INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Équipe ADEPT

Asynchronous Distributed Environments, Protocols, and Time

Rennes

THÈME 1C

Table des matières

1. Composition de l'équipe

Responsable scientifique
Michel Hurfin [CR, INRIA]

Assistante de projet
Maryse Auffray [adjoint technique, INRIA]
Lydie Letort [technicien de la recherche, INRIA, à partir de novembre 2002]

Personnel Cnrs
Emmanuelle Anceaume [CR]

Personnel Université de Rennes I
Maria Gradinariu [maître de conférences]
Achour Mostéfaoui [maître de conférences]
Michel Raynal [professeur]

Collaborateur extérieur
Jean-Pierre Le Narzul [maître de conférences à l'ENSTBr]

Chercheurs invités
Aline Andrade [Professeur, Federal University Bahia, Brésil, du 3 au 18 juin 2002]
Flavio Assis Silva [Professeur, Federal University Bahia, Brésil, du 3 au 18 juin 2002]
Raimundo Macedo [Professeur, Federal University Bahia, Brésil, du 3 au 18 juin 2002]
Krishnamurthy Vidyasankar [Professeur, Memorial University Newfoundland, Canada, du 12 novembre au 7 décembre 2002]
Yun Wang [Professeur, Southeast University, Chine, du 13 août au 14 septembre 2002]

Chercheurs doctorants
Frédéric Dang Ngoc [CDD France Télécom, à partir d'octobre 2002]
Erwan Demairy [ATER, jusqu'en novembre 2002]
Fabiola Greve [bourse du gouvernement brésilien, jusqu'en novembre 2002]
Eric Mourgaya [bourse INRIA]
Julien Pley [bourse du ministère, à partir de décembre 2002]
Philippe Raïpin Parvédy [bourse du ministère]
Matthieu Roy [allocation couplée]
Gwendal Simon [CDD France Télécom]
Frédéric Tronel [Ingénieur expert INRIA Rhones Alpes]

Chercheurs post-doctorants
Mourad Rabah [ACI GRID GénoGRID, du 18 février au 17 août 2002]
Xiaojun Ma [ministère de la recherche, à partir de décembre 2002]

2. Présentation et objectifs généraux

Les applications informatiques réparties sont omniprésentes dans tous les secteurs de l'économie. Désormais, un calculateur se trouve rarement isolé du reste du monde. Au contraire, il est implicitement intégré dans un réseau (local ou de plus grande ampleur) et la plupart des calculs qui lui sont soumis nécessitent une coopération avec d'autres calculateurs connectés de manière permanente ou temporaire. De façon générale, le paysage informatique actuel se caractérise donc par une interconnexion massive et évolutive de calculateurs hétérogènes, géographiquement distants et potentiellement mobiles.

Cet avènement de l'informatique répartie, qui a été rendu possible grâce aux progrès technologiques réalisés, est également le fruit de l'intense activité de recherche qui au cours de ces dernières années a eu pour objectif de maîtriser les problèmes fondamentaux propres aux systèmes répartis. Qu'ils soient de nature théorique ou pratique, la plupart de ces travaux ont été menés en prenant garde de ne pas se restreindre à une application

particulière, ni même à une architecture particulière. Ils ont permis la définition de concepts et d'abstractions qui ont ensuite été exploités pour concevoir des couches intergicielles (middleware) proposant des services de haut niveau, essentiels pour de nombreuses applications. Ces services sont avant tout destinés à faciliter la réalisation et le déploiement des applications réparties en masquant notamment au concepteur d'applications la complexité du système réparti qui, dans le pire des cas, peut être asynchrone, non fiable, mobile et évolutif. En définissant une machine virtuelle, indépendante des architectures utilisées, cette approche permet une maîtrise des coûts et des délais de production des systèmes et de leurs logiciels.

Des domaines aussi variés que l'observation, la synchronisation, la cohérence des données, la tolérance aux défaillances et la sécurité, pour ne citer qu'eux, ont bénéficié de ces avancées significatives et ces problèmes sont désormais plus ou moins bien pris en compte dans les architectures intergicielles proposées sur le marché. Mais beaucoup de travail reste encore à faire car les exigences des concepteurs d'applications ne cessent d'augmenter alors que dans le même temps les caractéristiques des environnements d'exécution deviennent de moins en moins favorables : les solutions existantes conçues pour des environnements plus stables, plus robustes ou plus synchrones se révèlent inappropriées. Compenser le manque d'adéquation entre les besoins applicatifs et les caractéristiques des supports d'exécution utilisés est un défi permanent. La répartition des ressources et des calculs est en effet de moins en moins le résultat d'un choix volontaire effectué lors de la phase de conception d'une application afin de bénéficier des avantages inhérents aux architectures réparties (amélioration des performances, diminution des coûts, fiabilité, flexibilité, extensibilité, ...). Pour de plus en plus d'applications, il s'agit plutôt d'un état de fait sur lequel le concepteur d'application ne peut (ni ne veut) influer et avec lequel il doit composer même si le support d'exécution n'est pas a priori parfaitement adapté aux contraintes de l'application. Le développement récent des réseaux de pairs ne fait qu'accentuer cette tendance en limitant en particulier les connaissances que l'on peut avoir quant au nombre et à l'identité des processeurs participant au calcul réparti.

Dans ce contexte, l'action ADEPT poursuit deux objectifs complémentaires. Le premier réside dans la compréhension des structures « profondes » des calculs et des systèmes répartis. Dans ce travail de recherche à caractère fondamental, nous considérons aussi bien des environnements totalement asynchrones (où la notion de temps n'est pas présente) que des environnements synchrones (ou du moins partiellement synchrones). L'asynchronisme total (caractérisé par l'absence de borne maximale sur les temps de transfert de messages et sur les temps d'exécution des pas de calcul) est une source de difficulté notable notamment lorsque d'autres facteurs tels que l'occurrence de fautes, la mobilité et l'évolutivité du système viennent se greffer. En faisant varier les hypothèses caractérisant l'environnement servant de support à l'exécution, notre but est d'étudier différents problèmes fondamentaux afin de les comprendre et de les résoudre le plus efficacement possible. Parmi les problèmes abordés en 2002 figure en particulier : les problèmes d'accord, la tolérance aux défaillances, les critères de cohérence, la modélisation des systèmes évolutifs et la prise en compte des contraintes temporelles.

Le deuxième objectif de l'action ADEPT est de valider et de promouvoir l'utilisation des solutions algorithmiques que nous préconisons en concevant et en développant des services qui s'appuient sur nos résultats de recherche et qui en retour en aiguillonnent la problématique. Cette activité nous amène à nous intéresser d'une part aux problèmes techniques et méthodologiques posés par la conception de ces services génériques et à considérer d'autre part les problèmes liés à l'intégration de ces services dans les architectures intergicielles actuellement commercialisées. Cette activité de conception logicielle s'accompagne d'une recherche de nouveaux domaines d'application pour les services proposés. Il s'agit là d'une opportunité d'ouverture de l'action ADEPT vers des acteurs industriels et des équipes de recherche ayant parfois des problématiques de recherche éloignées. Durant l'année 2002, nous avons poursuivi la réalisation d'Eden, un outil de communication de groupe tolérant aux pannes qui incorpore certains de nos résultats de recherche sur les problèmes d'accord et la tolérance aux défaillances. Cet outil est développé de façon modulaire en s'appuyant sur un mécanisme de communication par événements et en adoptant une approche par composant.

Ces deux dimensions de l'action ADEPT (théorie et pratique) sont complémentaires et étroitement imbriquées dans notre activité de recherche, que nous menons pour une part importante, dans le cadre de programmes de coopération internationaux.

3. Fondements scientifiques

3.1. Introduction

Dans la majorité des travaux réalisés au sein de l'action ADEPT, l'environnement considéré est réparti, asynchrone et stable au sens où, à tout instant, l'identité et la localisation exacte de l'ensemble des processeurs participant au calcul sont connues de tous. Dans un tel contexte, les problèmes étudiés se ramènent à des questions d'observation et de synchronisation. Cependant, selon les services de base considérés, les concepts sous-jacents et les paradigmes utilisés pour les concevoir diffèrent sensiblement. Pour cette raison, nous présentons de façon séparée, les trois services de base autour desquelles s'articule notre activité de recherche actuelle à savoir, les problèmes d'accord, les critères de cohérence et la tolérance aux défaillances.

Comme cela a été signalé dans la section précédente, les caractéristiques de l'environnement influent fortement sur les exigences du concepteur et la nature des solutions que l'on peut apporter. Pour cette raison, notre activité de recherche prend désormais en compte deux caractéristiques additionnelles des systèmes à savoir *le temps* d'une part et *la mobilité/évolutivité* d'autre part. Dans ces deux contextes nouveaux, notre activité consiste à redéfinir des modèles, des concepts et des abstractions qui permettent de tenir compte de la spécificité de l'environnement et qui nous permettent d'élaborer une approche nouvelle pour concevoir des services de base indispensables au concepteur d'applications.

Les travaux sur l'aide à la conception des applications multimédias, qui ne se poursuivront pas dans les années à venir, sont présentés à part dans une dernière section.

3.2. Problèmes d'accord dans les systèmes répartis

Mots clés : *algorithme réparti*, *problème d'accord*, *consensus*, *validation atomique*, *diffusion atomique*, *panne franche*.

Dans de nombreuses applications, il est indispensable que les processus aient une vision unanime de la progression du calcul pour lequel ils coopèrent. Pour élaborer cette vision commune, chaque processus doit participer à l'exécution d'un protocole d'accord en fournissant au départ sa vision locale de l'état du système. Durant l'exécution du protocole d'accord, tous devront progressivement converger vers une valeur unique obtenue à partir de l'ensemble des valeurs initiales. Au sein de la classe des problèmes d'accord, le problème du consensus fait figure d'exemple. Dans ce cas particulier, la valeur unanimement décidée doit être une des valeurs proposées. Différents autres problèmes d'accord (validation atomique, diffusion atomique,...) peuvent être résolus en utilisant une solution au problème du consensus comme brique de base. Malheureusement, il a été démontré que le problème du consensus n'admet aucune solution déterministe si le système est asynchrone et non-fiable. Pour être résolu, le problème doit être affaibli (terminaison du protocole ou unicité de la décision non garantie) ou alors les hypothèses caractérisant l'environnement doivent être renforcées (rajout de propriétés de synchronie).

Dans un environnement totalement asynchrone, il est impossible de déterminer au préalable une borne maximale sur les temps de transfert d'informations entre sites distants (à cause en particulier des variations inopinées de la charge du réseau) ou une borne maximale sur les durées d'exécution des actions (à cause par exemple des fluctuations de la charge des processeurs). Lorsqu'une analyse du comportement temporel permet de connaître ces bornes, leurs valeurs sont généralement si élevées que leur exploitation au niveau des protocoles (sous forme de délai de garde par exemple) est pénalisante du point de vue des performances. D'une manière plus générale, il est risqué de développer des algorithmes répartis s'appuyant sur des bornes de temps maximales qui peuvent aisément être mises en défaut lorsque se produisent des actions imprévues ou malveillantes (comme par exemple, une saturation du réseau par des entités externes à l'application).

L'absence d'information précise et exploitable concernant la réactivité du système (processeurs et canaux de communication) s'avère être une difficulté majeure lors de la conception des applications réparties. La plupart des problèmes d'observation et de synchronisation étudiés au cours des décennies passées (contrôle de la compétition pour les ressources, détection de propriétés, ...) nécessitent des solutions complexes uniquement

lorsque le système est asynchrone. Si l'hypothèse que le système est asynchrone est attrayante (car plus générale et bien souvent plus réaliste), elle s'avère être extrêmement faible lorsque l'on suppose que le système est en même temps non fiable. Si des défaillances peuvent aussi bien affecter le réseau de communication (perte de messages, partition, ...) que les calculateurs eux-mêmes (arrêt prématuré suite à une panne matérielle ou une malveillance, ...), il a été démontré que certains problèmes n'admettent pas de solution déterministe. C'est en particulier le cas des problèmes d'accord. Intuitivement, pour résoudre des problèmes d'accord au sein d'un groupe prédéfini de processus pouvant connaître des défaillances, il faut disposer de détecteurs de défaillances performants offrant notamment des garanties quant à la véracité des suspicions de défaillances qu'ils signalent. Dans un environnement asynchrone non fiable (où il est impossible - à distance - de faire la différence entre un processus lent et un processus en panne), les détecteurs de défaillances seront toujours partiellement incorrects au sens où ils peuvent soupçonner tardivement des processus réellement défaillants et inversement soupçonner de façon erronée et transitoire des processus corrects qui sont toujours restés en activité.

Des travaux récents ont mis en évidence le lien qui existe entre les problèmes nécessitant l'obtention d'un accord (élection, diffusion ordonnée, validation atomique non-bloquante, gestion des membres d'un groupe, etc.) et le problème abstrait du consensus. De fait, ce problème élémentaire est l'objet de nombreux travaux de recherche depuis quelques années. Le résultat le plus important est malheureusement négatif [54] : ce problème fondamental ne peut pas être résolu dans des systèmes asynchrones dès lors que les processus sont susceptibles de stopper prématurément leurs exécutions. Afin de surmonter ce résultat d'impossibilité, Chandra et Toueg [52] ont augmenté le modèle asynchrone en introduisant la notion de détecteur de défaillances. Un détecteur est associé à chaque processus et est chargé de détecter les défaillances externes. Chandra et Toueg définissent huit classes de détecteurs de défaillances en caractérisant chacune d'entre elles par une propriété de complétude et une propriété d'exactitude. Une propriété de complétude définit des contraintes concernant la détection des processus réellement arrêtés tandis que la propriété d'exactitude vise à limiter les suspicions erronées que peut commettre un détecteur de défaillances.

