UTILISATION DU CANAL AUDIO DANS LE COLLECTICIEL

Christophe Tronche
LRI (CNRS URA 410)
Bât. 490, Faculté d'Orsay
91405 ORSAY CEDEX
+ 33 (1) 69 41 69 10
tronche@lri.fr

Abstract

Les applications multi-utilisateurs nécessitent des protocoles pour maintenir la consistance entre les différentes instances des interfaces vues par les différents utilisateurs. Certains de ces protocoles sont imposés par la machine, d'autres sont des protocoles sociaux. Le *canal audio* est encore sous-exploité malgré la richesse de ses possibilités. Nous décrivons dans cet article la classification que nous avons été amené à faire parmi les différents traitements possibles sur ce canal, ainsi que l'expérimentation que nous avons mené pour mettre en œuvre ces traitements.

Mots clés : application multi-utilisateur, audio, collecticiel, groupware.

Il ne suffit pas pour écrire une application collecticielle de dupliquer une interface autant de fois qu'il y a d'utilisateurs et de permettre aux différentes instances de l'interface de communiquer. Pour permettre véritablement d'assister le travail coopératif, l'application collecticielle digne de ce nom doit fournir aux utilisateurs en session des outils leur permettant de structurer leur travail, et de structurer le fonctionnement de leur groupe, afin d'aboutir à la réalisation commune de leur tâche. Ceci nécessite que les différents membres du groupes puissent communiquer, non seulement à l'intérieur de la réalisation de la tâche, mais également à propos de la réalisation de la tâche, c'est à dire qu'ils doivent pouvoir disposer d'un niveau de *méta-communication*.

1 Protocoles matériels et sociaux

Les outils qui permettent cette structuration du groupe sont des protocoles de communication. On les divise généralement en protocoles *matériels* et protocoles *sociaux* [EGR91]. Les premiers sont imposés ou pour le moins assistés par la machine. Les seconds sont des conventions que se donnent les utilisateurs. Un exemple est la prise de contrôle. Dans certaines applications collecticielles, un seul utilisateur peut utiliser l'application à un instant donné (les autres le regardent) [SFB+87]. Lorsqu'un autre utilisateur veut la main, il la demande. Si le protocole est matériel, le système peut par exemple mettre le demandeur dans une file d'attente et signaler à celui qui détient la main que quelqu'un d'autre la veut. Lorsqu'il rendra (explicitement) la main, l'utilisateur suivant dans la file d'attente pourra la prendre. Dans le cas d'un protocole social, le demandeur dit (oralement, ou à l'aide d'un sous-système de communication tel que le *talk* d'Unix) : "laisse-moi faire", celui qui détient la main répond "d'accord, vas-y". En cas d'opération conflictuelle, par exemple si les deux utilisateurs déplacent le même objet vers deux positions distinctes, le résultat est plus ou moins prévisible, mais très probablement inconsistant.

Cet exemple illustre la différence qui existe souvent entre un protocole matériel et un protocole social. D'un coté, on a la garantie de la consistance et de l'autre l'aspect naturel et convivial d'une conversation. On peut dire que le premier est le protocole du programmeur: la machine régle les conflits mais au prix d'une remontée au niveau de l'utilisateur de contraintes liées à l'implémentation. Le second protocole pourrait en revanche être qualifié de protocole de l'utilisateur.

Les protocoles sociaux présentent un certain nombre d'avantages : De Koven et *al.* [DKR90] notent qu'ils favorisent le développement par les utilisateurs de leurs propres protocoles d'utilisation de l'outil collecticiel qui leur est confié. Les protocoles sociaux sont donc éminemment souples, et peuvent s'adapter à la grande variété des situations rencontrées en matière de travail collaboratif, contrairement aux protocoles implémentés qui sont figés par nature. Les protocoles sociaux peuvent même s'adapter au fur et à mesure de l'évolution du travail, par exemple quand un des membres du groupe en prend implicitement la direction en prenant des initiatives, ce que les autres acceptent en le laissant faire, ou rejettent. Enfin, même si un protocole mécanique existe, il peut nécessiter un protocole social. MPCAL par exemple [LS83], est un agenda multi-utilisateurs. Si les réunions prévues sont conflictuelles, les utilisateurs dont les emplois du temps se chevauchent sont notifiés et doivent régler seuls le problème, par exemple par téléphone.

