paquets	Débit UDP	Débit TCP
Ad-hoc	1000 octets	1.60 Mbits/s	1.28 Mbits/s
Ad-hoc	4000 octets	1.8 Mbits/s	1.54 Mbits/s
Ad-hoc	8000 octets	1.82 Mbits/s	1.65 Mbits/s
Infrastructure	1000 octets	1.56 Mbits/s	1.26 Mbits/s
Infrastructure	4000 octets	1.71 Mbits/s	1.56 Mbits/s
Infrastructure	8000 octets	1.74 Mbits/s	1.69 Mbits/s

Pour le mode TCP ou UDP, l'augmentation de la taille des paquets engendre l'augmentation du débit. Cependant, pour des distances plus élevées, les débits chutent lorsque l'on utilise des paquets de grandes tailles.

Figure 5.10 *Débits moyen pour les PDAs de type Jornada*

missions des gros paquets. Cependant, pour les distances limites du handoff[9] le débit décroît avec l'augmentation de la taille du paquet car les paquets de taille importante sont plus sensibles aux erreurs.

Pour les distances relativement élevées, l'utilisation de paquets de petite taille, moins sensibles aux erreurs de transmissions, est recommandée. En effet, l'augmentation de la distance induit une élévation du taux d'erreur. De plus, le débit est moins stable pour les paquets de grandes tailles.

L'augmentation de la distance dégrade considérablement les débits. En effet, ces dégradations ont pu être constatées lors de la mise en œuvre d'une expérience avec deux portables séparés d'une dizaine de mètres par une dalle en béton armé (entre deux étages). Ce phénomène est expliqué par l'utilisation pour les communications d'une gamme de fréquences sensible aux perturbations. En effet, les fréquences autour de 2.4 GHz traversent difficilement les obstacles, de plus, celles ci

[9] distance limite de raccordement

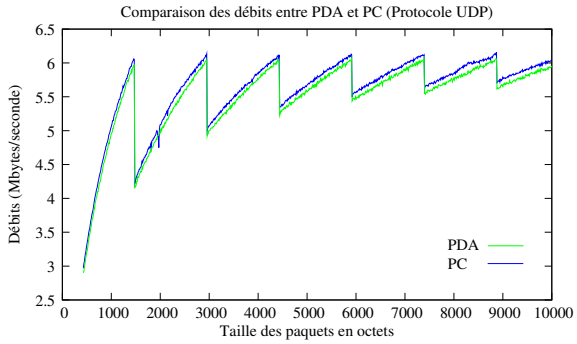


Figure 5.11 *Débits observés avec le protocole UDP pour un PDA (iPAQ) et un PC portable*

L'allure générale des courbes est très proche, cependant la courbe de débit pour le PDA est moins régulière et parsemée de décrochages. Cela peut s'expliquer par une plus faible robustesse de l'architecture par rapport aux PCs. Ces ruptures soudaines peuvent être dues à l'utilisation des ressources (processeur, mémoire) par le systeme d'exploitation. Les différences observées entre PDA et PC portable représentent, un écart variant autour de 0.10Mb/s pour le protocole UDP et autour de 0.8Mb/sec pour le protocole TCP.

étant dans le domaine public, sont très utilisées et donc plus "polluées".

5.4 Conclusion

Cette étude a permis d'obtenir de premiers résultats sur les performances du protocole 802.11b sur une plate-forme hétérogène et de mettre en évidence les principaux facteurs influençant le comportement du système. Des plans d'expériences plus précis permettraient d'affiner les résultats concernant l'influence des différents paramètres, mais de raffiner également les modélisations mathématiques effectuées par la suite (voir chapitre 10).

De manière plus générale, les résultats obtenus paraissent essentiels à la mise en place d'applications dans un environnement ambiant hétérogène. La connaissance des performances du réseau permet ainsi d'ajuster un certain nombre de paramètres assurant ainsi une meilleure qualité de service (voir chapitre 10). En effet, la perte de connectivité d'une station pouvant fortement perturber le fonctionnement d'une application, des temporisations ajustées convenablement peuvent permettre une détection rapide de la présence ou de l'absence d'une station dans le système et ainsi la mise en place rapide de procédures adéquates.

Bilan général 6

Cette première partie a été consacrée à la définition et à la caractérisation de l'environnement considéré au sein de notre problématique.
De manière globale, le contexte environnemental dans lequel nous nous plaçons met en œuvre divers "objets communicants" susceptibles d'échanger de l'information grâce à l'utilisation d'un ou de plusieurs protocoles de communication sans fil.
Aussi, les contraintes de l'environnement visé sont de plusieurs types :

- L'utilisation d'un réseau sans fil implique une certaine restriction des performances escomptées en terme de latences et de débit notamment. De plus, un environnement basé sur un protocole de communication sans fil affiche, une forte sensibilité aux perturbations extérieures due au mode de propagation du signal par les ondes. Cette sensibilité se traduit essentiellement par une grande variabilité au niveau des temps de communication. Enfin, dans un tel environnement, la mobilité des entités, même très faible, peut avoir un fort impact sur le système (déconnexions intempestives dues à des sorties de zone de portée) et doit donc être prise en compte.
- Le choix d'une architecture évolutive de type ad-hoc n'est pas sans contrainte. La topologie dynamique de l'environnement ainsi que l'absence de point de centralisation (machines stables) font partie des contraintes de notre environnement.
- Les entités composant cet environnement appartiennent au vaste ensemble nommé "objets communicants" et peuvent donc avoir des capacités (mémoire, CPU, batterie, etc...) diverses et variées. Il est donc impératif de tenir compte et de s'adapter à cette hétérogénéité des entités.

Dans ce contexte fortement dynamique, l'approche expérimentale proposée a permis de mettre en évidence le comportement général de l'environnement considéré. Étant relativement générique, cette méthodologie d'analyse expérimentale peut être reproduite dans un contexte où les conditions environnementales seraient quelques peu différentes de celles présentées dans ce document.
Cependant, cette approche mériterait d'être complétée notamment au niveau du passage à l'échelle,

c'est à dire par la prise en compte dans le système, d'un nombre bien plus important d'entités (plusieurs dizaines de machines).

Par ailleurs, suite à cette caractérisation de l'environnement, la principale question que l'on se pose est de savoir comment les entités vont parvenir à collaborer au sein d'un environnement pouvant présenté une forte instabilité générale. Autrement dit, face aux diverses spécificités environnementales décrites tout au long de cette partie, comment assurer la collaboration cohérente entre les entités appartement au système.

II

Algorithmique Distribuée de Décision

Les appareils connectés en communauté déclenchent des événements : connexion et/ou déconnexion d'une entité, apparition et/ou disparition d'un service distribué, etc...
La communauté ainsi constituée doit pouvoir survivre à ces événements. Cette partie sera donc consacrée à la description des outils algorithmiques nécessaires à l'évolution de cette communauté malgré les aléas générés par la forte variabilité à laquelle est soumis l'environnement.

Environnement instable 7

7.1 Introduction

Comme nous l'avons précédemment décrit dans le chapitre introductif de ce document, les applications visées dans le cadre du projet industriel Sidrah, concernent essentiellement la mise en relation spontanée d'un nombre relativement restreint d'entités hétérogènes et peut typiquement être illustré par un scénario de type réunion de travail.
Dans ce cadre, une part importante de la problématique générale réside dans la gestion de la "résilience". A savoir, comment assurer la cohésion d'un groupe d'entités évoluant en collaboration au sein d'un environnement totalement distribué et présentant une forte instabilité générale due essentiellement à l'utilisation d'un protocole de communication sans fil (diverses perturbations) mais aussi au fait que les entités ne sont pas statiques.
En effet, dans une architecture où certaines entités sont "privilégiées", c'est à dire qu'elles ne sont pas soumises aux mêmes contraintes que la majorité d'entre elles, l'approche peut être envisagée de telle sorte que tous les problèmes critiques reposent sur ces entités dites "stables". Cependant, dans le contexte étudié ici, où aucune relation de type client/serveur ne peut être considérée, toutes les entités doivent jouer le même rôle et sont donc toutes confrontées aux mêmes conditions d'instabilité ambiante.

Le maintien de la cohérence d'un groupe d'entités totalement distribué passe par la mise en place d'outils permettant la prise de décisions au sein du groupe. L'objectif fondamental est donc de fournir aux entités mises en jeu le moyen de coordonner leurs actions ou de s'accorder sur une ou plusieurs valeurs partagées malgré les aléas de l'environnement considéré.
En algorithmique distribuée tolérante aux pannes, différents problèmes d'accords (consensus, élections, exclusions mutuelles, etc...) concourent à atteindre cet objectif.

Comme nous l'avons décrit dans la partie précédente (partie I), l'environnement considéré possède un certain nombres de particularités. Il est donc nécessaire de formuler les différentes

contraintes caractérisant l'environnement distribué étudié. Aussi voyons désormais quelles hypothèses sont à effectuer sur le système distribué établi tant au niveau du modèle de communication à considérer qu'au niveau du modèle de défaillances.

7.2 Les contraintes environnementales

Contraintes au niveau des communications :

Aux vues des résultats obtenus lors des différentes expérimentations réalisées sur le réseau de communication sans fil de l'environnement considéré (voir partie I, chapitre 5), on retient en particulier la forte variabilité à laquelle sont soumises les communications. Cette forte variabilité des latences de communication s'explique essentiellement par le principe d'échange d'informations s'effectuant dans un réseau sans fil par propagation d'ondes radio et étant de ce fait fortement sensible aux perturbations extérieures. Bien qu'un certain nombre de facteurs influents aient été identifiés, la diversité des sources de perturbations extérieures ne permet pas de quantifier de manière déterministe la variabilité des latences de communication à laquelle peut être soumis le système. Aussi, ne pouvant définir de borne maximale sur le temps de communications, ceux-ci pouvant être arbitrairement longs, le système réparti considéré peut être qualifié de système asynchrone. Par ailleurs, l'asynchronisme du système provient également de l'utilisation au niveau matériel d'entités hétérogènes. En effet, le réseau sans fil que l'on considère est composé de diverses machines aux capacités diverses en calcul, en mémoire, en énergie, en stockage de données, en fiabilité, etc... Il faut noter que les vitesses des processeurs sont différentes, et que de plus elles s'adaptent aux différents modes d'économie d'énergie susceptibles d'être activés par les diverses entités.

D'autre part, il n'est pas envisageable dans ce contexte d'établir des hypothèses sur la fiabilité des communications. En effet, dans un environnement perturbé, non seulement les communications peuvent être qualifiées d'arbitrairement longues, mais il est également possible que certains messages soient perdus. Le système de communication considéré est donc finalement défini comme un **système asynchrone et non fiable**.

Contraintes vis à vis des défaillances :

On s'intéresse ici aux problèmes liés aux déconnexions intempestives (involontaires) des entités appartenant à l'environnement considéré. En effet, comme nous venons de le voir dans le paragraphe précédent, l'utilisation d'un réseau sans fil fragilise notre environnement dans le sens où sa forte sensibilité vis à vis des perturbations extérieures se répercute à différents niveaux. A cela viennent s'ajouter les problèmes liés à la mobilité des entités composant le système. En effet, bien que la mobilité considérée ne soit que relativement restreinte, il faut cependant considérer les sorties de zones de couverture pouvant advenir. Une entité sortant de la zone de portée des autres entités est considérée comme étant non atteignable, on dit qu'elle est déconnectée momentanément. Une entité momentanément isolée, réintégrera naturellement la communauté lors de sa reconnexion sans perte d'état. Comme l'on suppose que dans notre environnement, toutes les entités sont atteignables directement les unes aux autres, toute la difficulté vient du fait qu'une entité peut sembler déconnectée à une autre entité mais restée atteignable par l'ensemble de la communauté.

Par ailleurs, à l'instabilité introduite par la dynamicité de l'architecture, s'ajoute le fait que les entités mises en jeu peuvent éventuellement être soumises à des défaillances définitives de type pannes franches. Du point de vue de la terminologie, la littérature en algorithmique distribuée fait généralement référence à l'aspect "correct" ou "incorrect" des entités. Une entité est couramment qualifiée de "correcte" si elle ne subit aucune défaillance, et respectivement "incorrecte" si elle est susceptible de subir une défaillance définitive. Nous verrons par la suite que, dans notre contexte, une distinction sera faite entre des entités dites "instables", à savoir des entités susceptibles de subir des défaillances momentannées, et des entités "incorrectes" c'est à dire pouvant défaillir définitivement.

7.3 Choix algorithmiques

Ainsi, dans un environnement réparti affichant ce type de contraintes, l'évolution cohérente des entités du système nécessite la mise en œuvre d'algorithmes répartis permettant la résolution de problèmes d'accord. Parmi les différents problèmes d'accord existants, la littérature a porté un vif intérêt au problème de la résolution de consensus en environnement distribué. Ceci se justifie par le fait que l'algorithme de consensus apparaît comme le dénominateur commun entre tous ces problèmes d'accord [44], les différents problèmes d'accord pouvant d'une manière ou d'une autre être ramenés à un problème de consensus. En effet, les transformations des divers problèmes d'accord en un problème équivalent à celui du consensus sont présentées dans [56, 19, 45].

Aussi, nous avons dans un premier temps restreint notre champ d'investigation et focaliser notre approche algorithmique sur le problème fondamental de résolution du consensus en environnement fortement variable.

Le problème du consensus 8

S'inscrivant dans l'ensemble référencé des "problèmes d'accord" en algorithmique distribuée tolérante aux défaillances, le problème du consensus a été très largement étudié dans la littérature. Le principe commun de ces problèmes de décision [104, 74] réside dans le fait que toutes les entités d'un ensemble réparti puissent s'accorder sur une valeur commune et ce de façon irréversible. L'intérêt porté au paradigme du consensus s'explique par le fait que de nombreux problèmes d'accord peuvent être vus comme des extensions ou des déclinaisons du problème "de base" du consensus.

8.1 Le principe général

Permettant la prise de décision répartie, le consensus est un problème d'accord, offrant la possibilité à un ensemble de processus de s'accorder sur une valeur commune choisie parmi les valeurs initialement proposées.

Le principe d'un consensus peut être défini de manière non formelle, de la façon suivante :

- Chaque entité du système possède une valeur initiale, qu'elle propose comme valeur de décision commune à toutes les autres entités.
- L'objectif du consensus est de permettre aux entités "correctes" de l'ensemble, de s'accorder sur une valeur commune prise parmi les valeurs initialement proposées par chacune des entités.
- L'action de décision est qualifiée d'irrévocable : lorsqu'une entité décide d'une valeur lors du consensus, elle ne peut plus en changer.

Ainsi, dans la littérature, le problème du consensus est défini pour un ensemble P de processus corrects. Chaque processus $p_i \in P$ propose alors la valeur v_i qu'il possède initialement. L'algorithme ne se termine que lorsque tous les processus participant au consensus (ou plus précisément

tous les processus corrects) se sont accordés sur une valeur v ; v étant une des valeurs initialement proposées par un des processus ($\exists\, i\ tel\ que\ v_i = v$).

Les propriétés :

Trois propriétés permettent de donner la spécification d'un consensus [19] :

- **Propriété de Terminaison :** tous les processus corrects doivent finir par décider.
- **Propriété d'Accord :** si 2 processus corrects décident, ils décident d'une même valeur.
- **Propriété d'Intégrité :** si 1 processus correct décide, sa valeur de décision est une des valeurs initiales.

L'**accord** et l'**intégrité** représentent les propriétés de **sûreté**, et la **terminaison** quant à elle, est une propriété de **vivacité**.

Le consensus et ses extensions :

Que se passe-t-il si l'on modifie l'une des propriétés (Terminaison, Accord ou Intégrité) du consensus ?
Cette dégradation des spécifications peut s'avérer critique pour certaines applications, mais les conditions d'accord, d'intégrité et de terminaison, peuvent rester pertinentes pour d'autres types d'applications [29].

Les propriétés d'accord et d'intégrité sont modifiées pour le consensus uniforme, puisque tous les processus qui décident (corrects ou incorrects) le font sur une même valeur qui doit être l'une des valeurs initiales. Ainsi, contrairement au consensus, le consensus uniforme impose des contraintes aux processus définis comme étant incorrects. De ce fait, il permet d'éviter les effets que peut avoir le comportement incohérent d'un processus incorrect.

Le K-accord propose une généralisation du problème du consensus. En effet, celui-ci n'impose pas que la valeur de décision des processus soit unique, mais simplement que la valeur décidée par un processus appartienne à un ensemble d'au plus k éléments.

Consensus	Tous les processus "corrects" décident sur la même valeur.
Consensus Uniforme	Tous les processus "qui décident" décident sur la même valeur.
K-Accord	Les valeurs décidées appartiennent à un ensemble d'au plus K valeurs.
Accord Approximatif	Les valeurs décidées diffèrent d'au plus ϵ.
Validation Atomique	La décision est forcée à $abort$ si 1 des valeurs initiales est $abort$, à $commit$ si toutes les valeurs initiales sont à $commit$ et qu'il n'y a pas de pannes.

Le consensus et ses extensions [29]

8.2 L'impossibilité de Fisher, Lynch & Paterson

La spécification présentée précédemment peut être envisagée dans les deux modèles de systèmes principaux : les systèmes synchrones et les systèmes asynchrones. Cependant, dans un système asynchrone, d'une part les spécifications du consensus, et d'autre part les propriétés attribuées à ce type de systèmes, nous confrontent au résultat d'impossibilité défini par Fisher, Lynch et Paterson [40]. En effet, ce résultat montre qu'un consensus n'est pas réalisable de façon déterministe dans un système asynchrone soumis à des défaillances franches de processus, même si le système n'est soumis qu'à une seule panne et que les messages sont acheminés de façon fiable. De manière intuitive, ce résultat d'impossibilité est justifié par le fait qu'il est impossible de distinguer un processus lent d'un processus en panne, lorsque l'on ignore les vitesses d'exécution des processus et les délais de transmission des messages.

Ainsi, une des principales difficultés rencontrées dans le domaine de l'algorithmique distribuée tolérante aux pannes concerne ce résultat d'impossibilité dans un environnement asynchrone puisqu'il empêche la résolution de nombreux problèmes d'accord dans de tels environnements.

Cependant, différentes alternatives permettent de contourner cette impossibilité. En effet, de nombreuses recherches ont été réalisées dans le domaine de l'algorithmique distribuée, afin de permettre la résolution du problème du consensus dans un système asynchrone dans lequel les processus peuvent être soumis à des défaillances franches de processus.
Diverses approches ont ainsi été considérées pour palier à ce problème fondamental : algorithmes probabilistes [17], algorithmes auto-stabilisants [107], la définition de problèmes affaiblis et leurs solutions [32, 16], etc...
D'autre part, la résolution du consensus est également possible dans un environnement partiellement synchrone, tel que le système défini dans [34, 31]. Ainsi, suivant les hypothèses suivantes : (i) une majorité de processus ne tombent jamais en panne, (ii) on ne connaît pas les délais de transmission des messages, mais (iii) on sait qu'il existe un instant à partir duquel les délais de transmission et les vitesses des processus seront inférieurs à certaines bornes, le problème du consensus peut être résolu [29].
Par ailleurs, de nombreux travaux ont été menés autour de la proposition de résolution de ce problème fondamental faite par Chandra et Toueg [19], à savoir d'enrichir l'environnement par un mécanisme de détection de défaillances non fiable.

8.3 Conclusion

Dans ce cadre théorique de l'algorithmique distribuée, et face aux spécificités environnementales auxquelles nous sommes confrontés (partie I), le choix de l'approche algorithmique à retenir n'est pas trivial. Cependant, parmi les différentes stratégies proposées afin de palier au résultat d'impossibilité de résolution du consensus présenté dans [40], l'utilisation d'un mécanisme de détection de défaillances [19] paraît la mieux adaptée à notre contexte.

En effet, un tel mécanisme permet d'une part de garantir la correction des différents services de décision répartie, et d'autre part, son implantation semble assez flexible pour permettre une mise au point nécessaire à l'optimisation des performances de l'intergiciel.

Par ailleurs, la mise en place d'un module de détections de défaillances permet de concentrer toute l'analyse de la variabilité de l'environnement dans un seul composant (voir chapitre 10).

Aussi, comme nous le verrons dans la suite de ce document, l'option retenue pour la mise en œuvre d'un consensus dans notre environnement, a consisté à l'implantation d'un mécanisme de détections de défaillances tel que proposé initialement dans [19].

Les détecteurs de défaillances 9

Chandra et Toueg [19] ont d'une part défini formellement un détecteur de défaillances, et d'autre part spécifié, en fonction des propriétés attribuées aux détecteurs de défaillances, des classes correspondantes. Après un rapide rappel des propriétés attribuées aux détecteurs de défaillances suivant leur classe d'appartenance 9.1, nous présenterons en détail l'architecture retenue dans notre contexte 9.2. Enfin, une partie plus technique de ce chapitre 9.3, sera consacrée aux différents types d'implémentations possibles des détecteurs de défaillances ainsi que leurs avantages ou inconvénients vis à vis de notre environnement.

9.1 Principe théorique

9.1.1 Description

De manière générale, le principe de fonctionnement d'un détecteur de défaillances est de fournir, à un instant donné, à un processus particulier, une liste plus ou moins correcte des entités suspectées de défaillance.

Pour ce faire, chacune des entités e_i du système considéré est dotée de son propre module de détection de défaillances. Ainsi, pour une entité particulière, ce module permet, à intervalles de temps réguliers ou à la demande, l'obtention de la liste indiquant quelles sont les entités distantes suspectées d'être éventuellement défaillantes.

Cependant, les informations fournies par le détecteur de défaillances local ne dénotent pas forcément de l'état réel du système. Le détecteur de défaillances se contente d'émettre des suspicions. Il peut donc à tout moment suspecter à tort une entité correcte (c'est à dire toujours présente dans le réseau), ou bien ne pas suspecter une entité qui elle est réellement défaillante.

On suppose néanmoins que le détecteur de défaillances finira toujours par suspecter une entité défaillante, et qu'une entité présente finira par ne plus être suspectée par les détecteurs de défaillances des autres entités présentes.

Pour mieux comprendre son mécanisme de fonctionnement, considérons les propriétés pouvant être attribuées au détecteur de défaillances.

9.1.2 Propriétés

De part la présentation même des détecteurs de défaillances, deux propriétés peuvent être définies :

- La propriété d'**exactitude** qui signifie qu'un processus correct ne doit pas être suspecté.
- La propriété de **complétude** qui correspond au fait qu'un processus incorrect doit être suspecté.

Pour qu'un détecteur de défaillances soit pertinent, sa spécification doit englober ces deux propriétés, ou plus précisément un compromis entre ces deux propriétés. En effet, une des caractéristiques principales d'un détecteur de défaillances est qu'une partie de l'information qu'il fournit peut être erronée. De ce fait, une partie de son jugement est quant à elle correcte. Il faut donc qu'à la fois certains processus corrects ne soient pas suspectés (propriété d'exactitude) et que certains processus incorrects le soient (propriété de complétude).

Aussi, pour plus de précision, ces deux propriétés peuvent se décliner en différentes sous propriétés :

- La propriété de **complétude** :
  - **Complétude forte** : Tout processus incorrect finira par être suspecté par tous les processus corrects.
  - **Complétude faible** : Tout processus incorrect finira par être suspecté par au moins un processus correct.

- La propriété d'**exactitude** :
  - **Exactitude forte** : Aucun processus correct ne sera suspecté avant de tomber en panne.
  - **Exactitude faible** : Il existe au moins un processus correct qui n'est jamais suspecté.
  - **Exactitude ultime forte** : Après un certain temps t, tout processus correct ne sera suspecté par aucun processus correct.
  - **Exactitude ultime faible** : Après un certain temps t, il existe au moins un processus correct qui ne sera jamais plus suspecté par aucun processus correct.

9.1.3 Qualité de l'information fournie : Classes de détecteurs de défaillances

Des différentes propriétés présentées ci-dessus, toutes dérivées des propriétés d'**exactitude** et de **complétude**, plusieurs classes de détecteurs de défaillances peuvent être déclinées [19].

Dans [29] quatre classes ont été retenues :

1. La classe des détecteurs de défaillances parfaits, ou classe $\mathcal{P}$, qui assure la complétude forte et l'exactitude forte.

Ainsi, dans cette classe, toutes les défaillances sont détectées de manière exacte.

2. La classe qui respecte les propriétés de complétude forte et d'exactitude faible est la classe $\mathcal{S}$, appelée aussi classe des détecteurs de défaillances forts.
   Dans cette classe, la reformulation des spécifications signifie que tous les processus défaillants finiront par être suspectés et que certains processus corrects ne seront quant à eux jamais suspectés.

   La définition même de ces deux classes de détecteurs de défaillances implique une hiérarchie entre elles : intuitivement, on a $\mathcal{P} \subset \mathcal{S}$.
   Ceci est d'ailleurs démontré dans [19], qui justifie de plus que ces classes ont une capacité de résolution des problèmes d'accord différente. En effet, la classe $\mathcal{S}$ permet de résoudre le problème du consensus dans un environnement asynchrone quelque soit le nombre de processus pouvant être défaillants. Quant à la classe $\mathcal{P}$, elle permet de résoudre les consensus dans les mêmes conditions que $\mathcal{S}$, mais aussi de résoudre d'autres types d'accords (par exemple, la diffusion fiable terminante) non résolus par $\mathcal{S}$.

3. La combinaison de la propriété de complétude forte et de celle d'exactitude ultime forte, forme la classe $\diamond\mathcal{P}$ (classe des détecteurs de défaillances ultimement parfaits), caractérisée par le fait qu'après un certain temps, tous les processus incorrects seront détectés comme tels de manière exacte.

   Là encore, une relation d'ordre sur ces ensembles est à notée : $\mathcal{P} \subset \diamond\mathcal{P}$.

4. Quant à l'association des propriétés de complétude forte et d'exactitude ultime faible, elle est représentée par la classe des détecteurs de défaillances ultimement forts ($\diamond\mathcal{S}$).
   Sa spécification peut être décrite par le fait que tous les processus défaillants seront suspectés et qu'au bout d'un certain temps, certains processus corrects ne seront plus suspectés. Les détecteurs de défaillances appartenant à cette classe permettent la résolution des consensus dans un système asynchrone tolérant aux pannes, et ce lorsque les défaillances sont susceptibles de toucher moins de la moitié des processus du système [19].

De plus, on peut citer aussi la classe de détecteurs de défaillances $\diamond\mathcal{W}$ (Eventually weak) qui assure le respect des propriétés de complétude faible et d'exactitude ultime faible. Dans [18], cette classe est caractérisée afin d'englober les détecteurs de défaillances les plus faibles (minimums) permettant la résolution du problème du consensus dans un système asynchrone ayant une majorité de processus corrects.

Or, d'après [19], la classe $\diamond\mathcal{W}$ et la classe $\diamond\mathcal{S}$ sont équivalentes. Aussi, $\diamond\mathcal{S}$ représente la classe regroupant les détecteurs de défaillances minimaux nécessaires à la résolution du problème du consensus dans un environnement asynchrone.
Finalement, concernant les définitions de classes ci-dessus, on a les inclusions d'ensemble suivantes :

$$\mathcal{P} \subset \mathcal{S}, \quad \diamond\mathcal{P} \subset \diamond\mathcal{S}, \quad \mathcal{S} \subset \diamond\mathcal{S}, \quad \mathcal{P} \subset \diamond\mathcal{P}.$$

Ainsi, en résumé, on retiendra que suivant les propriétés qu'ils vérifient, les détecteurs de défaillances appartiennent à une certaine classe. Le choix de la classe de détecteurs de défaillances à utiliser doit donc être effectué en fonction des propriétes nécessaires, mais il est préférable que les spécifications de la classe retenue soient minimales. Dans un soucis de simplification, on s'est attaché par la suite à seulement garantir les propriétés associées à la classe minimale ($\diamond\mathcal{S}$) permettant la résolution du consensus dans un environnement asynchrone dans lequel les entités mises en jeu sont susceptibles de subir des défaillances.

9.2 Spécification des interfaces des détecteurs de défaillances

9.2.1 Architecture logicielle

D'un point de vue architectural, le modèle retenu impose de mettre en place un module permettant à la fois la détection des défaillances et la réalisation d'un consensus. En effet, comme décrit précédemment, étant données les caractéristiques particulières de l'environnement considéré, la résolution du consensus n'est envisageable que grâce à l'existence d'un mécanisme de détections de défaillances. Cependant, afin de préserver la flexibilité de l'architecture, nous avons choisi de différencier le module "Détections de défaillances" du module "Service de consensus". En effet, ceci permet d'envisager d'une part l'utilisation du module de détections de défaillances indépendamment du module de consensus, et d'autre part, la mise en place ultérieure d'autres modules pouvant éventuellement interagir avec ces deux premiers modules.

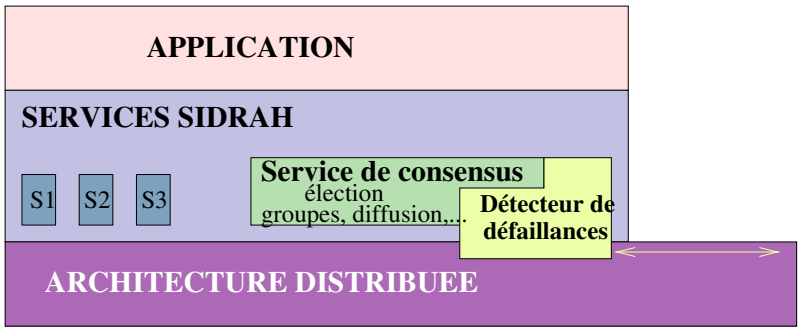


Figure 9.1 *Infrastructure d'un site transmise à l'initialisation via une capacité.*

Ces deux modules, le module de détections de défaillances ainsi que le module de service de consensus, font partie intégrante de l'infrastructure (voir figure 9.1) proposée et sont fournis à l'initialisation à toutes les entités du groupe.

Le principe de fonctionnement du module de détections des défaillances est qu'à tout instant, il peut être utilisé soit par le module "Service de consensus" soit directement par l'applicatif. Aussi, lorsque la réalisation d'un consensus est nécessaire au niveau applicatif, le module "Service de consensus" est appelé, lequel interroge à son tour le module "Détections de défaillances". Ce dernier informe donc régulièrement le service de consensus de sa vision vis à vis des autres participants, à savoir s'il détecte ou non la présence de chacun des autres membres. Bien sûr, cette

information concernant la présence ou non des autres membres n'est qu'une information locale, et peut de ce fait s'avérer plus ou moins exacte. Par ailleurs, le niveau applicatif peut aussi bénéficier directement des pronostics donnés par le mécanisme local de détections des défaillances.

Ainsi, le module de détections des défaillances peut être vu comme une boite noire ayant une interface avec la couche supérieure, et une interface avec le service de résolution de consensus. Cette boite noire fournit, pour une liste de membres donnée, la liste des suspects. Ainsi, pour chaque membre de la liste fournie au détecteur de défaillances local, une information sur sa présence ou non est obtenue en retour. Cependant, on rappelle que cette information n'est qu'une vue locale du détecteur de défaillances, elle peut donc être erronée, à savoir qu'une machine présente peut être suspectée et inversement.

Par ailleurs, il est tout à fait envisageable de raffiner l'information fournie par le détecteur de défaillances en enrichissant ses réponses. Ainsi, au lieu de fournir simplement une liste de suspicions, une notion de qualité pourra être introduite en donnant dans la réponse, un degré de suspicion associé à chacun des membres suspectés.

9.2.2 Architecture générique des Détecteurs de Défaillances

Le principe de fonctionnement d'un détecteur de défaillances peut être divisé en deux parties : d'une part il joue le rôle de collecteur d'informations, et d'autre part il diffuse de l'information. Aussi, l'architecture interne du détecteur de défaillances est composée d'un module d'exportation (voir figure 9.2) et d'un module d'importation (voir figure 9.3).

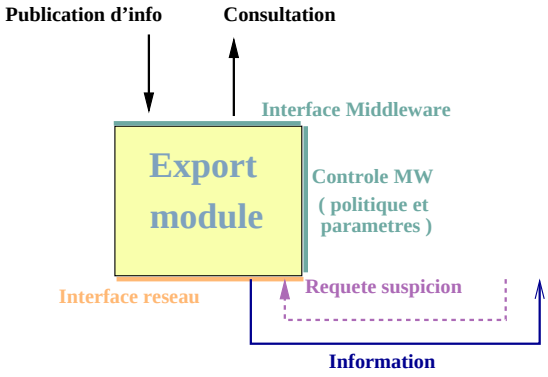


Figure 9.2 *Interfaces du module d'exportation des informations*

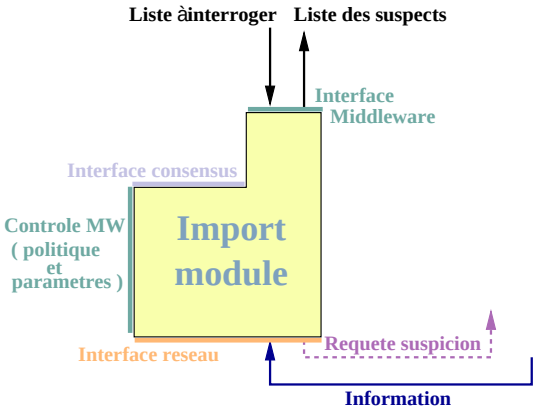


Figure 9.3 *Interfaces du module d'importation des informations*

Ainsi, grâce à son module d'exportation de son détecteur de défaillances, une entité du groupe peut renseigner les autres membres sur son état, soit à la demande, soit par anticipation. Quant au module d'importation du détecteur de défaillances, il implémente un modèle de suspicion de défaillances.

Comme le montre la figure 9.4, il y a interaction entre le module d'importation du détecteur de défaillances d'un site A et le module d'exportation du détecteur de défaillances d'un site B via le contrôle de l'intergiciel. En particulier, il peut mettre à disposition des informations concernant

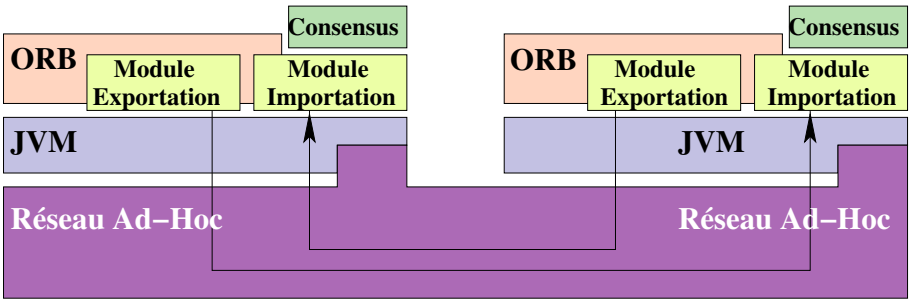


Figure 9.4 *Infrastructure globale : accord sur canaux de communication et formats d"échanges*

son état et fournir des informations de nature applicative telles que la disponibilité d'un service.