Parmi ces classes, la classe dénotée $\diamond\mathcal{S}$ est particulièrement intéressante puisqu'il s'agit de la classe la plus faible permettant de résoudre le consensus [51]. Cette classe est caractérisée par une propriété de complétude forte (tout processus défaillant finit par être suspecté de façon permanente par tout processus correct) et une propriété d'exactitude faible inéluctable (il existe un instant à partir duquel un processus correct ne sera plus jamais suspecté par aucun processus correct). En s'appuyant sur des détecteurs de défaillances de cette classe et à condition que les canaux de communication soient fiables et qu'une majorité de processus ne subit pas de défaillances, des algorithmes déterministes permettent de résoudre le problème du consensus. Tous s'appuient sur le paradigme du coordinateur tournant mais diffèrent par leur complexité en temps et en nombre de messages. L'utilisation de tels algorithmes comme briques de base dans la résolution de problèmes d'accord, et par conséquent leur utilisation dans la gestion des groupes de processus dupliqués, est actuellement un axe de recherche important.

Un vecteur d'entrée au problème du consensus est le vecteur constitué des n valeurs proposées par les n processus. Si V est l'ensemble des valeurs qui peuvent être proposées alors il existe $|V|^n$ vecteurs d'entrée possibles au consensus. Une manière de circonvenir le résultat d'impossibilité, proposée dans [57], consiste à n'autoriser qu'un sous-ensemble S de ces vecteurs. Il devient alors possible de définir un protocole de consensus générique, paramétré par S. Ce protocole garantit la propriété d'accord du consensus quel que soit le vecteur d'entrée. La propriété de terminaison est satisfaite soit en l'absence de défaillances soit lorsque le vecteur d'entrée appartient au sous-ensemble S.

Si le problème du consensus est le représentant générique d'une large classe de problèmes d'accord, il existe également des problèmes d'accord qui ne sont pas réductibles au consensus. C'est le cas notamment du problème de l'accord sur une donnée globale et du problème de la diffusion fiable avec terminaison. En se plaçant dans un cadre synchrone et en utilisant le paradigme des protocoles à rondes (il s'agit alors de rondes synchronisées), Dolev, Reischuk et Strong [53] ont montré qu'il fallait au moins $\min(t+1, f+2)$ rondes (où t est le nombre maximal de pannes tolérées, et f le nombre effectif de pannes ayant lieu pendant l'exécution) pour parvenir à l'accord global. Dans un contexte asynchrone avec pannes franches, ces problèmes d'accord « difficiles » ne peuvent être résolus que si le modèle est augmenté avec des détecteurs parfaits. Or, dans un tel

contexte, le modèle de calcul obtenu (rondes asynchrones + détecteurs parfaits) n'est pas aussi fort que celui des rondes synchronisées. Ceci soulève une importante question, théorique mais aussi pratique : quelle est la borne inférieure du nombre de rondes nécessaires pour parvenir à l'accord global ?

3.3. Critères de cohérence dans les systèmes répartis

Mots clés : *algorithme réparti*, *causalité*, *cohérence*, *mémoire virtuellement partagée*, *entrepôt de données*.

Un objet partagé est un objet qui peut être accédé de façon concurrente par plusieurs processus. Pour des raisons de performance, les objets sont parfois dupliqués. Se pose alors le problème de maintenir une forme de cohérence entre les différentes copies d'un objet et ceci même en présence de défaillances. Ce problème se rencontre notamment lorsque l'on définit une mémoire partagée. Lorsque l'utilisateur accède à une donnée partagée, il est souhaitable que la valeur retournée lors d'une opération de lecture soit la *dernière* valeur écrite. Le choix d'un critère de cohérence permet alors de définir formellement la signification associée à l'adjectif « dernière ». Dans le cas des entrepôts de données, un accès cohérent aux données doit également être offert.

Un grand nombre d'applications parallèles utilisent un modèle de programmation fondé sur le partage d'objets qui nécessite de fournir un seul espace d'adressage pour l'ensemble des objets. De plus, le développement de l'Internet et du Web sont à la base de nouvelles applications, inimaginables il y a quelques années, qui permettent maintenant aux utilisateurs d'accéder, de partager et de manipuler des informations de plus en plus riches et complexes.

Concernant la cohérence des données manipulées, plusieurs critères de cohérence ont été proposés dans la littérature : la linéarisabilité, la cohérence séquentielle, la cohérence hybride et la cohérence causale sont parmi les plus connus. Les relations entre ces critères de cohérence doivent encore faire l'objet d'études approfondies. Pour les principaux critères, il demeure intéressant de poursuivre la recherche de protocoles performants permettant de les mettre en œuvre. Par ailleurs, il convient de proposer des mécanismes permettant de tolérer des défaillances éventuelles.

Les entrepôts de données (data warehouses) sont devenus un composant majeur des bases de données distribuées afin d'y réaliser des requêtes dynamiques d'analyse et d'audit (online analytical processing ou OLAP). D'une manière très simplifiée, un entrepôt de données maintient une vue calculée par agglomération de plusieurs bases de données distribuées. Il est crucial d'assurer la cohérence de la vue présente sur l'entrepôt de données car les sources, distribuées et asynchrones, sont continuellement modifiées. Ce problème reste encore très mal résolu du fait qu'il n'est pas encore bien spécifié et il n'en existe pas de définition précise. Les solutions qui existent sont de ce fait non prouvées à cause de l'absence de propriétés simples à satisfaire.

3.4. Tolérance aux défaillances dans les systèmes répartis

Mots clés : *algorithme réparti*, *communication de groupe*, *consensus*, *pannes franches*, *fautes byzantines*.

Dans un système réparti asynchrone, il est important de pouvoir concevoir des applications tolérant les défaillances. Le recours à la redondance (réplication active ou passive) est une technique classique permettant d'atteindre cet objectif. Le concept de groupe s'avère dans ce cas particulièrement intéressant : les différentes copies d'un serveur critique forment un groupe qui a la charge, vis à vis du monde extérieur, d'assurer le traitement des requêtes soumises et de masquer les pannes qui peuvent survenir avant ou pendant ces traitements. Développer les services offerts dans un groupe en utilisant comme brique de base une solution au problème du consensus est une approche novatrice présentant de nombreux avantages. En particulier, le résultat d'impossibilité de Fischer-Lynch-Paterson peut être circonscrit à ce niveau. Le modèle de fautes que l'on considère actuellement ne tient compte que des pannes définitives. Des modèles de fautes plus riches peuvent être considérés. Ainsi, si on considère des fautes malignes, un processus peut durant l'exécution du protocole avoir un comportement qui n'est pas conforme à sa spécification. En ce qui concerne les pannes passagères, des solutions dites autostabilisantes peuvent également être envisagées.

Considérer qu'un système réparti est asynchrone est une hypothèse attrayante (car plus générale et réaliste). Cependant, elle semble difficilement conciliable avec la prise en compte de critères de qualité de service tels

que la tolérance aux défaillances et les contraintes temps-réel. Un des axes de recherche de l'action ADEPT est consacré à la conception et au développement de services permettant de répondre efficacement à ces nouvelles exigences dans un environnement distribué asynchrone.

La sûreté de fonctionnement ne peut être garantie que pour des types de défaillances préalablement identifiés. Dans le cadre de cette activité, nous considérons les défaillances de type panne franche (crash) : chaque processus peut, soit s'exécuter correctement, soit s'interrompre brutalement (et définitivement) suite à une panne (ou une agression extérieure). A condition de pouvoir coordonner efficacement l'activité des processus dupliqués, le concept de groupe se révèle être une excellente technologie *middleware* pour concevoir des mécanismes de tolérance aux défaillances.

Dans un système réparti, un service de *communication de groupe* est un outil puissant qui permet de gérer les communications multi-point en rassemblant les processus qui sont destinataires des mêmes messages au sein d'une entité d'adressage unique : *le groupe*. Au niveau applicatif, un groupe est donc un ensemble de processus ayant un comportement similaire au sens où tous les membres du groupe traitent les mêmes requêtes. Un processus peut appartenir à un ou plusieurs groupes. De même un message peut être adressé à un ou plusieurs groupes destinataires.

Un service de communication de groupe simplifie la tâche du concepteur d'applications en lui offrant deux fonctionnalités majeures. D'une part, celui-ci a la possibilité de créer des groupes de processus dont la composition évolue dynamiquement. D'autre part, il peut définir des contraintes concernant l'ordre de livraison des messages de l'application par chaque membre du groupe.

- A tout instant, un processus peut devenir membre d'un groupe en émettant une requête *join* à destination de ce groupe. De même, il peut volontairement cesser d'être membre du groupe en émettant une requête *leave* à destination du groupe. Il peut également cesser d'être membre du groupe à la suite d'une défaillance (panne franche) dont l'occurrence n'est bien évidemment pas prévisible. Tous ces changements de la composition du groupe doivent être observés de manière cohérente par l'ensemble des membres du groupe.
- Les messages destinés à un groupe ne sont pas nécessairement reçus dans le même ordre par chacun des membres du groupe. Ceci est en particulier le cas lorsque les membres d'un même groupe sont situés sur des sites géographiquement distants. Le concepteur d'applications peut imposer que les requêtes soient délivrées à l'application dans le même ordre par chacun des membres. Cette fonctionnalité permet aux différents membres d'évoluer de façon cohérente.

Pour mettre en œuvre ce concept, il convient d'apporter des solutions efficaces à divers problèmes d'accord entre processus. En particulier, les processus appartenant à un même groupe doivent être unanimes quant à l'identité des membres qui le composent (membership) et l'ordre dans lequel les messages qui leurs sont adressés seront délivrés (diffusion ordonnée).

De nombreux protocoles ont été conçus pour tolérer uniquement les pannes franches de processus. Mais de telles défaillances ne constituent qu'un cas particulier de défaillance. La prise en compte de défaillances plus sévères (comme les omissions d'envoi ou de réception, ou les défaillances arbitraires) constitue un défi majeur. L'adoption de ces modèles de défaillances plus riches permet en particulier de tenir compte des risques d'erreurs logicielles ou des attaques intentionnelles. Deux axes de recherche sont possibles. Une première approche consiste à redéfinir des protocoles entièrement nouveaux conçus au cas par cas pour tenir compte des nouvelles formes de défaillances. Une seconde approche consiste à ajouter des composants externes qui, additionnés à un protocole tolérant les pannes franches, rendent ce protocole tolérant aux défaillances plus graves. Dans ce cas, aucune modification du protocole original n'est nécessaire. Seuls les modules externes doivent être définis.

Parmi les fautes qui peuvent affecter les systèmes répartis, l'auto-stabilisation s'intéresse aux fautes transitoires. Les fautes transitoires (ou passagères), comme par exemple une coupure de courant momentanée, peuvent affecter les variables locales des processeurs mais également l'état des registres et des liens de communications. Un système est dit auto-stabilisant si : quel que soit son état, il est capable, sans aucune

intervention extérieure, de retrouver un fonctionnement correct au bout d'un temps fini. La combinaison de cette approche avec d'autres techniques de tolérance aux défaillances (fondées sur la redondance) peut s'avérer intéressante dans certains cas.

3.5. Prise en compte du temps

Mots clés : *temps-réel*, *modèle asynchrone*, *modèle synchrone*.

Un certain nombre d'applications réparties exhibent des contraintes temporelles. L'étude de telles applications s'appuie sur la distinction que l'on fait entre le comportement du système (qui peut s'avérer être synchrone) et la façon dont celui-ci est modélisé. En s'appuyant sur un modèle asynchrone, nous apportons des solutions pour lesquelles le taux de couverture des propriétés (temporelles) est égal à 1.

Nous souhaitons pouvoir prendre en compte la notion de contrainte temps-réel et ainsi être capable de développer des applications temps réelles « critiques » ou non « critiques » dans des environnements répartis. Un problème à résoudre sera donc caractérisé à la fois par des propriété de vivacité (qui garantissent que des résultats sont produits), des propriété de sûreté (qui garantissent que les résultats produits sont corrects) et des propriété de ponctualité (qui garantissent que la production des résultats est effectuée à temps).

Cette décomposition est le fondement de l'approche d'ingénierie système prouvable que nous préconisons. Une telle approche doit permettre d'une part d'éliminer les erreurs d'interprétation lors de la phase initiale de conception qui consiste à traduire les exigences formulées en langage naturel dans une spécification de problème informatique, non ambiguë, cohérente et intrinsèquement complète. D'autre part, dans une seconde étape, cette approche doit permettre de démontrer que la spécification du système informatique proposée satisfait la spécification du problème informatique préalablement formulée. Ces deux défis (spécification correcte des exigences applicatives et obligation de preuves de conception correcte de système) sont les clés permettant d'une part de réduire de façon significative les coûts de conception, de développement et d'évolution des systèmes informatiques et d'autre part d'augmenter de façon significative la confiance que l'on place en ces derniers.

Le fait de développer des applications temps réel ne signifie pas pour autant que l'on doit nécessairement considérer un modèle de calcul synchrone. Il est important de faire une distinction entre le comportement du système et la façon dont celui-ci est modélisé. Le fait d'avoir un système synchrone et de le modéliser par un modèle de calcul asynchrone n'a rien d'aberrant. Un système peut exhiber des propriétés fortes (par exemple, le temps de tansfert des messages est borné et connu) sans que l'on souhaite les exploiter directement aux niveaux des protocoles développés (via par exemple un recours systématique à l'utilisation de délai de garde).

