Proceedings of the CFIP'96, Rabat, pp17-31

Evaluation de Performance des Réseaux ATM: Étude et Perspectives en Utilisant la Simulation Distribuée

C.D. Pham et S. Fdida

Laboratoire MASI
Université Pierre et Marie Curie
4 place Jussieu 75252 Paris Cedex 05
E-mail : {pham,fdida}@masi.ibp.fr

Résumé. *L'évaluation des futurs réseaux haut-débit par simulation est longue et gourmande en puissance de calcul. Dans ce travail, nous utilisons des techniques de simulation distribuée dans le but d'une part, de réduire le temps de simulation et d'autre part, d'augmenter la taille du système simulé. Dans un premier temps, une méthode conservatrice est utilisée et permet d'obtenir des gains en vitesse satisfaisant. Ensuite, une méthode optimiste utilisant des estimations locales et une stratégie de blocage nous permet d'aller plus vite tout en gardant un faible pourcentage d'erreur.*
Mots-clés: Evaluation de performance, Simulation distribuée, Réseaux ATM.

Abstract. *Performance evaluation of future high-speed networks by simulation is a long and resource consuming process. In this work distributed simulation techniques are used to reduce the simulation time and to increase the size of the simulated system. In a first attempt, a conservative method is used that gives good speedup values. Then, an optimistic approach that uses local estimations coupled with a blocking strategy gives a better speedup while keeping the percentage of errors small.*
Keywords: Performance evaluation, Distributed simulation, ATM Networks.

1 Introduction

Les réseaux de télécommunication seront marqués dans les prochaines années par une demande de plus en plus diversifiée, par une compétition accrue et par des évolutions technologiques considérables qui bouleverseront les réseaux d'aujourd'hui. Parmi les techniques qui influenceront fortement une telle évolution au cours de la décennie, on retiendra surtout la technique de transfert asynchrone (ATM). L'ATM a été adoptée comme la technique de transfert des futurs réseaux à large bande, aujourd'hui en cours de normalisation à l'IUT-I et à l'ATM Forum, mais déjà présente de manière opérationnelle tant dans le domaine privé que public. Elle devra supporter les services variés des applications multimédia et coopératives. Une bonne adéquation entre cette technique et le besoin des applications ne sera obtenue qu'au travers d'un dimensionnement approprié des ressources du réseau. Il s'ensuit une activité intense au niveau international pour évaluer soigneusement les composantes du réseau afin d'en maîtriser le fonctionnement. De nouveaux outils sont aussi nécessaires car la complexité des systèmes étudiés augmente ainsi que leur dimension, rendant des études de phénomènes transitoires ou stationnaires délicates voire impossibles dans le contexte actuel.

Dans ce travail, nous nous intéressons à l'évaluation de performance des réseaux ATM par des techniques de simulation distribuée. L'article se compose de 3 parties: la première, constituée des sections 2 et 3, présente les motivations du choix de ces techniques et leurs concepts de base. Le modèle de réseau et l'environnement de l'étude, décrit dans les sections 4 et 5, constituent la deuxième partie. Enfin, la troisième partie présente les résultats et conclut par la section 6.

2 Évaluation de performance: la simulation

Les futurs réseaux ATM seront de grande dimension, impliqueront de multiples éléments souvent complexes et par conséquent poseront un grand nombre de problèmes. Parmi ceux-ci, les problèmes de routage et de contrôle de congestion sont deux exemples de problèmes très complexes à résoudre à grande échelle. Des critères de performances tels que la longueur moyenne des files d'attente, le taux de pertes de cellules ou la probabilité d'appels rejetés sont également très utiles pour bien dimensionner les éléments de commutation. Il est alors nécessaire de disposer d'outils d'évaluation permettant d'étudier tel ou tel mécanisme. La plupart des systèmes informatiques peuvent être considérés comme un ensemble de ressources et de clients concourant pour ces ressources. Des phénomènes d'attente inhérents à ces systèmes font que les méthodes de résolution analytique utilisent naturellement le formalisme des files d'attente. Les réseaux de files d'attente à forme produit ou non donnent la possibilité de modéliser un système donné puis d'en extraire des critères de performances. Ces méthodes sont les moins onéreuses puisque la résolution se fait analytiquement. En contre partie, le champ d'application de ces méthodes est relativement réduit car celles-ci imposent souvent des hypothèses de fonctionnement très restrictives qui nécessitent de fortes simplifications pouvant rendre l'étude non réaliste. La simulation, contrairement aux méthodes analytiques, ne présente pas de limites théoriques quand au niveau de détail du modèle mais cette fantastique souplesse se paye cher. Le principal facteur limitatif est le temps de simulation qui peut très vite devenir excessif lorsque le modèle est un tant soit peu réaliste.

