INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet ADP

Algorithmes distribués et protocoles

Rennes

THÈME 1C

Table des matières

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 CNRS
Emmanuelle Anceaume [CR]
Personnel Université de Rennes I
Jean-Michel Hélary [professeur]
Philippe Ingels [maître de conférences]
Achour Mostéfaoui [maître de conférences]
Maria Gradinariu [maître de conférences, à partir de septembre 2001]
Collaborateur extérieur
Jean-Pierre Le Narzul [maître de conférences à l'ENSTBr]
Chercheurs invités
Yun Wang [Professeur, Southeast University, Chine, 9-12 janvier 2001]
Raimundo Macedo [Professeur, Federal University of Bahia, Brésil, 13-23 mars 2001]
Shen Zhuowei [Doctorant, Southeast University, Chine, 11 mai-15 juin, 2001]
Leslie Lamport [Professeur, Compaq Research Lab, Palo Alto, USA, 13-20 mai 2001]
Roberto Baldoni [professeur, Université La Sapienza, Rome, Italie, 26 novembre-9 décembre 2001]
Chercheurs doctorants
Erwan Demairy [ATER]
Udo Fritzke [bourse du gouvernement brésilien, jusqu'en février 2001]
Fabiola Greve [bourse du gouvernement brésilien]
Eric Mourgaya [bourse INRIA]
Philippe Raïpin Parvédy [bourse du ministère, à partir d'octobre 2001]
Matthieu Roy [allocation couplée]
Frédéric Tronel [allocation couplée, jusqu'en septembre 2001]

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 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

[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. 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.

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

3.2 Systèmes transactionnels dupliqués fondés sur la communication de groupes

Mots clés : transaction, duplication, diffusion, tolérance aux défaillances.

Résumé : *La duplication augmente la disponibilité des données et permet d'exécuter des transactions même en cas de défaillances. Dans les systèmes transactionnels dupliqués, au delà d'assurer l'isolation des transactions, la duplication doit être rendue transparente à l'utilisateur, et cela tout en tolérant les défaillances. En raison de la distribution des données, on doit également mettre en œuvre une validation atomique répartie pour les transactions. Par ailleurs, contrairement aux systèmes non-dupliqués, les transactions impliquent des ressources qui sont multipliées sur un système réparti et par conséquent, elles sont plus sensibles aux problèmes d'interblocage. Le traitement de ces problèmes s'avère d'autant plus difficile que l'on se place dans le cadre des systèmes répartis asynchrones.*

Les primitives de diffusion peuvent y jouer un rôle déterminant dans la mesure où elles rendent les défaillances et l'asynchronisme transparents à l'application. Enfin, elles procurent des propriétés d'ordre qui rendent l'implémentation du contrôle de la duplication plus simple.

La duplication de données a été utilisée depuis longtemps pour assurer que les données accédées par des transactions demeurent disponibles en dépit des défaillances. De nombreux protocoles de contrôle de duplication qui assurent la "sérialisabilité à une copie" (one-copy serializability) des transactions ont été proposés. Ces protocoles tolèrent différentes situations de défaillances (défaillances de processus ou partitionnement du réseau, par exemple), mais d'autre part, ils négligent d'autres aspects liés à la vivacité des transactions. En l'occurrence, ni l'interblocage de transactions ni la validation atomique non-bloquante de celles-ci ne sont traités par ces protocoles. Certes, cela se doit en grande partie au fait que l'on ne pose que des hypothèses faibles concernant les propriétés des systèmes de communication sous-jacents. Souvent, on ne considère que des primitives de communication point-à-point, voire, sans garantie de fiabilité de la livraison des messages. La duplication s'avère encore plus difficile à gérer lorsque l'on se place dans le cadre des systèmes répartis asynchrones, dans lesquels aucune hypothèse sur les vitesses des communications ou des processeurs n'est faite. Il est connu que des problèmes d'accord concernant la validation atomique et le verrouillage réparti sont impossibles à résoudre dans les systèmes asynchrones.

[BFR96] O. Babaoglu, E. Fromentin, M. Raynal, « A Unified Framework for the Specification and Runtime 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.

À l'opposé des primitives de communication considérées par les protocoles traditionnels, les systèmes de communication de groupes procurent, entre autres, des primitives de diffusion de messages avec des garanties fortes de fiabilité et d'ordre de livraison. Par ailleurs, il existe des protocoles qui implémentent ces primitives de diffusion dans les systèmes asynchrones. Avec l'utilisation de tels services de diffusion au sein des systèmes transactionnels dupliqués, les aspects concernant le traitement d'interblocage et la validation atomique ont pu être facilement incorporés aux protocoles de contrôle de duplication, et cela même dans un cadre asynchrone.

Des travaux récents ont proposé cette intégration de la communication de groupe avec la gestion de la duplication. Pourtant, on y considère un modèle de duplication *complète*, où chaque processus du système réparti possède une copie complète d'une seule base de données. Contrairement à cette duplication complète, nous considérons un modèle de duplication *partielle* où chaque donnée peut avoir des copies sur un nombre quelconque de processus. Plus précisément, nous considérons que le système réparti est structuré en plusieurs groupes de processus, chaque groupe réalisant la duplication d'un objet particulier. Les transactions peuvent donc accéder à des objets sur différents groupes de processus. En conséquence, pour assurer les propriétés transactionnelles, on doit mettre en œuvre un système de gestion de transactions réparti.

Dans la duplication complète on utilise un cas particulier de diffusion, dans laquelle un message est destiné à tous les processus du système réparti ("broadcast"). En revanche, pour gérer de manière efficace la duplication partielle, les messages doivent être diffusés vers des groupes de processus, différents messages pouvant avoir des ensembles de groupes destinataires distincts ("multicast"). Les primitives de *Diffusion fiable uniforme* et de *Diffusion fiable totalement ordonnée* (ou *Diffusion atomique*) vers plusieurs groupes ont ces caractéristiques. La première primitive implémente une propriété de fiabilité qui garantit qu'un message est inéluctablement délivré par les processus corrects des groupes destinataires si l'émetteur est correct, ou si au moins l'un des processus a délivré ce message. La deuxième met en œuvre cette fiabilité et en plus assure que les messages sont délivrés en respectant un ordre total.

Pour faire face aux systèmes répartis en grande échelle, il est important par ailleurs que les protocoles de diffusion réalisent la propriété de *Minimalité*. Cette propriété garantit que seuls le processus émetteur et les processus des groupes destinataires d'un message sont impliqués dans sa diffusion et sa livraison.

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 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 [NJ95] ont introduit la notion de *Z-chemin* entre points de

[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.

[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.

[NJ95] 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.

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 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.

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

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 messages en 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* s'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.

systèmes répartis

3.4 Les problèmes d'accord dans les systèmes répartis

Mots clés : algorithme réparti, tolérance 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 exactitude faible inéluctable (il existe un instant à partir duquel un processus correct ne sera plus jamais suspecté par aucun processus correct). En s'appuyant sur des détecteurs de défaillances de cette classe et à condition que les canaux de communication soient fiables et qu'une majorité de processus ne subit pas de défaillances, des algorithmes déterministes permettent de résoudre le problème du consensus. Tous s'appuient sur le paradigme du coordinateur tournant mais diffèrent par leur complexité en temps et en nombre de messages. L'utilisation de tels algorithmes comme briques de base dans la résolution de problèmes d'accord, et par conséquent leur utilisation dans la gestion des groupes de processus dupliqués, est actuellement un axe de recherche important.

