Choix d'un service de communication pour un serveur vidéo tolérant aux pannes : étude du service GM.

Alice Bonhomme, Loïc Prylli

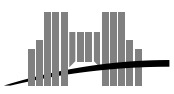
Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON
n° 5668

Choix d'un service de communication pour un serveur vidéo tolérant aux pannes : étude du service GM

Alice Bonhomme et Loïc Prylli Février 1999

Research Report N° 1999-13



École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr

Choix d'un service de communication pour un serveur vidéo tolérant aux pannes : étude du service GM

Alice Bonhomme et Loïc Prylli

Février 1999

Abstract

In the context of a video server project, we look at fault tolerant communication services. As the Myrinet network has been selected for this project, we focus on the GM system, which has the appropriate properties, especially in terms of error recovery. After looking what requirements a server video put on the communication system, we propose an error recovery strategy for the Myrinet network and we show how one can implement it with the GM system.

Keywords: Video Server, Fault Tolerance, Myrinet Network, GM

Résumé

Dans le cadre d'un projet de serveur vidéo, nous nous intéressons aux services de communication tolérants aux pannes. Le réseau Myrinet ayant été retenu pour ce projet, nous nous focalisons sur le système GM qui a les bonnes propriétés, notamment en terme de reprise sur erreur. Après une étude des besoins imposés par un serveur vidéo sur le service de communication (détection de panne réseau, rajout ou réinitialisation de machine, etc), nous proposons une stratégie de détection d'erreur pour le réseau Myrinet, et montrons comment le système GM permet de l'implémenter.

Mots-clés: Serveur Vidéo, Tolérance aux pannes, Réseau Myrinet, GM

1 Introduction

Les serveurs de fichiers, en particulier les serveurs vidéo, sont un domaine d'applications où les solutions distribuées s'imposent, étant donné le volume important de données à stocker et le débit global à fournir en sortie. On distingue les serveurs de fichiers "classiques" de type NFS, AFS ou autre, alimentant des stations, et les serveurs de flux vidéos qui rajoutent en plus des contraintes temps-réel. Pour tous ces serveurs, on ne peut accepter que la panne d'un composant de la solution distribuée entraîne une interruption du service. Par conséquent, pour fournir le haut degré de disponibilité requis, l'ensemble de l'architecture logicielle doit être tolérante aux pannes. Dans ce rapport, nous nous focalisons sur le service de communication interne pour un serveur vidéo (donc avec des contraintes temps-réel) : le réseau de distribution avec les clients n'est pas abordé. Pour des raisons incluant le rapport performance/prix et l'environnement logiciel disponible, le serveur de données motivant cette étude est basé sur un ensemble de PC reliés par un réseau Myrinet. Il nous a fallu choisir entre les nombreux systèmes de communication disponibles pour ce réseau :

- Fast Message FM, [3] (University of Illinois, USA),
- Active Message AM, [4] (University of Berkeley, USA),
- BIP [6] (LIP/LIGIM, Lyon, France),
- PM [7] (RWCP, Japon),
- GM [5], (Myricom Inc., USA.

Tous ces logiciels proposent une implémentation de MPI ainsi que la plupart du temps une API propre. En revanche, une majeure partie de ces systèmes ne proposent aucune gestion des pannes éventuelles, ce qui est limitant pour la conception d'un serveur vidéo scalable et tolérant aux pannes. En effet, pour servir de base à un serveur vidéo, un service de communication doit pouvoir permettre les opérations suivantes :

- rajout d'une machine à la configuration,
- détection de pannes réseau ou machine,
- réinitialisation d'une machine,
- isolement des pannes réseaux localisées.

Comme nous le verrons, GM s'est avéré le plus adapté pour répondre à ces besoins et nous décrirons son mode de fonctionnement et sa mise en œuvre.

