A mes parents

Remerciements

Je tiens tout d'abord à remercier Jean MENEZ, Professeur à l'Université de Nice - Sophia Antipolis, qui me fait l'honneur de présider mon jury de thèse.

J'exprime ma profonde gratitude à Christian HUITEMA, Directeur de Recherche à l'INRIA Sophia Antipolis, qui a mis à ma disposition tous les moyens nécessaires en m'accueillant au sein du projet RODEO, et qui a été mon directeur de thèse. Ses conseils judicieux ont permis une bonne orientation de mes recherches dans l'univers passionnant du multimédia.

Je voudrais remercier Peter Kirstein, Professeur à L'*University College London* (UCL) du Royaume Uni, pour le temps si précieux qu'il m'a accordé en acceptant d'être rapporteur de ma thèse.

J'aimerais également remercier Pierre Rolin, Professeur à TELECOM Bretagne pour avoir accepté d'être rapporteur et pour ses remarques pertinentes qui m'ont permis de compléter et d'éclaircir certains points importants de ce manuscrit.

Je remercie Philippe Dax, Directeur d'Etudes de l'ENST Paris, qui s'est intéressé à mes travaux et a bien voulu participer au jury.

Mes remerciements les plus vifs vont tout particulièrement à Jean BOLOT et Walid DABBOUS, Chargés de Recherche à l'INRIA Sophia Antipolis, qui ont dirigé mes travaux. La confiance qu'il ont su me témoigner, le temps qu'ils m'ont consacré, et les conseils qu'il m'ont donnés tout au long de mes travaux, m'ont largement aidé à mener à bien cette thèse.

Je n'oublie pas Christophe DIOT, Chargé de Recherche et Jean Noel PIC, Boursier de recherche dans le projet RODEO pour la patience et l'intérêt qu'ils ont eus à relire et à annoter mon manuscrit.

J'adresse également mes chaleureux remerciements à tous les membres du projet RODEO ainsi qu'aux participants du projet Européen MICE. Leurs conseils, leur amitié et leur bonne humeur m'ont beaucoup encouragé durant mes travaux.

Enfin, je tiens à remercier tous ceux qui m'ont soutenu, et plus particulièrement mes parents, Isabelle et mes amis.

Thierry TURLETTI

Table des matières

Table des figures

Chapitre 1

Introduction

La vidéoconférence sur ordinateur est devenu aujourd'hui une réalité. L'avènement des vidéoconférences sur l'Internet est très récent. Le premier essai grand public remonte à mars 1992: il s'agissait de la diffusion sur l'Internet d'une réunion de l'IETF [1] qui se tenait à San Diego [Casner92]. Depuis quelques années déjà, on assiste à une prolifération d'applications qui utilisent des médias de nature différente (parole, son, image, vidéo, texte, hyper-texte), regroupées sous l'appellation d'applications multimédia. Ces applications ont fait leur apparition sur de nombreux systèmes d'ordinateurs, les stations de travail classiques, et même les micro-ordinateurs [Schooler91].

Toutes ces nouvelles applications ont pu voir le jour grâce à l'essor des techniques numériques en télécommunication, à la spectaculaire montée en puissance des machines et aux progrès réalisés dans les techniques de compression de données. En effet, les applications multimédia manipulent des données de grande taille et leur transmission sur le réseau nécessite une bande passante élevée [Scotton93]. Les techniques de compression de parole et surtout d'images sont donc capitales pour ce genre d'application. Grâce aux progrès de la micro-électronique, on peut maintenant effectuer ces coûteuses opérations de compression de données sur des stations de travail courantes.

Pour que les communications puissent se faire à l'échelle planétaire, il a été indispensable de mettre en place une standardisation. Des normes de compression de données ont été développées pour des médias de nature différente ainsi que pour des applications différentes [Tawbi93]. Par exemple, la norme JPEG (*Joint Photographic*

1. L'IETF *Internet Engineering Task Force* regroupe une communauté de chercheurs, ingénieurs et commerciaux qui sont tous concernés par l'évolution de l'architecture de l'Internet.

Experts Group) [Wallace91] a été mise au point pour la compression d'images fixes et les normes H.261 [H261] [Guichard90] [Liou91] et MPEG (*Moving Picture Experts Group*) [MPEG] [Pancha92] pour la compression d'images animées. Le développement des systèmes multimédia distribués évolue très rapidement car il représente un enjeu de taille pour le monde de l'industrie, et de manière plus générale pour le monde de l'enseignement, des affaires, et de la politique. On peut présager que l'introduction du multimédia dans les stations de travail reliées via des réseaux à commutation de paquets classiques de type Internet aura dans un avenir proche autant d'impact que l'introduction de l'ordinateur personnel dans les foyers.

La vidéoconférence fait appel à un grand nombre de domaines dont les principaux sont:

- le codage des données (audio et vidéo),
- le contrôle de la transmission sur le réseau,
- la synchronisation entre les flots audio et vidéo,
- le contrôle de la session entre les participants de la conférence,
- le contrôle de la prise de la parole,
- la sécurité et l'authentification des données (audio, vidéo et session).

Dans mes travaux de thèse, j'ai abordé plus particulièrement les problèmes de codage et de contrôle de la transmission vidéo sur l'Internet. Les autres problèmes ont été laissés volontairement de côté dans la suite du manuscrit. A l'époque ou j'ai débuté mes travaux, c'est-à-dire à la fin 1991, on ne parlait pas encore de multimédia sur l'Internet et le MBONE n'était pas encore né. Nous projetions alors d'étudier des algorithmes de contrôle de congestion sur des applications de vidéoconférence réelles. Aucune application de vidéoconférence sur l'Internet n'étant alors disponible, nous avons préféré élaborer nous-mêmes un logiciel de vidéoconférence plutôt que d'utiliser les codecs cablés du commerce. Ces derniers étaient trop coûteux et n'étaient pas aussi souples qu'un codec logiciel, facilement paramétrable. En premier lieu, nous avons choisi une norme de compression vidéo car nos travaux ne consistaient pas en l'élaboration d'un nouveau schéma de compression de données. A l'époque, la norme MPEG 1 n'était pas encore finalisée (elle le fut dans le courant de l'année 1993) et avait pour but le stockage de l'information sur des supports numériques avec des débits de transferts pouvant aller jusqu'à 1.5 Mbps. Nous avons

opté pour le standard de compression H.261 car il a été spécialement conçu pour les application de vidéoconférence et de visiophonie. De plus, il représentait l'état de l'art des techniques de compression de données à bas débit [Guichard91].

Nos contributions ont porté sur deux points principaux. Le premier a été de montrer par notre application que la technologie actuelle permettait de réaliser des codecs vidéo en logiciel et donc à moindre coût. Le deuxième point a été de montrer qu'il était possible de transmettre de la vidéo de bonne qualité sur l'Internet actuel, et donc sans réservation de ressources. Ces travaux ont été mis en œuvre dans notre logiciel de vidéoconférence IVS (*INRIA Videoconferencing System*). Ce logiciel a été développé dans le cadre du projet Européen MICE[2] [Handley93], [Kirstein93]. Il est actuellement utilisé par un grand nombre de sites universitaires de recherche et par des industriels [Pagani93] [Sasse94]. Il est également utilisé pour retransmettre des conférences et des séminaires sur Internet (e.g. IETF de novembre 1992 à Washington, conférence JENC de mai 93 en Norvège et les séminaires MICE hebdomadaires).

Le standard de compression H.261 a été conçu à l'origine pour être utilisé sur le Réseau Numérique à Intégration de Service (RNIS). RNIS est un réseau à commutation de circuit qui a pour caractéristique d'offrir des services avec ressources garanties (e.g. bande passante de 64 kbps allouée tout le temps de la connexion). Pour pouvoir utiliser efficacement ce standard de compression vidéo sur des réseaux à commutation de paquets comme l'Internet, nous avons dû lui apporter un certain nombre de modifications. En effet, l'Internet n'offre actuellement qu'un service connu sous le nom du "meilleur effort possible" (ou "*best effort*"), c'est-à-dire que le réseau cherche à acheminer le plus rapidement possible les paquets entre une source et une destination, mais sans garantie sur le délai ou le taux de perte. En conséquence, ce type de réseau peut être sujet à des phénomènes néfastes comme la congestion qui provoque la perte de paquets. Principalement, nous avons développé les algorithmes suivants:

- découpage en paquets du flot de bits H.261,
- contrôle d'erreurs contre la perte des paquets,
- contrôle de débit pour codeur H.261,

2. Le projet MICE a pour objectif d'offrir aux chercheurs Européens des outils multimédia pour leur permettre de travailler ensemble et à distance depuis leur station de travail.

– contrôle de congestion pour adapter le débit du codeur à la bande passante disponible du réseau.

Nos résultats ont en partie contribué à ce que la communauté scientifique porte un nouveau regard sur les applications "temps réel" de type vidéoconférence, et s'interroger sur les services et les garanties qui sont vraiment nécessaires pour les applications multimédia. Auparavant, la vidéoconférence était associée implicitement à l'idée de codecs cablés, reliés entre eux par des canaux à bande passante fixe et garantie. L'approche était alors d'adapter le service réseau aux besoins supposés de l'application. Notre approche est différente car c'est à l'inverse l'application qui s'adapte aux conditions du réseau. L'impact sur l'architecture des réseaux est donc important.

Enfin la dernière contribution porte sur la place et l'utilisation des mécanismes que nous avons proposés dans l'évolution probable de l'architecture de l'Internet. En particulier, nous avons montré que ces mécanismes gagneront en efficacité lorsque les routeurs intégreront de nouvelles disciplines de service.

Le manuscrit est organisé en six chapitres. Le chapitre 2 décrit la recommandation H.261 ainsi que sa mise en œuvre en logiciel. On présente dans les chapitres 3 à 6 les problèmes associés à la transmission vidéo H.261 sur réseaux en paquets en général, l'Internet en particulier. De façon plus spécifique, le chapitre 3 décrit comment découper un flot vidéo H.261 en paquets qui pourront ensuite être transmis sur l'Internet. Le chapitre 4 décrit des mécanismes qui permettent de corriger les erreurs de transmission. Le chapitre 5 présente des méthodes de contrôle de débit du codeur vidéo. Le chapitre 6 décrit les méthodes de contrôle de congestion qui peuvent être appliquées dans l'Internet. Il présente aussi l'impact de nos résultats sur l'architecture des réseaux pour la transmission efficace d'applications multimédia. La conclusion est présentée au chapitre 7. L'annexe A donne les principes de la transmission multipoint ainsi qu'une vue d'ensemble du MBONE. L'annexe B contient une évaluation de l'algorithme d'estimation de l'état du réseau. L'annexe C donne une brève description des standards de compression vidéo actuels et présente succintement les divers logiciels de vidéoconférence que l'on peut trouver aujourd'hui dans le domaine public. Enfin, un glossaire de quelques termes clefs est donné à la fin du mémoire.