Si le problème du consensus est représentatif d'une large classe de problèmes d'accord, il en existe qui ne sont pas réductibles au consensus. Ce sont, notamment, le problème de l'accord global et le problème de la diffusion fiable avec terminaison. En se plaçant dans

[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.

un cadre synchrone et en utilisant le paradigme des protocoles à rondes (il s'agit alors de rondes synchronisées), Dolev, Reischuk et Strong [DDS90] ont montré qu'il fallait au moins $\min(t+1, f+2)$ rondes (où t est le nombre maximal de crashs tolérés, et f le nombre effectif de crashs ayant lieu pendant l'exécution) pour parvenir à l'accord global. Dans un contexte asynchrone avec pannes franches, ces problèmes d'accord "difficiles" ne peuvent être résolus que si le modèle est augmenté avec des détecteurs parfaits. Or, dans un tel contexte, le modèle de calcul obtenu (rondes asynchrones + détecteurs parfaits) n'est pas aussi fort que celui des rondes synchronisées. Ceci soulève une importante question, théorique mais aussi pratique : quelle est la borne inférieure du nombre de rondes nécessaires pour parvenir à l'accord global ?

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.

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.

3.5 Aide à la conception des applications multimédias

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

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

La généralisation des applications multimedia conduit inéluctablement à modifier l'infrastructure qui répartit les ressources entre les applications. Jusqu'à l'émergence du multimedia, les exigences de qualité de service des applications effectuant des transferts d'information se

[DDS90] R. R. D. Dolev, R. Strong, « Early Stopping in Byzantine Agreement », *Journal of the ACM 37*, 4, 1990, p. 720–741.

[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.

limitaient le plus souvent à vérifier l'intégrité des données échangées, afin d'éviter, par exemple, la corruption des fichiers transférés. Dans ce contexte, les ressources ne sont en général pas réservées, ce qui rend imprévisible la durée d'un traitement. Ce modèle, parfois qualifié de « paresseux (*best effort*) », est acceptable pour des applications où le traitement effectivement effectué est prépondérant sur les délais. *A contrario*, le multimedia se caractérise, outre l'exigence - devenue « seulement » statistique - d'intégrité des données, par le respect de garanties temporelles, ce qui nécessite une certaine disponibilité du processeur, de la bande passante, de la mémoire, etc. [BS97].

Le premier problème à résoudre est de vérifier la *cohérence des formats des données* échangées, afin de savoir si les différents équipements utilisés sont compatibles. Comme les flux multimedia sont maintenant bien standardisés, il n'est pas nécessaire de connaître le détail des protocoles utilisés. On peut considérer que ces flux sont typés pour effectuer la vérification de cohérence des formats des données échangées.

Le second problème est posé par le respect des *contraintes temporelles*. Il s'agit de savoir si les délais et les fréquences d'arrivée des données sont compatibles avec les attentes des utilisateurs. Les contraintes temporelles sont de trois types. Les contraintes de synchronisation intra-flux expriment la régularité du flux en précisant le délai d'arrivée entre deux trames d'information. Les synchronisations inter-flux précisent les délais qui peuvent séparer l'arrivée des trames sur plusieurs flux liés. Enfin, le délai de bout-en-bout concerne le délai qui s'écoule entre le passsage d'une trame en un point de l'application et son arrivée en un autre point. Afin de vérifier le respect de ces contraintes, il est nécessaire de pouvoir connaître les performances des éléments employés, et de pouvoir en déduire si leur composition respecte les contraintes temporelles qui sont exprimées sur l'application.

Le comportement temporel des éléments constituant l'application ne peut être garanti que si une certaine quantité de ressources est disponible. En effet, il est possible de trouver au sein de la plate-forme d'exécution plusieurs applications multimedia qui s'exécutent en concurrence. Dans ce cas, les ressources du calculateur et du réseau ont à être partagées par toutes les applications.

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

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

Le rôle des architectures qualité de service (que nous abrégeons dorénavant par AQoS) est

[BS97] G. Blair, J. Stephani, « Open Distributed Processing and Multimedia », 1997.

d'effectuer la réservation et la renégociation des ressources de « bout-en-bout »[ACH95]. Ces mécanismes sont suffisants pour des applications où le serveur est lié au client directement par un réseau, comme c'est le cas souvent pour les applications de vidéo à la demande ; mais ces AQDS ne permettent pas d'appréhender correctement les liaisons de bout-en-bout d'une application qui mettent en jeu plusieurs liens de machine à machine.

Quant aux architectures qualité de service qui ne font pas d'hypothèse sur les éléments qui sont placés entre les extrémités, la complexité de leur représentation rend douteuse l'utilité à grande échelle des algorithmes proposés.

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

4 Domaines d'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.

4.2 Le concept de communication de groupe

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

[ACH95] C. Aurrecoecha, A. Campbell, L. Hauw, « A review of Quality of Service Architectures », *ACM Multimedia Systems Journal*, nov 1995.

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

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

4.3 Utilisations considérées

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

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

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

gements de vue qui est réalisée par le service de communication de groupe s'avère alors être un service majeur.

5 Logiciels

5.1 EDEN : un service de communication de groupe

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

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

Résumé : *Depuis plus de trois ans, 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 répondant aux exigences mentionnées dans l'appel à propositions rédigé par l'OMG sur le thème de la tolérance aux défaillances dans CORBA.*

Description de EDEN La plate-forme EDEN offre des services de communication de groupe (construction d'un ordre total sur les requêtes adressées au groupe, gestion de la composition du groupe et de l'installation des vues). 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. Le système réparti servant de support à l'exécution du groupe est supposé *asynchrone* :aucune hypothèse n'est faite concernant les temps de transfert des messages ou les vitesse d'exécution des membres du groupe qui peuvent éventuellement s'exécuter sur des machines hétérogènes.

La plate-forme EDEN a été développée en utilisant un langage orienté objet (Java). EDEN est composée de deux modules appelés EVA et ADAM. EVA (EVent based Architecture) est un framework utilisant le concept de communication par événements pour permettre au programmeur une mise en oeuvre simple et efficace de protocoles de communication spécialisés. Les services de communication de groupe développés à l'aide de EVA sont regroupés au sein du module appelé ADAM.

- EVA

  Un protocole de communication de groupe est un exemple typique de protocole de communication spécialisé de haut niveau. La structuration par couche qui est habituellement utilisée pour spécifier les protocoles de communication élémentaires, présente de nombreux défauts lorsqu'il s'agit de concevoir des protocoles de plus haut niveau de la même façon. Pour éviter les inconvénients d'une architecture à couche tout en gardant les avantages d'une décomposition d'un problème en services de base indépendants, nous avons opté pour une architecture fondée sur le concept de notification d'événements. Les interactions entre les entités qui mettent en oeuvre la fonctionnalité désirée sont obtenues

via des productions d'événements (données en sortie) par des entités productrices et des consommations d'événements (données en entrée) par des entités consommatrices. Un canal d'événement se charge de diriger les événements produits vers les consommateurs intéressés. Cette approche permet de construire des applications performantes, très modulaires et reconfigurables dont les composants peuvent être exécutés en parallèle. Lors du développement d'Eva, nous avons cherché à utiliser des concepts bien établis (service de notification d'événements, *design patterns*) dont la combinaison permet de bâtir des applications de façon simple et méthodique sans pour autant sacrifier l'efficacité du code généré. Une description détaillée d'Eva et de ses principaux atouts est disponible dans [25].

– Adam
Le module Adam offre des services de communication de groupe. Plus précisément, il permet d'une part de gérer dynamiquement la composition d'un groupe (ajout et retrait de membres) et d'autre part de garantir que toutes les requêtes diffusées à l'intention des membres du groupe seront traitées dans le même ordre. Ces requêtes peuvent avoir été générées par un membre du groupe ou par des objets extérieurs au groupe. L'entrelacement entre les changements de composition du groupe et les livraisons de messages respectent les règles classiques définies dans les systèmes de communication de groupe existants (*view synchrony*). L'originalité de la solution proposée vient du fait que tous ces services sont développés au dessus d'un unique service d'accord générique. Les intérêts de cette approche (modularité, meilleure maîtrise du comportement en période de fort asynchronisme, pré-réglage automatique des délais de garde utilisés, détection simple des pertes de messages, possibilité de prendre en compte des contraintes temporelles) sont décrits aux travers des publications du projet parues durant ces trois dernières années.

