Architectures logicielles pour la musique

Dominique Fober - Stéphane Letz - Yann Orlarey

GRAME
Laboratoire de recherche
6, quai Jean Moulin BP 1185
69202 LYON CEDEX 01
Tel : (33) 72.07.37.00 Fax : (33) 72.07.37.01
E-mail : GRAME@AppleLink.Apple.com

RESUME. Le propos de cet article est de montrer d'une part, pourquoi des architectures logicielles particulières sont nécessaires au domaine de l'informatique musicale et d'autre part, comment les différents problèmes qui requièrent ces architectures peuvent être résolus. En adoptant un point de vue particulier qui est celui du temps et de la communication, nous présenterons un système d'exploitation dédié au domaine musical. Nous présenterons son extension aux réseaux locaux de type Ethernet et montrerons enfin comment cette architecture induit la collaboration entre applications.

MOTS-CLES. architectures logicielles, systèmes d'exploitation, informatique, temps-réel, communication, collaboration, musique, MIDI, MidiShare, réseaux, Ethernet.

1 Introduction

L'informatique musicale est une discipline qui remonte à la fin des années 50. Elle naît pour partie de l'évolution de la pensée musicale qui ressent le besoin d'outils plus puissants afin de mettre en oeuvre des formalismes nouveaux, reposant sur des modèles d'écriture inspirés par exemple de l'architecture, de la physique ou des mathématiques. C'est le cas de Xenakis [Xenakis I., 1963] qui, faisant appel au calcul de probabilité pour réaliser des effets de masse orchestrale, fut l'un des premiers à intégrer l'ordinateur comme outil d'aide à la composition, ou encore de Hiller et Isaacson qui composèrent en 1957, la *Suite Illiac pour quatuor à cordes,* une pièce calculée par ordinateur, fondée sur le hasard et les lois du contrepoint de première espèce.

Parallèlement, l'évolution de la lutherie instrumentale vers l'électronique, de la musique qui s'approprie de nouveaux espaces sonores, amènent aux premiers travaux sur la synthèse par ordinateur, conduits en 1959 par Max Mattews, dans le cadre des laboratoires Bell[1] [Mattews Max. V., 1969]. La synthèse sonore constituera rapidement le plus vaste et le plus important des domaines de recherche de l'informatique musicale.

C'est à partir de 1983, avec la publication de la norme MIDI[2] (Musical Instrument Digital Interface) et le développement de l'ordinateur personnel, que l'informatique musicale sort du domaine strict de la recherche pour se diversifier par des applications commerciales vers un public plus large. Aujourd'hui, l'informatique est présente à toutes les étapes de la production musicale : des langages d'aide à la composition à l'acoustique des salles, en passant par la synthèse sonore, le traitement du son numérique, la diffusion du son ou encore l'édition de partitions.

En retour, cette évolution ouvre de nouvelles voies de recherche : la multiplication des applications a rapidement mis en évidence l'impossibilité de les faire cohabiter sur une même machine, par manque de convention sur le partage des ressources critiques. Par ailleurs, leur développement sur des architectures et des systèmes d'exploitation standards a mis à jour les carences de ces systèmes relativement au domaine musical.

---

[1] Ils débouchent sur plusieurs générations de programmes (Music4 en 1962, Music5) permettant de faire de la synthèse sonore directe.

[2] La norme MIDI définit un brochage (DIN 5 broches), une vitesse de transmission (31,25 KBauds) et un format de messages permettant la communication entre instruments électroniques. Ses spécifications sont disponibles auprès de l' International MIDI Association (IMA).

Nous verrons donc pourquoi des architectures logicielles particulières sont nécessaires au domaine de l'informatique musicale. Nous les aborderons sous l'angle particulier du temps et de la communication. Nous présenterons ensuite *MidiShare* [Orlarey Y., Lequay H., 1989], une architecture logicielle développée à GRAME et dédiée au domaine musical, ainsi que son extension aux réseaux locaux à travers l'implémentation d'un flot de données temps réel sur Ethernet. Enfin, nous verrons comment sur la base de ce système, l'organisation de la communication et de la collaboration entre applications, peuvent conduire à des solutions originales.

2 Le temps et la communication dans les applications musicales.

Le système d'exploitation d'un ordinateur fournit les services logiciels indispensables à son utilisation. Il organise le partage et la gestion des ressources nécessaires au fonctionnement des applications. Il facilite le travail du programmeur en lui proposant une "machine virtuelle" plus homogène et plus simple à programmer que la "machine réelle" sous-jacente. Il assure une plus grande pérennité aux applications en les isolant autant que possible des particularismes matériels.

Ces rôles essentiels du système d'exploitation ne sont malheureusement que très partiellement remplis vis-à-vis des applications musicales. La nécessité de recourir à des architectures matérielles et logicielles spécialisées est apparue en informatique musicale dès l'instant où il s'est agit de communiquer avec des moyens non-standards et avec des contraintes de temps-réel importantes.