Ce rapport s'organise de la manière suivante. Après une description de la technologie Myrinet (section 2), nous décrivons (section 3) l'architecture matérielle et logicielle du projet. Ensuite nous présentons le système GM (section 4) et les raisons de son choix parmi les systèmes mentionnés, ainsi que ses performances (section 5). Enfin, dans la section 6, nous traiterons des stratégies de détection et de récupération en cas de panne.

2 Description de la technologie Myrinet

Myrinet est une technologie réseau, basée sur des liens de communication point à point reliant entre eux les commutateurs et les nœuds finaux du réseau. Cette technologie est construite pour des débits de 1,28 Gbits/s sur chaque

lien, tout en garantissant des latences très faibles. Le contrôle de flux sur les liens, effectué par rétropropagation, évite la possibilité de perte de messages par congestion. D'autre part, la technologie utilisée garantit un taux d'erreur extrêmement faible ($< 10^{-15}$). Les commutateurs possèdent une bande passante agrégée suffisante pour supporter simultanément le débit maximal sur tous les liens entrants et sortants. Myrinet est utilisé aussi bien en tant que réseau *local* (LAN) de petit diamètre, ou réseau *système* (SAN). Le routage est effectué par la source, chaque message doit contenir dans son entête des octets de routage qui sont consommés par les commutateurs au fur et à mesure de sa progression.

Les composants matériels et logiciels :

La technologie Myrinet s'appuie sur des composants matériels standards :

- *câble électrique pour LAN* : formé de 18 paires torsadées blindées. Ces câbles vont jusqu'à 10 mètres.
- *câble électrique pour SAN* : formé d'une nappe de 2x20 signaux plus des fils de masse. Depuis 1998, les connecteurs ont un petit verrou : attention, pour une raison inconnue, ces nouveaux câbles ne fonctionnent pas toujours avec les anciens matériels.
- *commutateur Myrinet* : 4, 8, 16 ou 32 ports avec des connecteurs SAN, LAN ou des configurations mixtes.
- *interface réseau* : elle possède un port et permet la connexion entre l'ordinateur hôte et le réseau. Elle possède un processeur et une mémoire SRAM (de 128Ko à 1Mo). Cette mémoire sert à la fois de tampon pour la gestion des communications et de support pour les variables et le code du programme de contrôle. La première génération des cartes largement diffusée comprend la gestion de 3 DMA (2 DMA SRAM-réseau et 1 DMA SRAM-PCI). A partir de 1999, la nouvelle génération introduite gère :
  - le PCI 64 bits (double la bande passante et augmente l'espace d'adressage)
  - 6 DMA indépendants : 2 DMA SRAM-réseau, 2 DMA SRAM-PCI (un dans chaque sens) et 2 DMA "gather-scatter" permettant de transférer des données non contigües en mémoire.
  - un mécanisme de "doorbell" : il permet de partager le réseau de manière sécurisée entre un grand nombre de processus, sur le même principe que les "doorbells" de VIA [1].

Du point de vue logiciel, le programme de contrôle (MCP) est chargé par l'hôte dans la mémoire SRAM de carte d'interface réseau et s'exécute dès que le pilote de la carte en donne l'ordre. Il doit assurer la réception et l'émission des paquets, et gérer des files d'attente. Chaque système logiciel disponible pour Myrinet vient avec son propre MCP.

3 Architecture matérielle et logicielle

Le système vidéo que nous considérons se décompose en trois parties :

- un ensemble de clients,
- un réseau de distribution entre le serveur principal et les clients,

– un serveur de données parallèle. Ce serveur possède une mémoire de masse importante pour le stockage des données et est chargé de gérer les accès à ces données (enregistrement et lecture).

Dans notre étude, nous considérons uniquement le serveur de données. Ce serveur (cf. figure 1) est composé d'un ensemble de PC possédant chacun des unités de stockage. Ces PC sont reliés par un réseau Myrinet indépendant du réseau de distribution.