9.3 Principe de mise en œuvre

L'implémentation des détecteurs de défaillances peut être réalisée grâce à différentes techniques (heartbeat, requête, ...). En effet, d'un point de vue général, deux grands principes peuvent être envisagés : chacun des modules d'exportation des détecteurs de défaillances informe de sa présence les autres participants soit régulièrement, soit à la demande.

Modèle "heartbeat" (ou modèle "pulsation") : Anticipation des demandes

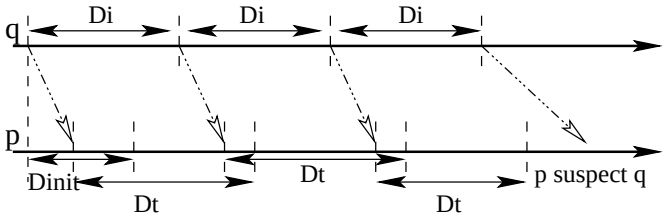
Le principe de cette technique (voir figure 9.3) est basé sur le fait que chacun des modules d'exportation de tous les détecteurs de défaillances envoie périodiquement des messages de présence à tous les détecteurs de défaillances des autres participants. Cette diffusion s'effectue par *broadcast* qui est l'opération de base dans les réseaux sans fil. Ainsi, à la réception d'un message d'une entité distante $e2$, le module d'importation du détecteur associé à l'entité $e1$ utilise sa fonction d'estimation de suspicion, ce qui revient à armer une temporisation dans le cas le plus simple. Ce mécanisme est renouvelé à chaque réception d'un message de $e2$. Par contre, si aucun message de $e2$ n'a été reçu après expiration de la temporisation fixée préalablement, $e2$ est ajouté à la liste des membres suspectés par l'entité $e1$. Elle sera retirée de cette liste s'il s'avère qu'un message de $e2$ est finalement reçu par l'entité $e1$.

Le module d'exportation du détecteur de défaillances associé à l'entité $e1$ applique cette même technique vis à vis de tous les autres membres du groupe. De plus, les détecteurs de défaillances de chacun des membres appliquent ce même mécanisme.

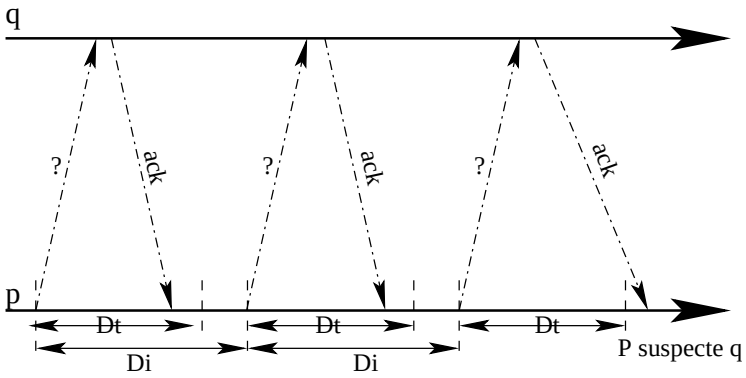
Aussi, telle que définie précédemment, cette stratégie de détection des défaillances est influencée par trois paramètres [98] : D_{init}, le délai initial, D_i, l'intervalle de temps entre les émissions successives de messages, et D_t, la durée de la temporisation. Pour davantage de détails, une analyse des performances de ce modèle d'implémentation des détecteurs de défaillances a été réalisée et proposée dans [38].

Modèle "Requête" (ou modèle "Interrogation") :

Contrairement à l'implémentation précédente où la mise à jour des suspicions est réalisée de façon passive (attente de messages en provenance des sites distants), ce modèle d'implémentation



peut être qualifié d'actif dans le sens où c'est l'entité $e1$ qui interroge, lorsque cela est nécessaire, les autres membres du groupe sur leur présence. Ce mécanisme est illustré dans la figure 9.3.



Avec cette technique, le détecteur de défaillances est déterminé par seulement deux paramètres [98] : D_i, la période d'interrogation, et D_t, la durée d'une temporisation. Il est cependant important de remarquer que ce second mécanisme nécessite davantage de transmissions de messages, et engendre donc plus de trafic, que le premier mécanisme (technique "heartbeat").

Solutions mixtes :

D'autres techniques d'implémentation bien plus judicieuses (en terme de nombre de messages nécessaire) sont proposées dans la littérature. Deux mécanismes spécifiques sont notamment décrits dans [98] : une implémentation spécifique nommée "silent" et une autre implémentation spécifique de type "heartbeat". La première pouvant induire des suspicions incorrectes dans certain cas, une amélioration a été proposée par la seconde implémentation spécifique. Une comparaison des différentes implémentations citées précédemment est proposée dans [37]. Cette analyse est basée sur l'efficacité relevée des différents modèles en fonction de la charge du système et de différents types de défaillances.

Solution adaptée au contexte environnemental :

Dans notre environnement, en raison de la forte variabilité des communications, une implémentation de type "pulsation", engendrant moins de messages en transit, semble plus adaptée qu'une implémentation de type "requête". Cependant la stratégie de type "requête" peut s'avérer intéressante afin d'éviter au maximum les fausses suspicions. En effet, il peut être intéressant de

privilégier la stratégie "pulsation" tout en utilisant la technique de type "requête" pour vérifier par exemple la véracité d'une suspicion.

De plus, il paraît judicieux de limiter au maximum le nombre de messages émis afin d'éviter au maximum de trop charger un réseau susceptible d'être fortement instable. Pour ce faire, quelque soit l'implémentation du détecteur de défaillances choisi, il peut être intéressant d'envisager d'utiliser parallèlement tous les messages reçus par les couches supérieures notamment. En effet, dédiés exclusivement au fonctionnement des détecteurs de défaillances ou non, tous les messages reçus peuvent faire office de messages de présence, et permettre la mise à jour des informations du détecteur de défaillances.

Cependant, face à la forte instabilité de notre environnement et étant donné que la diffusion est une opération unitaire dans les réseaux sans fil, une implémentation par le modèle "heartbeat" paraît satisfaisante, modulo bien sûr un paramétrage adéquat.

Suite à toutes ces propositions, l'idéal serait sans doute d'utiliser à la fois les deux implémentations de base présentées précédemment et d'apporter les modifications nécessaires afin de s'adapter au mieux à notre environnement. Par ailleurs, étant donnée la forte dynamicité de notre environnement, la mise en place d'un mécanisme autorisant un réglage dynamique des différents paramètres, serait certainement appropriée à notre contexte évolutif. Une implémentation adaptable des détecteurs de défaillances, utilisant une variante de la stratégie "pulsation", est proposée dans [12, 13] ; et bien qu'initialement conçue pour le support d'applications à grandes échelles, cette technique semble être transposable dans le contexte fortement variable des réseaux ambiants.

9.4 Conclusion

Les choix architecturaux concernant le module de détection de défaillances ont été motivés par la nécessité de flexibilité de l'environnement middleware développé au sein du projet Sidrah. En effet, l'architecture proposée se devait d'être relativement légère de façon à pouvoir s'adapter aux différents types d'entités matérielles mises en jeu, et suffisamment modulable de telle sorte que toutes améliorations ultérieures du traitement soient facilement intégrables.

D'autre part, bien que dans la suite de ce document le modèle d'implémentation des détecteurs de défaillances retenu s'apparente au modèle basique du "heartbeat", une étude succincte du type d'implémentation appropriée a été menée. Aussi, face à notre problématique, il paraît préférable d'envisager un modèle d'implémentation des détecteurs de défaillances spécifique à notre contexte. En effet, il apparaît évident que la nécessité d'un compromis entre la quantité de messages et la réactivité du détecteur de défaillances est d'autant plus vrai que l'on se place dans le contexte d'un réseau sans fil avec des latences (voir figure 5.5) et des débits particuliers. De plus, il faut aussi prendre en compte le compromis existant entre la fiabilité du détecteur de défaillances, au sens où l'idéal serait que le détecteur de défaillances ne fasse pas de fausse suspicion et la réactivité de celui-ci. Or, ce dernier compromis dépend directement du choix des paramètres associés aux détecteurs de défaillances. Aussi, quelle que soit l'implémentation retenue, le paramétrage à régler dépend étroitement de la qualité de service requise au détecteur de défaillances, et son efficacité varie en fonction de ce paramétrage (chapitre 10).

Qualité de service des détecteurs de défaillances

10

La mise en place et l'utilisation d'un détecteur de défaillances ont un coût (mémoire, temps). En effet, l'évaluation de l'état (présence ou absence) des entités peut nécessiter un nombre relativement grand d'échanges de messages en fonction de la qualité des informations souhaitée. Aussi, un compromis s'impose entre la qualité du détecteur de défaillances et la dégradation des performances impliquée. Il est donc nécessaire de qualifier la qualité de service des détecteurs de défaillances, ainsi que leur influence potentielle sur les performances du consensus.

Aussi, l'étude présentée ensuite consiste en la définition théorique de la qualité associée aux détecteurs de défaillances, et porte exclusivement sur le modèle d'implémentation de type "heartbeat". L'objectif de cette modélisation consiste en l'interprétation de l'impact du paramétrage, à savoir les délais d'émission des heartbeats et les fonctions d'estimation de suspicion, sur la qualité des détecteurs de défaillances. De manière simplificatrice, on supposera qu'il n'existe aucune corrélation entre les états (présence ou absence) des entités mises en jeu.

10.1 Qualité de service

La définition de la "qualité de service" offerte par les détecteurs de défaillances peut être exprimée de manière intuitive par : (i) la réactivité du détecteur de défaillances doit être la plus rapide possible et (ii) le détecteur de défaillance doit éviter au maximum les fausses suspicions. Ainsi, la qualité de service du détecteur de défaillances dépend directement de sa capacité à réagir rapidement aux événements extérieurs d'une part, et de sa capacité à fournir des informations correctes d'autre part. Cette notion de "qualité de service" a notamment été introduite et développée dans divers documents [55, 20].

Par ailleurs, cette notion de "qualité de service" est d'autant plus importante que l'évaluation des performances d'un algorithme de résolution du consensus dépend essentiellement du choix de l'implémentation des détecteurs de défaillances. En effet, les performances d'un algorithme

de consensus sont indissociables de la qualité de service offerte par le détecteur de défaillances, [98, 22].

D'autre part, le fonctionnement des détecteurs de défaillances tel qu'il est défini précédemment (chapitre 9), est influencé par deux paramètres principaux [98] : D_i, l'intervalle de temps entre les émissions de messages faites par le site i, et $\theta_j(i)$, la durée de la temporisation associée au site i sur le site j. De ce fait, la qualité associée aux détecteurs de défaillances dépend étroitement du réglage de ces deux paramètres. On entend par qualité d'un détecteur de défaillances à la fois sa capacité de réactivité, à savoir le temps nécessaire à la détection de la défaillance d'une entité distante, et sa fiabilité c'est à dire sa capacité à éviter les fausses suspicions.
Ainsi, une diminution du délai entre deux émissions de messages de présence augmentera la réactivité du détecteur de défaillances et donc limitera la durée d'une fausse suspicion. Cependant, l'augmentation du nombre de messages en transit a un coût au niveau de la charge réseau pouvant même engendrer une dégradation significative du système. Aussi, un compromis entre le nombre de messages en transit et la réactivité du détecteur de défaillances s'impose.
D'autre part, la fiabilité du détecteur de défaillances peut se décliner en deux points : (1) éviter la suspicion d'une entité distante présente dans le réseau et (2) éviter de ne pas suspecter une entité réellement défaillante. La fiabilité dépend donc du bon réglage de la durée de temporisation dont la valeur est elle-même fonction de l'intervalle de temps fixé entre les émissions de messages. Aussi, il existe également un compromis entre la réactivité du détecteur de défaillances et sa fiabilité.

10.2 Modèles stochastiques

L'analyse présentée dans cette section porte sur la modélisation du comportement des détecteurs de défaillances afin de qualifier et de quantifier la qualité de service souhaitée.
Dans l'étude du compromis escompté entre la réactivité et la fiabilité des détecteurs de défaillances, la principale difficulté rencontrée par ce type d'approche est le dimensionnement des temporisations. En effet, la qualité de service du détecteur dépend directement de l'estimateur de suspicion. Plusieurs études ayant déjà été menées sur ce thème [20], s'appuient sur une modélisation de l'environnement à partir des observations antérieures et supposent une "régularité statistique" du système.
Dans un premier temps, l'étude présentée ensuite a été menée sur des détecteurs de défaillances paramétrés de manière statique, à savoir le temps écoulé entre l'émission de deux "beats" successifs et la valeur de temporisation restent fixes quelque soit l'évolution de l'environnement.

Cependant, bien que l'on suppose que les détecteurs de défaillances émettent leurs "beats" à intervalles de temps réguliers, la variabilité environnementale ainsi que l'hétérogénéité des entités influent directement sur les durées entre deux réceptions de ces mêmes "beats" au niveau des détecteurs de défaillances distants. En effet, la figure 10.1 illustre les différents niveaux susceptibles d'influencer la variabilité des latences entre l'émission et la réception d'un "beat". Les différentes latences à prendre en compte correspondent à la descente ou respectivement à la remontée de la pile de protocoles (depuis le module d'exportation du détecteur de défaillances jusqu'au dépôt sur la carte réseau, et de la carte réseau au module d'importation du détecteur de défaillances distant respectivement), ainsi qu'à la traversée du réseau (diffusion sur les ondes).

Un premier modèle suppose l'indépendance entre les arrivées des "beats" au niveau du module

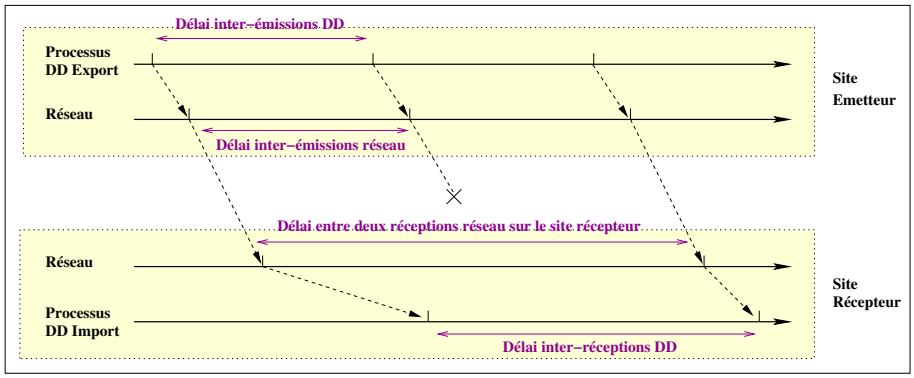


Figure 10.1 *Principe de fonctionnement entre deux détecteurs de défaillances*

d'importation du détecteur de défaillances. Le second modèle étudié ensuite, introduit une certaine forme de contention au niveau de la réception des "beats" sur le site distant. Ce modèle tient donc compte du délai entre l'arrivée d'un "beat" au niveau de la carte réseau de l'entité réceptrice et l'arrivée de cette information au niveau du module d'importation du détecteur de défaillances de cette même entité.

10.2.1 Indépendance des arrivées de heartbeats

Dans un premier temps, l'étude s'est restreinte aux détecteurs de défaillances avec temporisation fixe. C'est à dire que la suspicion d'une entité distante q par le détecteur de l'entité p se fait lorsque la dernière information reçue en provenance de q est antérieure à un seuil θ. La temporisation θ est fixée a priori et n'évolue pas pendant l'exécution de l'application.

On définit le taux de suspicion à tort $\phi_I(\theta)$ comme le rapport asymptotique entre le temps total de suspicion (zones grisées sur la figure 10.2) sur le temps total d'observation. On notera la fréquence de réception des "beats" par λ, c'est à dire que l'intervalle de temps entre deux réceptions sera en moyenne de $\frac{1}{\lambda}$. Il faut noter que cette valeur de λ est une approximation de la fréquence d'émission des "beats", fréquence λ_0 qui est initialisée au démarrage de l'application et qui est connue de tous. On peut remarquer que $\lambda = \lambda_0.(1-p)$ où p est le taux de perte des messages sur le réseau.

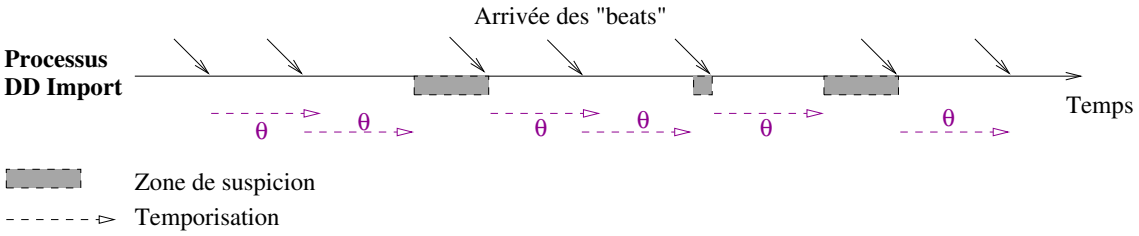


Figure 10.2 *Réception et zones de suspicion*

Pour analyser le comportement du détecteur de défaillances en régime "normal", le système sera considéré homogène, c'est à dire que l'environnement influe sur le système de manière stationnaire. Les perturbations sur les latences et les pertes de messages sont considérées comme des variables aléatoires dont la distribution est connue. Notons $\{X_n\}_{n\in\mathbb{N}}$ la suite des intervalles de temps séparant les réceptions des "beats". On supposera que ces intervalles de temps forment un processus stationnaire ergodique dont la loi stationnaire est connue.

Dans ce contexte il est possible de calculer mathématiquement $\phi_I(\theta)$. En effet, sur l'intervalle de temps i de durée X_i, le temps de suspicion est égal à $(X_i - \theta)^+ = \max(X_i - \theta, 0)$. Par suite,

$$\phi_I(\theta) = \lim_{n\to+\infty} \frac{\sum_{i=1}^{n}(X_i - \theta)^+}{\sum_{i=1}^{n} X_i} \tag{10.1}$$

et en passant à la limite, on obtient

$$\phi_I(\theta) = \lambda \mathbb{E}\left[X - \theta\right]^+ \tag{10.2}$$

Ce type de formule est très intéressant car il permet, dans un premier temps, de fixer la valeur de θ (la temporisation) en fonction du critère de qualité souhaité.

Loi exponentielle :

Par exemple, si l'on modélise la distribution des X_i par la loi exponentielle de paramètre λ (ajustement sur la valeur moyenne),

$$\phi_I(\theta) = \lambda \int_0^{+\infty} (x - \theta)^+ \lambda e^{-\lambda x} dx = e^{-\lambda\theta} \tag{10.3}$$

On retrouve naturellement l'impact de la décroissance exponentielle sur le taux de suspicion et dans ce cas des stratégies de type "additive increment" sembleront plus adaptées.

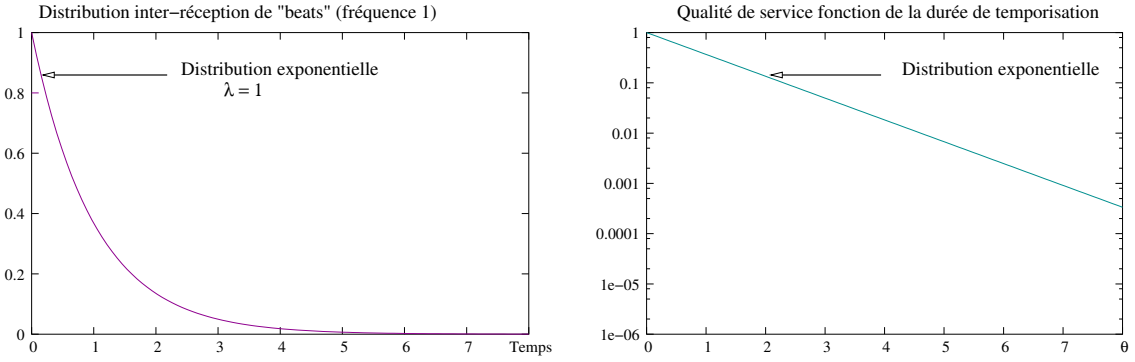


Figure 10.3 *Distribution des intervalles et probabilité de suspicion lorsque la distribution des X_i est modélisée par une loi exponentielle*

Ainsi, en fonction de la qualité de service escomptée au niveau des détecteurs de défaillances, on en déduit la valeur de temporisation minimale nécessaire à l'obtention de cette qualité. Par exemple, lorsque la distribution inter-beats est de type exponentielle, pour avoir une probabilité de suspicion à tort alors que le système fonctionne correctement fixée à 10^{-3}, la temporisation doit être de l'ordre de 7 fois la valeur moyenne d'inter-réception des "beats".

Loi d'Erlang :

Si la distribution des X_i est modélisée par une loi d'Erlang de paramètre $(k, k\lambda)$, "plus centrée" autour de la valeur moyenne λ (voir figure 10.4), on a :

$$\phi_I(\theta) = \lambda \int_0^{+\infty} \frac{(k\lambda)^k x^{k-1} e^{-k\lambda x}}{(k-1)!} (x-\theta)^+ dx \tag{10.4}$$

La variabilité est définie par le facteur k : plus k sera grand et plus la variance sera petite. En fixant k à la valeur 2, un calcul montre que :

$$\phi_I(\theta) = e^{-2\lambda\theta}(1+\theta) \tag{10.5}$$

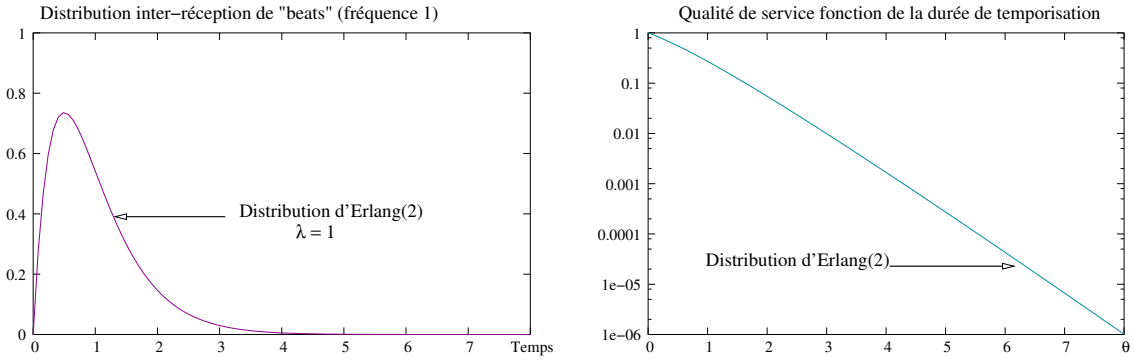


Figure 10.4 *Distribution des intervalles et probabilité de suspicion lorsque la distribution des X_i est modélisée par une loi d'Erlang*

Par exemple, pour avoir une probabilité de suspicion à tort de l'ordre de 10^{-3}, la valeur de temporisation choisie doit être approximativement de 4 fois la valeur moyenne inter-beats pour une distribution d'Erlang(2). Il apparaît alors que pour l'obtention d'une même qualité de service, lorsque la distribution des inter-réception de "beat" est de type exponentielle, la valeur de temporisation minimale nécessaire est bien plus grande que celle nécessaire dans le cas où la distribution est donnée par une loi d'Erlang.

Loi Normale :

De la même façon, si l'on modélise la distribution des X_i par une loi normale, faisant apparaître un petit bruit blanc autour de la valeur moyenne, de type $\mathcal{N}(\lambda, \sigma^2)$, (figure 10.5), on obtient :

$$\phi_I(\theta) \leq \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(\theta-\lambda)^2}{2\sigma^2}} \tag{10.6}$$

On fixe λ à 1 et σ à 0.25, et l'inégalité devient :

$$\phi_I(\theta) \leq \frac{0.25}{\sqrt{2\pi}} e^{-\frac{(\theta-1)^2}{0.12}} \tag{10.7}$$

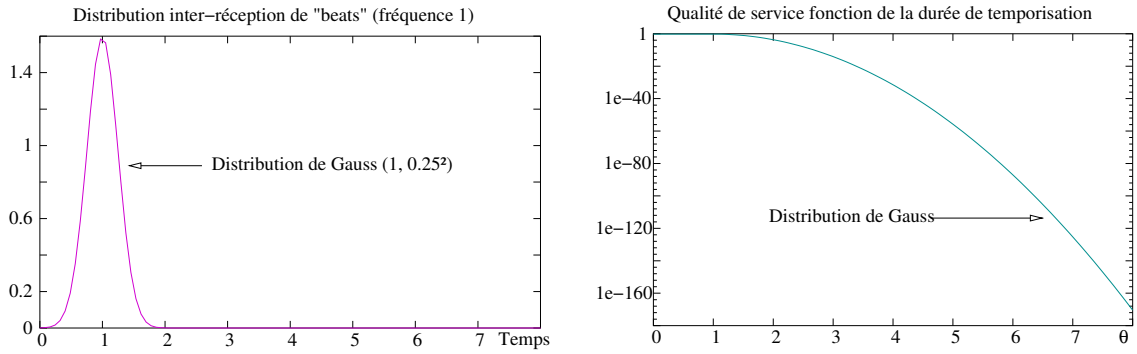


Figure 10.5 *Distribution des intervalles et probabilité de suspicion lorsque la distribution des X_i est modélisée par une loi Normale*

Loi de Pareto :

Si la distribution des X_i est désormais modélisée par une loi de Pareto de paramètres (λ, α), $\phi_I\theta$ (λ = 1) s'exprime sous la forme :

$$\phi_I(\theta) = \lambda \int_0^{+\infty} \frac{(\alpha-1)\,(\alpha-2)}{1+\frac{x}{\alpha-2}} (x-\theta)^+ dx \tag{10.8}$$

Prenons par exemple $\alpha = 3$, on a alors :

$$\phi_I(\theta) = \lambda \frac{1}{\theta+1} \tag{10.9}$$

La décroissance lente de cette distribution permet de considérer les situations dans lesquelles la variabilité reste forte.

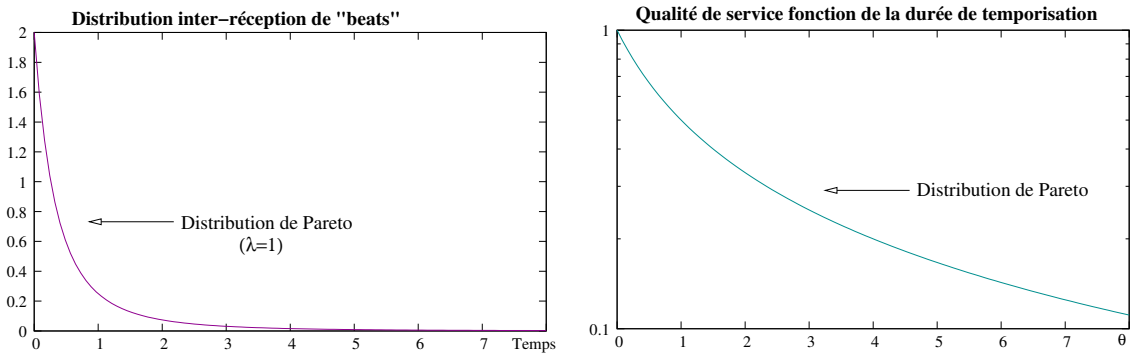


Figure 10.6 *Distribution des intervalles et probabilité de suspicion lorsque la distribution des X_i est modélisée par une loi de Pareto*

Synthèse :

On rappelle que λ correspond au débit d'émission des "beats".

Type de distribution	Propriétés	Forme de la temporisation
Exp(λ)	La distribution "la plus mélangée" sans d'autres informations que le débit moyen	$e^{-\lambda\theta}$
Erlang(k, $k\lambda$)	La distribution a une décroissance exponentielle rapide (émule la traversée de k étages de loi exponentielle)	$(1+\theta)e^{-2\lambda\theta}$ si k = 2
Gauss(λ, σ^2)	La distribution a une décroissance "exponentielle quadratique" très rapide (modèle de bruit blanc autour d'un signal déterministe, $\sigma \ll 1$)	Majoration : $\frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(\theta-\lambda)^2}{2\sigma^2}}$
Pareto(λ, α)	La distribution a une décroissance lente (cas de forte variabilité, observation de valeurs extrêmes)	$\frac{1}{\theta+1}$ si λ = 1, α = 3

Ainsi, en fonction de la variabilité attribuée au système étudié, et par suite du type de distribution considéré pour la modélisation des inter-arrivées de "beats", on déduit la forme de la temporisation préconisée.

Globalement, cette approche théorique peut facilement se combiner avec des données expérimentales en estimant la moyenne des inter-beats et en l'injectant dans le modèle.

10.2.2 Contention à l'arrivée des heartbeats

Lors de diverses expérimentations réalisées afin d'analyser le comportement des détecteurs de défaillances, nous avons pu constater une corrélation temporelle au niveau des inter-arrivées de "beats". En effet, les observations du réseau ("sniffer") lors de ces expérimentations mettant en œuvre plusieurs entités dotées de détecteurs de défaillances, ont mis en évidence que les inter-arrivées de "beats" au niveau des modules d'importation des détecteurs de défaillances n'étaient pas indépendantes.

Aussi, l'objectif de cette seconde analyse est de modéliser les inter-arrivées de "beats" de façon à considérer la variabilité du temps de remontée de la pile suivant les différents types d'entités.

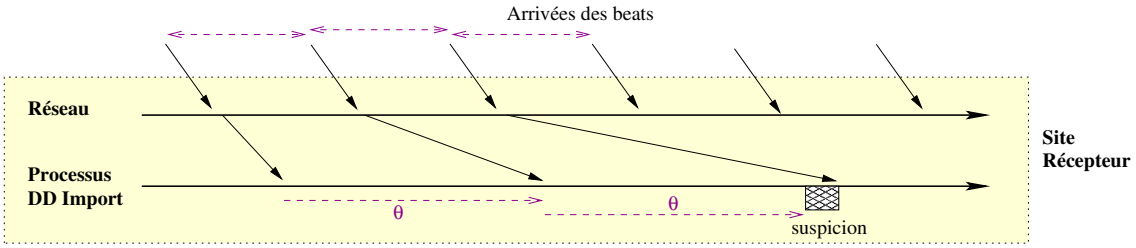


Figure 10.7 *Réception et zones de suspicion*

Pour ce faire, le mécanisme de réception des messages de heartbeat peut être modélisé par une file d'attente. Une file d'attente est caractérisée par trois variables principales [41] :

- un processus d'arrivée des clients dans la file (ici les messages ou "beats") qui permet de décrire l'entrée des clients dans le système,
- un processus de service décrivant le temps requis pour servir un client (ou temps de traitement),
- le nombre de serveurs dans le système (dans notre cas, un seul serveur).

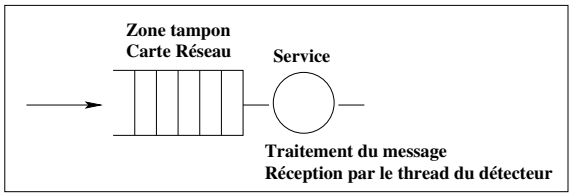


Figure 10.8 *File d'attente*

Description générale :

Chaque "beat" reçu correspond à l'arrivée d'un client dans la file d'attente considérée. On note $A(n)$ le temps écoulé entre l'arrivée du $(n-1)^{ieme}$ client (ou "beat") et l'arrivée du n^{ieme} client.

La densité de probabilité des inter-arrivées est notée $f_A(t)$ et sa transformée de Laplace est donnée

par :

$$f_A^*(s) = \int_0^\infty e^{-st} f_A(t)\, dt \tag{10.10}$$

Soit $S(n)$ le temps de service du n^{ieme} client, c'est à dire le temps que met un "beat" pour être traité et pris en compte par le détecteur de défaillances.
Si l'on note W_n le temps d'attente du n^{ieme} client, temps écoulé entre l'arrivée du client dans la file et le début de son service.
On a alors :

$$W_{n+1} = [W_n - A(n+1) + S(n)]^+ \tag{10.11}$$

Par ailleurs, on note $T(n)$ le temps de séjour du n^{ieme} client, et Z_n sa date de sortie de la file. La date de sortie du client $(n+1)$ dépendant de la date de sortie du client n, la relation entre les dates de sortie est donnée par l'équation suivante :

$$\begin{aligned} Z_{n+1} &= max\,[A_{n+1}, Z_n] + S(n+1) \\ Z_{n+1} - Z_n &= S(n+1) + [A_{n+1} - Z_n]^+ \end{aligned} \tag{10.12}$$

où A_{n+1} correspond à la date d'arrivée du $(n-1)^{ieme}$ client.

Lors de l'arrivée du client $(n+1)$, deux cas de figure peuvent être distingués :
- Soit la file est vide, voir figure 10.9, auquel cas le client $n+1$ est traité directement,
- Soit un certain nombre de clients précédents sont en attente de service, voir figure 10.10, auquel cas le client $n+1$ vient grossir la file d'attente et ne sera traité qu'après le traitement des clients précédents encore dans la file.

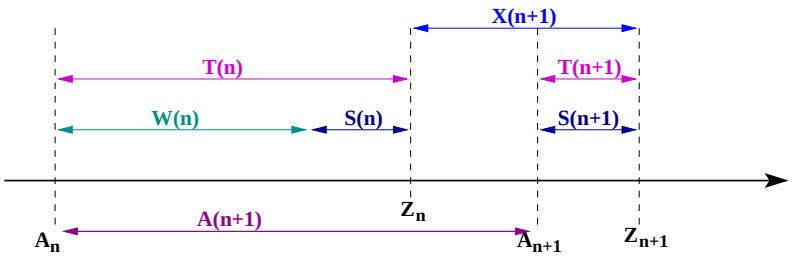


Figure 10.9 *La file est vide à l'arrivée du client (n+1)*

Ainsi, en supposant que la fréquence d'émission des "beats" ne soit pas source de surcharge réseau, lorsque le réseau reste fluide, la date de sortie de la file d'un "beat" dépend essentiellement de sa date d'entrée dans la file. Par contre, lorsque le réseau est saturé, la date de sortie de la file d'un "beat" dépend des "beats" précédents encore dans le cache à son arrivée.