C'est sous l'angle particulier de ces contraintes que nous allons examiner les besoins des applications musicales. Elles ne sont évidemment pas exhaustives mais permettent de mieux comprendre la spécificité de ces applications en regard de leur environnement.

2.1 Ordonnancement des événements et des tâches.

Les applications musicales doivent ordonnancer et restituer les événements musicaux avec une précision temporelle qui soit au moins égale au pouvoir séparateur de l'oreille. Elles doivent également travailler en collaboration avec d'autres outils multimédias et satisfaire à différents codes de synchronisation, le plus couramment utilisé étant le code SMPTE.

Dans les faits, une précision de l'ordre de la milliseconde est au moins nécessaire tant pour dater les événements entrants que pour l'exécution des différentes tâches qui permettront de les traiter et de les restituer. Les processus correspondants seront mis en oeuvre par le système d'exploitation de la machine. Ils nécessitent à chaque activation, un basculement de contexte qui se révèle souvent pénalisant pour les performances du système si ce basculement est trop fréquent. Par ailleurs, la conception même des systèmes multitâches de type Unix ne permet pas à un processus d'être éligible à une date donnée. Enfin pour les systèmes qui fournissent un accès à la gestion du temps, la précision obtenue est souvent largement insuffisante.

2.2 Gestion des ressources mémoire.

La gestion mémoire d'une application utilise généralement le gestionnaire du système d'exploitation. Celui-ci permet d'allouer et de libérer dynamiquement des blocs mémoire de longueur arbitraire. Il effectue également le compactage de la mémoire quand c'est nécessaire (dans le cas d'une fragmentation excessive par exemple). Mais de tels gestionnaires ne sont pas adaptés au temps réel pour les raisons suivantes :

- le temps d'allocation d'un bloc mémoire n'est pas déterministe. Il peut être très long si la mémoire a besoin d'être compactée.
- les gestionnaires mémoire traditionnels ne sont pas réentrants. Toute routine qui s'effectue sous interruption ne pourra donc pas y faire appel sans risquer de désorganiser totalement l'espace mémoire.

Enfin, et d'une manière générale, pour chaque bloc alloué, il y a un surcoût de plusieurs octets. Ce surcoût rend l'allocation de petits blocs peu rentable(sur Macintosh System 7 par exemple, il est de 16 octets). De tels gestionnaires sont donc peu adaptés à la taille des messages MIDI (3 octets d'une manière générale).

2.3 La communication temps réel.

Le besoin en communication des applications musicales ne leur est pas particulier : en effet, la communication inter-application s'impose aujourd'hui comme une nécessité pour l'ensemble des applications qui vont vers une spécialisation toujours croissante et sont de ce fait, contraintes à la collaboration pour toutes les tâches qui échappent à leurs compétences.

Dans les systèmes d'exploitation traditionnels, la communication inter-application s'effectue soit par effets de bord (communication par fichiers, couper / copier / coller), soit par l'établissement de conduits de communication (pipe) entre différentes applications. Ce second moyen de communication, souple et puissant, présente néanmoins l'inconvénient d'être implémenté sur des systèmes inadaptés soit au multitâche, soit au temps réel, alors que le domaine musical impose deux contraintes particulières :

- le temps de communication doit être déterministe et le plus faible possible.
- les relations temporelles entre les différents objets transmis doivent être précisément conservées.

Récemment, des systèmes tels que Système 7 sur Macintosh ont mis en oeuvre des moyens de communication plus sophistiqués, ce sont les 'Apple Events', peu compatibles cependant avec les impératifs de temps réel :

- le temps de transmission d'un 'Apple Event' est élevé : 10 ms minimum et 20 ms en moyenne sur un Quadra 700, hors charge en temps CPU.
- le temps de transmission d'un Apple Event n'est pas déterministe, il augmente avec l'occupation du temps CPU.

3 MidiShare : une architecture logicielle pour la musique.

Dans ce contexte, définir une architecture logicielle pour la musique permet de fournir aux applications musicales, un cadre de communication et de collaboration homogène et durable ainsi que des mécanismes simples et puissants pour la gestion du temps. MidiShare a été développé dans ce sens : c'est un système d'exploitation musical, MIDI, multitâches, temps-réel, qui vient se mettre en complément du système d'exploitation de la machine hôte.

3.1 Le modèle conceptuel.

MidiShare est basé sur un modèle client/serveur. Il est composé de six éléments principaux (figure 1) : un gestionnaire de mémoire, un gestionnaire du temps associé à un synchroniseur, un gestionnaire des tâches, un gestionnaire de la communication, un ordonnanceur et des drivers d'entrée / sortie MIDI.

Le gestionnaire de la communication :

La communication est basée sur des événements de haut niveau, plus simples à manipuler que des paquets d'octets MIDI. Le gestionnaire de communication distribue ces événements, reçus de l'ordonnanceur et des drivers d'entrée, aux applications clientes et aux drivers de sortie, conformément aux connexions courantes.