Au contraire, il est parfois plus avantageux de considérer un modèle de calcul asynchrone. Plus précisément, les solutions asynchrones dominent les solutions synchrones, en termes de taux de couverture des propriétés prouvées ou/et en termes de performances prouvées. Ainsi, à taux de couverture égal, les solutions asynchrones sont bien souvent plus rapides.

Par ailleurs, cette approche est particulièrement pertinente pour les systèmes de mobiles, les systèmes sans fil, les systèmes devant tolérer les intrusions, car la nature de ces systèmes impose de les concevoir dans des modèles de calcul asynchrones.

Les solutions algorithmiques que nous proposons dans un environnement asynchrone enrichi avec des détecteurs de défaillance se prête à une analyse d'ordonnançabilité. Les différents problèmes d'accord (diffusion atomique, validation atomique, ...) que l'on peut concevoir en utilisant comme brique de base le consensus, peuvent en effet avoir une durée maximale d'exécution qui est bornée à condition de pouvoir concevoir des détecteurs de défaillance de la classe $\mathcal{S}$ ou $\mathcal{P}$. Dans ces conditions, il est possible de définir une borne sur le pire cas de la complexité en temps du service de consensus.

3.6. Prise en compte de la mobilité

Mots clés : *algorithme réparti*, *mobilité*, *réseaux de pairs*, *réalité virtuelle*, *modélisation*.

Le concept de mobilité fait généralement référence à des changements de positions physiques des calculateurs (réseaux cellulaires et réseaux-ad-hoc). Ce concept peut être étendu pour prendre en compte des changements

de positions logiques et ainsi tenir compte des réseaux de pairs ou des systèmes de réalité virtuelle partagée où des changements de voisinage s'opèrent à un niveau logique sans que cela s'accompagne de déplacements physiques. D'autres similitudes entre ces différents systèmes (connexion/déconnexion, acheminement des données, partitionnement, ..) nous encourage à proposer une approche homogène étendant le concept de mobilité. Dans un contexte mobile, les modèles, abstractions et les définitions des problèmes fondamentaux tels que le consensus, l'exclusion mutuelle ou l'élection de leader doivent faire l'objet d'une redéfinition. Les solutions proposées dans un modèle réparti statique se révèle en effet être inappropriées.

Un environnement est dit mobile si les calculateurs qui le composent peuvent se déplacer physiquement tout en restant capables d'interagir et d'accéder à des données indépendamment de leurs positions géographiques. L'emploi du terme *mobilité* est donc justifié par le fait que des entités se déplacent physiquement. Les réseaux mobiles (ou sans fil) peuvent être classés en deux catégories : les réseaux avec infrastructure (réseaux cellulaires) et les réseaux sans infrastructure (réseaux ad-hoc).

Les réseaux cellulaires, plus anciens, sont les plus étudiés dans la littérature. Un système cellulaire est un ensemble d'entités mobiles et d'entités fixes. Chaque station fixe, également appelée station de base, gère une partie du réseau appelée cellule. Les entités mobiles communiquent entre elles via les stations de base. A tout moment une entité mobile peut changer de cellule et ce changement engendre des modifications au niveau des tables de routage des stations de base. Au sein d'une cellule, la communication entre les entités mobiles et la station de base se réalise à travers des liaisons sans fil. Cette interaction entre un mobile et sa station de base peut s'apparenter à une relation client/serveur classique où le client peut à tout instant changer de serveur.

Le développement des systèmes ad-hoc est plus récent. Ces systèmes se caractérisent principalement par le fait qu'ils ne reposent sur aucune structure fixe. Un système ad-hoc est un ensemble d'entités mobiles qui communiquent entre elles principalement via des ondes hertziennes (communication sans fil). Afin de pouvoir continuer à acheminer des données entre les différents hôtes du réseau, les tables de routage, gérées par l'ensemble des participants doivent systématiquement être mises à jour lorsqu'un calculateur se déplace physiquement. Dans un tel système, tout les participants ont les mêmes fonctionnalités et jouent le même rôle. En ce sens, ils sont à la fois client et serveur.

Nous souhaitons étendre le concept de mobilité. Comme nous venons de le voir, ce concept ne tient compte pour l'instant que des changements de positions physiques. Or, on peut également intégrer sous cette notion des changements de position logique ou virtuelle. L'idée est de parvenir à traiter tous ces types de changements de façon homogène. Avec cet objectif, nous nous intéressons d'une part aux réseaux de pairs (peer-to-peer) et d'autre part aux systèmes de réalité virtuelle partagée.

Les réseaux de pairs ont, jusqu'à maintenant, concerné principalement la communauté des utilisateurs d'Internet cherchant à échanger des documents multimédias (musiques, films, articles). Dans ces systèmes, les machines se comportent à la fois comme un serveur en offrant des ressources de calcul et de stockage, et comme un client en utilisant les ressources offertes par les autres machines : il s'agit donc d'une relation de parité (d'où le terme de ServEnt dans Gnutella [56]). Le premier exemple de la pertinence de cette approche a été le célèbre système Napster [55], mis en place pour la diffusion de fichiers MP3. En dépit des aspects légaux, Napster a été un grand succès : le logiciel a été téléchargé plus de 40 millions de fois !

La notion de communication sur Internet entre plusieurs serveurs n'est pas innovante. Par contre, le principe de mise à disposition des ressources dans un environnement évoluant dynamiquement (connexion/déconnexion) est nouveau et source de nombreuses difficultés. Dans ces systèmes, toutes les machines peuvent remplir le rôle de client et de serveur. Elles peuvent rentrer et sortir du système à tout moment, et donc rester inconnues d'une grande partie de ce dernier. Les ressources partagées peuvent être de différentes natures (données, espaces de stockage, puissance CPU). Le rôle de l'infrastructure système est donc de mettre en relation directe les machines recherchant des ressources (les clients) et celles disposant des ressources recherchées (les serveurs) à un instant donné.

Ainsi, deux pairs qui se trouvent géographiquement séparés par des milliers de kilomètres peuvent partager les mêmes ressources en fonction de leurs affinités. Dans ce cas, on parle de voisinage dans un sens logique. A tout moment, un participant peut décider de se déconnecter de ses voisins actuels et d'ouvrir de nouvelles

connexions ou tout simplement de ne plus jamais se reconnecter. Sans changer de position physique, il fait ainsi évoluer sa position logique en modifiant son voisinage.

A un niveau applicatif, la notion de changement de position peut également avoir un sens propre. Dans un système de réalité virtuelle partagée, où les participants peuvent également se connecter/déconnecter, on retrouve une notion de positionnement dans un monde 3D ainsi qu'une notion de voisinage proche. Là encore une harmonisation au travers un concept de mobilité étendu semble pertinente.

Notre premier constat est qu'une partie des difficultés rencontrées dans les réseaux cellulaires, les réseaux ad-hoc, les réseaux de pairs et les systèmes de réalité virtuelle partagée découlent du fait que ces systèmes sont tous confrontés aux mêmes problèmes : déconnexion des entités à tout moment, changement fréquent de topologie et risque de partitionnement. Il est donc souhaitable d'avoir une vision homogène mais pas nécessairement uniforme de ces systèmes qui sont qualifiés de purs quand tous les participants ont les même fonctionnalités ou d'hybride dans le cas contraire.

Un deuxième constat s'impose également. Il est évident que les modèles classiques proposés pour les systèmes répartis statiques sont inappropriées. De plus, les problèmes fondamentaux comme par exemple le consensus, l'exclusion mutuelle ou l'élection de leader doivent trouver de nouvelles spécifications et solutions adaptées à la mobilité du système.

3.7. Aide à la conception des applications multimédias

Mots clés : *application multimedia*, *architecture qualité de service*, *temps réel*, *allocation de ressources*.

Un nombre croissant d'applications reposent sur l'utilisation du multimedia :au sein d'une même présentation peuvent être utilisées à la fois des images animées, du son, du texte, etc. Ces applications se caractérisent par leur caractère distribué, par leur contraintes temporelles, et par la grande quantité de ressources requises. Pour tenter d'assouplir cette dernière caractéristique, un nombre croissant d'applications multimedia autorisent différents niveaux de qualité pour un même service, permettant ainsi d'adapter la qualité en fonction des ressources disponibles.

La généralisation des applications multimedia conduit inéluctablement à modifier l'infrastructure qui répartit les ressources entre les applications. Jusqu'à l'émergence du multimedia, les exigences de qualité de service des applications effectuant des transferts d'information se limitaient le plus souvent à vérifier l'intégrité des données échangées, afin d'éviter, par exemple, la corruption des fichiers transférés. Dans ce contexte, les ressources ne sont en général pas réservées, ce qui rend imprévisible la durée d'un traitement. Ce modèle, parfois qualifié de « paresseux (*best effort*) », est acceptable pour des applications où le traitement effectivement effectué est prépondérant sur les délais. *A contrario*, le multimedia se caractérise, outre l'exigence - devenue « seulement » statistique - d'intégrité des données, par le respect de garanties temporelles, ce qui nécessite une certaine disponibilité du processeur, de la bande passante, de la mémoire, etc. [50].

Le premier problème à résoudre est de vérifier la *cohérence des formats des données* échangées, afin de savoir si les différents équipements utilisés sont compatibles. Le second problème est posé par le respect des *contraintes temporelles*. Il s'agit de savoir si les délais et les fréquences d'arrivée des données sont compatibles avec les attentes des utilisateurs. Les contraintes temporelles sont de trois types. Les contraintes de synchronisation intra-flux expriment la régularité du flux en précisant le délai d'arrivée entre deux trames d'information. Les synchronisations inter-flux précisent les délais qui peuvent séparer l'arrivée des trames sur plusieurs flux liés. Enfin, le délai de bout-en-bout concerne le délai qui s'écoule entre le passage d'une trame en un point de l'application et son arrivée en un autre point.

Afin de permettre aux applications multimedia, très exigeantes en ressources, de s'exécuter avec le plus grand éventail possible de disponibilités en ressources, on a donné la faculté aux applications multimedia de fonctionner selon des modes plus ou moins dégradés. Dans un premier temps on a pu se satisfaire de solutions *ad hoc*, dans lesquelles les applications multimedia intégraient en leur sein des mécanismes d'adaptation de leur comportement en fonction des ressources disponibles. Mais ces solutions ont pour inconvénient d'être difficilement réutilisables et elles ne permettent pas une gestion coordonnée des ressources lorsque plusieurs applications s'exécutent.

Des solutions middleware - appelées *architectures qualité de service* - ont été proposées afin de décharger les applications de la gestion de l'adaptation. Pour profiter de leurs services, les applications doivent se conformer à un style d'architecture logicielle. Ce style spécifie entre autre les modalités d'interactions, l'interface des applications et le type des données pouvant être échangées.

Ces architectures qualité de service définissent une infrastructure de connexion, mais ne permettent de savoir qu'à l'exécution si une application multimedia peut s'exécuter ou non. Aussi nous sommes nous intéressés à la caractérisation de ces architectures afin de permettre de déterminer *a priori* si la combinaison d'une application multimedia et d'une AQDS est, d'une part, compatible pour le format des données échangées et, d'autre part, que cette combinaison permet bien le respect des contraintes temporelles de l'application multimedia.

4. Domaines d'application

4.1. Panorama

Mots clés : *application répartie*, *résistance aux défaillances*, *communication de groupe*, *consensus*, *hétérogénéité*, *intergiciel*.

Suite au développement récent des technologies réseaux, la conception de services nouveaux destinés à être exécutés dans des environnements distribués hétérogènes connaît un essor important dans de nombreux domaines industriels et plus particulièrement dans le domaine des télécommunications. L'émergence récente des applications de type travail coopératif est, à ce titre, un exemple significatif.

Enrichir les intergiciels (tels que CORBA) en spécifiant un ensemble de services de communication de groupe correspond donc à une attente de la part de nombreux industriels. En particulier, le concept de groupe est très utile pour mettre en œuvre des mécanismes de réplication (active ou passive) dans le but d'assurer la tolérance aux défaillances. Ce même concept peut être utilisé pour gérer la cohérence au sein d'un groupe d'entités distinctes (application coopératives, groupe de robots mobiles,...). C'est dans cette voie que nous développons actuellement des coopérations avec des partenaires industriels.

Le service de communication de groupe est un service générique dont les champs d'application sont vastes.

La tolérance aux défaillances est le domaine d'utilisation le plus couramment cité. Pour pouvoir tolérer des défaillances de type pannes franches, un serveur critique est répliqué sur plusieurs sites géographiquement distants. L'ensemble des copies constitue alors un groupe au sein duquel la panne d'une des copies n'a pas d'incidence sur l'exécution des autres copies. Dans un schéma de réplication dite *active*, toute les copies exécutent les requêtes émises par les clients. Partant de l'hypothèse que le serveur répliqué a un comportement déterministe, toutes les copies passent nécessairement par la même séquence d'états dès lors que l'on garantit que ces requêtes sont exécutées dans le même ordre par chacun des membres du groupe.