La situation est alors la suivante: d'un côté, la simulation séquentielle est dépassée par l'augmentation de la puissance de calcul requise alors que d'un autre côté, le calcul distribué et les architectures parallèles offrent une formidable puissance. Les techniques de simulation distribuée développées ces dix dernières années pourraient alors s'avérer efficaces si on les applique spécifiquement aux réseaux ATM. La simulation parallèle soulève des problèmes de synchronisation et de respect des contraintes de causalité qui rend cette tâche ardue mais d'un autre côté, il est envisageable de réduire les temps de simulation et ainsi d'augmenter la taille du système étudié. Deux types d'environnement différents sont rencontrés lors de l'évaluation de réseaux de communication. Le premier consiste à s'intéresser aux performances d'un système possédant de nombreuses composantes différentes. Il s'agit par exemple de l'étude d'empilement de couches de protocoles dans des stations d'extrémités. Dans ce cas, la simulation parallèle semble peu performante en raison, d'une part, de la forte synchronisation des processus et d'autre part, de leur faible nombre et de leur variété. Le second type est celui d'un grand réseau de commutateurs, tels que ceux rencontrés auprès des opérateurs publics ou des réseaux privés. Dans ce contexte, les commutateurs sont tous similaires et reproduits en grand nombre. On peut alors envisager une réplication du code associé à chaque commutateur dans un *processus logique* (PL) de la simulation distribuée.

3 La simulation distribuée

Un réseau ATM est essentiellement un *système asynchrone* car les cellules sont envoyées d'un commutateur à l'autre sans aucune relation avec une horloge globale. Ainsi donc, les techniques de simulation *synchrone* ou par *incrément fixe* ne pourront pas représenter fidèlement le comportement temporel du système. Nous nous intéressons donc aux approches *asynchrones* et en particulier ceux dits *discret* où les changements apportés au système se produisent à des points discrets dans le temps et où le modèle va avancer par bonds variables selon la date d'occurence des événements.

Dans cet article, nous supposons l'utilisation d'une architecture parallèle ou distribuée composée de n processeurs pour exécuter la simulation. Le système à simuler est divisé en sous-systèmes disjoints, chacun d'entre eux étant alors simulé par un processus logique s'exécutant sur un processeur physique $p_i, 1 \leq i \leq n$. Les interactions qui peuvent exister dans le système réel sont représentées par l'échange de messages estampillés par une horloge locale du PL source. Les messages sont recus par le PL destinataire et placés dans sa liste locale d'événements (LEV) en attendant d'être exécutés. D'autres méthodes de parallélisation existent mais leur champ d'application est relativement limité. La distribution des horloges et des listes d'événements pose des problèmes de respect de la causalité car les différents PL peuvent évoluer indépendamment et se trouver à des instants de simulation différents. Une *faute temporelle* se produit lorsqu'un message arrive à un PL avec une estampille inférieure à l'horloge locale du PL. Des mécanismes explicites de synchronisation sont donc nécessaires pour garantir l'intégrité de la simulation. On les classe habituellement en deux catégories: *conservatrice* et *optimiste*.

L'approche conservatrice ne permet à un PL p d'exécuter un événement que lorsqu'il peut être déterminé avec certitude que son exécution ne pourra pas produire de fautes temporelles dans le futur [Cha79]. De ce fait, les PLs i qui communiquent avec p utilisent des files d'événements distincts et l'algorithme consiste à sélectionner la file possédant l'événement d'estampille minimale. Si cette file est vide, un blocage se produit. Pour éviter les situations d'inter-blocage, des *null-messages* n'ayant aucune signification du point de vue du modèle sont utilisés pour faire avancer artificiellement le temps de simulation. Dans ce cas, une bonne faculté de "prédire" l'avenir (*lookahead*) est nécessaire pour propager rapidement les valeurs d'horloge et ainsi obtenir de bonnes performances. La deuxième approche se caractérise par son mode de fonctionnement totalement opposé. Ces approches optimistes, dont le plus connu des réprésentants est *Time Warp* [Jef85], ne font aucune tentative pour éviter les fautes temporelles mais un mécanisme de retour-arrière (*rollback*) leur permet de corriger la trajectoire de la simulation lorsqu'une telle faute survient. Les inconvénients majeurs de ces approches optimistes sont d'une part, la nécessité de sauvegarder périodiquement l'état du PL pour pouvoir le restaurer et d'autre part, l'utilisation d'anti-messages pour corriger la propagation des erreurs de trajectoire. De manière générale, les approches optimistes sont beaucoup plus complexes à mettre en œuvre que leurs homologues conservateurs mais peuvent d'un autre côté exploiter de manière transparente le parallélisme présent dans le modèle.

De nombreuses améliorations ont été proposées et une excellente synthèse peut être trouvée dans [Fuj90a]. La diversité des variations reflète très bien la difficulté de la tâche et il n'existe pas un protocole de simulation parallèle mais une multitude de protocoles adaptés pour des applications particulières. La récente focalisation sur les protocoles dits *adaptatifs* est indicatrice d'un besoin urgent de protocoles "multi-usages".

4 Modèle de réseau

Les futurs réseaux large-bande (RNIS-LB) seront basés sur la technologie ATM pour permettre le transfert à très haut-débit d'informations numériques. Ces informations seront véhiculées sous la forme de cellules de taille fixe de 53 octets par des commutateurs ATM dont les deux principales fonctions sont la réception et l'émission de cellules. De fait, deux événements seront considérés dans notre modèle pour la simulation au niveau cellule: un événement $< rec, t_r >$ indicant la réception d'une cellule à l'instant t_r et un événement $< env, t_d >$ indiquant l'envoi d'une cellule à l'instant t_d. Nous allons maintenant préciser les modèles de commutateurs et de réseaux utilisés dans cette étude.