L'ordonnanceur :

Il délivre les événements et les tâches à leurs dates d'échéance. Il est possible d'y insérer des événements ou des tâches pour une date quelconque dans le futur. L'algorithme d'ordonnancement [Orlarey Y., 1990] assure un coût de traitement borné et constant par événement, quelque soit la charge de l'ordonnanceur.

Le gestionnaire de mémoire :

Il fournit des primitives simples pour l'allocation, la copie et la libération des événements MidiShare, avec un coût d'allocation borné et faible. Il a été conçu pour une utilisation temps réel, sous interruption.

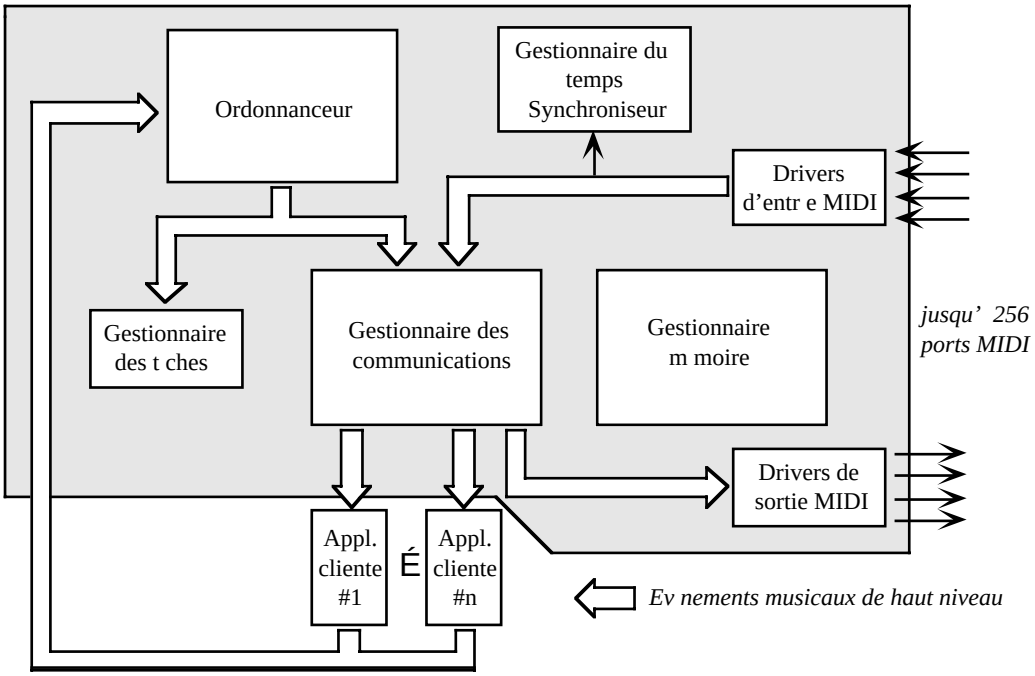

figure 1 : modèle conceptuel de MidiShare.

Le gestionnaire du temps et le synchroniseur :

Le gestionnaire de temps a pour charge de maintenir la date courante du système. Il a une précision d'une milliseconde. Le synchroniseur supporte le code SMPTE de manière transparente pour les applications clientes.

Le gestionnaire de tâches :

Il a la responsabilité d'appeler, pour le compte des applications clientes, les tâches qui sont délivrées à leur date d'échéance par l'ordonnanceur.

Les drivers d'entrée / sortie MIDI :

Ils sont chargés des communications MIDI sur les ports d'entrée / sortie. Ils peuvent gérer jusqu'à 256 ports.

3.2 Des mécanismes sophistiqués pour la gestion du temps.

Les applications musicales requièrent une gestion sophistiquée du temps : il ne leur suffit pas de pouvoir réagir en temps réel aux événements entrants, elles doivent également planifier le futur avec une grande précision (dans le cas d'une application qui crée un écho des événements entrants par exemple). MidiShare fournit plusieurs mécanismes pour la gestion du temps : ce sont les alarmes et les appels de fonction différés dans le temps.

3.2.1 Les alarmes

Les alarmes de réception :

Chaque application cliente peut installer une alarme de réception qui lui permettra de traiter les événements reçus en temps réel. Cette alarme sera appelée par MidiShare pour le compte de l'application, au moment où le gestionnaire de communication lui aura distribué un ou plusieurs événements.

Les alarmes de contexte :

Chaque application cliente peut installer une alarme de contexte qui lui permettra de réagir aux modifications de son environnement en temps réel. MidiShare appelle toutes les alarmes de contexte positionnées pour le compte des applications propriétaires, au moment où le changement

de contexte s'effectue. Ces changements de contexte peuvent concerner par exemple la modification de l'état des ports d'entre / sortie ou encore des connexions entre applications.

3.2.2 Les appels de fonction différés dans le temps.

Les tâches temps réel :

Une application peut effectuer des appels de fonction différés dans le temps en créant une tâche et en confiant son exécution à MidiShare pour une date donnée. Lorsque cette tâche arrive à échéance, elle est rendue par l'ordonnanceur et exécutée par le gestionnaire de tâches pour le compte de l'application qui l'a créée. Cet appel de fonction se fait en temps réel, sous interruption.