2 Le canal audio comme support de la méta-communication

Il faut donc fournir aux utilisateurs un média riche, supportant une communication suffisamment subtile pour leur permettre de gérer les protocoles sociaux. La voix (et les sons) forment une modalité intéressante à plus d'un titre (cf aussi [IHM93]). La voix est globalement plus rapide que le texte écrit ou dactylographié (il est plus rapide de dire "je travaille sur cette portion du dessin" que de le dactylographier). Elle est créditée d'un pouvoir d'expression plus important et la production et l'interprétation de la voix nécessiterait une charge cognitive plus faible que la production et l'interprétation de texte écrit ([CFK91] compare la voix et le texte dans un contexte d'annotation de documents). C'est une modalité qui est relativement peu exploitée jusqu'ici, malgré la présence maintenant fréquente de périphériques d'entrée–sortie audio sur les micro-ordinateurs et les stations de travail, comme le note Hindus et *al.* [HS92]. Elle est donc "libre", dans le sens où elle sert peu, pour le moment, à envoyer des commandes à l'interface contrairement aux canaux visuels, en sortie, et kinestésiques (clavier et souris) en entrée, qui sont, eux, intensivement exploités.

Il est évidemment possible de transmettre la voix par un canal séparé du canal de donnée, par exemple d'utiliser un téléphone ou un interphone pour le transport de la voix et un canal numérique pour le transport des données (ceci est fait par exemple dans [DC91]). Nous pensons cependant que l'étape suivante consiste à traiter, stocker et transmettre le son numériquement, en raison de la profusion de traitements possibles et de la relative facilité avec laquelle ils peuvent être effectués. D'un point de vue technique, les sons entrants dans l'ordinateur sont numérisés. Il sont ensuite traités, transportés et éventuellement traités une nouvelle fois numériquement sur la machine distante. Enfin, ils sont restitués à l'utilisateur.

Pour une introduction aux techniques du son numérique, on pourra consulter [Poh]. Nous travaillons pour le moment en 8 khz, 8 bits linéaire, ce qui donne une qualité raisonnable, au moins égale à celle du téléphone. Le réseau est un Ethernet, contrôlé en utilisant des protocoles IP; il ne constitue donc pas (en terme de débit) un goulet d'étranglement [1]. Il est à noter que nous utilisons le son à la fois comme modalité *d'entrée* (au moins sur le canal de méta-communication), et de *sortie*.

Dans la suite de cet article, nous nous intéresserons essentiellement aux conversation, qui sont par nature synchrone, et donc au traitement temps-réel du signal. Ceci n'exclut pas des traitements asynchrones (potentiellement plus riche [HS92]): il est possible de mémoriser les informations audio pour les réutiliser dans un cadre asynchrone, par exemple préparer un résumé vocal d'une session de travail. Dans le cas synchrone, toutes les opérations doivent être effectuées "à la volée", ce qui impose certaines limitations sur ce qui est réalisable, limitations découlant de la puissance de calcul des machines utilisées. Suivant le matériel spécialisé disponible sur la machine, la *restitution* des sons peut mobiliser plus ou moins de puissance de calcul, qui est alors indisponible pour les *traitements* sonores. Il faut garder à l'esprit que ce traitement est permanent : même si les quantités de données par unité de temps sont faibles, c'est un flux continu qu'il faut gérer dynamiquement. Or le reste de l'interface, le noyau fonctionnel, et possiblement d'autres applications, ont évidemment besoin d'une part de la puissance de calcul. Il résulte de tout ceci que tous les traitements ne sont pas possibles. Nous allons, pour quelques-uns d'entre-eux, examiner dans la section suivante leurs besoins respectifs en puissance de calcul.

