Projet ADP

Algorithmes distribués et protocoles

Localisation : Rennes

ADP *est un projet commun à l'*INRIA*, au CNRS et à l'université de Rennes I.*

1 Composition de l'équipe

Responsable scientifique

Michel Raynal [professeur, université de Rennes 1]

Assistante de projet

Maryse Auffray [adjoint administratif, Inria]

Personnel INRIA

Michel Hurfin [CR]

Personnel Université de Rennes I

Jean-Michel Hélary [professeur]
Philippe Ingels [maître de conférences]
Stéphane Lorcy [Ater Ifsic jusqu'au 31/08/99]
Achour Mostéfaoui [maître de conférences]
Noël Plouzeau [maître de conférences]

Collaborateur extérieur

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

Chercheurs invités

Roberto Baldoni [professeur, université La Sapienza de Rome, en novembre 1999]
Amr El Abbadi [professeur, université de Santa Barbara, en juillet 1999]
Paul Ezilchelvan [professeur, université de Newcaslte, en novembre 1999]
Giovanna Mélideo [doctorant, université de Rome, de novembre 1999 à janvier 2000]
Francisco Torres-Rojas [doctorant, Georgia-Tech Atlanta, en mars 1999]
Paulo Veríssimo [professeur, université des Sciences de Lisbonne, en décembre 1998]

Chercheurs doctorants

Udo Fritzke [bourse du gouvernement brésilien]
Fabiola Greve [bourse du gouvernement brésilien]
Géraldine Texier [bourse MENESR jusqu'au 30/11/99]
Frédéric Tronel [allocation couplée]

Chercheurs post-doctorant

Yun Wang [post doctorant Réutel de octobre 1999 à octobre 2000]
Francisco Brasileiro [post doctorant Réutel de octobre 1999 à octobre 2000]

Stagiaires
Cyril Ray [stagiaire de DEA (de février à juin 1999)]

2 Présentation et objectifs généraux

L'évolution de l'informatique a été marquée ces dernières années par le développement sans précédent des réseaux qui permettent l'interconnexion de machines (il suffit de penser à Internet) ainsi que par l'apparition des machines dites "massivement parallèles". Ceci continue de susciter une intense activité de recherche autour des *applications* et des *systèmes répartis* et de leur fondement que constitue l'*algorithmique répartie*.

Si tous ces efforts commencent aujourd'hui à porter leurs fruits (des systèmes répartis existent), la somme des problèmes ouverts n'en demeure pas moins importante. Ainsi, beaucoup de travail, tant du point de vue théorique que pratique, reste à faire dans des domaines tels que : la sécurité, la cohérence des données réparties, les environnements de programmation répartis, la tolérance aux défaillances, etc.

Le projet ADP poursuit, dans ce contexte, deux objectifs complémentaires. Le premier réside dans la compréhension des structures "profondes" des calculs et des systèmes répartis. Il s'agit là d'une recherche à caractère fondamental, explicitement axée (1) sur l'observation et l'analyse des exécutions réparties (détection de propriétés, définition de points de contrôle, etc.) (2) sur la synchronisation d'activités et la gestion cohérente de données réparties et (3) sur la prise en compte de contraintes non fonctionnelles telles que la tolérance aux défaillances et les contraintes temps-réel.

Le deuxième objectif, qui trouve ses fondements dans les résultats précédents et qui en aiguille la problématique, consiste en l'étude de problèmes concrets, à savoir la réalisation d'un noyau réparti de communication fiable, la définition et la réalisation d'un système pour le travail coopératif et la conception de protocoles pour les applications multimedia.

Comme indiqué précédemment, ces deux objectifs (théorie et pratique) sont étroitement imbriqués dans la philosophie du projet ADP et, en conséquence, y sont vus comme les deux faces indissociables d'une même entité, à savoir l'algorithmique du contexte réparti.

3 Fondements scientifiques

La compréhension des fondements des algorithmes répartis passe par la recherche des concepts sous-jacents et de nouveaux paradigmes. Elle doit permettre de concevoir de nouveaux algorithmes répartis, fournissant des services de base tels que le contrôle de la compétition pour les ressources, la détection du passage d'un calcul dans un ensemble d'états particuliers, la collecte d'informations réparties, l'incidence de la causalité, la tolérance aux défaillances, etc. Les travaux actuels s'articulent autour des thèmes ci-dessous.

3.1 Détection de propriétés

Mots clés : algorithme réparti, causalité, détection de propriétés, détection décentralisée.

Résumé : *La détection de propriétés dans les exécutions réparties est un problème fondamental qui se pose pour le concepteur d'environnements de programmation répartie ainsi que pour le contrôle des programmes répartis. Malheureusement, du fait de la structure de l'ensemble des états globaux cohérents d'une exécution répartie, la détection de propriétés est un problème* NP*-complet.*

La détection de propriétés dans les exécutions réparties est un problème fondamental qui se pose pour le concepteur d'environnements de programmation répartie ainsi que pour le contrôle des programmes répartis. Pour le concepteur d'environnements, il s'agit essentiellement de pouvoir fournir un outil de compréhension de ces programmes. Par exemple, lors d'une séance de déboggage, il peut demander si une somme de variables réparties représentant une ressource bornée est supérieure à une constante fixée, s'il existe une conjonction d'états locaux qui forment un état global cohérent tel que chacun de ces états locaux respecte une propriété donnée. Pour le contrôle de programmes répartis, la détection de comportement est un aspect du problème. Ainsi avant de mettre en oeuvre un algorithme de résolution d'interblocage, il faut savoir détecter si un sous-ensemble de processus est en situation d'interblocage.

Dans son cas le plus général, la détection de propriétés est un problème NP-complet. L'observation du comportement d'une exécution par des observateurs répartis pouvant donner lieu à des observations différentes pour un même comportement, il est nécessaire de générer toutes les observations possibles [SM94]. Ceci est dû à la structure de l'ensemble des états globaux cohérents d'une exécution répartie qui au pire, a une taille exponentielle par rapport au nombre de processus mis en jeu. Pour ne pas être limité par ces caractéristiques dans la conception d'algorithmes de détection, il est alors essentiel d'étudier ce problème de manière moins générale, soit en restreignant les types de propriétés que l'on souhaite détecter, soit en restreignant la structure sur laquelle sont détectées les propriétés.

Historiquement, c'est la restriction aux types de propriétés qui a d'abord été employée à travers la distinction *propriétés stables/propriétés instables*. Une propriété est stable si une fois vérifiée dans une exécution répartie, elle le demeure. On peut montrer qu'il existe des algorithmes de coût polynomial pour résoudre le problème de leur détection. Toutefois le problème demeure car la classe des propriétés stables ne rassemble qu'un nombre limité de cas particuliers.

Toujours dans le domaine de la restriction au type de propriétés, une voie plus prometteuse a été récemment étudiée. Il s'agit de la limitation syntaxique dans l'expression même des propriétés. C'est dans cette approche que s'inscrivent les travaux de V. Garg concernant les conjonctions de prédicats locaux [GW94] et les prédicats linéaires ou semi-linéaires [CG95], ainsi que les travaux de Stoller et Schneider sur la décomposition d'expressions en conjonctions.

Au sein de l'équipe ADP, nous nous intéressons plus particulièrement à la restriction de la structure sur laquelle sont détectées les propriétés. Nous avons présenté le problème de la détection de propriétés comme une variante d'un problème de reconnaissance de langage. Le calcul réparti est modélisé sous la forme d'un graphe orienté acyclique. Les sommets du graphe correspondent à des états (locaux ou globaux). Les arcs du graphe sont le reflet des relations de précédence causale qui existent entre les états. Des prédicats de bases (locaux ou globaux) sont exprimés sous forme d'expressions booléennes portant sur les valeurs des variables gérées par les processus de l'application. Ces prédicats de base sont identifiés de manière unique par une lettre (ou symbole). Par définition, un symbole est associé à un sommet du graphe si l'état correspondant satisfait le prédicat de base ainsi identifié. L'ensemble des symboles spécifiés constitue un alphabet. Nous appelons *motif* tout langage défini sur cet alphabet. Le rôle d'un algorithme de détection est de comparer un ensemble de motifs spécifiés avec l'ensemble des motifs observés au cours de l'exécution. Cette comparaison se fait en visitant les sommets du graphe selon la stratégie du tri topologique. Une règle de satisfaction (ou opérateur modal) précise les modalités de comparaison. En choisissant le type de graphe (états globaux, états locaux ou événements) nous pouvons régler le niveau de complexité de l'algorithme. Cette stratégie nous a notamment permis de

[SM94] R. SCHWARZ, F. MATTERN, « Detecting Causal Relationships in Distributed Computations : In Search of the Holy Grail », *Distributed Computing 7*, 3, 1994, p. 149–174.

[GW94] V. K. GARG, B. WALDECKER, « Detection of weak unstable predicates in distributed programs », *IEEE Trans. on Parallel and Distributed Systems 5*, 3, 1994, p. 299–307.

[CG95] C. M. CHASE, V. K. GARG, « Efficient Detection of Restricted Classes of Global Predicates », *in : 9th International Workshop on Distributed Algorithms (WDAG'95)*, J.-M. Hélary, M. Raynal (éditeurs), Springer-Verlag, LNCS 972, p. 303–317, 1995.

reformuler différents algorithmes de détection de propriétés dans un cadre général unique ainsi que d'en proposer de nouveaux [BFR96].

3.2 Cohérence d'objets dupliqués

Mots clés : algorithme réparti, causalité, cohérence, mémoire virtuellement partagée, transaction.

Résumé : *La conception d'applications réparties, dans le modèle de programmation par objets partagés, consiste à donner aux processus la vision d'un seul espace d'adressage. Pour des raisons de performance, les objets sont dupliqués. Le problème est alors le maintien de la cohérence entre les copies pour chaque objet et la proposition d'un mécanisme de tolérance aux défaillances. Ces problèmes sont similaires à ceux rencontrés dans les systèmes implémentant une* MVP *(Mémoire virtuelle partagée).*

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 aux utilisateurs d'accéder, de partager et de manipuler des informations de plus en plus riches et complexes (par exemple, les communautés virtuelles qui réunissent dans des forums un grand nombre d'utilisateurs dispersés sur de grandes étendues géographiques). En plus de la cohérence des données manipulées, tout système qui implémente un monde virtuel doit être à même de stocker un très grand nombre de données et de permettre aux utilisateurs de les accéder avec une qualité de service raisonnable. Concernant la tolérance aux défaillances, il faut prévoir un mécanisme de gestion des processus défaillants ayant effectué des opérations sur une donnée dupliquée et inversement la perte par le système d'une opération effectuée par un processus.