Les tâches temps différé :

Un mécanisme analogue permet d'effectuer des appels de fonctions en temps différé. A la différence des tâches temps réel, une tâche temps différé, rendue par l'ordonnanceur, n'est pas exécutée par le gestionnaire de tâches mais placée dans la liste des tâches à exécuter de l'application. Son appel est alors de la responsabilité de l'application qui l'a créée. Ce mécanisme permet entre autres d'ordonnancer des tâches qui ne peuvent être exécutées sous interruption.

3.3 Communication et collaboration.

En informatique, la notion de communication prend une importance croissante. Elle s'est notamment développé avec l'apparition des architectures parallèles et distribuées pour en permettre une meilleure exploitation. Mais les concepts de communication, présents dans d'autres domaines scientifiques, fournissent également des outils puissants pour la modélisation de phénomènes naturels ou le développement d'applications complexes.

Le modèle de communication de MidiShare est connexionniste. Chaque application peut être considérée comme un agent possédant une entrée et une sortie, recevant un flot d'événements entrants et produisant un flot d'événements sortants. Chacun de ces agents peut être librement connecté aux autres, de manière à constituer un réseau (figure 2) dont la complexité n'est limitée que par le nombre possible d'applications simultanées.

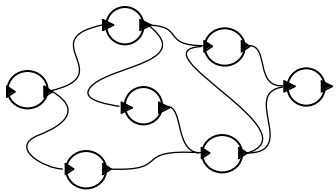

figure 2 : un réseau d'applications

Ce schéma de communication peut être dynamiquement configuré par les applications ou l'utilisateur. Il est le fondement de la richesse du système et de son extensibilité : d'emblée, toute nouvelle application se place dans un cadre collaboratif et vient enrichir les possibilités offertes par celles déjà existantes.

4 L'organisation de la communication

L'extension de ce schéma de communication aux réseaux locaux (développé plus loin) pose le problème de l'organisation de la communication. En effet, le nombre d'applications susceptibles d'être interconnectées étant multiplié par le nombre de machines présentes sur le réseau, les tâches de connexions se complexifient et peuvent devenir rapidement très fastidieuses. Pour simplifier ces tâches, nous introduisons un agent particulier sous forme de dossier de communication [Orlarey Y., 1991], analogue aux dossiers d'un système hiérarchique de fichiers.

4.1 La communication hiérarchique

De même que les agents élémentaires (qui ne peuvent contenir d'autres agents) que sont les applications MidiShare actuelles, les dossiers de communication possèdent une entrée et une sortie et peuvent donc être connectés à d'autres agents. Chaque agent est contenu par un dossier de

communication : le dossier père, hiérarchiquement supérieur. La seule exception concerne le dossier racine, qui représente tout le système et n'est contenu dans aucun autre.
Les agents élémentaires sont des applications musicales. Mais on peut également imaginer que des applications complexes puissent s'organiser en terme d'agents regroupés à l'intérieur d'un même dossier, avec un mécanisme permettant de cacher le contenu de ce dossier de manière à ce qu'il apparaisse à l'utilisateur comme un agent élémentaire. Ce système d'organisation hiérarchique peut donc s'appliquer tant à la communication entre agents qu'à l'architecture interne des applications.

4.2 Règles de connexion

Un agent peut être connecté à lui-même, à d'autres agents à l'intérieur du même dossier , à l'entrée et à la sortie de son dossier père. Une connexion ne peut pas traverser les frontières d'un dossier : un agent ne peut donc pas être connecté directement à un agent qui n'est pas situé dans le même dossier que lui.

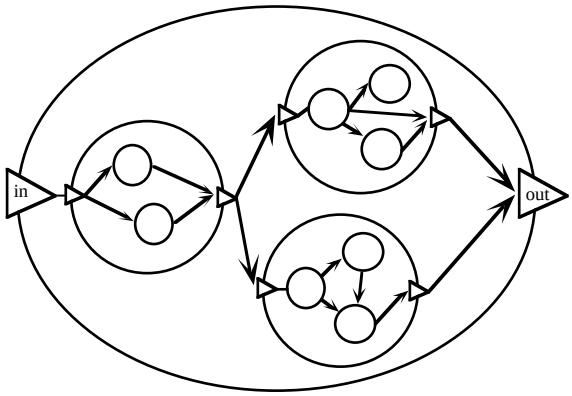

figure 6 : hiérarchie des connexions.

L'interconnexion des agents se fait sous forme d'une hiérarchie de graphes dont les noeuds peuvent former des sous-graphes, etc... (figure 6) La création ou la suppression des dossiers peuvent se faire dynamiquement, de même pour le déplacement d'un agent d'un dossier à un autre, l'établissement ou la suppression des connexions entre les agents.