3 Modalité et médialité

Nous distinguons entre les traitements qui permettent l'extraction de signaux encodés dans les échantillons, comme par exemple la reconnaissance de la parole, et ceux qui font des calculs systématiques sur des groupes d'échantillons, comme le multiplexage de canaux. Les premiers concernent la *modalité*, c'est-à-dire l'interprétation du sens du message transmis par le média. Par analogie, nous qualifions les seconds de traitements de la *médialité* car ils concernent seulement le codage du message.

Deux exemples de traitements de la modalité sont la reconnaissance de la parole et la reconnaissance du locuteur. Ces traitements sont très coûteux sur un ordinateur à usage général, en terme de temps de calcul; à tel point qu'il est impensable de le réaliser en temps réel avec les machines actuelles si ces machines doivent continuer à assurer le reste de leurs tâches. Le traitement de la médialité est en revanche plus abordable. Par exemple, le multiplexage de canaux nécessite pour n canaux n sommes et une division; la détection de silence nécessite une moyenne temporelle. Certains traitements peuvent cependant être plus gourmands, comme le filtrage et la transformée de Fourier suivie d'une re-synthèse.

Si on demande à l'interface de pouvoir utiliser la parole comme modalité d'entrée, les problèmes engendrés deviennent très complexes : l'ordinateur reçoit en entrée un signal composite qui est un mélange de commandes destinées à l'interface ou au noyau fonctionnel (via une carte de reconnaissance vocale), de messages adressés aux autres utilisateurs de la

[1] Une liaison audio non traitée utilise 64 Kbps de bande passante au niveau application (28 octets de plus par paquet pour IP et UDP, plus les en-têtes et les fins de message Ethernet) à comparer au débit théorique de 10 Mbps d'Ethernet et quelques Mbps pour un réseau normalement chargé.

session, et possiblement de bruits parasites additionnels. Une possibilité serait pour la carte de reconnaissance de reconnaître tout ce qu'elle peut et de rejeter le reste dans le canal de méta-communication. Mais ceci nécessite pour cette carte une très haute technicité, alors que celles qui sont disponibles sont déjà utilisées à la limite de leurs possibilités.

Dans la suite, nous allons nous concentrer sur la médialité afin de voir comment quelques traitements simples sur les sons peuvent être utilisés dans le cadre des interfaces multi-utilisateurs.

4 Différents traitements possibles et exemples d'utilisation

Certains traitements sont absolument indispensables, d'autres relèvent davantage du "gadget" et il en est de même des applications. Il est bien sûr encore difficile de les distinguer et seule l'expérimentation permettra de faire la différence. Dans cette section nous allons parcourir une liste de traitements possibles et d'exemples d'applications associées.

Multiplexage. Le multiplexage consiste à prendre deux canaux audio et à les additionner pour n'en former qu'un seul. Ceci permet de "réunir" numériquement des sons séparés, comme s'ils avaient été joués dans la même pièce. Nous considérons ceci comme le traitement de base, pour deux raisons. Tout d'abord, les ordinateurs disposent de peu de canaux physiques (généralement 1 à 4), et nous voulons disposer d'une palette de possibilités beaucoup plus riche. Multiplexer permet de disposer d'un nombre de canaux sonores logiques beaucoup plus importants que le nombre de canaux physiques, et ceci, de façon "portable", c'est à dire indépendante de la plateforme matérielle. La deuxième raison est d'autoriser la téléconférence. Dans sa forme la plus simple, la téléconférence se présente sous la forme d'un effet "salle de réunion" (tout le monde entend simultanément ce que tout le monde dit) qui est à la base du support vocal du travail collaboratif. Si l'application multi-utilisateurs sous-jacente s'intéresse à un nombre relativement important d'utilisateurs, disons plus de quatre, l'effet salle de réunion tourne rapidement à l'effet "hall de gare", qui rend tout travail impraticable. Pour que le groupe puisse continuer à travailler dans de bonnes conditions, il est nécessaire d'être sélectif et que chaque utilisateur en session puisse sélectionner l'ensemble de ceux avec qui il veut communiquer. Si l'interface possède une connaissance suffisante de l'application, elle peut modifier les groupes au fur et à mesure de l'exécution de la tâche. Un avantage décisif du traitement numérique du son apparaît clairement ici : pour réussir la même chose en analogique, il faut disposer d'un central téléphonique piloté par un ordinateur accessible par l'application.