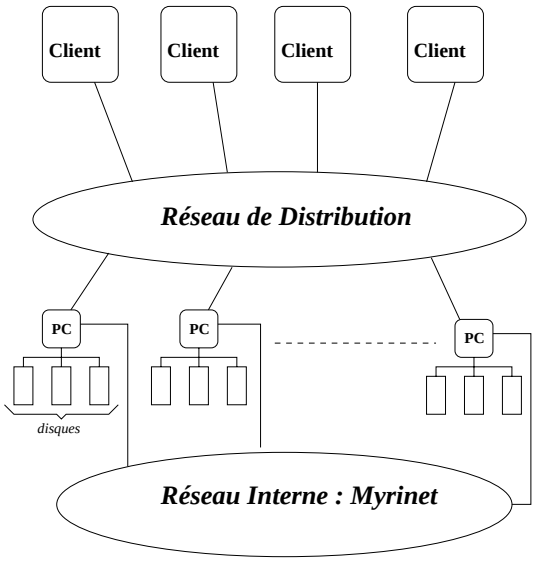


FIG. 1: Architecture matérielle du serveur vidéo

Du point de vue logiciel (cf figure 2), le serveur se décompose en deux sous-ensembles :

le gestionnaire client : chargé du dialogue avec le client : contrôle d'admission, traitement des requêtes (accès interactif : pause, avance rapide). Ce gestionnaire est distribué sur l'ensemble des nœuds du serveur.

le gestionnaire de stockage : il gère l'ensemble des unités de mémoire de masse du serveur. Pour la lecture, les accès aux données se font en trois étapes : localisation des données, lecture sur les différentes unités, et réassemblage des données pour former le flux final. L'écriture est symétrique.

Les données doivent être allouées de manière redondante en s'assurant de pouvoir prendre en compte une panne de n'importe quel nœud du réseau. Symétriquement, la localisation des données pour la lecture doit s'adapter dynamiquement en cas de panne. Il est ainsi possible d'allouer les données sur les nœuds suivant une allocation RAID-1 ou RAID-5.

4 Présentation de GM

GM[1] est un système de communication développé par Myricom pour le réseau Myrinet.

[1] `http:\\www.myri.com\GM`

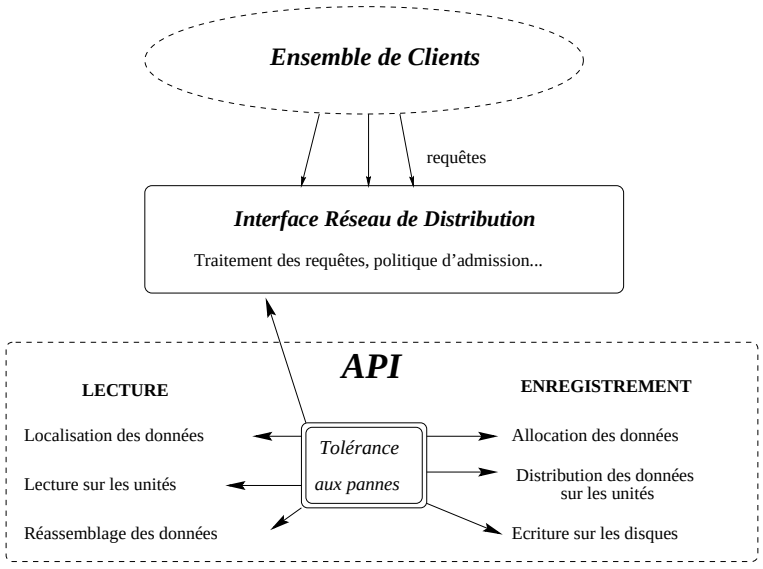


FIG. 2: Architecture logicielle du serveur vidéo

Ce système se veut portable, procure une faible latence et une forte bande passante tout en générant peu d'overhead de calculs.