4.3 L'algorithme de distribution des messages

Ce modèle de communication pose deux problèmes, liés à la possibilité de connecter directement l'entrée d'un dossier à sa sortie (figure 7) :

- la création de boucles infinies.
- la multiplicité des chemins d'un agent à un autre.

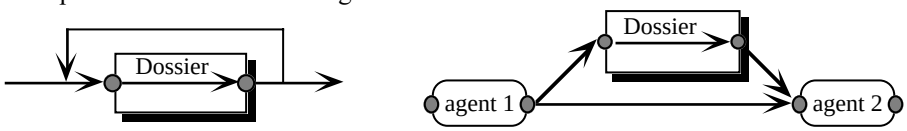

figure 7 : a) boucle infinie b) chemins multiples

Ces problèmes peuvent être résolus grâce à un algorithme de distribution des messages qui consiste à marquer le chemin parcouru par un message, de manière à ce qu'il ne soit parcouru qu'une seule fois et que le message ne soit distribué qu'une seule fois au même destinataire. Chaque message émis par une application porte un numéro de série unique pour l'ensemble du système. Quand un message traverse une connexion et quand il est distribué à son destinataire, il laisse sa marque sous la forme de ce numéro de série. Si une connexion porte sa marque le message ne la traversera pas, de même si le destinataire porte sa marque, il ne lui sera pas distribué.

4.4 La simplification des tâches de connexion

Il en résulte une simplification des tâches de connexion, illustrée par les deux exemples suivants :

- l'interconnexion totale de n agents entre eux, qui nécessite normalement n^2 connexions, peut s'effectuer avec seulement 2n+1 connexions en utilisant un dossier (figure 8).
- la connexion de n agents à m agents, qui nécessite normalement n*m connexions, peut s'effectuer avec n+m+1 connexions en utilisant un dossier vide (figure 9).

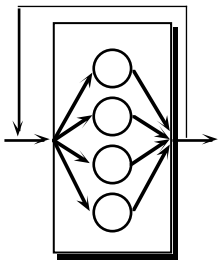

figure 8:
interconnexion de 4 agents entre eux gr ce un dossier.

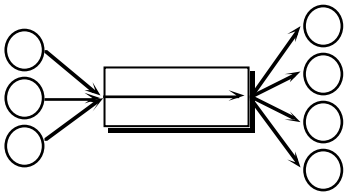

figure 9 :
interconnexion de 3 x 4 agents en utilisant un dossier vide.

De plus, les graphes peuvent être entièrement déconnectés en agissant sur une seule connexion : celle qui relie la sortie du dossier à son entrée dans le premier exemple, celle qui relie l'entrée du dossier à sa sortie dans le deuxième exemple.

5 Extension de la communication aux réseaux locaux : un modèle pour Ethernet

L'extension du schéma de communication de MidiShare aux réseaux locaux de type Ethernet [Fober D., 1994] ouvre la voie à la collaboration entre machines différentes. Cependant, Ethernet pose deux problèmes pour la transmission temps-réel d'événements MidiShare :

- le temps d'accès au réseau n'est pas déterministe.
- Ethernet est efficace pour transmettre occasionnellement de gros paquets de données, mais cette efficacité diminue très rapidement quand la taille des paquets diminue et quand leur fréquence augmente, or les événements MidiShare sont en grande majorité des cellules de 16 octets.

A travers les mécanismes d'Ethernet, nous verrons les raisons de ces limitations, les solutions adoptées et enfin les résultats obtenus.

5.1 Ethernet.

Ethernet est basé sur le protocole CSMA/CD (Carrier Sense, Multiple Access / Collision Detection) définit par la norme IEEE 802.3 [Millet, 1986]. Ce protocole permet à toutes les machines d'accéder librement au réseau et de transmettre un message sans avoir besoin d'y être invitées.

5.1.1 Mécanismes de transmission

Le réseau est caractérisé par son débit et la vitesse de transmission de son support. Chaque station peut simultanément lire et écrire sur le bus. La transmission d'un paquet peut être détectée par les transitions d'état correspondantes, on parle alors de la présence d'une porteuse. Une fois détectée, celle-ci interdit toute autre transmission sur le réseau.

Transmission d'un paquet :

Une machine qui désire émettre commencera par écouter le bus. Si elle détecte une porteuse, elle attendra sa disparition pour commencer à émettre. Le paquet est alors envoyé sous forme binaire et se propage sur le bus dans les deux directions à partir du point de connexion.

Détection de collision :

Si plusieurs machines sont en attente de transmission, lorsque la porteuse disparaît, elles vont commencer à émettre simultanément et leurs paquets vont interférer sur le bus : c'est ce que l'on nomme une collision. Les machines peuvent détecter une collision en comparant la valeur du bit qu'elles écrivent sur le bus avec celle qu'elles lisent : une différence indique que leur paquet a été

endommagé. Pour être sûr de détecter toutes les collisions, la durée minimale de transmission d'un paquet doit être au moins égale au double du temps de parcours bout à bout du réseau.