3.3 Points de contrôle et retour arrière

Mots clés : algorithme réparti, causalité, points de contrôle, cohérence, tolérance aux défaillances.

Résumé : *Déterminer des points de contrôle globaux cohérents est une tâche importante qui trouve des applications tant dans le domaine de la tolérance aux défaillances (points de reprise à partir desquels un calcul peut être relancé après défaillance) que dans la détection de propriétés des exécutions réparties (coupes cohérentes). De telles déterminations sont non triviales dans le cadre des systèmes répartis asynchrones. La théorie des Z-chemins de Netzer et Xu énonce qu'un tel chemin entre deux points de contrôle locaux révèle une dépendance qui leur interdit d'appartenir à un même point de contrôle global cohérent.*

Lors de l'exécution répartie d'une application, un point de contrôle global est un ensemble de points de contrôle locaux (états locaux), un par processus participant à l'exécution. La détermination d'un point de contrôle global cohérent est un problème important dans de nombreux domaines concernés par les applications réparties (résistance aux défaillances, mise au point répartie, détection de propriétés, etc.). De nombreux protocoles ont été proposés pour déterminer des points de contrôle locaux afin qu'ils forment des points de contrôle globaux cohérents [BHMR95,EJW96]. Si les points de contrôle locaux sont sélectionnés de manière non coordonnée, il est possible que ceux-ci ne puissent former aucun

[BFR96] O. BABAOĞLU, E. FROMENTIN, M. RAYNAL, « A Unified Framework for the Specification and Run-time Detection of Dynamic Properties in Distributed Computations », *The Journal of Systems and Software 3*, 33, 1996, p. 287–298, Special issue on Software Engineering for Distributed Computing.

[BHMR95] J. BRZEZINSKI, J. HÉLARY, A. MOSTEFAOUI, M. RAYNAL, « Semantics of recovery lines for backward recovery in distributed systems », *Annales des Télécommunications 50*, 11-12, 1995, p. 874–887.

[EJW96] E. ELNOZAHY, D. JOHNSON, Y. WANG, « A Survey of Rollback-Recovery Protocols in Message-Passing Systems », *Technical Report n° CMU-CS-96-181*, Carnegie-Mellon University, 1996.

point de contrôle global cohérent (ce risque est connu sous le nom d'*effet domino* [Ran75]). Une forme de coordination est donc nécessaire si l'on veut éviter - ou diminuer - cet effet. Les techniques de calcul d'état global cohérent introduites par Chandy-Lamport [CL85] sont basées sur une coordination explicite utilisant des messages de contrôle supplémentaires. En général, cette forme de coordination impose à tous les processus de prendre des points de contrôle locaux lorsque l'un d'entre eux décide d'en prendre un, même en l'absence de communication due à l'exécution répartie. Une autre forme de coordination, implicite, utilise des informations de contrôle véhiculées par les messages de l'application (technique de *piggybacking*). Pour cette raison, elle est connue sous le nom de *coordination induite par les communications* (communication-induced checkpointing). Dans cette approche, des points de contrôle locaux sont sélectionnés de manière non coordonnée (points de contrôle *de base*) et le protocole impose des points de contrôle locaux supplémentaires, appelés *points de contrôle forcés*, de manière à assurer la progression des points de contrôle globaux cohérents. Les points de contrôle forcés sont pris sur la base des informations de contrôle véhiculées par les messages de l'application. C'est cette approche qui a été principalement étudiée dans notre équipe depuis l'année 1995, bien que nous ayons abordé récemment l'approche non coordonnée.

Études dans le cadre de la notion classique cohérence Le fait que deux points de contrôle locaux ne soient pas liés causalement constitue une condition nécessaire pour appartenir à un même point de contrôle global cohérent. Malheureusement, cette condition n'est pas suffisante. Les points de contrôle locaux peuvent avoir des dépendances cachées (c'est-à-dire non captables par un mécanisme d'estampillage) qui les empêchent de participer au même point de contrôle global cohérent. Afin de capter l'ensemble des dépendances liant les points de contrôle locaux, Netzer et Xu [NX95] ont introduit la notion de *Z-chemin* entre points de contrôle locaux. Ils ont de plus démontré le théorème fondamental suivant : *un ensemble quelconque de points de contrôle locaux peut être étendu pour former un point de contrôle global cohérent si, et seulement si, il n'y a pas de Z-chemin connectant deux points de contrôle de cet ensemble*. Si, dans une exécution répartie, on ne considère que les points de contrôle locaux et leurs relations de dépendance, on obtient une abstraction de cette exécution répartie (cette abstraction ignore tous les états locaux qui ne sont pas des points de contrôle locaux). Une question importante est alors : "cette abstraction est-elle cohérente ?" Cette question peut être abordée dans deux contextes différents, selon la notion de cohérence particulière considérée pour l'abstraction définie par les points de contrôle.

Dans le premier cas, l'abstraction est cohérente si tout point de contrôle local appartient à au moins un point de contrôle global cohérent. En d'autres termes, il n'y a pas de point de contrôle local qui s'avère inutile du point de vue de la construction des points de contrôle globaux cohérents (du point de vue opérationnel ceci élimine l'effet domino lors de la construction des points de contrôle globaux). Netzer et Xu ont montré, au niveau de l'abstraction, qu'il n'y a pas de point de contrôle inutile si, et seulement si, aucun Z-chemin n'est un cycle (Z-cycle). Il s'agit alors de traduire cette propriété au niveau opérationnel et d'en déduire des protocoles efficaces, notamment dans le cadre de l'approche *coordination induite par les communications* (conception et analyse de protocoles de détermination de points de contrôle forcés prévenant l'occurrence de points de contrôle inutiles).

Une seconde notion de cohérence pour l'abstraction que constitue un ensemble de points de contrôle locaux réside dans l'absence de relations de dépendance cachées entre ces points de contrôle locaux. Cette notion de cohérence, plus forte que la précédente, a été proposée par Wang [Wan97]. Appelée propriété RDT (pour *Rollback-Dependency Trackability*), elle se révèle particulièrement utile pour résoudre des problèmes tels que le calcul au vol du point de contrôle global *minimal* cohérent auquel appartient un

[Ran75] B. RANDELL, « System Structure for Software Fault-Tolerance », *IEEE Transactions on Software Engineering SE1*, 2, 1975, p. 220–232.

[CL85] K. CHANDY, L. LAMPORT, « Distributed Snapshots: Determining Global States of Distributed Systems », *ACM TOCS*, February 1985, p. 63–75.

[NX95] R. NETZER, J. XU, « Necessary and Sufficient Conditions for Consistent Global Snapshots », *IEEE Transactions on Parallel and Distributed Systems 6*, 2, 1995, p. 165–169.

[Wan97] Y. WANG, « Consistent Global Checkpoints That Contain a Given Set of Local Checkpoints », *IEEE Transactions on Computers 46*, 4, 1997, p. 456–468.

point de contrôle local donné (dans [Wan97] d'autres exemples de problèmes dont la solution est facilitée lorsque les points de contrôle définis lors d'une exécution satisfont la propriété RDT. A titre d'exemples, les points de contrôle globaux cohérents minimaux facilitent la recherche d'erreurs logicielles et la relance des exécutions après la détection d'un interblocage). Dans le cadre de l'approche *coordination induite par les communications*, nous nous sommes intéressés à la recherche de conditions optimales sous lesquelles un processus sera obligé de prendre un point de contrôle forcé. Par là nous entendons la recherche de conditions telles que le nombre de points de contrôle forcés serait le plus faible possible. Dans une approche "théorique", il s'agit de définir un ensemble de Z-chemins dont la suppression (par prise de points de contrôles locaux) suffit à assurer la propriété RDT. Trivialement, cette propriété est assurée si *tous* les Z-chemins sont ainsi supprimés. Mais cette solution peut s'avérer coûteuse en termes du nombre de points de contrôle forcés qu'elle induit. C'est pourquoi la recherche d'ensembles *minimaux* de Z-chemins est un problème de toute première importance. Sur le plan "pratique", il s'agit de concevoir des protocoles "génériques" qui offrent à l'utilisateur des compromis entre taille des informations de contrôle véhiculées par les messages et nombre de points de contrôle forcés.

Extensions de la notion de cohérence Un examen approfondi des notions de cohérence d'états globaux (ou de points de contrôle globaux) fait apparaître que la notion usuelle, basée sur la causalité, ne représente qu'un aspect du problème. C'est pourquoi d'autres modèles de cohérence doivent être envisagés. Deux modèles, en particulier, présentent un grand intérêt pratique. Ce sont, respectivement, *l'absence de messages en transit (transitlessness)*, duale de la notion classique, et la *cohérence forte*, réunion de la cohérence classique et de l'absence de transit. La cohérence classique considère des points de contrôle globaux dans lesquels tout message est enregistré comme *reçu* seulement s'il y est enregistré comme *émis*. La cohérence *transitless* considère des points de contrôle globaux dans lesquels tout message doit être enregistré comme *reçu* si il y est enregistré comme *émis*. La cohérence forte considère des points de contrôle globaux dans lesquels tout message est enregistré comme *reçu* si et seulement si il y est enregistré comme *émis*. Le problème consiste alors à définir des formalismes permettant d'étendre à ces modèles de cohérence la condition nécessaire et suffisante pour qu'un ensemble de points de contrôle locaux puisse faire partie d'un point de contrôle global cohérent. Cette étude théorique sert de fondement à la conception de protocoles assurant la cohérence forte en combinant les deux techniques de points de contrôle forcés et de sauvegarde des messages (recording), ainsi qu'à l'étude des compromis entre ces deux techniques.

3.4 Contraintes non-fonctionnelles

Mots clés : algorithme réparti, résistance aux défaillances, temps réel, communication de groupe, consensus.

Résumé : *Dans un système réparti asynchrone, il est important de pouvoir concevoir des applications tolérant les défaillances tout en garantissant le respect de contraintes temps-réel. Le concept de groupe s'avère dans ce cas particulièrement intéressant. 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.*

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 du projet ADP 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. 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 en ce qui concerne l'identité des membres qui le composent (membership) et l'ordre dans lequel les messages qui leurs sont adressés seront délivrés (diffusion ordonnée).