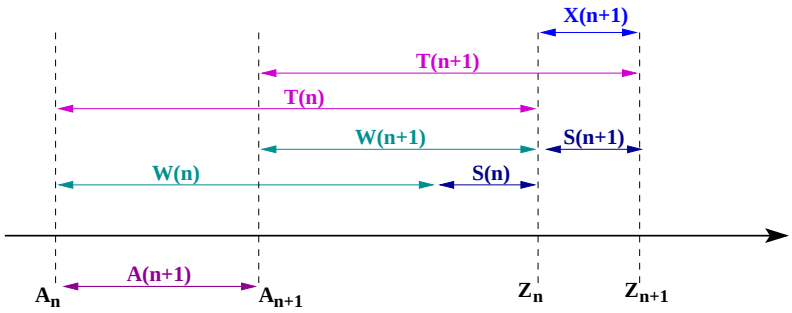


Figure 10.10 *Des clients sont en attente de traitement à l'arrivée du client (n+1)*

Indépendance des inter-arrivées : (au niveau réseau)

Dans notre contexte, on fait l'hypothèse que les inter-arrivées dans le système sont indépendantes de loi générale. On suppose également que S suit une loi exponentielle de paramètre μ. Ainsi, la file d'attente considérée est, dans la notation de Kendall, une file GI/M/1.

Soit λ le taux d'arrivées moyen :

$$\lambda = \frac{1}{\mathbb{E}\,[A]} \tag{10.13}$$

Les inter-arrivées n'étant pas sans mémoire, le processus $\{N_t\}_{t\in\mathbb{R}}$ modélisant le nombre de clients dans la file au cours du temps n'est pas markovien. Par contre, N_n, le nombre de clients dans la file à la date de la n^{ieme} arrivée est une chaîne de Markov incluse au processus $\{N_t\}$. Depuis l'état i, les transitions possibles vont d'une part vers l'état $i+1$ (arrivée d'un client supplémentaire), et d'autre part vers les états $i-x$, $x=0,...i$ (départ de un ou plusieurs clients).

Le temps de service S suivant une loi exponentielle de paramètre μ, le nombre de clients quittant le système pendant une durée D est distribué suivant une loi de Poisson de paramètre μD. On cherche la probabilité stationnaire $\pi_i, i = 0, 1, ...$ qu'à l'arrivée d'un nouveau client il y ait déjà i clients dans la file.
La transition de l'état i vers l'état $i+1-k$, [86, 112, 51], s'écrit :

$$\begin{aligned} p_{i,i+1-k} &= \int_0^\infty \frac{e^{-\mu t}(\mu t)^k}{k!} f_A(t)\,dt \qquad k = 0, 1, 2, ..., i, \\ p_{i,0} &= \int_0^\infty (1 - \sum_{l=0}^{i} \frac{e^{-\mu t}(\mu t)^l}{l!}) f_A(t)\,dt \\ &= \int_0^\infty \sum_{l=i+1}^{\infty} \frac{e^{-\mu t}(\mu t)^l}{l!} f_A(t)\,dt \qquad i = 1, 2, ... \end{aligned} \tag{10.14}$$

On a alors :

$$\pi_i = \sum_{j=0}^{\infty} \pi_j p_{j,i} = \sum_{j=i-1}^{\infty} \pi_j \int_0^\infty \frac{e^{-\mu t}(\mu t)^{j+1-i}}{(j+1-i)!} f_A(t)\,dt \tag{10.15}$$

Et finalement, la probabilité stationnaire, [112, 86], est de la forme :

$$\pi_i = (1-\beta)\beta^i \qquad i = 0, 1, ... \tag{10.16}$$

avec $0 < \beta < 1$.
C'est donc une distribution géométrique de paramètre β, où β est défini comme la probabilité que le système soit non vide à l'arrivée d'un client. Ce résultat vérifie bien que la probabilité d'avoir une file vide correspond à $\pi_0 = 1 - \beta$, et on a bien $\beta = 1 - \pi_0 = \sum_{i>0} \pi_i$.

Le temps d'attente moyen d'un client dans la file peut être calculé :

$$\begin{aligned} \mathbb{E}[W] &= \sum_{i=0}^{\infty} \pi_i i \mathbb{E}[S] \\ \mathbb{E}[W] &= \frac{\beta \mathbb{E}[S]}{1-\beta} \end{aligned} \tag{10.17}$$

Pour calculer β, on utilise la transformée de Laplace, et après quelques calculs, on obtient :

$$\beta = f_A^*(\mu - \mu\beta) = f_A^*(\mu(1-\beta)) \tag{10.18}$$

Si, à l'arrivée d'un nouveau client, k clients sont présents dans la file alors k services doivent être effectués avant que le service de ce nouveau client ne puisse commencer. Sachant que la somme géométrique de variables aléatoires exponentielles indépendantes est exponentiellement distribuée, et comme S suit une loi exponentielle de paramètre μ, la distribution du temps d'attente du client entrant dans le système est donné par une loi d'Erlang-k de paramètre μ. De la même façon, on obtient la fonction de répartition du temps de séjour dans la file :

$$F_T(t) = P\{T \leq t\} = 1 - e^{-\mu(1-\beta)t} \qquad t \geq 0 \tag{10.19}$$

Ainsi, la densité est donnée par :

$$f_T(t) = \mu(1-\beta)e^{-\mu(1-\beta)t} \tag{10.20}$$

Le temps de séjour dans la file suit donc une loi exponentielle de paramètre $(\mu(1-\beta))$.

Cependant, on cherche à calculer la durée écoulée entre deux inter-sorties de la file :
$X(n+1) = Z_{n+1} - Z_n$

$$X(n+1) = [A(n+1) - T(n)]^+ + S(n+1) \tag{10.21}$$

Spécialisation : D/M/1

Les observations effectuées suite à diverses expériementations, nous conduisent à faire l'hypothèse que la distribution des inter-arrivées est déterministe. On modélise donc une file D/M/1 [51].
Ceci revient à faire l'hypothèse que les émissions de "beats" sont périodiques.

Calcul de la fonction de répartition de $(A - T)^+$:

$$P\{(A-T)^+ \leq x\} = \begin{cases} 0 & si\ x < 0 \\ e^{-\mu(1-\beta)A} & si\ x = 0 \\ e^{-\mu(1-\beta)(A-x)} & si\ 0 \leq x \leq A \\ 1 & si\ x \geq A \end{cases} \tag{10.22}$$

Soit X la variable aléatoire à l'état stationnaire du processus $\{X_n\}$, on a donc :

$$\begin{aligned} P\{X \leq x\} &= P\{(A-T)^+ + S \leq x\} \\ &= \int_0^\infty P\{(A-T)^+ + S \leq x | S = s\} \mu e^{-\mu s} ds \\ &= \int_0^\infty P\{s + (A-T)^+ \leq x\} \mu e^{-\mu s} ds \\ &= \int_0^x P\{(A-T)^+ \leq x - s\} \mu e^{-\mu s} ds \\ &= \int_0^x P\{(A-T)^+ \leq t\} \mu e^{-\mu(x-t)} dt \\ &= \int_0^x P\{(A-T)^+ \leq t\} \mu e^{-\mu(x-t)} dt \\ P\{X \leq x\} &= \begin{cases} \frac{1}{2-\beta}(e^{-\mu(1-\beta)(A-x)} - e^{-\mu((1-\beta)A+x)}) & si\ x \leq A \\ 1 - \frac{e^{-\mu x}}{2-\beta}(e^{-\mu(1-\beta)A} + (1-\beta)e^{\mu A}) & si\ x \geq A \end{cases} \end{aligned} \tag{10.23}$$

Et la densité s'exprime comme suit :

$$f_X(x) = \begin{cases} \frac{\mu}{2-\beta} e^{-\mu(1-\beta)A}((1-\beta)e^{\mu(1-\beta)x} + e^{-\mu x}) & si\ x < A \\ \frac{\mu}{2-\beta} e^{-\mu x}(e^{-\mu(1-\beta)A} + (1-\beta)e^{\mu A}) & si\ x \geq A \end{cases} \tag{10.24}$$

L'hypothèse de la périodicité des émissions de "beats" implique : $f_A^*(s) = e^{-\frac{s}{\lambda}}$.
Par suite, on a :

$$\beta = e^{-\frac{\mu}{\lambda}(1-\beta)} \tag{10.25}$$

Exemple 1 *Sachant que la condition de stabilité est donnée par* $\frac{\lambda}{\mu} < 1$*, si l'on fixe* $\lambda = 1$ *et dans un premier temps* $\mu = 2$*, le calcul de* β *donne donc* $\beta = 0.2031878699$
□

On connaît désormais la distribution du temps écoulé entre deux sorties consécutives de la file d'attente. Cette distribution correspond à l'intervalle de temps entre les réceptions de "beats". Concrètement, pour le détecteur de défaillances du récepteur, cet intervalle de temps X doit être de durée inférieure à la valeur de temporisation θ. Si tel n'est pas le cas, le détecteur de défaillances du récepteur suspectera à tort l'entité émettrice pendant $(X - \theta)$ unité de temps.

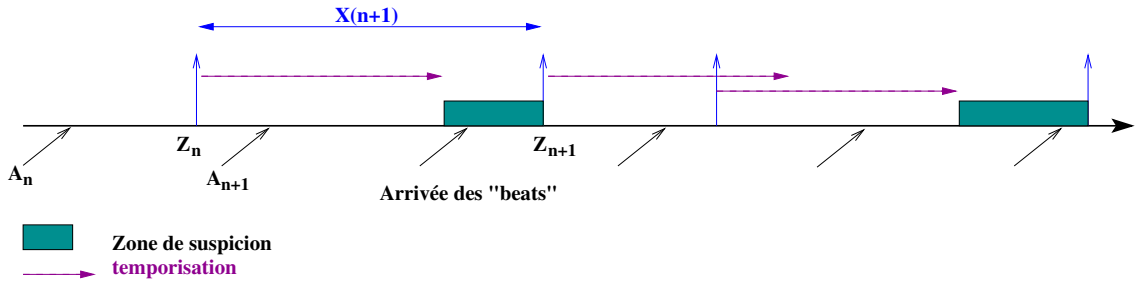


Figure 10.11 *Zones de suspicion*

Il reste donc à calculer le taux de suspicion à tort

$$\phi_I(\theta) = \lim_{n\to\infty} \frac{\sum_{i=1}^{n}(X_i - \theta)^+}{\sum_{i=1}^{n}(A_{i+1} - A_i)}$$

.
Et par suite :

$$\phi_I(\theta) = \lambda \mathbb{E}\left[X - \theta\right]^+$$

Pour ce faire, on doit donc calculer

$$\mathbb{E}\left[X - \theta\right]^+ = \int_0^\infty (x-\theta)^+ f_X(x)\, dx$$

On obtient finalement :

$$\mathbb{E}\left[X - \theta\right]^+ = \begin{cases} \int_0^A (x-\theta)\left[c_1 e^{\mu(1-\beta)x} + c_2 e^{-\mu x}\right] dx + \int_A^\infty (x-\theta) c_3 e^{-\mu x}\, dx & si\ \theta \le A \\ \int_\theta^\infty (x-\theta) c_4 \mu e^{-\mu x}\, dx & si\ \theta \ge A \end{cases} \quad (10.26)$$

Rappelons que θ est la valeur de temporisation choisie, c'est à dire que θ correspond au temps écoulé entre la dernière réception d'un "beat" par le détecteur de défaillances et le moment où celui-ci commence à suspecter le site distant s'il n'a pas reçu un nouveau "beat". Aussi, la valeur de temporisation ne paraît pertinente que si elle est supérieure à l'intervalle de temps entre deux émissions successives de "beats". Pour le calcul des suspicions à tort, on ne s'intéresse donc qu'au cas où $\theta \ge A$. Par suite, on obtient donc :

$$\mathbb{E}\left[X - \theta\right]^+ = \frac{1}{(2-\beta)\mu} e^{-\mu\theta}(e^{-\mu(1-\beta)A} + (1-\beta)e^{\mu A}) \qquad \theta \ge A \quad (10.27)$$

Aussi, le taux de suspicion à tort est donné par :

$$\phi_I(\theta) = \lambda \frac{1}{(2-\beta)\mu} e^{-\mu\theta}(e^{-\mu(1-\beta)A} + (1-\beta)e^{\mu A}) \qquad \theta \ge A \quad (10.28)$$

Cependant, ces résultats sont obtenus dans le cas d'une file d'attente de type D/M/1, à savoir la distribution des inter-arrivées est déterministe. Aussi, la prise en compte de la variabilité des inter-arrivées de "beats" dans le système conduit à considérer une file de type M/M/1. Or, le théorème de Burke [51] stipule que lorsque la distribution de service est de type exponentielle, si le processus

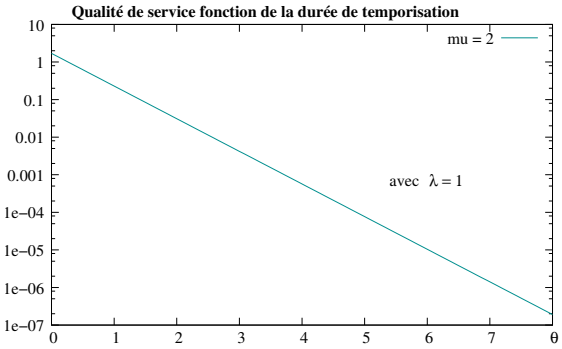


Figure 10.12 *Probabilité de suspicion à tort, si $\mu = 2$*

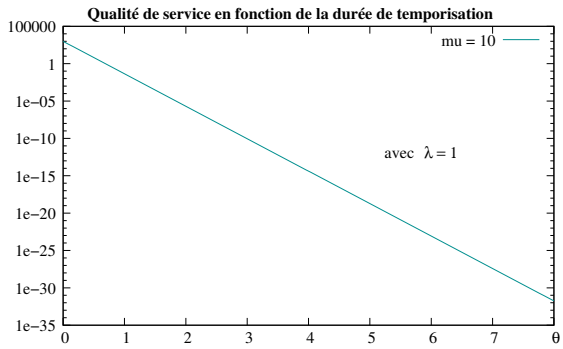


Figure 10.13 *Probabilité de suspicion à tort, si $\mu = 10$*

d'entrée dans la file d'attente est un processus de Poisson de paramètre λ, alors le processus de sortie d'une file M/M/1 est un processus de Poisson de paramètre λ. Ceci implique donc une majoration de la probabilité de suspicion à tort (ϕ_I) obtenue dans le cas d'une file M/M/1 par rapport à une file M/D/1 (voir 10.2.1).

10.3 Adaptation à la dynamicité de l'infrastructure

Les modélisations présentées dans la section précédente sont basées sur l'hypothèse que la temporisation est fixée préalablement en fonction de la qualité de service escomptée. Cependant, afin de prendre en compte les fluctuations environnementales, la valeur associée aux temporisations peut évoluer au cours du temps. En effet, nous pouvons envisager une adaptation dynamique de la valeur des temporisations en fonction de l'évolution des conditions environnementales et ce afin d'assurer une même qualité de service des détecteurs de défaillances. Pour ce faire, le calcul de l'estimation des temporisations peut être réalisé en fonction de l'historique des messages reçus en provenance d'un site distant. Dans la littérature, plusieurs travaux ont proposé une adaptation dynamique des temporisations :

- Le protocole proposé par Fetzer et al. [39] qui consiste à ajuster les temporisations en utilisant l'intervalle maximal entre les réceptions de deux messages consécutifs,
- L'estimation de Chen et al. [20] qui est basée sur l'estimation de la date d'arrivée d'un message à laquelle une marge de sécurité constante est ajoutée,
- L'algorithme de Jacobson [59] utilisé pour TCP, qui propose un calcul dynamique de la marge de sécurité en fonction des erreurs d'estimation précédentes,
- La fonction d'estimation de Bertier et al. [13] qui combine les deux estimations citées précédemment.

Aussi, notre approche permet tout d'abord de définir un ordre de grandeur de la valeur des temporisations en fonction de la qualité de service souhaitée. Par la suite, l'utilisation d'une technique d'ajustement dynamique des temporisations peut être intégrée afin de permettre l'évolution de ces valeurs de temporisations en fonction des conditions environnementales de façon à conserver une qualité de service similaire ou à la modifier volontairement.

10.4 Conclusion

Ce chapitre illustre toute la complexité attribuée aux réglages des paramètres. En effet, le surcoût en terme de trafic généré par le mécanisme des détecteurs de défaillances, peut avoir un impact conséquent si le réglage du délai entre les émissions de heartbeat n'est pas adéquat. En effet, étant données les caractéristiques particulières de l'environnement considéré, un mauvais paramétrage du délai entre les émissions de messages de présence peut engendrer une dégradation plus ou moins brutale des conditions environnementales.

Aussi, la modélisation proposée a été motivée par la nécessité de déterminer le plus finement possible les valeurs associées aux temporisations. Cette étude s'est donc focalisée sur la définition de la qualité de service associée aux détecteurs de défaillances en terme de fiabilité de l'information (éviter au maximum les fausses suspicions), et ce en fonction du rythme d'émission des "beats".

Par ailleurs, cette étude s'oriente désormais vers la mise en œuvre d'un paramétrage dynamique des temporisations. Quant au paramétrage dynamique du délai d'émission, il paraît également envisageable afin de s'adapter au plus juste aux aléas du réseau sous-jacent et aux besoins des applications en cours d'exécution, bien que cette évolution dynamique reste une tâche délicate. En effet, la modification du délai d'émission, valeur connue de toutes les entités composant le système, nécessite la réalisation d'un consensus distribué sur la nouvelle valeur du délai à adopter.

Réalisation d'un consensus 11

De nombreux algorithmes de résolution de consensus ont été proposés dans des systèmes rendus partiellement synchrones grâce à la mise en place de mécanismes de détecteurs de défaillances [97, 55, 19]. Cependant, tous ces algorithmes considèrent un environnement où les communications sont fiables. Version adaptée de l'algorithme de résolution du consensus basée sur les détecteurs de défaillances initialement décrit par Chandra & Toueg [19], l'implémentation proposée permet de pallier à cette nécessité de fiabilité des communications.

11.1 Algorithme de consensus en environnement sans fil

11.1.1 Consensus distribué : l'algorithme de Chandra & Toueg

Le principe de l'algorithme proposé par Chandra & Toueg repose sur le paradigme du coordinateur tournant. A savoir, le déroulement de cet algorithme de consensus se décompose en *rounds* [1] asynchrones chacun géré par une entité particulière du système appelée coordinateur. Soit N le nombre d'entités p_i, $i = 1...N$, participant à l'algorithme de consensus. Chaque entité p_i, $i = 1...N$, a une connaissance *a priori* des coordinateurs associés à chaque *round* .

Ainsi, durant un *round* R, le problème est temporairement centralisé au niveau du coordinateur correspondant. En effet, le coordinateur a un rôle majeur car il est chargé de la gestion du protocole d'accord (et donc de la prise de décision cohérente). Ce protocole consiste en la réalisation de deux phases principales : (i) la collecte des estimations des participants suivie de la communication de la proposition concernant la valeur de décision et (ii) la collecte des acquittements des participants et la transmission de la décision. Dans l'algorithme de Chandra & Toueg, collecter signifie que le coordinateur doit réceptionner une majorité[2] des messages envoyés par les participants ($\lceil \frac{(N+1)}{2} \rceil$).

[1] Par commodité de langage, ce terme sera employé tout au long du document (peut être traduit par "tour").

[2] Utilisant les détecteurs de défaillances $\diamond S$, cet algorithme permet de résoudre le consensus lorsque le nombre d'entités susceptibles d'être défaillantes est strictement inférieur à $f = \frac{N}{2}$. Aussi, dans le pire des cas, le coordinateur ne pourra collecter que $N - f$ valeurs.

Aussi, après l'étape de collecte, le coordinateur diffuse soit, lors de la première phase, une proposition réalisée via une fonction sur l'ensemble des estimations collectées, soit, lors de la seconde phase, la valeur de décision.

Le principe de fonctionnement de cet algorithme est qu'à chaque *round* plusieurs issues sont envisageables : la défaillance d'une entité ou de plusieurs entités, les temps de communication entre les entités plus ou moins longs, ... peuvent avoir diverses influences sur le déroulement de l'algorithme. Aussi, si les circonstances ne sont pas favorables à la prise de décision lors d'un *round* R, la résolution du consensus est remise à un *round* ultérieur. Les conditions environnementales pouvant évoluer rapidement, le passage au *round* suivant $R + 1$ peut permettre la résolution du consensus. Ainsi, lorsqu'une entité p_i est en attente d'un message de proposition en provenance du coordinateur du *round* R, elle interroge son détecteur de défaillances et, s'il s'avère que celui-ci suspecte le coordinateur, cette entité p_i passe au *round* suivant $R + 1$.

Cet algorithme de résolution de consensus proposé par Chandra & Toueg est décrit de façon plus détaillée en annexe D.

Principe de convergence vers une décision unique :

Lorsqu'une proposition est diffusée par le coordinateur chaque participant qui reçoit le message remplace son estimation par cette valeur et note le numéro de *round* associé à cette proposition. Ainsi, si un nouveau *round* est engagé les participants ayant déjà reçu une proposition la transmette indirectement au nouveau coordinateur qui adopte automatiquement la proposition précédente. Le fait d'avoir une majorité de participants corrects (non défaillants) et des communications fiables assure la convergence vers une décision unique. Le système converge donc vers une solution lorsque l'on parvient à une situation stable c'est à dire lorsque toutes les conditions de l'algorithme sont vérifiées.

11.1.2 Adaptation de l'algorithme de Chandra & Toueg

Pourquoi une adaptation est-elle nécessaire ?

L'algorithme de Chandra & Toueg repose sur des communications fiables. Hors, dans notre contexte, une telle propriété ne peut être assurée par la technologie de communication sous-jacente. En effet, dans un réseau sans fil, les communications se faisant par propagation d'ondes radio, elles sont fortement sensibles aux perturbations extérieures. Aussi, la variabilité des signaux peut introduire un grand nombre de pertes de communications.

D'autre part, l'algorithme proposé par Chandra & Toueg est basé sur l'existence d'une majorité d'entités correctes parmi les participants au consensus, à savoir des entités qui ne subiront aucune défaillance quelles qu'elles soient pendant le déroulement de l'algorithme. Cette hypothèse primordiale reste difficilement envisageable dans l'environnement considéré puisqu'à tout moment les entités mises en jeu peuvent subir des isolations réseau plus ou moins longues. Ces isolations dues essentiellement à une mauvaise propagation du signal peuvent être assimilées à des pertes de communications.

Du point de vue des détecteurs de défaillances, un participant momentanément isolé devrait faire l'objet d'une suspicion de défaillance. Il est alors possible que l'algorithme s'exécute pendant

plusieurs *rounds* et éventuellement finisse par décider (à condition qu'il y ait une majorité de participants et un coordinateur vus présents) avant que le participant isolé ne soit de nouveau visible des autres. Le retour d'un participant isolé devient alors équivalent du point de vue des détecteurs de défaillances à la correction d'une suspicion à tort. Cependant, l'absence potentielle d'une majorité de participants nécessite une adaptation de l'algorithme de Chandra & Toueg.

De plus, souvent résultant d'un éloignement important (sortie de zone de couverture), les déconnexions imprévisibles suivies éventuellement de reconnexions, conforte la possibilité de ne pas avoir une majorité d'entités présentes à chaque instant. D'autre part, ces déconnexions intempestives peuvent avoir une répercussion sur la diffusion de la valeur de décision. A savoir, il est nécessaire d'assurer que toutes les entités ne subissant pas de défaillance définitive finiront par être informées de la valeur de décision malgré leurs possibles déconnexions/reconnexions.

Modifications de l'algorithme :

Fiabilisation des communications :

Les communications peuvent être fiabilisées en ajoutant un module chargé de la retransmission périodique des messages. Cependant, la politique de réémission choisie (nombre de réémissions, période de réémission, choix des messages à réémettre) doit être adaptée à notre contexte au risque de générer une surcharge réseau qui ne ferait qu'augmenter la forte variabilité de l'environnement. Cette possibilité de réémission des messages doit donc permettre une certaine fiabilisation des communications mais en évitant au maximum de perturber le système.

De plus, la mise en place de ce module de fiabilisation permettra aussi de pallier à certains problèmes dûs aux isolations succintes. En effet, bien qu'une isolation très brève puisse avoir aucune incidence sur le déroulement de l'algorithme, elle peut être à l'origine d'une perte de message. Ceci n'entravera en rien le bon déroulement de l'algorithme si les messages ainsi perdus sont retransmis. Ainsi, pendant l'exécution de l'algorithme de consensus, une entité momentanément isolée pourra à son retour poursuivre activement sa participation au consensus.

Gestion des entités :

La forte variabilité de l'environnement considéré ne permet pas de garantir à tout instant la présence dans le système d'une majorité d'entités. Aussi, il s'avère indispensable de tenir compte des fluctuations du nombre d'entités présentes afin d'éviter de trop longues attentes ou de possibles interblocages. Pour ce faire, pendant l'exécution de l'algorithme de consensus, une utilisation plus systématique des détecteurs de défaillances est nécessaire afin de forcer le passage au *round* suivant lorsque les conditions de stabilité ne sont pas respectées. Pour le coordinateur du *round* courant, le non respect de la condition de stabilité correspond à la suspicion de plus de la majorité des participants, alors que pour un participant, elle correspond à la détection de la défaillance du coordinateur.

De plus, cette veille active qui consiste au sondage périodique des détecteurs de défaillances peut être couplée au module de retransmission des communications. Ainsi, après avoir évalué l'état du système en terme de défaillances, une entité en attente d'un évènement retransmettra son dernier message si la situation est stable ou passera au *round* suivant dans le cas contraire. Ce mécanisme permet donc à la fois de limiter le nombre de réémissions successives et d'éviter les

attentes trop longues. Cependant, dans un système très chaotique, ceci pourra conduire à un grand nombre de *rounds* et d'échange de messages.

Afin de permettre une resynchronisation temporaire des participants isolés, la transition d'un *round* vers un autre peut aussi être réalisée lorsque l'entité reçoit des informations estampillées par un numéro de *round* supérieur au sien.

Diffusion de la décision :

La mise en place d'un protocole de diffusion agressive de la valeur de décision *a posteriori*, couplée à l'utilisation d'un historique permet d'assurer que toutes les entités présentes finiront par avoir l'information. Ainsi, toute entité momentanément isolée sera informée de la prise de décision lors de sa reconnexion.

11.2 Description de l'algorithme de consensus

11.2.1 Modèle des communications

Les primitives de communication

L'algorithme de consensus présenté dans ce chapitre repose sur l'existence de deux primitives de communication. A savoir, une primitive de communication de type point à point et une primitive de diffusion de messages :

- L'émission d'une entité vers une autre (envoi point à point) :

$$sendMsg(< type, emetteur, \textsf{round}, v1, v2 >)$$

- L'émission d'une entité vers toutes les entités participantes (diffusion à tous) :

$$broadcastMsg(< type, emetteur, \textsf{round}, v1, v2 >)$$

Dans cette notation, le paramètre $type$ représente le type de messages émis, $type \in \{Estimate, Proposition, Ack, Nack, Decision, Forward\}$. Le paramètre $emetteur$ correspond à l'identifiant de l'entité émettrice (un identifiant unique par entité). Le paramètre *round* est le numéro du *round* en cours, c'est à dire le numéro du *round* dans lequel est l'entité émettrice. Enfin, les paramètres $v1$ et $v2$ correspondent quant à eux aux valeurs à transmettre.

Il est important de noter que les primitives de communication ainsi définies correspondent à des émissions de messages non blocantes. D'autre part, une hypothèse importante est celle concernant l'intégrité des messages. En effet, on suppose par la suite qu'il n'y a d'altération possible des messages.

Propriétés

Le modèle asynchrone dans lequel nous nous plaçons suppose que tout message émis finira par être reçu. Cependant, le modèle asynchrone considéré n'implique aucune hypothèse sur la durée d'une communication. En effet, d'une part le temps de communication (temps écoulé entre l'émission d'un message sur une entité et sa réception par une autre entité distante) n'est pas connu, et d'autre part, il n'y a pas de borne connue sur la durée des communications.

Propriété 1 *Les communications sont dites fiables au sens où tout message émis finira par être reçu.*

La propriété de fiabilité est assurée par l'existence d'une couche de fiabilisation permettant la ré-émission des messages (voir section 11.2.4). Cependant, pour des raisons d'optimisation, cette propriété est affaiblie. En effet, afin d'éviter une dégradation importante des performances, on considère que du point de vue de l'implémentation, seul le dernier message émis "compte". Aussi, cette propriété 1 peut être reformulée de la façon suivante :

Propriété 2 *Si, suite à une émission, l'entité émettrice n'a pas réalisé d'action (envoi ou réception de message), le dernier message émis finira par être reçu.*

Ainsi, deux hypothèses prises sur le réseau permettent de justifier en partie la propriété 2 énoncée précédemment :

Hypothèse 1 *Les pertes de communication sont indépendantes entre elles, et de même distribution.* □

Hypothèse 2 *Les isolations réseau sont indépendantes entre elles.* □

La première hypothèse correspond au fait que d'une manière générale, le comportement du réseau reste homogène. Cependant, on peut se demander si en cas de surcharge réseau, cette hypothèse d'indépendance des pertes de communications est encore vérifiable. Cette hypothèse semble relativement forte est mériterait sans doute d'être affaiblie. Par ailleurs, d'après la seconde hypothèse, on peut supposer qu'il existe un temps où une majorité de processus seront stables, c'est à dire dans notre contexte, connectés entre eux.

11.2.2 Automate spécifiant le comportement de l'algorithme

La figure 11.1 présente l'automate permettant de décrire l'algorithme de consensus proposé. Pour détailler rapidement cet automate, les états C-states (partie haute de l'automate) correspondent aux rôles attribués au coordinateur et se décompose en deux phases : la phase d'attente de réception d'une majorité de valeurs d'$Estimation$ (état $CWaitEstimate$) et la phase d'attente de réception d'une majorité d'acquittement (état $CWaitAck$). A l'opposé, les états P-states correspondent aux rôles attribués aux participants (autres que le coordinateur) et se décompose en deux phases : la phase représentant l'attente de la réception d'une valeur de $Proposition$ (état $PWaitProposition$) et la phase d'attente de réception de la valeur de $Decision$ (état $PWaitDecision$). Par ailleurs, deux opérations importantes sont illustrées dans cette figure 11.1 : l'action CHECKDETECTOR qui est systématiquement déclenchée après réception d'une $Proposition$ et l'action CHANGEROUND qui sera déclenchée après détection d'un problème (non stabilité du système) suite à une attente trop longue.

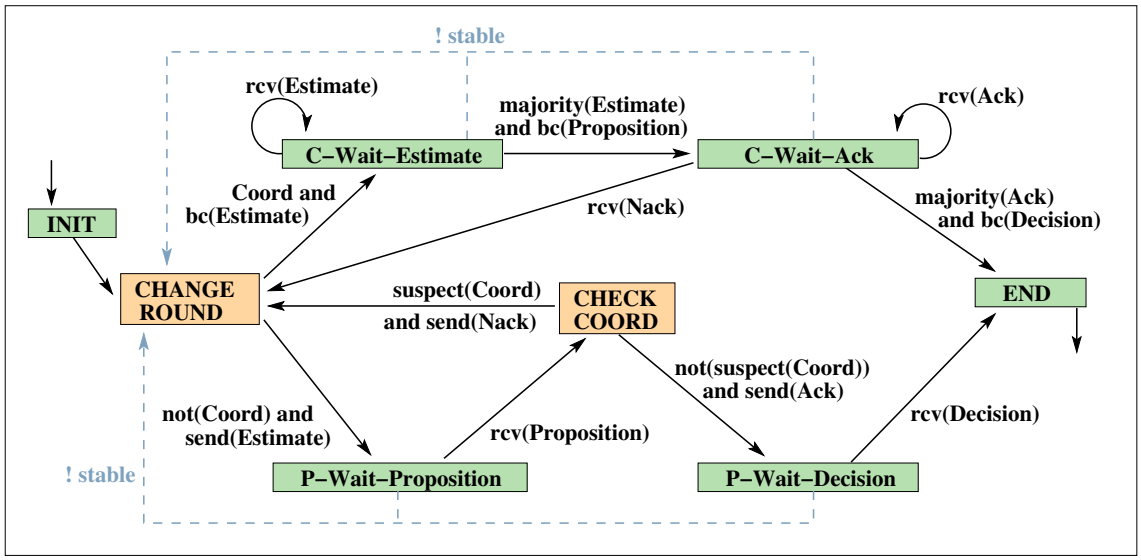


Figure 11.1 *Comportement local du module de consensus*

Les différents statuts possibles :

Comme l'illustre la figure 11.1, tous les participants à un consensus passent progressivement dans différents statuts spécifiques aux différents stades de l'évolution de la résolution de l'algorithme. Aussi, décrivons ces différents stades d'évolution de l'algorithme :

- $Status = \mathbf{INIT}$: Initialement (lancement de la procédure CONSENSUS()), chaque processus p participant possède ce statut. Cependant, après avoir exécuté la procédure CHANGE-ROUND(), le processus p ne reviendra plus dans ce statut, même lorsqu'il passera successivement dans les *rounds* suivants.
- $Status = \mathbf{CWaitEstimate}$: Un processus p ne peut avoir ce statut que s'il est coordinateur du *round* en cours. Ce statut correspond à l'étape où le coordinateur, ayant préalablement diffusé sa demande de valeur d'$Estimation$, est dès lors en attente de réception des différentes valeurs d'$Estimation$ des autres processus. Le coordinateur gardera ce statut jusqu'à ce qu'il diffuse la valeur de $Proposition$ retenue, auquel cas il passera dans le statut $CWaitAck$.
- $Status = \mathbf{PWaitProp}$: Seul un processus p non coordinateur du *round* en cours peut avoir ce statut. Etre dans ce statut, pour un processus p, signifie qu'il est en attente de la réception de la valeur de $Proposition$ diffusée par le coordinateur du *round* . Il est à noter que tout processus p ayant ce statut a préalablement envoyer sa valeur d'$Estimation$ au coordinateur.
- $Status = \mathbf{CWaitAck}$: Un processus p ne peut avoir ce statut que s'il est le coordinateur du *round* en cours. Lorsque le coordinateur a ce statut c'est qu'il est en attente de réceptions des acquittements (Ack) des autres processus. Le coordinateur ne passe dans ce statut qu'au moment de la diffusion de sa valeur de $Proposition$ et y restera jusqu'à ce qu'il ait reçu un message de type $Nack$ ou une majorité de messages d'acquittement (Ack).