Résolution des collisions :

Lorsque plusieurs machines détectent une collision, elles commencent par la renforcer pour être sûres que toutes les machines vont la détecter, elles arrêtent ensuite leur transmission et calculent un délai de retransmission aléatoire en fonction notamment d'un historique récent des collisions.

5.1.2 Efficacité du réseau

Nous définissons l'efficacité du réseau comme le rapport entre les périodes d'utilisation productives et le temps total d'utilisation du réseau. Les périodes productives sont les périodes pendant lesquelles la transmission s'effectue avec succès. Le temps total d'utilisation du réseau représente la somme des périodes productives et des périodes improductives qui sont les périodes de compétition entre les machines pour l'acquisition de l'usage exclusif du bus et donc dédiées à la résolution des collisions.
En raison des mécanismes de transmission, quand la fréquence d'émission des paquets augmente, les collisions augmentent, les temps improductifs d'utilisation du bus également, et donc l'efficacité du réseau diminue.

La transmission d'un paquet consiste en 2 périodes (figure 3) :

- une période de contention : égale au double du temps de parcours bout à bout du réseau et pendant laquelle une collision peut survenir.
- une période productive : qui succède à la période de contention et pendant laquelle l'émetteur est sûr de pouvoir effectuer sa transmission correctement. En effet, au bout de la période de contention, toutes les machines sur le réseau peuvent entendre la porteuse. S'il n'y a pas eu jusque là de collision, aucune station sur le réseau ne tentera d'émettre un paquet et la transmission se déroulera correctement.

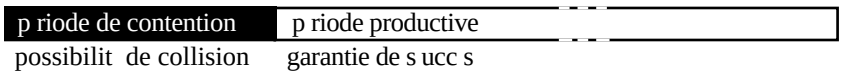

figure 3 : périodes de transmission d'un paquet.

Quand la taille des paquets augmente, la durée des périodes productives augmente en conséquence et donc l'efficacité du réseau également.

5.1.3 Indétermination du temps d'acquisition du réseau

Du au protocole CSMA/CD, le temps d'acquisition du réseau est indéterminé. Dans les faits, avant qu'une station puisse émettre avec succès peuvent se succéder 2 périodes qui sont :

- le temps d'attente de libération du réseau s'il y a présence d'une porteuse au moment où la station est prête à émettre.
- le temps de résolution des collisions s'il y a compétition entre plusieurs machines pour acquérir l'usage exclusif du bus.

5.2 Les solutions.

Deux mécanismes dans la transmission d'événements MidiShare permettent de palier à ces limitations :

- le groupage d'événements qui permet de conserver l'efficacité du réseau.
- un délai de compensation des temps d'accès au réseau, qui permet la restitution correcte des événements transmis.

5.2.1 Groupage d'événements

Nous avons introduit un premier délai dans la transmission : le délai de groupage des événements. Il agit de manière similaire aux levées postales : il permet de regrouper les événements à transmettre, d'augmenter la taille des paquets qui transitent sur le bus et de diminuer leur fréquence. De cette manière nous avons pu conserver l'efficacité du réseau et faire cohabiter les services temps réel et les services traditionnels du réseau. Dans l'implémentation actuelle, ce délai est de 50 ms.

5.2.2 Délai de compensation des temps d'accès

Lors de la transmission d'événements, la variation des temps d'accès au réseau produit lors de leur restitution, une déformation de leurs relations temporelles. Si par exemple des événements sont régulièrement envoyés toutes les 20 ms (figure 4 a), à réception, ils seront restitués avec un délai qui variera selon le temps d'accès au réseau (figure 4 b).
Nous avons donc introduit un second délai : le délai de compensation des temps d'accès au réseau qui représente le temps maximal autorisé pour l'acquisition du réseau et pour une restitution temporelle correcte (figure 4 c). Ce délai est systématiquement ajouté à la date de restitution de tous les événements transmis. Des travaux précédents [Hutchinson, Merabti, 1987] ont montré que, pour une certaine charge du réseau, les temps d'acquisition étaient bornés. La valeur du délai de compensation signifie donc que la restitution temporelle des événements sera correcte jusqu'à une certaine charge du réseau. Dans l'implémentation actuelle, la valeur retenue est de 30 ms.

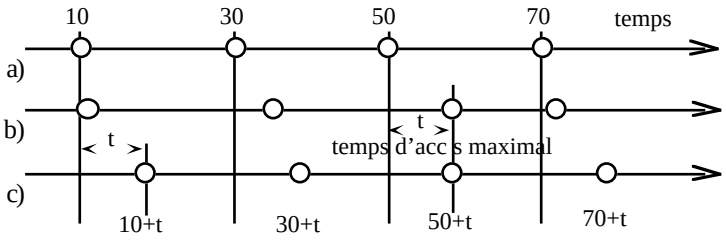

figure 4 : conservation des relation temporelles

Le délai de groupage et le délai de compensation introduisent donc un délai total de 80 ms dans la transmission des événements. Il est le résultat d'un compromis entre le délai introduit par le transport et le maintien de l'efficacité du réseau.