Contrôle automatique du gain Une bonne application collecticielle doit se faire oublier. Il n'est pas question de demander aux utilisateurs de régler en permanence le volume de leurs micros. L'interface peut renormaliser le volume sonore pour le maintenir à peu près constant. Ceci est particulièrement important si l'utilisateur n'est pas assis à sa table de travail, mais debout devant un tableau, comme c'est par exemple le cas avec les expériences de "video whiteboard" menées à Xerox [TM91]. Il est évidemment possible d'avoir un micro-cravate, mais il est beaucoup plus agréable d'avoir un micro approximativement orienté vers le locuteur et d'introduire ensuite dans le signal sonore les corrections nécessaires ([HS92] cite plusieurs situation indiquant la difficulté de capturer du son en situation réelle).

Détection des silences Une application multi-utilisateur reçoit un flot d'évènements de chaque utilisateur. Mémoriser ces évènements pour construire un historique de la collaboration et pouvoir les utiliser a été identifié comme étant une possibilité importante d'une interface multi-utilisateurs [Spe93]. Il est également important que chaque utilisateur puisse annoter une partie de son travail, aux fins d'archivage par exemple, ce qui permet de l'utiliser dans des applications asynchrones (courrier vocal). Le flot d'échantillons brut est difficile à exploiter à cette fin. La détection des silences permet de transformer le flot d'échantillons en informations semi-structurées, sémantiquement plus intéressantes [HS92].

Accélération Il est possible de jouer des successions d'échantillons en accéléré. Ceci peut être utile par exemple au cours d'une réunion où un utilisateur prend des notes vocales: l'accélération permet de repasser rapidement les dernières notes prises ou d'effectuer une recherche rapide. Il est également possible de ralentir le passage du flot d'échantillons, même si l'intérêt est moins évident.

Effets spéciaux Les premières applications multi-utilisateurs mettaient en scène le bureau ou la salle de réunion habituels, mais de nouvelles métaphores d'interaction multi-utilisateurs se développent maintenant, comme par exemple celle implémentée par DIVE [FSBC93], qui s'appuie sur la description d'un monde à trois dimensions dans lequel les utilisateurs sont des personnages entourée d'une "aura", ayant un "focus" d'attention. Dans de tels univers, le son peut-être exploité pour indiquer par exemple le degré de conscience, ou le degré de présence d'autres utilisateurs. [BBC+93] suggère des variations dans le volume sonore, mais d'autres effets sont possibles, comme l'écho (ou la réverbération), ou une modification de la voix pour la "teinter" et permettre d'identifier plus facilement des utilisateurs-clé de la session. L'identification des autres utilisateurs peut être améliorée et la perception du volume renforcée grâce à la spatialisation, qui donne la sensation que le son vient d'une direction particulière.

5 Implémentation

Nous avons implémenté toutes les fonctions énumérées ci-dessus, à l'exception de la spatialisation (plus délicate en termes d'implémentation). Le fonctionnement global du système est symbolisé sur la figure 1. Il est organisé en une batterie de serveurs, technique qui a prouvé sa valeur (cf par exemple [Aro92]). Les serveurs sont reliées par des bandes de transports. Il y a trois catégories de serveurs : ceux qui fournissent des sons (comme un microphone, un enregistrement, un générateur interne), ceux qui les transforment et enfin le haut-parleur qui restitue le son à l'utilisateur.

Les flots générés de manière interne permettent d'occuper l'espace sonore avec des outils comme une horloge parlante, de la même façon que l'espace graphique de l'écran est occupé par d'autres outils comme l'horloge ou la calculatrice. Un nombre croissant de travaux soulignent l'intérêt du son en sortie dans les interfaces, suivant en cela des pionniers comme Bill Gaver [GST91, Gav89].