4.1 *Le modèle de commutateur*

De nombreuses architectures ont été proposées dans la littérature pour les commutateurs ATM (cf [Han94] pour une complète description des architectures existantes). Pour notre part, nous adoptons un modèle de commutateur à architecture non-bloquante avec des tampons en sortie. La stratégie de commutation est alors la suivante: à chaque *time slot* une cellule arrivant sur un port d'entrée est commutée sur un port de sortie. Le choix du port de sortie est effectué de manière uniforme. Une seule cellule peut arriver sur un port d'entrée par *time slot* mais plusieurs cellules peuvent être commutées sur le même port de sortie au même instant. Dans ce cas, elles attendent dans les tampons de sortie. Parallèlement, les cellules en tête des tampons de sortie sont transmises sur le lien de sortie à chaque *time slot* en suivant une politique *FIFO*. Au niveau de la simulation, l'arrivée d'une cellule sur un port d'entrée se traduit par la réception d'un événement externe $< rec, t_r >$ qui déclenche la programmation d'un événement interne $< env, t_d >$ avec t_d l'instant de départ de la cellule du port de sortie. Un événement $< rec, t_a >$ sera ensuite envoyé au PL destinataire avec $t_a = t_d + delai\_lien$.

4.2 *Le modèle de réseau*

Le modèle de réseau que nous adoptons est un réseau de N_s commutateurs ATM complètement maillé. Une source de cellules est connectée à chaque commutateur et génère n_c cellules dont le temps d'inter-arrivée est distribué de manière exponentielle. Les liens entre deux commutateurs sont bi-directionnels, de ce fait chaque commutateur possède N_s entrées et $N_s - 1$ sorties. Lorsque les sources ont généré toutes leurs cellules, le nombre de cellulles dans le réseau reste constant et les commutateurs continuent d'envoyer les cellules jusqu'à la fin de la simulation. Le délai de propagation des liens est fixé à 2 *time slot*. La figure 1 représente le modèle de réseau pour $N_s = 5$.

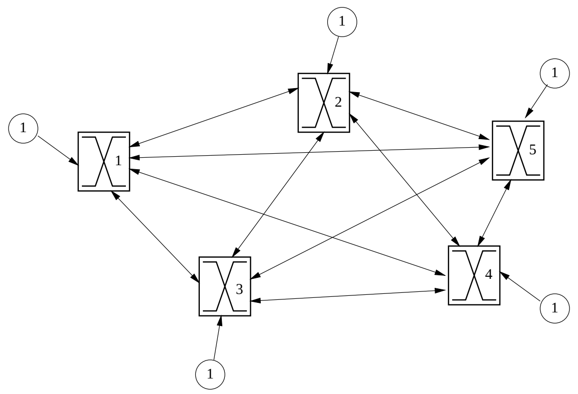

Figure 1. *Réseau complètement maillé pour* $N_s = 5$

Le choix de ce modèle n'est pas arbitraire. En effet, comme nous l'avons dit dans la section 2, nous nous intéressons surtout à l'évaluation de "grand réseaux de commutateurs, tels que ceux rencontrés auprès des opérateurs publics ou des réseaux privés". Nous pensons que la grande connectivité de notre modèle est représentative des réseaux fédérateurs futurs.

5 Environnement d'étude

Les simulations distribuées sont écrites en C en utilisant la librairie PVM [Pvm93] pour les échanges de messages entre les processus. Deux machines sont utilisées: la première consiste en une SparcStation 20 monoprocesseur et la seconde en une Connection-Machine (CM5) de 32 processeurs du *Centre National de Calcul Parallèle en Sciences de la Terre*. Dans ce dernier cas, nous utilisons l'optimisation CM/PVM de PVM pour CM5 et des précautions ont été prises pour s'assurer que la simulation était bien la seule charge utilisatrice de la CM5.

La SS20 est utilisée pour les simulations séquentielles (notée seq^{SS20}) afin de quantifier le gain en vitesse (speedup) obtenu en distribuant la simulation. Dans le reste de l'article, nous noterons par^{CM5} une simulation distribuée effectuée sur la CM5 et seq^{SS20} la version séquentielle sur SS20.