Le travail coopératif assisté par ordinateur (Computer Supported Cooperative Work) connaît depuis quelques années un développement important. L'objectif est de permettre à des groupes d'usagers de collaborer à des buts communs. Il peut s'agir par exemple de l'édition simultanée d'un même document par plusieurs co-auteurs ou du développement d'un projet (de CAO, de génie logiciel) faisant appel à la compétence de différentes équipes, chaque équipe étant elle-même constituée de plusieurs membres. La répartition géographique des différents intervenants des groupes induit naturellement une dimension répartie. Par ailleurs, la collaboration repose généralement sur l'utilisation d'objets partagés et pose inévitablement le problème de la cohérence de ces objets et de leur capacité à résister aux défaillances. Outre ces contraintes de cohérence, certaines applications coopératives procèdent de façon intensive à des changements de la composition des groupes. C'est en particulier le cas des applications s'appuyant sur la métaphore du bâtiment virtuel. Le groupe d'individus réunis au sein d'une même pièce (bureau ou salle de réunion virtuelle) constitue un groupe dont la composition évolue continuellement en fonction des arrivées et des départs. Dans un tel contexte, la gestion automatique des changements de vue qui est réalisée par le service de communication de groupe s'avère alors être un service majeur.

Un nouveau champ d'application intéressant peut être défini dans le domaine de la robotique. Considérons, par exemple, une équipe de robots qui ont pour mission de réaliser une certaine tâche. Les robots doivent naturellement collaborer d'une manière ou d'une autre afin de coordonner leurs actions pour remplir collectivement la tâche qui leur a été affecté. L'abstraction de groupe et l'utilisation d'un service de consensus semblent être a priori des approches intéressantes pour aborder les problèmes posés par ce type d'application. Il serait intéressant d'étudier les liens existant entre les problèmes soulevés par ce nouveau type d'application et les réponses apportées par des travaux réalisés dans d'autres domaines (celui des systèmes distribués avec un modèle asynchrone). Il faudrait identifier les spécificités d'un environnement constitué de robots mobiles, les caractéristiques des modèles de communication, les types de pannes, etc. pour ensuite vérifier si les solutions trouvées dans le contexte des systèmes distribués peuvent être facilement adaptées au contexte des robots mobiles et/ou si d'autres solutions spécifiques sont nécessaires. Dans ce dernier cas de figure, il faudra également étudier les conditions sous lesquelles ces solutions sont valides.

5. Logiciels

5.1. Eden : un service de communication de groupe

Participants : Emmanuelle Anceaume, Fabiola Greve, Michel Hurfin [correspondant], Jean-Pierre Le Narzul, Frédéric Tronel.

Mots clés : *applications réparties*, *résistance aux défaillances*, *temps réel*, *communication de groupe*, *consensus*, *CORBA*, *hétérogénéité*.

EDEN est un système de gestion des comportements de groupe, conçu à l'aide de ADAM, une bibliothèque de protocoles d'accord orientée composant et de EVA, une plate-forme de communication de type producteur/consommateur permettant la structuration de protocoles de haut niveau. Les prototypes EDEN, ADAM et EVA sont développés en Java et ont été utilisés dans le système OPENEDEN, une mise en œuvre de la spécification relative à la tolérance aux défaillances pour des architectures de type CORBA. Ces différents logiciels sont actuellement en cours de validation et seront prochainement diffusés librement.

5.1.1. Description de Eden

Le système EDEN offre des services de communication de groupe (construction d'un ordre total sur les requêtes adressées au groupe, gestion de la composition du groupe et de l'installation des vues, ...). Son originalité réside d'une part dans le fait que tous les services offerts sont conçus comme des instances d'un service d'accord élémentaire et d'autre part dans son architecture qui repose sur un mécanisme de communication par événements. Le système réparti servant de support à l'exécution du groupe est supposé *asynchrone* : aucune hypothèse n'est faite concernant les temps de transfert des messages ou les vitesses d'exécution des membres du groupe qui peuvent éventuellement s'exécuter sur des machines hétérogènes.

Le système EDEN est développé avec le langage de programmation par objets Java. Il repose sur l'utilisation de ADAM, une bibliothèque de protocoles d'accord orientée composant et de EVA, une plate-forme de communication de type producteur/consommateur.

- **Adam**
  La bibliothèque ADAM est conçue suivant une approche orientée composant. Dans cette approche, un composant est vu comme une entité logicielle coopérant avec ses pairs via deux types d'interface : asynchrone et synchrone. L'interface asynchrone permet à un composant de communiquer par production/consommation d'événements. L'interface synchrone autorise une communication plus classique par invocation/publication de services.
  Les avantages de l'approche par composants, à savoir principalement la réutilisation, la composition dynamique et le déploiement coïncident avec nos besoins quant à la construction de diverses abstractions d'accord toutes basées sur un service d'accord générique. Le comportement d'un

composant pouvant être configuré par le biais d'attributs, cela permet, par exemple, l'adaptation d'une instance du composant d'accord générique à une utilisation spécifique dans une application. La bibliothèque ADAM est le résultat de l'adoption de cette approche par composants pour la définition d'un ensemble d'abstractions utiles à la construction d'un système de gestion des comportements de groupe. En l'état actuel, la bibliothèque ADAM contient cinq composants réalisant (1) l'accord générique, (2) la gestion dynamique de la composition du groupe (3) la communication fiable et ordonnée dans le groupe (4) la communication synchronisée avec le changement de vue et (5) la détection des défaillances d'un site distant.
La bibliothèque ADAM est décrite en détail dans la thèse de Fabiola Greve [10].

- **Eva**
  Un protocole de communication de groupe est un exemple typique de protocole de communication spécialisé de haut niveau. La structuration par couche qui est habituellement utilisée pour spécifier les protocoles de communication élémentaires, présente de nombreux défauts lorsqu'il s'agit de concevoir des protocoles de plus haut niveau de la même façon. Pour éviter les inconvénients d'une architecture à couche tout en gardant les avantages d'une décomposition d'un problème en services de base indépendants, nous avons opté pour une architecture fondée sur le concept de notification d'événements. Les interactions entre les entités qui mettent en œuvre la fonctionnalité désirée sont obtenues via des productions d'événements (données en sortie) par des entités productrices et des consommations d'événements (données en entrée) par des entités consommatrices. Un canal d'événement se charge de diriger les événements produits vers les consommateurs intéressés. Cette approche permet de construire des applications performantes, très modulaires et reconfigurables dont les composants peuvent être exécutés en parallèle. Lors du développement d'EVA, nous avons cherché à utiliser des concepts bien établis (service de notification d'événements, *design patterns*) dont la combinaison permet de bâtir des applications de façon simple et méthodique sans pour autant sacrifier l'efficacité du code généré. Une description détaillée d'EVA et de ses principaux atouts est disponible dans [28].

5.1.2. Description de OpenEDEN

OPENEDEN désigne un prototype, fondé sur l'utilisation de EDEN, mettant en œuvre la tolérance aux défaillances dans une architecture intergicielle de type CORBA. De telles architectures connaissent un succès grandissant car elles facilitent le travail des concepteurs d'applications réparties en leur permettant de s'abstraire d'un grand nombre de problèmes (localisation, hétérogénéité, ...) qui sont pris en charge directement par l'intergiciel. Malheureusement, alors que les applications développées au dessus de ces architectures ont un besoin crucial de tolérance aux défaillances, la plupart d'entre elles n'offrent pas un tel service ; afin de pallier ce manque, il était donc intéressant de réfléchir à un moyen d'intégrer un système de groupe (mettant en œuvre la réplication) dans une architecture intergicielle.

La principale difficulté à surmonter dans la réalisation de OPENEDEN était de réussir à connecter deux « mondes » différents (CORBA et EDEN), chacun d'eux ayant été développé complètement indépendamment de l'autre. Cette connexion devait naturellement se faire, si possible, de manière non intrusive, c'est-à-dire sans modification ni de l'un ni de l'autre. Nous avons pour cela mis en œuvre une approche combinée fondée sur l'interception de requêtes et la notion de service CORBA. Une première version d'OpenEDEN a été conçue pour l'ORB Orbacus. Une autre, qui est actuellement en cours de développement, s'appuye sur l'ORB Orbus.

6. Résultats nouveaux

6.1. Problèmes d'accord dans les systèmes répartis

Participants : Emmanuelle Anceaume, Fabiola Greve, Michel Hurfin, Jean-Pierre Le Narzul, Eric Mourgaya, Achour Mostéfaoui, Philippe Raïpin Parvedy, Michel Raynal, Matthieu Roy.

Mots clés : *algorithme réparti*, *résistance aux défaillances*, *consensus*, *consensus sur k valeurs*, *validation atomique*.

Les travaux réalisés au cours de l'année 2002 sur les problèmes d'accord ont porté sur deux aspects. Le premier concerne la solvabilité du consensus (nouvelles familles de détecteurs de défaillances et utilisation de conditions sur l'ensemble des valeurs proposées). Le second concerne l'étude des problèmes d'accord dérivés (l'accord global, le consensus sur k valeurs, la validation atomique).

Si aucune hypothèse supplémentaire n'est adoptée, il est prouvé que le problème d'accord élémentaire n'admet pas de solution déterministe dans un système réparti asynchrone dès lors qu'un processus au moins est susceptible de connaître une défaillance de type panne franche. Notre axe de recherche principal est donc naturellement consacré à l'étude des solutions permettant de contourner ce résultat d'impossibilité, principalement les solutions déterministes fondées sur les détecteurs de défaillances.

6.1.1. Solution au problème du consensus

Afin de contourner le résultat d'impossibilité, Chandra et Toueg ont proposé d'enrichir le système sous-jacent par des détecteurs de défaillances. [17] constitue une introduction au concept d'oracle et plus spécialement aux détecteurs de défaillances introduit par S. Toueg et son équipe. La construction d'un protocole qui ne requiert qu'un nombre fini de messages pour résoudre le problème de la diffusion fiable uniforme sert de fil conducteur et d'illustration à cette introduction.

Contrairement à quasiment tous les travaux existants qui implémentent les détecteurs de défaillances en utilisant le temps physique (temporisateurs et délais de garde) et supposent un système partiellement synchrone, nous avons proposé dans [49] une implémentation de détecteurs de défaillances qui ne repose que sur un mécanisme simple d'appel/réponse et sur le nombre maximal de défaillances dans le système. La correction de l'implémentation suppose que l'ordre de réception des messages respecte une certaine propriété qui varie selon la classe de détecteurs visée. Dans le cas limite où le nombre de défaillances est inférieur ou égal à 1, la propriété nécessaire à l'implémentation d'un détecteur de la famille $\diamond\mathcal{S}$ se réduit à l'existence d'un canal reliant deux processus quelconques p_i et p_j tel que ce canal ne soit jamais le plus lent parmi ceux connectant p_i ou p_j aux autres processus.

Le but n'est donc pas de détecter quels sont les processus défaillants. En ce sens, ce travail se rapproche de celui présenté dans [26]. Cet article propose d'enrichir le modèle asynchrone avec un mécanisme d'adaptation aux conditions d'exécution. Ce mécanisme dénommé *oracle temporel distribué non-fiable (OTD)* capture l'état du réseau et estime un temps d'attente. Cette estimation augmente l'efficacité des protocoles par rondes du point de vue du nombre de messages collectés et de la durée de la collecte. Cet oracle n'a pas pour but de détecter les fautes, mais plutôt de borner l'attente de chaque processus le plus précisément possible. Une illustration de l'utilisation de l'OTD est proposée en spécifiant un algorithme de résolution du problème du consensus.

Nous nous sommes principalement intéressés aux solutions déterministes s'appuyant sur des détecteurs de défaillances appartenant aux classes $\mathcal{S}$ et $\diamond\mathcal{S}$. La classe des détecteurs de défaillances forts (dénotée $\mathcal{S}$) regroupe tous les détecteurs de défaillances qui ont pour propriété de suspecter les processus défaillants (propriété de complétude) mais de ne pas suspecter au moins un des processus non défaillants (propriété d'exactitude). La classe des détecteurs de défaillances faibles (dénotée $\diamond\mathcal{S}$) regroupe quant à elle tous les détecteurs de défaillances qui vérifient la même propriété de complétude mais ne satisfont la propriété d'exactitude qu'après un laps de temps indéterminé. De fait, les détecteurs de défaillances appartenant à ces deux classes sont intrinsèquement non fiables puisqu'ils peuvent suspecter arbitrairement des processus corrects. Bien que faibles, les propriétés d'exactitude caractérisant ces deux classes sont suffisantes pour permettre de résoudre le problème du consensus si au moins un processus est correct (classe $\mathcal{S}$) ou si au moins une majorité de processus est correcte (classe $\diamond\mathcal{S}$).

Nous proposons dans [15], un protocole générique qui peut s'adapter aisément aux deux classes de détecteurs de défaillances $\diamond\mathcal{S}$ et $\mathcal{S}$. Différentes instanciations du protocole général permettent de retrouver des protocoles existants (par exemple, celui de Chandra et Toueg) ainsi que de nouveaux protocoles. Cette approche, également intéressante par son aspect méthodologique, permet de comprendre les fondements

conceptuels de cette famille de protocoles. Ceci est rendu possible par la caractérisation de deux ensembles de processus qui, pour un pas d'itération donné, assurent respectivement les propriétés de vivacité et de sûreté. Cette approche permet également de montrer que, contrairement à une idée admise, les estampilles utilisées dans le protocole de Chandra et Toueg peuvent être bornées (utilisation d'une fenêtre coulissante). Par ailleurs, ce protocole permet, lorsque cela est nécessaire, de réduire le nombre d'étapes de calcul au prix bien évidemment d'un accroissement du nombre de messages échangés par étape. Le compromis qui résulte du choix d'un schéma d'échange de messages peut être modifié à chaque pas d'itération. Ainsi le nombre de messages échangés par pas de communication peut varier de O(n) (schéma centralisé) à O(n^2) (schéma totalement décentralisé).