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 [FLP85] : 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 [CT96] 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. La mise en œuvre du mécanisme de détection se fait en définissant des délais de garde: un processus est suspecté s'il ne s'est pas manifesté avant l'expiration du délai de garde. Cela signifie que (1) la détection d'une défaillance réelle est généralement différée et que (2) un détecteur de défaillances peut commettre des erreurs en suspectant à tort un processus d'avoir stoppé son exécution. 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 [CHT96]. 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 de précision 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 subissent 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 brique 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.

Comme indiqué précédemment, nous souhaitons intégrer la notion de contrainte temps-réel dans les solutions que nous proposons aux problèmes d'accord ainsi que dans les services de duplication mis en oeuvre pour garantir la sûreté de fonctionnement. Prendre en compte des contraintes temps-réel nécessite "à un niveau ou à un autre" de considérer le temps physique. C'est pour cela que nous comptons remplacer le modèle "totalement" asynchrone par le modèle "asynchrone temporisé". Ce modèle réaliste est défini par la caractéristique suivante. Tout processus peut définir des délais maximaux sur les temps de transfert et de traitement. Mais comme le support est asynchrone, ces délais peuvent être violés : dans ce cas, l'application en est avertie (c'est la notion de *fail-awareness* décrite dans [FC96]) et réagit par un traitement d'exception approprié.

[FLP85] M. FISCHER, N. LYNCH, M. PATERSON, « Impossibility of Distributed Consensus with One Faulty Process », *Journal of the ACM 32*, 2, April 1985, p. 374–382.

[CT96] T. CHANDRA, S. TOUEG, « Unreliable Failure Detectors for Reliable Distributed Systems », *Journal of the ACM 34*, 1, March 1996, p. 225–267.

[CHT96] T. CHANDRA, V. HADZILACOS, S. TOUEG, « The Weakest Failure Detector for Solving Consensus », *Journal of the ACM 43*, 4, July 1996, p. 685–722.

[FC96] C. FETZER, F. CRISTIAN, « Fail-Awareness in Timed Asynchronous Systems », *in : Proc. of the fifteenth ACM Symposium on Principles of Distributed Computing (PODC'96)*, Philadelphie, May 1996.

La définition de “bons” délais, dépend du support et de sa charge en “régime de croisière”; les manquements à ces délais n’apparaissent qu’en périodes d’instabilité dues à des surcharges occasionnelles ou à l’occurrence de situations imprévisibles.

assisté par ordinateur

3.5 Environnements pour le travail coopératif assisté par ordinateur

Mots clés : algorithme réparti, application répartie, architectures à objets réparties, travail coopératif, TCAO, CSCW, multimedia, World Wide Web, VRML, Java.

L’expansion actuelle des technologies de l’information sur le lieu de travail ou à domicile n’aurait pas été possible sans la croissance exponentielle de l’ensemble des usagers connectés au réseau, à titre individuel ou professionnel. Le système élémentaire de partage d’information (le Web) a ouvert la voie à l’étape suivante : la généralisation de l’emploi d’applications de travail coopératif assisté par ordinateur. Cette catégorie d’application offre un ensemble de services permettant la collaboration d’un groupe d’individus géographiquement dispersés, en leur évitant de se déplacer pour se réunir. La collaboration à distance permet d’améliorer la productivité, de réduire les frais et de mieux allouer les ressources (sans compter les gains annexes, par exemple en terme d’écologie). Les domaines de travail coopératif les plus courants sont la télé-formation, la télé-expertise (par exemple, la télé-médecine), les systèmes d’aide à la prise de décision et la conception assistée par ordinateur.

La construction d’environnements de travail coopératif efficaces nécessite une maîtrise scientifique et technologique dans de nombreux domaines : systèmes répartis, réseau, interface homme-machine, communication multimedia (audio, vidéo), ergonomie, organisation du travail, entre autres. Un certain nombre de projets de l’Irisa détiennent un savoir-faire scientifique et technologique avancé dans les domaines précités. Pour cette raison, l’Irisa a créé l’action Rusken en janvier 1996 afin de créer un pôle d’expertise en matière d’environnements de travail coopératif. Cette action s’est terminée au printemps 1999. Elle avait pour objectifs :

1. la conception et la réalisation d’une plate-forme opérationnelle et innovante fournissant des services de travail coopératif assisté par ordinateur,
2. l’acquisition par l’Irisa d’une maîtrise scientifique et technologique dans le domaine du travail coopératif,
3. la définition et la mise en place de travaux transversaux aux projets de recherche de l’Irisa, dès l’étape de conception de l’environnement.

L’action Rusken était hébergée par le projet ADP de l’Inria ; en 1999 les travaux ont été poursuivis dans le cadre de ce projet. La stratégie suivie par l’action Rusken a porté sur les points fondamentaux suivants :

- suivi de l’évolution des nouvelles technologies de l’information,
- identification des verrous scientifiques et technologiques,
- recherche de l’innovation en terme de services offerts ou en terme de qualités intrinsèques de la plate-forme (par ex. performances, résistance aux défaillances, modularité, etc.)
- application de résultats de recherche issus de projets de l’Irisa, pour leur évaluation et leur valorisation dans le contexte des environnements de travail réparti.

Le type de plate-forme visée était un environnement de collaboration de type travail de bureau, comportant d’une part un système d’information partagé (bases de documents, système de diffusion d’information), d’autre part des moyens de communication de groupe (courrier électronique, communication audio et vidéo depuis le poste de travail, suivi de l’activité de groupe).

4 Domaines d'applications

4.1 Panorama

Mots clés : application répartie, résistance aux défaillances, temps réel, communication de groupe, consensus, Corba, hétérogénéité.

Résumé : *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 une norme telle que Corba en spécifiant un ensemble de services destiné à assurer simultanément le respect de contraintes de sûreté de fonctionnement et de contraintes temps-réel correspond donc à une attente de la part de nombreux industriels. C'est dans cette voie que nous développons actuellement des coopérations avec des partenaires industriels.

Les travaux de recherche et développement effectués initialement dans le cadre de l'action Rusken ont pour domaines d'applications les activités de télé-coopération sur des informations. On peut nommer par exemple le télétravail de bureau, la télé-expertise (par exemple télé-bibliothèque avec assistance de bibliothécaires, télé-diagnostic médical, télé-assistance informatique) ou la télé-réunion.

5 Logiciels

5.1 Panorama

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

Résumé :

Les travaux de recherche menés sur le thème de la tolérance aux défaillances et sur le thème du travail coopératif, nous ont conduit à développer des prototypes afin de valider nos propositions.

Le plate-forme Eva qui offre des services de communication de groupe a été conçue pour être intégrée par la suite dans un ORB *et offrir un service de tolérance aux défaillances conforme à la future norme Corba, actuellement en cours de définition.*

Dans le cadre de l'action Rusken, une plate-forme de travail assisté par ordinateur a été conçue au cours des quatre dernières années.

5.2 Plate-forme Eva

Participants : Fabiola GREVE, Michel HURFIN[correspondant], Jean-Pierre LE NARZUL, Frédéric TRONEL.

Résumé : *Depuis plus d'un an, un effort important est fourni pour développer dans un environnement Corba un prototype convaincant qui intègre les solutions que nous proposons pour garantir la sûreté de fonctionnement et le respect de contraintes temporelles. Notre objectif dans les années à venir est d'obtenir un démonstrateur convaincant répondant aux*

*exigences mentionnées dans l'appel à propositions rédigé par l'*OMG *sur le thème de la tolérance aux défaillances dans Corba.*

La Plate-forme Eva offre un ensemble de services de gestion de groupe permettant entre autre de maintenir une cohérence forte au sein d'un ensemble d'objets dupliqués. En conséquence, cette plate-forme est une base idéale pour développer à l'avenir un service de tolérance aux défaillances basé sur le principe de la duplication. Un appel à contribution a été lancé par l'OMG afin de normaliser un service de tolérance aux défaillances fondé sur le principe de la duplication (active ou passive). Nous suivons actuellement le processus de normalisation qui est en cours et débutons de nouveaux développements afin d'être en accord avec le futur standard qu'adoptera l'OMG d'ici une année ou deux.

La plate-forme Eva 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). L'originalité de la plate-forme 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. Nous allons par la suite détailler ces deux points :

- Un service d'accord entre membres du groupe constitue le coeur de la plate-forme Eva. Il s'agit d'un service de consensus étendu. L'adoption d'une approche probabiliste nous permet de ne pas utiliser de détecteurs de défaillances à ce niveau. La connaissance de la fonction de répartition des messages nous permet de concevoir un protocole d'accord dont la durée maximale d'exécution sera fixée par les services de plus haut niveau. Le protocole de consensus qui a été proposé par Chandra et Toueg et qui nécessite à l'origine des détecteurs de défaillances non fiables de la classe $\diamond\mathcal{S}$, constitue l'ossature de notre brique de base. Ce protocole a cependant été considérablement remanié et enrichi par de nouvelles extensions. La durée des tours et le nombre maximal de tours qui seront effectués sont calculés à l'initialisation. La durée d'un tour étant bornée, le détecteur de défaillance devient inutile: la durée d'un tour correspond au temps qui est imparti à un coordinateur pour imposer sa valeur. Les extensions offertes par ce nouveau service d'accord permettent 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). Le protocole s'exécute en permanence. Lorsque aucune valeur significative n'est proposée, l'exécution du protocole engendre un motif de communication restreint dont la régularité peut être mise à profit pour assurer la synchronisation des horloges et la mise en oeuvre de détecteurs de défaillances. Les services de haut niveau, clients de ce service de base, fournissent des valeurs (liste de message à ordonner, liste de processus à éliminer du groupe, liste de processus à insérer dans le groupe, etc.). Les décisions générées par ce service d'accord sont ensuite réexploitées par les services de haut niveau.
- La plate-forme Eva a été développée en utilisant un langage orienté objet (Java). Pour faciliter ses extensions futures ainsi que son intégration dans un environnement Corba, l'ensemble des interactions entre les différents composants de la plate-forme se fait grâce à un mécanisme de communication par événements. Chaque composant de la plate-forme est défini comme étant un producteur et/ou un consommateur d'évènements. Chaque événement possède un type, une priorité et une méthode pour le consommer qui diffèrent d'un consommateur à un autre. Sur chaque site, un médiateur et des coupleurs destinés à transformer un type d'événement en un autre, permettent aux producteurs et aux consommateurs de communiquer de façon asynchrone sans être au courant de leurs existences réciproques. Ce mécanisme de communication permet d'établir dynamiquement des chemins de communication. De fait, ce mécanisme nous permet de développer des solutions modulaires et flexibles adaptées aux contraintes d'une application.