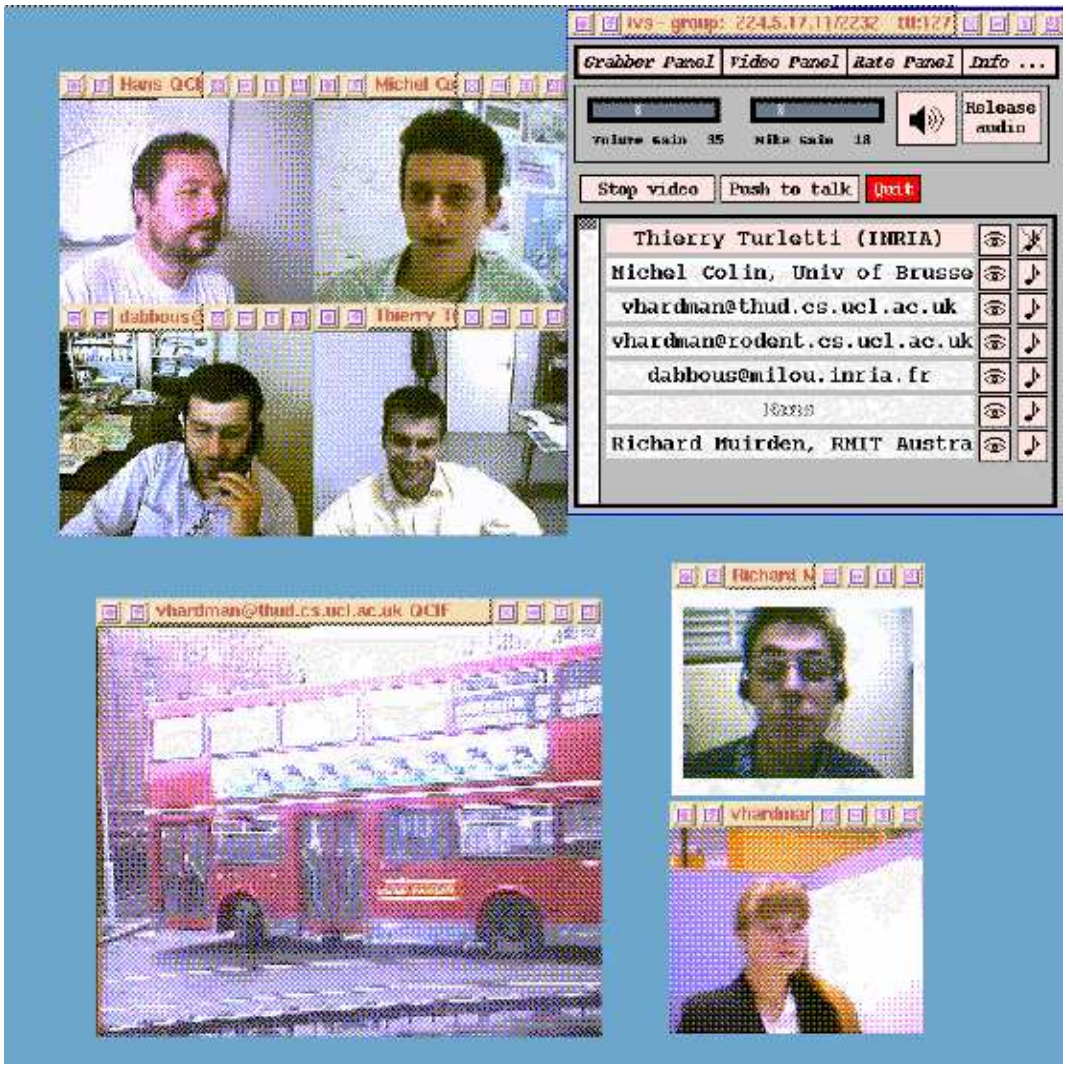


FIG. 1.1 - *Exemple de vidéoconférence entre partenaires du projet MICE*

Chapitre 2

Mise en oeuvre d'un codec logiciel H.261

Ce chapitre donne une description du standard de compression vidéo H.261 ainsi que les choix de conception que nous avons effectués lors de sa mise en œuvre logicielle. La première section expose les différentes techniques de compression de données qui sont utilisées. La deuxième section donne l'architecture de notre codec logiciel ainsi qu'une discussion sur nos choix de conception. Enfin, la troisième section introduit les chapitres suivants en mettant en évidence la nécessité d'y apporter des modifications pour pouvoir l'utiliser efficacement sur l'Internet.

2.1 Le standard de compression H.261

2.1.1 Généralités

Avec l'introduction du Réseau Numérique à Intégration de Services (RNIS), le CCITT[1] a mis en place en décembre 1984 le groupe d'étude SG-XV chargé de l'étude d'une norme de compression vidéo pour des applications de visiophonie.

A l'origine, le but était de concevoir un codage vidéo pour des transmissions à des débits de $m \times 384kbps$, $(m \in [1,5])$. Par la suite, un standard à $n \times 64kbps$, $(n \in [1,5])$ a été étudié. Finalement, en décembre 1990 a été proposé et approuvé le standard "Recommandation H.261: Codec pour Services Audiovisuels à $p \times 64kbps$, $(p \in [1,30])$", qui permet de couvrir entièrement les possibilités de transmission du RNIS,

1. Comité Consultatif International Télégraphique et Téléphonique, aujourd'hui remplacé par l'Union Internationale des Télécommunications (UIT).

de l'accès de base [2] à l'accès primaire [3]. La recommandation H.261 donne les spécifications minimales pour assurer la compatibilité entre un codeur et un décodeur. Il s'agit plutôt d'une spécification de décodeur car une grande liberté est laissée au concepteur pour la réalisation du codeur. En effet, ne sont spécifiés que la nature et le format des signaux transmis et non la manière de les obtenir: par exemple, le choix du mode INTER/INTRA ou les procédés de régulation de débit sont laissés à l'initiative du concepteur.

2.1.2 Format des images

Dans cette section nous introduisons les formats d'image ainsi que les représentations du signal vidéo qui sont utilisés par le standard de compression vidéo H.261. Nous donnons ensuite une description de la structure hiérarchique du codage.

Une application de vidéoconférence nécessite une définition d'image plus fine, et une visualisation de plus grande taille qu'une application de type visiophone. En outre, la définition du format d'image doit prendre en compte les problèmes de compatibilité entre les régions du monde qui possèdent des standards de télévision différents (e.g. Europe: 625 lignes/50Hz, Amérique du Nord et Japon: 525 lignes/60Hz). Pour résoudre ces problèmes, le CCITT a adopté le Format Intermédiaire Commun (CIF). Un format réduit au quart (QCIF) est aussi défini et constitue la base minimum que chaque codec conforme à la norme H.261 doit être capable de traiter. Le choix du format CIF ou QCIF dépend de la capacité du canal disponible. Avec $p = 1$ ou 2, c'est-à-dire pour un débit de 64 kbps ou 128 kbps, le format QCIF est généralement utilisé pour des applications de visiophonie. En prenant $p \geq 6$, le format CIF est plus approprié à des applications de vidéoconférence.

Trois composantes de couleur rouge (R), vert (V) et bleu (B) suffisent pour reconstituer un signal vidéo couleur. Ces trois composantes peuvent être transformées par matriçage en trois variables (Y,U,V). La luminance (Y) est représentative de l'image en niveaux de gris et les chrominances rouge et bleue ($U = C_R = R - Y$ et $V = C_B = B - Y$) correspondent à la coloration de l'image. Dans le nouveau format YUV, le signal vidéo peut se comprimer grâce aux deux propriétés suivantes: les éléments Y, U et V sont moins corrélés que les éléments R, V et B et la plus grande partie de l'information est contenue dans la luminance Y. Pour des images CIF ou

2. L'accès de base est constitué de deux canaux numériques à 64 kbps (canaux B) et d'un canal numérique de signalisation à 16kbps (canal D).

3. L'accès primaire est constitué de 30 canaux B pour un canal D.

QCIF, la conversion du format RVB au format YUV est notée 2:1. Cela signifie que si un pixel est codé sur 24 bits dans le format RGB, il est codé sur deux fois moins de bits dans le format YUV. Par exemple, un carré de quatre pixels dans le format RGB, est codé par quatre coefficients de luminance Y et deux coefficients de chrominance (U, V). En effet, les composantes de chrominances sont sous-échantillonnées dans l'espace, horizontalement et verticalement, avec un rapport 2. Si l'on suppose que chaque coefficient est représenté par 8 bits, le nombre total de bits utilisés par ce carré de pixels vaut dans le format RVB $4 * 3 * 8 = 96$ bits et $6 * 8 = 48$ après conversion dans le format YUV.

Les composantes Y, U, V et les codes représentant leurs valeurs échantillonnées sont définies dans la Recommandation 601 du CCIR [CCIR601]. La structure d'échantillonnage est orthogonale; ce qui signifie que la position des coefficients (ou échantillons) de chrominance est telle que les frontières de blocs sont les mêmes que pour la luminance (figure 2.1). Les techniques de traitement numérique d'images

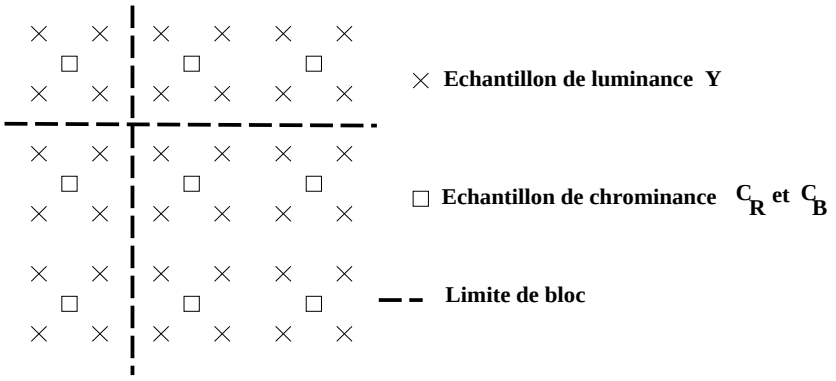


FIG. 2.1 - *Positionnement des coefficients de luminance et de chrominance*

conduisent à échantillonner le signal vidéo dans ses trois dimensions, en retenant les grandeurs caractéristiques suivantes:

- dans la direction horizontale, le nombre de points par ligne ou pixels: pour une image CIF, il est de 352 pour la luminance et de 176 pour les chrominances C_R et C_B.
- dans la direction verticale, le nombre de lignes par image: pour une image CIF, il est de 288 pour la luminance et de 144 pour les chrominances C_R et C_B.
- dans le temps, le nombre d'images par seconde: le codeur de source agit sur

des images non entrelacées apparaissant à une fréquence de 30 images par seconde.

Pour une image QCIF, le nombre de coefficients pour chaque composante est réduit d'un facteur 2 dans chaque direction (horizontale et verticale).

Les images CIF et QCIF sont organisées selon une structure hiérarchique en quatre couches: la couche Image (I), la couche Groupe de Blocs (GOB), la couche Macro-Bloc (MB) et enfin la couche Bloc (B) (voir figure 2.2). La couche image

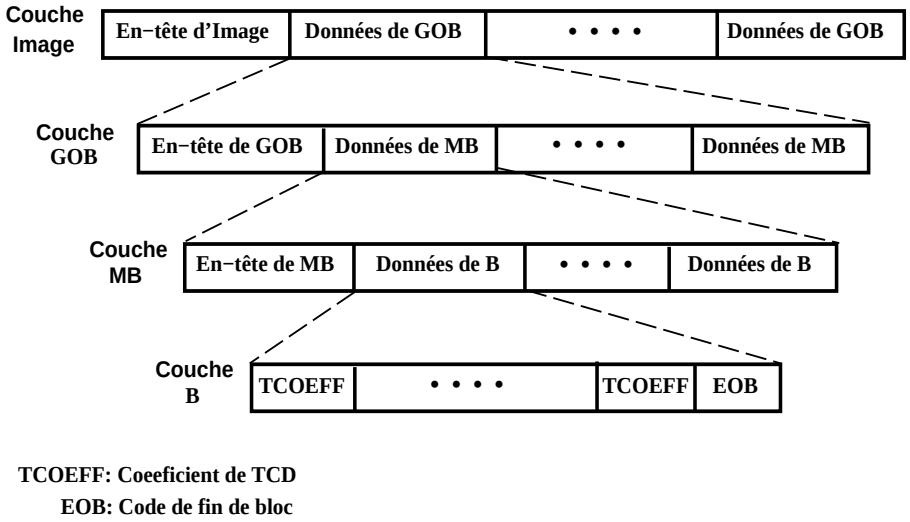


FIG. 2.2 - *Structure hiérarchique d'une image H.261*

est constituée d'une en-tête d'image suivi des données de GOB. L'en-tête d'image débute par un code de début d'image (PSC) de 20 bits suivi d'informations (comme la référence temporelle (TR) ou le format d'image (CIF ou QCIF)), relatives à l'image entière. Une image CIF contient 2 colonnes de 6 GOBs (soit 12 GOBs) tandis qu'une image QCIF contient une seule colonne de 3 GOBs.

Un GOB est constitué de 11 colonnes de 3 MBs (soit 33 MBs). Cela correspond pour l'information de luminance à 48 lignes de 176 coeeficients et pour chacune des deux chrominances 24 lignes de 88 coefficients.

Un MB contient 4 blocs de luminance et 2 blocs de chrominances. Chaque bloc est constitué de 8 lignes de 8 coefficients. L'ensemble est récapitulé à la figure 2.3.

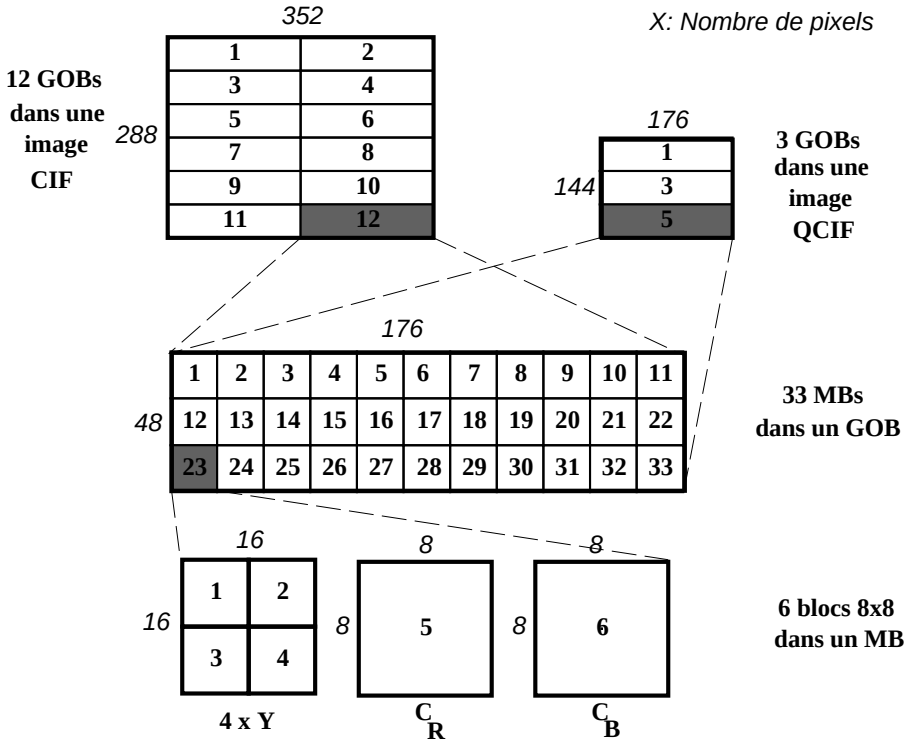


FIG. 2.3 - *Arrangement des GOBs, MBs et blocs dans une image CIF et QCIF*

2.1.3 L'algorithme de codage

L'algorithme de codage utilisé fait partie de la classe générale des algorithmes hybrides à prédiction-transformation [Lynch85]. Il associe une détection de mouvement, une prédiction temporelle compensée en mouvement, un codage par transformation, une quantification et enfin un codage entropique [Lynch85]. Nous détaillons par la suite les différents modules de compression. Le synoptique général du codeur est présenté dans la section 2.1.3, figure 2.8.