La plupart des constructeurs livrent maintenant du matériel et du logiciel pour le son, avec malheureusement encore quelques idiosyncrétismes. Nous avons fait nos essais dans un environemment hétérogène constitué de stations de travail Unix et de Macintoshs. Leur liaison peut poser des problèmes; par exemple, le serveur audio livré avec les stations de travail Hewlett-Packard ne peut délivrer du son en continu que sur une connection TCP, ce qui rend le multicast difficile. Le temps d'aller-retour typique d'un paquet UDP sur un réseau Ethernet normalement chargé est de l'ordre de 3 ms (10-15 ms en traversant un routeur), ce qui n'est pas réellement limitatif. En revanche, les machines peuvent imposer des longueurs minimales d'échantillon en deçà desquelles le rendu est de mauvaise qualité,

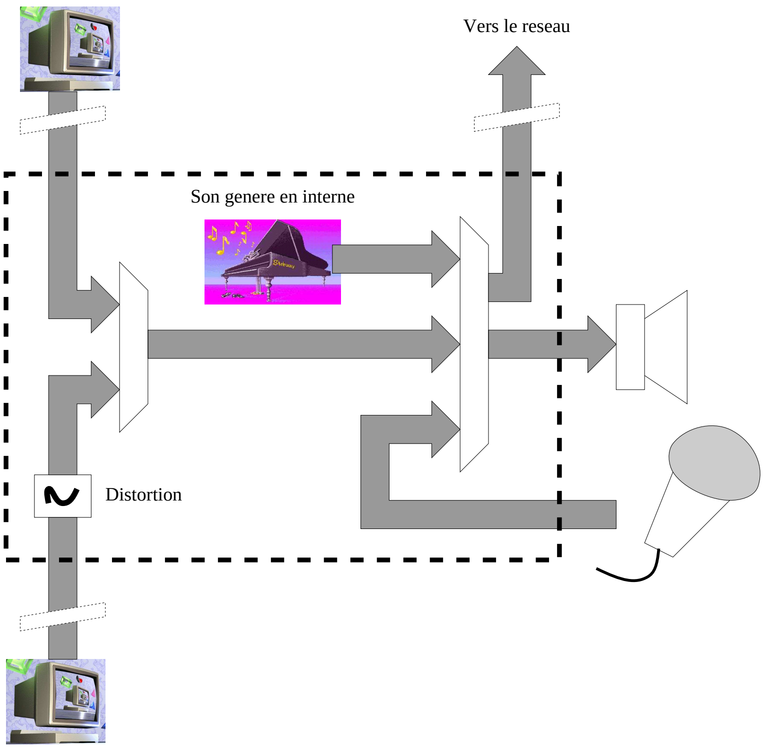

Figure 1: Architecture

ce qui pose des problèmes. Par exemple, avec des blocs d'échantillons de 2K, le délai séparant l'émission d'un son de sa réception est de l'ordre de 1/8 à 1/4 de seconde, ce qui donne l'impression d'une conversation transatlantique et est clairement la limite. Une bonne taille pour les blocs est sans doute à 8 kHz de l'ordre de 256 échantillons (31 ms). A titre d'exemple, l'expérience décrite dans [CD92] utilise avec un certain succès des paquets de 180 échantillons (22.5 ms).

6 Conclusion

Nous avons décrit notre expérience en matière de traitement numérique du son et montré comment ce traitement peut être mis en œuvre dans le cadre des applications multi-utilisateurs. En particulier, nous avons évoqué comment des traitements simples (c'est à dire réalisables en temps réel) pouvaient être utilisés dans ces mêmes interfaces.

L'exploitation du canal audio n'en est qu'à ses premiers balbutiements. Il est difficile de prévoir quelles seront les applications qui se révèleront réellement utiles et quelles sont celles qui seront finalement jugées sans intérêt. C'est pourquoi il est important de lancer des expérimentations afin de déterminer ce qui est techniquement faisable, et ce qui est en fin de compte utile à l'utilisateur.

References

[Aro92] Barry Arons. Tools for building asynchronous servers to support speech and audio applications. In *UIST'92*, pages 71–78. ACM, ACM Press, November 1992.