Nous avons essayé de comprendre dans [38] la manière dont les différents oracles (par exemple, générateurs de nombres aléatoires, détecteurs de défaillances, sélecteurs de leader) permettent de circonvenir le résultat d'impossibilité associé au problème du consensus. Nous avons proposé un protocole modulaire procédant par rondes asynchrones constituées de deux phases chacune. La modularité et l'adaptativité découlent du rôle et de la nature des ces deux phases. La première requiert les services d'un ou d'une combinaison d'oracles et vise à faire converger les processus et donc à assurer la propriété de terminaison. La seconde (qui utilise les quorums) est une étape de consultation entre les processus et permet d'assurer la propriété d'accord.

Dans [27], nous proposons d'étendre l'approche basée sur les conditions, développée au sein du projet en 2001 [57], en caractérisant l'ensemble $\mathcal{V}_f^n$ de tous les vecteurs d'entrée pouvant être proposés par un ensemble de n processus. L'approche basée sur les conditions consiste en l'identification des vecteurs permettant la résolution directe (en un pas de communication) du problème du consensus dans un modèle asynchrone en dépit d'au plus f défaillances de processus. Nous nous concentrons sur tous les autres vecteurs. Parmi ceux-ci nous identifions ceux qui permettent la résolution du consensus en deux pas de communication. Et enfin, pour les autres, nous nous reposons sur un oracle distribué pour converger vers un *bon* vecteur. Nous spécifions un protocole qui utilise cette approche pour résoudre le problème du consensus très simplement et efficacement.

6.1.2. Autres problèmes d'accord

Le consensus est un problème d'accord élémentaire dont la définition simple est unanimement reconnue par tous. Il s'agit d'un problème purement théorique qui permet d'abstraire différents problèmes d'accord. La simplicité de ce problème permet, lors de son étude, de se focaliser uniquement sur le résultat d'impossibilité qui est associé à tout problème d'accord. En pratique, ceci ne signifie pas pour autant qu'un service de consensus est le dénominateur commun idéal à partir duquel on peut toujours dériver des solutions efficaces à des problèmes d'accord plus complexes. Diverses variantes au problème du consensus ont donc été explorées. Le but poursuivi est de concevoir une brique de base parfaitement adaptée au problème d'accord que l'on désire résoudre.

Nous présentons une extension du problème du consensus [11] afin d'en améliorer les performances. Cette extension, appelée Accord sur un Préfixe Uniforme, permet à tous les processus du groupe de proposer au cours d'une même exécution de ce module, un flot de messages au lieu d'un seul comme c'est le cas dans le problème du consensus. Tirant profit de ce flot de messages, le protocol construit sa valeur de décision en utilisant tous ces messages. Nous utilisons cette extension pour construire un algorithme de Diffusion Atomique Uniforme efficace et peu coûteux.

Parmi les variantes possibles au problème du consensus, nous avons étudié le problème de l'accord global : l'accord ne doit pas se limiter à une valeur proposée par l'un des processus, mais se faire sur un ensemble de valeurs, incluant les valeurs de tous les processus corrects. Dans un modèle de calcul asynchrone avec pannes franches, ce problème ne peut être résolu que si l'on dispose de détecteurs de défaillances parfaits. L'importance du problème nous avait déjà conduit à rechercher un protocole permettant des décisions au plus tôt. Ce protocole, fondé sur un modèle de rondes, atteint l'accord, dans le pire des cas, après $\min(t+1, 2f+2)$ rondes (où t est le nombre maximal de pannes pouvant se produire et f le nombre de pannes effectives) (voir le rapport d'activité 2000). Le défi qui restait ouvert était de savoir si ce nombre était minimum, sachant que de toutes façons l'accord ne peut être obtenu en moins de $\min(t+1, f+2)$ rondes (borne inférieure établie depuis plusieurs années dans le cas synchrone). En collaboration avec Carole Delporte-Gallet et Hugues Fauconnier,

du LIAFA, nous avons relevé ce défi et proposé un protocole permettant une décision, dans le pire des cas, après $\min(t+1, f+2)$ rondes [47]. Ce protocole est donc optimal en termes du nombre de rondes. De plus, ce résultat montre que l'on peut être aussi efficace dans un modèle asynchrone soumis aux pannes franches et avec détecteur parfait que dans le modèle synchrone, ce qui n'est pas trivial puisque le premier modèle est connu pour être moins puissant que le second.

Nous avons introduit récemment une nouvelle approche visant à résoudre le problème du consensus dans les systèmes répartis asynchrones. Il s'agit d'une approche fondée sur les conditions qui, lorsqu'elles sont satisfaites par le vecteur des valeurs proposées, permettent de résoudre le problème et ce même en présence de défaillances. Une condition peut être vue comme un ensemble de vecteurs de valeurs proposées par les processus qui permettent de résoudre le problème. Nous avons étendu ces travaux au cas de l'accord global (appelé aussi cohérence interactive), nous avons caractérisé les conditions qui permettent de résoudre ce problème dans la cadre d'un système asynchrone sujet à des défaillances de processus ainsi qu'à des propositions erronées. Il s'est avéré que les conditions ainsi caractérisées correspondent exactement aux codes auto-correcteurs de la théorie de l'information [32][39].

Le problème du k-consensus généralise le problème du consensus en ce sens qu'il ne limite pas l'ensemble des valeurs de décision à une seule mais à k valeurs, il est de ce fait moins dur à résoudre. Dans un système asynchrone composé de n processus et où au plus f d'entre eux peuvent être défaillants, le problème du k-consensus peut être facilement résolu si $f < k$. Il a par ailleurs été démontré qu'aucune solution déterministe n'existe lorsque $f \geq k$. Nous avons montré dans une première étude que le k-consensus peut être résolu en utilisant des détecteurs de défaillances à portée réduite.

Dans [24], nous avons démontré que, pour certaine valeur de k, le problème peut être résolu même si le nombre de défaillance est supérieur à $n/2$. Plus précisément, le nombre maximal de défaillance doit respecter la formule $f < (n+k-1)/2$. Nous avons montré qu'une décomposition du protocole sous la forme de trois modules permet d'adapter le nombre de modules exécutés en fonction de la portée du détecteur de défaillances. Cette stratégie permet de gérer un compromis entre le nombre de messages échangés et la qualité des détecteurs de défaillances utilisés.

En utilisant une approche fondée sur les conditions, nous avons défini, dans [40], des classes de conditions qui permettent de résoudre ce problème même lorsque $f \geq k$. Le protocole proposé possède la propriété d'être toujours sûr (« safe ») : lorsqu'il termine, il ne décide pas plus de k différentes valeurs même lorsque le vecteur d'entrée n'appartient pas à la condition avec laquelle le protocole a été instancié.

Dans [35], nous présentons des protocoles de validation non-bloquants s'appuyant sur des protocoles de consensus qui ont la propriété de pouvoir terminer en une seule étape de communication.

6.2. Critères de cohérence dans les systèmes répartis

Participants : Maria Gradinariu, Achour Mostéfaoui, Michel Raynal, Matthieu Roy.

Mots clés : *algorithme réparti*, *causalité*, *cohérence*, *mémoire virtuellement partagée*, *entrepôt de données*.

Les contributions dans ce domaine ont porté d'une part sur les critères de cohérence (proposition de protocole et analyse des liens entre critères de cohérence) et d'autre part sur la mise à jour incrémentale dans les entrepôts de données.

Les interactions dans le contexte des applications réparties sont modélisées par les modifications apportées aux objets partagés. Ainsi, les systèmes répartis sont censés donner des garanties concernant les valeurs retournées suite à l'accès aux données partagées. Un critère de cohérence définit d'une manière formelle les garanties offertes par le système, plus précisément les valeurs renvoyées quand l'on accède à un objet partagé. La cohérence de données dans les systèmes répartis a donné lieu à des nombreux résultats théoriques. Les particularités des systèmes mobiles font qu'il est quasiment impossible de garantir une cohérence forte. Dans [33] nous avons montré que le critère de cohérence le mieux adapté pour les systèmes mobiles est la normalité puisque c'est le seul critère qui reflète l'ordre local des opérations. Plus précisément, deux opérations concernant deux objets distincts et invoqués par deux processus différents ne sont pas ordonnés par la normalité. De plus, nous avons proposé une implémentation de la normalité dans le cas où les opérations

invoquent plusieurs objets à la fois. Notre implémentation n'est basée ni sur la diffusion atomique, ni sur l'utilisation des horloges globales.

Dans [45], nous montrons à l'aide d'un protocole simple que la cohérence séquentielle peut être vue comme une forme de linéarisabilité paresseuse.

Une approche classique au problème d'entrepôts de données (data warehouses) est d'utiliser la mise à jour incrémentale afin de réduire le trafic réseau. Malheureusement, il n'existe pas de définition précise de ce problème, nous avons donc développé dans [21][20] un modèle formel pour la gestion des mises à jour d'un tel entrepôt dans un modèle distribué asynchrone. Ce problème est d'abord défini en termes d'opérations abstraites de mise à jour, qui doivent vérifier certains critères de cohérence. Nous présentons ensuite un protocole et montrons qu'il vérifie les contraintes de cohérence et de vivacité. Nous avons ensuite présenté une version plus efficace du protocole ainsi qu'un protocole plus général (vis à vis de la forme des vues considérées). Dans notre étude, nous avons considéré un problème du domaine des bases de données auquel nous avons apporté une solution en utilisant des modèles et des outils issus du calcul réparti.

6.3. Tolérance aux défaillances dans les systèmes répartis

Participants : Maria Gradinariu, Fabiola Greve, Michel Hurfin, Jean-Pierre Le Narzul.

Mots clés : *algorithme réparti*, *tolérance aux défaillances*, *communication de groupe*, *panne franche*, *faute byzantine*, *autostabilization*.

La mise en œuvre d'un mécanisme de tolérance aux défaillances s'appuyant sur un mécanisme de communication de groupe s'est poursuivie au cours de l'année 2002. La prise en compte des fautes byzantines a constitué un second sujet de recherche. Enfin, une contribution ponctuelle dans le domaine de l'auto-stabilisation a porté sur un algorithme de routage.

Le concept de groupe est de plus en plus utilisé pour concevoir des couches logicielles permettant de supporter l'exécution d'applications réparties fiables. Ce paradigme recouvre deux services de base, à savoir, un service de gestion de la composition du groupe et un service de communication de groupe. Un groupe est un ensemble de processus coopérant à la réalisation d'une tâche commune. Étant donné que de nouveaux processus peuvent souhaiter rejoindre le groupe, que des membres du groupe peuvent souhaiter le quitter ou connaître des défaillances de type panne franche, la composition du groupe doit pouvoir évoluer de façon dynamique. L'ensemble des processus qui constituent le groupe à un instant donné est appelé la vue courante du groupe.

Dans [10], nous nous intéressons aux problèmes de la spécification et du développement d'un service de gestion de la composition du groupe ne tolérant qu'une partition primaire (une seule vue à un instant donné). Nous proposons tout d'abord une spécification du problème. Ensuite, nous présentons un protocole qui satisfait cette spécification dans un système réparti asynchrone équipé de détecteurs de défaillances. Le service de gestion de la composition du groupe via une partition primaire est obtenu en spécialisant de manière appropriée un protocole d'accord générique. Pour cela, nous avons défini un cadre général dans lequel la définition d'un problème d'accord particulier est fixée par le biais de 6 paramètres. Le protocole proposé par Chandra et Toueg pour résoudre le problème du consensus à l'aide de détecteurs de défaillance appartenant à la classe $\diamond\mathcal{S}$ constitue l'ossature du protocole générique proposé. Les extensions offertes par ce nouveau service d'accord permettent notamment aux membres d'un groupe de processus de décider unanimement sur une collection de valeurs proposées (plutôt que sur une seule des valeurs proposées) tout en autorisant les processus à effectuer de multiples propositions (plutôt qu'une seule au démarrage du protocole d'accord).

La boite à outil EVA permettant de développer tous ces protocoles sous forme de composants est décrite dans [28]. Une intégration de l'outil de communication de groupe dans un environnement CORBA est décrite dans [34].

Les pannes franches ne constituent qu'un cas particulier de comportement défaillant. Aussi, concevoir des protocoles répartis capables de résister à un comportement erroné arbitraire des processus constitue un réel défi, compte-tenu de la possibilité d'attaques malveillantes ou d'erreurs logicielles imprévues. Par conséquent, être capable de construire un protocole masquant les défaillances arbitraires au-dessus de protocoles tolérant

les pannes franches constituerait une avancée majeure dans le domaine de l'ingénierie logicielle. L'approche adoptée, modulaire, est basée sur l'encapsulation de la détection de défaillances de tout type dans des modules spécifiques. Elle peut constituer un point de départ pour la conception d'outils de transformation automatique. Dans [48], cette méthodologie a été appliquée par Jean-Michel Hélary et Roberto Baldoni au problème du consensus et au problème du calcul de l'état global. Ces travaux prolongent ceux réalisés sur le thème des fautes byzantines au cours des années précédantes.

Parmi les fautes qui peuvent affecter les systèmes répartis, l'auto-stabilisation s'intéresse aux fautes transitoires. Dans [29] nous avons proposé pour la première fois un algorithme de routage de type « wormhole » autostabilisant.

6.4. Prise en compte du temps

Participants : Emmanuelle Anceaume, Fabiola Greve, Michel Hurfin, Michel Raynal.

Mots clés : *temps réel*, *contraintes temporelles*, *modèle synchrone*, *modèle asynchrone*, *inversion de priorité*.