La plate-forme Eva est toujours en phase de développement. Dans sa version courante, elle permet de gérer dynamiquement un groupe de processus: seuls les ajouts de processus au groupe initial ne sont pas encore supportés. Parmi les développements qui seront réalisés dans un futur proche figurent la mise en oeuvre d'un service de transfert d'état, la conception et le développement d'un gestionnaire centralisé de copies, ainsi bien évidement que les travaux d'intégration d'Eva dans un ORB.

5.3 Plate-forme rusken

Participants : Alain CÉDELLE, Stéphane LORCY, Géraldine TEXIER, Noël PLOUZEAU[correspondant].

Résumé : *La plate-forme de travail assisté par ordinateur conçue dans le cadre de l'action Rusken offre d'une part des services de gestion et de communication de documents et d'autre part des services de télécommunication de groupe. La plate-forme est construite à partir de logiciels du commerce assistés d'un nombre important d'adaptations et d'extensions spécifiques conçues par les membres de Rusken.*

La conception et la réalisation d'une plate-forme de travail coopératif assisté par ordinateur sont au centre de l'activité de recherche et développement qui a été menée par l'action Rusken. Du point de vue des services, cette plate-forme de travail coopératif offre un environnement pour le travail de bureau. Il s'agit d'une part de services de gestion et de communication de documents (stockage, édition, diffusion, partage) et d'autre part de services de télécommunication de groupe (téléréunion de plusieurs personnes, vidéocommunication depuis le poste de travail, outils graphiques d'aide à la perception de l'activité du groupe). Du point de vue de l'architecture, la plate-forme est construite à partir de logiciels du commerce (Lotus Notes, lecteurs Web, logiciels de vidéocommunication, systèmes graphiques d'affichage de scènes tridimensionnelles) assistés d'un nombre important d'adaptations et d'extensions spécifiques conçues par les membres de Rusken.

La plate-forme est conçue autour d'un noyau de gestion de la collaboration, développé spécifiquement par l'action Rusken. Ce système assure le suivi des activités des utilisateurs (espaces de travail, règles d'interaction, droits, etc). Autour du noyau gravitent plusieurs systèmes spécialisés :

- le système Notes de la société Lotus est chargé du stockage des documents et de la communication asynchrone entre utilisateurs (courrier électronique),
- un système de suivi de la qualité de service, développé par Rusken, est chargé de l'évaluation des performances d'exécution, notamment en matière de communication entre les différents sites participant à une session de travail,
- un système de contrôle du parallélisme des actions, développé spécialement par Rusken,
- une application de navigation dans un espace d'information présenté selon la métaphore du bâtiment virtuel, développée en collaboration avec le projet Siames de l'Irisa,
- un système de gestion de la coopération entre utilisateurs, développé par Rusken,
- une application d'intégration des différents outils mis à disposition des utilisateurs de la plate-forme.

Les différents utilisateurs de la plate-forme étant géographiquement dispersés, l'architecture a été conçue pour être totalement répartie. Ce haut degré de répartition permet de mieux résister à l'hétérogénéité potentielle des ressources matérielles et logicielles des différents postes de travail des utilisateurs.

La diversité de la palette des services à apporter, jointe au caractère réparti de la mise en œuvre nous ont amené à développer une série d'extensions spécifiques. Celles-ci répondent à différents types de problèmes. La première catégorie de difficulté concerne l'intégration de logiciels hétérogènes pour produire un système d'aspect externe homogène. Le mécanisme fondamental utilisé est l'emploi des interfaces de programmation (Api) et la réalisation de modules adaptateurs entre ces interfaces d'applications. Les difficultés à résoudre portent sur l'encodage des types de données et sur la synchronisation des actions des applications. Deux types d'architectures de contrôle d'application ont été étudiées par l'action Rusken : le modèle Ole/Com de Microsoft et des modèles fondées sur le langage Java. Ces derniers modèles ont été préférés pour les réalisations spécifiques de Rusken, en raison de la relative simplicité des mécanismes fondés sur Java par rapport à ceux du modèle Ole/Com, et en raison des meilleures possibilités d'extension. La seconde catégorie de difficulté traitée par les extensions conçues dans le cadre de Rusken concerne le développement de services nouveaux ou l'amélioration des propriétés de services existants. Le travail effectué en 1999 concerne la gestion de la coopération entre utilisateurs et l'intégration des modules développés en 1998.

Module de gestion de la qualité de service des communications

Pour prendre en compte la gestion de la qualité de service du réseau de communication, les membres de l'action Rusken ont conçu et développé un système de suivi de la qualité de communication entre objets répartis. L'architecture de ce système s'appuie sur les résultats d'un travail de recherche et développement. Le module a été réalisé entièrement en Java et utilise les mécanismes de communication RMI de Java ou le service de diffusion de la famille de protocoles IP (*multicast*).

Application de gestion de la coopération

L'interaction des utilisateurs avec leur espace de travail partagé est contrôlé par une application qui s'appuie sur le modèle de coopération développé par Rusken. Un composant graphique fondé sur la technologie ActiveX et intégrable dans des applications de travail coopératif permet de naviguer dans l'ensemble des domaines d'utilisateurs, d'activités déclarées et de percevoir les sessions automatiquement établies par le système de gestion de coopération [33].

6 Résultats nouveaux

6.1 Détection de propriétés

Participants : Michel HURFIN, Michel RAYNAL.

Mots clés : algorithme réparti, causalité, détection de propriétés, détection décentralisée, état global partagé, conjonction de prédicats locaux.

Résumé : *Les horloges logiques (scalaires, vectorielles ou matricielles) sont un mécanisme puissant utilisé par de nombreux algorithmes de contrôle réparti. Nous nous sommes intéressés à étudier leur puissance et leur limitation pour détecter des propriétés d'exécutions réparties.*

L'horlogerie logique vectorielle permet de capter la relation de causalité entre les évènements produits par une exécution répartie. Plus précisément, une horloge vectorielle est un tableau d'entiers dont l'entrée i mesure la progression du processus p_i. Nous avons examiné deux problèmes rencontrés dans la mise en oeuvre de programmes répartis, et étudié l'apport des horloges vectorielles pour les résoudre. Le premier problème est la détection d'une conjonction de prédicats locaux stables. Il s'avère que les vecteurs d'horloges sont l'outil idoine pour résoudre ce problème. Le second problème est la détection d'un motif formé par des évènements d'une exécution répartie. Nous avons montré que les vecteurs d'horloge sont insuffisants pour résoudre ce problème. Un système d'horlogerie logique plus sophistiqué s'avère en fait nécessaire. Nous avons montré que la solution à ce problème requiert des vecteurs de vecteurs d'horloge. Un protocole résolvant le problème a aussi été proposé [30, 31].

6.2 Cohérence de mémoires réparties d'objets

Participants : Udo FRITZKE, Philippe INGELS, Achour MOSTÉFAOUI, Cyril RAY, Michel RAYNAL.

Mots clés : algorithme réparti, causalité, cohérence, mémoire virtuellement partagée, transaction, diffusion atomique.

Résumé : *D'une part, nous avons proposé une primitive de diffusion atomique dans plusieurs groupes où des processus peuvent être défaillants. Une telle primitive permet de mettre en œuvre la cohérence séquentielle en environnement non fiable. D'autre part, nous avons explorer les différentes possibilités de mise en œuvre d'un système (*MVP + *processus) tolérant les défaillances.*

Dans le modèle mémoire virtuelle partagée, nous avons étudié la cohérence séquentielle et la cohérence causale ainsi que leurs équivalents dans le modèle transactionnel, à savoir, la sérialisabilité et la cohérence causale. Nous avons proposé un protocole mettant en œuvre une primitive de diffusion atomique dans plusieurs groupes qui serait moins coûteux et résistant au facteur d'échelle. Sa conception repose sur deux briques de base, en l'occurrence, la *diffusion fiable uniforme* et le *consensus uniforme*. La construction de ce protocole assure deux propriétés principales. La *minimalité* : seuls les processus des groupes destinataires d'un message et le processus émetteur participent au protocole de diffusion (réduction le coût). La seconde est la propriété de *localité* : un consensus ne peut concerner que les membres d'un même groupe (résistance au facteur d'échelle). Ces travaux ont été implémentés au dessus de TCP.

D'autre part, nous avons proposé un modèle de défaillance pour les systèmes à base de MVP. Ce modèle définit 5 situations possibles en fonction du type des processus (déterministes ou non), de la disponibilité de la MVP (peut-elle perdre des opérations) et du besoin ou non de reprise des processus après une défaillance. De là nous avons proposé une taxonomie des différents algorithmes existants dans la littérature. Parmi les cinq cas du modèle, un n'a jamais été étudié (processus déterministes par morceaux). Ce cas est très intéressant car rendant possible le découplage entre le mécanisme de reprise de la MVP et celui des processus, permettant ainsi une conception plus modulaire et beaucoup plus simplifiée. Nous avons enfin proposé une mise en œuvre d'un tel système.

6.3 Points de contrôle et retour arrière

Participants : Roberto BALDONI, Mimmo CIOFFI, Jean-Michel HÉLARY, Achour MOSTÉFAOUI, Michel RAYNAL.

Mots clés : causalité, point de contrôle, cohérence, tolérance aux défaillances.

Résumé : *Nous avons essentiellement raffiné les résultats essentiels obtenus en 1996 et 1997, notamment dans l'étude de la cohérence forte et dans la recherche d'un ensemble minimal de Z-chemins dont la suppression suffit à assurer la propriété* RDT.

Nous nous sommes intéressés au problème de la détermination de points de contrôle globaux cohérents selon les deux approches coordination induite par les communications *(communication-induced checkpointing) et* non coordonnée.

Etudes dans le cadre de la notion classique de cohérence Nous nous sommes intéressés à la question essentielle : "l'abstraction d'un calcul obtenue en ne considérant que les points de contrôle locaux et leurs relations de dépendance, est-elle cohérente ?" Nous avons examiné deux réponses à cette question, chacune considérant une notion de cohérence particulière pour l'abstraction définie par les points de contrôle.