4.1 Caractéristiques du service GM

- Les messages dans GM sont adressés à des ports. Le port se décompose en un numéro de nœud, *NodeId* et un numéro de port local au nœud, *PortId*.
- GM est "connectionless", un message peut être envoyé à n'importe quel port sur le réseau, sans négociations préalables.
- GM établit de manière transparente 2 canaux entre chaque paire de ports, appelés abusivement HIGH_PRIORITY et LOW_PRIORITY, bien qu'ils soient symétriques. Sur chaque canal, l'ordre d'envoi des messages est préservé et ils sont délivrés de manière sure (tolérance aux pannes du réseau). En revanche, les canaux sont indépendants, un message bloqué sur l'un d'eux n'empêche pas d'envoyer et de recevoir des messages ultérieurs sur l'autre canal.
- Sur un canal, la correspondance entre les envois et les buffers de réception se fait à l'aide d'un tag entre 0 et 31. Ce tag impose par ailleurs une borne sur le logarithme de la longueur du message.
- Les programmes utilisant GM communiquent directement avec le processeur LANai de la carte Myrinet (sans passer par le système d'exploitation).
- GM procure des accès utilisateurs concurrents et protégés à la carte d'interface Myrinet : plusieurs programmes utilisant GM peuvent être exécutés sur la même machine,
- GM est "scalable" : on peut rajouter des machines sur le réseau, elles sont alors automatiquement détectées (par le démon Mapper présent sur une

des machines du réseau) et rajoutées dans la topologie du réseau.

Actuellement GM est disponible pour Linux (Alpha, x86), Solaris 2.5, Solaris 2.6 (UltraSparc et x86) et NT 4.0 (x86). Courant 1999, devraient se rajouter Irix, NT 4.0 (alpha), NT 5.0 (alpha, x86). Son installation sur NT est commentée dans l'annexe du présent rapport.

4.2 Motivations pour le choix de GM

Dans les systèmes disponibles sur Myrinet, BIP [6] et FM [3] n'offrent pas certaines fonctionnalités indispensables : la remontée d'erreurs fatales (par exemple le redémarrage d'une machine) et l'adaptation au changement du réseau (rajout de machines ou changement de topologie). Si IP ou PVM possèdent ces fonctionnalités, ils n'ont pas été retenus principalement pour des raisons de performance. Après cette première sélection, deux systèmes, AM et GM, conviennent.

Dans ces deux systèmes, les communications se font en mode non connecté plutôt que par canaux de communication comme sous VIA [1] ou TCP. Les messages sont adressés à des ports (appelés "endpoint" dans AM), équivalents aux "TaskIDentifier" de PVM. Dans tous les cas, la transmission se fait de manière sûre (sauf en cas de panne prolongée où une erreur est remontée à l'application). L'ordre d'envoi des messages entre deux nœuds est préservé. La présence d'un démon *mapper* sur une des machines du réseau permet de détecter tout rajout de machine sur le réseau ou tout changement de sa topologie. Nous avons choisi GM, plutôt que AM, pour des raisons pratiques (support pour la plupart des systèmes d'exploitation et en particulier NT), et pour certaines possibilités de bas niveau comme la possibilité de manipuler explicitement la table de routage.

5 Performances

Dans cette partie, nous présentons deux types de résultats : les performances obtenues pour un ping-pong entre deux machines, et celles calculées sur l'envoi d'un flux de paquets pipelinés. Les expérimentations sont faites pour NT sur des Pentium 2 (350 MHz), et pour Linux sur des PentiumPro 200 MHz, reliés dans les deux cas par un réseau Myrinet. L'influence du type de stations peut être observée par le biais des courbes GM disponibles sur les deux architectures.

5.1 Ping-pong

La figure 3 nous montre l'évolution de la latence en fonction de la taille des messages pour GM sous NT et GM sous Linux, ainsi que pour IP ou BIP sous Linux. Le temps représenté est la moitié du temps nécessaire pour l'aller-retour d'un paquet. Pour un message de très petite taille on obtient avec GM une latence de 30 μs. On remarque une évolution en *escalier*, ceci correspond au découpage du message envoyé en paquets (GM utilise des paquets de 4096 octets).