La première image est encodée en mode intra-image. Dans la suite et sauf avis contraire, la plupart des blocs seront codés en mode inter-image. La décision de codage en mode intra-image peut être prise pour rafraîchir certains macroblocs après plusieurs encodages successifs en mode inter-image. On peut aussi choisir un mode intra-image après un changement de scène ou bien à la demande d'un récepteur. Dans le cas du codage inter-image, une détection de mouvement est tout d'abord appliquée à l'image. Seuls les blocs d'image qui sont suffisamment différent des blocs correspondants dans l'image précédente seront encodés.

La prédiction temporelle compensée en mouvement

Dans un codage inter-image à prédiction, la compensation en mouvement a pour objet d'augmenter l'efficacité de la prédiction. Elle ne doit pas être confondue avec l'opération de détection de mouvement citée plus haut.

Plusieurs techniques de compensation en mouvement existent [Kim92] [Petajan92], [Sullivan92]. La "méthode par mise en correspondance" est la plus courante pour le codage de séquences d'images. Cette technique consiste à anticiper le déplacement d'un objet dans l'image à partir de la connaissance des images antérieures. Pour cela, on cherche à mettre en correspondance les parties en mouvement de l'image courante avec celles de l'image précédente. La compensation s'opère en deux temps, [Guichard86a] (figure 2.4).

- Estimation de mouvement
  Pour un bloc de l'image, on recherche un bloc dans l'image précédente dans une zone centrée sur la projection orthogonale du bloc analysé tel que la différence entre ces deux blocs soit minimale.

- Compensation de mouvement
  On calcule un vecteur de déplacement afin de faire coïncider la position antérieure du bloc et sa projection orthogonale à partir de laquelle la prédiction pourra se faire. Le bloc trouvé sera considéré comme le bloc prédit et le vecteur de déplacement calculé fera partie du codage de l'image courante.

La compensation en mouvement est facultative dans le codeur, cependant un décodeur qui suit la norme H.261 doit pouvoir accepter un vecteur par macrobloc. En général, les mouvements des objets d'une image à l'autre sont de faible ampleur. L'estimation du mouvement [Gilge88] [RM8] se fait pour chaque macrobloc dans un voisinage limité autour de la position du macrobloc à coder. La norme H.261 spécifie que la recherche doit se faire sur un voisinage 15 pixels. Cela réduit considérablement la complexité de l'algorithme. Le vecteur de macrobloc trouvé est utilisé pour les quatre blocs de luminance dans le macrobloc considéré. Au décodage, le vecteur relatif aux 2 blocs de chrominance, se déduit du vecteur de macrobloc en divisant par deux ses composantes verticale et horizontale.

La compensation en mouvement utilise des vecteurs de déplacement multiples d'un pixel. Cette granularité peut engendrer des perturbations dans l'image lorsque le déplacement réel de l'objet dans l'image ne correspond pas à un multiple d'un

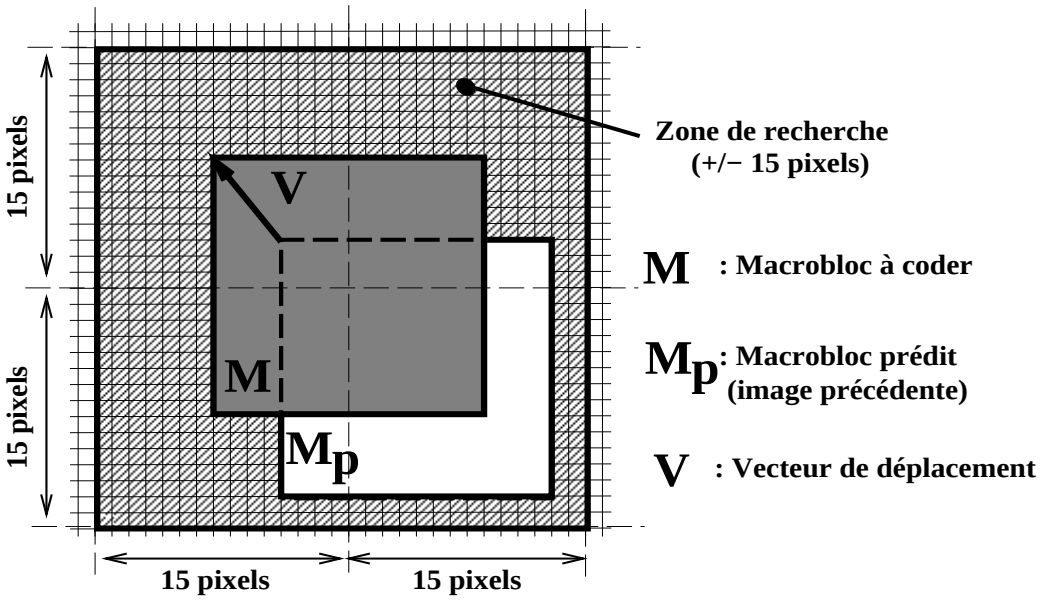


FIG. 2.4 - *Compensation de mouvement*

pixel. Pour atténuer les effets sur la qualité d'image, la compensation en mouvement peut être suivie d'un filtrage passe-bas. Sur chacune des dimensions horizontale et verticale des blocs, on applique un filtre non récursif de coefficients $[\frac{1}{4}, \frac{1}{2}, \frac{1}{4}]$ à l'exception des pixels situés sur les côtés des blocs. Dans ce dernier cas, on utilise un filtre de coefficients $[0, 1, 0]$. L'équation 2.1 montre les coefficients du filtre séparable dans les dimensions horizontale et verticale ainsi que la matrice 3×3 du filtre 2D.

$$\begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{1}{4} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} \tag{2.1}$$

L'utilisation de ce filtre a pour effet de lisser l'image et donc de la rendre un peu plus floue. Ce filtre peut aussi être utilisé lorsque les détails de l'image sont plus précis que ne le permet la résolution de l'affichage, afin d'éliminer les perturbations dans l'image.

Le codage par transformation orthogonale

Les codages par transformation orthogonale permettent de décorréler le signal à encoder. Le but est d'isoler dans le domaine transformé, les zones où se trouve l'énergie du signal, c'est-à-dire son information utile. Le codage par transformée se contente de changer la représentation du signal sans aucune réduction de débit. La

compression s'effectue ensuite dans l'étape de quantification et de codage des coefficients obtenus. Un grand nombre de transformations orthogonales existe dans la littérature [Rao90] [Reibman91]. Les plus connues sont la transformée de Karhunen-Loeve (TKL), la Transformée de Fourier Discrete (TFD), la Transformée en Cosinus Discrète (TCD), la Transformée de Walsh-Hadamard (TWH) et la Transformée de Haar (TH). La TH est la moins coûteuse en calcul, elle est d'ailleurs utilisée dans le schéma de compression du logiciel de vidéoconférence *NV* [Frederick94]. La TKL fournit la meilleure concentration en énergie dans le domaine transformé. Nous en donnons ici une description et expliquons ensuite pourquoi le CCITT a préféré utiliser la TCD dans le standard de compression H.261.

La Transformée de Karhunen-Loeve (TKL)

Soit une suite de variables aléatoires centrées X_i ($i = 0$ à $N-1$) et sa matrice de covariance Γ_X d'éléments $\gamma_{ij} = E(X_i, X_j)$. Il existe une unique transformation unitaire K appelée la transformation de Karhunen-Loeve, qui fait correspondre à la suite de variables aléatoires X_i une nouvelle suite de variables aléatoires complètement décorrélées Y_i ($i = 0$ à $N-1$):

$$Y = K \,.\, X \qquad \Gamma_X = E(X \,.\, X) \qquad \Gamma_Y = K \,.\, \Gamma_X \,.\, K^t. \tag{2.2}$$

Γ_Y est une matrice diagonale dont les éléments λ_i sont les valeurs propres de Γ_X et les variances des Y_i : $\lambda_i = E(Y_i^2)$. Cette propriété est liée à la décroissance de ses valeurs propres, qui représentent les variances des coefficients transformés, meilleure que pour toute autre transformée unitaire. Si l'on suppose que les variances des coefficients transformés d'une transformation unitaire quelconque sont classés en ordre décroissant $\sigma_0^2 \geq \sigma_1^2 \geq \ldots \geq \sigma_{N-1}^2$ et que les valeurs propres de la matrice de covariance sont classées dans le même ordre $\lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_{N-1}$, on peut montrer que pour toute valeur de $M \leq N$:

$$\Sigma_{i=0}^{M-1} \lambda_i \;\geq\; \Sigma_{i=0}^{M-1} \sigma_i^2 \,. \tag{2.3}$$

Cependant la TKL est rarement utilisée dans la pratique en raison du coût élevé de calcul qu'elle nécessite. On lui préfère la TCD qui utilise une base de fonctions fixes indépendante de l'image à encoder et donc beaucoup moins onéreuse en calculs.

La Transformée en Cosinus Discrète (TCD)

La TCD a été introduite par Ahmed, Natarajan et Rao en 1974 [Ahmed74]. Pour des raisons de complexité, on n'effectue pas la TCD d'emblée sur l'ensemble de l'image. On décompose préalablement cette dernière en blocs élémentaires carrés de pixels (en général de côté 8), puis on calcule la transformation sur chacun des blocs élémentaires. La TCD et son inverse sont définies respectivement dans le cas bi-dimensionnel et pour des blocs de taille (8×8) par les relations 2.4 et 2.5.

$$\begin{aligned} F(u,v) &= 1/4\, C(u)\, C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \\ &\quad \cos(\pi\,(2x+1)\,u\,/\,16)\ \cos(\pi\,(2y+1)\,v\,/\,16) \end{aligned} \tag{2.4}$$

$$\begin{aligned} f(x,y) &= 1/4 \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)\, C(v)\, F(u,v) \\ &\quad \cos(\pi\,(2x+1)\,u\,/\,16)\ \cos(\pi\,(2y+1)\,v\,/\,16) \end{aligned} \tag{2.5}$$

avec $C(0) = 1/\sqrt{2}$ et $C(x) = 1$ pour x compris dans l'intervalle [1..7].

La transformée en cosinus apporte deux principaux avantages par rapport à la transformée de Fourier:

- La TCD utilise une base de fonctions réelles et non complexes, ce qui simplifie les calculs et permet d'économiser de la place mémoire.

- La TCD, qui n'utilise que des fonctions cosinus, a pour effet de symétriser l'image. En revanche, la TFD périodise au préalable la séquence, ce qui peut engendrer des discontinuités et donc des erreurs en haute fréquence. La périodicité naturelle de la TCD permet donc d'éviter les erreurs à la frontière des blocs d'image [Gonzales92]. Notons qu'en traitement du signal, ces erreurs de discontinuité sont connues sous l'appellation de phénomène de Gibbs [Kunt84]. La périodisation de la TFD ainsi que la périodicité naturelle de la TCD sont illustrées par la figure 2.5.

De nombreux algorithmes rapides [Haque85] [Hou87], [Chan91], existent dans la littérature. Cette transformation est dite "industrielle" car elle fait l'objet de nombreuses réalisations commerciales aussi bien cablées que programmées.