- $Status = \mathbf{PWaitDecision}$: Un processus p ne peut avoir ce statut que s'il n'est pas coordinateur du *round* en cours. Un processus p ne passera dans ce statut que suite à la réception de la valeur de $Proposition$ et ce seulement après vérification de la stabilité du système (interrogation de son détecteur de défaillances : procédure CHECKDETECTOR) et envoie d'un message d'acquittement (Ack). Etre dans ce statut pour un processus p signifie qu'il attend la réception d'un message de type $Decision$ en provenance du coordinateur du *round* en cours.
- $Status = \mathbf{END}$: Tout processus ayant obtenu la valeur de $Decision$ passe finalement dans ce statut.

Types de valeurs utilisés :

Le type de valeurs sur lequel est réalisé le consensus ($Estimate_i$) peut être choisi en fonction du contexte. En effet, c'est de la fonction GETESTIMATE (appelée dans le procédure CHANGEROUND ligne 2) que dépend le type de valeurs considéré (entier, booléen, etc...). La fonction déterminant la politique de choix d'une valeur d'$Estimation$ parmi plusieurs (fonction COMPUTEPROPOSITION dans la procédure (TREATMSG ligne 24), doit être adaptée en fonction du type de valeurs d'$Estimation$ choisi.

Types de messages en transit :

Au cours de la réalisation d'un consensus, et suivant les diverses étapes d'évolution de l'algorithme, plusieurs types de messages peuvent être envoyés :

- $Estimate$: Le coordinateur comme tous les autres participants peuvent envoyer ce type de messages qui contient la valeur d'$Estimation$ de l'entité émettrice.
- $Proposition$: Seul le coordinateur envoie ce type de messages qui permet de diffuser la valeur de $Proposition$ faite par le coordinateur.
- Ack : Ce sont les participants (autres que le coordinateur) qui envoient ce type de messages à destination du coordinateur, lorsqu'ils ne suspectent pas ce dernier d'être défaillant.
- $Nack$: A l'inverse du message de type Ack, les participants notifient le coordinateur de leur suspicion à son égard par ce type de messages.
- $Decision$: Ce type de messages est exclusivement attribué au coordinateur puisqu'il lui permet de diffuser sa valeur de $Dcision$ aux autres participants.
- $Forward$: Tous les participants, coordinateur compris, sont susceptibles d'envoyer ce type de messages afin de permettre la propagation de la valeur de $Dcision$.

11.2.3 Fonctions associées à l'algorithme de consensus

```
CONSENSUS()
1  Status_p ← INIT
2  ValueDecid_p ← ∅
3  CHANGEROUND(0)
4  while Status_p ≠ END
5      do wait until receive message =< type_q, q, Round_q, Estimate_q, RoundEstim_q >
6          TREATMSG(message)
```

```
CLEARTABLES()
1  SetEstimates_p ← ∅
2  SetAcks_p ← ∅
```

```
CHECKDETECTOR()
1  coord ← GETCOORDINATOR(Round_p)
2  S ← {suspected_p}
3  if coord = p
4    then return Card(S) < N/2                    /* Optimisation */
5    else  return coord ∉ S
```

```
CHANGEROUND(r)
 1  if Status_p = INIT
 2    then Estimate_p ← GETESTIMATE
 3         RoundEstim_p ← −1
 4
 5  CLEARTABLES()
 6  coord ← GETCOORDINATOR(r)
 7  Round_p ← r
 8  if coord = p
 9    then Status_p ← CWaitEstimate
10         broadcastMsg(< ESTIMATE, p, Round_p, Estimate_p, RoundEstim_p >)
11    else  Status_p ← PWaitProp
12         sendMsg(< ESTIMATE, p, Round_p, Estimate_p, RoundEstim_p >) to coord
13
```

```
TREATMSG(msg)
 1  if (Status_p = END) and (type_q ≠ DECISION) and (type_q ≠ FORWARD)
 2    then broadcastMsg(< FORWARD, p, ∅, ValueDecid_p, ∅ >)
 3         return
 4  if (Status_p ≠ END) and ((type_q = DECISION) or (type_q = FORWARD))
 5    then ValueDecid_p ← Estimate_q
 6         Status_p ← END
 7         return
 8  Initialisation
 9  if Status_p = INIT or Round_p < Round_q
10    then CHANGEROUND(Round_q)
11    else if Round_p > Round_q
12           then return
13
14  Traitement en fonction du type de message reçu
15  switch
16    case type_q = ESTIMATE :
17         Si le message reçu est une ESTIMATION
18         et que p (=coord) est en attente d' ESTIMATION
19         if Status_p = CWaitEstimate
20           then SetEstimates_p ← SetEstimates_p ∪ [Estimate_q, RoundEstim_q]
21                if Card(SetEstimates_p) > N/2
22                  then M ← max_roundestim([estim, roundestim] ∈ SetEstimates_p)
23                       if M.roundestim = −1
24                         then Estimate_p ← ComputeProposition(SetEstimates_p.estim)
25                         else Estimate_p ← M.estim
26                       RoundEstim_p ← Round_p
27                       Status_p ← CWaitAck
28                       broadcastMsg(< PROP, p, Round_p, Estimate_p, RoundEstim_p >)
29
30         return
31    case type_q = PROP :
32         Si le message reçu est une PROPOSITION
33         et que p (!=coord) est en attente de la PROPOSITION du coord
34         if Status_p = PWaitProp
35           then Estimate_p ← Estimate_q
36                RoundEstim_p ← round_p
37                stable ← CHECKDETECTOR()
38                if stable
39                  then Status_p ← PWaitDecision
40                       sendMsg(< ACK, p, Round_p, ∅, ∅ >) to coord
41
```

```
42                  else sendMsg(< NACK, p, Round_p, ∅, ∅ >) to coord
43                       CHANGEROUND(Round_p + 1)
44          return
45      case type_q = ACK :
46          Si le message reçu est un ACK
47          et que p (=coord) est en attente d'ACK
48          if Status_p = CWaitAck
49            then SetAcks_p ← SetAcks_p ∪ {q}
50                 if Card(SetAcks_p) > N/2
51                   then ValueDecid_p ← Estimate_p
52                        broadcastMsg(< DECISION, p, Round_p, ValueDecid_p, ∅ >)
53
54          return
55      case type_q = NACK :
56          Si le message reçu est un NACK
57          et que p (=coord) est en attente d'ACK
58          if Status_p = CWaitAck
59            then CHANGEROUND(Round_p + 1)
60          return
```

11.2.4 Une implémentation de la fiabilisation des communications

```
RECEIVETIMER()
1  Procédure de retransmission
2  stable ← CHECKDETECTOR()
3  if stable
4    then RETRANSMIT()
5         now ← GETTIME()
6         SETTIMER(now + POLL)
7    else CHANGEROUND(Round_p)
```

Illustrant le protocole de fiabilisation des communications, la procédure RECEIVETIMER() présentée ci-dessus, permet notamment la retransmission périodique de certaines communications en fonction de la vision offerte par le détecteur de défaillances interrogé régulièrement (à intervalles de temps $Poll$).

Au sein de l'algorithme de consensus, le principe de l'utilisation de cette couche de fiabilisation des communication consiste à armer une temporisation (de durée $Poll$) après chaque émission (diffusion ou envoi point à point). Ainsi, si aucun événement nouveau (réception d'un nouveau message) n'a eu lieu à l'expiration de cette temporisation, la procédure RECEIVETIMER() est

alors appelée. De façon à pallier aux trop longues attentes voire aux éventuels blocages dûs à la perte de message, cette procédure permet donc à un participant, en fonction de sa vision locale du système global, de réémettre éventuellement le dernier message préalablement émis. Aussi, lors d'un *round* , cette procédure peut être appelée à diverses reprises :

- Par le coordinateur du *round* , lorsque celui-ci est en attente d'une majorité de réponses (majorité d'$Estimation$, dans la procédure CHANGEROUND(), ou majorité d'acquittement (Ack), dans la procédure TREATMSG()).
- Par chacun des participants, lorsqu'ils sont en attente d'un message en provenance du coordinateur (message de type $Proposition$, dans la procédure CHANGEROUND(), ou message de type $Decision$, dans la procédure TREATMSG()).

On remarque que pour un processus p, même si l'avant dernier message $E1$ a été perdu, seul le dernier message $E2$ est retransmis. Ceci parait être suffisant étant donné que les différentes étapes de l'algorithme de résolution du consensus sont une succession de transmissions des participants vers le coordinateur ("tous vers un") et du coordinateur vers les participants ("un vers tous") alternativement. Aussi, que le message $E1$ est été perdu ou non, si le processus p a émis un nouveau message $E2$, c'est qu'il a entre temps reçu un message en provenance du coordinateur (celui ci n'est donc pas bloqué).

Seulement, la réémission d'un message n'est effectivement réalisée que si la "stabilité" du système est avérée suite à l'interrogation du détecteur de défaillances (procédure CHECKDETECTOR()).

Définition 1 *Le système global est vu "stable" par le coordinateur du round si d'après son détecteur de défaillances, une majorité des entités du système n'est pas suspecté de défaillance.*
Du point de vue d'un participant (autre que le coordinateur), le système est vu "stable" si son détecteur de défaillances ne suspecte pas le coordinateur du round .

Ainsi, le couplage de ce mécanisme de retransmission des messages à l'utilisation du module de détection de défaillances CHECKDETECTOR() permet d'éviter une surcharge réseau inutile. Ainsi, si un des participants est en attente d'un message du coordinateur du *round* en cours, après expiration de sa temporisation, cette procédure lui permettra soit de réémettre son dernier message préalablement émis (cas où le système lui paraît "stable"), soit s'il suspecte le coordinateur du *round* en cours, de passer directement au *round* suivant. Symétriquement, si le coordinateur du *round* courant n'a pas réceptionné une majorité de messages en provenance des autres participants après le temps attribué à la temporisation définie, passera au *round* suivant s'il suspecte de défaillance une majorité de participants distants ou, dans le cas contraire, rediffusera son dernier message. Quelque soit le participant considéré (coordinateur ou non), après toute réémission d'un message, le même mécanisme de retransmission en fonction de la stabilité est enclenché de nouveau.

Ainsi, pour résumé, la procédure RECEIVETIMER() permet d'éviter les attentes infinies en retransmettant le précédent message émis ou bien en forçant le passage au *round* suivant.

11.2.5 Démonstration de l'algorithme de consensus

Définition 2 *En algorithmique distribuée tolérante aux pannes, il est commun de qualifier un processus de* **correct** *s'il n'est soumis à aucune défaillance quelle qu'elle soit.*
Parallèlement, un processus **incorrect** *correspond à un processus susceptible de subir une défaillance définitive.* □

Définition 3 *On dira d'un processus qu'il est* **instable***, s'il ne subit pas de défaillances définitives, mais peut cependant être momentanément défaillant.* □

Ainsi, dans notre contexte, et d'après les définitions proposées précédemment, on considérera qu'une majorité au moins des entités du système sont soit **correctes**, soit **instables**, les entités restantes pouvant quant à elles être qualifiées d'**incorrectes**.

Définition 4 *Un processus non défaillant à un instant t est dit* **présent**.
□

Ainsi, d'après cette définition, un processus **correct** sera donc toujours qualifié de **présent**. Les processus **instables**, quant à eux, seront par moments qualifiés de **présents**.

Hypothèse 3 *A terme, on finira par avoir une majorité de processus* **présents** *sur une période arbitrairement longue.* □

Cette hypothèse forte signifie implicitement qu'après un certain temps non défini, certains processus dits **instables** pourront être assimilés à des processus **corrects** et ce pendant une période arbitrairement grande. Autrement dit, il existe une durée suffisamment grande (relativement à la durée de plusieurs *rounds* , au moins $\frac{N}{2} + 1$, du consensus) pendant laquelle une majorité de processus impliqués peuvent être qualifiés de **corrects**.

Après avoir préalablement donné les définitions et hypothèses nécessaires, vérifions désormais le respect des propriétés d'**intégrité uniforme**, d'**accord uniforme** et de **terminaison** de notre algorithme de consensus.

Définition 5 ***Propriété d'Intégrité uniforme :***
"Si un processus décide, sa valeur de décision appartient à l'ensemble des valeurs proposées initialement." □

Démonstration Initialement, chaque processus p_i, $i = 1...N$, participant au consensus possède sa propre valeur d'estimation $Estimate_{p_i}$ (procédure CHANGEROUND, ligne 2). Lors de chacun des *rounds* , les processus p_i, $i = 1...N$, transmettent leur valeur $Estimate_{p_i}$ (procédure CHANGEROUND, lignes 10 ou 12).

Deux cas sont à distinguer selon le rôle du processus dans le *round* en cours :

- **Si p_i n'est pas le coordinateur du *round* en cours**, alors la seule modification possible de la valeur d'estimation $Estimate_{p_i}$ se situe lors de la réception par ce processus p_i de la valeur de $Proposition$ diffusée par le coordinateur du *round* en cours p_c (procédure TREATMSG, ligne 35). Or, dans ce cas, le processus p_i prend comme valeur d'estimation $Estimate_{p_i}$ la valeur proposée par le coordinateur $Estimate_{p_c}$, valeur contenue dans le message de $Proposition$ diffusé : $Estimate_{p_i} \leftarrow Estimate_{p_c}$. Aussi,
- **Si p_i est le coordinateur ($p_i = p_c$) du *round* en cours**, alors sa valeur d'estimation $Estimate_{p_c}$ ne peut être modifiée uniquement après réception d'une majorité de messages de type $Estimation$ en provenance des autres processus participants (procédure TREATMSG, lignes 24 ou 25). En effet, soit le coordinateur prend comme valeur d'estimation, une valeur parmi celles qu'il a reçu (procédure TREATMSG, lignes 24), $Estimate_{p_c} \leftarrow Estimate_{p_k}$, la politique de choix étant définie par la fonction COMPUTEPROPOSITION. Soit, le coordinateur du *round* en cours choisit comme valeur d'estimation, une valeur particulière parmi celles qu'il a reçu (procédure TREATMSG, lignes 25). En l'occcurence, il prendra la valeur envoyée par le processus p_j, ($Estimate_{p_c} \leftarrow Estimate_{p_j}$), p_j étant le processus, parmi ceux pour lesquels le coordinateur a reçu leur message d'$Estimation$, qui a modifié sa valeur $Estimate_{p_j}$ le plus récemment.

Ainsi, lors du premier *round* , chaque processus p_i, $i = 1...N$, envoie sa propre valeur initiale $Estimate_{p_i}$ (procédure CHANGEROUND, lignes 10 ou 12) au premier coordinateur, et au cours de chacun des *rounds* suivants, et suivant les modifications survenues pendant ce *round* , chacun des processus p_i, $i = 1...N$ enverra soit sa valeur initiale, soit la valeur modifiée au profit de la valeur d'un autre participant, aux coordinateurs des *rounds* ultérieurs. Aussi, d'après l'hypothèse d'absence de comportement malicieux et d'altération des messages en transit, lorsqu'un processus p_i, $i = 1...N$, modifie sa valeur c'est pour prendre la valeur envoyée par un autre processus participant au consensus. Ainsi, si finalement une valeur de $Decision$ est retenue, elle correspondra à une des valeurs d'estimation possédées initialement par les différents processus participants. □

Définition 6 ***Propriété d'Accord uniforme :***
"Si deux processus décident, ils décident sur la même valeur." □

Démonstration Commençons par séparer le cas où la décision collective est réalisée pour chacun des participants lors d'un même *round* , du cas où les participants vont décider lors de différents *rounds* .

- **Si tous les processus décident au même *round*** :
  Lorsque le coordinateur du *round* en cours décide, il diffuse de suite sa valeur de $Decision$ (procédure TREATMSG, lignes 52). Or, sous l'hypothèse d'intégrité des communications, comme tous les participants (autres que le coordinateur) ne pourront décider que suite à la réception du message de type $Decision$ envoyé par le coordinateur du *round* en cours, chacun des participants prendra comme valeur de $Decision$ la valeur diffusée par le coordinateur et retenue comme valeur de $Decision$ par ce dernier.

- **Si tous les processus ne décident pas pendant le même *round*** :
  Le seul cas dans lequel deux processus peuvent décider différemment correspond à un scénario où plusieurs *rounds* (au moins deux) ont été effectués et ces deux processus ont chacun été coordinateur d'un *round* différent. Essayons donc de montrer que le déroulement de l'algorithme de consensus peut conduire à ce que le processus p_1, coordinateur du *round* r_1, décide sur la valeur v_1, et que le processus p_2, coordinateur du *round* r_2, décide sur la valeur v_2, et ce sachant que $r_1 < r_2$.

  Si le processus p_1 décide, c'est qu'il a auparavant reçu les acquittements (Ack) d'une majorité ($\lfloor \frac{N}{2} + 1 \rfloor$) de participants.

  D'autre part, un processus n'est susceptible d'envoyer un acquittement (Ack) au coordinateur qu'après avoir préalablement reçu la valeur de $Proposition$ diffusée par le coordinateur et modifié sa valeur d'estimation : $Estimate_p \leftarrow v_1$. Aussi, dire que le processus p_1, coordinateur du *round* r_1, décide revient à considérer que plus de la majorité des participants (coordinateur compris) possède comme valeur d'estimation la même valeur que celle pour laquelle le coordinateur décide, à savoir ici v_1.

  De la même manière, les déductions faites sur l'état du système lors de la décision du processus p_1 au *round* r_1 sont menées similairement pour la décision sur la valeur v_2 du coordinateur p_2 du *round* r_2.

  Aussi, en considérant l'état de plus d'une majorité de processus dans le *round* r_1 et dans le *round* r_2, on peut en déduire qu'au moins un processus a participé activement à la fois au *round* r_1 et au *round* r_2, et par suite qu'il a donc changé sa valeur d'estimation d'un *round* à l'autre ($v_1 \rightarrow v_2$). Or, ce cas de figure n'est pas envisageable. En effet, nécessitant la réception de plus d'une majorité de valeurs d'estimation, la décision prise par le processus p_2, coordinateur du *round* r_2, devra impérativement prendre en compte la valeur d'estimation proposée par au moins un des participants du *round* r_1 (donc une valeur d'estimation égale à v_1). Aussi, le processus p_2, coordinateur du *round* r_2, devant nécessairement tenir compte de la valeur $RoundEstim_i$ associée à la valeur d'estimation, numéro du *round* le plus récent durant lequel $Estimate_i$ a été modifiée (procédure TREATMSG lignes 21 à 25), le coordinateur p_2 ne pourra alors que proposer la valeur d'estimation v_1.

Ainsi, que deux processus décident ou non pendant le même *round* , ils décident forcément sur la même valeur. □

Définition 7 ***Propriété de Terminaison :***
"Tout processus correct ou instable finit par décider" □

Démonstration Commençons tout d'abord par différencier le cas du coordinateur de celui des autres participants. Dans un premier temps, montrons que : "Il existe un *round* R où le coordinateur décide". Si tel est le cas (le coordinateur de *round* R décide), alors implicitement cela signifie qu'aucun message de type $Nack$ n'a été échangé durant le *round* R. Montrons alors qu'il existe un *round* R suffisamment grand dans lequel aucun message de type $Nack$ ne sera envoyé :

Un processus envoie au coordinateur un message de type $Nack$, s'il suspecte celui-ci de défaillance. Or, comme les détecteurs de défaillances respectent la propriété d'*exactitude ultime*

faible, il existe un processus correct p et un temps t tels que aucun processus correct ne suspecte p après le temps t. Implicitement, ceci signifie qu'après un certain temps, aucun processus assimilé correct n'envoie de message de type $Nack$ lorsque p est coordinateur du round R.

Aussi, sans message de type $Nack$ échangé pendant le *round* R, le coordinateur du *round* R est susceptible de prendre une décision. Seulement, le coordinateur de ce *round* R ne pourra décider que si la majorité des processus est présente (à savoir, si une majorité des processus sont assimilés corrects pendant une durée suffisamment longue). Or, d'après l'hypothèse : "A terme, on finira par avoir une majorité de processus présents sur une période arbitrairement longue", on peut supposer qu'au *round* R, R suffisamment grand, une majorité de processus sera effectivement présente. C'est à dire qu'en plus des processus corrects, un certain nombre de processus instables pourront être assimilés à des processus correct suffisamment longtemps, et ainsi s'ajouter de façon à ce qu'un majorité de processus puissent être qualifiés de corrects.

Dans ces conditions, le coordinateur du *round* R peut décider :

On sait que la majorité des processus présents ne suspectent pas le coordinateur pendant le *round* R (pas de message de type $Nack$).

Cependant, le coordinateur peut quant à lui suspecter une majorité de processus et ainsi passé au *round* suivant.

Or, comme la majorité des processus est réellement présente, et d'après la propriéte de symétrie du réseau au niveau des communications ainsi que les propriétés du détecteur de défaillances, le coordinateur finira par recevoir une majorité de messages de type Ack et finira donc ensuite par décider.

Par suite, si le coordinateur du *round* R décide, alors les autres participants finiront par décider. En effet, un participant finira soit par recevoir le message de $Decision$ envoyé par le coordinateur (procédure TREATMSG, ligne 4), soit par recevoir un message "FORWARD" (procédure TREATMSG, ligne 4). □

11.3 Bilan

Afin de pallier le problème de résolution du consensus en environnement asynchrone dans lequel les entités concernées peuvent être soumises à des défaillances, un algorithme de consensus basé sur un mécanisme de détections de défaillances a été présenté. Cet algorithme de consensus est une adaptation de l'algorithme proposé par Chandra & Toueg [19] permettant de prendre en compte les caractéristiques particulières de notre environnement. Ainsi, cet algorithme permet de résoudre un consensus malgré la non fiabilité des communications et les problèmes de déconnexions intempestives des entités composant notre système fortement variable. La principale particularité de cet algorithme repose sur l'utilisation d'un module de retransmission de certains messages couplé à une utilisation plus systématique du mécanisme de détection des défaillances afin d'éviter les attentes trop longues et d'empêcher les interblocages.

Afin de valider expérimentalement le fonctionnement de cet algorithme de consensus, les premières évaluations sont présentées dans la suite de ce document (partie III).

Conclusion 12

Cette partie s'est attachée à présentée nos choix algorithmiques vis à vis de la résolution de problèmes d'accord dans les systèmes distribués particuliers que nous considérons. Ces choix se sont donc portés sur les problèmes de résolution du consensus et toutes ses implications en environnement asynchrone devant supporter les défaillances des entités mises en jeu.

Aussi, nous nous sommes particulièrement intéressés à la mise en œuvre de mécanismes de détections de défaillances. Pour ce faire, après avoir présenté le principe général de fonctionnement des détecteurs de défaillances (chapitre 9), nos travaux se sont essentiellement focalisés sur le problème du réglage des paramètres liés aux détecteurs de défaillances. Motivé par la nécessité de tenir compte de la forte variabilité de notre environnement, l'étude du paramétrage nous a conduit à proposer une modélisation des détecteurs de défaillances afin de définir la qualité de service attendu en fonction du dimensionnement des temporisations (chapitre 10).

Par ailleurs, un algorithme de consensus adapté à la forte variabilité de l'environnement considéré a été proposé et développé (chapitre 11). Cet algorithme permet de contourner les obstacles que sont la variabilité des latences de communications, la non fiabilité du protocole de communication, ainsi que les déconnexions intempestives des entités du système. D'une manière générale, le principe de cet algorithme de consensus repose sur l'utilisation systématique du mécanisme indépendant de détections de défaillances.

Enfin, le principal avantage de l'architecture modulaire telle qu'elle a été implémentée concerne le module de détections de défaillances qui peut être utilisé indépendamment du module de consensus, directement par l'applicatif par exemple.

III

Mise au point et Dimensionnement

Après avoir détaillé l'environnement dans lequel nous nous plaçons et ses caractéristiques particulières dans une première partie, après avoir présenté les principes algorithmiques nécessaires à la prise de décision dans un environnement distribué ainsi que l'adaptation de l'algorithme de consensus retenu et implanté dans une seconde partie, cette dernière partie est consacrée à l'évaluation expérimentale de cet algorithme dans un environnement type de notre contexte. Aussi, dans un premier chapitre (chapitre 13), nous présentons une analyse de performances des détecteurs de défaillances implémentés. Le chapitre suivant (chapitre 14), quant à lui, est spécifiquement consacré à la validation expérimentale de l'algorithme de consensus développé ainsi qu'à une première étude des performances obtenues en terme de durée de réalisation d'un consensus et de la charge associée à l'exécution de cet algorithme.

Détecteurs de défaillances 13

Ce chapitre s'attache à analyser le comportement du mécanisme de détection de défaillances. Pour ce faire, ayant préalablement étudié l'impact du paramétrage sur la qualité de service des détecteurs de défaillances (chapitre 10), nous nous attarderons sur l'étude des délais de mises à jour de l'information (réactivité) des détecteurs de défaillances. Les diverses expérimentations présentées dans ce chapitre ont été réalisées dans différents contextes expérimentaux (milieu "idéal", milieu perturbé).

13.1 Première approche

Lors d'expériences préliminaires, nous avons constaté que le type de la machine (portable ou PDA) influe fortement sur le taux de perte et les latences. De manière générale, les expérimentations présentées ici portent sur la mise en œuvre d'un mécanisme de détections de défaillances de type "heartbeat". L'objectif principal est d'évaluer la capacité d'une entité à mettre à jour ses informations concernant les autres entités distantes.

Conditions expérimentales :

- 2 portables linux 800 Mhz
- 2 PDAs linux 200 Mhz
- Protocole de "heartbeat" par diffusion sur le réseau ad-hoc 802.11b
- Paramètre : temporisation des émissions de "beats" : 100 ms
- Mesures effectuées : envoi/réception de "beats"
- Durée de l'expérience : environ 15 min (environ 10000 mesures par échantillon).

Il faut noter que le système est fortement stressé et offre une réactivité de l'ordre de la seconde, ce qui pourrait sans doute être augmenté dans le cadre des applications envisagées sur ce type

d'architecture (partage de documentation, commande de présentation,...). On observe ainsi un nombre de "beats" non reçus pouvant être significatif (taux de perte de l'ordre de 50% pour les réceptions des PDAs).

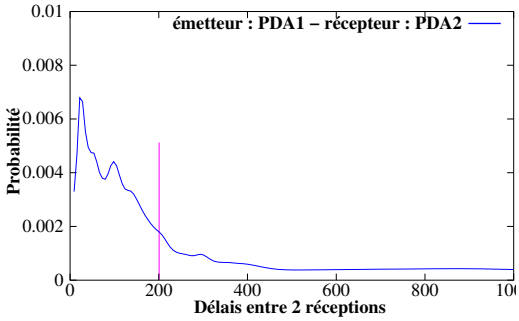


Figure 13.1 *Répartition des délais de mise à jour sur un PDA de l'information concernant un autre PDA distant.*

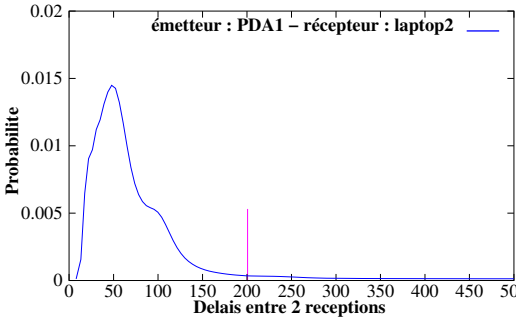


Figure 13.2 *Répartition des délais de mise à jour sur un PC portable de l'information concernant un PDA distant.*

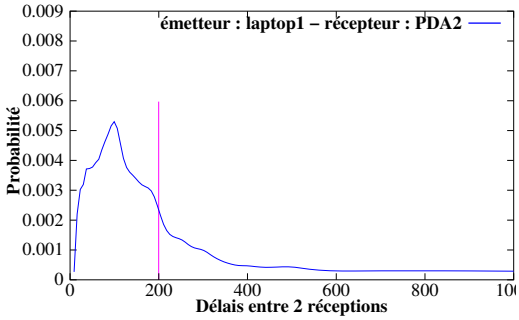


Figure 13.3 *Répartition des délais de mise à jour sur un PDA de l'information concernant un PC portable distant.*

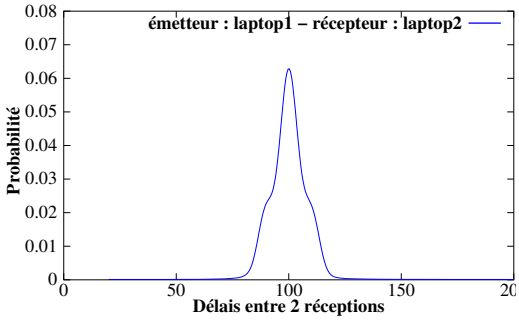


Figure 13.4 *Répartition des délais de mise à jour sur un PC portable de l'information concernant un autre PC portable distant.*

Il apparaît ainsi qu'une temporisation fixée à 200 ms ($2*$ le délai d'envoi des "beats") serait pour certains récepteurs beaucoup trop faible, et donc générateur de nombreuses fausses suspicions. En effet, lorsque l'émetteur et le récepteur sont tous deux des PCs portables (figure 13.4), les délais entre deux réceptions de "beats" sont centrés autour de la valeur de la durée moyenne d'émission (100 ms) ; et bien que ces délais peuvent être très longs, la plupart sont compris entre 50 ms et 150 ms. Dans ce cas, une temporisation fixée à 200 ms paraît être relativement bien adaptée dans le sens où malgré le fait qu'elle engendre quelques fausses suspicions, elle permet de conserver la réactivité du système. Par contre, lorsque le récepteur est un PDA (figures 13.1 et 13.3), aux vues des courbes de répartition des délais, il est évident qu'une temporisation fixée à 200 ms n'est pas vraiment adaptée car elle engendre un nombre beaucoup trop important de fausses suspicions.

Aussi, face à ces premiers résultats, il apparaît nettement une forte diversité dans la répartition des délais d'obtention d'informations de présence suivant les différents types de machines mises en jeu. Les figures 13.1 à 13.4 illustrent donc l'importance d'un bon dimensionnement des temporisations. En effet, dans le cadre de cette expérience (intervalle de 100 ms entre deux émissions), une temporisation paramétrée à 200 ms n'engendrera pas la même qualité des informations suivant le type des machines considérées :

- Un portable ne suspectera pas à tort un autre portable (figure 13.4)
- Un PDA risque par contre de régulièrement suspecter à tort un portable (figure 13.3)

Cependant, le paramétrage utilisé pour cette première approche expérimentale s'avère être générateur d'une importante surcharge du réseau. En effet, le réglage des pulsations à 100 ms est semble-t-il un facteur "stressant".

13.2 Expérimentation en milieu "idéal"

13.2.1 Conditions expérimentales

L'expérience précédente a montré qu'une fréquence d'émission de "beats" de 100 ms était génératrice d'un certain "stress" du réseau puisque l'on a observé de nombreuses pertes de messages ainsi que des délais importants entre les réceptions des "beats". Aussi, l'objectif de cette expérience est d'obtenir un échantillon de données susceptible de servir comme référence. Les paramètres seront donc réglés de façon à ce que le réseau fonctionne correctement (sans stress, sans surcharge, etc...). Les ressources mises en œuvre se composent de 6 machines : 3 PDAs (ipaq linux 200 Mhz), 2 PCs portables (linux 800 Mhz) et une machine servant de capteur de trafic (portable linux 800 Mhz). Écoutant en permanence le réseau et enregistrant tous les paquets concernant le mécanisme d'émissions de "beats" sur le réseau, le capteur permet d'avoir une vue plus ou moins extérieure à l'expérience.

Paramètres utilisés :

- Fréquence d'émission des "beats" : 500 ms
- Timeout : aucun (très long)
- Durée de l'expérience : environ 15 minutes

13.2.2 Pertes

La fréquence d'émission des "beats" (toutes les 500 ms) étant beaucoup plus grande que lors de l'expérience précédente (5 fois plus), le réseau se trouve bien moins chargé par le mécanisme de "heartbeat" et les pertes de messages engendrées sont donc limitées.

Émetteur	PDA 1	PDA 3	PDA 2	Laptop 1	Laptop 2
Nombre de messages émis	1281	1282	1224	1268	1294
Non réceptions de PDA 1	0	1	3	0	0
Non réceptions de PDA 3	0	0	2	1	0
Non réceptions de PDA 2	1	0	0	1	0
Non réceptions de Laptop 1	0	1	0	0	0
Non réceptions de Laptop 2	0	1	3	0	0

Figure 13.5 *Pertes (non réceptions)*

13.2.3 Analyse des Réceptions

Dans cette étude, on s'intéresse principalement à la durée écoulée entre 2 messages provenant du même émetteur, reçus sur un même site.

		Émissions	Réceptions				
			PDA 1	PDA 3	PDA 2	Laptop 1	Laptop 2
Émetteur : PDA 1	Moyenne	500	500	500	493	500	500
	Écart type	3	8	7	159	12	4
Émetteur : PDA 3	Moyenne	500	500	500	492	500	500
	Écart type	7	21	15	156	26	16
Émetteur : PDA 2	Moyenne	515	516	516	514	515	516
	Écart type	150	154	153	159	153	154
Émetteur : Laptop 1	Moyenne	500	500	501	501	500	500
	Écart type	20	26	30	170	20	22
Émetteur : Laptop 2	Moyenne	500	500	500	496	500	500
	Écart type	4	8	10	165	20	4

Figure 13.6 *Délai de mise à jour [en ms]*

Il est à noter que les moyennes des délais entre deux réceptions sont en majorité égales au temps écoulé entre deux émissions de "beats". Cependant, nous remarquons que les écarts type sont plus importants dès que l'entité PDA 2 est soit l'émetteur soit le récepteur des "beats". De plus, sachant que le délai entre deux émissions de "beats" consécutifs est fixé à 500 ms, il apparaît que contrairement aux autres entités, l'entité PDA 2 a une durée moyenne entre deux émissions supérieure à 500 ms (515 ms). Cette distinction s'explique par une particularité au niveau matériel de l'entité PDA 2 par rapport aux autres entités composant le réseau sans fil. En effet, les entités considérées sont mises en réseau grâce à des cartes sans fil et, mis à part pour l'entité PDA 2, les cartes utilisées sont des cartes de type PCMCIA identiques. Quant à l'entité PDA 2, la carte

insérée dans ce PDA est de type compact flash. Ainsi, ces résultats expérimentaux confirment que les particularités matérielles des entités présentes dans le réseau sans fil ont une influence significative sur les performances de ce même réseau.