La figure 4 permet de visualiser les mêmes résultats sous forme de débit au lieu de latence (les ordonnées correspondent donc au nombre d'octets passés sur le réseau pendant une seconde de ping-pongs successifs). Malgré la légère différence entre GM sous NT ou Linux, due à l'utilisation de machines différentes

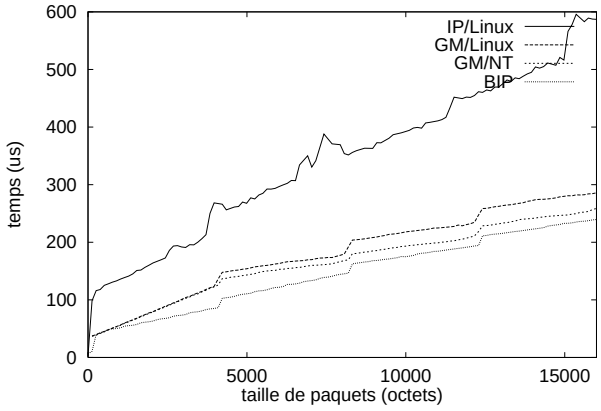


FIG. 3: Latence pour GM (NT ou Linux), BIP et IP

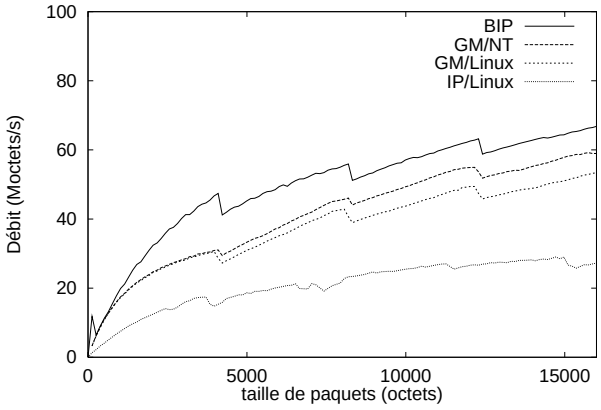


FIG. 4: Débit (sans pipeline) pour GM (NT ou Linux), BIP et IP

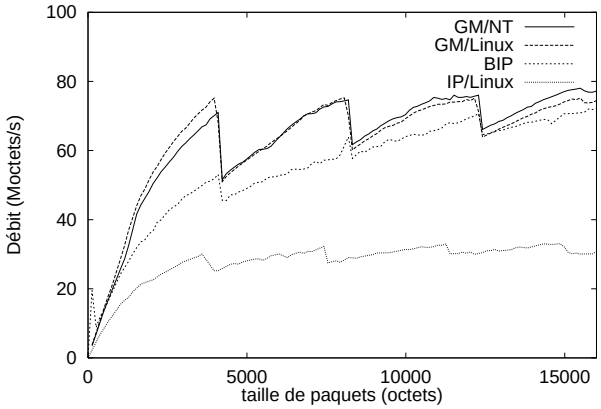


FIG. 5: Bande Passante pour GM (NT, Linux) BIP et IP

dans les expériences, le choix d'un système d'exploitation tel que Linux ou NT n'a que peu d'influence sur les performances. Ceci s'explique par le fait que GM ne fait pas intervenir le système d'exploitation sur le chemin critique des communications.

5.2 Pipeline

Ce deuxième jeu d'expériences évalue le nombre maximum d'octets que peut supporter un réseau pour des envois pipelinés les uns derrière les autres, de messages de taille fixée. La figure 5 présente les résultats obtenus pour GM sous NT ou Linux, IP ou BIP. On remarque qu'on atteint très rapidement un débit important alors, que pour l'expérience de la figure 4, le débit augmente avec la taille du message. Si BIP dominait en terme de ping-pong, on remarque qu'il ne pipeline pas aussi efficacement que GM. Dans le cadre d'accès aux données réparties sur l'ensemble des nœuds du serveur, cette expérience pipeline est représentative de l'utilisation réelle, pour des paquets de l'ordre de la dizaine de koctets, qui correspondent à la granularité de distribution envisagée. Les courbes montrent que cette taille permet bien une utilisation efficace du matériel.