Les contributions dans ce domaine ont porté sur trois points. A savoir, la réalisation d'un serveur fiable soumis à des contraintes temporelles dans un environnement Byzantin, la gestion de l'inversion de priorité au sein d'un groupe de processus répliqués, et enfin la démonstration que les solutions s'appuyant sur un modèle asynchrone dominent les solutions synchrones en terme de taux de couverture des propriétés/performances.

Nous avons travaillé à la réalisation d'un serveur fiable soumis à des contraintes temporelles. Il s'agit de construire le sous-système serveur dans un système client/serveur. L'implémentation de ce serveur est basée sur l'approche connue de la triple redondance modulaire (TMR). Les trois processus serveurs coopèrent pour traiter les requêtes des clients dans le même ordre, afin de garantir la cohérence de l'état du serveur. Deux serveurs au moins sont fiables tandis que le troisième peut avoir un comportement byzantin. On suppose connues deux bornes sur les délais de communication, l'une entre les clients et les processus serveurs (D), l'autre entre deux processus serveurs (d). Les clients sont sujets aux pannes franches mais peuvent émettre des requêtes invalides. Un serveur correct doit donc vérifier la validité d'une requête avant de la traiter. A cause du grand nombre de clients, une telle vérification peut s'avérer coûteuse en temps, d'autant qu'elle doit être effectuée par tous les serveurs corrects (au moins deux). La conception du TMR préconisée ici vise à alléger cette charge en utilisant une technique d'ordonnancement par correspondance. Mais afin de rester compatible avec les contraintes de mémoire des serveurs, cette technique impose une borne de temps Σ : aucune requête de client ne peut rester plus de Σ unité de temps dans la mémoire locale d'un processus serveur. Le problème abordé dans ce travail est nouveau. Il s'agit de concevoir un protocole qui garantit le traitement de toutes les requêtes par les processus serveurs corrects dans un ordre identique, en dépit de l'effet combiné de la borne Σ, du comportement byzantin possible d'un des processus serveurs, et des pannes franches possibles des clients. Deux résultats ont été obtenus. Le premier est un protocole d'ordonnancement qui suppose $\Sigma > D + 3d$. Le deuxième est un résultat d'impossibilité, qui stipule qu'aucun protocole d'ordonnancement n'est possible lorsque $\Sigma < D$ [30]. Des avancées récentes nous permettent d'espérer un affinement de ces bornes à la valeur commune $D + 2d$.

Par ailleurs, nous avons considéré le problème de l'inversion de priorité au sein d'un groupe de processus [19]. À l'origine l'inversion de priorité a été définie dans le contexte des systèmes non-répliqués. Nous étendons cette notion d'inversion de priorité locale au contexte d'un groupe de processus répliqués. Nous présentons ensuite les propriétés que doit garantir un protocole d'ordonnancement pour assurer un ordre total sur l'exécution de requêtes soumises par des clients tout en évitant l'inversion de priorité de groupe. Ces propriétés sont implantées dans un système asynchrone temporisé augmenté d'un détecteur de défaillance non fiable de la classe $\diamond\mathcal{S}$. La solution que nous proposons permet de répliquer un serveur critique en garantissant d'une part que l'exécution des requêtes est cohérente et d'autre part qu'elle est prédictible. Par conséquent, le protocole d'ordonnancement que nous décrivons est une brique de base pour le développement d'applications temps réel tolérantes aux fautes évoluant dans un modèle asynchrone temporisé.

En ce qui concerne la prise en compte de contraintes temps-réel, nous avons entamé une collaboration forte avec Gérard Le Lann (Inria Rocquencourt). Notre constat est que les solutions s'appuyant sur un modèle de

calcul asynchrone dominent les solutions synchrones, en termes de taux de couverture des propriétés prouvées ou/et en termes de performances prouvées. Par exemple, à taux de couverture égal, les solutions asynchrones sont plus rapides.

6.5. Prise en compte de la mobilité

Participants : Emmanuelle Anceaume, Maria Gradinariu, Eric Mourgaya, Gwendal Simon, Matthieu Roy.

Mots clés : *mobilité*, *dissémination d'information*, *réseau de pairs*.

Les contributions dans ce domaine ont porté sur la proposition d'un nouveau schéma de dissémination de l'information de type diffusion/souscription. L'intérêt d'un tel modèle réside dans sa simplicité d'implémentation, son coût de maintenance nul ce qui permet de faire face à la mobilité constante du réseau et à son extensibilité illimitée.

6.5.1. Dissémination d'information

Le modèle fréquemment utilisé pour la dissémination des informations dans un système mobile est l'inondation. Le principal inconvénient de cette technique est l'engorgement - le réseau est saturé et il y a un fort risque de perte de l'information. Un autre modèle communément utilisé est le multicast. Les solutions proposées jusqu'à présent introduisent un surcoût lié à la maintenance des arbres de diffusion dans un monde complètement dynamique.

Dans [23] nous avons proposé un nouveau schéma de dissémination de type "diffusion souscription" qui n'introduit aucun surcoût au niveau de la maintenance et qui est adapté au départs et arrivées intempestives. Notre schéma est construit sur les bases d'une orientation acyclique du réseau. Ainsi, les noeuds puits sont les seuls noeuds qui peuvent diffuser des informations. De plus, nous avons proposé un modèle pour l'organisation d'un réseau mobile. Plus précisément, les entités sont organisés en couches logiques, chaque couche étant caractérisée par son propre graphe de communication et son orientation. Chaque couche peut être vue comme un médium de transmission pour des informations ayant certaines caractéristiques en commun. Une entité mobile a la possibilité de se connecter à plusieurs couches à la fois. Ce modèle permet la modélisation des systèmes où les souscriptions se réalisent en fonction des thèmes ou bien, des souscriptions basées sur le contenu des informations.

6.5.2. Réalité virtuelle dans les systèmes peer-to-peer

Les implémentations actuelles pour la réalité virtuelle partagée sont basées sur l'utilisation des serveurs centraux. Dans [36], nous avons présenté une architecture totalement distribuée pour un système de réalité virtuelle partagée. Chaque entité participe à l'élaboration d'une topologie adaptée à la réalité virtuelle (connexions entre entités proches) supportant les déplacements et la volatilité des entités. En outre, un mécanisme de négociation facilitant les échanges multimédias a été proposé. Il tient compte non seulement des contraintes des mondes virtuels, mais encore de la capacité des intervenants.

6.6. Aide à la conception d'applications multimedia

Participants : Emmanuelle Anceaume, Erwan Demairy.

Mots clés : *application multimedia*, *architecture qualité de service*, *temps réel*, *allocation de ressources*.

Les travaux réalisés ont porté sur l'aspect « dynamique » d'une application multimedia. Plus précisément, nous avons conçu un algorithme permettant une répartition dynamique des ressources entre les applications qui s'exécutent sur une architecture qualité de service donnée.

Cette année a été consacrée d'une part à la mise en œuvre d'une plate-forme permettant de valider notre approche ; et d'autre part à la rédaction de la thèse correspondant à ce travail.

Nos travaux s'inscrivent dans le cadre de l'aide à la conception des applications multimedia. Nous nous intéressons plus particulièrement aux applications multimedia capables d'adaptation entre différents niveaux de qualité. A partir des algorithmes statiques de vérification de la cohérence des formats échangés et du respect des contraintes temporelles, nous avons traités le problème de la répartition des ressources entre

applications. Nous supposons que ces applications s'exécutent sur une architecture qualité de service donnée. Cet algorithme est utilisé par l'environnement pour piloter une AQDS au moment de l'admission ou du retrait d'une application en permettant une redistribution des ressources entre les différentes applications. L'environnement agit comme une sur-couche d'une architecture de qualité de service et permet d'obtenir une qualité « optimale » ; le critère d'optimalité étant précisé ultérieurement.

La plate-forme constituée pour valider cette approche se décompose en deux parties : le premier algorithme est destiné à être utilisé dans un environnement d'aide à la conception des applications multimédias. Nous avons utilisé comme base de cet environnement OLAN, un système de description d'architecture logicielle créé au sein du projet SIRAC (INRIA). Dans cet environnement, on dispose d'un ensemble d'éléments logiciels, qui peuvent être des éléments simples (composants ou connecteurs) ou bien des éléments hiérarchiques. À chaque élément simple sont associés par son programmeur les différents niveaux de qualité auquel il peut s'exécuter. Chacun de ces niveaux conditionne la qualité du service fourni, le comportement temporel et la consommation en ressources de l'élément. L'architecte logiciel spécifie la topologie de son application, c'est-à-dire la manière de connecter les éléments pour obtenir l'application désirée. L'algorithme statique est ensuite capable de déterminer à partir de cette description quels sont les formats de données échangés au sein de l'application qui peuvent être utilisés de façon à ce que celle-ci fonctionne correctement. Sont également vérifiées les contraintes temporelles pouvant être fournies. Tout ceci permet d'obtenir finalement les niveaux de qualité que peut fournir l'application.

Dans un deuxième temps, on utilise la description de l'application avec ses niveaux de qualité et la consommation en ressources qui y est associée afin de permettre la répartition des ressources entre les applications. À chaque application est ajoutée une information sur la priorité qui lui est accordée dans le système. L'algorithme de répartition dynamique collabore avec le mécanisme d'adaptation de l'AQDS employée pour modifier lorsque c'est nécessaire le niveau de qualité fourni par une application donnée. Le mécanisme qui permet d'adapter les applications est construit par extension de l'AQDS QUO.

Il existe des interactions entre ces deux composantes : la partie statique agit comme un client du mécanisme dynamique et gère les demandes d'admission, de retrait ou de changement de priorité pour chacune des applications.

La thèse couvrant ce travail a été soutenue le 18 novembre 2002 [9].

7. Contrats industriels

7.1. ACI GénoGRID

Participants : Emmanuelle Anceaume, Michel Hurfin, Eric Mourgaya, Jean-Pierre Le Narzul, Julien Pley, Philippe Raïpin Parvédy.

Mots clés : *grille de calcul*, *placement de tâches*, *équilibrage de charge*, *sûreté de fonctionnement*.

Dans le cadre du programme ACI GRID (Actions Concertées Incitatives - Globalisation des Ressources Informatiques et des Données) financé par le ministère de la recherche, le projet GénoGRID qui implique plusieurs équipes de recherche (notamment les équipes ADEPT et SYMBIOSE de l'Irisa) a démarré fin décembre 2001 pour une période de trois ans. L'objectif global de ce projet, qui est géré par Dominique Lavenier, est de mettre en œuvre une grille de calcul qui fédérera des ressources de calcul disponibles dans le grand ouest de la France et sera exploitée par une communauté de biologistes. Les applications ciblées nécessitent des calculs intensifs qui peuvent se décomposer en une multitude de calculs indépendants les uns des autres.

Dans ce cadre, la contribution de l'action ADEPT porte sur la définition et la conception d'un mécanisme de placement des tâches ("resources management") et d'équilibrage de charge ("load balancing") qui soit tolérant aux défaillances. Un biologiste doit pouvoir soumettre un traitement sans avoir à se soucier de la sélection des machines qui seront en charge de traiter sa demande. Bien évidemment, cette transparence doit être maintenue même en cas d'occurrences de défaillances (panne de machine, ralentissement des communications,

partitionnement du réseau, ...) ou lorsque le parc de machines évolue volontairement (ajout ou retrait de machines par les administrateurs pour des raisons de maintenance ou de mise à disposition).

Le choix des machines sélectionnées pour effectuer un traitement peut être fait par un processus unique (approche centralisée) ou au contraire être le résultat d'une concertation entre les différents sites susceptibles d'exécuter le traitement. Dans un environnement non fiable où des défaillances peuvent se produire, une approche centralisée n'est pas judicieuse. La répartition du rôle de coordinateur de la grille de calcul entre les différents composants de la grille permet de mieux résister aux défaillances qui sont susceptibles de se produire mais pose naturellement des problèmes de cohérence entre les différents coordinateurs. Dans le cadre de cette ACI, nous souhaitons exploiter les fonctionnalités offertes par le logiciel Eden qui est développé au sein de l'action ADEPT. Ce logiciel permet de prendre des décisions unanimes au sein d'un groupe de machines, dont la composition peut évoluer dynamiquement. Il peut être utilisé pour mettre en œuvre différentes politiques d'ordonnancement.

Une description plus complète des objectifs de cette ACI a été publiée dans les actes du congrès [37].

7.2. Contrat ags : Architectures Génériques pour le Spatial

Participants : Emmanuelle Anceaume, Michel Hurfin.

Mots clés : *systèmes répartis asynchrones*, *sûreté de fonctionnement*, *temps réel*.

En collaboration avec G. Le Lann (Inria Rocquencourt), C. Gallet-Delporte (LIAFA - Paris VII) et H. Fauconnier (LIAFA, Paris VII), nous menons une étude ayant pour thème « les architectures génériques pour les futurs systèmes de satellites ». Ce travail, qui a débuté au début de l'année 2002, est financé par le Cnes. La proposition d'une architecture répondant aux besoins exprimés lors de la phase de capture des exigences doit être effectuée au début de l'année 2003.

Le premier objectif de cette étude est de concevoir un support intergiciel permettant de rendre la conception des applications indépendante de l'environnement servant de support à l'application. Ce support qui doit prendre en compte les spécificités de ce secteur industriel doit également faciliter le passage d'une approche de conception plutôt centralisée traditionnellement employée à une approche répartie. Le temps réel et la sûreté de fonctionnement (en présence notamment de fautes byzantines) sont les deux exigences majeures dont nous devons tenir compte.

7.3. Contrat speeral

Participants : Emmanuelle Anceaume, Maria Gradinariu, Michel Hurfin, Eric Mourgaya, Matthieu Roy.