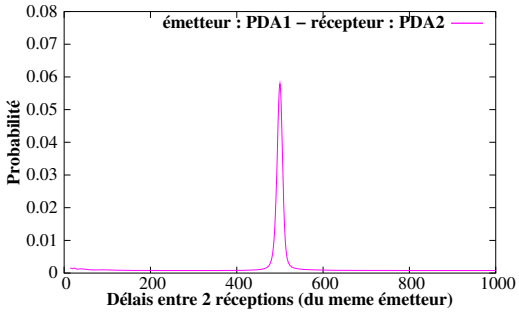


Figure 13.7 *Répartition des délais de mise à jour sur un PDA de l'information concernant un autre PDA*

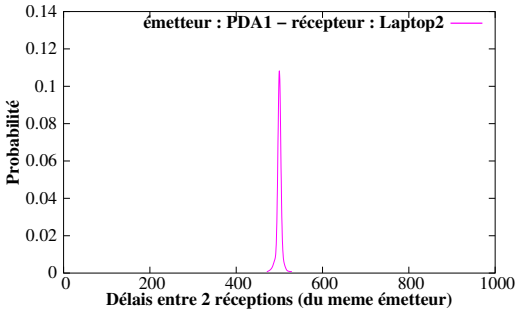


Figure 13.8 *Répartition des délais de mise à jour sur un laptop de l'information concernant un PDA*

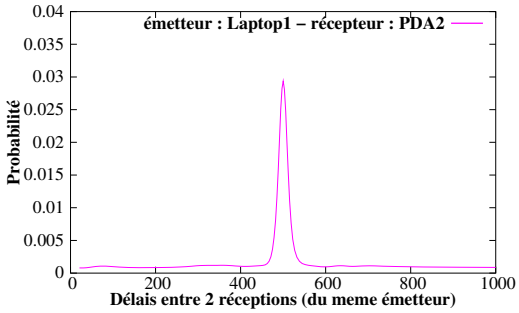


Figure 13.9 *Répartition des délais de mise à jour sur un PDA de l'information concernant un laptop*

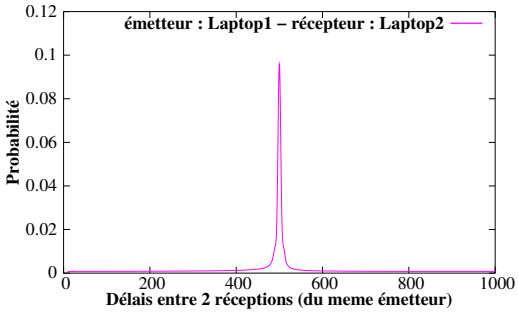


Figure 13.10 *Répartition des délais de mise à jour sur un laptop de l'information concernant un autre laptop*

D'autre part, les courbes présentées ci-dessus permettent d'observer une diminution générale de la variabilité. En absence de "stress", les différences entre les comportements des différents types de machines diminuent. Des temporisations fixées à $2 * (periode\ de\ pulsation)$ fournissent un taux de suspicion à tort de l'ordre de 10^{-3} lorsque les récepteurs sont des PCs, et de 10^{-2} lorsque les récepteurs sont des PDAs.

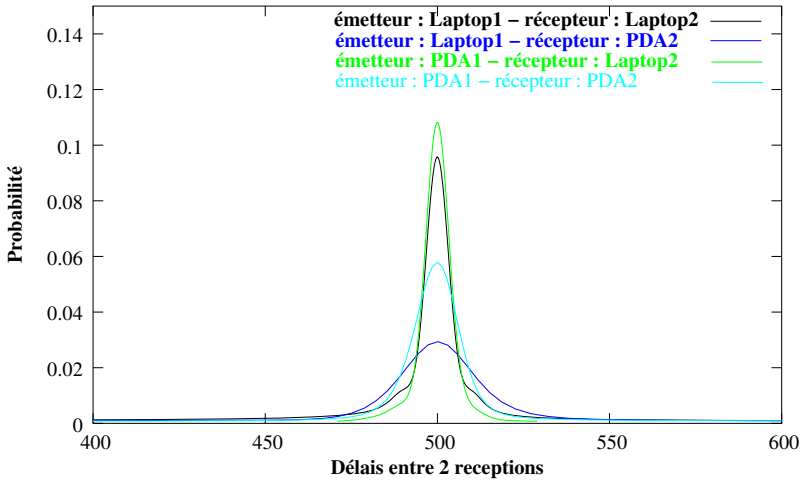


Figure 13.11 *Répartition des délais de mise à jour (pour comparaison)*

13.3 Expérimentation en milieu perturbé

13.3.1 Conditions expérimentales

Les conditions expérimentales de cette expérience sont identiques à celles présentées précédemment. Cependant, cette expérimentation a été réalisée en milieu "stressé", à savoir nous avons ajouté un bruit de fond afin de perturber le réseau. Pour ce faire, un PC portable (Laptop 1) a été utilisé pour générer un transfert de données (ping avec des paquets de 4 KB/20ms) vers une machine extérieure à l'expérience.

13.3.2 Pertes

Émetteur	PDA 1	PDA 3	PDA 2	Laptop 1	Laptop 2
Nombre de messages émis	1257	1154	1107	1268	1268
Non réceptions de PDA 1	0	3	30	38	0
Non réceptions de PDA 3	20	0	27	38	15
Non réceptions de PDA 2	21	12	0	37	12
Non réceptions de Laptop 1	171	143	97	0	143
Non réceptions de Laptop 2	21	17	23	38	0

Figure 13.12 *Pertes (non réceptions)*

Il apparaît une augmentation des pertes de messages par rapport à l'expérience précédente. De plus, la quantité de non réceptions de "beats" est beaucoup plus importante pour l'entité "Laptop 1" qui est générateur d'une surcharge réseau.

13.3.3 Analyse des Réceptions

De même que précédemment dans les expérimentations en milieu non perturbé (section 13.2), on a tout d'abord cherché à quantifier la durée écoulée entre 2 messages en provenance d'un même émetteur et reçus par un même site.

		Émissions	Réceptions				
			PDA 1	PDA 3	PDA 2	Laptop 1	Laptop 2
Émetteur : PDA 1	Moyenne	501	501	509	501	580	509
	Écart type	26	27	255	320	544	70
Émetteur : PDA 3	Moyenne	546	548	546	543	626	555
	Écart type	229	236	275	394	582	241
Émetteur : PDA 2	Moyenne	560	573	572	560	614	572
	Écart type	298	388	458	359	663	386
Émetteur : Laptop 1	Moyenne	500	517	516	512	500	516
	Écart type	17	149	289	346	17	147
Émetteur : Laptop 2	Moyenne	500	500	506	499	564	500
	Écart type	4	26	231	306	509	4

Figure 13.13 *Délai de mise à jour [en ms]*

La figure 13.13 montre qu'en moyenne les délais entre deux émissions de "beats" sont augmentés pour les entités "PDA 2" et "PDA 3". De plus, le temps entre deux réceptions de "beats" de "Laptop 1" est en moyenne bien supérieur par rapport à l'expérience précédente. Il apparaît donc que la charge réseau générée par l'entité "Laptop 1" a une forte influence sur le fonctionnement du mécanisme de heartbeat. Les valeurs des écarts type obtenus confirme l'impact de cette charge réseau sur le mécanisme de heartbeat. Il sera donc nécessaire d'en tenir compte dans le calcul des valeurs de temporisation.

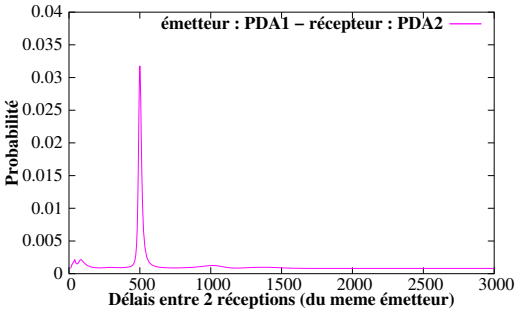


Figure 13.14 *Répartition des délais de mise à jour sur un PDA de l'information concernant un autre PDA*

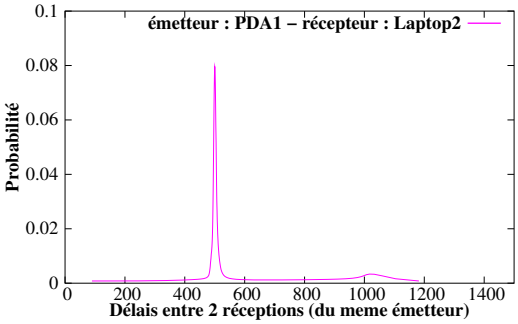


Figure 13.15 *Répartition des délais de mise à jour sur un laptop de l'information concernant un PDA*

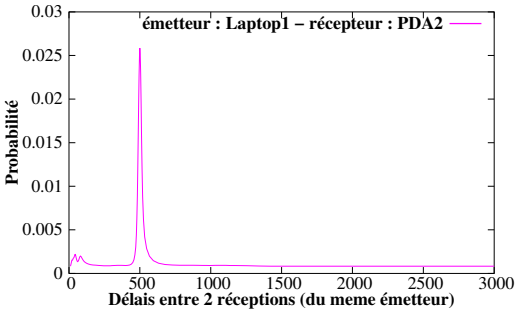


Figure 13.16 *Répartition des délais de mise à jour sur un PDA de l'information concernant un laptop*

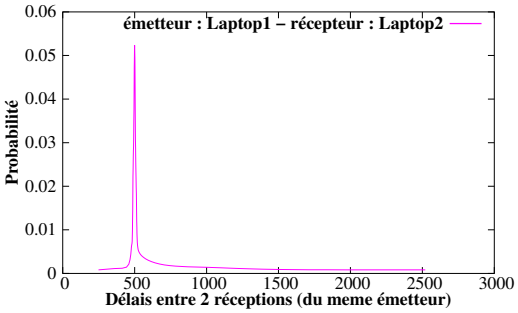


Figure 13.17 *Répartition des délais de mise à jour sur un laptop de l'information concernant un autre laptop*

Les courbes 13.14 à 13.17 confirment les valeurs des écarts type obtenus dans le tableau 13.13. En effet, il apparaît que les délais entre deux réceptions successives peuvent être jusqu'à plus de 6 fois supérieurs aux délais moyens d'émission des "beats". De plus, on peut noter qu'il y a beaucoup de délais relativement grands lorsque le récepteur est une entité de type PDA (figures 13.14 et 13.16). Par ailleurs, on peut remarquer que les délais peuvent être très courts. Ceci s'explique par le fait que suite à une longue attente (délai très long), les "beats" étant envoyés régulièrement, plusieurs messages ("beats") consécutifs peuvent arriver à intervalles de temps minimes.

Aussi, il semble qu'une corrélation existe entre des délais (temps entre deux arrivées) successifs.

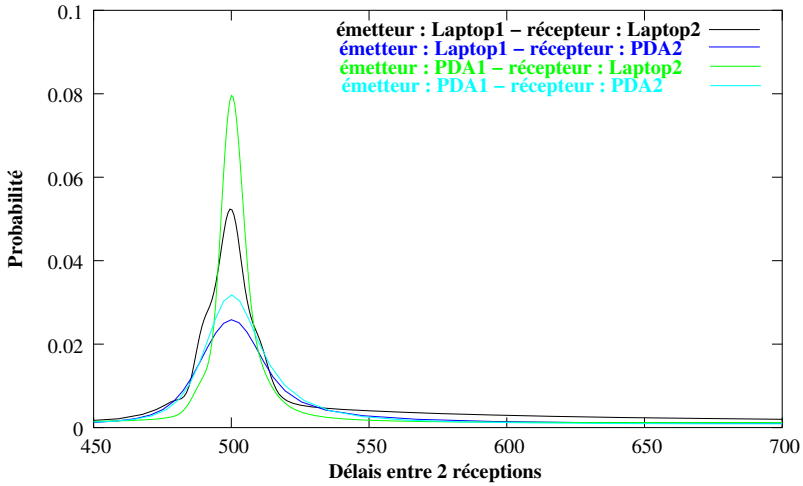


Figure 13.18 *Répartition des délais de mise à jour (pour comparaison)*

13.4 Bilan

Au cours de ces expériences, nous avons constaté la très forte dissymétrie entre les comportements des détecteurs de défaillances. Ceci est essentiellement dû aux types d'architectures mises en jeu. En effet, d'une part cette dissymétrie entre les types de machines (PCs portables, PDAs) est flagrante en milieu "stressé", mais très peu visible dans un contexte "idéal" (non stressé). Et d'autre part, nous avons mis en valeur l'impact que peut avoir le type de matériel utilisé (hétérogénéité des composants) ; impact qui n'est cependant pas forcément très significatif en milieu "stressé". Aussi, la temporisation apparaît alors comme fonction du type de l'émetteur et du récepteur. Les qualités de service seront donc très différentes malgré les fréquences d'émission similaires. Des expériences ultérieures ont confirmé ce fait. Il est donc nécessaire de procéder à une classification des architectures pour "caler" les modèles théoriques. De plus, ces expérimentations ayant mis en valeur une certaine relation entre deux délais successifs, il serait intéressant de mener une étude sur la corrélation existante entre plusieurs délais successifs (délais très longs suivis de délais très courts).

Consensus 14

L'algorithme de consensus présenté dans le chapitre 11 a été implanté sur un réseau ad-hoc sans fil de type 802.11b. De nombreuses prises de mesures ont été réalisées afin de permettre la génération de traces d'exécution de l'algorithme de consensus et ainsi de valider expérimentalement le fonctionnement de cet algorithme. Après avoir décrit le contexte expérimental dans lequel les expériences ont été réalisées, une section est consacrée aux difficultés liées à la prise de mesure et essentiellement au problème de datation en environnement distribué. Enfin, la présentation des résultats issus des tests expérimentaux réalisés dans différentes conditions environnementales, est proposée afin de permettre la validation de notre approche.

14.1 Implantation de la plateforme

14.1.1 Configuration matérielle et système

Une plateforme Java a été mise en place sur un réseau sous-jacent de type réseau 802.11b en mode ad-hoc. La configuration matérielle des différentes entités mises en jeu dans nos expériences est proposée dans le tableau 14.2.

Par ailleurs, précisons dès à présent que les expérimentations ont été menées sur un réseau sans fil 802.11b en mode ad-hoc dans lequel aucun mécanisme de routage ou d'encryption n'ont été mis en place.

14.1.2 Détecteurs de défaillances et consensus

L'objectif général consistant en l'implémentation d'une infrastructure flexible, l'architecture proposée est divisée en deux modules distincts : un module de détection de défaillances et le module moteur de consensus. Comme précisé précédemment, le principal avantage de la distinction effectuée entre ces deux modules réside dans la préservation de la possibilité de modification

	PC	PDA	Laptop
Type	HP Workstation	HP Ipaq 3800	HP Omnibook
Architecture	X86 (P4)	Strong ARM	X86 (PIII)
Processeur	2.4 GHz	207 MHz	1.0 GHz
Mémoire	512 MB	32 MB	256 MB
OS	Linux Debian (kernel 2.4)	Linux Familiar (kernel 2.4)	Linux Debian (kernel 2.4)
JVM	Sun Java JRE 1.4.2_01-b06	Jeode EVM Version 1.10.2	Sun Java JRE 1.4.2_01-b06
Wireless Card	Compaq Orinoco 802.11b	Compaq Orinoco 802.11b	Compaq Orinoco 802.11b

TAB. 14.1 *Configuration matérielle des machines.*

d'un module indépendamment de l'autre (changement au niveau des détecteurs de défaillances ou extension générique du consensus).

L'implémentation des détecteurs de défaillances présentée dans ce chapitre s'appuie sur la mise en place d'un mécanisme de "heartbeat" (chapitre 10 et chapitre 9). On rappelle que deux paramètres principaux sont associés à ce modèle d'implémentation des détecteurs de défaillances : la période de heartbeat (temps écoulé entre deux émissions successives de messages d'une même entité) et le délai de temporisation (temps entre la réception du message en provenance de l'entité q et le temps où l'entité p commence à suspecté l'entité q).

Par ailleurs, la réalisation d'un consensus nécessite que chaque entité composant le système étudié, à savoir les entités participant à l'algorithme de consensus, possède préalablement :

- un identifiant unique,
- la liste des participants (identique pour chaque entité),
- un algorithme de sélection (politique de choix d'une valeur parmi plusieurs).

Enfin, un autre point important à préciser concerne la fiabilisation des communications, laquelle est réalisée à intervalles de temps réguliers donnés par un paramètre de "polling" permettant ainsi de régler à la fois la fréquence d'interrogation du détecteur de défaillances et la retransmission des messages.

14.1.3 Modèle expérimental

De manière à décrire le plus précisément possible les expériences ayant été réalisées, la définition des conditions expérimentales fixées est nécessaire.

Aussi, commençons par préciser que l'étude présentée ensuite porte sur la réalisation de plusieurs expérimentations mettant en œuvre au total 7 entités hétérogènes : 4 PDAs, 2 ordinateurs portables et un PC fixe. Afin de capturer les communications réseau sans interférer dans les expérimentations, une entité particulière (un ordinateur portable) ne participant pas réellement aux tests de consensus, a fait office de "sniffer". Ainsi, hormis cette entité au rôle particulier, toutes

les autres entités sont sensées exécuter l'algorithme de consensus distribué présenté dans la partie précédente (chapitre 11).

Aussi, deux points importants sont à préciser : d'une part, le capteur de trafic réseau n'est pas intrusif et, d'autre part, les entités mises en jeu restent physiquement à la même place et toutes ont leur mode d'économie d'énergie inactivé et de plus restent alimentées électriquement tout au long des expériences.

Par ailleurs, chacune des expériences effectuées consiste en une série de 1000 élections consécutives (identifiées de manière unique) sans aucune synchronisation additionnelle, chacune des élections étant initiée par un des participants (par forcément le même). Rappelons le principe d'une élection dans cette série : une élection est réalisée grâce à l'exécution de l'algorithme de consensus, afin d'obtenir d'un accord sur la proposition d'un leader parmi les différents participants. Pour ce faire, lorsqu'une élection est engagée, chaque participant soumet sa propre estimation, laquelle correspondant en fait dans nos expérimentations à une valeur aléatoire. La politique de choix d'une valeur parmi plusieurs a été déterminée telle que l'algorithme d'élection sélectionne le participant ayant la valeur d'estimation la plus grande (celui ayant l'adresse IP la plus grande en cas d'égalité sur la valeur d'estimation).

D'autre part, comme présenté précédemment dans les chapitres 9 et 10, les détecteurs de défaillances sont dépendants de plusieurs paramètres. Dans les expériences présentées ensuite, les paramètres utilisés ont été fixés à :

- 500 ms pour la période d'émission des heartbeats (2 pulsations par seconde au niveau du module d'exportation d'un détecteur de défaillances),
- 1500 ms pour la valeur de temporisation (temps après lequel le détecteur de défaillances suspecte un site distant si aucune information ne lui est parvenue en provenance de ce site).

14.2 Prise de mesures et datation

L'objectif principal visé concerne l'observation du déroulement des élections et la prise de mesures de performances en termes de latences. Une méthode analytique a été mise en place afin de pouvoir construire des diagrammes temporels représentant le déroulement des élections et ce en vue de valider la fonctionnalité de notre algorithme.

14.2.1 Phase d'instrumentation

La collecte d'informations est une étape délicate. En effet, la prise de mesure est réalisée en cours d'exécution : chaque événement (réception d'un message, émission d'un message, changement d'état, etc...) doit être enregistré localement par chacun des participants et estampillé par sa date locale. Cependant, la prise de mesures représente une perturbation dans l'exécution normale et a donc un coût (prise de date, enregistrement) dont il faudra tenir compte lors de l'interprétation des données ainsi récoltées. Aussi, afin de limiter au maximum ce coût lié à la prise de traces, une analyse précise des événements significatifs et nécessaires à la fois à l'évaluation et à la visualisation, doit être effectuée.

Les enregistrements des événements locaux de chaque participant ainsi que ceux du capteur de trafic réseau permettront par la suite de construire des diagrammes "post mortem" représentant

l'exécution des élections. Cependant, la représentation de traces d'exécution nécessite une relation de causalité entre les différents événements ainsi récupérés. Or, dans un système réparti où les événements sont estampillés par la date locale de l'entité concernée, une relation d'ordre entre ces événements n'a de sens que localement. On est donc confronté au problème de datation globale des événements dans un système distribué.

14.2.2 Extrapolation d'une horloge globale

Chaque entité se référençant à sa propre horloge physique, l'objectif est de définir une horloge de référence permettant d'avoir une datation globale. En effet, définir des relations entre une horloge de référence et chacune des horloges locales de chaque participant permet d'avoir une approximation d'horloge globale. Cette solution proposée dans différents travaux [50, 76, 90] permet de calculer des facteurs correctifs pour chaque datation locale afin de les projeter dans un référentiel de temps commun. Ce principe de datation globale qui consiste en une synchronisation des différentes datations locales utilisées, ne cherche pas à extrapoler une horloge globale précise mais simplement à définir une relation d'ordre partielle entre les événements s'étant produits sur différentes entités réparties.

D'après Lamport [71], dans un système distribué, une relation partielle de précédence causale peut être définie comme suit :

Définition 8 *On dit qu'un événement e_i précède causalement un événement e_j ($e_i \prec e_j$) si et seulement si une des trois conditions suivantes est vérifiée :*

1. *les événements e_i et e_j se produisent sur le même processus et e_i se produit avant e_j localement.*
2. *les événements e_i et e_j se produisent sur deux processus distincts et e_i est l'émission d'un message tandis que e_j est la réception de ce même message.*
3. *$\exists k$ tel que $e_i \prec e_k$ et $e_k \prec e_j$*

Chaque événement, initialement daté localement, doit donc être daté approximativement dans un référentiel commun. La date locale d'un événement e_i s'exprime donc dans le référentiel global par la formule mathématique suivante 14.1 :

$$d_{ref}^{e_i} = \alpha d_{loc}^{e_i} + \beta \tag{14.1}$$

où α représente la dérive et β le décalage initial.

Aussi, si l'on ne s'intéresse qu'aux événements de types émissions et réceptions de messages, la relation de causalité existant entre les divers événements est basée sur le fait que la date d'émission d'un message E_i exprimée dans le référentiel global précède forcément la date de réception de ce message R_i dans le même référentiel :

$$d_{ref}^{E_i} \leq d_{ref}^{R_i}.$$

Si le message i a été émis par l'entité A à la date $d_A^{E_i}$ et réceptionné par l'entité B à la date $d_B^{R_i}$, on a donc :

$$\alpha_A d_A^{E_i} + \beta_A \leq \alpha_B d_B^{R_i} + \beta_B \tag{14.2}$$

Bien évidemment, de telles relations existent entre les différentes entités ayant au moins une communication entre elles, mais suivant les expériences réalisées certaines entités n'échangent pas de message. En effet, l'algorithme de consensus définit des communications seulement du coordinateur vers les autres participants ou des participants vers l'entité coordinatrice du round. Aussi, quelques soient les conditions expérimentales, l'élection se déroulant en un round au minimum, le référentiel de temps du premier coordinateur (première entité sur la liste des coordinateur) semble être le plus adapté comme référence de temps global.

On cherche donc les α_k et β_k tel que :

$$\alpha_A d_A^{E_i} + \beta_A \leq \alpha_B d_B^{R_i} + \beta_B, \forall i \tag{14.3}$$

$$\alpha_B d_B^{E_i} + \beta_B \leq \alpha_A d_A^{R_i} + \beta_A, \forall i \tag{14.4}$$

Devant le grand nombre d'événements, les calculs réalisés portent sur les enveloppes convexes de chaque type d'événements (émissions, réceptions) et sur chacune des entités afin de réduire l'ensemble des contraintes.

On cherche donc une droite passant entre deux enveloppes convexes (définies par l'équation 14.3) afin de satisfaire toutes les contraintes. Le système à résoudre se ramène donc à :

$$v_1 \leq \frac{\alpha_A}{\alpha_B} \leq v_2 \tag{14.5}$$

$$v_3 \leq \frac{\beta_A - \beta_B}{\alpha_B} \leq v_4 \tag{14.6}$$

Ces relations existent entre chaque couple d'entités mises en jeu ayant des interactions au cours de l'exécution d'une expérience.

Ainsi, on définit les α_k et β_k, $k = 0...N$ (où N est le nombre d'entités dans le système), de façon à ce que toutes les relations de causalité soient respectées. De ce fait, rien n'assure de l'exactitude de la datation dans le référentiel global, mais seulement de l'ordre des événements en relation directe. Aussi, la nouvelle datation ainsi obtenue (datation globale) peut éventuellement être biaisée du fait de l'imprécision dans la prise de dates (top d'horloge, dérive due aux températures, la prise de date fait suite à l'événement, ...). De plus, les événements n'ayant aucune relation de causalité entre eux sont susceptibles de ne pas être placés dans l'ordre de leur exécution réelle.

Finalement, cette méthode permet simplement de construire une approximation linéaire d'une horloge globale qui respecte analytiquement la causalité des événements de l'exécution observée.

14.2.3 Interprétation

Les différents événements enregistrés sur les diverses entités participant aux élections, une fois ordonnés de façon cohérente et datés approximativement dans un référentiel commun, peuvent être représentés graphiquement. Un environnement de visualisation des exécutions de programmes [70] est utilisé pour réaliser les diagrammes "post mortem". Le déroulement des différentes élections réalisées peut ainsi être représenté graphiquement dans l'outil de visualisation Pajé [70] puis analysé.

14.3 Tests expérimentaux

14.3.1 Comportement "idéal"

La construction de diagrammes temporels représentant le déroulement d'une élection permet de vérifier le fonctionnement de l'algorithme proposé. Ces traces d'exécution sont obtenues *post-mortem* grâce aux informations collectées pendant l'exécution, cependant chaque événement collecté est daté localement. Aussi, la méthode utilisée consiste en la construction d'une horloge globale linéaire basée sur le respect de la causalité des événements.

14.3.1.1 Pas de suspicion

La trace d'exécution 14.1 illustre les différentes étapes du déroulement d 'une élection particulière dans un cas "idéal" (aucune perturbation extérieure, aucune suspicion, ...). Cette trace d'exécution permet de bien distinguer les 3 phases du déroulement d'une élection : la phase d'$Estimation$, la phase de $Proposition$ et la phase de $Decision$.

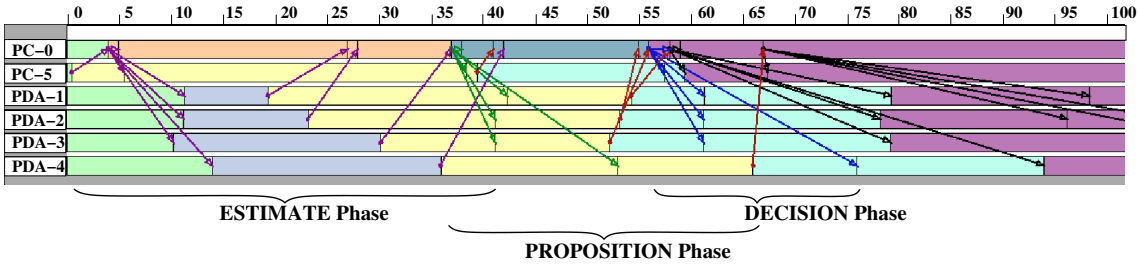


Figure 14.1 *Diagramme post-mortem d'une élection ne donnant lieu à aucune suspicion*

Sur ce diagramme, chaque ligne correspond à une entité participant à l'algorithme et permet d'illustrer leurs changements d'états survenus au cours de l'élection. De plus, les communications entre les différentes entités sont représentées par des flèches allant de l'émetteur vers le ou les récepteur(s). Il est important de constater que la diffusion d'un message est représentée graphiquement par plusieurs flèches ayant la même origine, mais il est bien évident qu'en réalité un seul message est réellement émis sur les ondes.

Il est à noter que l'élection représentée par le diagramme 14.1 a été réalisée en un round seulement, round durant lequel le coordinateur était l'entité PC-0. Ceci est bien visible sur le diagramme puisque les groupes de messages (diffusions) sont en partance de l'entité PC-0 et que toutes les autres entités émettent vers cette entité PC-0.

De plus, dans cette trace, la représentation d'une diffusion de messages utilisée met bien en évidence les réceptions différées suivant les sites. Ce délai inclut d'une part les transmissions réseau ainsi que la traversée des différentes couches, du niveau UDP/IP à l'application Java (remontée de la pile).

Mais, détaillons désormais les différentes étapes du protocole illustrées dans cette trace d'exécution :

- L'initiation du consensus est effectuée par un des participants qui envoie sa valeur d'$Estimation$. Dans cette élection, l'entité PC-5, la première à s'engager dans ce consensus, déclenche l'implication de l'entité PC-0 qui est le coordinateur de ce round. Le coordinateur diffuse alors sa valeur d'estimation ce qui déclenche l'engagement dans l'algorithme de tous les autres participants.
- Suite à cette diffusion, le coordinateur est en attente des estimations d'une majorité de participants. Aussi, dès que le coordinateur a obtenu une majorité d'estimations, il génère une proposition qu'il diffuse aussitôt.
- Notons que la valeur d'estimation de l'entité PDA-4 n'est pas utilisée pour le calcul de la proposition car elle parvient plus tard au coordinateur. Mais, ceci n'empêche pas cette entité de poursuivre sa participation à l'obtention d'une décision commune. Cependant, une majorité des participants a déjà acquitté la proposition, aussi le coordinateur peut dès lors diffuser la décision.
- Ensuite, lorsque le coordinateur reçoit les messages d'acquittement des entités PDA-1 et PDA-4, il diffuse de nouveau la décision (message de type Forward).

14.3.1.2 Avec une suspicion

La trace d'exécution 14.2 montre le déroulement d'une élection pendant laquelle un des participants suspecte le coordinateur.

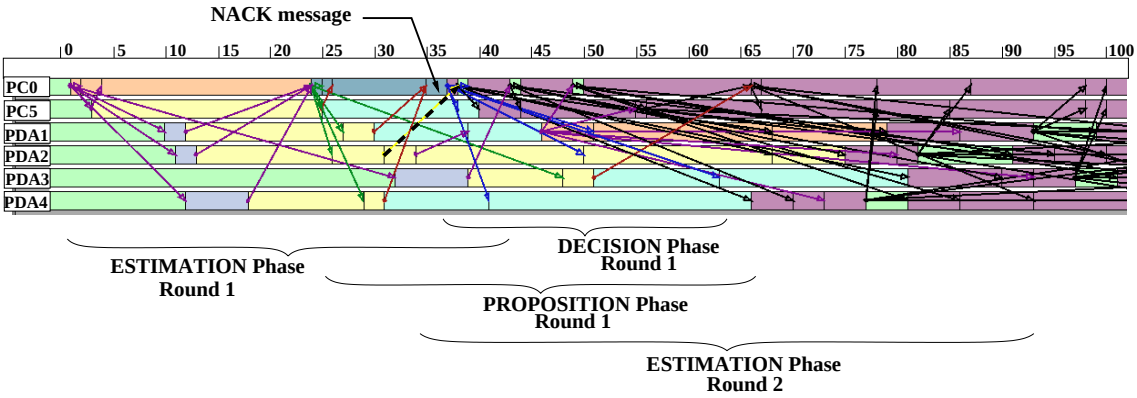


Figure 14.2 *Diagramme post-mortem d'une élection pendant laquelle une fausse suspicion a lieu*

Comme on peut le voir sur le diagramme *post-mortem* 14.2, lors de cette exécution c'est le coordinateur du premier round (PC-0) qui initie le consensus par la diffusion de sa valeur d'estimation. A ce message chaque participant répond par l'envoi de leur propre valeur d'$Estimation$. Dès lors que le coordinateur a reçu une majorité de valeurs d'$Estimation$, il réalise une $Proposition$ qu'il diffuse afin de la soumettre à l'approbation des autres participants. Pendant ce temps, l'entité PDA-2 vient à suspecter l'absence du coordinateur actuel (PC-0), et envoie donc un non-acquittement (message $Nack$) en réponse à la $Proposition$ faite par le coordinateur, puis passe

au round suivant.

Comme le montre la trace d'exécution, cette suspicion peut être qualifiée de "fausse suspicion" car le coordinateur de ce round (PC-0) n'est pas réellement absent. Dans ce cas de figure l'ordre des événements a une grande importance. En effet, avant de réceptionner ce non-acquittement (message $Nack$), le coordinateur a préalablement reçu une majorité d'acquittements positifs (messages de type Ack) et par suite diffuser la $Decision$. Aussi, en réponse à ce message $Nack$, le coordinateur (PC-0) diffuse simplement un nouveau message contenant sa valeur de $Decision$.

Par ailleurs, la suspicion du coordinateur courant par l'entité PDA-2 a pour conséquence immédiate le passage au round suivant de cette entité ainsi que l'envoi de sa valeur d'$Estimation$ au coordinateur de ce nouveau round (PDA-1). Or, à la réception de cette information, l'entité PDA-1 (coordinateur du second round) n'ayant pas encore reçu de message de $Decision$ émis dans le round courant (le premier round), elle s'engage elle aussi dans le round suivant et diffuse donc son $Estimation$ à tous les participants.

A ce moment là, la valeur d'$Estimation$ diffusée est la valeur proposée lors du round précédent ce qui permet de respecter la propriété de convergence vers une décision unique. Finalement, l'entité PDA-1 finit par recevoir la valeur de $Decision$ retenue durant le round précédent et adopte à son tour cette décision.

Aussi, on peut constater la désynchronisation des participants puisque ceux-ci sont dans des états différents (rounds différents, phases différentes). Cependant, lorsqu'un participant connaît la valeur de $Decision$, à chaque réception d'un message estampillé par l'identifiant du consensus il répond par diffusion de cette valeur de décision (message $FORWARD$). Ainsi, la fin de la trace 14.2 montre que tous les participants finissent par connaître la valeur de décision.

Une optimisation de la diffusion agressive de la décision (message $Forward$) permettrait sans doute d'alléger la quantité de messages émis tout à préservant la garantie de terminaison de l'algorithme.

14.3.2 Qualité de service

Ces diagrammes sont utilisés afin d'observer et d'analyser le comportement de l'algorithme dans deux scénarios particuliers. La première étude consiste à examiner le comportement dans un scénario où un ensemble fixe d'entités sans fil participant au consensus lesquelles ne subissent qu'un minimum de perturbations (mouvement ou arrêt) afin d'éviter les déconnexions. Le but de ce scénario est d'évaluer l'impact des communications sans fil sur l'algorithme et d'obtenir ainsi des mesures de performances. La seconde étude, quant à elle, porte sur un scénario où des isolations réseau peuvent avoir lieu. L'objectif est alors de valider les modifications apportées à l'algorithme de Chandra & Toueg afin s'adapter au contexte considéré.