5.2.3 Protocoles

Les paquets transmis sur le réseau contiennent soit des événements MidiShare, soit des messages pour la gestion du réseau.

Les messages de gestion du réseau sont au nombre de 4 et servent à maintenir la liste des machines présentes et à les identifier.

Les événements sont transmis sans acquittements et donc sans procédure de récupération en cas d'erreur de transmission. Néanmoins, en consultant l'en-tête des paquets, un récepteur peut connaître le nombre exact de paquets éventuellement perdus. L'absence d'acquittement permet d'éviter la surcharge du réseau et contribue à conserver son efficacité. Cependant, si un événement nécessite au moins 5 paquets consécutifs pour être émis, sa transmission se fera selon un protocole différent (figure 5) qui a pour but de fiabiliser la transmission et d'assurer la synchronisation entre émetteur et récepteur.

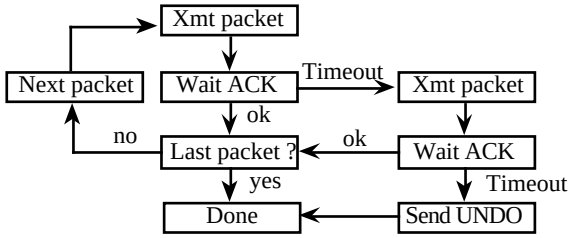

figure 5 : protocole de transmission des grands événements.

5.3 Les résultats.

Les mesures de performances suivantes ont été effectuées avec 7 machines Ethernet. Les résultats obtenus indiquent que 25 utilisateurs au moins peuvent bénéficier des services de transmission temps-réel avec un débit équivalent au plein débit MIDI. Par ailleurs, pour le même débit et jusqu'à 5 utilisateurs, nous n'avons pas pu mesurer de variation dans les services autres du réseau (ces mesures ont été faites sur le temps de transfert de fichiers allant de 100 Kb à 1 Mb).

6 L'organisation de la collaboration

Les principes de communication de MidiShare impliquent des possibilités de collaboration étendues pour les applications musicales. A l'heure actuelle, cette collaboration est mise en oeuvre explicitement par l'utilisateur lorsqu'il établit son schéma de communication et surtout, lorsqu'il définit la qualité des services de chaque application en réglant leurs différents paramètres. Dans le cadre d'une architecture qui permet une collaboration systématique et surtout si les applications sont réparties sur un réseau, cette manière de procéder pose problème : il peut devenir rapidement fastidieux de mettre en oeuvre et de régler toutes les applications engagées dans une même collaboration, cela peut être rendu impossible par leur localisation sur des machines différentes. Pour pouvoir se développer, la collaboration entre applications suppose donc une organisation minimale qui permette de l'automatiser.

6.1 Les règles générales de la collaboration.

L'automatisation de la collaboration requiert que les applications soient contrôlables de manière externe et selon des modalités homogènes. Pour ce faire, une application doit identifier chacun des services qu'elle souhaite rendre contrôlables par un message unique et non ambigu. Elle émettra ces messages lors de chaque modification de ces services et inversement, à réception d'un de ces messages, elle effectuera la modification correspondante.

Étant donné que ces applications traitent du domaine musical, l'implémentation de ces messages sera faite à l'aide de messages MIDI standard (contrôleurs ou system exclusives). Dans ces conditions, le contrôle d'une application pourra se faire par le biais de la communication à l'intérieur d'une même machine, sur un réseau ou encore via MIDI.

L'utilisation de messages MIDI rapproche ces principes de collaboration de ce qui existe déjà du point de vue contrôle d'équipements externes. C'est donc de manière homogène et avec les mêmes outils, que l'ensemble des ressources mises en oeuvre dans un projet musical, pourront être gérées.

6.2 La macro-construction d'applications.

Les outils de communication du système permettent de gérer jusqu'à l'architecture interne d'une application (vue comme assemblage de composants communicants réunis à l'intérieur d'un même dossier). Le fait de pouvoir assembler et 'coller' ensemble des applications de manière externe, c'est à dire sans que ce soit explicitement définit au moment de leur développement, ouvre la voie à d'autres modes de conception et d'utilisation des application musicales. Elles peuvent être vues alors comme des agents communicants, inter-agissants et capables de produire un comportement émergeant. La macro-construction d'applications restitue ainsi une dimension de programmabilité de la machine à l'utilisateur.

Mais au-delà du problème de l'automatisation de la collaboration, se pose maintenant celui des outils qui permettent d'effectuer de manière simple cette macro-construction d'applications. C'est la problématique des langages pour utilisateurs finaux [Canfield et al., 1994] qui est soulevée, dans la mesure où le public des applications de l'informatique musicale est avant tout un public de musiciens et non d'informaticiens. La solution à laquelle nous travaillons, consiste en une application capable d'interroger les applications en cours de fonctionnement, afin d'obtenir une description de leur comportement, puis de "calculer" le comportement résultant et enfin de "synthétiser" une application équivalente.