Mots clés : *réseaux de pairs*, *recherche de ressources*, *modelisation*, *application P2P*.

Ce contrat de recherche externe financé par France Télécom R&D est actuellement en cours de notification (durée 30 mois). Partant du constat que les modèles et protocoles conçus pour des environnements répartis traditionnels ne sont pas adaptés au cas des réseaux de pairs, l'objectif de ce contrat est d'étudier les caractéristiques de ces réseaux afin de proposer des modèles et des protocoles plus appropriés. Les travaux menés sont en connexion directe avec notre activité de recherche sur le thème de la mobilité.

8. Actions régionales, nationales et internationales

8.1. Actions nationales

8.1.1. GDR ARP (Architecture, Réseaux et Parallélisme) du CNRS

Participants : Achour Mostéfaoui, Michel Raynal.

Le GDR ARP, qui est coordonné par Luc Bougé, est articulé autour de trois pôles. L'un d'entre eux est le pôle « Réseaux et Systèmes » dont Michel Diaz (LAAS) assure la responsabilité. Ce pôle intègre le thème « Algorithmes distribués pour les systèmes répartis » auquel nous participons.

8.2. Actions européennes

8.2.1. Cabernet

Participant : Michel Raynal.

Michel Raynal participe au projet Esprit Cabernet (*Network of Excellence in Distributed Dependable Computing Systems*) qui a été créé en 1991.

8.3. Actions internationales

8.3.1. Brésil (Université fédérale de Bahia)

Participants : Fabiola Greve, Michel Hurfin, Jean Pierre Le Narzul, Michel Raynal.

Ce projet de coopération avec l'Université Fédérale de Bahia (Salvadore, Brésil) est financé dans le cadre des programmes de coopération INRIA/CNPq. Le projet qui est géré par Raimundo Macedo côté Brésilien et par Michel Hurfin côté Français, a débuté en 1999 et s'achève en 2002. L'objectif du projet est d'étudier et de concevoir des mécanismes de duplication active s'appuyant sur des services de communication de groupe afin de tolérer des fautes (de type panne franche) dans des systèmes répartis asynchrones. La mise en œuvre de ces services nécessite d'apporter une solution générique et efficace aux problèmes d'accord entre processus. Dans le cadre de cette coopération, Jean-Pierre Le Narzul s'est rendu en Mai 2002 à l'Université Fédérale de Bahia pour y donner un séminaire sur la "tolérance aux défaillances dans CORBA" et pour assurer deux cours (CORBA) pour des étudiants en Mestrado Professional (équivalent DESS). Durant ce séjour, il a participé à la conférence SBRC (Brazilian Symposium on Computer Networks) [34][35] dans laquelle l'Université de Bahia est très impliquée. En juin 2002, trois membres de l'Université de Bahia sont venus à l'IRISA afin notamment de participer à la journée de présentation des résultats du projet (le 13 juin 2002 à l'IRISA), une manifestation qui est organisée chaque année dans l'un ou l'autre des pays et qui est ouverte à tous.

8.3.2. Chine (Southeast University)

Participants : Michel Hurfin, Jean-Pierre Le Narzul, Philippe Raipin Parvedy.

Un projet de coopération intitulé "Fault Tolerant Corba" a été sélectionné par le LIAMA (Laboratoire franco-chinois d'informatique, d'automatique et de mathématiques appliquées) pour bénéficier d'un financement durant les années 2000-2002. Cette coopération qui implique l'université du sud-est à Nankin est gérée par Yun Wang côté chinois et Michel Hurfin côté français. L'objectif est d'intégrer les mécanismes de communication de groupe proposés par l'action ADEPT au sein d'un ORB développé par l'équipe chinoise en respectant les spécifications CORBA concernant la tolérance aux défaillances récemment adoptées par l'OMG. En 2002, cette coopération s'est concrétisée par une nouvelle publication [19]. A noter que la coopération avec l'université du Sud-Est à Nankin se poursuivra en 2003 via la venue en post-doctorat d'un étudiant de cette université, Ma Xiaojun, qui bénéficiera d'une bourse du ministère de la recherche.

8.3.3. USA (Université de Santa Barbara)

Participants : Achour Mostéfaoui, Matthieu Roy, Michel Raynal.

Un projet de coopération avec les professeurs D. Agrawal et A. El Abbadi de l'Université de Santa Barbara en Californie, sur les problèmes liés à la mise à jour des entrepôts de données (data warehouses), a été accepté par la NSF et l'INRIA et bénéficie, depuis juillet 2001, d'un financement conjoint de 3 ans. Dans le cadre de cette coopération, des études ont été menées en 2002 sur la mise à jour d'entrepôts de données dans un modèle distribué asynchrone [21][20].

9. Diffusion des résultats

9.1. Actions d'enseignement

La majorité des membres permanents de l'action ADEPT sont enseignants-chercheurs. A ce titre, ils participent à bon nombre d'enseignements de second et de troisième cycles et certains assurent la responsabilité

de filière d'enseignement. Achour Mostefaoui est responsable de la filière Langages et Systèmes Informatiques du DIIC (Diplôme d'Ingénieur en Informatique de l'IFSIC) et Jean-Pierre Le Narzul est responsable d'un certain nombre de modules d'enseignement au sein du département RSM de l'ENST Bretagne.

En 2002, les membres de l'action ont assuré plusieurs enseignements liés à leur thèmes de recherche :

- dans le DEA d'informatique de Rennes, Michel Raynal dispense un cours sur les algorithmes et systèmes répartis (21 heures) ;
- dans le DIIC 3e année (filières LSI et ARC), Michel Raynal dispense un cours sur les algorithmes et les systèmes répartis (30 heures) ;
- dans le DESS ISA, Michel Raynal donne un cours sur les systèmes transactionnels, la cohérence et la sécurité des données ;
- à l'ENSTBr (antenne de Rennes), Jean-Pierre Le Narzul assure plusieurs cours en Formation Initiale et en Formation Continue sur les systèmes répartis et les langages objet ;
- à l'ENSTBr (antenne de Rennes), un cours sur l'algorithmique répartie est assuré par Michel Raynal ;
- à l'ENSTBr (Brest), Michel Hurfin et Michel Raynal dispensent respectivement un cours sur la tolérance aux défaillances et l'algorithmique répartie (6 heures chacun) ;
- à Supelec, Michel Hurfin dispense un cours sur les systèmes et l'algorithmique répartis (9 heures).
- à l'Université Fédérale de Bahia, Jean-Pierre Le Narzul a dispensé deux cours de présentation de CORBA pour des étudiants en Mestrado Professional (6 heures).
- Achour Mostéfaoui a dispensé un cours sur l'algorithmique répartie (15 heures) dans le cadre du DEA MILS (Modélisation et Ingénierie du Logiciel Scientifique) de l'Université de Beyrouth (Liban).
- Achour Mostéfaoui a dispensé un cours sur les systèmes répartis dans le cadre du DESS NTIM de l'Université de Cocody à Abidjan.
- Michel Raynal a donné un cours dans le cadre du DEA de l'Université de Yaoundé (Cameroun)

9.2. Présentation de travaux

Les membres de l'équipe ont participé à diverses conférences et *workshops* (se reporter à la bibliographie pour en avoir la liste).

9.3. Animation de la communauté scientifique

Michel Raynal est membre du comité de lecture des revues suivantes *Foundations of Computer and Decision Sciences*, *Journal of Computer Systems Science and Engineering* et *Journal of Distributed Systems Engineering*.

Depuis 2002, Michel Raynal est président du « steering committee » du symposium « Distributed Computing ». Il fait également partie du bureau du TCDP (Technical Committee on Distributed Proceessing) de l'IEEE Computer Society.

En 2002, Michel Raynal a été impliqué dans les conférences suivantes :

- membre du comité de programme de
  IEEE Int. Conf. on Dependable Systems and Networks (DSN'02),
  Washington D.C., USA, juin 2002.
- Co-président du comité de programme de
  22th IEEE Int. Conf. on Distributed Computing Systems (ICDCS'02),
  Vienne, Autriche, juillet 2002.
- Co-président de
  5th IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing (ISORC'02),
  Washington DC, USA, avril 2002.

- membre du comité de programme de
  6th Workshop on Fault-Tolerance in Parallel and Distributed Systems,
  For-Lauderdale, Floride, USA, avril 2002.
- membre du comité de programme de
  First Eurasian Conference on Advances in Information and Communication Technology (EURASIA-ICT'02),
  Shiraz, Iran, octobre 2002. (LNCS #2510).
- membre du comité de programme de
  2nd ACM Workshop on Principles of Mobile Computing,
  Toulouse, France, octobre 2002.
- membre du comité de programme de
  5th IEEE Int. Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02),
  Pékin, Chine, octobre 2002.
- membre du comité de programme de
  21th IEEE Symposium on Reliable Distributed Systems (SRDS'02),
  Osaka, Japon, octobre 2002.
- membre du comité de programme de
  9th IEEE Pacific Rim Int. Symposium on Dependable Computing (PRDC'2002),
  Tsukuba, Japon, décembre 2002.

En 2003, Michel Raynal sera impliqué dans les conférences suivantes :

- membre du comité de programme de
  8th IEEE Int. Workshop on Object-Oriented Real-time Dependable Systems (WORDS'03).
  Guadalaraja, Mexique, janvier 2003.
- membre du comité de programme de
  6th IEEE Int. Symp. on Autonomous Decentralized Systems (ISADS'03),
  Pise, Italie, avril 2003.
- membre du comité de programme de
  2nd IEEE Int. Symposium on Network Computing and Applications,
  Cambridge, Massachusetts, USA, avril 2003.
- **Program Co-Chair**, co-président du comité de programme de
  9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03),
  San Juan, Porto Rico, USA, mai 2003.
- Co-président du comité de liaison internationale de
  23th IEEE Int. Conf. on Distributed Computing Systems,
  Providence, Rhode Island, USA, mai 2003.
- membre du comité de programme de
  First Latin-American Symposium on Dependable Computing (LASDC'03),
  São Paulo, Brésil, octobre 2003.

Emmanuelle Anceaume a été membre du comité de programme du *22th IEEE Int. conf. on Distributed Computing Systems (ICDCS-2002)* qui s'est déroulé à Vienne, en Autiche, en juillet 2002.

Maria Gradinariu a été membre du comité de programme du *Int. Workshop on Self-Repairing and Self-Configurable Distributed Systems (RCDS 2002)* qui s'est déroulé à Osaka au Japon en octobre 2002.

Michel Hurfin a été membre du comité de programme du *Int. Workshop on Aspect Oriented Programming for Distributed Computing Systems - Distributed Auto Adaptive Systems (AOPDC-DAAS)* qui s'est déroulé à Vienne en juillet 2002.

Michel Hurfin est membre du comité de programme du *3rd Int. Workshop on Distributed Auto-adaptive and Reconfigurable Systems (DARES)* qui se déroulera à Providence (RI) en mai 2003.

Achour Mostéfaoui est membre du comité de programme du *9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)* qui se déroulera à Porto Rico en mai 2003.

Achour Mostéfaoui est membre du comité de programme du *6th Int. Symposium on Programming and Systems (ISPS'03)* qui sera organisé en mai 2003 à Alger.

Emmanuelle Anceaume et Michel Hurfin ont mis en place, organisé et animé le Séminaire Réseaux et Systèmes (http://www.irisa.fr/adp/srs/srs.html). Depuis septembre 2002, Michel Hurfin n'exerce plus cette fonction qui est désormais assurée par Bruno Tuffin (projet Armor) en collaboration avec Emmanuelle Anceaume.

10. Bibliographie

Bibliographie de référence

[1] O. BABAOGLU, E. FROMENTIN, M. RAYNAL. *A Unified Framework for Expressing and Detecting Run-Time Properties of Distributed Computations.* in « Journal of Systems and Software, Numéro spécial sur Software Engineering for Distributed Computing », numéro 3, volume 33, juin, 1996, pages 287-298.

[2] M. HURFIN, A. MOSTÉFAOUI, M. RAYNAL. *A Versatile Family of Consensus Protocols Based on Chandra-Toueg's Unreliable Failure Detectors.* in « IEEE Transactions on Computers », numéro 4, volume 51, avril, 2002, pages 395-408.

[3] M. HURFIN, N. PLOUZEAU, M. RAYNAL. *Detecting Atomic Sequences of Predicates in Distributed Computations.* in « Proc. of the ACM Conference on Parallel and Distributed Debugging », pages 32-42, San Diego, Californie, mai, 1993, Reprinted in SIGPLAN Notices, vol. 28,12, décembre 1993.

[4] J. M. HÉLARY, A. MOSTEFAOUI, M. RAYNAL. *A general scheme for token and tree based distributed mutual exclusion algorithm.* in « IEEE Transactions on Parallel and Distributed Systems », numéro 11, volume 5, novembre, 1994, pages 1185-1196.

[5] A. MOSTÉFAOUI, S. RAJSBAUM, M. RAYNAL. *Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems.* in « Proc. of the 33rd ACM Symposium on Theory of Computing (STOC'01) », Heraklion, Crète, Grèce, juillet, 2001.

[6] M. RAYNAL, A. SCHIPER, S. TOUEG. *The Causal Ordering Abstraction and a Simple Way to implement it.* in « Information Processing Letters », volume 39, septembre, 1991, pages 343-351.

[7] M. RAYNAL, M. SINGHAL. *Logical Time : Capturing Causality in Distributed Systems.* in « IEEE Computer », numéro 2, volume 29, février, 1996, pages 49-57.

[8] A. SCHIPER, M. RAYNAL. *From group communication to transactions in distributed systems.* in « Communications of the ACM », numéro 4, volume 39, avril, 1996, pages 84-90.