La quantification

Les coefficients de la TCD sont quantifiés de manière à réduire le nombre de niveaux nécessaire pour leur transmission et à accroître le nombre de coefficients à

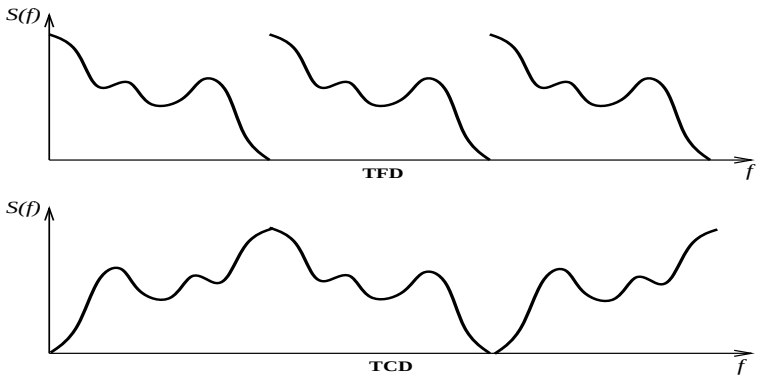


FIG. 2.5 - *Périodicité naturelle des TFD et TCD*

valeur nulle. Sans cette opération et le codage entropique qui suit, la TCD ne permettrait aucune réduction de débit. La quantification est responsable de la perte d'information induite par la compression vidéo H.261. En effet, l'opération de quantification inverse ne permet de retrouver qu'une valeur approchée des coefficients initiaux avant quantification [Scotton93]. Les coefficients de trop grande amplitude sont écrêtés de manière à borner leur amplitude. A l'inverse, on élimine tous les coefficients qui ont une amplitude trop faible, inférieure à un seuil: c'est l'opération de seuillage. Une quantification uniforme est ensuite appliquée à ces coefficients et le pas de quantification (ou quantificateur) choisi détermine à la fois le nombre de bits transmis et la finesse de reconstruction de l'image. Les valeurs quantifiées sont obtenues à partir des formules (2.6), (2.7) et les amplitudes de reconstitution, à partir des formules (2.8), (2.9).

Si Q est impair:

$$C(u,v) \;=\; F(u,v) \,/\, 2\,Q \tag{2.6}$$

Si Q est pair:

$$C(u,v) \;=\; \begin{cases} \frac{F(u,v)+1}{2\,Q} & \text{si } F(u,v) > 0 \\ \frac{F(u,v)-1}{2\,Q} & \text{si } F(u,v) < 0 \end{cases} \tag{2.7}$$

Si Q est impair:

$$\dot{F}(u,v) \;=\; \begin{cases} (2\,C(u,v) \;+\; 1)\,Q & \text{si } C(u,v) > 0 \\ (2\,C(u,v) \;-\; 1)\,Q & \text{si } C(u,v) < 0 \end{cases} \tag{2.8}$$

Si Q est pair:

$$\dot{F}(u,v) \;=\; \begin{cases} (2\,C(u,v) \;+\; 1)\,Q \;-\; 1 & \text{si } C(u,v) > 0 \\ (2\,C(u,v) \;-\; 1)\,Q \;+\; 1 & \text{si } C(u,v) < 0 \end{cases} \tag{2.9}$$

Où $C(u,v)$ représente un coefficient quantifié, $F(u,v)$, un coefficient avant quantification et $\dot{F}(u,v)$ un coefficient après quantification inverse. Dans un macrobloc, le même quantificateur est utilisé pour tous les coefficients sauf pour le premier coefficient du bloc dans le mode INTRA encore appelé composante continue (DC). Cette dernière est toujours quantifiée linéairement avec un pas de 8, en raison de son importance dans la qualité de l'image. Les autres coefficients sont quantifiés linéairement avec un des 31 quantificateurs disponibles. Généralement, le même pas de quantification est utilisé pour l'ensemble des macroblocs du GOB. Pour des valeurs d'entrée sur 9 bits (entre -255 et +255), la dynamique des coefficients transformés est de 12 bits (entre -2048 et +2047) et la quantification parvient à la réduire à quelques bits.

Le codage entropique

Le codage entropique exploite les propriétés statistiques des coefficients quantifiés pour diminuer le débit de transmission en utilisant des mots courts pour représenter les évènements les plus probables et des mots plus longs pour les occurrences rares. Afin d'augmenter l'efficacité du codage entropique, les coefficients quantifiés sont ordonnés en parcourant un trajet "zigzag" représenté par la figure 2.6. On fait

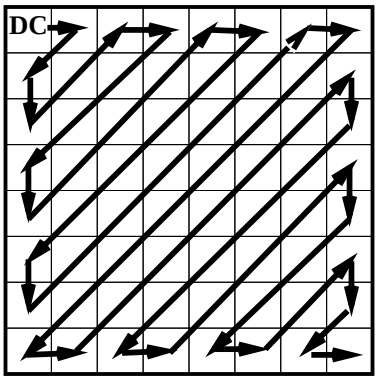


FIG. 2.6 - *Echantillonnage en "zigzag" d'un bloc* 8×8 *de coefficients*

l'hypothèse que les coefficients de basse fréquence (localisées dans le coin supérieur gauche du bloc) ont des amplitudes plus grandes que les coefficients de fréquence plus élevée qui sont souvent nuls pour la plupart des images. On applique ensuite un schéma de codage de Huffman modifié [Jain89] sur la suite de coefficients ordonnés en "zigzag". Le code de Huffman a la propriété d'être optimal car il donne la plus petite moyenne de longueur de code de toutes les techniques de codage statistique.

De plus, il est instantané ce qui signifie qu'aucun code n'est le préfixe d'un autre. Cette propriété s'assure de l'unicité de décodage de chaque code.

Le nombre de bits nécessaire pour encoder un symbole en utilisant le codage de Huffman est donné par la formule (2.10)

$$b = f\,(-\log_2 P\,) \tag{2.10}$$

Où P représente la probabilité d'occurrence du symbole et $f(x)$, l'entier le plus proche supérieur ou égal à x.

La figure 2.7 montre à l'aide d'un exemple simple comment établir le codage de Huffman à partir des statistiques effectuées sur la probabilité d'occurrence des symboles. Les symboles sont tout d'abord classés dans l'ordre décroissant de proba-

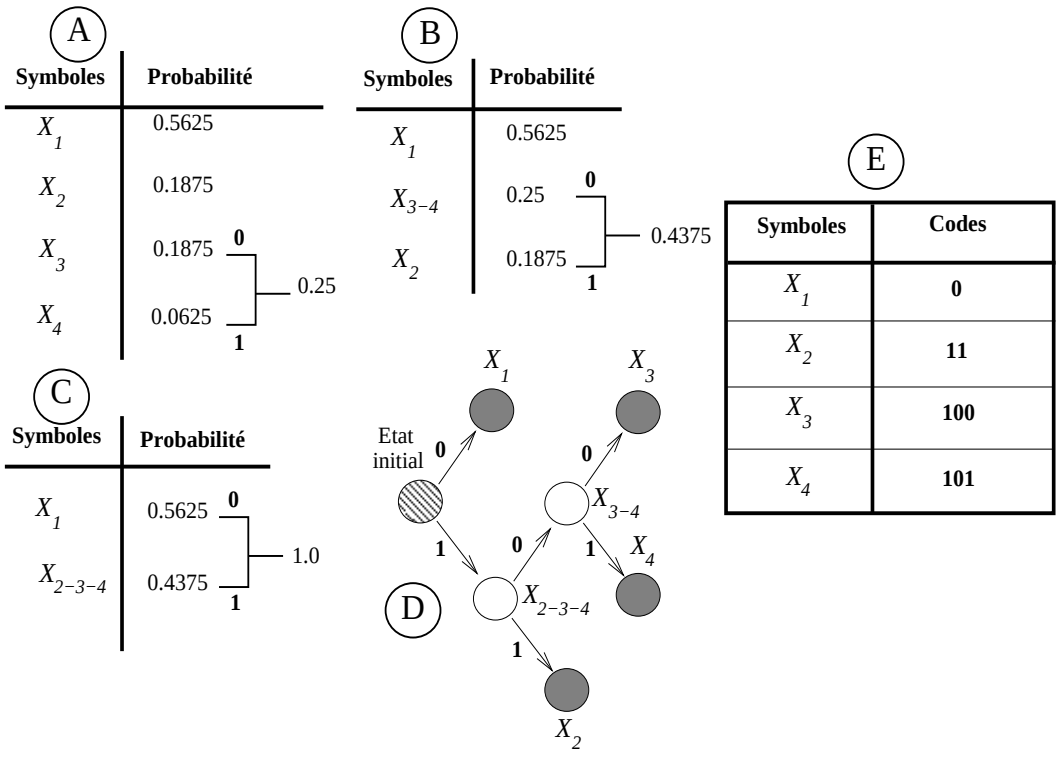


FIG. 2.7 - *Exemple de codage de Huffman*

bilité d'occurrence. L'algorithme couple ensuite le symbole de plus faible probabilité avec le symbole ou le couple de symboles voisin. Dans l'exemple, les symboles X_3 et X_4 sont couplés pour former le nœud X_{3-4} de probabilité totale 0.25, cf étape A. Les symboles sont de nouveaux reclassés dans l'ordre de probabilité d'occurrence décroissant et l'algorithme continue jusqu'à l'obtention d'une probabilité totale uni-

taire: le couple X_{3-4} est joint avec X_2 pour produire un noeud de probabilité totale de 0.4375, (étape B). Finalement, le noeud représentant les probabilités d'occurrence X_2, X_3 et X_4 fusionne avec X_1 pour former un noeud de probabilité d'occurrence unitaire (étape C). En associant à chaque branche des noeuds un élément binaire "0" et "1", on peut dériver l'arbre de construction des codes de Huffman (étape D) et les codes associées à chaque symbole (étape E).

L'entropie du codage, ou le nombre moyen de bits pour coder chaque symbole se calcule par la formule (2.11).

$$H(P) = -\sum_{k=0}^{n} P_k \, log_2 \, P_k \tag{2.11}$$

où P_k représente la probabilité d'apparition du symbole k. L'entropie vaut dans l'exemple (2.7) 1.64 bits par symbole. Si on le compare avec un code à longueur fixe de 2 bits, on peut calculer que le gain en bits est de 18 % (équation 2.12).

$$\text{Gain } (\%) \;=\; \frac{2 \,-\, 1.64}{2} \,\times\, 100 \;=\; 18.0. \tag{2.12}$$

Dans le cas du codage H.261, les coefficients dans l'ordre "zigzag" sont groupés en couple (longueur, amplitude), où longueur représente la séquence de zéro qui précède le coefficient et amplitude, sa valeur. Les couples de plus grande occurrence sont codés via un codage de Huffman, tandis que les autres, dont l'occurrence est plus faible sont codés avec un mot fixe de 20 bits composé de 6 bits d'échappement, 6 bits de longueur et 8 bits d'amplitude.

Fonctionnement de l'ensemble

Le synoptique de fonctionnement d'un codeur H.261 est montré figure 2.8. Le codage par transformation précédemment décrit s'applique soit sur le bloc de coefficients lui-même (mode intra-image ou INTRA) soit sur la différence entre ce bloc et sa prédiction temporelle (mode inter-image ou INTER) [Guichard86b] [Koga81] [Puri89]. Le choix de la stratégie la plus favorable se fait macrobloc par macrobloc. Ainsi, si un seul bloc d'un macrobloc nécessite un codage intra-image, la totalité du macrobloc doit être encodée. Le flot de données à la sortie du codeur de Huffman est très variable. Il dépend principalement du mouvement, des détails dans l'image mais aussi de l'éclairage et de la qualité de la caméra.

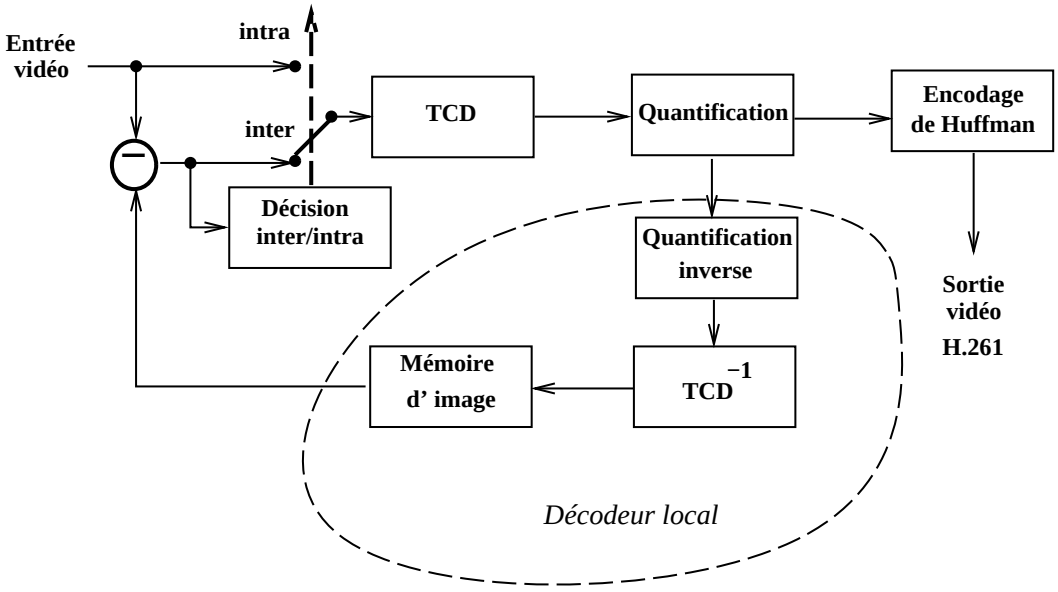


FIG. 2.8 - *Schéma du codeur H.261*

2.2 Choix de conception du codec vidéo

Comme nous l'avons déjà mentionné dans l'introduction, nous avons choisi de réaliser le codec vidéo en logiciel car il est beaucoup plus facile avec un logiciel de modifier les algorithmes qu'avec un algorithme cablé. Cette souplesse est capitale dans l'optique d'expérimentation de nouveaux algorithmes de contrôle vidéo. Lorsque nous avons débuté nos investigations en 1991, les codecs disponibles sur le marché étaient tous "cablés". Dans l'ensemble, ils étaient assimilables à des "boites noires" conçues pour être directement branchées sur une ligne de transmission. Nous avons donc décidé de développer un "codec logiciel", gageant sur la montée en puissance des stations de travail. Les stations de travail dont nous disposions à la fin de 1991 avaient déjà une capacité d'environ 30 MIPS et l'on estimait alors qu'une capacité de quelques centaines de MIPS était nécessaire pour une application de vidéoconférence de bonne qualité. Conscient que la limitation en performance du codec serait principalement dûe à une capacité en calcul insuffisante de la machine, nous avons cherché à utiliser des algorithmes qui soient les moins coûteux en calculs.