6 Application des techniques de simulation distribuée

6.1 *Approche conservatrice*

Dans un premier temps, une approche conservatrice avec *null-messages* est utilisée comme mécanisme de synchronisation pour l'implantation parallèle. Le code parallèle a ensuite été exécuté sur la SS20 et la figure 2 représente le speedup de la version séquentielle sur la version parallèle tournant sur 1 processeur (notée par^{SS20}). Nous avons fait varier d'une part, la densité de messages (en abscisse) définie comme étant le rapport du nombre total de messages dans le système et du nombre de commutateurs (dans notre cas, on trouve n_c) et d'autre part, la taille N_s du réseau. Les résultats que nous obtenons sont dans une grande part prévisibles: la version séquentielle va presque toujours plus vite que son homologue parallèle tournant sur un seul processeur puisque ce dernier comporte des mécanismes de synchronisation et de passage de messages très lourds. Néanmoins le "presque toujours" est étonnant car on s'attendrait plutôt à un "toujours". On constate en effet que pour $N_s = 20$ et $n_c = 20$, seq^{SS20} et par^{SS20} affichent les mêmes performances. Ensuite pour $n_c = 30$, seul le cas $N_s = 5$ permet à seq^{SS20} d'aller plus vite. En fait ces résultats illustrent clairement les limitations de la version séquentielle lorsque la taille du réseau devient grande. Ces limitations ont essentiellement pour origine l'explosion de la taille de la file d'attente des événements qui entraîne une chute des performances. La distribution de la simulation, bien qu'introduisant des mécanismes lourds, à l'avantage de distribuer les files d'événements, ce qui

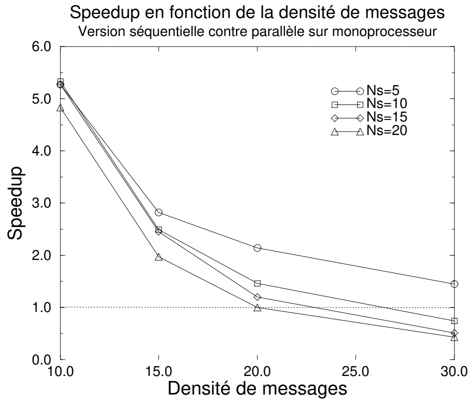

Figure 2. *Speedup entre séquentielle et parallèle monoprocesseur*

donne des files moins grandes. Cependant, il nous a été impossible de simuler plus de 20 commutateurs avec par^{SS20} à cause du nombre élevé de processus UNIX que cela demande.

Dans un deuxième temps, les mesures utilisent seq^{SS20} et par^{CM5}. La figure 4(a) représente le temps d'exécution, exprimé en s, de seq^{SS20} et la figure 4(b) celui de par^{CM5}. Comme précédemment, nous avons fait varier la densité de messages et la taille du réseau. Ces temps d'exécution ont été obtenus en simulant 3.000 cellules à chaque commutateur (le temps d'exécution devient alors linéaire en fonction du nombre de cellules). Au fur et à mesure que la taille du réseau et la densité de messages augmentent, le temps d'exécution de seq^{SS20} augmente très rapidement. D'un autre coté, on constate que la version parallèle est pratiquement insensible aux variations de taille lorsque le nombre de PLs est inférieur au nombre de processeurs physiques disponibles, comme c'est ici le cas. On constate même que par^{CM5} est moins performante lorsque la densité de messages est faible. Cela s'explique par l'augmentation du trafic de *null-messages* nécessaires pour éviter les blocages lorsque le nombre de messages dans le système est petit. La figure 4 représente les valeurs de speedup que l'on obtient. On peut alors constater que la version séquentielle est toujours plus lente que la version parallèle, même pour le cas $N_s = 5$ et $n_c = 10$. Des valeurs de speedup sur-linéaires montrent que les performances de la version séquentielle se dégradent réellement lorsque la densité de messages est grande ($n_c = 30$).

Cette première étude montre donc que l'utilisation d'une approche conservatrice relativement simple nous permet déjà d'une part, d'accélérer la simulation et d'autre part, d'augmenter la taille du système sans dégradation des performances. La question maintenant est de savoir s'il est possible de faire mieux en utilisant une approche optimiste et en prenant en compte les spécificités des réseaux de commutateurs.