14.3.2.1 QoS au niveau système : utilisation des ressources

Théoriquement, dans un environnement idéal où aucune perturbation n'intervient (pas de déconnexions, pas de défaillances, ...), une élection est réalisée en un seul round. Dans ce cas, le coordinateur diffuse seulement trois messages ($Estimation$, $Proposition$ et $Decision$), et chaque participant n'envoie à ce même coordinateur que deux messages ($Estimation$ et Acquittement (Ack)). Dans notre contexte expérimental où 6 entités sont mises en jeu pour la réalisation d'une

élection, ceci signifie que, au minimum, treize messages doivent transiter afin de pouvoir effectuer une élection.

On rappelle que chaque expérimentation consiste en une série de 1000 élections consécutives. Le tableau 14.2 donne le nombre moyen de messages émis pour la réalisation de ces élections lorsqu'aucune perturbation n'est constatée.

Exp1 [1]	**PC-0**	**PDA-1**	**PDA-2**	**PDA-3**	**PDA-4**	**PC-5**	Total
Broadcast	6.4	0	0	0	0	0	6.4
Unicast	0	2	2	2	2	2	10
Total	6.4	2	2	2	2	2	16.4

TAB. 14.2 *Nombre moyen de messages par entité pour une élection*

Comme il n'y a ni défaillance ni suspicion de défaillance, toutes les élections sont effectuées lors du premier round dont PC-0 est le coordinateur. Ceci est dû au fait que la même liste donnant l'ordre des coordinateurs successifs est conservée entre les différentes élections.

Comme annoncé théoriquement, on constate bien que le nombre moyen de messages émis par chacun des participants est en moyenne de deux par élection. D'autre part, le nombre moyen de diffusion de messages par le coordinateur (PC-0) est approximativement de 6,4 ce qui correspond à deux fois plus que l'attente théorique. Les messages supplémentaires émis par le coordinateur sont essentiellement dûs à l'envoi de messages de type $Forward$, ces messages n'intervenant que pour assurer que chaque participant prenne connaissance de la décision le plus rapidement possible. Cependant, certains messages supplémentaires émis par le coordinateur tout comme ceux émis par les participants s'expliquent par l'existence du mécanisme de retransmission actionné chaque fois que l'attente d'un événement futur (côté participants comme côté coordinateur) devient trop longue.

Le tableau 14.3 donne la répartition des messages émis en fonction du type de messages.

Exp1 [1]	**Estimation**	**Proposition**	**Ack**	**Nack**	**Decision**	**Forward**	Total
Broadcast	1	1	-	-	1	3.4	6.4
Unicast	5	-	5	0	-	-	10
Total	6	1	5	0	1	3.4	16.4

TAB. 14.3 *Nombre moyen de messages émis par type de messages et par élection*

Le principe de la diffusion systématique de messages $Forward$ paraît quelque peu coûteux, aussi une amélioration devrait être prochainement envisagée.

14.3.2.2 QoS au niveau applicatif : latence

Du point de vue de l'application, seule la durée nécessaire à réalisation d'une élection est importante. La durée d'une élection correspond au temps écoulé entre l'initiation de l'élection et

[1] L'erreur statistique calculée est inférieure à 1 %

l'obtention de la décision, mais elle peut être mesurée de plusieurs façons différentes :

1. Temps écoulé entre l'initiation de l'élection par une des entités et le moment où cette entité initiatrice connaît le résultat de cette élection.
2. Temps écoulé entre l'initiation de l'élection et le moment où l'élu obtient la décision.
3. Temps écoulé entre l'initiation et le moment où toutes les entités participant à l'élection ont connaissance de la décision.

En effet, l'intérêt de ces différentes mesures est donné par le fait que, suivant les cas ("causes" de l'élection), le système sera "débloqué" dès lors qu'une entité particulière obtiendra la décision (l'élu lui-même ou l'initiateur de l'élection) ou bien dès que tous les participants auront l'information.

Exp 1	$\bar{x}$ [1]	min	max	σ	Q_1	Q_2	Q_3
1 : initiateur	71	25	2063	190	39	45	52
2 : élu	78	26	2065	191	42	48	57
3 : tous	224	37	2070	231	136	199	254

TAB. 14.4 *Distribution des différentes durées d'une élection (ms)*

Le tableau 14.4 résume la distribution de ces différentes durées ; les statistiques ont été calculées sur une série de 1000 élections dans le cadre d'une expérimentation sans perturbation. Les différentes colonnes du tableau représentent dans l'ordre : la durée moyenne, la durée minimale, la durée maximale, l'écart type, le premier quartile, la médiane et le troisième quartile.

Cependant, il est important de noter que d'un point de vue théorique, l'initialisation de l'élection est déclenchée par n'importe quel membre du groupe. Or, au cours de cette expérience, on a pu constater que l'initiateur d'une élection correspond à la première entité ayant obtenu la décision de l'élection précédente dans la série d'élections réalisée, et ce malgré l'absence de synchronisation entre les différentes élections au niveau applicatif.

D'autre part, l'hétérogénéité des entités semble avoir un fort impact puisque les auteurs de l'initialisation sont en grande majorité des entités plus puissantes (PCs).

Cette constatation mise à part, on peut noter que la première durée (temps écoulé entre l'initiation de l'élection et la réception de la décision par l'entité initiatrice) représente la durée minimale de résolution d'une élection dans notre environnement. D'un point de vue général, on peut constater que les deux premières durées (temps écoulé entre l'initialisation de l'élection et la réception de la décision pour l'initiateur ou pour l'élu) montrent des résultats très proches, alors que la troisième durée (temps écoulé entre l'initialisation et le moment où tous les participants connaissent la décision) est en moyenne bien plus grande. De plus, si l'on regarde la répartition de ces durées, il apparaît que quelque soit le type de durée, les 75 % des valeurs obtenues sont relativement faibles (environ 71 ms pour la première durée, 78 ms pour la seconde durée et 224 ms pour la troisième durée). Cependant, il apparaît qu'une quantité non négligeable d'élections (environ 1%) se réalisent en une durée beaucoup plus longue : de l'ordre de 2 secondes. Ceci peut s'expliquer en partie par la non fiabilité des communications et donc par la nécessité des ré-émissions périodiques. Mais une étude plus approfondie sera nécessaire à l'étude des causes de ces phénomènes.

Pour pouvoir analyser les durées de réalisation des élections, les temps passés à attendre une majorité de réponses (attente des estimations et des acquittements) côté coordinateur ont été isolés 14.5. La première durée mesurée correspond au temps écoulé entre la diffusion de l'estimation du coordinateur et la réception du message d'estimation permettant au coordinateur d'avoir une majorité de réponse. Il en est de même pour le temps écoulé entre la phase de proposition et celle de réception des acquittements.

Exp 1	$\bar{x}$ [1]	min	max	σ	Q_1	Q_2	Q_3
Estimations	37	1	2021	143	18	21	27
Acquittements	26	7	161	20	17	21	26

TAB. 14.5 *Distribution du temps d'attente d'une majorité de réponses pour le coordinateur (ms)*

Le tableau 14.5 montre que pour le coordinateur, le temps moyen passé à attendre une majorité est relativement important. Si l'on considère que le coordinateur correspond souvent à l'entité initiatrice de l'élection, cette durée moyenne représente plus de 80 % du temps moyen nécessaire à la réalisation d'une élection. Il semble donc que ces phases d'estimation et d'attente d'acquittements permettent une certaine synchronisation des entités participant à l'élection.

14.3.3 L'impact d'une déconnexion

Tout d'abord, la déconnexion imprévisible d'une entité participante est considérée. Bien sûr, si l'entité se déconnectant brutalement n'est pas coordinatrice du round (ou des rounds) en cours, sa déconnexion n'aura pas de véritable impact sur la résolution de l'élection, si ce n'est sur la durée qui peut éventuellement être augmentée (mais seulement dans le cas où cette entité correspond à une des machines les plus puissantes mises en jeu). Aussi, l'étude menée a porté sur la déconnexion de l'entité coordinatrice du premier round (dans la liste initiale des participants) et son impact sur les performances.

Lorsque le premier coordinateur est absent, et si aucune autre perturbation n'intervient au cours de l'expérience, toutes les élections sont réalisées en deux rounds. Ceci est illustré dans la trace d'exécution 14.3.

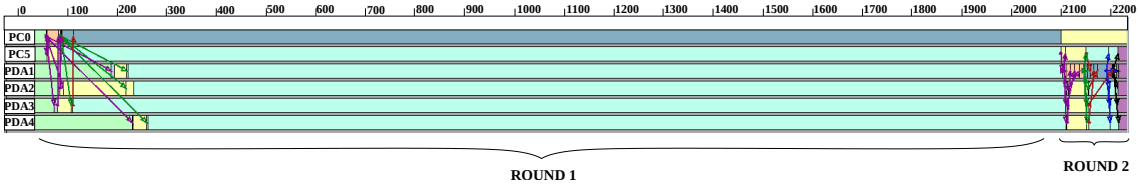


Figure 14.3 *Trace d'exécution post-mortem d'une élection où le coordinateur du premier round (PC-0) se déconnecte (ou subit une défaillance)*

Comme on peut le voir au début de la trace 14.3 (figure 14.4), tous les participants sont présents. Cependant PC-0, le premier coordinateur, se déconnecte (ou subit une défaillance) durant

[1] L'erreur statistique est inférieure à 1 %

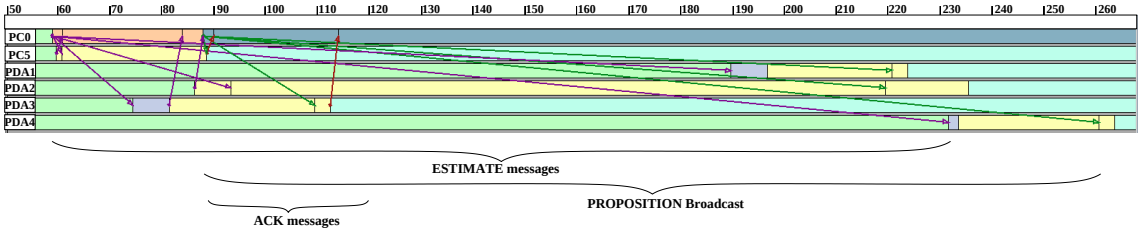


Figure 14.4 *Zoom sur le début du round 1 dans la trace 14.3*

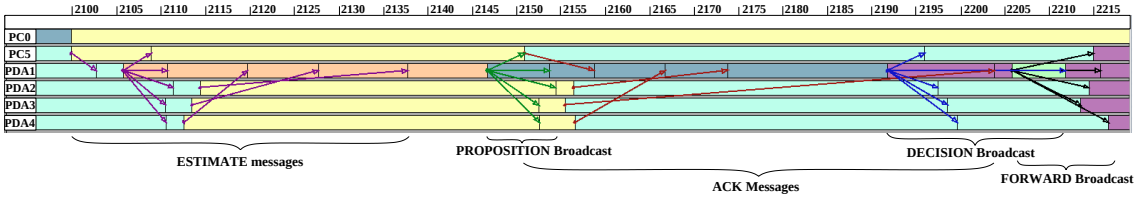


Figure 14.5 *Décision lorsque le premier coordinateur est absent (fin de la trace 14.3)*

le premier round. Cette déconnexion (ou défaillance) peut être qualifiée de déconnexion définitive (ou défaillance définitive) par rapport à l'élection en cours. Après un délai dépendant des valeurs attribuées aux paramètres des détecteurs de défaillances, les entités participantes finissent par passer au round suivant (fin de la trace 14.3). L'illustration du second round où l'entité PDA-1 est coordinatrice est donnée dans la trace d'exécution *post-mortem* 14.5.

Exp2	**PC-0**	**PDA-1**	**PDA-2**	**PDA-3**	**PDA-4**	**PC-5**	Total
Broadcast	-	4	0	0	0	0	4
Unicast	-	1	3	3	3	3	13
Total	-	5	3	3	3	3	17

TAB. 14.6 *Nombre moyen de messages par entité pour une élection (PC-0 absent)*

En comparaison avec 14.2, lorsque le coordinateur du premier round est absent, en moyenne chaque participant envoie seulement un message point à point supplémentaire (un message d'$Estimation$). Ceci s'explique par le déroulement de l'élection en deux rounds successifs : au cours de chaque round, chaque participant connecté envoie sa valeur d'$Estimation$ au coordinateur du round en cours. Aussi, comme l'interrogation des détecteurs de défaillances n'est effectuée qu'après envoie d'un message de type $Estimation$, les participants envoient tout d'abord leur valeur d'$Estimation$ à l'entité PC-0 (coordinateur du premier round) avant de se rendre compte de son absence et par conséquent de passer au round suivant. Ce principe permet d'éviter autant que possible un passage au round suivant en cas de fausse suspicion. Rappelons cependant qu'un participant passe directement au round suivant en cas de réception d'un message en provenance d'un autre participant déjà dans le round suivant.

Par ailleurs, comme l'illustre le tableau 14.7, la répartition par type de messages en transit est quelque peu modifiée.

Exp1	**Estimation**	**Proposition**	**Ack**	**Nack**	**Decision**	**Forward**	Total
Broadcast	1	1	-	-	1	1	4
Unicast	9	-	4	0	-	-	13
Total	10	1	4	0	1	1	17

TAB. 14.7 *Nombre moyen de messages émis par type de messages et par élection (PC-0 absent)*

Le tableau 14.7 montre par ailleurs que le nombre de messages diffusés ($Estimation$ et $Proposition$) est similaire au tableau 14.3, seul le nombre de messages de type $Forward$ est en nette diminution. Ceci s'explique par le rapport existant entre le nombre d'entités présentes dans le système et le nombre d'entités nécessaire pour avoir la majorité. En effet, lorsque toutes les entités sont présentes ($Nb_{tot} = 6$), la prise de décision s'effectue entre une majorité de participants (à savoir au moins 4 entités). Or, lorsqu'une entité est défaillante, le nombre d'entités présentes est alors de $Nb_{tot} - 1$ (à savoir 5 entités présentes) mais la majorité de participants reste de 4. Aussi, lorsque toutes les entités sont présentes, seuls deux participants sont susceptibles d'être désynchronisés et donc être à l'origine d'au moins 2 messages de type $Forward$, alors que seulement 5 entités sont réellement présentes, un seul participant peut éventuellement être désynchronisé et donc être à l'origine d'1 message de type $Forward$.

De plus, comme supposé préalablement, la durée moyenne d'une élection est alors quelque peu augmentée. Cependant, cette augmentation peut certainement être restreinte grâce à une meilleure configuration des paramètres associés aux détecteurs de défaillances.

14.3.4 Le problème de la majorité

Cette section permet d'illustrer le comportement de l'algorithme lorsque au moins la moitié des entités participant à l'élection est momentanément déconnectée. En effet, comme nous l'avons abordé au cours du chapitre 11, l'environnement fortement instable dans lequel nous nous plaçons ne permet pas de garantir la présence à tout instant d'une majorité d'entités dans le système. La trace d'exécution exposée dans la figure 14.6, bien que peu visible, permet de valider le fonctionnement correct et d'analyser le déroulement de l'algorithme proposé en l'absence d'une condition nécessaire à la prise de décision. Dans cette figure, les triangles représentent un round : la taille d'un triangle est proportionnelle à la durée du round auquel il correspond et à chaque couleur de triangle est associé un round différent. Bien que peu visible sur cette trace, les flèches représentent toujours les messages échangés. Les flèches quasi-verticales pouvant être distinguées plus nettement, représentent en fait un groupe d'événements au cours duquel plusieurs messages sont échangés.

Directement dû à l'absence d'un majorité d'entités dans le système, le grand nombre d'événements successifs du début de la trace d'exécution s'explique par le fait que chacun des participants présents envoient leur estimation au différents coordinateurs successifs en avançant ainsi de round en round. La fin de cette trace d'exécution illustre le retour à une situation bien plus stable dans laquelle une majorité de participants est de nouveau obtenue (reconnexion de l'entité PDA-2), et correspond donc au déroulement "normal" de l'élection.

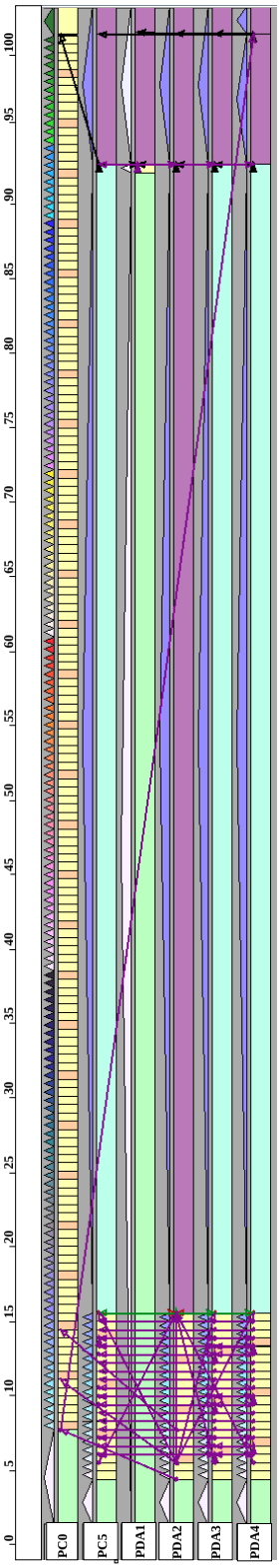


Figure 14.6 *Trace d'un consensus devant attendre la présence d'une majorité de participants*

Globalement, cette trace d'exécution réalisée *post mortem* permet de bien comprendre l'évolution de l'état des entités présentes en attendant le retour de certaines entités déconnectées momentanément, conduisant ainsi au retour d'une majorité de présence dans le système. En effet, les entités connectées poursuivent l'élection en cours en "augmentant les rounds" progressivement jusqu'à la reconnexion d'une quantité suffisante d'entités. Cette attente pouvant être arbitrairement longue, la durée d'une élection en environnement très instable ne peut être définie et dépend donc directement des conditions d'instabilité. Cependant, quelque soit ce délai d'attente, le retour de la dernière entité permettant d'accéder à une majorité de présences, déclenche automatiquement le déroulement de la fin de l'algorithme et ainsi la prise de décision.

14.4 Conclusion

Les expérimentations réalisées et présentées dans ce chapitre étaient essentiellement destinées à la validation de l'algorithme de consensus proposé. Aussi, bien que des expérimentations aient été effectuées avec différents paramétrages, d'autres campagnes d'expérimentations devront être menées afin de compléter cette étude. En effet, une analyse plus approfondie des performances de cet algorithme de consensus sera nécessaire afin de définir la qualité d'une élection en fonction des différents paramètres (paramètres associés aux détecteurs de défaillances, et paramètres de réémission au niveau du module de consensus). Pour ce faire, et afin de considérer les diverses perturbations pouvant affecter le déroulement de l'algorithme de consensus, une approche méthodologique complète doit être définie et réalisée. La complexité de cette approche réside principalement dans le grand nombre de facteurs ayant une influence sur l'algorithme (nombre d'entités dans le système, type d'entités mises en jeu, l'ordre des coordinateurs, etc...) ainsi qu'en la forte instabilité ambiante de l'environnement considéré (impact des conditions extérieures).

Conclusion 15

Objectifs de la thèse :

De manière générale, l'objectif principal de cette thèse résidait dans l'étude et la mise au point d'outils permettant d'assurer la cohésion d'un groupe d'entités, offrant différents services et souhaitant les partager, dans des environnements faiblement connectés et soumis à une forte variabilité. Les environnements ainsi concernés sont des environnements ad-hoc intégrant un ou plusieurs protocoles de communication sans fil et mettant en oeuvre des "objets communicants" hétérogènes.. Typiquement, les scénarios caractéristiques visés correspondent aux situations nécessitant une mise en réseau spontanée telles que les réunions de travail, conférences, rencontres sportives, etc... Aussi, dans ce contexte, la problématique abordée dans nos travaux s'attache à la fois à caractériser la dynamicité de l'environnement totalement distribué considéré et à résoudre les problèmes de cohésion de groupe dans un tel environnement.

Démarche proposée :

Face au grand nombre de nouvelles problématiques introduites par l'utilisation de technologies sans fil, nous avons concentré nos travaux sur l'étude des problèmes liés à la prise de décision dans un environnement aux caractéristiques particulières. Le cadre dans lequel nous nous sommes placé considère les réseaux ad-hoc sans fil composés d'entités hétérogènes. Mais ce cadre est intrinsèquement restreint par la formulation de diverses conditions simplificatrices telles qu'une limitation sur le nombre d'entités composant le système distribué, l'accès direct des entités les unes par rapport aux autres (pas de mécanisme de routage), etc...

Dans ce contexte, une première approche visait à déterminer précisément les conditions environnementales pouvant être escomptées dans les systèmes distribués particuliers envisagés. Aussi, dans cette optique, une analyse comportementale de l'environnement considéré à été menée. Bien que réalisée dans un environnement expérimental très particulier, la méthodologie d'approche ainsi

proposée peut facilement être adaptée à l'étude d'autres environnements susceptibles de présentés des comportements particuliers. Cette approche expérimentale a permis d'une part d'identifier et de qualifier plus précisément l'instabilité ambiante de l'environnement considéré, et d'autre part de quantifier quelque peu la forte variabilité de cet environnement.

Par ailleurs, la prise en considération de cette instabilité générale de l'environnement dans notre approche algorithmique, nous a conduit à nous intéresser principalement aux algorithmes de consensus s'appuyant sur des mécanismes de détections de défaillances. De ce fait, face aux contraintes spécifiques imposées par un contexte environnemental fortement sensible aux perturbations extérieures, un algorithme de consensus particulièrement adapté à la dynamicité ambiante a été développé. Parallèlement, une étude approfondie de la qualité de service (compromis entre la réactivité et la fiabilité du détecteur de défaillances) associée aux détecteurs de défaillances mis en place, a été réalisée en fonction du paramétrage choisi et prenant en compte la variabilité ambiante. Globalement, les outils algorithmiques ainsi proposés et développés offrent une certaine "robustesse" face à la variabilité de l'environnement.

Réalisations :

Un prototype mettant en oeuvre les principes algorithmiques retenus a été implémenté en Java. L'architecture ainsi mise en place se décompose en deux modules principaux : un module de service de consensus et un module de détections de défaillances (exportation des informations locales et importation d'informations distantes). Dans le cadre du projet exploratoire Sidrah, cette première implémentation a également été intégrée à la plateforme logicielle Sidrah développée (démonstrateur).

Afin d'étudier le paramétrage délicat des détecteurs de défaillances, le module de détections de défaillances a fait l'objet de nombreuses évaluations. Ces expérimentations ont permis de mettre en évidence toute la complexité liée à l'obtention localement d'informations relatant du le système global ; complexité induite principalement par les difficultés du réglage des temporisations. Quant au module de service de consensus, les expérimentations réalisées visaient essentiellement à la validation de l'approche algorithmique proposée, et une évaluation plus approfondie des performances de l'algorithme développé devra être menée.

Perspectives :

Bien évidemment, le travail réalisé reste exploratoire et n'est donc pas complet. Non seulement les travaux déjà effectués gagneraient à être complétés, mais les nombreuses problématiques nouvelles soulevées au cours de la réalisation de cette thèse mériteraient d'être étudiées à part entière.

Analyse environnementale :

L'approche originale consistant en une analyse expérimentale des conditions environnementales est une approche intéressante dans le sens où elle permet d'obtenir de meilleures connaissances des spécificités environnementales et ainsi de bénéficier de cet atout au cours des réalisations ultérieures. Cependant, la méthodologie proposée reste relativement succinte et une étude complémentaire (prise en compte d'autres facteurs influents, raffinement par rapport aux différents niveaux des facteurs) voire des améliorations ne seraient qu'avantageuses. Par ailleurs, comme nous l'avons préalablement précisé, cette méthodologie d'analyse comportementale factorielle peut facilement être reproduite sur divers environnements (réseaux Bluetooth, réseaux GSM, réseaux de capteurs) modulo quelques adaptations au niveau contextuel.

Algorithmique répartie :

Concernant le domaine de l'algorithmique distribuée tolérante aux défaillances, les travaux exposés dans ce document, ont été focalisés sur l'étude de l'algorithme de consensus basé sur l'utilisation d'un mécanisme de détections de défaillances. De la même manière que pour l'algorithme de consensus, beaucoup d'autres approches algorithmiques (algorithmes permettant la gestion du groupe, la continuité de service ou la gestion des ressources) peuvent être adaptées de façon à prendre en compte voire supporter les caractéristiques spécifiques de l'environnement considéré. Aussi, par le biais d'une démarche plus ou moins similaire à celle proposée pour l'algorithme de consensus, de nombreux autres algorithmes pourraient être développés et ainsi venir enrichir l'architecture modulaire implantée et notamment présentée dans le cadre du projet Sidrah.

Qualité de service :

Une part importante de ces travaux de recherches a été consacrée à l'étude du paramétrage délicat des détecteurs de défaillances. En effet, la réalisation d'une estimation locale de l'état global du système plus précise nécessite la mise au point de modèles mathématiques plus raffinés de façon à améliorer la capacité d'adaptation des détecteurs face à la variabilité de l'environnement. Par ailleurs, la qualité de service associée aux détecteurs de défaillances pourrait sans doute être améliorée davantage en introduisant un mécanisme permettant l'auto-adaptation du paramétrage des détecteurs en fonction de l'évolution des conditions environnementales. Parallèlement, il serait intéressant d'étudier l'impact sur la qualité de l'algorithme de consensus proposé de la mise en place de détecteurs de défaillances qualifiés d'actifs, lesquels informeraient le module de consensus à chaque modification de leur vision des entités distantes.

De manière plus générale, ces travaux ont été développés autour d'une problématique particulière ne considérant qu'un nombre restreint d'entités mises en jeu dans les environnements considérés. Aussi, la question est de savoir si le passage à l'échelle est envisageable : les solutions proposées peuvent-elles être adaptées à un système de taille plus importante ?

De plus, l'environnement expérimental considéré peut également être complexifié par l'introduction d'un plus grand panel d'entités hétérogènes (diversification des entités au niveau de leurs caractéristiques matérielles), mais aussi par la multiplication des technologies sans fil offertes dans le système (permettre la communication entre des entités utilisant différents protocoles de communication par l'intermédiaire d'entités ayant à disposition plusieurs technologies de communication sans fil).

Par ailleurs, cette étude n'a considéré seulement les systèmes distribués dans lesquels chaque entité le composant est directement atteignable par les autres et peut en un seul saut atteindre toutes les autres entités du système. Aussi, une des principales perspectives à ces travaux de recherches concerne donc la prise en compte de systèmes beaucoup plus complexes et notamment les environnements dynamiques dans lesquels les communications entre les entités le composant nécessitent l'intervention de mécanismes de routage.

IV

Annexes

Projets Industriels et Scientifiques A

A.1 Projet SIDRAH

Le projet RNRT Sidrah [33], projet exploratoire, a été organisé en quatre thématiques différentes chacune correspondant à un sous-projet :

Le sous-projet Réseau :

Ce sous-projet consiste d'une part en une étude expérimentale du comportement du protocole de communication WIFI en termes de débits et de robustesse aux déconnexions. D'autre part, ce sous-projet a porté sur l'étude de l'extension de la bulle d'interaction Sidrah. Aussi, la modification de l'ORB Jonathan a été réalisée afin d'améliorer les capacités de communication. La solution retenue repose sur l'encapsulation des requêtes IIOP dans des requêtes HTTP, type de liaison nommé HIOP (GIOP/HTTP).

Le sous-projet Services :

Ce sous-projet définit l'utilisation d'une architecture OSGI permettant de réaliser des services complexes par composition de services élémentaires. Tout objet de l'environnement (bulle Sidrah) ayant la possibilité d'offrir un ou plusieurs services et aussi la liberté d'utiliser les autres services, l'architecture asymétrique de type client/serveur est donc évitée.

Le sous-projet Résilience :

Cette thématique concerne l'étude d'algorithmes répartis capables de gérer les aléas rencontrés dans un réseau radio ayant des performances inférieures à celles d'un réseau filaire :

temps de latence variables, débits non homogènes, variations amplifiées par l'hétérogénéité des composants. Deux modules ont été développés : un module de détection de défaillances, et un module permettant la réalisation d'un consensus. Intégrés à une infrastructure de services répartis, ces deux modules permettent d'assurer la fiabilité de fonctionnement dans un "réseau faiblement connecté".

Le sous-projet Démonstrateur :

Cette thématique correspond à la réalisation d'une bulle Sidrah sur un ensemble de machines. Le démonstrateur ainsi développé permet de mettre en évidence le fonctionnement des détecteurs de défaillances, la réalisation d'un consensus, l'usage d'un nouveau protocole de communication (HIOP) et enfin d'illustrer la composition de services sur des équipements variés.

Une bulle Sidrah ainsi définie et créée se compose d'un ensemble de machines hétérogènes susceptibles d'interagir de manière sporadique au travers de différents réseaux non permanents (Bluetooth et/ou WIFI). Chacune des machines de la bulle Sidrah propose un certain nombre de services à l'ensemble de la communauté et peut en contre-partie bénéficier d'autres services distants.

Les informations techniques nécessaires à la mise en place d'une bulle Sidrah sont regroupées dans une capacité d'amorçage à disponibilité des participants. Dans le cadre d'une réunion, d'un séminaire, d'un salon, ou encore d'une compétition sportive, cette capacité pourra être préalablement distribuée à chacun des participants. Suite à l'installation de cette capacité, aucune configuration supplémentaire ne sera nécessaire pour bénéficier des avantages du système Sidrah.

Les services Sidrah sont déclarés auprès du système Sidrah de manière unique, au lancement, quel que soit l'état de disponibilité des différents réseaux. La plateforme Sidrah se charge dès lors de rendre les services en question disponibles sur les nouveaux réseaux découverts au fur et à mesure qu'ils deviennent accessibles. De même, la plateforme permet aux différents applicatifs, services compris, d'être notifiés de manière asynchrone de l'apparition de nouveaux services au sein de la bulle, comme de l'accessibilté/inaccessibilité de sous-réseaux déclarés.

Les appels aux services distants sont véhiculés par l'intermédiaire de l'ORB Jonathan, utilisé dans sa personnalité CORBA. L'interaction des différents participants peut nécessité le verrouillage de certaines ressources en exclusivité pour un seul utilisateur (vidéo-projecteur par exemple) du groupe. Le réseau étant sporadique, il est possible qu'un utilisateur ayant obtenu l'accès exclusif à une ressource se trouve déconnecté de manière brutale. Dans ce type de situation, il est nécessaire que l'entité fautive soit détectée, et que la ressource verrouillée soit libérée. Rien ne garantissant que toutes les entités possèdent une vision cohérente de l'ensemble des participants accessibles et en bon état de fonctionnement, la plateforme Sidrah offre un service de présomption de défaillances basé sur les détecteurs de défaillances.

A.2 Projet DECORE

Dans le projet académique DECORE, différentes thématiques de recherche ont été définies et ont fait l'objet de travaux spécifiques en relation avec la problématique générale du projet :

Modélisation et performance de réseaux sans fil :

L'objectif de cette thématique consiste en la construction de modèles de comportement des réseaux sans fil en utilisant diverses approches : Réseaux d'Automates Stochastiques, Réseaux de Files d'Attente, Modèles Fluides, ...
Cette modélisation est guidée par des expérimentations effectuées sur les plates-formes de réseaux ambiants mises en place.

Exploitation des symétries dans les modèles des réseaux :

La construction de protocoles de communication repose sur le fait que les sites exécutent les mêmes codes, avec éventuellement des paramètres différents. L'objectif de cette thématique est donc de développer des méthodes permettant d'intégrer les hypothèses de symétries dans les modèles et les méthodes de résolution afin de réduire l'espace d'état et ainsi de permettre une résolution numérique des modèles.

Simulation de configurations représentatives :

Plusieurs approches sont possibles pour analyser le comportement de modèles Markoviens de réseaux. Une technique consiste à générer des configurations "typiques" du modèle. Par "typique" nous entendons que le système est en régime stationnaire et que l'on échantillonne selon la distribution correspondante.
Dans un premier temps la construction des algorithmes de simulation directe selon la loi stationnaire sans passer par la simulation du processus sur une longue période est envisagée. Ces algorithmes, utilisés en physique statistique ("Perfect Simulation", Propp & Wilson 1996) n'ont jamais été adapté dans le contexte de l'évaluation de performances de réseaux. Aussi, une comparaison de ces algorithmes avec les techniques "plus traditionnelles" de simulation comme NS par exemple peut être réalisée.
Par la suite, cette technique de simulation dite "parfaite" ayant été implémentées au sein du logiciel PSI (Perfect SImulator), elle pourra être comparée aux techniques numériques telles que les méthodes analytiques et approximatives.

Méthodes analytiques approximatives :

La grande complexité des systèmes de communication exprimée en termes de nombre d'entités communicantes et de nombre de clients servis par le réseau rend très difficile la réso-

lution des modèles complets du réseau. On peut facilement montrer que la taille du modèle est exponentielle dans le nombre des serveurs du système et linéaire en la taille des buffers. L'utilisation des modèles fluides peut résoudre le problème lié au nombre des clients dans le système. Une décomposition du système en sous-systèmes élémentaires permet d'obtenir une solution approximative qui contourne la complexité liée au nombre de serveurs du modèle initial. Une étude plus détaillée des modèles de flux permettra d'améliorer les méthodes analytiques afin de prendre en compte des modèles de trafic très variés.

Commande robuste pour l'allocation de ressources :

L'objectif de cette thématique concerne l'étude de la commande déterministe des systèmes à événements discrets sous perturbations stochastiques et, ensuite, la synthèse de la commande robuste. La problématique abordée est bien connue dans les systèmes temps réels.

Méthodologie : encombrement et pertes dans les réseaux (niveau paquet) :

Les travaux menés dans cette thématique ont dans un premier temps portés sur la réalisation d'une plate-forme expérimentale spécifique afin de permettre ensuite le développement d'un système de mesure du trafic et de mesure des pertes sur le réseau.

Le standard 802.11 B

B.1 Généralités

Apparue en 1997, la norme 802.11 est un standard de l'IEEE [5] permettant la description des caractéristiques des réseaux locaux sans fil (WLAN). Ce standard se décline désormais en plusieurs normes (802.11a, 802.11b, ...) ayant des caractéristiques spécifiques.