6 Résultats nouveaux

6.1 Détection de propriétés

Participants : Emmanuelle Anceaume, Jean-Michel Hélary, 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.*

Les horloges vectorielles constituent le mécanisme le plus approprié pour capter la causalité entre les événements produits lors de l'exécution d'un calcul réparti. Les implémentations traditionnelles des horloges vectorielles nécessitent d'attacher des vecteurs d'entiers de taille n aux messages de l'application (où n est le nombre de processus). Dans ce cadre, nous nous intéressons à la capture des dépendances causales sur un sous-ensemble d'événements que l'application considère comme pertinents. Dans [40] nous présentons une famille de protocoles

qui associent à chaque événement pertinent une estampille identifiant de manière unique ses prédécesseurs immédiats. Cette famille est définie par une condition abstraite qui permet aux messages applicatifs de transmettre des informations de contrôle de taille bornée ($< n$, n étant le nombre de processus). Cette famille définit une suite de protocoles IPT efficaces.

Nous avons aussi étudié la communication logiquement instantanée (LI) [18]. C'est un mode de communication plus contraignant que la communication causale mais moins que le rendez-vous. En effet lors d'un rendez-vous un événement d'émission s'exécute en même temps que l'événement de réception correspondant. La communication LI assure qu'il existe un estampillage logique (référentiel logique de temps) tel que cette contrainte d'instantanéité est vérifiée.

6.2 Points de contrôle et retour arrière

Participants : Jean-Michel Hélary, Achour Mostéfaoui, Michel Raynal.

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

Résumé : *Nous avons obtenu un résultat montrant l'impossibilité d'assurer la propriété RDT avec un protocole à coordination induite par les communications basé sur les horloges scalaires. Nous avons aussi affiné les résultats essentiels obtenus les années précédentes, notamment à travers le concept de cohérence d'intervalle (Interval Consistency).*

Un résultat d'impossibilité Les protocoles à coordination induite par les communications constituent une solution efficace pour la détermination en ligne d' ensembles de points de contrôles ayant de bonnes propriétés telles que, par exemple, l'absence d'effet domino. Ils n'ajoutent pas de messages de contrôle au calcul sous-jacent, mais utilisent les messages de l'application pour véhiculer d'éventuelles informations de contrôle. Parmi ces protocoles, ceux basés sur une horloge scalaire sont particulièrement intéressants, car l'information de contrôle y est réduite à un entier. Par ailleurs, l'une des propriétés intéressantes des ensembles de points de contrôle est appelée RDT (Rollback Dependency Trackability) : elle assure que toutes les dépendances entre points de contrôle locaux sont observables pendant l'exécution du calcul. Il serait donc intéressant de concevoir des protocoles à coordination induite par les communications, basés sur les horloges scalaires, et assurant la propriété RDT. De tels protocoles n'ont encore jamais été proposés, et pour cause : nous avons prouvé que de tels protocoles sont impossibles [11].

La notion de cohérence d'intervalle (Interval Consistency) Dans le contexte d'un processus séquentiel, un intervalle est un séquence d'évènements consécutifs. L'ensemble des intervalles définis sur un calcul réparti constitue une abstraction de ce calcul, et la relation classique de causalité sur les évènements induit une relation sur l'ensemble des intervalles, appelée *I-précédence*. On s'intéresse alors à la question suivante : "un ensemble d'intervalles donné constitue t-il une abstraction cohérente du calcul réparti ?". Pour répondre à cette question, nous avons défini un critère de cohérence, appelé " cohérence d'intervalles" (Interval

Consistency). Intuitivement, la cohérence d'intervalles signifie que la relation de I-précédence associée ne contredit pas la séquentialité de chaque processus. Plus formellement, la cohérence d'intervalles est définie comme une propriété sur un graphe de précédence associé à l'ensemble d'intervalles considéré sur le calcul réparti. De manière opérationnelle, la cohérence d'intervalles est caractérisée par une propriété sur des estampilles, à valeurs dans un treillis. Nous avons utilisé cette caractérisation opérationnelle pour concevoir un protocole générique qui, étant donné des intervalles définis au fur et à mesure de l'exécution par un " adversaire" imprévisible, produit une abstraction cohérente (au sens de la cohérence d'intervalles). Ce protocole peut décider de terminer prématurément certains intervalles (et donc de commencer de nouveaux intervalles), mais ne peut jamais refuser la décision de l'adversaire de démarrer un nouvel intervalle. Il produit donc une partition plus fine que celle résultant des actions (imprévisibles) de l'adversaire.

6.3 Systèmes transactionnels dupliqués fondés sur la communication de groupe

Participants : Udo Fritzke, Philippe Ingels, Achour Mostéfaoui, Michel Raynal.

Mots clés : transaction, duplication, diffusion, tolérance aux défaillances.

Résumé : *Nous avons étudié le problème de la diffusion fiable atomique dans un groupe et dans plusieurs groupes. Nous avons ensuite implémenté une primitive de diffusion atomique dans plusieurs groupes où des processus peuvent être défaillants. D'autre part, nous avons étudié l'utilisation d'une telle primitive dans le contrôle de duplication dans les systèmes transactionnels.*

Nous avons étudié la diffusion fiable atomique dans les systèmes sujets aux défaillances. Il a été prouvé que ce problème est équivalent au problème de consensus. Nous avons donc cherché les meilleures implémentations en utilisant le consensus comme brique de base dans le cas où les processus peuvent repartir après défaillance [20].

Nous avons proposé un protocole mettant en œuvre une primitive de *diffusion fiable totalement ordonnée* (ou diffusion atomique) vers plusieurs groupes qui serait moins coûteuse et plus résistante au facteur d'échelle [16]. Ce problème est plus dur à résoudre que le précédent. 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 du coût). La seconde propriété est la *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. Par ailleurs, des analyses sur les performances du protocole de consensus et de celui de la diffusion atomique ont été menées sur la maquette réalisée.

Nous avons également étudié l'utilisation des primitives de diffusion fiable uniforme et de diffusion atomique dans le support aux protocoles de contrôle de duplication de données [30]. Contrairement aux modèles de duplication complète trouvés dans la littérature, où une base de données est entièrement dupliquée sur un groupe de processus, nous nous sommes basés sur le

modèle de duplication partielle dans lequel chaque objet est dupliqué de manière indépendante sur un certain groupe de processus. Le protocole décrit assure la "sérialisabilité à une copie" des transactions, et cela grâce aux propriétés de la primitive de diffusion atomique et au verrouillage à deux phases. Par ailleurs, on garantit l'absence d'interblocage de transactions en permettant au protocole d'annuler certaines transactions. On tire profit de deux politiques d'annulation pour aboutir à deux variantes de ce protocole de base, l'une privilégiant les opérations de lecture des transactions et l'autre privilégiant les opérations d'écriture. Le premier protocole procure des exécutions dans lesquelles les transactions de lecture seule ne sont jamais annulées. Le deuxième procure des exécutions dans lesquelles les transactions d'écriture seule ne sont jamais annulées. Ces deux algorithmes garantissent que toutes les transactions ne sont pas annulées.

La notion de Z-cycle permet de définir une condition nécessaire et suffisante pour qu'un ensemble de points de reprise locaux définis par des processus puissent appartenir à point de reprise global cohérent. Dans [14] nous avons étendu cette notion aux systèmes transactionnels et avons défini plusieurs protocoles de défintion de points de reprise cohérents pour de tels sytèmes.