Nous présentons dans cette annexe l'architecture de fonctionnement suivi des choix de conception que nous avons fait pour le codec vidéo IVS.

2.2.1 Architecture du codec vidéo

La figure 2.9 montre l'architecture du codec vidéo IVS pour trois flots vidéo A, B et C. Le logiciel IVS est organisé en plusieurs processus. Le processus père

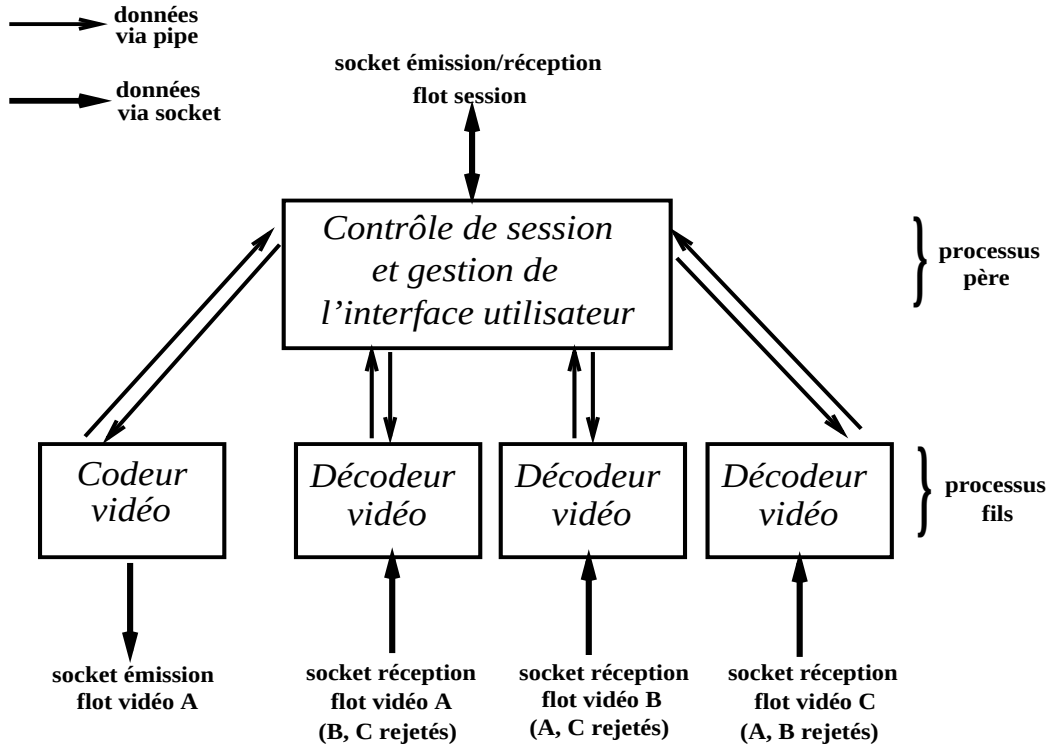


FIG. 2.9 - *Architecture du codec vidéo IVS*

est responsable de la gestion de la conférence ainsi que de l'interface utilisateur. En particulier, il s'occupe de la mise à jour des noms et des caractéristiques des participants de la conférence (via les messages RTCP, cf. section 3.2) ainsi que de la gestion des choix de réception et d'émission de l'utilisateur. Chaque utilisateur (ou participant de la conférence) est libre d'encoder de la vidéo et de décoder le (ou les) participants qu'il désire. Un processus se crée lors d'une émission vidéo ainsi que pour pour chaque décodage vidéo effectué. Il n'y a pas de limitation du nombre de décodages simultanés de flots vidéo, si ce n'est à cause de la puissance et des capacités mémoire limitées de la machine utilisée. Chaque processus se crée sa propre *socket* d'émission/réception de données et peut communiquer avec le processus père via des *pipes*[4]. L'inconvénient d'une telle architecture est que chaque processus

4. Les *sockets* et les *pipes* sont des interfaces de communication généralement utilisées pour des processus respectivement distants et locaux, [Stevens90].

décodeur doit ouvrir une socket en écoute sur l'adresse vidéo du groupe et doit donc, avant décodage, démultiplexer les paquets reçus selon l'identité de l'émetteur. Il en résulte que les paquets qui proviennent des autres flots vidéo du groupe s'accumulent inutilement dans les sockets avant d'être rejetés par le démultiplexage. Des travaux sont en cours pour permettre le décodage des flots vidéo par un seul processus. Ces modifications permettront d'augmenter l'efficacité du décodage en utilisant une seule socket en réception sur le flot vidéo du groupe.

2.2.2 Détection de mouvement

Le rôle de l'algorithme de détection de mouvement est d'identifier les blocs d'image qui sont "suffisamment différents" des blocs correspondants dans l'image précédente. Ces blocs sont d'abord "marqués" puis encodés; on estime que les autres blocs n'apportent aucune information significative par rapport à l'image précédente. Cet algorithme n'est utilisé que pour les blocs d'image dont on sait qu'il ne nécessitent pas un codage intra-image. En effet, il est inutile de procéder à une détection de mouvement sur un bloc qui nécessite d'être obligatoirement encodé en mode intra-image. Par exemple, on n'utilise pas cet algorithme après une demande explicite d'un récepteur (NACK) ou après un certain nombre d'encodages successifs en mode inter-image. D'autre part, on choisit de n'effectuer la détection de mouvement que sur les blocs de luminance de l'image et on force l'encodage des blocs de chrominances correspondant aux blocs "marqués".

Afin de réduire le coût de calcul de l'algorithme, seule une partie des pixels de chaque bloc est testée. Il est apparu par expérimentation que quatre comparaisons de pixels par bloc 8×8 suffisaient pour suivre correctement les mouvements dans une scène de vidéoconférence classique. Pour déceler tous les mouvements dans l'image, il est nécessaire de choisir des pixels suffisamment espacés les uns des autres dans chaque bloc. De plus, pour éviter que certaines zones ne restent ignorées par l'algorithme, les pixels que l'on teste doivent être différents d'une image à l'autre. La figure 2.10 montre l'ordre des comparaisons des pixels dans un bloc de 8×8 pixels. De cette manière, on s'assure qu'au bout de 16 comparaisons, soit après 16 images encodées, tous les pixels de l'image auront été testés par l'algorithme.

Une fois que l'on a identifié les pixels qui doivent être testés, il reste à effectuer pour chacun de ces pixels la comparaison entre la nouvelle image et l'image précédente afin de "marquer" les blocs de l'image. Soit V_t^i la valeur du i-ème pixel du bloc au temps t et S la valeur d'un seuil de détection de mouvement, un bloc est

	0	1	2	3	4	5	6	7
0	1	12	15	5	1	12	15	5
1	14	4	8	10	14	4	8	10
2	9	6	2	13	9	6	2	13
3	3	16	11	7	3	16	11	7
4	1	12	15	5	1	12	15	5
5	14	4	8	10	14	4	8	10
6	9	6	2	13	9	6	2	13
7	3	16	11	7	3	16	11	7

FIG. 2.10 - *Ordre des comparaisons des pixels dans un bloc* 8×8

dit “marqué” si:

$$V = \sum_{i=1}^{4} |V_t^i - V_{t-1}^i| \geq S$$

On associe de cette manière à chaque bloc d'image, un poids P utilisé dans l'algorithme de contrôle du débit. Le choix de la valeur du seuil S est délicat car il détermine la précision de l'algorithme de détection de mouvement. Choisir une valeur trop élevée ne permet pas de suivre le mouvement dans l'image. A l'inverse, une valeur de seuil trop faible peut engendrer un codage inutile de blocs. L'expérience nous a montré qu'il n'existait pas de valeur de seuil idéale pour toutes les conditions d'expérience. Par exemple, le scintillement d'un néon, invisible à l'oeil nu, ou une caméra de mauvaise qualité peuvent engendrer un encodage régulier voire permanent de tous les blocs de l'image. Nous avons choisi d'utiliser par défaut une valeur de seuil qui fonctionne dans la plupart des cas et de vérifier que cette valeur n'engendre pas d'anomalies. Par exemple, si le taux de blocs “marqués” est anormalement élevé pour chaque image encodée, la valeur du seuil S est alors augmentée.

2.2.3 Compensation en mouvement

La recommandation H.261 ne rend pas obligatoire l'utilisation de la compensation en mouvement. Cette opération comporte deux principaux inconvénients. Tout d'abord, elle rend la transmission du flot H.261 beaucoup plus sensible à la perte de paquets, voir section 4. D'autre part, la recherche par “mise en correspondance” reste coûteuse en calcul malgré les recherches effectuées à ce sujet, (voir l'algorithme rapide “3-steps” ou à “gradient conjugué” [Gilge88]). Dans le commerce, on peut trouver processeurs spécialisés qui effectuent ces opérations coûteuses (par exemple,

le circuit L64720 *Video Motion Estimation Processor* de LsiLogic). Dans notre algorithme logiciel de codage, nous avons décidé de ne pas utiliser de compensation en mouvement. Toutefois, pour que le décodeur puisse être compatible avec un codec standard H.261, le décodage des vecteurs de mouvement est prévu. L'opération de décodage pose moins de problèmes car elle est beaucoup moins coûteuse en calcul: il suffit de translater dans l'image les macroblocs selon les vecteurs de déplacement spécifiés.

2.2.4 Transformation en cosinus

La transformation en cosinus est l'étape la plus coûteuse de l'algorithme de compression. Il existe dans le commerce plusieurs processeurs qui réalisent les opérations de TCD directes et inverses (par exemple, le circuit L64730 *Discrete Cosine Transform Processor* de LsiLogic). Dans notre logiciel, nous avons implanté l'algorithme rapide de Chan [Chan91]. Afin de réduire au maximum le coût de calcul, les mesures suivantes ont été prises pour chaque bloc de pixels 8×8:

- Le codeur mémorise lorsqu'il effectue la TCD la dernière colonne du bloc où se trouve le dernier coefficient non nul. Cette information va servir à réduire les calculs dans l'opération de TCD inverse de la boucle de contre-réaction.

- Lors de la TCD, le codeur mémorise aussi la valeur maximale du coefficient fréquentiel de chaque bloc. Cette valeur va servir ensuite dans le choix du quantificateur.

- Pour limiter les accès à la mémoire, en mode inter-image l'addition des coefficients à l'image précédente se fait au sein même de la transformation inverse.

2.2.5 Quantification

L'étape de quantification ne pose aucun problème particulier. La recommandation H.261 donne les spécifications de 31 quantificateurs différents. 31 tableaux pré-compilés à 256 entrées sont utilisés pour effectuer la quantification inverse; la valeur de tout coefficient quantifié étant un entier compris dans l'intervalle [-127,127].

La valeur maximale du coefficient fréquentiel de chaque bloc est calculée pendant la TCD. En effet, il faut nous assurer que le quantificateur courant est capable d'encoder la valeur maximale du bloc. Dans le cas contraire, il est nécessaire de

choisir un quantificateur qui autorise une amplitude de reconstruction plus grande [5].

Pour gagner du temps, on met à jour dans l'opération de quantification un compteur de nombre de coefficients non nuls pour tous les blocs de l'image. Ces compteurs seront utilisés dans l'opération d'encodage de Huffman qui suit.

2.2.6 Codage entropique

La recommandation H.261 spécifie des tables de codes de Huffman pour:

- l'adressage des macroblocs dans un GOB,
- le type de macrobloc,
- les vecteurs de déplacement dans un macrobloc,
- les blocs encodés dans un macrobloc et
- les coefficients fréquentiels dans un bloc.

L'opération de codage de Huffman est immédiate en se servant des tables de Huffman. Le codage de Huffman des coefficients fréquentiels doit se faire dans l'ordre "zigzag" (figure 2.6). La connaissance du nombre de coefficients de chaque bloc nous permet d'arrêter le parcours en "zigzag" du bloc dès que le dernier coefficient a été rencontré.