B.1.1 Présentation

Comme tout 802.x, la norme 802.11 définit les méthodes d'accès et de contrôle des réseaux locaux mais spécifique aux réseaux sans fil. Cette norme correspond aux couches physique (couche PHY) et de liaison de données du modèle OSI, mais elle divise la couche de liaison de données en deux sous-couches : la couche MAC[1] et la couche LLC[2]. Alors que la couche physique définit le mode de transmission des signaux, la couche de liaison de données gère l'accès au support, l'adressage des paquets, le formatage des trames, la fragmentation et le réassemblage des trames, ... Ainsi, en plus des fonctions habituellement rendues par la couche MAC, la couche MAC 802.11 offre d'autres fonctions qui sont normalement confiées aux protocoles supérieurs, comme la fragmentation, les retransmissions de paquets et les accusés de réception. En effet, dans un environnement de réseau local sans fil, il est nécessaire d'avoir des paquets de petites tailles, d'où l'obligation d'utiliser des fonctions de fragmentation et de réassemblage au niveau de la couche MAC.

B.1.2 Les normes IEEE 802.11

Afin d'augmenter les performances en matière de débit, de zone de couverture, de portée, ou encore de types de services proposés, de nombreuses spécifications pour le standard 802.11 ont

[1] Medium Accès Control
[2] Logical Link Control

été ou sont actuellement développées. Quelques révisions de la norme 802.11 sont décrites dans le tableau B.1.2.

Norme	Caractéristiques	
802.11	Date de normalisation	1997
	Bande de fréquence	2.4 GHz
	Débit	théorique : 2 Mbps - réel : < 1 Mbps
	Portée théorique	100 m
802.11a	Date de normalisation	1999
	Bande de fréquence	5 GHz
	Débit	théorique : 54 Mbps - réel : 30 Mbps
	Portée théorique	50 m
	Spécificité	8 canaux radio
802.11b	Date de normalisation	1999
	Bande de fréquence	2.4 GHz
	Débit	théorique : 11 Mbps - réel : 6 Mbps
	Portée théorique	100 m
	Spécificité	3 canaux radio
802.11e	Amélioration de la qualité de service (niveau MAC) pour le support audio et vidéo	
802.11f	Interopérabilité entre les points d'accès	
802.11g	Date de normalisation	2003
	Bande de fréquence	2.4 GHz
	Débit	théorique : 54 Mbps - réel : 30 Mbps
	Portée théorique	20 m
	Spécificité	compatibilité 802.11b
802.11h	Adaptation de 802.11a aux normes d'émission électromagnétiques européennes	
802.11i	Amélioration de la sécurité des transmissions sur les bandes de fréquence 2,4 GHz et 5 GHz	

Communément appelé "Wifi", le standard 802.11b domine actuellement le marché des équipements de réseaux locaux sans fil (WLAN). Il a été déployé notamment dans les gares, aéroports, universités, entreprises, ... Cependant, la norme 802.11g, apparue plus récemment, possède tous les atouts pour s'imposer sur le marché des WLAN (compatibilité 802.11b, débits relativement élevés, ...).

D'autre part, les groupes de travail 802.11f, 802.11i et 802.11e interviennent sur les problèmes liés aux WLANs tels que le roaming, la sécurité et la qualité de service.

B.2 Fonctionnement des réseaux 802.11

B.2.1 Le mode infrastructure

Le mode "infrastructure" est basé sur une architecture cellulaire (le système est subdivisé en cellules) semblable à celle employée pour les téléphones portables. Chaque cellule (appelée Basic Service Set ou BSS dans la nomenclature 802.11) est contrôlée par une station de base (appelée Access Point ou AP -Point d'Accès en français-). Les points d'accès servent de ponts entre le réseau sans fil et le réseau filaire et sont chargés des services d'authentification et d'association.

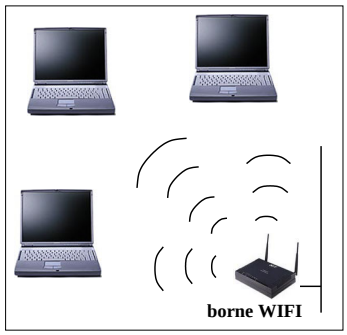


Figure B.1 *Le mode Infrastructure*

Le réseau local sans fil configuré en mode infrastructure peut être formé par une cellule unique, c'est à dire avec un seul Point d'Accès. Cependant, il est possible de relier plusieurs BSS entre eux (c'est à dire plusieurs points d'accès) via un système de distribution (DS ou Distributed System) permettant ainsi la mobilité des terminaux entre les différentes cellules (BSS). Un ensemble de plusieurs BSS formant un sous-réseau est appelé ensemble de services étendu (ESS ou Extended Service Set).

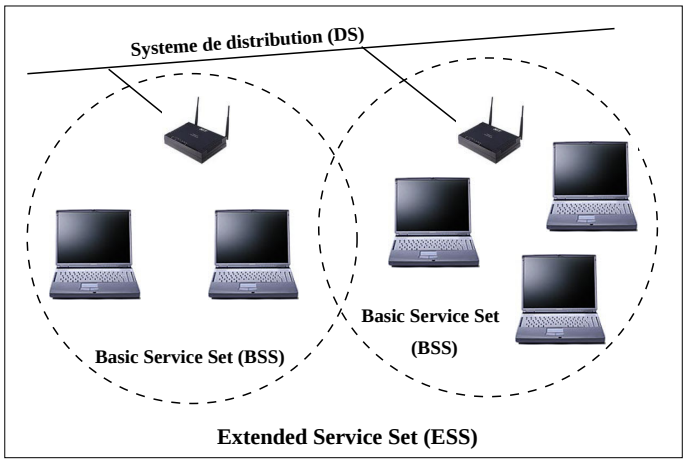


Figure B.2 *Extended Service Set*

B.2.2 Le mode ad-hoc

Cependant, une des particularité de ce mode de fonctionnement est que les stations doivent être capables d'effectuer toutes les opérations nécessaires à l'établissement et au maintien du réseau

Pour le mode "ad-hoc", appelé aussi mode "peer-to-peer", aucune infrastructure supplémentaire n'est nécessaire, les stations communiquent entre elles de façon autonome afin d'assurer leur connectivité et celle des autres stations par voie hertzienne. La rapidité de déploiement et la mobilité physique des stations permettent une grande souplesse d'utilisation (pas de raccordement, pas de câbles,...).

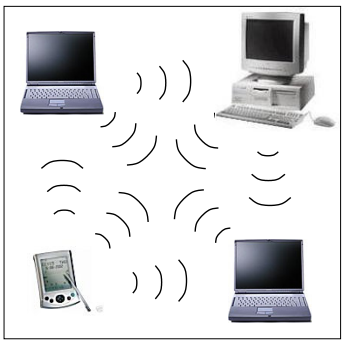


Figure B.3 *Le mode Ad-Hoc*

comme les procédures d'authentification et d'association.

Le réseau ad'hoc minimal est constitué de deux stations dans la zone de couverture radio l'une de l'autre, mais il n'y a aucune limitation de la taille du réseau. Cette architecture est aussi appelée IBBS (Independent Basic Service Set).

Une autre particularité du mode ad-hoc est son fonctionnement en point à point ou en multipoint, mais sans fonction de routage. Aussi, un terminal ne peut communiquer avec un autre terminal que si celui-ci se situe dans sa zone de couverture. Dans un réseau ad-hoc, la portée du IBSS est donc déterminée par la portée de chacun des terminaux le constituant.

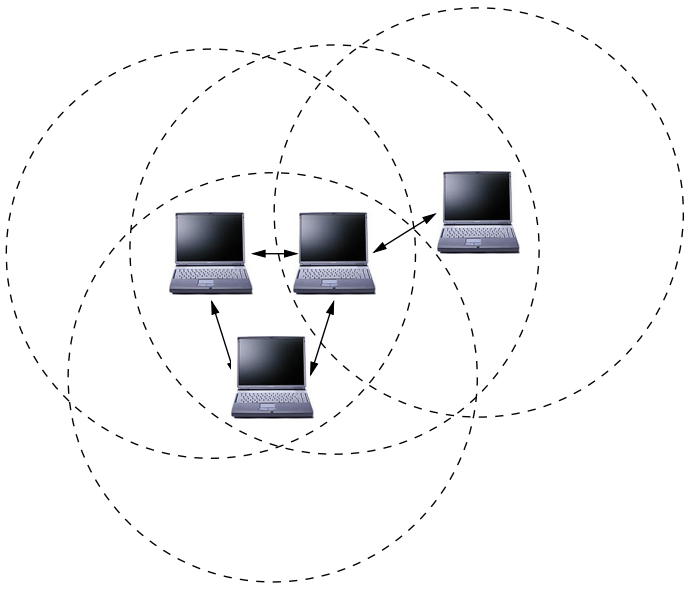


Figure B.4 *Zones de portée*

B.3 Architecture des réseaux 802.11

La norme 802.11 couvre les deux premières couches du modèle OSI : la couche physique et la couche liaison de données[57]. Le standard définit actuellement une seule couche MAC qui interagit avec 3 couches physiques : Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS) et Infrarouge.

<table>
<tr><td colspan="3">802.2
802.11</td><td>Liaison de données</td></tr>
<tr><td>FHSS</td><td>DSSS</td><td>IR</td><td>Physique</td></tr>
</table>

B.3.1 La couche physique

Le standard 802.11 spécifie quatre types de couches physiques (PHY) [43] : trois dans la bande des 2.4 GHz :

- FHSS (Frequency Hopping Spread Spectrum)
- DSSS (Direct Sequence Spread Spectrum)
- HR/DSSS (High Rate direct sequence) seulement pour 802.11b

et une couche physique pour la bande des 5 GHz : OFDM (Orthogonal Frequency Division Multiplexing) seulement pour 802.11a.

Ces appelations correspondent en fait aux techniques de transmission des signaux utilisées.

FHSS : technique d'étalement de spectre Cette technique de transmission utilise le principe de sauts de fréquences : l'émetteur passe d'une fréquence à une autre à intervalle régulier selon une règle de saut et un rythme spécifique. La bande des 2,4 GHz étant divisée en 75 sous-canaux de 1 MHz de largeur de bande, chaque sous-canal offre un débit d'au moins 1 Mbps, mais est limité à la vitesse de 2 Mbps. Ainsi, le principe de cette technique est basé sur le fait que l'émetteur et le récepteur s'accordent sur le choix d'un modèle de sauts de fréquences définissant la séquence de sauts de fréquence à réaliser afin d'envoyer les données successivement sur les différents sous-canaux. Ces modèles sont conçus de telle sorte que la probabilité que deux émetteurs utilisent le même sous-canal simultanément soit minimisée.

DSSS : technique d'étalement de spectre Cette technique de signalisation en séquence directe divise la bande des 2,4 GHz en 14 canaux de 22 MHz de largeur de bande, espacés de 5 MHz. Il y a donc chevauchement entre les canaux adjacents. Une telle largeur de bande fournit un signal généralement très bruité car les canaux adjacents ont des bandes passantes qui se recouvrent partiellement. Aussi, l'introduction d'une forte redondance dans le codage binaire est nécessaire (utilisation de la technique du "chipping"). En cas de bruit important, le standard 802.11b, qui utilise cette technique de transmission, utilise un changement de rythme dynamique. Le type de codage étant modifié, le débit chute à 5,5 Mbps, 2 Mbps voir 1 Mbps. Lorsque la perturbation diminue, le débit remonte automatiquement.

OFDM : Cette technique divise le canal disponible en plusieurs sous-canaux et encode une partie du signal sur chacun des sous-canaux en parallèle. Ainsi, le signal étant émis sur plusieurs

fréquences à la fois, les perturbations peuvent difficilemennt empecher la transmission puisqu'il suffit qu'un seul signal passe pour que le récepteur puisse reconstruire le message.

B.3.2 La couche liaison

Comme toutes les normes de réseaux locaux de l'IEEE, la couche de liaison 802.11 est constituée de deux sous-couches :

- LLC (Logical Link Control) qui permet d'adapter les données en provenance des couches supérieures à la couche physique.
- MAC (Medium Access Control) qui en plus de ses fonctionnalités standards (idem couche MAC 802.3), gère notamment la retransmission, l'acquittement, la fragmentation de trames. De plus, la couche MAC définit deux méthodes d'accès au support physique différentes : DCF (Distributed Coordination Function) et PCF (Point Coordination Function).

B.3.2.1 DCF (Distributed Coordination Function)

Cette methode d'accès au support est la méthode d'accès élémentaire aux réseaux 802.11 [52, 26]. Elle est fondée sur la méthode CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) elle-même basée sur la méthode CSMA/CD (Carrier Sense Multiple Access/Collision Detection) utilisée dans les réseaux locaux classiques. Afin de réduire les risques de collisions, le protocole CSMA/CA utilise un mécanisme de réservation du canal de transmission : RTS/CTS (Request To Send / Clear To Send) B.5.
Le principe de ce mécanisme réside dans la mise en place d'un protocole d'entente préalable pour l'utilisation du médium.

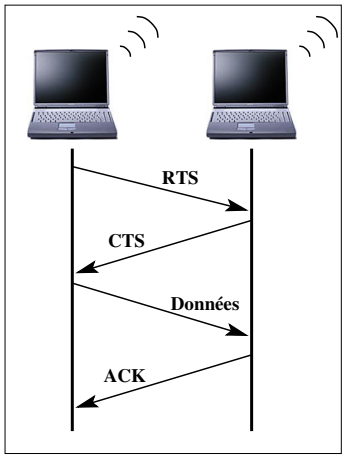


Figure B.5 *Mécanisme RTS/CTS*

En effet, avant chaque transmission entre deux terminaux, le terminal émetteur écoute la canal :
– si le canal est libre pendant un temps donné DIFS (Distributed Inter-Frame Space), le terminal peut émettre
– dans le cas contraire, la transmission est différée (le terminal doit impérativement attendre un temps DIFS après libération du canal avant d'émettre à son tour).

Le terminal émetteur peut alors envoyer une demande d'émission (RTS) indiquant notamment le volume de données à transmettre et sa vitesse de transmission. Le récepteur répond après un temps SIFS (Short Inter-Frame Space) par un CTS. Les autres terminaux ayant détectés ces trames peuvent ainsi retardés leur transmission (mise à jour de leur temporisateur NAV, Network Allocation Vector). Suite à cet échange, ie à la réception de la trame CTS, le terminal émetteur est assuré que le support est réservé à sa transmission et peut donc émettre ses données après avoir attendu un temps SIFS. Enfin, après transmission complète des données, le terminal récepteur renverra un acquittement.

L'avantage de l'utilisation des intervalles de temps (DIFS, SIFS, ...) consiste d'une part à réduire la probabilité de collision, et d'autre part à faciliter la synchronisation entre les stations. Issu de [43], le schéma B.6 illustre le principe d'accès au support décrit précédemment.

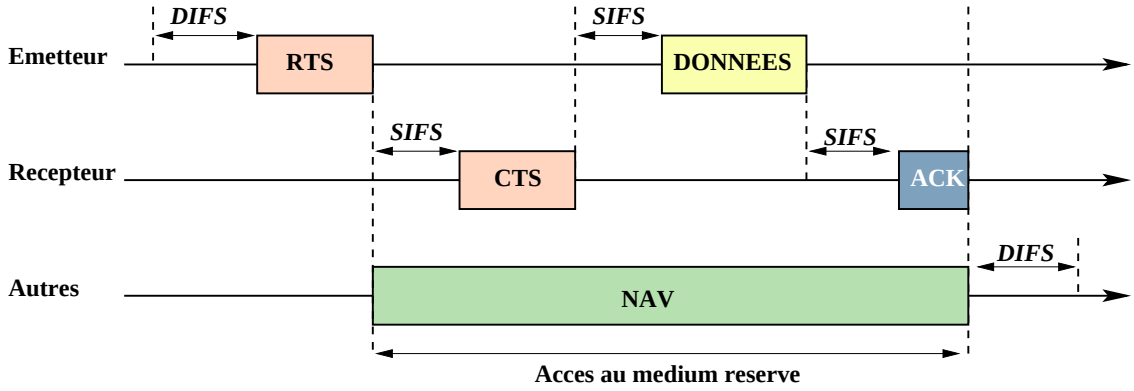


Figure B.6 *Réservation RTS/CTS*

B.3.2.2 PCF (Point Coordination Function)

Cette méthode d'accès consiste à contrôler les échanges sur le réseau par un PC (Point Coordinator). Le PC, généralement représenté par le point d'accès (AP) [26], permet d'avoir une coordination centralisée de l'accès au médium pendant la période dite CFP (Contention Free Period) durant laquelle la méthode PCF est utilisée. Cette période CFP est alternée par une trame permettant de synchroniser les terminaux avec une autre période dite CP (Contention Period) pendant laquelle la méthode DCF est utilisée. Utilisant des intervalles de temps PIFS (PCF Inter-Frame Space) plus courts que ceux utilisés par les terminaux, le point d'accès (AP) est donc privilégié. Aussi, cette méthode est particulièrement adaptée à la transmission de données audio ou vidéo.

La norme Bluetooth C

Initialement conçue pour remplacer les liaisons filaires utilisées jusqu'alors entre différents périphériques, la technologie Bluetooth est désormais à la portée du grand public (par exemple, les téléphones portables et les assistants personnels en sont équipés) et tend à se développer dans le domaine de la domotique. Cette technologie n'entre pas vraiment en concurrence avec la technologie WiFi mais lui est plutôt complémentaire. En effet, alors que WiFi est adapté aux réseaux de type WLAN (Wireless ....), la technologie Bluetooth s'adresse essentiellement aux réseaux de types WPAN (Wireless ....).

C.1 Généralités

Initié par Ericson et rapidement rejoint par IBM, Intel, Nokia et Toshiba au sein du SIG (Bluetooth Special Interest Group), le but de Bluetooth est d'unifier l'ensemble des constructeurs autour d'une seule norme pour la connectivité sans fil à courte portée. Aujourd'hui, plus de 2400 constructeurs se sont ralliés à ce consorium [106].

C.1.1 Caractéristiques

La technologie Bluetooth est un standard 1 Mbps économique conçu pour les réseau sans fil personnels. Bluetooth propose de simplifier tous les problèmes de connexions en permettant à tous les périphériques (PC portable, organiseurs, téléphones mobiles, ...) et appareils distants d'une dizaine de mètres (ou 100 mètres avec un amplificateur) de se connecter les uns aux autres. Ainsi, malgré sa faible portée (de l'ordre d'une dizaine de mètres), la technologie Bluetooth offre plusieurs avantages dont notamment une faible consommation d'énergie, une sensibilité aux interférences limitées grâce aux sauts de fréquences, etc ...

Spectre de fréquence :	2.4 GHz
Débit maximal :	1 Mbps
Portée :	10 mètres
Méthode d'accès :	saut de fréquence (FHSS)
Multicast :	oui
Mode économique :	oui
Sécurité :	cryptage 128 bits

Bluetooth utilise donc dans bande de fréquence 2.4 GHz, désignée bande ISM (Industrial, Scientific and Medical) utilisable désormais sans licence dans la plupart des pays. Afin de réduire les probabilités d'interférence, Bluetooth utilise les techniques d'étalement de spectres et de sauts de fréquences (FHSS : Frequency Hoping Spread Spectrum). Les sauts de fréquences sont réalisés sur 79 canaux espacés de 1 MHz [49]. Ainsi, en fonction de l'encombrement des transmissions, le signal peut être amené à effectuer jusqu'à 1600 sauts de fréquences pas seconde.

C.1.2 Spécification de la pile de protocole

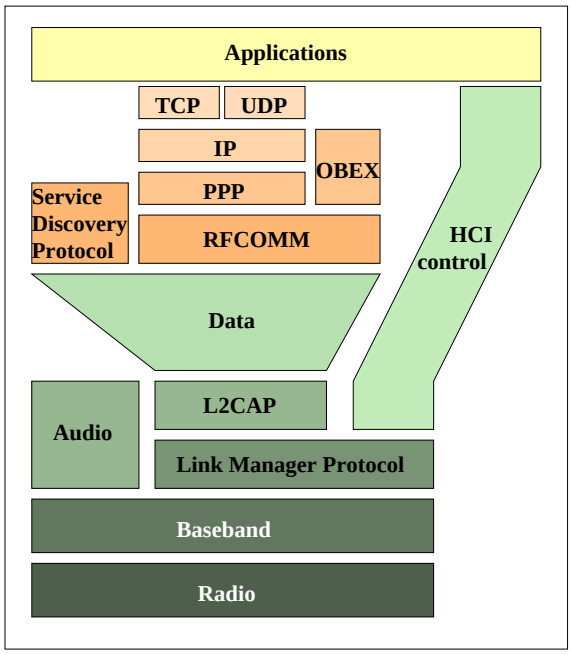


Figure C.1 *Schéma de la pile Bluetooth*

Dans [14], la pile Bluetooth est décrite grâce au schéma simplifié C.1.

- La couche "L2CAP" (Link Layer Control & Adaptation) permet l'adaptation des protocoles supérieurs (comme TCP/IP) au réseau Bluetooth : elle supporte la segmentation et le réassemblage, ainsi que le multiplexage de protocole.

- La couche "Link Manager" est responsable de la supervision des différentes connexions, de l'authentification des appareils, et du chiffrement. Elle gère également les mises en veille des différents appareils.
- La couche "Baseband Layer" définie les séquences de sauts de fréquences ainsi que les fonctionnalités permettant l'établissement de la connexion et la synchronisation des différentes unités.
- Les couches supérieures représentent entre autres les adaptations des standards au protocole Bluetooth (PPP, IP, UDP, ...). On trouve également dans cette couche une émulation du port série RS-232 nommée RFCOMM. De plus, un ensemble de protocoles dédiés à la téléphonie (gestion de standard téléphonique, sans fil, à main) et à l'identification des autres périphériques Bluetooth (SDP : Service Discovery Profil) sont regroupés dans ce groupe.

C.2 Topologie réseau

C.2.1 Les piconets

Les machines d'un réseau Bluetooth se rassemblent en sous-réseaux appelés "piconets".

Définition 9 *Un piconet représente un ensembles d'entités (maximum 8) connectées entre elles de manière ad-hoc.* □

Les entités composant un piconet ont entre elles une relation de type maître/esclave. A savoir, l'entité établissant le piconet prend automatiquement le rôle de maître. De plus, il n'y a qu'un seul maître par piconet, aussi les entités rejoignant ultérieurement le piconet deviennent les esclaves. Chaque maître peut accueillir jusqu'à 7 esclaves actifs, soit 8 appareils actifs au maximum par piconet.

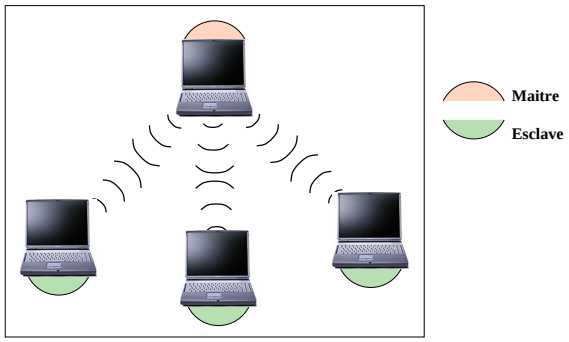


Figure C.2 *Piconet Bluetooth*

Le rôle du maître est multiple : d"une part, il gére une horloge commune, permettant ainsi la coordination de tous les membres d'un même piconet, et d'autre part, il définit une séquence de sauts de fréquences sur laquelle les différents esclaves du piconet se synchronisent. De plus, le

maître contrôle le trafic au sein du piconet afin d'éviter les collisions. Pour ce faire, la technique de polling est utilisée afin que les esclaves ne puissent transmettre des données simultannément.

Définition 10 *Un scatternet est formé par un ensemble de piconets.* □

L'intérêt d'un scatternet est que les différents piconets le composant peuvent interagir entre eux mais restent indépendant les uns des autres. Il est à noter qu'au sein d'un même scatternet, qu'une entité étant qualifiée d'esclave dans un des piconets peut être simultanément le maître d'un autre piconet.

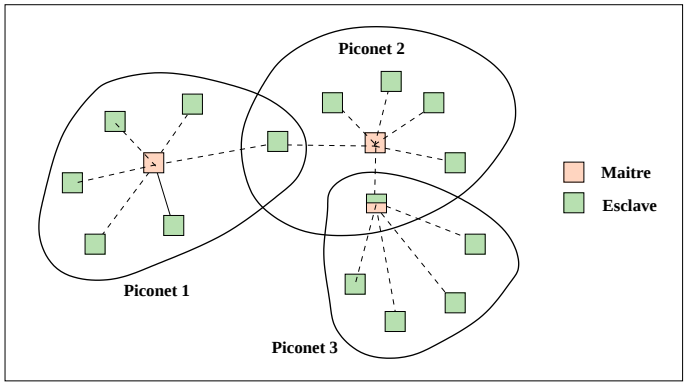


Figure C.3 *Scatternet*

C.2.2 Type de transmission

Divisé en slots de temps, le canal est partagé entre les transmissions du maître du piconet et celles des esclaves. La technologie Bluetooth définit deux types de liaisons différentes :

Les liaisons synchrones (SCO : Synchronous Connection-Oriented) : Dans ce mode de communication de type point-à-point entre le maître et un esclave particulier du piconet, le canal possède un débit bidirectionnel de 64 kb/s. Plus précisément, le débit de 64 kb/s est assuré simultannément en envoi et en réception et s'avère donc particulièrement adapté à la transmission de la voix. Le maître du piconet maintient le lien SCO en utilisant les slots réservés à intervalles réguliers aux paquets synchrones. Il est à préciser qu'une entité maître ne peut supporter en simultanné qu'au maximum 3 liaisons de ce type avec ses esclaves.

Les liaisons asynchrones (ACL : Asynchronous Connection Less) : Le mode asynchrone peut d'une part privilégier un débit élevé dans une direction particulière : 732.2 kb/s dans un sens (du maître vers l'esclave), contre seulement 57,6 kb/s dans l'autre sens (de l'esclave vers le maître), ou d'autre part proposer un débit symétrique de 433.9 kb/s. La liaison ACL permet au

maître d'un piconet d'avoir une communication multipoint vers tous les esclaves de son piconet. Le maître peut établir un lien ACL pendant les slots non réservés aux liens SCO.

Différents couples maître-esclave dans un même piconet peuvent utiliser différents types de liaisons, et peuvent en changer arbitrairement pendant une même session. Chaque type de liaison supporte jusqu'à 16 types de paquets différents dont 4 sont des paquets de contrôle et sont communs aux liaisons SCO et ACL.

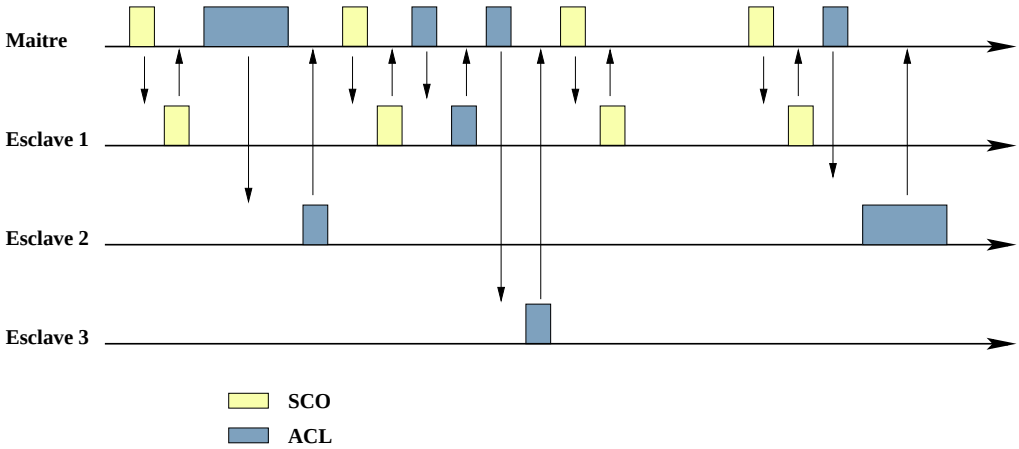


Figure C.4 *Partage des ressources dans un piconet.*

C.3 Protocole de connexion

C.3.1 Principe de connexion

La norme Bluetooth définit deux états principaux [106] : l'état *Standby* qui correspond à l'état par défaut et l'état *Connection* dans lequel maître et esclave peuvent échanger des paquets. Avant toute connexion, toutes les unités Bluetooth sont en mode passif ou mode d'écoute (état *Standby*), ce qui signifie principalement une réduction de la consommation d'énergie. Dans ce mode, une unité non connectée "écoute" des éventuels messages chaque 1,28 secondes. La procédure est inhibée par l'une des unités (n'importe laquelle) qui deviendra alors maître du piconet.

A ces deux états s'ajoutent sept sous-états, ou états intermédiaires [106, 81, 49], utilisés notamment à l'établissement de la connexion lors de l'insertion d'un nouvel esclave au sein d'un piconet : *Page*, *Page scan*, *Inquiry*, *Inquiry scan*, *Master response*, *slave response*, et *Inquiry response* (figure C.5).

La procédure *Inquiry* permet à une unité de découvrir tous les services à sa portée (ceux-ci incluant entre autres les imprimantes, les fax, ...), et est donc utilisée lorsque l'unité ne possède aucune information concernant les services distants. Quant à la procédure *Page*, elle permet d'établir la connexion proprement dite puisque l'unité étant dans l'état *Page* envoie des séries de messages identiques afin de localiser un esclave. Aussi, une unité établissant une connexion via la procédure *Page*, devient automatiquement le maître de la connexion.

Ainsi, de manière simplifiée, une connection est initialisée soit par un message de type *Page* si l'adresse est déjà connue ou soit par un message de type *Inquiry* suivi d'un message de type *Page* dans le cas contraire. De plus, le délai moyen pour qu'un maître découvre un esclave est inférieur à 1 seconde (délai maximum : 2,56 s).

Pour une description détaillée des procédure *Page* et *Inquiry*, voir [106].

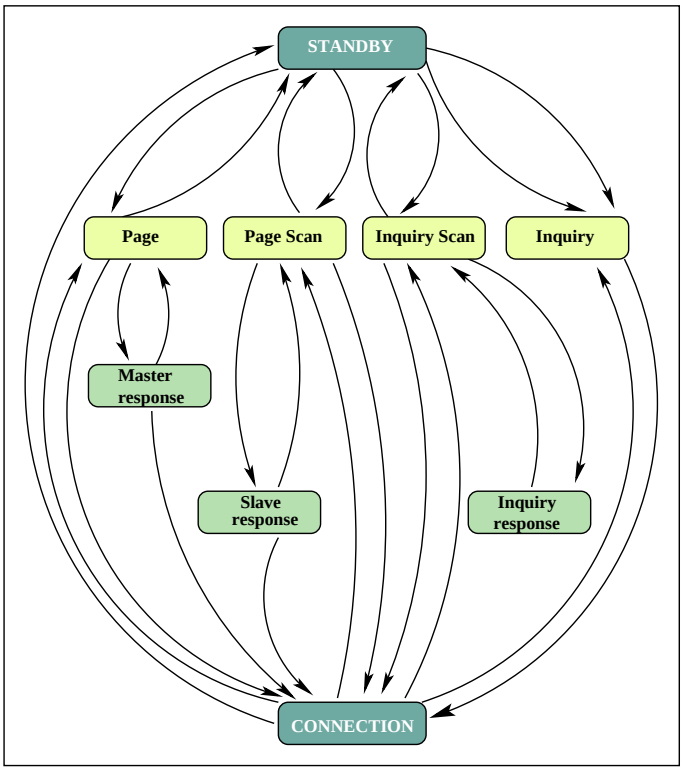


Figure C.5 *Graphe d'état de l'établissement d'une connexion.*

C.3.2 Etat de connexion

Une fois la connexion établie, l'unité Bluetooth peut être dans un des quatre modes suivants : le mode *Actif*, le mode *Sniff*, le mode *Hold* ou le mode *Park*, les trois derniers modes étant des modes économiques en terme d'énergie.

- Mode *Actif* : Dans ce mode, l'unité Bluetooth utilise activement le canal. En plus de la gestion des transmissions, le maître envoie périodiquement des messages permettant le maintien de la synchronisation des esclaves sur le canal.
- Mode *Sniff* : Ce mode d'économie d'énergie est le plus faible des trois modes *Sniff*, *Hold* et *Park*. Dans ce mode, l'esclave écoute le piconet mais avec une fréquence lente.
- Mode *Hold* : Ce mode d'économie d'énergie est le mode intermédiaire. Dans ce mode, seuls les messages de type SCO (Synchronous Connection Oriented) peuvent être reçus.

– Mode *Park* : Ce mode d'économie d'énergie est le plus fort des trois modes *Sniff*, *Hold* et *Park*. Le périphérique reste synchronisé avec le piconet mais ne participe pas au traffic.

Ainsi, un périphérique dans un réseau Bluetooth est soit maître soit esclave et il peut adopter 4 modes d'action : transmettre des données (mode actif), ou bien rester en écoute SNIFF, HOLD ou PARK.

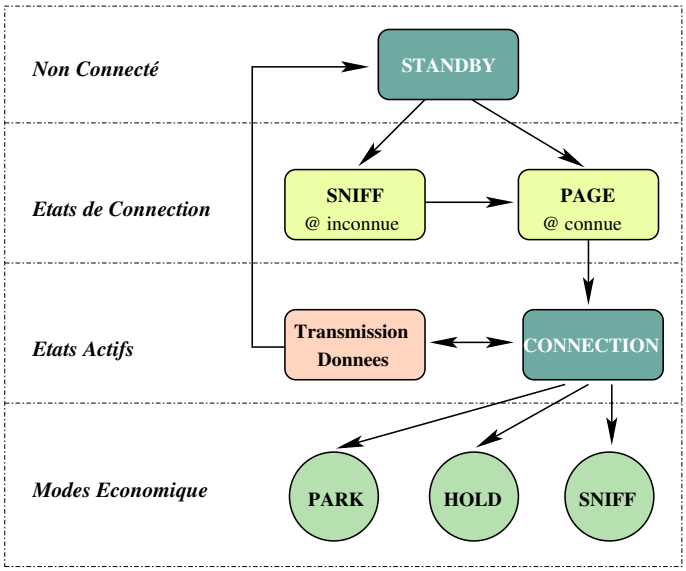


Figure C.6 *Les états possibles pour une unité Bluetooth. : Actif, Park, Hold ou Sniff.*

Algorithme de Chandra et Toueg D

D.0.3 Généralités

Les différentes classes de détecteurs de défaillances (voir chapitre 9) permettent de définir la "capacité" potentielle de résolution des problèmes d'accord par les algorithmes basés sur ces divers mécanismes. En effet, suivant la classe à laquelle appartient le détecteur de défaillances considéré, le nombre de défaillances pouvant être supportées est variable [29]. Ainsi, la classe $\mathcal{S}$ permet de résoudre le consensus quelque soit le nombre de processus susceptibles d'être défaillants, alors que la classe $\diamond\mathcal{S}$ permet de résoudre le consensus seulement sous la condition que le nombre de processus susceptibles de défaillir soit inférieur à la moitié du nombre de processus dans le système [19].
Par ailleurs, il est à noter que l'algorithme proposé dans [19] tolère $N-1$ défaillances de processus dans un système asynchrone composé de N processus, ce que ne permettent pas les algorithmes basés sur la classe de détecteurs de défaillances $\diamond\mathcal{P}$.