6.4 Les problèmes d'accord dans les systèmes répartis

Participants : Emmanuelle Anceaume, Udo Fritzke, Fabiola Greve, Jean-Michel Hélary, Michel Hurfin, Philippe 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 au cours de l'année 2001 sur les problèmes d'accord ont porté sur trois aspects. Le premier concerne la solvabilité du consensus (nouvelles familles de détecteurs de défaillances, introduction des probabilités et utilisation de conditions sur l'ensemble des valeurs proposées) ainsi que la notion de problèmes d'accord "difficiles" tels que l' accord global, pour lequel nous avons obtenu une solution basée sur le paradigme des protocoles à rondes, optimale en terme du nombre de rondes. Le second aspect concerne l'utilisation du problème du consensus comme brique de base pour résoudre des problèmes d'accord. Enfin, le troisième aspect concerne l'élargissement du champ d'application de nos résultats en considérant les fautes byzantines et les contraintes temporelles. La prise en compte de contraintes temporelles nous a aussi conduit à étudier la réalisation d'un serveur, pouvant faire face à de nombreuses requêtes ne pouvant subsister qu'un temps limité dans la mémoire du serveur. Cette étude nous a conduit à un protocole de réalisation du serveur et à un résultat d'impossibilité, selon les contraintes imposées. Les différents résultats obtenus ont, en partie, été intégrés à la plate-forme* EDEN *que nous développons.*

Si aucune hypothèse supplémentaire n'est adoptée, il est prouvé que le problème d'accord élémentaire n'admet pas de solution déterministe dans un système réparti asynchrone dès

lors qu'un processus au moins est susceptible de connaître une défaillance de type panne franche. Notre axe de recherche principal est donc naturellement consacré à l'étude des solutions permettant de contourner ce résultat d'impossibilité principalement les solutions déterministes fondées sur les détecteurs de défaillances.

- **Solvabilité du consensus**
  Afin de contourner le résultat d'impossibilité nous avons suivi trois directions. Le première direction, initiée par Chandra et Toueg, consiste à enrichir le système sous-jacent par des détecteurs de défaillances. [38] constitue une introduction au concept de détecteurs de défaillances introduit par S. Toueg et son équipe. La construction d'un protocole qui ne requiert qu'un nombre fini de messages pour résoudre le problème de la diffusion fiable uniforme sert de fil conducteur et d'illustration à cette introduction.
  Une autre façon de contourner le résultat d'impossibilité consiste à affaiblir le problème (la terminaison est seulement probabiliste). Ben-Or a été le premier à trouver une solution probabiliste au problème du consensus binaire. Nous avons proposé un protocole de consensus multi-valué purement probabiliste [28].
  Nous avons introduit une nouvelle approche dans la résolution du problème du consensus dans les systèmes répartis asynchrones [33, 34, 35]. Un vecteur d'entrée au problème du consensus est le vecteur constitué des n valeurs proposées par les n processus. Si V est l'ensemble des valeurs qui peuvent être proposées alors il existe $|V|^n$ vecteurs d'entrée possibles au consensus. Une manière de circonvenir le résultat d'impossibilité consiste à n'autoriser qu'un sous-ensemble S de ces vecteurs. Le premier résultat que nous proposons est un protocole de consensus générique, paramétré par S, pour le modèle à mémoire partagée. Ce protocole garantit la propriété d'accord du consensus quel que soit le vecteur d'entrée. La propriété de terminaison est satisfaite soit en l'absence de défaillances soit lorsque le vecteur d'entrée appartient au sous-ensemble S. Nous avons donné une condition nécessaire et suffisante que doit vérifier un sous-emsemble de vecteurs d'entrée pour résoudre le problème du consensus. Ces travaux effectués pour le modèle à mémoire partagée ont été étendus au modèle de communication par messages.
  Les pannes franches ne constituent qu'un cas particulier de comportement défaillant. Aussi, concevoir des protocoles répartis capables de résister à un comportement erroné arbitraire des processus constitue un réel défi, compte-tenu de la possibilité d'attaques malveillantes ou d'erreurs logicielles imprévues. Par conséquent, être capable de construire un protocole masquant les défaillances arbitraires au-dessus de protocoles tolérant les pannes franches constituerait une avancée majeure dans le domaine de l'ingénierie logicielle. L'approche adoptée, modulaire, est basée sur l'encapsulation de la détection de défaillances de tout type dans des modules spécifiques. Elle peut constituer un point de départ pour la conception d'outils de transformation automatique. Cette méthodologie a été appliquée au problème du consensus [13].
- **Solutions au problème du consensus**
  Nous nous sommes principalement intéressés aux solutions déterministes s'appuyant sur des détecteurs de défaillances appartenant aux classes $\mathcal{S}$ et $\diamond\mathcal{S}$. La classe des détecteurs de défaillances forts (dénotée $\mathcal{S}$) regroupe tous les détecteurs de défaillances qui ont pour propriétés de suspecter les processus défaillants (propriété de complétude) mais de ne pas suspecter au moins un des processus non défaillants (propriété d'exactitude). La

classe des détecteurs de défaillances faibles (dénotée $\diamond\mathcal{S}$) regroupe quant à elle tous les détecteurs de défaillances qui vérifient la même propriété de complétude mais ne satisfont la propriété d'exactitude qu'après un laps de temps indéterminé. De fait, les détecteurs de défaillances appartenant à ces deux classes sont intrinsèquement non fiables puisqu'ils peuvent suspecter arbitrairement des processus corrects. Bien que faibles, les propriétés d'exactitude caractérisant ces deux classes sont suffisantes pour permettre de résoudre le problème du consensus si au moins un processus est correct (classe $\mathcal{S}$) ou si au moins une majorité de processus est correcte (classe $\diamond\mathcal{S}$).
Nous proposons dans [17, 32], un protocole générique qui peut s'adapter aisément aux deux classes de détecteurs de défaillances $\diamond\mathcal{S}$ et $\mathcal{S}$. Différentes instantiations du protocole général permettent de retrouver des protocoles existants (par exemple, celui de Chandra et Toueg) ainsi que de nouveaux protocoles. Cette approche, également intéressante par son aspect méthodologique, permet de comprendre les fondements conceptuels de cette famille de protocoles. Ceci est rendu possible par la caractérisation de deux ensembles de processus qui, pour un pas d'itération donné, assurent respectivement les propriétés de vivacité et de sûreté. Cette approche permet également de montrer que, contrairement a une idée admise, les estampilles utilisées dans le protocole de Chandra et Toueg peuvent être bornées (Utilisation d'une fenêtre coulissante). Par ailleurs, ce protocole permet, lorsque cela est nécessaire, de réduire le nombre d'étapes de calcul au prix bien évidemment d'un accroissement du nombre de messages échangés par étape. Le compromis qui résulte du choix d'un schéma d'échange de messages peut être modifié à chaque pas d'itération. Ainsi le nombre de messages échangés par pas de communication peut varier de $O(n)$ (schéma centralisé) à $O(n^2)$ (schéma totalement décentralisé).
Un des défauts des protocoles fondés sur la classe $\diamond\mathcal{S}$ réside dans le fait qu'à une étape donnée seule la valeur du coordonnateur peut être décidée. Si nous considérons le cas favorable où tous les processus proposent la même valeur, il est malgré tout possible de ne pas décider avant longtemps si la propriété d'exactitude met du temps à se vérifier. Nous avons donc défini un protocole [26] qui permet aux processus de décider en une étape de communication dans des circonstances favorables. Celles-ci apparaissent lorsque $(n - f)$ processus proposent la même valeur (où f est le nombre maximal défaillances). Le protocole requiert la condition $f < n/3$. Il est montré que, dans le cas général, cette condition est nécessaire. Cette condition peut être affaiblie à $f < n/2$ au prix d'une étape de communication supplémentaire.
Enfin, nous présentons une extension du problème du Consensus [8] afin d'en améliorer les performances. Cette extension, appelée Accord sur un Préfixe Uniforme, permet à tous les processus du groupe de proposer au cours d'une même exécution de ce module, un flot de messages au lieu d'un seul comme c'est le cas dans le problème du Consensus. Tirant profit de ce flot de messages, il construit sa valeur de décision en utilisant tout ces messages, et non un seul comme c'est le cas dans le problème du Consensus. Nous utilisons cette extension pour construire un algorithme de Diffusion Atomique Uniforme efficace et peu coûteux.