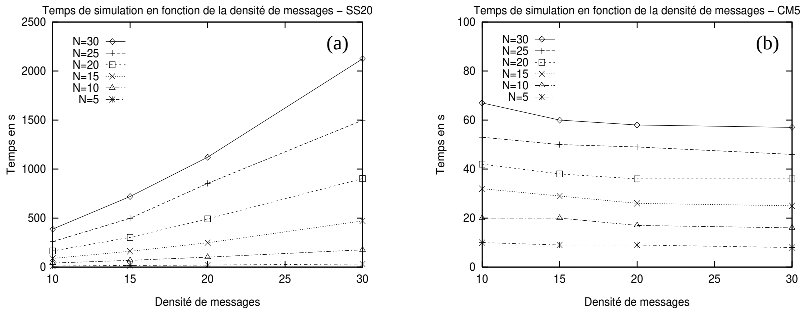

Figure 3. *(a) Séquentielle sur SS20 (b) Parallèle sur CM5*

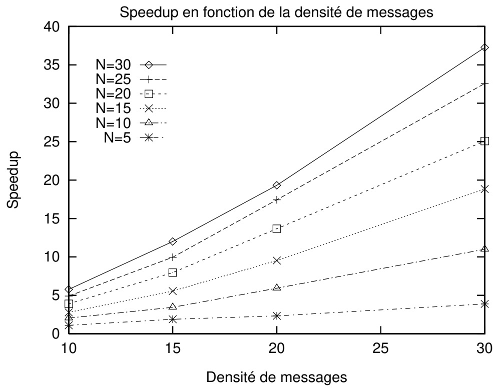

Figure 4. *Speedup entre parallèle multiprocesseur et séquentielle*

6.2 *Approche optimiste pure*

Une approche optimiste pure a le fonctionnement suivant: lorsqu'un message m_{past} d'estampille t_{past} est reçu par un PL dont l'horloge locale a la valeur t_{now}, $t_{now} > t_{past}$, le PL doit effectuer un retour-arrière pour corriger cette erreur de causalité. Cette correction se fait en restaurant l'état du PL, qui est périodiquement sauvegardé, avant l'instant t_{past}, en retournant à l'instant t_{past} et en envoyant les anti-messages pour annuler les messages produits par trop d'optimisme. Ce mécanisme élégant peut cependant diverger lorsque le nombre de fautes temporelles devient grand, créant ainsi un effondrement des performances du fait des retours en arrière incessants (*rollback-trashing*).

La difficulté d'écrire un noyau de protocole optimiste nous a d'abord incité à prédire le comportement d'une telle approche sur notre modèle de réseau. Pour cela, quelques définitions sont nécessaires. Premièrement, la *localité* pour un PL p caractérise la taille du voisinage, c'est-à-dire le nombre de destinations possibles pour un message de p. On peut exprimer ce critère en terme de fraction de nœuds de manière à le rendre relatif au nombre total de PLs. Ainsi exprimés, tous les PLs dans notre modèle de réseau possèdent une localité de $\frac{N_s-1}{N_s}$ qui est presque maximale. Deuxièmement, la *granularité* désigne le temps CPU nécessaire pour exécuter un message. Du fait de la sémantique relativement limitée associée aux messages dans notre modèle de réseau, cette granularité sera le plus généralement faible.

Ces deux nouveaux paramètres, avec la densité de messages déjà définie, ont un impact important sur les performances du simulateur. De manière générale, une grande localité associée à une faible granularité tend à rendre les fautes temporelles très pénalisantes car celles-ci se propagent alors "loin" et très rapidement. De plus, lorsque la densité de messages est grande, une grande localité crée une synchronisation très forte entre les différents PLs, ce qui engendre un nombre important de fautes temporelles de faible distance. Contrairement aux approches optimistes, l'approche conservatrice est favorisée par une grande localité et une grande densité de messages car dans ce cas le trafic de *null-messages* est fortement réduit. De ce fait, en se basant sur les travaux réalisés dans [Fuj90b], on peut alors penser que dans notre cas, une approche purement optimiste peut difficilement être plus performante que son homologue conservateur. C'est pourquoi, plutôt qu'utiliser une approche optimiste traditionnelle, nous proposons dans la section suivante une méthode originale qui consiste à utiliser des estimations locales pour remplacer le mécanisme du retour-arrière jugé trop pénalisant.

6.3 Relâchement de l'ordonnancement

La plupart des améliorations apportées aux protocoles de simulation n'ont jusqu'à présent pas remis en question la causalité. Seul *Moving Time Window* défini dans [Sok89] permettait dans une certaine mesure d'exécuter des événements dont l'estampille était inférieure à l'horloge locale du PL. Il n'y a pas d'autre moyen que le retour-arrière si l'on veut réellement corriger les erreurs propagées par l'envoi de mauvais messages. Maintenant, si l'on accepte ces erreurs, on peut envisager d'utiliser d'autres mécanismes moins lourds que le retour-arrière. Récemment, Pham et Fdida ont proposé une telle approche qui relâche des contraintes de synchronisation au dépend de la précision [Pha95]. Dans cette approche, le mécanisme du retour-arrière est remplacé par des estimations locales qui ont pour but de corriger la trajectoire du PL à partir d'un historique d'informations périodiquement sauvegardées.

Les estimations sont susceptibles de donner de bons résultats lorsque la sémantique associée aux messages échangés est faible. C'est ici le cas puisque les messages échangés ne représentent que des réceptions de cellules. Dans ce cas, l'exécution d'un certain nombre d'événements en "désordre" ne devrait pas affecter de manière significative les résultats de la simulation, cela étant d'autant plus vrai lorsque l'on s'intéresse à des valeurs moyennes de performance. Il faut préciser que l'utilisation des estimations est également motivée par le fait que le comportement du modèle de réseau est susceptible de produire un grand nombre de fautes temporelles de petite distance. Si cela n'est pas le cas, par exemple si l'on a peu de fautes mais des distances plus grandes, une approche traditionnelle pourrait sûrement donner de bons résultats. La petite distance des fautes plaide aussi en la faveur des estimations puisqu'on peut espérer que les erreurs commises seront faibles.