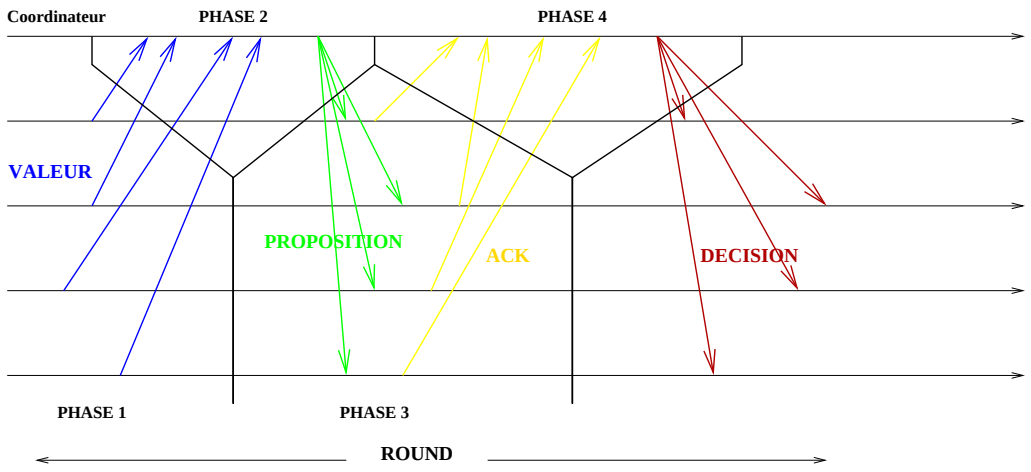
D'autre part, il est important de noter que la classe de détecteurs de défaillances $\diamond\mathcal{P}$ permet d'obtenir un service de communication de groupe, ce que n'offrent ni $\diamond\mathcal{S}$, ni $\mathcal{S}$.

D.0.4 Description informelle de l'algorithme de Chandra & Toueg

Nous nous sommes principalement intéréssés à l'algorithme de résolution de consensus utilisant la classe de détecteurs de défaillances de type $\diamond\mathcal{S}$ proposé dans [19].
On rappelle (chapitre 9) que cette classe de détecteurs de défaillances respecte la propriété de **complétude forte**, à savoir que tout processus incorrect finira par être suspecté par tous les processus corrects, et la propriété d'**exactitude ultime faible** assurant qu'après un certain temps, il existe un processus correct qui ne sera plus suspecté par aucun processus correct. De plus, dans un environnement asynchrone, la résolution du problème du consensus grâce à un détecteur de défaillances appartenant à cette classe $\diamond\mathcal{S}$ ne sera réalisable que sous une condition essentielle.

Ainsi, l'hypothèse à laquelle le système doit répondre est que plus de la moitié des processus du système doivent être corrects, c'est à dire qu'ils ne doivent subir aucune défaillance.

Le principe de cet algorithme repose sur le paradigme du coordinateur rotatif, à savoir que l'algorithme se décompose en diverses étapes asynchrones chacune étant gérées par un processus différent, le processus coordinateur de l'étape. Ce processus coordinateur est le processus $c = (r \ mod \ N) + 1$ où r représente l'étape (ou round) courante et N est le nombre de processus du système. Ainsi, à une étape donnée, tous les messages en transit sont soit à destination du processus coordinateur soit en sa provenance.
Chaque étape peut être décomposée en 4 phases asynchrones.
Durant la première phase, chaque processus du système envoie au coordinateur son estimation de la valeur de décision affranchie avec le numéro de l'étape durant laquelle il a choisi cette estimation.
La phase 2 consiste en la réception par le processus coordinateur de $(N+1)/2$ messages envoyés durant la phase 1. Après avoir rassemblé ces diverses estimations, le processus coordinateur sélectionne celle ayant le plus grand affranchissement et envoie cette nouvelle estimation à tous les processus du système.
Chaque processus doit désormais recevoir cette nouvelle estimation. Cette réception ainsi que son traitement représentent la troisième phase qui offre deux possibilités distinctes. Soit le processus qui reçoit cette nouvelle estimation envoie un acquittement au coordinateur pour signifier qu'il adopte cette estimation. Soit après avoir consulté son module de détection de défaillance, le processus qui reçoit cette nouvelle estimation (ou ne la reçoit pas) suspecte la panne du coordinateur, et envoie un "nack" à ce même coordinateur.
Enfin, pour la quatrième phase, le coordinateur attend $(N+1)/2$ réponses ("ack" ou "nack"). Si toutes les réponses sont des acquittements, alors le coordinateur sait que la majorité des processus a retenu cette nouvelle estimation, il envoie donc à tous les processus la décision sur cette estimation. Sinon, si le coordinateur a reçu un "nack", il n'envoie pas de message de décision et passe directement à l'étape suivante.



Etant donné que tous les processus ne décident pas forcément la même étape, il est nécessaire qu'un verrou soit posé sur la valeur de l'estimation décidée à une certaine étape.

Le comportement de l'algorithme lorsqu'un processus quelconque (autre que le coordinateur) subi une défaillance définitive, est illustré par la figure D.1.

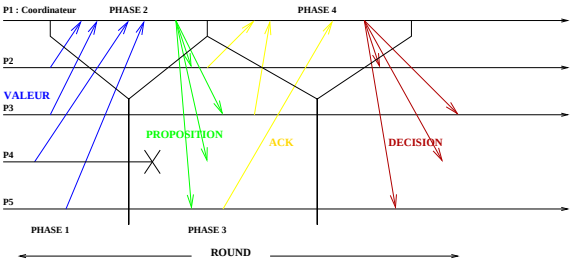


Figure D.1 *Crash d'un processus quelconque (pas le coordinateur).*

Dans le cas de la figure D.1, le processus $P4$ est défaillant en début de la $phase3$. Malgré ceci, la décision sera prise pendant ce round car le coordinateur $P1$ reçoit une majorité d'acquittement.

Par ailleurs, la fausse suspicion d'un participant envers le coordinateur du round en cours implique le passage au round suivant du participant concerné (voir figure D.2).

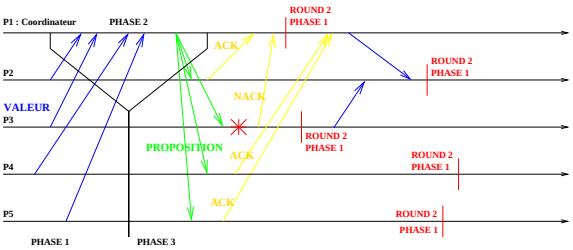


Figure D.2 *Suspicion du coordinateur.*

Comme le processus $P3$ suspecte le coordinateur, il envoie à celui-ci un "$NACK$" et passe au round suivant. Recevant un "$NACK$", le coordinateur $P1$ qui fonctionne correctement passe lui aussi au round suivant. Les autres processus ayant armé un timeout, s'ils n'ont rien reçu en provenance du coordinateur avant la fin du timeout, ils passent au round suivant. Après être passé au round suivant, chaque processus envoie sa valeur d'estimation au processus $P2$ qui est le nouveau coordinateur.

Enfin, la figure D.3 illustre le comportement de l'algorithme de consensus dans le cas de la défaillance définitive du coordinateur du round en cours.

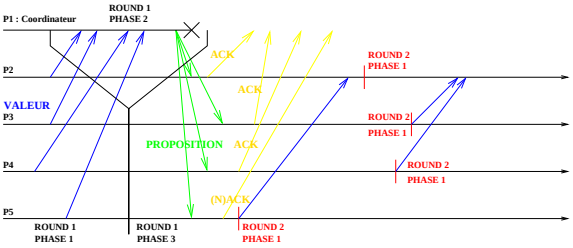


Figure D.3 *Crash du coordinateur.*

Dès qu'un processus suspecte le coordinateur $P1$, il passe au round suivant (cas du processus $P5$). Les autres processus ayant armé un timeout, s'ils n'ont rien reçu en provenance du coordinateur avant la fin du timeout, ils passent au round suivant.

D.0.5 Quelques extensions

A l'algorithme de résolution du consensus dans un environnement asynchrone augmenté de détecteurs de défaillances de la classe $\diamond\mathcal{S}$ proposé dans [19], diverses améliorations ont été proposées. En effet, basé sur le même principe que l'algorithme de résolution du consensus de [19], à savoir le coordinateur tournant, l'algorithme proposé dans [97] utilise lui aussi le mécanisme des détecteurs de défaillances de la classe $\diamond\mathcal{S}$. Cet algorithme de résolution du consensus, "the early consensus algorithm", se différencie de l'algorithme de Chandra & Toueg par le fait que tout n'est pas centralisé au niveau du coordinateur. En effet, contrairement à l'algorithme proposé dans [19] où tous les messages sont soit en provenance soit à destination du coordinateur, dans l'algorithme présenté dans [97] les échanges de messages se font entre tous les processus du système. Ainsi, au cours d'une même étape, tout d'abord, le coordinateur envoie à tous les processus son estimation de la valeur de décision. Ensuite, chaque processus qui reçoit cette estimation la réexpédie à tous les processus. Enfin, la décision est prise par un processus dès qu'il a reçu, d'une majorité de processus, la même valeur d'estimation. Si un processus suspecte le coordinateur, il broadcaste un message de suspicion. Dès qu'un processus sait que le coordinateur est suspecté par une majorité de processus, il envoie son estimation à tous les processus et assure la cohérence entre les différentes estimation avant de passer à l'étape suivante.

Dans cet article, [97], A. Schiper introduit une nouvelle notion, celle de "latency degree", qui permet de définir le nombre d'étapes de communication minimal nécessaire à la résolution du consensus. Cette notion permet de comparer différents algorithmes de résolution du consensus en terme d'étapes de communication. Ainsi, il apparaît que l'algorithme proposé par Chandra & Toueg a un degré de latence supérieur à celui proposé par Schiper.

Un autre protocole concernant la résolution du consensus dans un environnement asynchrone enrichie de détecteurs de défaillances de la classe $\diamond\mathcal{S}$ a été proposé par M. Hurfin & M. Raynal dans [55]. Comme les algorithmes précédemment présentés, le protocole proposé dans cet article est basé sur le principe du coordinateur tournant et procède en étapes asynchrones.
Le principe de ce protocole est basé sur un mécanisme de vote. En effet, à l'étape r, chaque processus doit choisir soit de décider pendant cette étape, soit de passer à l'étape suivante.
Le comportement d'un processus est modélisé par un automate à trois états : l'état q_0 qui correspond au fait que le processus n'a pas encore voté, l'état q_1 qui signifie que le processus a voté pour que la décision soit prise durant l'étape courante, et l'état q_2 qui signifie que le processus a voté en faveur d'un passage à l'étape suivante. Cet automate comporte trois transitions :

- Une transition de l'état q_0 à l'état q_1 a lieu lorsque le processus ne suspecte pas le coordinateur et reçoit de celui-ci un message signifiant son choix de décider durant l'étape courante (le coordinateur est passé dans l'état q_1). Le processus informe alors tous les processus de son vote.
- Une transition de l'état q_0 à l'état q_2 est effectuée lorsque le processus suspecte le coordinateur.Le processus informe ensuite tous les processus de son vote.
- Une transition de l'état q_1 à l'état q_2 signifie que le processus change son vote. Ceci est une possibilité offerte afin d'éviter les inter-blocages. Par contre, le choix de passer à l'étape suivante est un choix définitif.

Un processus décide de la valeur proposée par le coordinateur courant dès qu'il a reçu une majorité de messages signifiant un passage dans l'état q_1. Un processus passe à l'étape suivante s'il a reçu une majorité de messages signifiant un passage à l'état q_2 .

Bibliographie

[1] Coda & Odyssey,
http ://www-2.cs.cmu.edu/afs/cs/project/coda/Web/coda.html.

[2] General Packet Radio Service (GPRS),
http ://www.mobilegprs.com/.

[3] Global System for Mobile Communications (GSM),
http ://www.gsmworld.com/index.shtml.

[4] HiperLAN2,
http ://www.hiperlan2.com/.

[5] IEEE 802.11 Documentation,
http ://grouper.ieee.org/groups/802/11/.

[6] Monarch (Mobile Networking Architectures),
http ://www.monarch.cs.cmu.edu/.

[7] MosquitoNET,
http ://mosquitonet.standford.edu.

[8] Projet CESURE,
http ://www.essi.fr/ riveill/recherche/99-01-cesure.pdf.

[9] TIMELY (The Illinois Mobile Environments LaboratorY),
http ://timely.crhc.uiuc.edu/.

[10] Alanko, T. and Kojo, M. and Liljeberg, M. and Raatikainen, K. Mowgli : Improvements for Internet Applications Using Slow Wireless Links. In *8th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Helsinki, Finland, September 1997.

[11] Bernard, G. and Marvie, R. and Putrycz, E. and Taconet, C. Spécification de l'infrastructure système. Technical report, Projet RNRT CESURE, November 2000.

[12] Bertier, M. and Marin, O. Implémentation d'un Détecteur de Défaillances Adaptable. In *ASF 2002 (Journées francophones des jeunes chercheurs en systèmes d'exploitation)*, pages 467–472, Hammamet, Tunisie, April 2002.

[13] Bertier, M. and Marin, O. and Sens, P. Implementation and Performance of an Adaptable Failure Detector. In *International Conference on Dependable Systems and Networks (DSN'02)*, Washington DC, USA, June 2002. IEEE Society Press.

[14] Bhagwat, P. Bluetooth : Technology for Short-Range Wireless Apps. *IEEE Internet Computing*, 5(3), May 2001.

[15] Bharghava, V. and Lee, K-W. and Lu, S. and Ha, S. and Li, J-R. and Dwyer, D. The TIMELY Adaptive Resource Management Architecture. *IEEE Personal Communications Magazine*, 5(4), August 1998.

[16] Bridgland, M. and Watro, R. Fault-tolerant Decision Making in Totally Asynchronous Distributed Systems. In ACM Press, editor, *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, pages 52–63, August 1987.

[17] Canetti, R. and Rabin, T. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *In 25th ACM Symposium on Theory of Computing*, pages 42–51, San Diego, California, United States, 1993. ACM Press.

[18] Chandra, T. and Hadzilacos, V. and Toueg, S. The Weakest Failure Detector for Solving Consensus. In Maurice Herlihy, editor, *11th Annual ACM Symposium on Principles of Distributed Computing (PODC'92)*, pages 147–158, Vancouver, BC, Canada, 1992. ACM Press.

[19] Chandra, T. and Toueg, S. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2) :225–267, March 1996.

[20] Chen, W. and Toueg, S. and Aguilera, M. On the Quality of Service of Failure Detectors. In *International Conference on Dependable Systems and Networks (DSN 2000)*, New York, 2000. IEEE Computer Society Press.

[21] Chun, B-G. and Jaisinghani, D. and Baker, M. Evaluation of Scheduling Algorithms in Mobile Ad Hoc Networks. *ACM Mobile Computing and Communications Review*, 6, July 2001.

[22] Coccoli, A. and Urbán, P. and Bondavalli, A. and Schiper, A. Performance Analysis of a Consensus Algorithm Combining Stochastic Activity Networks and Measurements. In *Proc. Int'l Conf. on Dependable Systems and Networks (DSN)*, pages 551–560, Washington, DC, USA, June 2002.

[23] Conan, D. and Bretelle, B. and Chabridon, S. and Bernard, G. Support pour l'Exécution, en Mode Déconnecté, d'Applications Distribuées dans les Environnements Mobiles. In *Proceedings of the 1st Symposium sur les Objets Communicants*, October 2001.

[24] Conan, D. and Chabridon, S. and Bernard, G. Disconnected Operations in Mobile Environments. In *2nd International Workshop on Parallel and distributed Computing Issues in Wireless Networks and Mobile Computing*, April 2002.

[25] Coulouris, G. and Dollimore, J. and Kindberg, T. *Distributed systems (3rd ed.) : concepts and design*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[26] Crow, B. Performance Evaluation of the IEEE 802.11 Wireless Local Area Network Protocol, 1996.

[27] Das, S. and Mukherjee, A. and Bandyopadhyay, S. and Saha, D. and Paul, K. An adaptive framework for QoS routing through multiple paths in ad hoc wireless networks. *J. Parallel Distrib. Comput.*, 63(2) :141–153, 2003.

[28] Deering, S. and Hinden, R. Internet Protocol, Version 6 (IPv6) Specification. Technical report, Internet Engineering Task Force RFC 2460, December 1998.

[29] Delporte-Gallet, C. *Sur l'Algorithmique Distribuée Tolérante aux Pannes*. Habilitation à diriger des recherches, Université Paris VII, 2001.

[30] Demers, A. and Peterson, K. and Spreitzer, M. and Terry, D. and Theimer, M. and Welch, B. The Bayou Architecture : Support for Data Sharing among Mobile Users. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 2–7, Santa Cruz, California, USA, December 1994.

[31] Dolev, D. and Dwork, C. and Stockmeyer, L. On the Minimal Synchronism Needed for Distributed Consensus. *Journal of the ACM*, 34(1) :77–97, January 1987.

[32] Dolev, D. and Lynch, N. and Pinter, S. and Stark, E. and Weihp, W. Reaching Approximate Agreement in the Presence of Faults. *Journal of the ACM*, 33(3) :499–516, July 1986.

[33] Durand, Y. and Perret, S. and Vincent, J-M. and Marchand, C. and Ottogalli, F-G. and Olive, V. and Martin, S. and Dumant, B. and Chambon, S. SIDRAH : A software infrastructure for a resilient community of wireless devices. In *Smart Objects Conference*, pages 134–137, Grenoble, France, 2003.

[34] Dwork, C. and Lynch, N. and Stockmeyer, L. Consensus in the Presence of Partial Synchrony. *Journal of the ACM (JACM)*, 35(2) :288–323, 1988.

[35] Edwards, W. and Mynatt, E. and Petersen, K. and Spreitzer, M. and Terry, D. and Theimer, M. Designing and Implementing Asynchronous Collaborative Applications with Bayou. In *10th ACM Symposium on User Interface Software and Technology (UIST'97)*, pages 119–128, Banff, Canada, October 1997.

[36] Ernst, T. and Castelluccia, C. and Lach, H. Les Réseaux Mobiles dans IPv6. In *DNAC'99 (De Nouvelles Architectures pour les Communications)*, Paris, France, December 1999. Ministère Chargé des Télécoms.

[37] Estefanel, L. and Jansch-Porto, I. On the Evaluation of Failure Detectors Performance. In *IX Brazilian Symposium on Fault-Tolerant Computing (SCTF)*, pages 73–84, Florianopolis, March 2001.

[38] Estefanel, L. and Jansch-Porto, I. On the Evaluation of Heartbeat-like Detectors. In *2nd Latin-American Test Workshop (LATW'01)*, Cancun, Mexico, 2001.

[39] Fetzer, C. and Raynal, M. and Tronel, F. An adaptive failure detection protocol. In *8th IEEE Pacific Rim Symp. on Dependable Computing (PRDC-8)*, pages 146–153, Seoul, Korea, December 2001.

[40] Fischer, M. and Lynch, N. and Paterson, M. Impossibility of Distributed Consensus with One Faulty. *Journal of the ACM*, 32(2) :374–382, 1985.

[41] Fortier, P. and Michel, H. *Computer Systems Performance Evaluation and Prediction*. Digital Press, 2003.

[42] Garg, V. and Wilkes, J.-E. *Wireless and Personal Communication Systems*. Prentice Hall, 1996.

[43] Gast, M. *802.11 Wireless Networks : The Definitive Guide*. O'Reilly, 2002.

[44] Guerraoui, R. and Hurfin, M. and Mostéfaoui, A. and Oliveira, R. and Raynal, M. and Schiper, A. Consensus in Asynchronous Distributed Systems : A Concise Guided Tour. In

Advances in Distributed Systems, Advanced Distributed Computing : From Algorithms to Systems, pages 33–47. Springer-Verlag, 1999.

[45] Guerraoui, R. and Schiper, A. Consensus Service : A Modular Approach For Building Fault-Tolerant Agreement Protocols in Distributed Systems. In *26th International Symposium on Fault-Tolerant Computing (FTCS-26)*, pages 168–177, Sendai, Japan, June 1996.

[46] Guttman, E. and Perkins, C. and Veizades, J. and Day, M. Service Location Protocol, Version 2. Technical report, IETF RFC 2608, http ://www.ietf.org/rfc/rfc2608.txt, June 1999.

[47] Haahr, M. and Cunningham, R. and Cahill, V. Supporting CORBA Applications in a Mobile Environment. In *MobiCom'99 : 5th International Conference on Mobile Computing and Networking*, pages 36–47, August 1999.

[48] Haahr, M. and Cunningham, R. and Cahill, V. Towards a Generic Architecture for Mobile Object-Oriented Applications. In *IEEE Workshop on Service Portability and Virtual Customer Environments*, pages 91–96, December 2000.

[49] Haartsen, J. The Bluetooth Radio System. *IEEE Personal Communications*, February 2000.

[50] Haddad, Y. *Performance dans les systèmes répartis : des outils pour les mesures*. PhD thesis, Université Paris-Sud, Orsay, France, 1988.

[51] Haverkort, B. *Performance of Computer Communication Systems*. John Wiley & Sons, 1998.

[52] Heindl, A. and German, R. Performance Modeling of the IEEE 802.11 Wireless LANs with Stochastic Petri Nets. *Performance Evaluation*, 44 :139–164, 2001.

[53] Holma, H. and Toskala, A. *UMTS, les réseaux mobiles de troisième génération*. Osman Eyrolles Multimédia, 2001.

[54] HomeRF Working Group, http ://www.palowireless.com/homerf/homerfspec.asp. *HomeRF Specification, Revision 2.01*, July 2002.

[55] Hurfin, M. and Raynal, M. A Simple and Fast Asynchronous Consensus Protocol Based on a Weak Failure Detector. *Distributed Computing*, 12(4) :209–223, 1999.

[56] Hurfin, M. and Raynal, M. and Tronel, F. and Macêdo, R. A General Framework to Solve Agreement Problems. In *18th IEEE Symposium on Reliable Distributed Systems*, Lausanne, Switzerland, October 1999.

[57] IEEE 802.11. Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification.

[58] Infrared Data Association, http ://irda.affiniscape.com/displayemailforms.cfm. *IrDA Physical Layer Specification v1.4*, May 2001.

[59] Jacobson, V. Congestion avoidance and control. In *ACM SIGCOMM'88*, Stanford, CA, USA, August 1988.

[60] Jing, J. and Helal, A. and Elmagarmid, A. Client-server computing in mobile environments. *ACM Comput. Surv.*, 31(2) :117–157, 1999.

[61] Johnson, D. Validation of Wireless and Mobile Network Models and Simulation. In *DARPA/NIST Workshop on Validation of Large-Scale Network Models and Simulation*, Fairfax, Virginia, USA, May 1999.

[62] Johnson, D. and Maltz, D. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications*, February 1996.

[63] Johnson, D. and Maltz, D. and Hu, Y-C. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). Internet-draft, IETF MANET Working Group, April 2003. draft-ietf-manet-dsr-09.txt.

[64] Johnson, D. and Perkins, C. Mobility Support in IPv6. Technical report, Internet Draft, July 2001. draft-ietf-mobileip-ipv6-15.txt.

[65] Joseph, A. and Delespinasse, A. and Tauber, J. and Gifford, D. and Kaashoek, M. Rover : A Toolkit for Mobile Information Access. In *Fifteenth Symposium on Operating Systems Principles*, December 1995.

[66] Katz, R. and Brewer, E. and Amir, E. and Balakrishnan, H. and Fox, A. and Gribble, S. and Hodes, T. and Jiang, D. and Nguyen G. and Padmanabhan, V. and Stemm, M. The Bay Area Research Wireless Access Network (BARWAN). In *41th IEEE Computer Society International Conference : Technologies for the Information Superhigh-way (COMPCON'96)*, pages 15–20, Santa Clara, California, USA, February 1996.

[67] Kim, T. and Bharghavan, V. Multicast Routing in Heterogeneous Wireline/Wireless Environments. In *IEEE Wireless Communications and Networking Conference*, New Orleans, LA, September 1999.

[68] Kistler, J. and Satyanarayanan, M. Disconnected Operation in the Coda File System. In *ACM Transaction on Computer Systems*, volume 10, pages 3–25, February 1992.

[69] Kumar, P. and Satyanarayanan, M. Flexible and Safe Resolution of File Conflicts. In *USENIX Winter 1995 Technical Conference*, pages 95–106, New Orleans, Louisiana, USA, January 1995.

[70] Laboratoire Informatique et Distribution. Pajé Home Page. http ://www-apache.imag.fr/software/paje/.

[71] Lamport, L. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7) :558–566, July 1978.

[72] Le Mouël, F. *Environnement adaptatif d'exécution distribuée d'applications dans un contexte mobile*. PhD thesis, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA) / INRIA Rennes, UNIVERSITE RENNES I, 2003.

[73] Lee, S-J. and Hsu, J. and Hayashida, R. and Gerla, M. and Bagrodia, R. Selecting a routing strategy for your ad hoc network. *Computer Comunications*, 26(7) :723–733, May 2003.

[74] Lynch, N. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., 1996.

[75] Lynch, N. Supporting Disconnected Operation in Mobile CORBA. Master's thesis, Trinity College, Dublin, September 1999.

[76] Maillet, E. *Le traçage logiciel d'applications parallèles : conception et ajustement de qualité*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 1996.

[77] Marchand, C. Projet MIRRA : Maintien d'Infrastructure Répartie sur des Réseaux Ambiants. Technical report, France Télécom R&D - INRIA, 2001.

[78] Marchand, C. Projet MIRRA : Analyse comportementale & Algorithmique répartie. Technical report, France Télécom R&D - INRIA, December 2002.

[79] Marchand, C. Projet MIRRA : Modèles et Algorithmes, approche middleware. Technical report, France Télécom R&D - INRIA, July 2002.

[80] Marchand, C. Projet MIRRA : Etudes complémentaires, Implémentation et Expérimentations. Technical report, France Télécom R&D - INRIA, July 2003.

[81] Maric, I. Connection Establishment in the Bluetooth System. Master's thesis, The State University of New Jersey, October 2000.

[82] Microsoft Corporation. Understanding Universal Plug and Play : a White Paper. Technical report, http ://www.upnp.org/resources/whitepapers.asp, June 2000.

[83] Monks, J. Power Controlled Multiple Access in Ad Hoc Networks. In *Multiaccess, Mobility and Teletraffic for Wireless Communications (MMT)*, Duck Key, FL, December 2000.

[84] Mullender, S. *Distributed Systems*. ACM Press, 1994. second edition.

[85] Mummert, L. and Ebling, M. and Satyanarayanan, M. Exploiting Weak Connectivity for Mobile File Access. In *15th ACM Symposium on Operating System Principles (SOSP'95)*, pages 143–155, Copper Mountain Resort, Colorado, USA, December 1995.

[86] Nelson, R. *Probability, Stochastic Processes, and Queueing Theory : The Mathematics of Computer Performance Modeling*. Springer-Verlag, 1995.

[87] Noble, B. System Support for Mobile, Adaptive Applications. *IEEE Personal Computing Systems*, 7(1) :44–49, February 2000.

[88] Noble, B. and Satyanarayanan, M. Experience with Adaptative Mobile Applications in Odyssey. *Mobile Networks and Applications*, 4(4) :245–254, 1999.

[89] Noble, B. and Satyanarayanan, M. and Tilton, J. and Flinn, J. and Walker, K. Agile Application-Aware Adaptation for Mobility. In *16th ACM Symposium on Operating Systems Principles (SOSP'16*, Saint Malo, France, October 1997.

[90] Ottogalli, F-G. *Observations et analyses quantitatives multi-niveaux d'applications à objets réparties*. PhD thesis, Université Joseph Fourier, Grenoble, France, 2001.

[91] Pahlavan, K. and Probert, T.-H. and Chase, M.-E. Trends in Local Wireless Networks. *IEEE Communications Magazine*, 33 :40–45, April 1995.

[92] Perkins, C. Mobile IP. *IEEE Communications Magazine*, 35(5) :84–99, May 1997.

[93] Perkins, C. and Johnson, D. and Arkko, J. Mobility Support in IPv6. Technical report, February 2003. draft-ieft-mobileip-ipv6-21.txt.

[94] Projet RNRT SIDRAH, 2001, // http ://www.telecom.gouv.fr/rnrt/projets/res_01_21.htm.

[95] Riveill, M. and Pellegrini, M-C. and Potonniée, O. and Marvie, R. Adaptabilité des Applications pour des Usagers Mobiles". In *OCM 2000*, Nantes, France, May 2000.

[96] Salutation Consortium. Salutation Architecture Specification, Version 2.1. Technical report, http ://www.salutation.org, 1999.

[97] Schiper, A. Early Consensus in an Asynchronous System with a Weak Failure Detector. *Distributed Computing*, 10(3) :149–157, April 1997.

[98] Sergent, N. and Défago, X. and Schiper, A. Impact of a Failure Detection Mechanism on the Performance of Consensus. In *Proc. IEEE Pacific Rim Symp. on Dependable Computing (PRDC)*, Seoul, Korea, December 2001.

[99] Sun Microsystems. Jini Architectural Overview. Technical report, Technical White Paper, http :
www.sun.com/jini/whitepapers/architecture.pdf, 1999.

[100] Sun Microsystems. Jini Architecture Specification. Technical report, version 2.0, http :
wwws.sun.com/software/jini/specs/, 2003.

[101] Tanenbaum, A. *Réseaux*. Dunod / Prentice Hall, 1997. 3e édition.

[102] Tanenbaum, A. and van Steen, M. *Distributed Systems : Principles and Paradigms*. Prentice Hall, 2002.

[103] Tang, D. and Baker, M. Analysis of a Local-Area Wireless Network. In *6th annual international conference on Mobile Computing and Networking*, pages 1–10, Boston, Massachusetts, USA, August 2000.

[104] Tel, G. *Introduction to Distributed Algorithms*. Cambridge University Press, 1994.

[105] Terry, D. and Theimer, M. and Peterson, K. and Demers, A. and Spreitzer, M. and Hauser, C. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *15th ACM Symposium on Operating System Principles (SOSP'95)*, pages 172–183, Copper Mountain Resort, Colorado, USA, December 1995.

[106] The Bluetooth Special Interest Group (SIG). Specification of the Bluetooth System,
http ://www.bluetooth.com/.

[107] Tixeuil, S. *Auto-Stabilisation Efficace*. PhD thesis, Université Paris Sud, 2000.

[108] Trezentos, D. 2002,
http ://www.techniques-ingenieur.fr/.

[109] Trigila, S. and Bonafede, V. Project ACTS 036 DOLMEN : Final Report. Public, DOLMEN, February 1999. Version 02.

[110] UPnP Forum. UPnP Device Architecture 1.0. Technical report, version 1.0.1, http ://www.upnp.org/resources/documents.asp, December 2003.

[111] Verdu, J-P. 2000,
http ://membres.lycos.fr/lfm/index.htm.

[112] Wolff, R. *Stochastic Modeling and the Theory of Queues*. Prentice-Hall International Editions, 1989.

[113] Zhao, X. and Castelluccia, C. and Baker, M. Flexible Network Support for Mobility. In *4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98)*, pages 145–156, Dallas, Texas, USA, October 1998. MobiCom'98.

[114] Zhao, X. and Castelluccia, C. and Baker, M. Flexible Network Support for Mobile Hosts. *Mobile Networks and Applications (MONET)*, 6(2) :137–149, April 2001.

Résumé

Mise au point d'algorithmes répartis dans un environnement fortement variable, et expérimentation dans le contexte des pico-réseaux.

L'évolution technique des "objets communicants" tels que les ordinateurs portables, les assistants personnels, etc..., confirme l'intérêt porté aux technologies "sans fil". Le développement rapide des protocoles de communication sans fil (Bluetooth, WIFI, ...) a permis la généralisation de nouveaux "réseaux locaux ad-hoc" principalement caractérisés par leur topologie dynamique et l'hétérogénéité de leurs composants. Aussi, les infrastructures logicielles reposant sur de tels environnements doivent pouvoir s'adapter et gérer cette dynamicité (connexions/déconnexions fréquentes ainsi que la forte variabilité des communications.

Dans ce contexte, l'objectif général est de concevoir des algorithmes permettant de maintenir la cohérence d'un groupe d'entités hétérogènes partageant des services sur des architectures fortement instables. La problématique concerne donc la prise de décision en environnement distribué en considérant les particularités du réseau sous-jacent.

Dans cette thèse, nous proposons une méthodologie d'approche concernant la caractérisation des performances envisageables dans un environnement totalement distribué s'appuyant sur un protocole de communication sans fil. Cette approche consiste également en l'identification des différentes perturbations susceptibles d'interférer dans le bon déroulement des applications.

Par ailleurs, un algorithme de consensus adapté aux spécificités de l'environnement et basé sur un principe d'interaction avec un mécanisme de détection de défaillances a été développé et implanté. Afin de valider cette approche algorithmique, diverses expérimentations et évaluations qualitatives et quantitatives ont été réalisées.

Mots-clés Algorithmique distribuée, consensus et détecteurs de défaillances, réseaux sans fil, modèles stochastiques et évaluation de performances.

Abstract

Adaptation of Distributed Algorithms in Highly Dynamic Environments, Experimentation in Wireless Ad-hoc Networks.

Technological advances in wireless devices (laptop computers, personal digital assistants (PDAs), mobile phones, ...) bring up significance to new wireless technologies. Progress in wireless communication protocols (Bluetooth, WIFI, ...) allow the use of new ad-hoc networking schemes. Then, new challenges arise from the communication variability in wireless networks and the unpredictable disconnections of those heterogeneous devices, creating very dynamic topologies named ad-hoc wireless networks.

In this context, to maintain the consistency of a wireless device group, we built a middleware addressing the distributed agreement problem in unreliable environment. Therefore, to characterize wireless environments performances, we first express a method, which tries to identify the factors acting upon system variability.

Based on unreliable failure detectors, the consensus algorithm proposed is able to make a decision for distributed systems despite of unreliable communications or process failures (disconnections or crashes). An evaluation on either the behavior analysis and the quality of services has been performed to validate the method above.

Keywords Distributed algorithms, consensus and failure detectors, wireless ad-hoc networks, quality of service and modeling, performance evaluations.