– **Autres problèmes d'accord**
Le consensus est un problème d'accord élémentaire dont la définition simple est unanimement reconnue par tous. Il s'agit d'un problème purement théorique qui permet

d'abstraire différents problèmes d'accord. La simplicité de ce problème permet, lors de son étude, de se focaliser uniquement sur le résultat d'impossibilité qui est associé à tout problème d'accord. En pratique, ceci ne signifie pas pour autant qu'un service de consensus est le dénominateur commun idéal à partir duquel on peut toujours dériver des solutions efficaces à des problèmes d'accord plus complexes. Diverses variantes au problème du consensus ont donc été explorées. Le but poursuivi est de concevoir une brique de base parfaitement adaptée au problème d'accord que l'on désire résoudre.
Parmi les variantes possibles au problème du consensus, nous avons étudié le problème de l'accord global : l'accord ne doit pas se limiter à une valeur proposée par l'un des processus, mais se faire sur un ensemble de valeurs, incluant les valeurs de tous les processus corrects. Dans un modèle de calcul asynchrone avec pannes franches, ce problème ne peut être résolu que si l'on dispose de détecteurs de défaillances parfaits. L'importance du problème nous avait déjà conduit à rechercher un protocole permettant des décisions au plus tôt. Ce protocole, basé sur un modèle de rondes, atteint l'accord, dans le pire des cas, après $\min(2f+2, t+1)$ rondes (où t est le nombre maximal de pannes pouvant se produire et f le nombre de pannes effectives) (voir le rapport d'activité 2000). Le défi qui restait ouvert était de savoir si ce nombre était minimum, sachant que de toutes façons l'accord ne peut être obtenu en moins de $\min(t+1, f+2)$ rondes (borne inférieure établie depuis plusieurs années dans le cas synchrone). En collaboration avec Carole Delporte-Gallet et Hugues Fauconnier, du LIAFA, nous avons relevé ce défi et proposé un protocole permettant une décision, dans le pire des cas, après $\min(t+1, f+2)$ rondes [41]. Ce protocole est donc optimal en termes du nombre de rondes. De plus, ce résultat montre que l'on peut être aussi efficace dans un modèle asynchrone soumis aux pannes franches et avec détecteur parfait que dans le modèle synchrone, ce qui n'est pas trivial puisque le premier modèle est connu pour être moins puissant que le second.
Le problème du "k-Set Agreement" généralise le problème du consensus en ce sens qu'il ne limite pas l'ensemble des valeurs de décision à une seule mais à k valeurs, il est de ce fait moins dur à résoudre. Dans un système asynchrone composé de n processus et où au plus f d'entre eux peuvent être défaillants, le problème du "k-Set Agreement" peut être facilement résolu si $f < k$. Il a par ailleurs été démontré qu'aucune solution déterministe n'existe lorsque $f \geq k$.
Nous avons montré que les détecteurs de défaillances de Chandra et Toueg ne sont pas minimaux pour résoudre le "k-Set Agreement" nous avons donc étudié une approche fondée sur les détecteurs de défaillances à portée réduite. Dans [37] nous présentons une approche probabiliste pour résoudre ce problème. Le protocole proposé ne nécessite pas la connaissance *a priori* de l'ensemble des valeurs proposables. Dans les deux cas, nous avons mis en évidence un aspect contre-intuitif du problème : plus k augmente moins on tolère de défaillances. Cependant, on gagne sur l'aspect impossibilité puisque dans la solution déterministe on a besoin de détecteurs de défaillances plus faibles. Tout comme dans le cas du consensus, on a présenté dans [36] un protocole fondé sur une restriction sur la composition de l'ensemble des valeurs proposées qui permet de résoudre ce problème quel que soit f. En l'occurrence, nous avons montré que s'il existe parmi toutes les valeurs proposées un sous-ensemble de k valeurs proposées par plus de $(kn+f)/(k+1)$ processus alors le problème du "k-Set Agreement" a une solution.

– **Prise en compte des contraintes temporelles**
Notre objectif est de maîtriser la terminaison des protocoles exécutés en permettant de gérer au mieux une durée maximale d'exécution.
Nous avons étudié une notion de cohérence de données qui fait intervenir le temps physique [23]. Cette notion (appelée "timed consistency") permet de prendre en compte la durée de vie des objects (que celle-ci soit définie statiquement ou dynamiquement). Un protocole qui garantit une telle cohérence est developpé. Ce protocole a été implémenté dans une application de controle de trafic routier.
Nous considérons le problème de l'inversion de priorité au sein d'un groupe de processus [22, 39]. À l'origine l'inversion de priorité a été défini dans le contexte des systèmes non-répliqués. Nous étendons cette notion d'inversion de priorité locale au contexte d'un groupe de processus répliqués. Nous présentons ensuite les propriétés que doit garantir un protocole d'ordonnancement pour assurer un ordre total sur l'exécution de requêtes soumises par des clients tout en évitant l'inversion de priorité de groupe. Ces propriétés sont inplantées dans un système asynchrone temporisé augmenté d'un détecteur de défaillance non fiable de la classe à S. La solution que nous proposons permet de répliquer un serveur critique en garantissant d'une part que l'exécution des requêtes est cohérente et d'autre part qu'elle est prédictible. Par conséquent, le protocole d'ordonnancement que nous décrivons est une brique de base pour le développement d'applications temps réel tolérantes aux fautes évoluant dans un modèle asynchrone temporisé.
Nous avons travaillé à la réalisation d'un serveur fiable soumis à contraintes temporelles. Il s'agit de construire le sous-système serveur dans un système client/serveur. L'implémentation de ce serveur est basée sur l'approche connue de la triple redondance modulaire (TMR). Les trois processus serveurs coopèrent pour traiter les requêtes des clients dans le même ordre, afin de garantir la cohérence de l'état du serveur. Deux serveurs au moins sont fiables tandis que le troisième peut avoir un comportement byzantin. On suppose connues deux bornes sur les délais de communication, l'une entre les clients et les processus serveurs (D), l'autre entre deux processus serveurs (d). Les clients sont sujets aux pannes franches mais peuvent émettre des requêtes invalides. Un serveur correct doit donc vérifier la validité d' une requête avant de la traiter. A cause du grand nombre de clients, une telle vérification peut s'avérer coûteuse en temps, d'autant qu'elle doit être effectuée par tous les serveurs corrects (au moins deux). La conception du TMR préconisée ici vise à alléger cette charge en utilisant une technique d'ordonnancement par correspondance. Mais afin de rester compatible avec les contraintes de mémoire des serveurs, cette technique impose une borne de temps Σ : aucune requête de client ne peut rester plus de Σ unité de temps dans la mémoire locale d'un processus serveur. Le problème abordé dans ce travail est nouveau. Il s'agit de concevoir un protocole qui garantit le traitement de toutes les requêtes par les processus serveurs corrects dans un ordre identique, en dépit de l'effet combiné de la borne ΣS, du comportement byzantin possible d'un des processus serveurs, et des pannes franches possibles des clients. Deux résultats ont été obtenus. Le premier est un protocole d'ordonnancement qui suppose $\Sigma > D + 3d$. Le deuxième est un résultat d'impossibilité, qui stipule qu'aucun protocole d'ordonnancement n'est possible lorsque $\Sigma < D$ [27, 42]. Des avancées récentes nous permettent d'espérer un affinement de ces bornes à la valeur commune $D + 2d$.