6 Tolérance aux pannes et extensibilité

6.1 Les besoins d'un serveur vidéo

Dans le cadre d'un serveur de taille importante, les défaillances possibles à prendre en compte sont : une carte réseau, une unité de stockage ou un nœud dans sa totalité. Pour respecter le cahier des charges du serveur, une panne individuelle doit être complètement transparente vis-à-vis des clients, ils doivent continuer à recevoir leur flux sans perturbation visible. D'autre part, on doit pouvoir étendre le serveur par ajout d'unités supplémentaires sans interruption du service donc sans réinitialisation des nœuds existants.

De nombreuses études ont déjà été effectuées dans ce domaine, essentiellement au niveau algorithmique et simulation [2]. Notre but est d'étudier comment

mettre en oeuvre ces techniques sur une plate-forme réelle en commençant par les fonctionnalités requises au niveau du service de communication.

6.2 Stratégies de détection et reprise sur erreur

Les trois cas de panne possibles : unité de stockage, carte ou lien réseau, ou nœud complet, vont rendre le nœud correspondant complètement inutilisable. Les données servies par ce nœud doivent soit être rapatriées à partir d'autres nœuds dans le cas d'un stockage redondant par copies multiples des blocs de chaque flux, soit être reconstruites sans le nœud défectueux dans le cas d'une redondance logicielle par des techniques générales de parité ou de code correcteur d'erreurs. Dans le cas de panne de l'unité de stockage, la détection est immédiate. Dans les autres cas, il est nécessaire de suivre les étapes décrites ci-dessous.

6.2.1 Détection initiale d'un problème

Au niveau du service de communication, une panne doit être détectée le plus rapidement possible, plusieurs méthodes sont proposées :

non réception de l'accusé de réception : cet accusé est systématique dans GM, permettant la détection de panne dans un délai fixe après l'envoi d'un message.

non réception d'un message attendu : si à un instant donné, un nœud est uniquement récepteur, il ne peut pas compter sur un acquittement pour détecter une panne de son correspondant. On peut utiliser une connaissance a priori sur la date de réception de certaines données pour détecter une panne éventuelle.

non réception d'un jeton de contrôle : on fait circuler un jeton entre les différents nœuds du réseau, toute interruption de cette circulation détecte une panne.

En pratique la combinaison des deux premières méthodes est nécessaire pour détecter les pannes quel que soit le trafic de communication. Toutefois, cette première étape ne suffit pas car elle ne permet pas de différencier les situations suivantes :

- la machine correspondante est en panne,
- la machine effectuant la détection est isolée du réseau,
- la panne se situe au niveau des liens inter-commutateurs.

6.2.2 Identification de la cause de la panne

Lors de la détection d'un problème, on teste la connectivité des commutateurs sur cette route par distance croissante, jusqu'à l'absence de réponse. Ceci permet de localiser le lien défaillant ou diagnostiquer la panne du correspondant. Cet algorithme peut conduire soit à la suppression d'une machine si c'est son lien station-réseau qui est en cause, soit à l'utilisation d'une nouvelle route évitant le lien inter-commutateur défaillant, s'il en existe une (ceci implique de connaître la topologie complète du réseau sur chaque nœud).

6.2.3 Élimination d'une machine

À l'issue de la confirmation et localisation d'une panne machine, tous les autres nœuds en sont avertis, et peuvent alors modifier leur approvisionnement en données en conséquence. Nous n'aborderons pas les problèmes de l'ajout ou modification de fichiers lors d'une panne, qui nécessitent une reconstruction ultérieure complexe à l'issue de la réparation. Pour un serveur de fichiers classiques c'est une limitation sérieuse, en revanche pour des utilisations spécialisées comme la vidéo, les modifications ne sont pas des opérations critiques. Le retour en service d'une machine requiert à nouveau une phase de synchronisation avec tous les nœuds. Ce protocole est dirigé par la machine qui était en panne.