Nous ne présentons pas ici les détails concernant le calcul des estimations locales, les lecteurs intéressés pourront se référer à [Pha95]. Il nous faut cependant préciser que ces estimations sont dépendantes de l'application à simuler et qu'en général, changer d'application nécessite la définition de nouvelles estimations. Les résultats généraux sont les suivants: il est possible d'obtenir des résultats précis lorsque le nombre de fautes temporelles est faible. C'est le cas pour de simples réseaux unidirectionels. Lorsque la complexité du modèle est plus grande, comme c'est le cas pour notre modèle de réseau, le nombre de fautes temporelles augmente fortement réduisant considérablement la précision des estimations. Ces dernières s'avèrent alors inefficaces à corriger correctement la trajectoire de la simulation. Ces résultats impliquent trois directions possibles: (*i*) abandonner l'utilisation des estimations, (*ii*) réduire le nombre de fautes temporelles et (*iii*) améliorer la précision des estimations. Nous avons décidé d'écarter la première et de regarder dans un premier temps la deuxième en étudiant les possibilités de réduire l'optimisme dans les protocoles.

6.4 Limitation de l'optimisme

Plusieurs méthodes pour limiter l'optimisme ont été proposées. Elles peuvent être classées en deux catégories: celles uti-lisant une fenêtre et celles utilisant un mécanisme de blocage. La première catégorie cherche d'une part, à limiter l'espace de recherche de l'événement d'estampille minimale et d'autre part, à éviter que certains PLs aillent trop "loin" dans le futur [Ste91, Mat93]. Les protocoles de la seconde catégorie utilisent le plus souvent un mécanisme de blocage basé sur le temps pour retarder l'exécution des événements [Bal90, Ham94, Fer95]. Très récemment, l'intérêt s'est porté sur ces derniers protocoles qui possèdent la faculté de se positionner entre les deux points extrêmes représentés par les comportements conservateur et purement optimiste. Nous nous intéressons donc à une telle approche pour limiter l'optimisme.

Même si une stratégie de blocage interrompt l'exécution des événements, on peut noter que dans beaucoup de cas, la probabilité de provoquer une faute temporelle décroîe lorsque le nombre d'événements dans la LEV augmente. Cela s'explique par le fait que le PL a plus de "choix" lorsqu'il sélectionne un événement pour l'exécuter. Ensuite exécuter un événement dès qu'il est disponible peut être très intéressant si cet événement est effectivement celui que l'on attend (l'événement est sur le chemin critique de la simulation), mais peut s'avérer très hasardeux sinon. En effet, cette "aggressivité" peut conduire à un nombre excessif de fautes temporelles qui dégradent fortement les performances du simulateur. Finalement, le blocage permet donc de réduire la probabilité de fautes temporelles en permettant à plus d'événements d'être placés dans la LEV. Bien sûr cela est seulement avantageux si le temps passé en blocage est inférieur au temps passé à corriger les fautes temporelles. Dans [Sri95], les auteurs ont fait remarquer qu'une mauvaise décision de blocage peut causer un grand nombre de fautes temporelles supplémentaires.

Contrairement aux approches basées sur un temps de blocage, nous proposons l'utilisation d'une stratégie de blocage basée sur la longueur de la LEV [Pha96]. Un PL est suspendu tant que la longueur de sa LEV demeure inférieure à un seuil L_{min} initialement égal à L_{init}. Il est cependant nécessaire de relâcher la condition de seuil après un certain temps pour éviter les situations de blocage lorsqu'il n'y a pas suffisamment de messages dans le système. Cela est obtenu en décrémentant L_{min} tous les D_t unités de temps (ut). La figure 5 représente l'organigramme de l'algorithme.

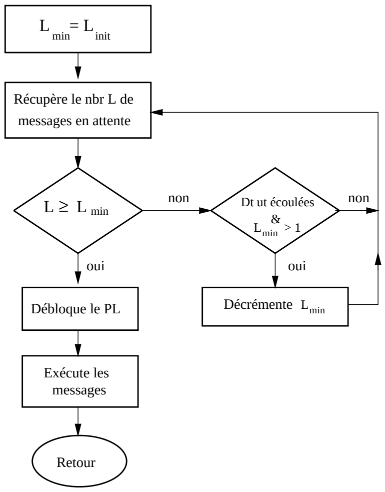

Figure 5. *Algorithme de blocage.*

Peu de travaux ont cependant été effectués sur l'utilisation d'un seuil basé sur la longueur de file plutôt que sur le temps. De manière paradoxale, il est pourtant plus naturel de fixer une longueur de seuil plutôt qu'un temporisateur car on peut difficilement prédire les instants d'arrivée des messages. D'une part, les approches basées sur le temps autorisent un réglage très précis puisque le temporisateur peut prendre des valeurs réelles mais d'un autre coté, avec une longueur de seuil, il est possible de réduire le coût du blocage puisque la condition peut être satisfaite immédiatement. Il peut donc y avoir des cas où les premiers doivent bloquer et pas les seconds, sans risque supplémentaire.

Nous avons mis en application cette méthode et la figure 6 représente le taux de fautes temporelles (TER) en fonction de la valeur initiale L_{init} pour notre modèle de réseau avec $N_s = 5$ et $n_c = 10$. La courbe en pointillés a été obtenue en utilisant par^{SS20} et celle en trait plein par^{CM5}. On peut remarquer deux choses: (i) au fur et à mesure que L_{init} augmente, le TER diminue jusqu'à atteindre un minimum pour ensuite augmenter de nouveau, (ii) le TER diminue plus rapidement sur la CM5 que sur la SS20 résultant en une valeur de L_{init} optimale plus faible sur la CM5.