– **Gestion de la composition d'un groupe**
Le concept de groupe est de plus en plus utilisé pour concevoir des couches logicielles permettant de supporter l'exécution d'applications réparties fiables. Ce paradigme recouvre deux services de base, à savoir, un service de gestion de la composition du groupe et un service de communication de groupe. Un groupe est un ensemble de processus coopérant à la réalisation d'une tâche commune. Étant donné que de nouveaux processus peuvent souhaiter rejoindre le groupe, que des membres du groupe peuvent souhaiter le quitter ou connaître des défaillances de type panne franche, la composition du groupe doit pouvoir évoluer de façon dynamique. L'ensemble des processus qui constituent le groupe à un instant donné est appelé la vue courante du groupe.
Dans [31], nous nous intéressons aux problèmes de la spécification et du développement d'un service de gestion de la composition du groupe ne tolérant qu'une partition primaire (une seule vue à un instant donné). Nous proposons tout d'abord une spécification du problème. Ensuite, nous présentons un protocole qui satisfait cette spécification dans un système réparti asynchrone équipé de détecteurs de défaillances. Le service de gestion de la composition du groupe via une partition primaire est obtenu en spécialisant de manière appropriée un protocole d'accord générique. Pour cela, nous avons défini un cadre général dans lequel la définition d'un problème d'accord particulier est fixée par le biais de 6 paramètres. Le protocole proposé par Chandra et Toueg pour résoudre le problème du consensus à l'aide de détecteurs de défaillance appartenant à la classe $\diamond\mathcal{S}$ constitue l'ossature du protocole générique proposé. Les extensions offertes par ce nouveau service d'accord permettent notamment aux membres d'un groupe de processus de décider unanimement sur une collection de valeurs proposées (plutôt que sur une seule des valeurs proposées) tout en autorisant les processus à effectuer de multiples propositions (plutôt qu'une seule au démarrage du protocole d'accord).

6.5 Aide à la conception des applications multimedia

Participants : Emmanuelle Anceaume, Erwan Demairy.

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

Résumé : *Les travaux réalisés ont tout d'abord porté sur l'aspect "statique" d'une application multimedia puis sur son aspect "dynamique". Plus précisément, nous avons conçu un algorithme permettant de vérifier statiquement une application vis-à-vis de la cohérence des données échangées et du respect des contraintes temporelles. Nous nous sommes ensuite intéressés à la répartition dynamique des ressources entre les applications qui s'exécutent sur une architecture qualité de service donnée.*

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

Nos travaux s'inscrivent dans le cadre de l'aide à la conception des applications multimedia. Nous nous intéressons plus particulièrement aux applications multimedia capables d'adaptation

entre différents niveaux de qualité. Afin de faciliter le travail des concepteurs d'applications multimedia, nous définissons tout d'abord un algorithme qui permet de vérifier statiquement une application vis-à-vis de la cohérence des données échangées et du respect des contraintes temporelles. L'application est décrite par son concepteur sous la forme de composants et de connecteurs, habituelle dans le domaine des architectures logicielles. On associe à ces éléments les configurations qu'ils supportent, c'est-à-dire la qualité qu'ils peuvent fournir ou accepter. Chacune de ces configurations impose une exigence en ressources pour le système sous-jacent. Ces exigences sont obtenues en mesurant la consommation des applications, par le biais de moniteurs. À ce stade, nous pouvons vérifier que l'association des composants et connecteurs constitue bien un ensemble cohérent vis-à-vis des qualités comprises et fournies par les éléments constituant l'application. D'autre part, des contraintes temporelles probabilistes sont placées sur les flux de l'application. Nous définissons une méthode qui permet de déterminer si les contraintes peuvent être respectées, en utilisant les garanties données par les éléments constituant l'application.

Ce premier algorithme fonctionne en considérant que les ressources nécessaires pour garantir le bon fonctionnement des éléments de l'application sont disponibles. Cette hypothèse n'étant en général pas satisfaite, nous nous intéressons dans un deuxième temps, à la répartition des ressources entre applications. Nous supposons que ces applications s'exécutent sur une architecture qualité de service donnée. Cet algorithme est utilisé par l'environnement pour piloter une AQDS au moment de l'admission ou du retrait d'une application en permettant une redistribution des ressources entre les différentes applications. L'environnement agit comme une sur-couche d'une architecture de qualité de service et permet d'obtenir une qualité "optimale" ; le critère d'optimalité étant précisé ultérieurement.

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

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

applications est construit par extension de l'AQdS QuO.

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

La thèse couvrant ce travail devrait être soutenue prochainement.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Contrat FT R&D sur les contraintes non-fonctionnelles

Participants : Fabiola Greve, Michel Hurfin, Eric Mourgaya, 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 à* FT R&D.

7.2 Contrat Réutel 2000

Participants : Udo Fritzke, Fabiola Greve, Michel Hurfin, Philippe Ingels, Jean-Pierre Le Narzul, Frédéric Tronel.

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, qui a été signé en 1997 et s'est achevé au début de l'année 2001. 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 les services offerts dans un environnement CORBA en proposant des extensions logicielles qui permettront aux programmeurs de définir des services tolérant les défaillances (duplication active). L'approche proposée consiste à concevoir des services nécessaires à la gestion d'un groupe de serveurs dupliqués (e.g., gestion de la composition du groupe, diffusion atomique) en considérant ceux-ci comme des variantes d'un problème d'accord élémentaire :le consensus. Notre travail consiste donc à développer un prototype en y intégrant l'ensemble des innovations proposées tout en respectant la spécification d'un service CORBA de tolérance aux défaillances qui a été récemment adoptée par l'OMG.*

8 Actions régionales, nationales et internationales

8.1 Actions nationales

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

Participants : Achour Mostéfaoui, Michel Raynal.

Le thème Réseaux et Systèmes de ce GDR est coordonné 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 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 l'action REFLECS de l'INRIA-Rocquencourt depuis sa création en 1991.

8.2.2 Italie

Participants : Emmanuelle Anceaume, Jean-Michel Hélary, Michel Raynal.

Le projet de coopération avec l'Université de Rome La Sapienza sur le thème des points de contrôle et du retour arrière qui avait été retenu pour financement dans le cadre des programmes d'accord CNRS-CNR 1998-1999, reconduit un première fois pour 1999-2000, a été reconduit pour 2000-2001.

Suite aux résultats positifs d'une précédente collaboration avec l'Université "La Sapienzá" à Rome, une nouvelle proposition de programme CNRS-CNR a été déposée en juillet 2001. Le thème de cette proposition concerne les problèmes liés à la tolérance aux fautes arbitraires.

8.3 Actions internationales

8.3.1 Amériques

- **USA (Californie)**
  Participants : Jean-Michel Hélary, Achour Mostéfaoui, Michel Raynal.

  Un projet de coopération avec les professeurs D. Agrawal et A. El Abbadi de l'Université de Santa Barbara, sur les problèmes liés à la mise à jour des "warehouse", a été accepté par la NSF et l'INRIA pour un financement conjoint de 3 ans à partir de juillet 2001.

- **Mexique**

Participants : Emmanuelle Anceaume, Achour Mostéfaoui, Michel Raynal.

Un projet de coopération avec S. Rajsbaum et M. Romero de l'UNAM, Université national de Mexico, sur le thème des problèmes d'accord dans les systèmes asynchrones, a été retenu pour financement dans le cadre des programmes d'accords CNRS-CONACYT 2000-2001. En 2001, cette coopération s'est concrétisée par trois publications [33, 34, 35].