Dans le premier cas, l'abstraction est cohérente si tout point de contrôle local appartient à au moins un point de contrôle global cohérent. Nous avons caractérisé cette propriété de cohérence par une propriété sur l'estampillage des points de contrôle par des entiers, montré comment cet estampillage permet d'obtenir rapidement "en ligne" des points de contrôle globaux cohérents, et proposé un protocole générique, coordonné par les communications. Ce protocole recouvre de nombreux protocoles connus [6].

Nous avons proposé un protocole non coordonné de prises de points de contrôle locaux. Dans une stratégie non-coordonnée, les processus prennent des points de contrôle locaux de manière indépendante. Le problème est alors le suivant: étant donné un sous-ensemble de points de contrôle locaux (au plus un par processus), la théorie de Netzer et Xu permet de répondre à la question *existe-t-il un point de contrôle global cohérent qui contient ce sous-ensemble?* par une caractérisation théorique. Le protocole proposé est capable de répondre *en ligne* à cette question; de plus, si la réponse est positive, il fournit le point de contrôle global cohérent *le plus récent* englobant le sous-ensemble donné, sinon il fournit le "prochain" point de contrôle global cohérent [18].

La seconde notion de cohérence, plus forte que la précédente, est la propriété RDT (pour *Rollback-Dependency Trackability*) introduite par Wang. Dans le cadre de l'approche *coordination induite par les communications*, nous nous sommes intéressés à la recherche de conditions optimales sous lesquelles un processus sera obligé de prendre un point de contrôle forcé. Par là nous entendons la recherche de conditions telles que le nombre de points de contrôle forcé serait le plus faible possible. Dans une approche "théorique", il s'agit de définir un ensemble de Z-chemins dont la suppression (par prise de points de contrôles locaux) suffit à assurer la propriété RDT. Nous avons donc caractérisé un ensemble *minimal* de Z-chemins dont la suppression assure la propriété RDT [19] et proposé un protocole induit par les communications qui réalise "au mieux" cette caractérisation optimale [2].

Extensions de la notion de cohérence Les deux autres modèles de cohérence que nous avons développé, respectivement *l'absence de messages en transit (transitlessness)*, duale de la notion classique, et la *cohérence forte*, réunion de la cohérence classique et de l'absence de transit nous ont conduit à proposer un formalisme abstrait, englobant les trois modèles précédents. Dans ce formalisme, nous avons exprimé la condition nécessaire et suffisante pour qu'un ensemble de points de contrôle locaux puise faire partie d'un point de contrôle global cohérent ([8]). Cette étude théorique nous a servi de fondement pour concevoir de nouveaux protocoles assurant la cohérence forte, en combinant les deux techniques de points de contrôle forcés et de sauvegarde des messages (recording), ainsi qu'à l'étude des compromis entre ces deux techniques ([7]).

6.4 Contraintes non-fonctionnelles

Participants : Udo FRITZKE, Fabiola GREVE, Jean-Michel HÉLARY, Michel HURFIN, Philipe INGELS, Jean-Pierre LE NARZUL, Achour MOSTÉFAOUI, Michel RAYNAL, Frédéric TRONEL.

Mots clés : algorithme réparti, résistance aux défaillances, duplication, communication de groupe, temps réel, consensus.

Résumé : *Les travaux réalisés ont essentiellement porté sur le problème du consensus et son utilisation comme brique de base pour résoudre des problèmes d'accord. Les résultats obtenus ont été en partie intégrés à la plate-forme Eva que nous élaborons en parallèle. Qu'ils soient de nature théorique ou pratique, ces résultats permettent avant tout de mieux comprendre les problèmes posés. A la lumière de ces travaux qui explorent différentes voies, nous pouvons effectuer et justifier nos choix d'implémentation.*

Étant donné que nous utilisons le consensus comme une brique de base pour développer l'ensemble des services de groupe, la majeure partie de notre activité de recherche est focalisée sur ce service élémentaire. Plusieurs actions de recherche complémentaires ont été entreprises :

- **Solutions au problème du consensus**
  Il est désormais reconnu que les services de communication de groupe (diffusion ordonnée, gestion de la composition du groupe) peuvent être perçus comme des instances d'un problème d'accord unique: le consensus. Ce problème d'accord élémentaire semble trivial en apparence. Or, il est prouvé que ce problème n'admet pas de solution déterministe. Notre axe de recherche principal est donc naturellement consacré à l'étude des solutions déterministes permettant de contourner ce résultat d'impossibilité[5].
  Nous nous sommes plus particulièrement intéressés aux solutions déterministes s'appuyant sur des détecteurs de défaillances appartenant à la classe $\diamond\mathcal{S}$. Tous les protocoles proposés requièrent qu'une majorité de processus soit constamment en activité. Tous s'appuient sur le paradigme du coordinateur tournant. Un processus exécute une série de tentatives dans le but d'aboutir à une décision commune. Chaque tentative est dirigée par un processus prédéterminé appelé coordinateur.

Le nombre de tentatives effectuées est arbitraire ; il dépend à la fois des occurrences de pannes et du comportement des détecteurs de défaillances.

Les travaux menés ont abouti à la définition de deux nouveaux protocoles performants utilisant des détecteurs de défaillances appartenant à la classe $\diamond\mathcal{S}$ [9, 28]. Ces deux protocoles utilisent un schéma de communication décentralisé. Les détecteurs de défaillances sont généralement fiables durant de longues périodes : les délais de gardes employés sont en effet calculés de façon à ce que les défaillances annoncées soient le plus souvent réelles. Dans ces conditions, ces deux algorithmes s'avèrent être plus performant (en terme de latence) que les algorithmes précédemment proposés.

Dans le premier protocole [9], chaque processus vote (en diffusant un message) afin de favoriser soit une décision durant la tentative courante, soit un abandon de cette tentative afin de débuter une nouvelle tentative. L'action recevant une majorité de suffrages est finalement exécutée. Les votes (au plus deux) émis par un processus durant une tentative sont régis par un automate simple qui permet de gérer les changement d'avis éventuel et d'éviter ainsi tout interblocage (tout en garantissant l'unicité des valeurs décidées par les processus durant des tentatives éventuellement distinctes). Cet algorithme s'avère être peu sensible aux suspicions erronées générées par des détecteurs de défaillances non fiables tant que leur nombre reste dans des limites raisonnables.

Outre sa simplicité, le second protocole [28] offre une dimension générique nouvelle puisqu'il s'adapte aisément a deux classes de détecteurs de défaillances différentes à savoir la classe $\diamond\mathcal{S}$ et la classe $\mathcal{S}$. Contrairement à l'algorithme précédent, le choix effectué au cours d'une tentative (acquittement positif ou négatif) est définitif.

Nous avons également défini et montré l'intérêt de nouvelles classes de détecteur caractérisées par une propriété d'exactitude de portée plus limitée [29]. La portée de la propriété d'exactitude d'un détecteur de défaillances est le nombre maximum de processus dont on peut exiger qu'ils ne suspectent pas un processus correct. La portée des détecteurs de défaillances proposés par Chandra et Toueg est de n (tous les processus du système). Nous avons proposé un protocole de réduction permettent de construire un détecteur de défaillances appartenant à la classe $\diamond\mathcal{S}$ (ou à la classe $\mathcal{S}$) à partir d'un détecteur de défaillance de la même classe mais dont la portée y est plus faible ($y < n$). Un tel protocole existe à condition que le nombre maximum f de défaillances dans le système soit inférieur à y. Un protocole de consensus adapté à de tels détecteurs de défaillances a également été proposé [29]. Dans [39], nous avons introduit le problème du "k-Set Agreement" qui généralise le problème du consensus puisque l'unicité de la valeur décidée n'est plus obligatoire: k valeurs de décisions distinctes sont tolérées (et non pas une seule comme dans le problème original). Un protocole utilisant des détecteurs de défaillances de portée limitée a été conçu pour résoudre ce nouveau problème sous l'hypothèse que certaines contraintes portant sur le nombre de pannes f, la portée du détecteurs de défaillances y et la valeur du paramètre k soient satisfaites.

– **Classification des problèmes**

Nous avons proposé une classification des problèmes en fonction de leur résistances intrinsèque aux défaillances [22]. Par hypothèse, nous avons considéré des systèmes répartis asynchrones munis de canaux de communication fiables. Un problème est considéré comme facile dès lors qu'il peut être résolu quelque soit le nombre de défaillances : c'est le cas du problème de la diffusion fiable. Dans le cas contraire, un problème est considéré comme potentiellement difficile. Une distinction peut être faite entre les problèmes potentiellement difficiles : un sous ensemble de ceux-ci sont des problèmes intrinsèquement difficiles (comme par exemple le problème de la validation atomique non bloquante ou le problème de la construction d'un détecteur de défaillance parfait). Nous avons établi que le consensus est un problème potentiellement difficile mais pas intrinsèquement difficile. Par ailleurs, nous avons montré que tout problème potentiellement difficile peut se réduire au problème de la construction d'un détecteur de défaillances parfait.

– **Définition d'une brique de base**

Le consensus est un problème d'accord élémentaire dont la définition simple est unanimement reconnu 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 dériver des services de communication de groupe efficaces. Diverses variantes au problème du consensus ont donc été explorées. Le but poursuivi est de concevoir une brique de base parfaitement adaptée aux problèmes d'accords que l'on désire résoudre.

Dans cette optique, nous avons défini un protocole d'accord générique qui peut être instancié en fonction du problème d'accord que l'on souhaite résoudre (diffusion ordonnée, gestion de la composition du groupe) [24]. L'idée consiste à générer automatiquement des protocoles d'accord ad hoc. 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).

Ce genre d'extension présente des intérêts très variés comme le montre les travaux consacrés au problème du consensus dans un environnement composé de stations mobiles [17, 16]. Ces travaux ont été réalisés en collaboration avec Raimundo Macedo (Université de Bahia, Brésil) et Nadjib Badache (Université d'Alger, Algérie).

Le problème du calcul d'une donnée globale cohérente (un vecteur comportant une entrée par processus) peut être résolu grâce à ce protocole générique. L'importance du problème nous a conduit à rechercher un protocole spécifique, efficace en temps, utilisable uniquement si on dispose de détecteurs de défaillances parfaits [23]. Ce protocole permet des décisions au plus tôt. Soit t le nombre maximal de pannes pouvant se produire et soit f le nombre de pannes effectives ($f < t$). Dans le pire des cas, le protocole termine après $min(2f+2, t+1)$ tentatives. De plus, ce protocole ne nécessite pas d'échanges d'informations sur la perception locale des pannes.