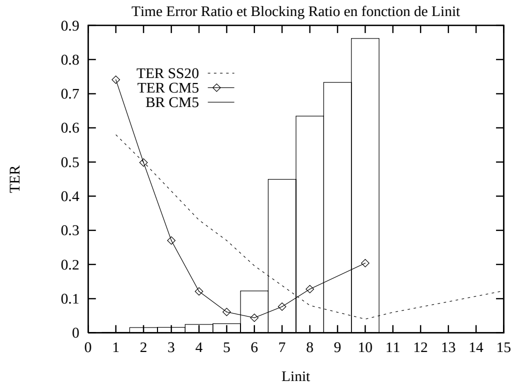

Figure 6. *TER en fonction de L_{init}*

Lorsque L_{init} est trop grand, les PLs se bloquent mutuellement car la condition de seuil est difficilement réalisable. Après un certain temps, L_{min} ayant diminué, certains PLs sont brutalement relâchés et envoient des messages qui sont ensuite exécutés de manière très "agressive" par les autres PLs (puisque L_{min} est maintenant très bas). Ceci explique le premier phénomène (similaire au phénomène décrit dans [Sri95]). Le deuxième phénomème est expliqué par le faible coût des communications et par l'augmentation de la puissance de calcul sur la CM5, ce qui permet aux messages d'être échangés plus rapidement entre les PLs. On peut également noter le fort taux de fautes temporelles lorsque $L_{init} = 1$, équivalent à une approche purement optimiste, et en particulier lorsque la granularité est faible comme c'est plus le cas sur la CM5 que sur la SS20. Notre première impression concernant le comportement de notre modèle est donc justifiée. La figure 6 montre également la proportion de temps passé en blocage qui est particulièrement faible pour les valeurs intéressantes de L_{init}.

Finalement, on constate qu'il est possible de réduire considérablement le nombre de fautes temporelles avec une surcharge supplémentaire très faible. Ce succès est pour une partie lié à la topologie du réseau qui engendre un grand nombre de fautes temporelles de faible distance. Cependant, cette stratégie de blocage est suffisamment générale pour pouvoir être utilisée avec d'autres protocoles optimistes.

6.5 Limiter l'optimisme et utilisation des estimations locales

Combiner la limitation de l'optimisme et l'utilisation des estimations locales est une suite logique de notre démarche. Nous avons vu que les estimations peuvent dans une certaine mesure corriger les erreurs de trajectoire sans effectuer de retour-arrière, mais qu'elles peuvent aussi être inefficaces si le nombre de fautes temporelles est trop élevé. En limitant l'optimisme, il est possible d'en réduire le nombre afin de permettre aux estimations de mieux fonctionner. La table 1 présente les résultats obtenus sur CM5 pour le modèle de réseau

L_{init}	TER	L	% d'erreur
6	0.044	1.27	+1.56%
CMB	0	1.25	0%
5	0.061	1.22	-1.69%
4	0.121	1.20	-3.72%
3	0.270	1.07	-13.6%
2	0.498	0.93	-25.6%
1	0.740	0.77	-33.88%

Table 1. *TER et pourcentage d'erreur en fonction de* L_{init}

avec $N_s = 5$ et $n_c = 10$. Nous nous sommes intéressé à la longueur moyenne L des files de sortie des commutateurs. Seuls les résultats pour la première sortie du commutateur 1 sont présentés, les autres étant très similaires. Cette table indique le taux de fautes temporelles et le pourcentage d'erreur par rapport aux résultats fournis par la méthode conservatrice (CMB) en fonction de L_{init}. On constate alors que la précision que l'on obtient en utilisant les estimations locales augmente de manière significative au fur et à mesure que L_{init} augmente. Pour $L_{init} = 4, 5$ les erreurs sont inférieures à 5%, ce qui est très raisonnable. Cette petite perte en précision est en contre-partie compensée par une vitesse accrûe puisqu'il est maintenant possible d'aller plus vite que l'approche conservatrice. La figure 7 représente le speedup par rapport à l'approche conservatrice.

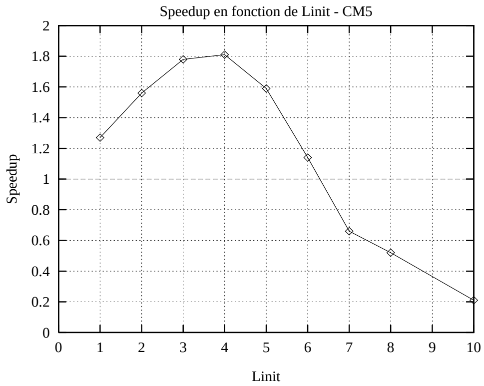

Figure 7. *Speedup en fonction de* L_{init}

Cette figure présente une caractéristique intéressante dans la mesure où l'on peut noter un pic se situant à $L_{init} = 4$ alors que pour des petites valeurs de L_{init} on s'aperçoit que le gain est moindre. Ce phénomène peut être expliqué de la manière suivante: lorsque L_{init} est petit, le nombre de fautes temporelles est grand, ce qui veut dire qu'un grand nombre d'estimations sont faites. Bien que ces estimations ne doivent pas prendre beaucoup de temps (car la distance des fautes est relativement petite), un grand nombre de fautes peut cependant ralentir de manière significative la simulation. Lorsque L_{init} est grand, on perd beaucoup de temps à la fois en blocage et en estimations car le TER est élevé (cf figure 6).