- **Brésil (Université de Bahia)**
  Participants : Fabiola Greve, Michel Hurfin, Michel Raynal, Frédéric Tronel.

  Un projet de coopération avec R. Macedo de l'Université Fédérale de Bahia (Salvador) à été retenu dans le cadre du programme de coopération INRIA/CNPq. L'objectif de ce projet d'une durée de deux ans (2000/2001) est de concevoir et de développer un prototype commun offrant des services de communication de groupe. Côté français, ce projet est géré par Michel Hurfin. En 2001, cette coopération s'est concrétisée par une publication [32]. Dans le cadre de cette coopération, Michel Hurfin et Frédéric Tronel se sont rendus en juin 2001 à Bahia où se tenait le 1er workshop ARGO. Raimundo Macedo a séjourné une semaine dans le projet ADP du 13 au 23 mars 2001.

- **Chine**
  Participants : Michel Hurfin, Jean-Pierre Le Narzul, Frédéric Tronel.

  Un projet de coopération intitulé "Fault Tolérant Corba", dont Michel Hurfin est le coordinateur coté Français, a été sélectionné par le LIAMA (Laboratoire franco-chinois d'informatique, d'automatique et de mathématiques appliquées) pour bénéficier d'un financement durant les années 2000-2001. L'objectif est d'intégrer les mécanismes de communication de groupe que nous développons au sein de l'ORB développé par l'équipe chinoise en respectant les spécifications récemment adoptées par l'OMG. En 2001, cette coopération s'est concrétisée par deux publications [22, 39].

9 Diffusion de résultats

9.1 Actions d'enseignement

A l'exception de deux chercheurs à temps plein, tous les membres permanents 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 des enseignements du diplôme d'ingénieur en informatique de l'université de Rennes 1 (DIIC) et A. Mostefaoui est responsable de la filière Langages et Systèmes Informatiques du DIIC.

Le projet a assuré plusieurs enseignements liés à ses thèmes de recherche :

- dans le DEA d'informatique de Rennes, M. Raynal dispense un cours sur les algorithmes et systèmes répartis (21 heures) ;
- dans le DIIC 3^e année (filières LSI et ARC), Jean-Michel Hélary et Michel Raynal dispensent un cours sur l'algorithmique répartie (30 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 (respectivement de 3 et 6 heures chacun) ;
- dans le DESS ISA, M. Raynal donne un cours sur les systèmes transactionnels, la cohérence et la sécurité des données.
- Achour Mostefaoui 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.
- Michel Hurfin a dispensé un cours sur la tolérance aux défaillances à l'Université Fédérale de Bahia (6 heures).

9.2 Présentation de travaux

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

Michel Raynal a présenté les travaux de l'équipe et fait des séminaires dans les universités et centres de recherche suivants :
- Université La Sapienza de Rome (Rome, Italie), janvier 2001.
- Université d'Austin (Texas) (USA), avril 2001,
- Universite de Boston, mars 2001,

Jean Michel Hélary a fait un exposé sur le problème d'accord global lors de la journée "Algorithmique Répartie (ACM-SIGOPS France)" organisée à l'IRISA le 18 mai 2001.

9.3 Animation de la communauté scientifique

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

Michel Raynal a été (en 2001) et sera (en 2002) membre des comités de programme des conférences suivantes :
- *Fith International Symposium on Autonomous Decentralized Systems.*
  Dallas, Texas, mars 2001.
- Publicity Chair *21th IEEE Int. Conf. on Distributed Computing Systems.*
  Phoenix, Arizona, avril 2001, (Actes publiés par IEEE)
- *20th ACM Symposium on Principles of Distributed Computing (PODC'01).*
  Newport, Road Island, août 2001, (Actes publiés par ACM).
- co-président et co-organisateur (avec Luis Rodrigues, Lisboa, PT) du *Int. IEEE Workshop on Applied Reliable Group Communication* (in conjunction with the 21th IEEE Int. conf. on Distributed Computing Systems.
  Phoenix, Arizona, avril 2001. (Actes publiés par IEEE).
- *2d Int. Conf. on Dependable Systems and Networks (IEEE Symposium on Fault-Tolerant Systems FTCS'31 and DCCA'9).*
  Goteborg, Juillet 2001. (Actes publiés par IEEE).
- *6th International Parallel Computing Technologies (PaCT'01).* Novosibirsk, Russia, September 2001. (Actes publiés par Springer-Verlag, LNCS).

- *20th IEEE Symposium on Reliable Distributed Systems (SRDS'01).* Actes publiés par IEEE).
- *IEEE Int. Symposium on Network Computing and Applications.* Cambridge (MA), October 2001. (Actes publiés par IEEE).
- Program Chair, *22th IEEE Int. conf. on Distributed Computing Systems.* Vienne, Autriche, juillet 2002. (Actes publiés par IEEE).
- Program co-Chair, *5th IEEE Int. Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'02).* Washington DC, April 2002. (Actes publiés par IEEE).

Michel Hurfin est membre du comité de programme du *21st International Conference on Distributed Computing Systems* (ICDCS-2001) qui s'est déroulé à Phoenix en Arizona en avril 2001.

Michel Hurfin est membre du comité de programme du workshop *Aspect Oriented Programming for Distributed Computing Systems - Distributed Auto Adaptive Systems* (AOPDC-DAAS) qui se déroulera à Vienne en juillet 2002.

Emmanuelle Anceaume est membre du comité de programme du *22th IEEE Int. conf. on Distributed Computing Systems* (ICDCS-2002) qui se déroulera à Vienne en juillet 2002.

Achour Mostéfaoui a été élu au Conseil d'Administration de l'Association SPECIF des enseignants et des chercheurs en Informatique.

Emmanuelle Anceaume et Michel Hurfin ont mis en place, ont organisé et animent le Séminaire Réseaux et Systèmes (`http://www.irisa.fr/adp/srs/srs.html`).

Emmanuelle Anceaume et Michel Hurfin ont organisé la journée " Algorithmique Répartie (ACM-SIGOPS France)" à l'IRISA le 18 mai 2001.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] O. BABAOGLU, 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

[8] E. Anceaume, « Efficient Solution to Uniform Atomic Broadcast », *International Journal of Foundations of Computer Science (IJFCS 01)*, 2001, to appear.

[9] R. Baldoni, G. Cioffi, J. Helary, M. Raynal, « Direct Dependency-Based Determination of Consistent Global Checkpoints », *International Journal of Computer and Systems Science and Engineering 16*, 1, Sep 2001, p. 43–50.

[10] R. Baldoni, J. Helary, G. Melideo, M. Raynal, « Efficient Causality-Tracking Timestamping », *IEEE Transactions on Knowledge and Data Engineering*, 2001, to appear.

[11] R. Baldoni, J. Helary, A. Mostefaoui, M. Raynal, « Impossibility of scalar clock-based communication-induced checkpointing protocols », *Information Processing Letters 80*, 2, Oct 2001, p. 105–113.

[12] R. Baldoni, J. Helary, M. Raynal, « Rollback-dependency trackability : a minimal characterization and its protocol », *Information and Computation 16*, 2, Mar 2001, p. 144–173.

[13] R. Baldoni, J. Hélary, M. Raynal, L. Tanguy, « Consensus in Byzantine Asynchronous Systems », *Journal of Discrete Algorithms, Special Issue on selected papers from Sirocco'2000*, 2001.

[14] R. Baldoni, F. Quaglia, M. Raynal, « Consistent Checkpointing for Transaction Systems », *The Computer Journal 44*, 2, 2001, p. 92–100.