– **Prise en compte du contraintes temporelle**

Notre objectif est de maîtriser la terminaison des protocoles exécutés en permettant de gérer au mieux une durée maximale d'exécution. Or, sans hypothèses supplémentaires, l'approche adoptée pour résoudre le problème du consensus ne permet pas de maîtriser le délai nécessaire à une prise de décision en période de fort asynchronisme.

Nous avons montré qu'il était possible de gérer "la qualité de service" fournie par un protocole d'accord et d'optimiser celle-ci en fonction de l'environnement et des pannes pouvant survenir. Pour cela nous nous plaçons dans un modèle asynchrone augmenté : nous supposons que les canaux de communication peuvent être décrits de manière probabiliste. Les processus sont supposés avoir accès à une horloge locale dont la dérive par rapport au temps réel est bornée. L'approche probabiliste que nous proposons [37] permet, sans remettre en cause la structure interne des protocoles de consensus proposés, de limiter dans le temps la durée des tentatives tout en synchronisant leur déroulement. Le but est alors de déterminer pour un temps maximal souhaité (ou une probabilité de succès donnée) le meilleur couple (nombre de tentatives, durée d'une tentative). Pour réaliser ce calcul, des paramètres caractérisant l'environnement d'exécution (fonction de répartition des délais de transmission des messages, probabilité de perte des messages, probabilité de panne) doivent être pris en compte.

Nous avons également conçu un algorithme de détection des défaillances de groupe et proposé une interprétation probabiliste de cet algorithme [21, 11]. On dit qu'un groupe est défaillant lorsque l'un de ses composants est lui-même défaillant. Lorsqu'une telle défaillance survient, l'ensemble des membres du groupe survivants en sont informés par le biais d'un protocole de détection de défaillance de groupe. Dans l'absolu un tel protocole doit être vivace (les défaillances doivent être détectées) et sûr (lorsqu'une défaillance du groupe est détectée, alors au moins un des membres du groupe doit être défaillant). Nous avons analysé le comportement d'un protocole dont la vivacité est naturellement assurée. Nous avons montré qu'en le réglant de façon appropriée (via deux paramètres), il est possible de garantir sa propriété de sûreté avec une probabilité aussi proche de 1 que voulu.

– **Prise en compte des fautes Byzantines**
Les solutions actuelles ne permettent de résister qu'à des défaillances de type pannes franches. Un de nos objectifs est de les adapter afin de tolérer d'autres types de défaillances. Nous étudions en particulier comment étendre les protocoles de consensus que nous avons proposé afin de prendre en compte des fautes de type Byzantine.

– **Réplication de base de données**
Les primitives de communication de groupe peuvent être utilisées dans la mise en œuvre des bases de données répliquées. Chaque transaction utilise un nombre quelconque d'objets. Plutôt que de répliquer la base de donnée dans son intégralité sur chaque site, chaque objet peut être répliqué individuellement : l'ensemble des copies d'un même objet constitue alors un groupe. Nous travaillons actuellement sur la définition de services assurant à la fois la cohérence entre les copies d'objet et la sérialisation des transactions exécutées.

6.5 Environnements pour le travail coopératif assisté par ordinateur

Participants : Alain CÉDELLE, Stéphane LORCY, Géraldine TEXIER, Noël PLOUZEAU.

Mots clés : architectures à objets réparties, qualité de service, parallélisme, transactions, session de coopération.

Classification des mécanismes de programmation par contrats Les besoins en matière de suivi de la qualité de service reçue et offerte par les applications de travail coopératif ont nécessité la définition d'un nouveau modèle d'exécution de requête pour des architectures à objets réparties. Ce modèle d'exécution est fondé sur la notion de contrat entre un objet client et un système fournisseur de service. Dans le cadre de Rusken, le concept de contrat classique [Mey92] a été étendu par réification, pour permettre sa manipulation explicite lors de l'exécution du code des différents objets qui composent l'application. Dans le système que nous avons développé, un contrat est une classe d'objets qui permet de définir des propriétés d'exécution des services disponibles. Un contrat lie un objet client à un objet prestataire de service, en définissant les obligations de chacun, par exemple les pré- et postconditions d'exécution de méthode, les contraintes temporelles (délais ou instants) ou le degré d'exactitude d'une information (dernière valeur ou bien valeur approchée d'un résultat). La donnée de telles contraintes contribue à l'amélioration de la fiabilité des logiciels [RHG90]. Les difficultés inhérentes aux applications réparties (sensibilité aux variations de la qualité de service offerte par les systèmes de communication, défaillances de sites, partitions de réseaux) imposent la prise en compte de ces problèmes par le mécanisme de contrat. Le modèle proposé en 1997 par Rusken offre une abstraction des divers aléas de communication via une extension du concept de contrat. En 1999 une classification des types de contrats a été conçue en collaboration avec Antoine Beugnard (ENSTBr), Jean-Marc Jézéquel (projet Pampa de l'Irisa) et Damien Watkins (Monash university, Australie) et publiée dans un numéro spécial de la revue *Computer* consacré aux composants logiciels [3]. Cette classification examine les différents services que peuvent rendre les contrats et regroupe les différents types en quatre classes, depuis la seule donnée des types des paramètres jusqu'à la construction négociée de propriétés de qualité de service. Le type de contrat développé dans Rusken en 1998 correspond au quatrième niveau de cette classification.

Gestion de consensus pour des applications de réalité virtuelle multi-utilisateurs La simulation réaliste multi-utilisateurs se heurte à de nombreux problèmes liés notamment aux aléas de communication. En effet, ces aléas de la communication ne permettent pas de garantir une cohérence complète pour tous les utilisateurs connectés à un moment donné à un espace virtuel.

[Mey92] B. MEYER, « Applying "Design by Contract" », *Computer (IEEE) 25*, 10, octobre 1992, p. 40–51.

[RHG90] I. M. H. R. HELM, D. GANGOPADHAY, « Contracts : Specifying Behavioral Compositions in Object-Oriented Systems », *in : ECOOP/OOPSLA*, p. 169–179, octobre 1990.

Les travaux menés dans le cadre d'une coopération entre le projet Siames de l'Irisa et l'action Rusken ont conduit en 1999 à une proposition d'évolution de l'environnement de simulation mécanique développé depuis des années par le projet Siames . Cette évolution a pour but de rendre la plate-forme plus performante et mieux adaptée à la simulation répartie. L'évolution de la plate-forme Siames et la présentation générale des mécanismes de gestion de la cohérence ont été présentés dans [25].

Contrôle du parallélisme et de la coopération dans les applications de travail coopératif Les applications de travail coopératif ont des caractéristiques spécifiques liées à la nature même du service qu'elles rendent. En effet, toute application de travail coopératif est à la fois parallèle (plusieurs utilisateurs l'utilisent simultanément) et répartie (les utilisateurs sont géographiquement dispersés). Comme pour toute application parallèle, il est nécessaire que le parallélisme soit parfaitement traité et maîtrisé pour assurer la cohérence et l'exactitude des résultats. Cependant, les modèles de contrôle du parallélisme exigés par les applications de travail coopératif diffèrent notablement de ceux plus fréquemment rencontrés dans les domaines du calcul parallèle ou des bases de données [Ela92]. L'échelle de temps est radicalement différente : les opérations concurrentes effectuées sur des données par des applications de travail coopératif peuvent durer de quelques secondes à quelques mois. En effet, lorsque plusieurs utilisateurs travaillent simultanément dans un espace virtuel, ils accèdent à des objets tels que des lieux, des documents, des applications. La coopération des utilisateurs passe par l'usage concurrent de certains de ces objets. Pour structurer et expliciter cet usage, de nombreux modèles d'application de travail coopératif s'appuient sur la définition explicite et a priori de sessions de coopération. Notre modèle s'appuie au contraire sur la détection des accès concurrents et la construction de sessions dynamiques. Notre notion de session est moins contraignante car pilotée par les actes des utilisateurs et non imposée *a priori*. La mise en œuvre répartie s'appuie sur les services de contrats. Les travaux commencés en 1998 ont permis de proposer un algorithme de gestion de la coordination des actions des utilisateurs. L'algorithme a été présenté à la conférence PDPTA'99, dans la session spéciale consacrée au travail coopératif [33].

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Contrat avec Aérospatiale (projet Topase II)

Participant : Achour MOSTÉFAOUI.

Mots clés : Aérospatiale, téléformation, communication de groupe, recouvrement.

Résumé : *Le projet Topase II est la suite du projet Topase I soutenu par le ministère de l'industrie (convention Serics 96 2 93 02 91). Topase II est un contrat établi pour une durée de 18 mois (à partir de novembre 1999) principalement entre le* CNRS *(le projet* ADP *et le Laas), et Aérospatiale Matra Lanceurs. L'objectif de ce contrat est de considérer le domaine de l'Espace par le biais des formations multimedia assurées par Aérospatiale Matra Lanceurs pour le projet Ariane V. La diversité des sujets couverts par cette formation et sa généricité conduisent à mettre en place des moyens assurant des accès à une expertise répartie. Cette expertise sera associée à la classe virtuelle de Topase I. Cela conduit donc à la mise en place d'une coordination multi-groupe permettant la gestion expert/animateur/stagiaire et expert/animateur. Les situations pédagogiques abordées sont nouvelles. L'objectif est de produire des outils génériques permettant d'instrumenter des situations pédagogiques pour nombres d'autres domaines d'application.*

[Ela92] A. ELAMGARMID (éditeur), *Database transaction models for advanced applications*, Morgan Kauffman, 1992.

7.2 Contrat CNET sur les contraintes non-fonctionnelles

Participants : Fabiola GREVE, Michel HURFIN, Michel RAYNAL, Frédéric TRONEL.

Mots clés : sûreté de fonctionnement, temps réel, Corba.

Résumé : *Suite aux Consultations Thématiques Informelles lancées par le Cnet en mars 1997, un contrat a été établi entre le Cnet et l'équipe* ADP *pour les années 1998 à 2001 (convention 1 98 C 172 00 31401 012). Ce contrat porte sur l'étude et le développement d'un ensemble de services garantissant, dans un environnement asynchrone composé de calculateurs hétérogènes, deux types de contraintes non-fonctionnelles à savoir la sûreté de fonctionnement et le temps réel. Ce contrat doit déboucher sur la définition d'un ensemble de protocoles et la réalisation d'un prototype qui sera développé dans un environnement propre au* CNET *(Object Request Broker Jonathan).*

7.3 Contrat RÉUTEL 2000

Participants : Francisco BRASILEIRO, Fabiola GREVE, Michel HURFIN, Achour MOSTÉFAOUI, Jean-Pierre LE NARZUL, Frédéric TRONEL, Yun WANG.