7 Conclusion

Dans cet article, nous avons montré la nécessité d'utiliser des techniques de simulation distribuée pour évaluer les futurs réseaux large bande. L'utilisation d'une approche conservatrice montre que de très forts speedups peuvent être obtenus. En particulier, il est clairement apparu qu'une approche séquentielle est très rapidement "débordée".

Nous avons ensuite utilisé une approche optimiste avec des estimations locales, dépendantes de l'application, pour remplacer le mécanisme du retour-arrière jugé trop pénalisant dans notre cas. Il s'avére que cette solution donne de bons résultats lorsque le nombre de fautes temporelles est réduit, mais qu'elle peut être inefficace dans le cas contraire. Pour réduire le nombre de fautes temporelles, nous avons proposé de manière générale une stratégie de blocage basée sur la longueur de la file d'attente. Les résultats montrent que le nombre de fautes temporelles peut être réduit considérablement avec une très faible proportion de blocage. Une suite logique consistant à combiner les deux approches, estimations locales et stratégie de blocage montre qu'il est possible d'aller plus vite qu'une approche conservatrice tout en gardant une grande précision.

On ne peut pas dire que notre approche soit générale, en particulier ce qui concerne les estimations locales, mais cependant, lorsque la sémantique des messages est faible, il est envisageable d'utiliser la même démarche pour réduire le coût de la simulation. Nous souhaitons étendre nos travaux dans deux directions: améliorer la précisions des estimations et évaluer dans quelles situations (topologie du réseau, type de trafic, critères de performances étudiés) un speedup d'un ordre de grandeur peut être obtenu.

Références

[Bal90] D. Ball and S. Hoyt, "The Adaptive Time-Warp Concurrency Control Algorithm", *Proceeding of the 1990 SCS Multiconference on Distributed Simulation, San Diego, California, pp174-177.*

[Cha79] K. M. Chandy and J. Misra,"Distribution Simulation: A Case Study in Design and Verification of Distributed Programs", *Transactions on Software Engineering, Vol. 5(5), May 1979, pp440-452.*

[Fer95] A. Ferscha and J. Lüthi, "Estimating Rollback Overhead for Optimism Control in Time Warp", *Proceedings of the 28th Annual Simulation Symposium, Phoenix, Arizona, April 9-13, 1995, pages 2-12.*

[Fuj90a] R. M. Fujimoto, "Parallel Discrete Event Simulation", *Communication of the ACM, Vol. 33(10), October 1990, pp31-53.*

[Fuj90b] R. M. Fujimoto, "Performance of Time Warp under synthetic work-loads", *Proceeding of the 1990 SCS Multiconference on Distributed Simulation, San Diego, California, pp23-28.*

[Ham94] D. O. Hamnes and A. Tripathi, "Investigations in Adaptive Distributed Simulation", *Proceedings of 8th Workshop on Parallel and Distributed Simulation (PADS'94), Edinburg, Scotland, U. K., pp20-23.*

[Han94] R. Händel, M. N. Nuber and S. Schröder, "ATM Networks: Concepts, Protocols, Applications", *Addison-Wesley, 1994.*

[Jef85] D. R. Jefferson, "Virtual Time", *ACM Transaction on Programming Languages and Systems, Vol. 7(3), July 1985, pp405-425.*

[Mat93] Y. Matsumoto and K. Taki, "Adaptive Time-Ceiling for Efficient Parallel Discrete Event Simulation", *Proceedings of the 1993 Object-Oriented Simulation Conference, pp101-106.*

[Pha95] C. D. Pham and S. Fdida, "The Statistically Correct Approach to Distributed Simulation of ATM Network", *Proceedings of the 6th IFIP Conference on Performance of Computer Networks, Istanbul, Turkey, October 23-26 1995, pp352-366*

[Pha96] C. D. Pham and S. Fdida, "Length-Based Blocking and Local Estimation in Distributed Simulation: A Case Study", *To appear in Proceeding of the 29th Annual Simulation Symposium Louisiana, New Orleans, April 8-11 1996.*

[Pvm93] A. Geist and al., "PVM 3 User's Guide and Reference Manual", *May 1993.*

[Sok89] L. M. Sokol, B. K. Stucky and V. S. Hwang, "MTW : A Control Mechanism for Parallel Discrete Simulation", *Proceedings of the International Conference on Parallel Processing, 1989, pp250-254.*

[Sri95] S. Srinivasan and P. F. Reynolds Jr.,"Adaptive Algorithms vs. Time Warp: An Analytical Comparison", *To appear in Proceedings of the 1995 Winter Simulation Conference.*

[Ste91] J. S. Steinman, "Interactive SPEEDES", *The 24th Annual Simulation Symposium, New Orleans, Louisiana, April 1-5,1991, pp149-158.*