7 Conclusion

La nécessité de disposer d'architectures spécialisées pour les applications musicales s'est imposée à l'ensemble des développeurs, tant et si bien que d'une situation où les applications étaient en concurrence pour disposer des ressources de la machine, nous sommes passés à une situation où ce sont maintenant des systèmes d'exploitation musicaux différents qui entrent en conflit pour ces mêmes ressources, de sorte que nous avons été amenés à développer un système nommé 'Open Share', qui permet la cohabitation de différentes architectures .

Historiquement l'année 1989 a vu l'apparition simultanée de 3 systèmes :

- MidiShare, pour Macintosh et Atari
- MIDI Manager, développé par Apple pour Macintosh.
- M-ROS, de Steinberg Soft und Hardware GmbH, pour Atari.

Depuis, ce sont d'autres systèmes encore qui sont apparus, soit par portage sur de nouvelles plates-formes matérielles, soit en se cumulant à ceux existants sur une même machine. Sur Macintosh par exemple, en plus des systèmes déjà cités, sont successivement apparus : M-ROS (portage, fin 1990), OMS (Opcode Systems, Inc.) et FreeMidi (Mark of the Unicorn, Inc.).

Chacun de ces systèmes possède ses particularités, leur seul réel point commun est de permettre le partage des ports d'entrée / sortie de la machine. Le partage du temps, l'accès à des tâches temps réel sont plus ou moins pris en compte, de même pour les différents aspects de la communication. D'autres domaines sont parfois abordés : c'est le cas par exemple de la gestion des ressources matérielles d'un studio.

Quoiqu'il en soit, le problème soulevé au début de cet article n'a été que partiellement résolu dans la mesure où il s'est en fait déplacé des applications vers les systèmes d'exploitation musicaux. Néanmoins, outre la nécessité de ces architectures, nous avons pu montrer qu'elles pouvaient simplifier grandement la gestion du temps et offrir un support cohérent et durable, de communication et de collaboration entre les applications. Par sa disponibilité, sa modularité et son extensibilité, le système que forme MidiShare et ses applications, constitue aujourd'hui un système ouvert au sens définit par Hewitt [Hewitt 1985].

Enfin, cette problématique des architectures logicielles pour la musique est en mouvement permanent pour avoir ses fondements sur une base matérielle en constante évolution, mais également parce qu'en se développant, elle touche à des questions qui intéressent d'autres domaines, scientifiques ou informatiques, en leur apportant l'éclairage particulier de l'informatique musicale.

Références

Anderson D. , Kuivila R., 1986 Accurately Timed Generation of Discrete Musical Events. Computer Music Journal Vol 10 N°3, MIT Press, pp.48-56.

Canfield Smith D., Cypher Allen, Spoher Jim, 1994, KIDSIM: Programming Agents Without a Programming Language, Communications of the ACM, July 1994, Vol.37, N°7, pp. 55-67.

Dannenberg R. B., 1896, A Real Time Scheduler/Dispatcher. Proceedings of the ICMC 1986. ICMA, San Francisco.

Fober D., 1994, Real time, musical data flow on Ethernet and MidiShare software architecture. Proceedings of the ICMC, 1994, ICMA, San Francisco.

Hewitt C.E., 1985, The Challenge of Open Systems. Byte Vol. 10 N°4, April 1985, pp. 223-242.

Hutchinson D., Merabti M., 1987, Ethernet for real-time applications. IEEE proceedings. Part E. Computers and digital techniques, Vol 134, N°1, pp.47-53.

Mattews Max.V., 1969 The Technology of Computer Music. M.I.T. Press, 1969

Mattews Max.V., Pierce John R. 1989 Current Directions in Computer Music Research. M.I.T. Press, 1989

Maxemchuk N.F., 1982, A variation on CSMA/CD that yelds movable TDM slots in integrated voice/data local networks. Bell Syst. Tech. J. 61, pp.1527-1550.

Metcalfe Robert M., Boggs David R., 1983, Ethernet : Distributed Packet Switching for Local Computer Networks. Communication of the ACM, Vol 26, N°1, pp.90-95.

Millet P., 1986, Transmission et réseaux locaux, architecture I.E.E.E. 802. Masson 1986

Orlarey Y., Lequay H., 1989, MidiShare : a Real Time multi-tasks software module for Midi applications. Proceedings of the ICMC, 1989, ICMA, San Francisco.

Orlarey Y., 1990 An Efficient Scheduling Algorithm for Real-Time Musical Systems. Proceedings of the ICMC, 1990, ICMA, San Francisco.

Orlarey Y., 1991, Hierarchical Real Time Inter application Communications. Proceedings of the ICMC, 1991, ICMA, San Francisco.

Pierce John.R., 1984, Le son musical, Ed. Belin, collection Pour la science.

Tannenbaum A.S., 1992, Modern Operating Systems, Prentice-Hall International Editions.

Xenakis I., 1963, Musiques formelles. La revue musicale, n°253-254, Paris, 1963