[15] R. Baldoni, M. Raynal, « Fundamentals of Distributed Computing : a Practical Tour of Vector-Clock Systems », *IEEE Distributed Systems Online*, 2001, to appear.

[16] U. Fritzke, P. Ingels, A. Mostefaoui, M. Raynal, « Fault-tolerant total order multicast to asynchronous groups », *IEEE Transactions on Parallel and Distributed Systems 12*, 2, Feb 2001, p. 147–156.

[17] M. Hurfin, A. Mostefaoui, M. Raynal, « A Versatile Family of Consensus Protocols Based on Chandra-Toueg's Unreliable Failure Detectors », *IEEE Transactions on Computers*, 2002, to appear.

[18] A. Mostefaoui, M. Raynal, P. Verissimo, « The Logically Instantaneous Communication Mode : a Communication Abstraction », *Future Generation Computer Systems 17*, 6, Mar 2001, p. 669–678.

[19] A. Mostefaoui, M. Raynal, « Leader-Based Consensus », *Parallel Processing Letters 11*, 1, 2001, p. 95–107.

[20] M. Raynal, L. Rodrigues, « Atomic Broadcast in Asynchronous Crash-Recovery Distributed Systems and its Use in Quorum-Based Replication », *IEEE Transactions on Knowledge and Data Engineering*, 2001, to appear.

[21] M. Raynal, M. Singhal, « Mastering Agreement Problems in Distributed Systems », *IEEE Software 18*, 4, Jul 2001, p. 40–47.

[22] Y. Wang, E. Anceaume, F. Brasileiro, F. Greve, M. Hurfin, « Solving the Group Priority Inversion Problem in a Timed Asynchronus System », *Transaction on Computers, Special Issue on Asynchronous Real-Time Distributed Systems*, 2001, to appear.

Communications à des congrès, colloques, etc.

[23] M. Ahamad, D. Bakken, V. Krishnaswamy, M. Raynal, « Shared State Consistency for Time-Sensitive Distributed Applications », *in : Proc. of the 21st IEEE Int. Conf. on Distributed Computing Systems (ICDCS-01)*, IEEE, p. 606–614, Phoenix, AZ, Apr 2001. Best paper award.

[24] M. Ahamad, M. Raynal, F. Torres, « Realtime Based Strong Consistency for Distributed Objects », *in : Proc. of the 5th Int. IEEE Workshop on Object-Oriented Real-time Distributed Systems (WORDS-01)*, IEEE, Roma, Jan 2001.

[25] F. Brasileiro, F. Greve, M. Hurfin, J. L. Narzul, F. Tronel, « Eva : an Event-Based Framework for Developing Specialised Communication Protocols », *in : Proc. of the 1st Int. Symp. on Network Computing and Applications (ISNCA-01)*, IEEE, Cambridge, MA, Oct 2001.

[26] F. Brasileiro, F. Greve, A. Mostefaoui, M. Raynal, « Consensus in One Communication Step », *in : Proc. of the 6th Int. Conference on Parallel Computing Technologies (PACT'01)*, *LNCS*, 2127, Springer Verlag, p. 42–50, Novosibirsk, Sep 2001.

[27] P. Ezhilchelvan, J. Hélary, M. Raynal, « Building TMR-Based Reliable Servers Despite Bounded Input Lifetimes », *in : Proc. of Euro-Par 2001*, LNCS2150 (éditeur), p. 482–485, Aug 2001. Research Note.

[28] P. Ezhilchelvan, A. Mostefaoui, M. Raynal, « Randomized Multivalued Consensus », *in : Proc. of the 4th IEEE Int. Sym. on Object-Oriented Real-time distributed Computing (ISORC'01)*, IEEE, p. 195–200, Magdeburg, May 2001.

[29] C. Fetzer, M. Raynal, F. Tronel, « An Adaptive Failure Detection Protocol », *in : Proc. of the IEEE Pacific Rim Int. Symposium on Dependable Computing (PRDC'01)*, IEEE, Seoul, Dec 2001. to appear.

[30] U. Fritzke, P. Ingels, « Transactions on Partially Replicated Data based on Reliable and Atomic Multicasts », *in : Proc. of the 21st IEEE Int. Conf. on Distributed Computing Systems (ICDCS-01)*, IEEE, p. 284–291, Phoenix, AZ, Apr 2001.

[31] F. Greve, M. Hurfin, M. Raynal, F. Tronel, « Primary Component Asynchronous Group Membership as an Instance of a Generic Agreement Framework », *in : Proc. of the 5th Int. Symp. on Autonomous Decentralized Systems (ISADS-01)*, IEEE, p. 93–100, Dallas, TX, Mar 2001.

[32] M. Hurfin, R. Macedo, A. Mostefaoui, M. Raynal, « A Consensus Protocol Based on a Weak Failure Detector and a Sliding Round Window », *in : Proc of the 20th IEEE Symposium on Reliable Distributed Systems (SRDS'01)*, IEEE, p. 120–129, New-Orleans, Oct 2001.

[33] A. Mostefaoui, S. Rajsbaum, M. Raynal, M. Roy, « Efficient Condition-Based Consensus », *in : Proc. of the 8th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO'01)*, p. 275–291, Val de Nuria, Spain, Jun 2001.

[34] A. Mostefaoui, S. Rajsbaum, M. Raynal, M. Roy, « A Hierarchy of Conditions for Consensus Solvability », *in : Proc. of the 20th ACM SIGACT-SIGOPS Int. Symposium on Principles of Distributed Computing (PODC-2001)*, ACM, Newport, RI, Aug 2001.

[35] A. Mostefaoui, S. Rajsbaum, M. Raynal, « Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems », *in : Proc. of the 33rd ACM Symposium on Theory of Computing (STOC'01)*, ACM, p. 153–162, Crete, Jul 2001.

[36] A. Mostefaoui, M. Raynal, « A Condition for k-Set Agreement in Asynchronous Distributed Systems », *in : Proc of the Int. Parallel and Distributed Processing Symposium (IPPS/SPDP)*, IEEE, San Francisco, CA, Apr 2001.

[37] A. Mostefaoui, M. Raynal, « Randomized k-Set Agreement », *in : Proc. of the 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA'2001)*, ACM, p. 291–297, Crete, Jul 2001.

[38] M. Raynal, « Quiescent Uniform Reliable Broadcast as an Introduction to Failure Detector Oracles », *in : Proc. of the 6th Int. Conference on Parallel Computing Technologies (PACT'01)*, *LNCS*, 2127, Springer Verlag, p. 98–111, Novosibirsk, Sep 2001.

[39] Y. Wang, F. Brasileiro, E. Anceaume, F. Greve, M. Hurfin, « Avoiding Priority Inversion on the Processing of Requests by Active Replicated Servers », *in : Proc. of the International Conference on Dependable Systems and Networks (DSN 2001)*, IEEE, p. 97–106, Göteborg, Jul 2001.

Rapports de recherche et publications internes

[40] E. Anceaume, J. Helary, M. Raynal, « Tracking Immediate Predecessors in Distributed Computations », *Research Report n°1344*, IRISA, Jul 2000, `http://www.irisa.fr/bibli/publi/pi/2000/1344/1344.html`.

[41] C. Delpote-Gallet, H. Fauconnier, J. Hélary, M. Raynal, « Early Stopping in Global Data Computation », *rapport de recherche*, Rapport de recherche IRISA, 2001, `http://www.irisa.fr/bibli/publi/pi/2001/1383/1383.html`.

[42] P. Ezhilchelvan, J. Hélary, M. Raynal, « Building TMR-Based Reliable Servers with Arbitrary Large Number of Clients », *rapport de recherche n°PI#1398*, Rapport de recherche IRISA, 2001, `http://www.irisa.fr/bibli/publi/pi/2001/1398/1398.html`.