Thèses et habilitations à diriger des recherche

[9] E. DEMAIRY. *Aide à la conception des applications multimédia.* thèse de doctorat, MATISSE, novembre, 2002.

[10] F. GREVE. *Réponses efficaces au besoin d'accord dans un groupe.* thèse de doctorat, MATISSE, novembre, 2002.

Articles et chapitres de livre

[11] E. ANCEAUME. *Efficient Solution To Uniform Atomic Broadcast.* in « International Journal of Foundations of Computer Science (IJFCS) », numéro 5, volume 13, octobre, 2002, pages 695-717.

[12] E. ANCEAUME, J.-M. HÉLARY, M. RAYNAL. *A Note on the Determination of the Immediate Predecessors in a Distributed Computation.* in « International Journal of Foundations of Computer Science (IJFCS) », numéro 6, volume 13, décembre, 2002, pages 865-872.

[13] R. BALDONI, M. RAYNAL. *Fundamentals of Distributed Computing : A Practical Tour of Vector-Clock Systems.* in « IEEE Distributed Systems Online », numéro 2, volume 3, 2002, pages 1-18, http://www.computer.org/dsonline.

[14] J. BEAUQUIER, M. GRADINARIU, C. JOHNEN, J. DURAND-LOSE. *Token based self-stabilizing uniform algorithms.* in « Journal of Parallel and Distributed Computing (JPDC) », numéro 5, volume 62, mai, 2002, pages 899-921.

[15] M. HURFIN, A. MOSTÉFAOUI, M. RAYNAL. *A Versatile Family of Consensus Protocols Based on Chandra-Toueg's Unreliable Failure Detectors.* in « IEEE Transactions on Computers », numéro 4, volume 51, avril, 2002, pages 395-408.

[16] J.-M. HÉLARY, A. MOSTÉFAOUI, M. RAYNAL. *Interval Consistency of Asynchronous Distributed Computations.* in « Journal of Computer and System Sciences », numéro 2, volume 64, 2002, pages 329-349.

[17] A. MOSTÉFAOUI, E. MOURGAYA, M. RAYNAL. *An Introduction to Oracles for Asynchronous Distributed Systems.* in « Future Generation Computer Systems (Special issue on Parallel Computing Technologies) », numéro 6, volume 18, mai, 2002, pages 757-767.

[18] F. TORRES, M. AHAMAD, M. RAYNAL. *Real-Time Based Consistency Models for Distributed Objects.* in « Journal of Computer Systems Science and Engineering », numéro 2, volume 17, 2002, pages 133-142.

[19] Y. WANG, E. ANCEAUME, F. BRASILEIRO, F. GREVE, M. HURFIN. *Solving the Group Priority Inversion Problem in a Timed Asynchronous System.* in « IEEE Transactions on Computer. Special Issue Asynchronous Real-Time Disttributed Systems », numéro 8, volume 51, août, 2002, pages 900-915.

Communications à des congrès, colloques, etc.

[20] D. AGRAWAL, A. EL ABBADI, A. MOSTÉFAOUI, M. RAYNAL, M. ROY. *The Lord of the Rings : Efficient Maintenance of Views at Data Warehouses.* in « Proc. of the 16th Int. Symposium on Distributed Computing (DISC-02) », série LNCS, numéro 2508, Springer Verlag, pages 33-47, Toulouse, France, octobre, 2002.

[21] D. AGRAWAL, A. EL ABBADI, A. MOSTÉFAOUI, M. RAYNAL, M. ROY. *Towards a Formal Model for View Maintenance in Data Warehouses.* in « Proc. of the 21th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC-2002) », pages 129, Monterey, Californie, juillet, 2002, Brief announcement.

[22] M. AHAMAD, M. RAYNAL. *Ordering vs Timeliness : two Facets of Consistency ?.* in « Proc. of the 1st Int. Workshop on Future Directions in Distributed Computing (FuDiCo 2002) », éditeurs O. BABAOGLU, K. BIRMAN, K. MARZULLO., pages 98-104, Bertinoro, Italie, juin, 2002.

[23] E. ANCEAUME, A. DATTA, M. GRADINARIU, G. SIMON. *Publish/Subscribe Scheme for Mobile Networks.* in « Proc. of the 2nd Int. Workshop on Principles of Mobile Computing (POMC 02) », pages 74-81, Toulouse, France, octobre, 2002.

[24] E. ANCEAUME, M. HURFIN, PH. RAÏPIN PARVÉDY. *An Efficient Solution to the k-set Agreement Problem.* in « Proc. of the 4th European Dependable Computing Conference (EDCC-4) », série LNCS, numéro 2485, Springer Verlag, pages 62-78, Toulouse, France, octobre, 2002.

[25] E. ANCEAUME, J.-M. HÉLARY, M. RAYNAL. *Tracking Immediate Predecessors in Distributed Computations.* in « Proc. of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA-02) », pages 210-219, Winnipeg, Canada, août, 2002.

[26] E. ANCEAUME, E. MOURGAYA. *Unreliable Distributed Timing Scrutinizer : Adapting Asynchronous Algorithms to the Environment.* in « Proc. of the 5th IEEE Int. Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2002) », pages 70-77, Washington D.C., avril, 2002.

[27] E. ANCEAUME, E. MOURGAYA, PH. RAÏPIN PARVÉDY. *Converging Towards Conditions.* in « Proc. of the 6th Int. Conference on Principles of Distributed Systems (OPODIS'02) », Reims, France, décembre, 2002.

[28] F. BRASILEIRO, F. GREVE, M. HURFIN, J.-P. LE NARZUL, F. TRONEL. *Eva : an Event-Based Framework for Developing Specialised Communication Protocols.* in « Proc. of the 1st IEEE Int. Symp. on Network Computing and Applications (NCA 2001) », pages 108-119, Cambridge, Massachusetts, février, 2002.

[29] A.K. DATTA, M. GRADINARIU, A. KENITZKI, S. TIXEUIL. *Self-stabilizing wormhole routing in ring networks.* in « Proc. of the 9th Int. Conference on Parallel and Distributed Systems (ICPADS 2002) », Taiwan, ROC, décembre, 2002.

[30] P. EZHILCHELVAN, J.-M. HÉLARY, M. RAYNAL. *Building Responsive TMR-Based Servers in Presence of Timing Constraints.* in « Proc. of the 21th ACM SIGACT-SIGOPS Int. Symposium on Principles of Distributed Computing (PODC-02) », pages 127, Monterey, Californie, juillet, 2002, Brief announcement.

[31] C. FETZER, M. RAYNAL. *Approximate Real-Time Clocks for Scheduled Events.* in « Proc. of the 5th Int. IEEE Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2002) », pages 54-61, Washington D.C., avril, 2002.

[32] R. FRIEDMAN, A. MOSTÉFAOUI, S. RAJSBAUM, M. RAYNAL. *Distributed Agreement and its Relation with Error-Correcting Codes.* in « Proc. of the 16th Int. Symposium on Distributed Computing (DISC-02) », série LNCS, numéro 2508, Springer Verlag, pages 63-87, Toulouse, octobre, 2002.

[33] M. GRADINARIU. *Normality versus system mobility.* in « Proc. of the 9th Int. Conference on Parallel and Distributed Systems (ICPADS 2002) », Taiwan, ROC, décembre, 2002.

[34] F. GREVE, J.-P. LE NARZUL. *Implementing FT-CORBA With Portable Interceptors : Lessons learned.* in

« Proc. of the Workshop on Fault-Tolerant Computing, in conjunction with the Brazilian Symposium on Computer Networks », Buzios, Brésil, mai, 2002.

[35] F. GREVE, J.-P. LE NARZUL. *Um Protocolo de Validação Atômica Não-Bloqueante Eficiente.* in « Proc. of the Brazilian Symposium on Computer Networks (SBRC 2002) », Buzios, Brésil, mai, 2002.

[36] J. KELLER, G. SIMON. *Toward a Peer-To-Peer Shared Virtual Reality.* in « Proc. of the Workshop on Resource Sharing in Massively Distributed Systems (RESH'02) in conjunction with the 22nd Int. Conference on Distributed Computing Systems (ICDCS'02) », Vienne, Autriche, juillet, 2002.

[37] D. LAVENIER, H. LEROY, M. HURFIN, R. ANDONOV, L. MOUCHARD, F. GUINAND. *Le projet GénoGRID : une grille expérimentale pour la génomique.* in « Actes de la troisième édition des Journées Ouvertes Biologie Informatique Mathématiques (JOBIM 2002) », pages 27-31, Saint-Malo, France, juin, 2002.

[38] A. MOSTÉFAOUI, S. RAJSBAUM, M. RAYNAL. *A Versatile and Modular Consensus Protocol.* in « Proc. of the IEEE Int. Conf. on Dependable Systems and Networks (DSN 2002) », pages 364-373, Washington D.C., juin, 2002.

[39] A. MOSTÉFAOUI, S. RAJSBAUM, M. RAYNAL. *Asynchronous Interactive Consistency and its Relation with Error-Correcting Codes.* in « Proc. of the 21th ACM SIGACT-SIGOPS Int. Symposium on Principles of Distributed Computing (PODC 2002) », pages 253, Monterey, Californie, juillet, 2002, Brief announcement.

[40] A. MOSTÉFAOUI, S. RAJSBAUM, M. RAYNAL, M. ROY. *Condition-Based Protocols for Set Agreement Problems.* in « Proc. of the 16th Int. Symposium on Distributed Computing (DISC-02) », série LNCS, numéro 2508, Springer Verlag, pages 48-62, Toulouse, France, octobre, 2002.

[41] E. MOURGAYA. *Oracle temporel distribué non fiable : adéquation des algorithmes distribués à un environnement asynchrone.* in « Actes de la cinquième édition des Journées Doctorales Informatique et Réseaux (JDIR 2002) », Toulouse, France, mars, 2002.

[42] M. RAYNAL. *An Introduction to the Renaming Problem.* in « Proc. of the 9th IEEE Pacific Rim Int. Symposium on Dependable Computing (PRDC-02) », Tsukuba, Japon, décembre, 2002.

[43] M. RAYNAL. *Consensus in Synchronous Systems : a Concise Guided Tour.* in « Proc. of the 9th IEEE Pacific Rim Int. Symposium on Dependable Computing (PRDC-02) », Tsukuba, Japon, décembre, 2002.

[44] M. RAYNAL. *Sequential Consistency as Lazy Linearizability.* in « Proc. of the 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA-02) », pages 151-152, Winnipeg, Canada, août, 2002.

[45] M. RAYNAL. *Sequential Consistency as Lazy Linearizability.* in « Proc. of the 1st Eurasia Conference on Advances in Information and Communication Technology (Eurasia ICT 2002) », série LNCS, numéro 2510, Springer Verlag, pages 866-873, Shiraz, Iran, octobre, 2002.

[46] M. RAYNAL. *Wait-Free Objects for Real-Time Systems ?.* in « Proc. of the 5th Int. IEEE Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2002) », pages 413-420, Washington D.C., avril, 2002, Position paper.

[47] C. DELPORTE GALLET, H. FAUCONNIER, J.-M. HÉLARY, M. RAYNAL. *Early Stopping in Global Data Computation.* in « Proc. of the 21th ACM SIGACT-SIGOPS Int. Symposium on Principles of Distributed Computing (PODC 2002) », pages 258, Monterey, Californie, juillet, 2002, Brief announcement.

Rapports de recherche et publications internes

[48] R. BALDONI, J.-M. HÉLARY. *Strengthening Distributed protocols to Handle Tougher Failures.* Research Report, numéro 1477, IRISA, septembre, 2002, http://www.irisa.fr/bibli/publi/pi/2002/1477/1477.html.

[49] A. MOSTÉFAOUI, E. MOURGAYA, M. RAYNAL. *Asynchronous Implementation of Failure Detectors.* Research Report, numéro 1484, IRISA, septembre, 2002, http://www.irisa.fr/bibli/publi/pi/2002/1484/1484.html.

Bibliographie générale

[50] G. BLAIR, J. STEPHANI. *Open Distributed Processing and Multimedia.* 1997.

[51] T. CHANDRA, V. HADZILACOS, S. TOUEG. *The Weakest Failure Detector for Solving Consensus.* in « Journal of the ACM », numéro 4, volume 43, July, 1996, pages 685-722.

[52] T. CHANDRA, S. TOUEG. *Unreliable Failure Detectors for Reliable Distributed Systems.* in « Journal of the ACM », numéro 1, volume 34, March, 1996, pages 225-267.

[53] D. DOLEV, R. REISCHUK, R. STRONG. *Early Stopping in Byzantine Agreement.* in « Journal of the ACM », numéro 4, volume 37, 1990, pages 720-741.

[54] M. FISCHER, N. LYNCH, M. PATERSON. *Impossibility of Distributed Consensus with One Faulty Process.* in « Journal of the ACM », numéro 2, volume 32, April, 1985, pages 374-382.

[55] G. KAN. *Peer-to-peer : Harnessing the Benefits of a Disruptive Technology.* O'Reilly & Associates, Oram edition, 2001, chapitre 2, Listening to Napster.

[56] G. KAN. *Peer-to-peer : Harnessing the Benefits of a Disruptive Technology.* O'Reilly & Associates, Oram edition, 2001, chapitre 8, Gnutella.

[57] A. MOSTÉFAOUI, S. RAJSBAUM, M. RAYNAL. *Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems.* in « Proc. of the 33rd ACM Symposium on Theory of Computing (STOC'01) », Heraklion, Crète, Grèce, juillet, 2001.