Le décodage de Huffman est un peu plus compliqué car il est nécessaire de "dérouler" un automate de décodage pour chaque code. Pour cela, on se sert de tables pré-compilées qui nous donne pour chaque bit décodé: l'état de l'automate (fin de code ou non) et le mot-code courant, (par exemple "00110"). On arrête l'automate dès que la fin de code est trouvée et on lit alors dans la table la valeur qui correspond au mot-code décodé.

2.3 Utilisation du codage H.261

La recommandation H.261 a été conçue à l'origine pour une utilisation sur le Réseau Numérique à Intégration de Services (RNIS). Pour ce genre d'utilisation, un certain nombre de dispositions doivent être prises:

- On doit tout d'abord se protéger contre les altérations dues aux erreurs de transmission. La norme H.261 rend obligatoire l'insertion de bits de correction

5. En effet, les quantificateurs compris entre 1 et 8, plus précis que les autres ne permettent pas d'atteindre les amplitudes limites de reconstruction à savoir [-2048, 2048].

d'erreurs par le codeur. Le code utilisé est un BCH[6] (511, 493): 493 bits de données sont protégés par 18 bits de correction d'erreurs. Le polynôme générateur est $g(x) = (x^9 + x^4 + 1)(x^9 + x^6 + x^4 + x^3 + 1)$. L'utilisation de ces 18 bits par le décodeur est d'ailleurs optionnelle.

- Le débit à la sortie du codeur doit être constant et multiple de 64 kbit/s en raison de la capacité fixe des canaux RNIS. Pour cela, le codeur doit assurer une régulation du débit de sortie. Le débit de sortie peut être ajusté en agissant sur le quantificateur, le critère de sélection des blocs et le sous-échantillonnage temporel. Cependant, l'importance relative de ces actions dans la stratégie globale de régulation ne fait l'objet d'aucune recommandation. Une mémoire tampon est ajoutée à la sortie du codeur pour ajuster son débit à celui de la liaison. Le stockage des bits à transmettre permet de lisser les pics de débit en cas de fort mouvement. Pour être efficace, elle ne doit ni se vider, car alors des bits de bourrage sont transmis, ni se remplir totalement.

- La mise en trames et le multiplexage du flot H.261 avec les flots d'audio et de contrôle est décrit dans la recommandation H.221 [H221] de l'UIT. Cette norme définit une structure de trame pour les canaux simples à 64 kbit/s ou multiples (pouvant ainsi couvrir la gamme de 64 kbit/s à 2 Mbit/s), destinée aux services audiovisuels.

La transmission du flot vidéo H.261 via l'Internet peut se faire en utilisant l'un des deux protocoles de transport disponibles, à savoir TCP (*Transport Control Protocol*) [RFC793] ou UDP (*User Datagram Protocol*) [RFC768]. Nos premiers essais de transmission vidéo [Huitema92] ont été effectués par des connexions TCP en "point à point". Cette méthode de transmission, ne demande pratiquement aucune modification du codage H.261, seule l'utilisation du BCH est redondante avec le protocole de transmission fiable TCP. Cependant, cette méthode comporte un certain nombre d'inconvénients. Tout d'abord, les délais engendrés par le protocole de retransmission de paquets perdus sont incompatibles avec la contrainte temps réel de ce genre d'application. D'autre part, la transmission multipoint (ou diffusion) à un nombre quelconque de récepteurs n'est pas possible avec TCP.

L'autre solution que nous avons adoptée consiste à utiliser un protocole de transport minimal et à demander à l'application d'implanter les fonctionnalités indispen-

6. Bose, Chaudhuri Hocquengham

sables à la transmission de la vidéo. Par exemple, on peut utiliser UDP comme protocole de transport et laisser le soin à l'application le soin d'effectuer les opérations de mise en paquets des données, de contrôle d'erreur et de contrôle de congestion. Ce sont ces trois opérations que nous allons étudier dans les trois chapitres suivants.

2.4 Analyse des limitations en performance du codec

Dans cette section nous récapitulons où se trouvent les principales limitations en performance du codec et dans quelles mesures le *hardware* est en mesure d'accroître les performances obtenues.

Tout d'abord, le codeur est limité par les performances de la carte d'acquisition vidéo. Par exemple, si l'on utilise la carte d'acquisition vidéo VideoPix[7], la vitesse d'encodage est limitée à 5 images par seconde. De nos jours, le goulot d'étranglement s'est déplacé car la plupart des cartes vidéo disponibles sur le marché permettent d'acquérir en mémoire 25 images par seconde, au moins avec le format QCIF (par exemple, la carte SunVideo[8] ou la carte VigraPix[9]). Aujourd'hui, la limitation des performances est principalement dûe à la limitation de la puissance disponible de la machine. Comme nous l'avons mentionné précédemment, les étapes du codage H.261 les plus coûteuses sont la TCD, la compensation en mouvement (qui n'est pas utilisée par notre codeur) et le codage de Huffman. Des machines plus puissantes permettront un encodage plus rapide. On peut imaginer d'utiliser des processeurs spécialisés pour la TCD et la compensation en mouvement. L'optique de nos travaux étant de réaliser un codec vidéo à moindre coût et d'utilisation universelle (i.e. sur n'importe quelle station de travail), nous n'avons pas approfondi cette possibilité.

La vitesse de décodage est aussi limitée par la puissance disponible de la machine. Les opérations coûteuses sont la TCD inverse, le décodage de Huffman et sur certaines plate-formes l'affichage de l'image. Comme pour le codeur, l'utilisation de machines plus puissantes permet d'accélérer le décodage. D'autre part, le coût de l'affichage en couleur peut être réduit en utilisant des écrans en "vraie couleur" de 24 bits de "profondeur". Ce type d'écran permet de coder chaque pixel avec 24 bits, soit 8 bits pour chaque composante des pixels (rouge, vert et bleu). Avec des écrans moins performants, des opérations supplémentaires sont nécessaires car la gamme

7. VideoPix est développée par Sun Microsystems.

8. SunVideo est développée par Sun Microsystems.

9. VigraPix est développée par Vigra.

des couleurs utilisable est réduite[10].

2.5 Conclusion

La principale contribution de ce chapitre est de montrer qu'une mise en œuvre logicielle d'un codec H.261 est possible avec la technologie actuelle. Les performances obtenues sont sans cesse améliorées en raison à la fois de l'arrivée de nouvelles machines, de nouvelles cartes d'acquisition vidéo et aussi à l'optimisation du code. A titre d'exemple, la version 3.4 de notre d'IVS permet d'encoder sur une plate-forme SS10 une moyenne de 20 images par seconde en format QCIF et de cinq images par seconde en format CIF. Ce codec est le maillon de base du logiciel de vidéoconférence IVS qui a été porté sur de nombreuses plate-formes (SUN, Silicon Graphic, DEC, HP, PC).

10. Par exemple, avec un écran de "profondeur" 8 bits, seules 256 couleurs sont disponibles. Il devient alors nécessaire d'associer à chaque pixel une des 256 couleurs disponibles en tenant compte des pixels voisins pour réduire la distorsion de l'image.

Chapitre 3

Transformation du flot H.261 en paquets

Ce chapitre décrit en détails le schéma de découpage en paquets du flot vidéo H.261 que nous proposons comme standard à l'IETF [1] [Turletti95], [Turletti93b]. Pour transmettre un flot vidéo H.261 via le protocole de transport UDP, il faut tout d'abord transformer le flot de bits à la sortie du codeur vidéo en un flot de paquets (ou datagrammes). Pour raison d'efficacité, nous avons choisi un découpage qui utilise le principe de "mise en trame par l'application" (*Application Level Framing*, ALF) [Clark90]. Nous avons décidé d'utiliser le protocole de transmission temps réel (*Real Time Protocol*, RTP [2]) pour avoir un format d'encapsulation qui soit commun avec les autres applications multimédia qui se développent actuellement sur l'Internet.

Le principe ALF est exposé dans la première section de ce chapitre. Nous décrivons ensuite brièvement le RTP dans la deuxième section et nous donnons enfin une description détaillée de notre schéma de découpage dans la dernière section.

3.1 Mise en trame par l'application

Les applications multimédia manipulent des données de volume important. Afin qu'elles puissent communiquer entre elles en temps réel, une bande passante élevée

1. L'IETF (Internet Engineering Task Force) est le forum où sont définis les standards de l'Internet [RFC1602].

2. Ce protocole est actuellement en cours de standardisation à l'IETF, la dernière version est [Schulzrinne95].

est nécessaire. Ces applications ont des besoins nouveaux qui ne correspondent pas aux services offerts par les protocoles en couches [Zimmermann80]. Ces besoins nouveaux ont motivé la définition des principes de "mise en trame par l'application" (*Application Level Framing* ALF) ainsi que du traitement intégré des couches (*Integrated Layer Processing ILP*) [Clark90] qui regroupe un ensemble de concepts d'architecture de réseaux. Le principe d'ALF et ILP est de minimiser les accès mémoire et le nombre d'aller/retour des paquets dans le réseau. En effet, ces opérations constituent le principal goulot d'étranglement des protocoles en couches. Dans le concept ALF, les opérations sont regroupées en deux phases: le traitement des données et le contrôle de transfert des données. Le traitement des données concerne la lecture et la copie d'octets, la détection d'erreurs, le stockage et la présentation des données à l'application. Le contrôle du transfert regroupe le contrôle de flux, le contrôle de congestion, le traitement des acquittements, la détection de perte de paquets, le reséquencement, etc... La phase la plus critique est le traitement des données en raison du volume important de données transmises. Pour obtenir une efficacité de traitement des données maximale, il est nécessaire de minimiser le nombre de fois où ces données traversent le bus de données. D'où la technique de traitement intégré des couches (ILP) qui repose sur les deux conditions suivantes: le traitement d'un paquet par une suite de protocole ne doit pas être bloquant (sinon ILP ne peut être réalisé) et les opérations des différentes couches doivent pouvoir être combinées entre elles. Ainsi, on essaiera de manipuler les données le plus souvent dans une seule boucle de traitement de manière à minimiser les accès mémoire. On peut noter que cette méthode est contraire à l'architecture traditionnelle en couches qui suggère que chaque couche à terminer son traitement des données avant que la couche suivante ne commence le sien. Le concept ALF utilise la terminologie suivante:

- Le NDU (*Network Data Unit*) représente l'unité d'échange du réseau sous-jacent, e.g. un paquet IP ou une cellule ATM [3]. Dans la suite, on ne fera aucune distinction entre les termes paquet et NDU.

- l'ADU (*Application Data Unit*) est la plus petite unité de donnée que l'application peut traiter sans avoir recours à un reséquencement préalable. Les ADUs sont transmis sous forme de NDUs à travers le réseau. Il est légitime

3. ATM (*Asynchronous Transfert Mode*) est un mode de transmission qui utilise la commutation de cellules [Haendel94]

de laisser à l'application le choix de son ADU étant donné qu'elle est la plus à même de connaître la structure des informations transmises. Lorsqu'une ADU arrive, l'application effectue le traitement des données (détection d'erreurs, séquencement, présentation, etc...) en une seule passe. De cette manière, on réduit au minimum le nombre de copies et de changements de contexte qui surviennent par ADU.

L'utilisation de ALF et ILP apporte plusieurs avantages aux applications de vidéoconférence [Heybey92]:

- Ils permettent de mettre en œuvre efficacement les services requis par la vidéoconférence. Par exemple, l'application n'est pas contrainte d'utiliser les services dont elle n'a pas besoin: elle peut choisir de créer un service de flots multiples avec différentes priorités plutôt que d'utiliser le service de flot d'octets fiable.

- Ils facilitent le multiplexage au niveau du décodeur, en permettant le décodage des ADUs hors séquence dans une même trame.

- Ils permettent d'éviter d'utiliser le service de fragmentation de la couche réseau, sous réserve que la taille des ADUs soit compatible avec celle des NDUs, et donc à condition de connaître la taille maximale des NDUs dans le réseau utilisé [4].

Dans le cas d'un codeur vidéo H.261, on distingue deux cas de figure, selon la conception du codeur. Si le codeur est réalisé en logiciel, il peut être en mesure de manipuler directement les ADUs. Dans le cas contraire (par exemple un codec H.261 cablé du commerce), le codeur peut se schématiser en une boite noire qui génère un flot d'octets au format H.261. Les principes de ALF/ILP étant inverses à ceux de la boite noire, ils ne pourront pas être mis en œuvre complètement. L'opération de découpage en paquets doit dans ce cas se faire à la sortie du codeur vidéo, et ce au détriment des performances.

3.2 Le protocole RTP

Le protocole *Real Time Protocol* (RTP) [Schulzrinne95] est en cours de développement au sein du groupe de travail AVT (*Audio Video Transport*) de l'IETF.

4. Cette taille varie en fonction du réseau considéré, e.g. elle est de 1536 octets pour le réseau Ethernet mais est souvent de 576 octets pour l'Internet.