Mots clés : sûreté de fonctionnement, temps réel, Corba.

Résumé : *L'équipe* ADP *est responsable de l'une des quatre actions définies dans le contrat Réutel 2000 (une description détaillée de ce contrat, qui a été signé en 1997, figure dans le rapport d'activité scientifique du projet Pampa). Dans le cadre de ce contrat Alcatel Alsthom Recherche souhaite bénéficier des travaux de recherche que nous effectuons dans les domaines de la reprise après défaillance et de la communication de groupe. Les travaux réalisés ont pour objectif d'enrichir la norme Corba en proposant des extensions logicielles qui permettront aux programmeurs de définir des services tolérant les défaillances dans un environnement asynchrone. L'approche proposée consiste à concevoir des services nécessaires à la gestion d'un groupe de serveurs dupliqués (gestion de la composition du groupe, diffusion atomique,...) en considérant ceux-ci comme des variantes d'un problème d'accord élémentaire : le consensus. Notre objectif est donc de développer un prototype en y intégrant l'ensemble des innovations proposées tout en respectant la future norme d'un service Corba de tolérance aux défaillances actuellement en cours de discussion au sein de l'*OMG.

8 Actions régionales, nationales et internationales

8.1 Actions nationales

8.1.1 GDR ARP (Architecture, Reseaux et Parallélisme) du CNRS

Participants : Achour MOSTÉFAOUI, Michel RAYNAL.

Le thème Réseaux et Systèmes de ce GDR est coordonnée par Michel Diaz (Laas). Le sous-groupe "Algorithmes pour les Systèmes Répartis" est animé par Joffroy Beauquier (LRI) et Michel Raynal (Irisa).

8.2 Actions européennes

8.2.1 Broadcast

Participants : Michel HURFIN, Michel RAYNAL.

Le projet ADP fait partie avec le projet Solidor du *working group* européen Esprit Broadcast sur les systèmes répartis de grande dimension. (Broadcast est un acronyme pour *Basic Research on Advanced Distributed Computing, from Algorithms to Systems*). Ce *working group* a été constitué en 1995, lorsque le projet européen Esprit du même nom s'est achevé.

8.2.2 Cabernet

Participants : Michel HURFIN, Michel RAYNAL.

Le projet ADP participe au projet Esprit Cabernet (*Computer Architecture for Basic European Research, Network of Excellence*) en collaboration avec le projet Solidor de l'Irisa et le projet Reflex de l'Inria-Rocquencourt depuis sa création en 1991.

8.2.3 Portugal

Participant : Noël PLOUZEAU.

L'action Rusken entretient des contacts étroits avec l'équipe de Paulo Veríssimo, à l'université de Lisbonne, Portugal. Un financement Inria/ICCTI de collaboration franco-portugaise pour une durée de deux ans a été obtenu en 1998. La collaboration porte sur les mécanismes de communication asynchrones temporisés utilisés par le projet ADP et l'action Rusken.

8.2.4 Italie

Participants : Jean-Michel HÉLARY, Michel RAYNAL.

Un projet de coopération avec l'Université de Rome La Sapienza sur le thème des points de contrôle et du retour arrière a été retenu pour financement dans le cadre des programmes d'accords CNRS-CNR 1998-1999.

8.3 Actions internationales

8.3.1 Amériques

USA **(Georgie)** *Participants* : Achour MOSTÉFAOUI, Michel RAYNAL.

Un projet de coopération avec M. Ahamad du Georgia Institute of Technology, sur le thème de la cohérence des mémoires réparties, a été accepté par la NSF et l'Inria pour un financement conjoint. Ce projet a débuté en 1998.

9 Diffusion de résultats

9.1 Actions d'enseignement

A l'exception d'un seul chercheur à temps plein, tous les membres du projet ADP sont enseignants-chercheurs. A ce titre, ils participent à bon nombre d'enseignements de second et de troisième cycles.

Par ailleurs J.M. Hélary est responsable de la filière Langages et Systèmes Informatiques du diplôme d'ingénieur en informatique de l'université de Rennes 1 (Diic).

Le projet a assuré plusieurs enseignements liés aux thèmes de ses recherches :

- dans le DEA d'informatique de Rennes, M. Raynal dispense un cours sur les algorithmes et systèmes répartis (31 heures) ; N. Plouzeau participe au cours *Conception à Objets et Test de Logiciels* (9 heures) ;
- dans le Diic 3^e année (filières langages et systèmes et architecture), J.M. Hélary et M. Raynal dispensent un cours sur l'algorithmique répartie (30 heures) et N. Plouzeau participe au cours *Logiciels à Objets* (12 heures) ;
- à l'ENSTBr (antenne de Rennes), un cours sur l'algorithmique répartie est assuré par M. Raynal. M. Hurfin et M. Raynal dispensent également un cours sur la tolérance aux défaillances à l'ENSTBr de Brest (6 heures chacun) ;
- dans le DESS ISA, M. Raynal donne un cours sur les systèmes transactionnels, la cohérence et la sécurité des données. N. Plouzeau donne un cours sur la conception des systèmes à objets.
- A. 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 libanaise de Beyrouth.
- M. Hurfin a dispensé un cours sur la tolérance aux défaillances aux étudiants de Master of Science à l'Université de Bahia au Brésil.

Un stagiaire de DEA a effectué son stage au sein de l'équipe ADP. Il s'agit de Cyril Ray qui a travaillé sur la tolérance aux défaillances par reprise arrière dans les MVP.

9.2 Presentation de travaux

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

M. Raynal a présenté les travaux de l'équipe et fait des séminaires dans les universités et centres de recherche suivants :

- Unam, université national de Mexico (Mexico, Mexique),
- UAM, Université Autonome de Mexico (Mexico, Mexique),
- Georgia Institute of Technology, (Atlanta, Géorgie)
- université d'Austin, (Austin, Texas),
- université La Sapienza de Rome (Rome, Italie).

J.M. Hélary a présenté les travaux de l'équipe à :

- Georgia Institute of Technology, (Atlanta, Géorgia)
- université de Reims.

A. Mostéfaoui a présenté les travaux de l'équipe à l'université d'Alger (Algérie).

M. Hurfin a présenté les travaux de l'équipe à l'université de Bahia (Salvador, Brésil).

9.3 Animation de la communauté scientifique

M. 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*.

M. Raynal a été (en 1999) ou sera (en 2000) membre des comités de programme des conférences suivantes :

- *4th Workshop on Fault-Tolerance in Parallel and Distributed Systems*. Workshop satellite de *13th Int. Parallel Processing Symposium (*IPPS*)*.
  Porto-Rico, avril 1999. (actes publiés par Springer-Verlag.)

- Président du *2d* IEEE *Int.Symposium on Object-Oriented Real-Time Distributed Computing (*ISORC'99*)*.
  Saint-Malo, mai 1999. (Actes publiés par IEEE.)
- Président du thème *Distributed Synchronization* de *19th* IEEE *Int. Conf. on Distributed Computing Systems*.
  Austin TX, (mai 1999). (Actes publiés par IEEE.)
- *International Workshop on Network-Based Information Systems (*NBIS'99*)*.
  Florence, septembre 1999. (Actes publiés par IEEE.)
- *6th International Conference on Parallel and Realtime Systems (Part'99)*.
  Melbourne, Australia, Dec 1999.
- *20th* IEEE *Int. Conf. on Distributed Computing Systems*.
  Taiwan, avril 2000. (Actes publiés par IEEE.)
- *3rd* IEEE *Int. Symposium on Object-Oriented Real-time Distributed Computing*.
  Newport Beach, CA, mars 2000. (Actes publiés par IEEE.)
- *Int. Conf. on Dependable Systems and Networks (previously* IEEE *Symposium on Fault-Tolerant Systems)*.
  New York, june 2000. (Actes publiés par IEEE.)

Achour Mostéfaoui est membre du comité de programme de trois conférences.

- *3rd European Research Seminar on Advances in Distributed Systems (*ERSADS*)*.
  Madeira, Portugal, avril 1999.
- *19th* IEEE *Int. Conf. on Distributed Computing Systems*.
  Austin TX, mai 1999.
- *4th Int. Symposium on Programming and Systems (*ISPS'99*)*,
  Alger, octobre 1999.

Michel Hurfin a été co-président du "Advisory and publicity commitee" du *2nd* IEEE *Int.Symposium on Object-Oriented Real-Time Distributed Computing (*ISORC'99*)*, Saint-Malo, mai 1999.

Noël Plouzeau a animé un atelier "Programmation par contrats" à Objets'99 en mai 1999 à l'Ecole des Mines de Nantes.

9.4 Coopération et échanges internationaux

Amr El Abbadi, Professeur à l'Université Santa Barbara (Californie) a été accueillis durant un mois en juillet-août. Il a abordé avec les membres de l'équipe les problèmes liés à la mise à jour de "warehouse". Une proposition de coopération NSF/Inria a été déposée.

Francisco Torres-Rojas, Doctorant au GeorgiaTech (Atlanta) a effectué un séjours d'une semaine (en mars) dans le cadre de la convention NSF/Inria sur les données dupliquées.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

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

[2] J. BRZEZINSKI, J. M. HÉLARY, M. RAYNAL, M. SINGHAL, « Deadlock models and a general algorithm for distributed deadlock detection, », *Journal of Parallel and Distributed Computing 31*, 2, 1995, p. 112–125.

[3] J. M. HÉLARY, A. MOSTEFAOUI, M. RAYNAL, « A general scheme for token and tree based distributed mutual exclusion algorithm », *IEEE Transactions on Parallel and Distributed Systems 5*, 11, November 1994, p. 1185–1196.

[4] 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*, p. 32–42, San Diego, Californie, May 1993. Reprinted in SIGPLAN Notices, vol. 28,12, December 1993.

[5] M. RAYNAL, A. SCHIPER, S. TOUEG, « The Causal Ordering Abstraction and a Simple Way to implement it », *Information Processing Letters 39*, September 1991, p. 343–351.

[6] M. RAYNAL, M. SINGHAL, « Logical Time: Capturing Causality in Distributed Systems », *IEEE Computer 29*, 2, février 1996, p. 49–57.