6.3 Validation expérimentale de GM pour les cas de panne

6.3.1 Pannes usuelles

GM a son propre système de récupération sur erreurs en cas de problèmes transitoires sur le réseau. Nous avons réalisé une application-test pour trois machines. Au début de l'application, chaque machine a un jeton numéroté avec son numéro de nœud. On associe à chaque jeton un compteur permettant de savoir le nombre d'envois qu'a subit ce jeton. Ensuite, chaque machine tire au sort une autre machine parmi les trois et lui envoie le jeton avec son compteur. Ainsi les jetons circulent aléatoirement entre les trois machines.
Pour effectuer les tests de tolérance aux pannes, on fait tourner cette application indéfiniment, et on perturbe ensuite le réseau (débranchement de câble, arrêt de machine, arrêt de mapper....) et on étudie le comportement du réseau et de GM. Ceci nous a permis de vérifier les propriétés suivantes de GM :

- en cas de panne réseau transitoire (perte de paquets, déconnexion inférieure à 10 secondes), GM achemine de manière fiable les paquets en les retransmettant de manière transparente,
- au bout d'une dizaine de secondes, l'impossibilité d'envoyer est reportée à l'application,
- si un correspondant est en panne puis redémarré, il est possible de communiquer à nouveau avec lui sans réinitialisation des autres parties. De même si un problème réseau est finalement résolu après le report d'un échec de transmission, la transmission peut reprendre,
- des problèmes de communication avec une destination particulière n'influencent pas les communications avec les autres, même si elles ont des messages en cours avec la destination fautive.

Dans le cadre de la vidéo, des délais de détection de l'ordre de la dizaine de secondes sont beaucoup trop grands par rapport aux contraintes temps-réel. C'est pourquoi la détection d'un problème réseau doit suivre la méthode du paragraphe 6.2.1 plutôt que d'attendre la remontée d'une erreur par GM.

6.3.2 Perturbations du Mapper

Pour rendre le réseau "auto-reconfigurable", GM utilise la notion de *mapper*. Sur l'ensemble des nœuds du réseau, un nœud est désigné *mapper* : il va détecter l'arrivée ou le départ de machines sur le réseau et mettre à jour la table de routage. Il est alors important d'étudier le degré de dépendance des nœuds du

réseau par rapport au nœud mapper. Nous avons effectué un ensemble de tests en supprimant le mapper et en changeant la topologie du réseau. Les résultats montrent que l'utilisation d'un unique mapper au sein du réseau n'est pas problématique. En effet, la panne du mapper n'engendre pas de perturbations par rapport aux autres nœuds du réseau (la table des routes est sauvegardée). Toutefois, l'arrivée d'un nouveau nœud sur le réseau ou le changement de topologie ne seront pas détectés. Il n'est pas restrictif de limiter ces opérations manuelles aux cas où le mapper fonctionne.

Certaines précautions sont à prendre si on veut redémarrer le mapper sur une autre machine. Le problème est de garder la numérotation des nœuds qui a été choisie par le premier mapper. Ces informations sont stockées dans un fichier `mapper.hosts` que le mapper consulte au démarrage et qu'il met à jour en fonction des rajouts de machines. Ce fichier doit donc être récupéré sur l'ancienne machine mapper et installé sur la nouvelle.

7 Conclusion

Ce rapport présente un aspect de la réalisation d'un serveur vidéo tolérant aux pannes : les contraintes sur le service de communication (détection des pannes et reprise sur erreurs). Pour l'architecture Myrinet, au vu des fonctionnalités requises, le système GM apparaît comme le choix le plus pertinent.