RTP fournit un format d'encapsulation pour les applications de type temps-réel qui permet de gérer le temps, le démultiplexage, l'identification du contenu et la sécurité des paquets. Il est principalement utilisé pour les applications d'audio et de vidéoconférence, telles que *IVS* [Turletti93a] [Turletti94b], *NEVOT* [Schulzrinne92], *NV* [Frederick94], *VAT* [Jacobson92]. RTP est conçu pour fonctionner au dessus de protocoles de transport de "bout en bout" tels que UDP, TCP, OSI TP1 et TP4. Il utilise les services offerts par la couche transport comme le transport des données de "bout en bout", le démultiplexage de plusieurs connexions de transport sur une même connexion réseau et la transmission multipoint des données. Il peut aussi être utilisé directement comme protocole de transport au dessus d'IP ou de ST-II pour accroître les performances et réduire le sur-coût généré par les octets de l'entête. Cette dernière solution est intéressante pour certaines applications temps-réel qui n'utilisent pas les services offerts par UDP (contrôle d'erreurs, démultiplexage). Etant donné qu'on ne se sert d'aucune hypothèse sur la fiabilité du réseau sous-jacent, l'application se doit de pouvoir gérer les pertes de paquets, les paquets reçus hors séquence ainsi que les délais non garantis. RTP peut être utilisé conjointement avec le protocole de contrôle *Real Time Control Protocol* (RTCP [5]) qui fournit un ensemble minimal de fonctionnalités pour la conférence comme l'identification de l'émetteur et le support des passerelles entre les différents codages. Cependant, RTP peut aussi être utilisé avec moins de fonctionnalités sans aucun protocole de contrôle. Les relations entre RTP, RTCP et les autres protocoles de l'Internet sont schématisées dans la figure 3.1.

RTP a été éconçu avec les préoccupations suivantes:

- Indépendance avec les protocoles de couches inférieures: le minimum d'hypothèses est fait sur le protocole de transport sous-jacent. Par exemple, RTP doit fonctionner aussi bien avec UDP qu'avec des protocoles expérimentaux supportant la réservation de ressources et des garanties de qualité de services (cf. section 6.3.2).

- Efficacité: Les applications multimédia génèrent un trafic important. Par exemple, pour une transmission audio, le découpage en paquets se fait cou-

5. La description du protocole RTCP ne fait pas partie de l'objet du chapitre, nous invitons les lecteurs à se reporter à [Schulzrinne95] pour plus de détails

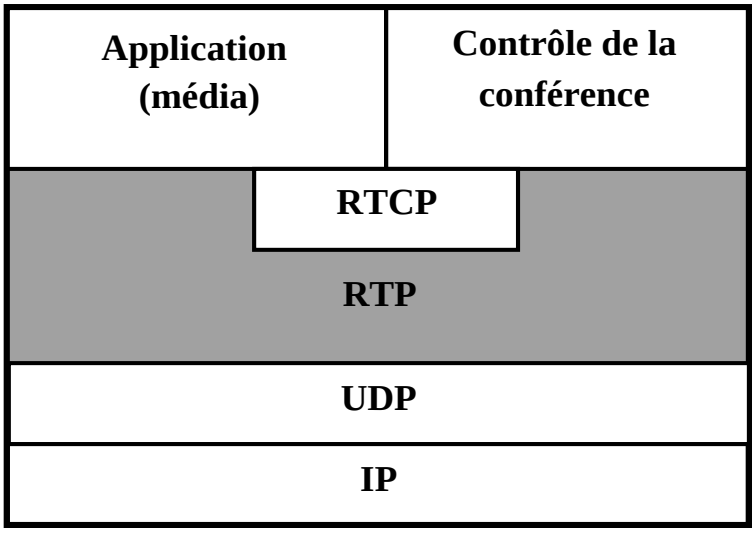


FIG. 3.1 - *Relations entre l'application, RTP, RTCP et la pile de protocoles de l'Internet*

ramment sur des intervalles[6] de 20 ms. Avec un codage de type PCM[7], cela représente une émission de 50 paquets de 160 octets de données par seconde. Si ces paquets sont émis en UDP, l'en-tête généré par IPv4 est de 20 octets, et celui d'UDP est de 8 octets. En rajoutant l'en-tête minimal de RTP de 12 octets, cela donne un en-tête de 40 octets pour 160 octets de données; soit un coût supplémentaire de 25 % d'octets. Il est donc nécessaire de minimiser à la fois la longueur de l'en-tête RTP et la charge cpu nécessaire pour le traiter (par exemple, en faisant coïncider la taille de l'en-tête avec celle d'un mot[8]).

- Flexibilité du type des données transmises: RTP doit pouvoir être extensible à des applications autre que l'audio et la vidéoconférence.

- Compatibilité avec des passerelles: les passerelles de niveau RTP permettent de concaténer plusieurs flots de média en un flot unique en changeant la taille des paquets et/ou le codage et/ou le protocole de transport.

Nous donnons ici une brève description de la version 1 du protocole

6. Des intervalles courts permettent de minimiser à la fois l'impact de la perte des paquets sur le son ainsi que les délais de transmission de "bout en bout". D. Minoli montre qu'un intervalle de 16 à 32 ms est considéré comme optimal pour la voix [Minoli79].

7. PCM (*Pulse Code Modulation*) est une technique d'encodage de la parole à 64 kbits/s défini par le standard G.711 du CCITT.

8. Un mot vaut 4 octets dans la majorité des machines actuelles.

RTP [Schulzrinne93], car c'est cette version que nous avons utilisé dans notre logiciel. Une version plus récente (version 2) [Schulzrinne95] est en cours de standardisation à l'IETF.

L'en-tête de RTP est schématisé dans la figure 3.2. Le tableau 3.1 donne la signification de chacun des champs de l'en-tête de RTP.

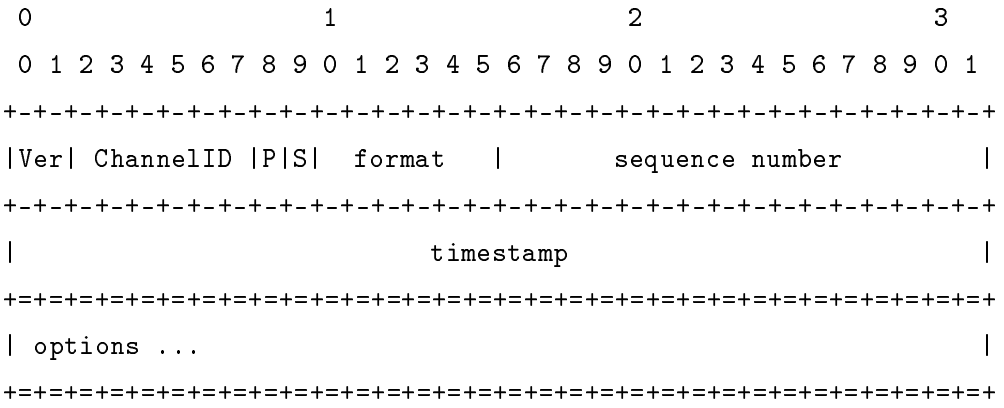


FIG. 3.2 - *Format de l'en-tête RTP*

Les principaux services rendus par RTP sont résumés ci-dessous:

- segmentation et réassemblage (différent selon le type de média, e.g. le schéma que nous proposons pour le codage H.261),
- démultiplexage selon le type de média: e.g. audio PCM, vidéo H.261,
- suppression de la gigue de délai introduite par la transmission au niveau du récepteur (en utilisant l'estampille),
- détection de perte de paquets (en vérifiant la continuité des numéros de séquence des paquets reçus),
- contrôle de la qualité de réception (en émettant des rapports de qualité de réception),
- encryption (en permettant l'utilisation de méthodes de cryptage [Handley95]).

Champs	Taille (en bits)	Contenu
Ver	2	Numéro de version du protocole RTP (ici 1).
ChannelID	6	Identificateur de flot. C'est un niveau supplémentaire de multiplexage à la couche RTP.
P	1	Bit qui vaut 1 si l'en-tête RTP est suivi d'une ou plusieurs options.
S	1	Bit qui vaut 1 si ce paquet est le dernier d'une unité de synchronization.
format	6	Type de média contenu dans le paquet, e.g. la valeur 31 est reservée pour H.261.
sequence number	16	Numéro de séquence du paquet, incrémenté de 1 pour chaque paquet émis.
timestamp	32	Estampille qui reflète l'instant d'échantillonnage des données contenues dans le paquet.
options	variable	Des options facultatives peuvent suivre l'entête RTP. Chaque option contient un bit final (F), la longueur de l'option et un nom qui l'identifie.

TAB. 3.1 - *Description de l'en-tête RTP*

3.3 Schéma de découpage du flot vidéo H.261

Le protocole RTP que nous venons de décrire est utilisé pour transmettre les données vidéo H.261. Ces données nécessitent d'être découpées en ADUs. Pour être efficace, le schéma de découpage en ADUs du flot vidéo H.261 doit être élaboré en prenant en compte les objectifs suivants:

- Le coût du découpage/réassemblage doit être minimal pour permettre d'élaborer un algorithme rapide.
- La taille des paquets (ou NDUs) générés doit être assez grande pour éviter de congestionner le réseau par l'emission d'une multitude de petits paquets.
- Le décodeur doit pouvoir se resynchroniser facilement en cas de pertes de paquets.

- L'algorithme doit engendrer un délai minimal. Par exemple, on doit pouvoir afficher une image dès que toutes les ADUs correspondantes sont reçues, sans devoir attendre la réception de l'ADU suivante [9].

- En cas de décodage à débit constant (e.g. codec H.261 cablé du commerce), on doit pouvoir maintenir la synchronisation. Par exemple, en ajoutant des bits de bourrage (ou *padding*) si le débit reçu est inférieur au débit escompté.

- Pour minimiser l'effet de la perte de paquets sur l'image décodée, le décodeur doit si possible, pouvoir utiliser tous les paquets qu'il reçoit. Ceci implique un découpage en paquets qui tient compte de la structure du codage H.261. Nous verrons plus loin que cela est réalisé lorsque la taille de l'ADU correspond à celle du paquet.

- L'algorithme de découpage en paquets doit pouvoir être utilisé à la fois par les codecs logiciels que par les codecs cablés disponibles dans le commerce.

Choix de l'ADU

Un problème délicat et essentiel dans l'algorithme de découpage consiste à choisir l'ADU. Pour des raisons d'efficacité, le choix de l'ADU doit se faire en fonction de la structure intrinsèque des données. Le découpage doit donc tenir compte de la structure hiérarchique du codage H.261. Cependant, un tel découpage n'est possible qu'à la condition de supprimer le code de correction d'erreur BCH (cf. section 2.3). En effet, ce code rend impossible tout découpage structuré du flot d'octets émis et est de plus totalement inefficace en cas de perte de paquets. Un schéma de contrôle d'erreur peut le remplacer avantageusement, (cf. chapitre 4).

Nous avons vu en section 2.1.2 que le flot H.261 était structuré en quatre couches (figure 3.3), à savoir la couche image, la couche GOB, la couche MB et la couche bloc. Pour que l'application (c'est-à-dire le décodeur vidéo) puisse décoder toutes les ADUs qu'il reçoit dans un ordre quelconque, il faut que ces dernières soient indépendantes les unes des autres. Admettons que nous choisissions la couche image comme ADU. Dans ce cas, aucune information supplémentaire n'est nécessaire à l'en-tête RTP, chaque image étant indépendante. En ce qui concerne les dépendances entre les GOBs, on note l'estampille (déjà présente dans l'en-tête RTP) et le format d'image

9. Puisque le codage H.261 ne prévoit pas de code spécifique pour la fin d'image, on peut utiliser le bit de synchronisation (S) de RTP pour marquer le dernier paquet d'une image, cf. [Schulzrinne95]

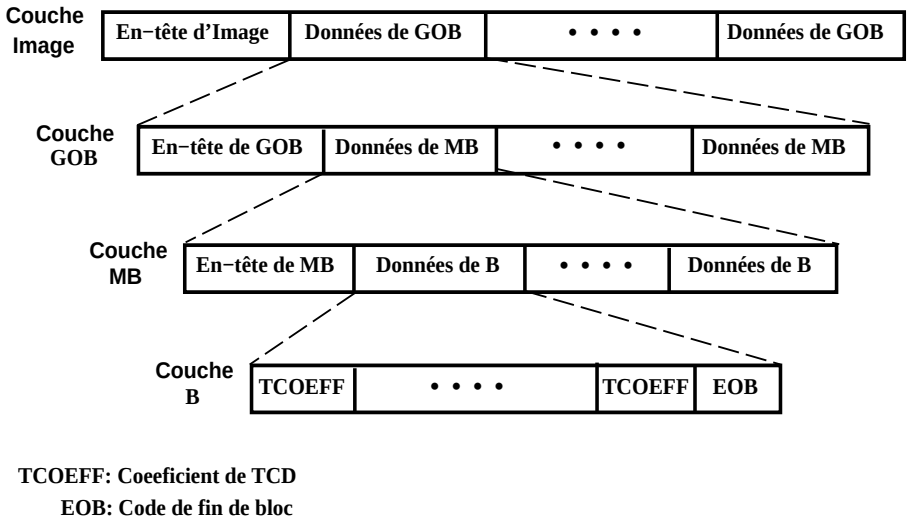


FIG. 3.3 - *Structure hiérarchique d'une image H.261*

(là un nouveau champ est nécessaire). On trouve davantage de dépendance entre les MBs: le numéro d'un MB est relatif au MB qui le précède et le quantificateur utilisé qui est soit celui défini dans la couche GOB, soit un nouveau quantificateur défini dans l'en-tête courant du MB ou dans un en-tête de MB précédent. Il faut donc rajouter dans le cas où l'on choisit d'associer l'ADU au MB, et en plus du champ de format, les champs suivants: le numéro de MB précédemment encodé dans le GOB courant, le numéro de GOB courant et le précédent quantificateur utilisé dans le GOB courant.