[BBC+93] Steve Benford, Adrian Bullock, Neil Cook, Paul Harvey, Rob Ingram, and Ok-Ki Lee. A spatial model of cooperation for virtual worlds. In *L'interface des mondes réels et virtuels*, pages 445—456, 1993. Acte des journées informatique'93.

[CD92] Stephen Casner and Stephen Deering. First IETF internet audiocast. *ACM SIGCOMM Computer Communications Review*, 3(22), July 1992.

[CFK91] Barbara L. Chalfonte, Robert S. Fish, and Robert E. Kraut. Expressive richness: a comparison of speech and text as media for revision. In Scott P. Robertson, editor, *Reaching through technologies*, pages 21–26. ACM, May 1991. Proc. of the ACM conference on human factors in computing systems, New Orleans, Louisiana, April 27–May 2, 1991.

[DC91] A. Derycke and P. Croisy. La conception d'une architecture d'accueil pour des conférences multimédia en temps réel: premier bilan. In *Troisièmes journées sur l'ingénierie des interfaces homme-machine*, pages 115–122, September 1991.

[DKR90] C. De Koven and T. Radhakrishnan. An experiment in distributed group problem solving. In Simon J. Gibbs and A.A. Verrijn-Stuart, editors, *Multi-User Interfaces and Applications*, pages 61–76. IFIP WG 8.4, North-Holland, September 1990.

[EGR91] Clarence A. Ellis, Simon J. Gibbs, and Gail L. Rein. Groupware: some issues and experiences. *Communications of the ACM*, 34(1):38–58, January 1991.

[FSBC93] Lennart E. Fahlén, Olov Ståhl, Charles Grant Brown, and Christer Carlsson. A space based model for user interaction in shared synthetic environments. In Stacey Ashlund, Kevin Mullet, Austin Henderson, Erik Hollnagel, and Ted White, editors, *Bridges between worlds*, pages 43—48. ACM, ACM Press, April 1993. Proceedings of the INTERCHI'93 Conference.

[Gav89] William W. Gaver. The sonicfinder: An interface that uses auditory icons. *Human-Computer Interaction*, 4(1), 1989.

[GST91] William W. Gaver, Randall B. Smith, and O'Shea Tim. Effective sounds in complex systems: the arkola simulation. In Scott P. Robertson, editor, *Reaching through technologies*, pages 85–90. ACM, May 1991. Proc. of the ACM conference on human factors in computing systems, New Orleans, Louisiana, April 27–May 2, 1991.

[HS92] Debby Hindus and Chris Schmandt. Ubiquitous audio: Capturing spontaneous collaboration. In *Sharing perspectives*, pages 210–217. ACM SIGCHI, ACM Press, New York, November 1992.

[IHM93] Ihm'92, travaux des ateliers. travaux de l'atelier collecticiel, 1993.

[LS83] Barbara Liskov and Robert Scheifler. Guardians and actions: Linguistic support for robust, distributed, programs. *ACM Transactions on Programming Languages and Systems*, 5(3):381–404, July 1983.

[Poh] Ken C. Pohlmann. *Principles of Digital Audio.* Audio Library. Howard W. Sams & Company.

[SFB+87] M. Stefik, G. Foster, D. G. Bobrow, K. Kenneth, S. Lanning, and L. Suchman. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1):32–47, January 1987.

[Spe93] Michael Spenke. From undo to multi-user applications — the demo. In Stacey Ashlund, Kevin Mullet, Austin Henderson, Erik Hollnagel, and Ted White, editors, *Bridges between worlds*, pages 468–469. ACM, ACM Press, April 1993. Proceedings of the INTERCHI'93 Conference.

[TM91] John C. Tang and Scott L. Minneman. Videowhiteboard : Video shadows to support remote collaboration. In Scott P. Robertson, editor, *Reaching through technologies*, pages 315–322. ACM, May 1991. Proc. of the ACM conference on human factors in computing systems, New Orleans, Louisiana, April 27–May 2, 1991.