Nous avons présenté ce système, en insistant plus particulièrement sur son comportement en cas de pannes. Il fournit les services de base nécessaires, en particulier c'est un système dynamique permettant de rajouter des nœuds sans interruption ou de faire une reprise après une panne sans réinitialisation. De plus, il permet de construire des fonctionnalités plus spécialisées : comme le changement du routage en cas de lien défectueux.

En faisant l'hypothèse de pouvoir tolérer une seule panne, nous avons décrit la démarche pour détecter et isoler si nécessaire une telle panne, en respectant la disponibilité du service, et les contraintes temps-réel. Les travaux futurs vont concerner l'implémentation des stratégies considérées pour réaliser le sous-ensemble de gestion des données mentionné dans la partie 3.

Références

[1] Philip Buonadonna, Andrew Geweke, and David E. Culler. Implementation and analysis of the virtual interface architecture. In *SC'98 : High Performance Networking and Computing : ACM/IEEE SC'98, Orlando, USA*, November 1998.

[2] Leana Golubchik, John C. S. Lui, and Maria Papadopouli. Survey of approaches to fault tolerant design of VOD servers : techniques, analysis and comparison. *Parallel Computing*, 24(1) :123–155, January 1998.

[3] M. Lauria and A. Chien. MPI-FM : High performance MPI on workstation clusters. *Parallel and Distributed Computing*, 40(1) :4–18, January 1997.

[4] Alan M. Mainwaring and David E. Culler. Active messages : Organization and applications programming interface. `http://now.cs.berkeley.edu/Papers/Papers/am-spec.ps`, 1995.

[5] Myricom, Inc. *The GM Message Passing System*, 1996. `http://www.myri.com/GM`.

[6] Loïc Prylli and Bernard Tourancheau. BIP : a new protocol designed for high performance networking on myrinet. In *Parallel and Distributed Processing, IPPS/SPDP'98*, volume 1388 of *Lecture Notes in Computer Science*, pages 472–485. Springer-Verlag, April 1998.

[7] Hiroshi Tezuka, Atsushi Hori, Yutaka Ishikawa, and Mitsuhisa Sato. PM : An Operating System Coordinated High Performance Communication Library. In *High-Performance Computing and Networking'97*, volume 1225 of *Lecture Notes in Computer Science*, pages 708–717, April 1997.

A Installation de GM

Lors de l'installation de GM sur une plate-forme de 3 PC sous NT4.0 et des tests de cette installation, nous avons rencontré quelques points critiques, en voici une brève présentation :

- **Installation de GM** :

  Les opérations à effectuer pour installer GM (à partir des binaires récupérable sur le site `http:\\www.myri.com\GM`) sous NT sont très simples. Il peut nécessiter au préalable une modification de l'attribution des IRQ au niveau du BIOS.

- **Démarrage du Mapper** :

  A chaque redémarrage d'une machine, un démon *gm_mapper_service* est lancé, il permet de mettre à jour la table de routage des machines présentes sur le réseau en dialoguant avec la machine mapper. Si ce démon est arrêté, la table de routage reste en mémoire, elle ne sera simplement plus actualisée.

- **Réinitialisation du pilote et de la carte** :

  Si l'on n'utilise pas le pilote IP associé à Myrinet mais des applications utilisant directement l'API GM, il est possible de réinitialiser GM et la carte Myrinet sans redémarrer la machine. Il faut avoir au préalable supprimer le fichier gm_miniport.sys (dans `c : \winnt\System32\drivers\`). La réinitialisation se fait en arrêtant le mapper puis GM pour ensuite les relancer (`net stop gm_mapper ; net stop gm ; net start gm_mapper ; net start gm`).
  Ceci est particulièrement intéressant lorsque l'on interrompt un programme effectuant des communications au dessus de GM, ceci peut aboutir à un état de GM indéfini vis à vis des messages qui étaient en cours de transmission. Dans ce cas, si on ne réinitialise pas, on peut aboutir à des comportements aberrants : impossibilité d'émettre, réception de messages périmés.