[7] A. SCHIPER, M. RAYNAL, « From group communication to transactions in distributed systems », *Communications of the ACM 39*, 4, april 1996, p. 84–90.

Articles et chapitres de livre

[1] M. AHAMAD, M. RAYNAL, G. THIA-KIME, « An adaptive architecture for causally consistent distributed services », *Distributed Systems Engineering Journal 6*, 2, 1999, p. 63–70.

[2] R. BALDONI, J. HÉLARY, M. RAYNAL, « Rollback-dependency trackability: a minimal characterization and its protocol », *Information and Computation*, 2000, à paraître.

[3] A. BEUGNARD, J. JÉZÉQUEL, N. PLOUZEAU, D. WATKINS, « Making Components Contract Aware », *Computer 32*, 7, juillet 1999, p. 38–45.

[4] V. GARG, M. RAYNAL, « Normality: a Consistency criterion for concurrent objects », *Parallel Processing Letters 9*, 1, 1999, p. 123–134.

[5] R. GUERRAOUI, M. HURFIN, A. MOSTEFAOUI, R. OLIVEIRA, M. RAYNAL, A. SCHIPER, *Advances in Large Scale Distributed Computing*, *LNCS*, 1752, Springer-Verlag, 1999, ch. Consensus in Asynchronous Distributed Systems: A Concise Guided Tour, nom d'auteur collectif W.G. Broadcast.

[6] J. HÉLARY, A. MOSTEFAOUI, R. NETZER, M. RAYNAL, « Communication-based prevention of useless checkpoints in distributed computations », *Distributed Computing 13*, 1, 1999.

[7] J. HÉLARY, A. MOSTEFAOUI, M. RAYNAL, « Communication-induced determination of consistent snapshots », *IEEE Transactions on Parallel and Distributed Systems 10*, 9, sept 1999, p. 865–877.

[8] J. HÉLARY, R. NETZER, M. RAYNAL, « Consistency issues in distributed checkpoints », *IEEE Transactions on Software Engineering 25*, 4, 1999, p. 274–281.

[9] M. HURFIN, M. RAYNAL, « A Simple and Fast Asynchronous Consensus Protocol Based on a Weak Failure Detector », *Distributed Computing 4*, 12, 1999, p. 209–223.

[10] A. MOSTEFAOUI, M. RAYNAL, M. TAKIZAWA, *Dependable Network Computing*, Kluwer Academic Press, 1999, ch. Chapter 4: Reliable Logical Clocks for Unreliable Process Groups.

[11] M. RAYNAL, F. TRONEL, « Group membership failure detection: a simple protocol and its probabilistic analysis », *Distributed Systems Engineering Journal*, 1999, p. 95–109.

[12] M. RAYNAL, F. TRONEL, « Restricted Failure Detectors: Definition and Reduction Protocols », *Information on Processing Letters 6*, 4, 2000.

[13] M. RAYNAL, P. VERISSIMO, *Advances in Large Scale Distributed Computing*, *LNCS*, 1752, Springer-Verlag, 1999, ch. Time in Distributed Systems: Models and Algorithms, nom d'auteur collectif W.G. Broadcast.

[14] M. RAYNAL, *Encyclopedia of Distributed Computing*, Kluwer Academic Press, 1999, ch. Synchronization.

[15] M. RAYNAL, « Non-Blocking Atomic Commitment in Distributed Systems: A tutorial based on a generic protocol », *Journal of Computer Systems Science and Engineering 14*, 1999.

Communications à des congrès, colloques, etc.

[16] N. BADACHE, M. HURFIN, R. MACEDO, « A Solution for the Consensus Problem in a Mobile Environment », *in : Proc. of the 17th Simposio Brasileiro de Redes de Computadores*, p. 273–288, Salvador, Bahia, 1999.

[17] N. BADACHE, M. HURFIN, R. MACEDO, « Solving the Consensus Problem in a Mobile Environment », *in : Proc. of the 18th IEEE International Performance, Computing, and Communications Conference (IPCCC'99)*, p. 29–35, Phoenix, Arizona, 1999.

[18] R. BALDONI, G. CIOFFI, J. HELARY, M. RAYNAL, « Direct dependency-Based Determination of Consistent Global Checkpoints », *in : Proc. of the 3rd Int. Symposium On Principles Of Distributed Systems (OPODIS'99)*, p. 11–28, Hanoi, octobre 1999.

[19] R. BALDONI, J.-M. HELARY, M. RAYNAL, « Rollback Dependency Trackability: Visible Characterizations », *in : Proc. of the 18th ACM SIGACT-SIGOPS Int. Symposium on Principles of Distributed Computing (PODC'99)*, p. 33–42, Atlanta, 1999.

[20] R. BALDONI, F. QUAGLIA, M. RAYNAL, « Consistent checkpointing for distributed databases », *in : Proc. of the 5th Int. European Parallel Computing Conference (EUROPAR'99), LNCS 1685*, p. 450–458, Toulouse, 1999.

[21] R. BOLLO, J. LE NARZUL, M. RAYNAL, F. TRONEL, « Probabilistic Analysis of a Group Failure Detection Protocol », *in : Proc. of the 4th IEEE Int. Workshop on Object-Oriented Real-Time Dependable Systems*, p. 156–162, Santa Barbara, CA, janvier 1999.

[22] E. FROMENTIN, M. RAYNAL, F. TRONEL, « On Classes of Problems in Asynchronous Distributed Systems with Process Crashes », *in : Proc. of the 19th IEEE Int. Conf. on Distributed Computing Systems*, p. 471–477, Austin, 1999. Best paper award.

[23] J. HÉLARY, M. HURFIN, A. MOSTEFAOUI, M. RAYNAL, F. TRONEL, « Computing Global Functions in Asynchronous Distributed Systems with Process Crashes », *in : Proc. of the 20th IEEE Int. Conf. on Distributed Computing Systems*, Taipei, 2000. à paraître.

[24] M. HURFIN, R. MACEDO, M. RAYNAL, F. TRONEL, « A General Framework to Solve Agreement Problems », *in : Proc. of the 18th IEEE Symposium on Reliable Distributed Systems, (SRDS'99)*, p. 56–65, Lausanne, octobre 1999.

[25] D. MARGERY, N. PLOUZEAU, B. ARNALDI, « A General Framework for Cooperative Manipulation in Virtual Environments », *in : 5th eurographics workshop on virtual environments (EGVE'99) Special focus on Augmented Reality*, Vienna, may 1999.

[26] A. MOSTEFAOUI, M. RAYNAL, M. TAKIZAWA, « Consistent Lamport's clocks for asynchronous groups with process crashes », *in : Proc. of the 5th Int. Conference on Parallel Computing Technologies (PACT'99), LNCS 1662*, p. 98–107, St-Petersburg, 1999.

[27] A. MOSTEFAOUI, M. RAYNAL, P. VERISSIMO, « Efficient Logically Instantaneous Communications in Asynchronous Distributed Systems », *in : Proc. of the 5th Int. Conference on Parallel Computing Technologies (PACT'99), LNCS 1662*, p. 258–270, St-Petersburg, 1999.

[28] A. MOSTEFAOUI, M. RAYNAL, « Solving Consensus Using Chandra-Toueg's Unreliable Failure Detectors: a General Quorum-Based Approach », *in : Proc. of the 13th Int. Symposium on Distributed Computing (DISC'99) (formerly, WDAG), LNCS 1693 (P. Jayanti Ed.)*, p. 49–63, Bratislava, 1999.

[29] A. MOSTEFAOUI, M. RAYNAL, « Unreliable failure Detector with Limited Scope Accuracy and an Application to Consensus », *in : Proc. of the 19th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'99), LNCS 1738, (P.S. Thiagarajan Ed.)*, p. 317–328, Chennai, India, décembre 1999.

[30] M. RAYNAL, « Illustrating the Use of Vector Clocks in Property Detection: an Example and a Counter-Example », *in : Proc. of the 5th Int. European Parallel Computing Conference (EUROPAR'99), LNCS 1685*, p. 806–814, Toulouse, 1999.

[31] M. RAYNAL, « Simple Vector Clocks are Limited to Solve some Causality-Related Problems », *in : Proc. of the 3rd Int. Symposium On Principles Of Distributed Systems (OPODIS'99)*, p. 199–208, Hanoi, octobre 1999. invited paper.

[32] L. RODRIGUES, M. RAYNAL, « Atomic Broadcast in Asynchronous Crash/Recovery Distributed Systems », *in : Proc. of the 20th Int. Conf. on Dictributed Computing Systems*, Taipei, 2000. à paraître.

[33] G. TEXIER, N. PLOUZEAU, « Automatic Management of Sessions in Shared Spaces », *in : Proc. of Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, C. Press (éditeur), *1*, 1999.

[34] G. TEXIER, « Travail coopératif : des applications réparties particulières », *in : Actes de la première Conférence Française sur les Systèmes d'Exploitation (CFSE'1)*, INRIA (éditeur), 1999.

[35] F. TORRES, M. AHAMAD, M. RAYNAL, « Timed Consistency for Shared Distributed Objects », *in : Proc. of the 18th ACM SIGACT-SIGOPS Int. Symposium on Principles of Distributed Computing (PODC'99)*, p. 163–172, Atlanta, 1999.

Rapports de recherche et publications internes

[36] A. FERNÁNDEZ, A. MOSTEFAOUI, M. RAYNAL, « A Necessary and Sufficient Condition for Transforming Limited accuracy Failure Detectors », *rapport de recherche n° 1280*, IRISA, décembre 1999, 15 pages, `http://www.irisa.fr/EXTERNE/bibli/pi/1287/1287.html`.

[37] E. FROMENTIN, F. TRONEL, « A Probabilistic Analysis of the Consensus Problem », *rapport de recherche n° 1226*, IRISA, janvier 1999, 23 pages, `http://www.irisa.fr/EXTERNE/bibli/pi/1226/1226.html`.

[38] A. MOSTEFAOUI, M. RAYNAL, F. TRONEL, « The Best of Both Worlds: a Hybrid Approach to Solve Consensus », *rapport de recherche n° 1280*, IRISA, novembre 1999, 20 pages, `http://www.irisa.fr/EXTERNE/bibli/pi/1280/1280.html`.

[39] A. MOSTEFAOUI, M. RAYNAL, « K-set Agreement with Limited Accuracy Failure Detectors », *rapport de recherche n° 1268*, IRISA, octobre 1999, 20 pages, `http://www.irisa.fr/EXTERNE/bibli/pi/1268/1268.html`.