Pour réduire l'effet de la perte de paquets (ou NDUs) sur la qualité de l'image, il faut permettre à l'application de décoder tous les NDUs qu'elle reçoit. Ceci implique que l'ADU ne soit pas transmise par l'intermédiaire de plusieurs NDUs. Dans le cas contraire, la perte d'un seul NDU entraine le rejet de l'ADU complète, ce qui réduit considérablement l'efficacité du mécanisme de découpage.

D'autre part, pour atteindre un maximum d'efficacité, il est nécessaire que la taille des NDUs soit voisine de la taille maximale de transmission des unités de données (*Maximal Transmission Unit* MTU) du chemin[10]. On élimine de cette manière le sur-coût généré par les opérations de fragmentation et de réassemblage de la couche réseau qui peut s'avérer prohibitif (cf. exemple de la section 3.2).

Rappelons que le flot généré par un codeur H.261 est un flot à débit variable. La

10. Cette taille varie en fonction du réseau considéré, e.g. elle est fixée à 1536 octets pour le réseau Ethernet mais est souvent de 576 octets pour l'Internet.

recommandation H.261 spécifie que la taille d'une image CIF ne doit pas excéder 32 koctets, ce qui donne une taille maximale d'environ 3 koctets pour un GOB, 90 octets pour un MB et 15 octets pour un bloc. Dans la pratique, on observe que la taille des images s'étend de quelques dizaines d'octets à une vingtaine de koctets en fonction du mouvement et des détails dans la scène vidéo encodée. Pour être performant en toutes circonstances, le schéma de découpage en paquets doit donc s'adapter dynamiquement à la taille des images.

Nous avons décider d'associer l'ADU au MB. Cette solution permet d'approcher la taille optimale des paquets (MTU) en émettant un nombre variable et entier d'ADUs par NDU. Ainsi, selon la nature de l'image encodée, on peut aussi bien transmettre tous les MBs d'une l'image dans le même paquet (cas courant lorsque la scène vidéo ne comporte que très peu de mouvement), ou émettre une vingtaine de paquets de taille voisine de 1000 octets par image (e.g. lorsqu'on encode toute une image CIF en mode INTRA).

Cependant, un tel découpage est difficilement réalisable dans le cas de codecs H.261 cablés. En effet, un découpage de flot qui a pour granularité le MB, implique un décodage de Huffman complet du flot de données à la sortie du codeur H.261. En revanche, en prenant pour granularité le GOB, il suffit de reconnaître les codes de débuts de GOB pour identifier les frontières de GOB. Afin d'autoriser les deux solutions dans le cas de codecs cablés, un indicateur **F** est utilisé pour informer que les champs relatifs au MB ne sont pas valides. D'autre part, pour permettre un algorithme de reséquencement rapide de paquets coté décodeur, on demande à ce que tous les paquets émis soient de même taille. Ainsi, ils pourront être placés dès qu'ils arrivent à la bonne place dans le tampon circulaire d'entrée. Par exemple, si un GOB a une taille de 2600 octets, il peut être émis en 3 paquets dont les deux premiers font 1000 octets et le troisième 600 octets. Si dans ce cas, le GOB suivant est de taille inférieure à 400 octets, il peut être concaténé au paquet précédent de 600 octets. Cette possibilité est intéressante car la recommandation H.261 spécifie que tous les en-têtes de GOB doivent être encodés même si aucune donnée ne suit l'en-tête. On évite ainsi de transmettre dans certains cas des paquets qui ne contiennent aucune donnée utile.

Mais, comme nous l'avons vu précédemment, il n'est possible d'associer l'ADU au MB qu'en ajoutant un certain nombre de champs à l'entête RTP. C'est le but de l'en-tête (appelé H.261/RTP) qui est présenté dans la figure 3.4. L'en-tête H.261/RTP est composé d'un mot de 32 bits qui est ajouté à la suite de l'en-tête

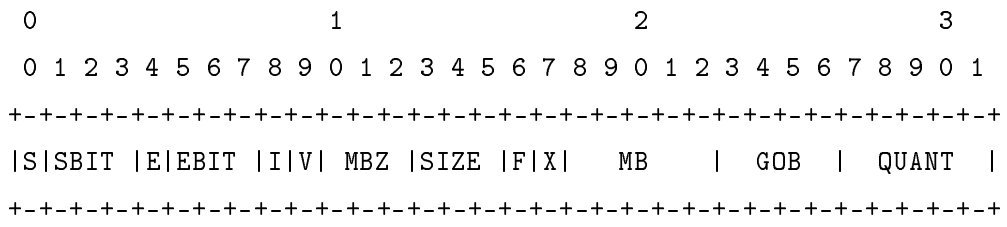


FIG. 3.4 - *Format de l'en-tête H.261/RTP*

RTP et qui précède les données H.261 comme le montre la figure 3.5.

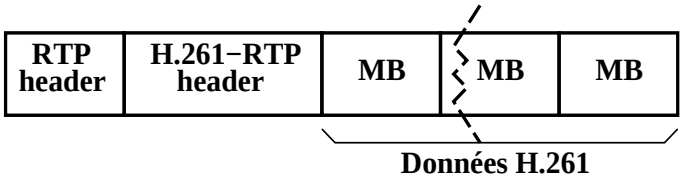


FIG. 3.5 - *Constitution d'un paquet vidéo*

- L'indicateur **S** (1 bit) est utilisé pour indiquer que le paquet débute par un en-tête de GOB. Avec cette information, on sait que les 16 premiers bits de données H.261 correspondent au code de début de GOB et on peut donc s'abstenir de les transmettre (ils seront ajoutés automatiquement à la réception).

- L'indicateur **E** (1 bit) signifie que le paquet contient une fin d'ADU et peut donc être décodé. Il est toujours à 1 dans le cas d'un codeur logiciel et signifie que le paquet contient un fin de GOB dans le cas d'un codeur cablé.

- **SBIT** (3 bits) indique le nombre de bits à ignorer au début du premier octet de données et **EBIT** (3 bits) le nombre de bits à ignorer à la fin du dernier octet de données.

- L'indicateur **I** permet de typer une image encodée entièrement en mode INTRA. Il est utile par exemple en début de décodage pour ne commencer l'affichage qu'à la réception d'une image entièrement codée.

- L'indicateur **V** (1 bit) permet d'accélérer[11] le décodage lorsque les vecteurs de mouvement ne sont pas codés.

11. Dans ce cas, la sauvegarde en mémoire de l'image précédente est inutile

- Le champ **SIZE** (3 bits) rend le GOB indépendant de la couche image en cas de changement de format en cours de codage conjugué avec la perte de l'en-tête d'image.

- L'indicateur **F** (1 bit) annonce que les champs **MB**, **GOB** et **QUANT** qui suivent sont valides: il vaut 0 dans le cas d'un codec cablé.

- Enfin **MB** (5 bits) correspond au numéro de MB précédemment encodé dans le GOB courant, **GOB** (4 bits) au numéro de GOB courant et **QUANT** (5 bits) au précédent quantificateur utilisé dans le GOB courant.

Notons que le décodage des ADUs reçues hors séquence est possible à l'intérieur d'une même image. En revanche, l'ordre de décodage des images doit être respecté, d'une part pour respecter la continuité visuelle et d'autre part l'utilisation du codage inter-image implique une dépendance entre les images successives.

3.4 Conclusion

Dans ce chapitre, nous avons proposé un schéma de découpage des données vidéo H.261 pour l'Internet. Ce découpage est performant car il applique les techniques ALF et ILP. Le protocole RTP est utilisé pour encapsuler les ADUs d'une manière standard pour la plupart des applications multimédia en vogue sur l'Internet. Le schéma de découpage que nous venons de décrire est proposé comme standard dans le groupe de travail AVT (*Audio Video Transport*) de l'IETF. Il est utilisé dans IVS et aussi par plusieurs codecs cablés ce qui assure l'intéropérabilité entre ces codecs [Handley93].

Chapitre 4

Le contrôle d'erreurs

Comme nous l'avons vu à la section 2.3, la recommandation H.261 a été conçue à l'origine pour une utilisation sur le RNIS. Rappelons qu'avec ce type de réseau, le problème de contrôle d'erreurs se résume à corriger les bits qui ont été erronés par la transmission. Dans l'Internet, le problème de contrôle d'erreurs est de nature différente car le réseau est sujet à la congestion, ce qui va entrainer la perte de paquets entiers de données. Nous décrivons dans ce chapitre les méthodes de correction d'erreurs que nous avons conçues pour adapter H.261 aux contraintes de la transmission sur l'Internet. Ces méthodes sont regroupées en deux classes qui sont décrites dans les deux premières sections. La troisième section présente les choix que nous avons fait dans notre codec logiciel.

Généralités

La perte des paquets représente un des plus grands problèmes de la transmission vidéo sur l'Internet. Plusieurs niveaux de protection sont envisageables, la protection minimale étant la possibilité de resynchroniser le récepteur à la suite d'une perte de paquets.

De manière générale, plus on réduit la redondance du signal, plus l'information transmise est significative et donc plus la conséquence des erreurs sur la restitution de l'image devient grave. C'est pour cette raison que le codage inter-image est beaucoup plus sensible à la perte de paquets que le codage intra-image. En effet, l'impact sur l'image se propage dans le temps jusqu'à ce que la partie altérée soit rafraîchie en codage intra-image. L'impact est encore plus grand lorsque la compensation en mouvement est utilisée: l'erreur se propage alors aussi bien dans le temps

que dans l'espace, c'est-à-dire vers les macroblocs voisins. Notons qu'en attendant que l'erreur soit corrigée, il est possible de geler l'image du récepteur. Cependant cette méthode rompt le rythme continu de décodage d'images et ne doit être utilisée que dans les cas où l'on ne tolère aucune erreur à l'affichage.

4.1 Méthode à base de feed-back

La méthode de correction la plus rapide consiste à demander un rafraîchissement d'image dès qu'une perte de paquet est constatée. Cependant, en raison des contraintes temps-réel de la vidéoconférence, il n'est pas souhaitable de demander la retransmission des paquets perdus. Des essais réalisés avec le protocole de transport fiable TCP montrent que les délais engendrés par les retransmissions ne sont pas acceptables [Huitema92]. En revanche, si le récepteur a la possibilité de demander à l'émetteur de rafraîchir une zone d'image dès qu'il constate une perte de paquet, l'émetteur pourra alors forcer un codage intra-image de cette zone dans le codage de la prochaine image. La souplesse d'un codec logiciel permet d'implémenter assez facilement ce genre de schéma correcteur d'erreurs: il suffit que le codeur conserve en mémoire la date de dernier encodage de chaque macrobloc de l'image.

La figure 4.1 montre l'interaction entre émetteur et récepteur. On peut noter les deux flots de données qui sont transmis sur l'Internet: les données H.261 circulent du codeur (plus précisément du module de découpage en paquets) vers le décodeur (module de réassemblage des paquets) tandis que les données de feed-back circulent en sens inverse, c'est-à-dire du décodeur (module de détection de perte de paquet) vers le codeur (module de décision de codage en mode inter-image ou intra-image).

L'exemple de la figure 4.2 illustre le protocole de rafraîchissement d'image pour une perte de paquet. Lorsque le récepteur s'aperçoit que le paquet (p+1) est perdu, il envoie à l'émetteur un NACK (*Negative ACKnowledgement*) indiquant que la perte a eu lieu pour les données du GOB 3 de l'image acquise au temps t1. L'émetteur force alors en codage intra-image tous les macroblocs du GOB 3 qui ont été codés depuis le temps t1 dans l'image suivante acquise au temps t2. Ainsi, le rafraîchissement obligatoire intra-image est limité aux seuls macroblocs touchés par la perte de paquet, ce qui réduit le débit nécessaire au rétablissement de l'image. L'effet de la perte d'un paquet sur l'image suivi d'un rafraîchissement intra-image est montré à la figure 4.3.

Dans le cas d'un codec cablé, une telle sélectivité n'est pas possible. On peut

néanmoins demander un rafraîchissement intra-image de toute l'image. Toutefois,

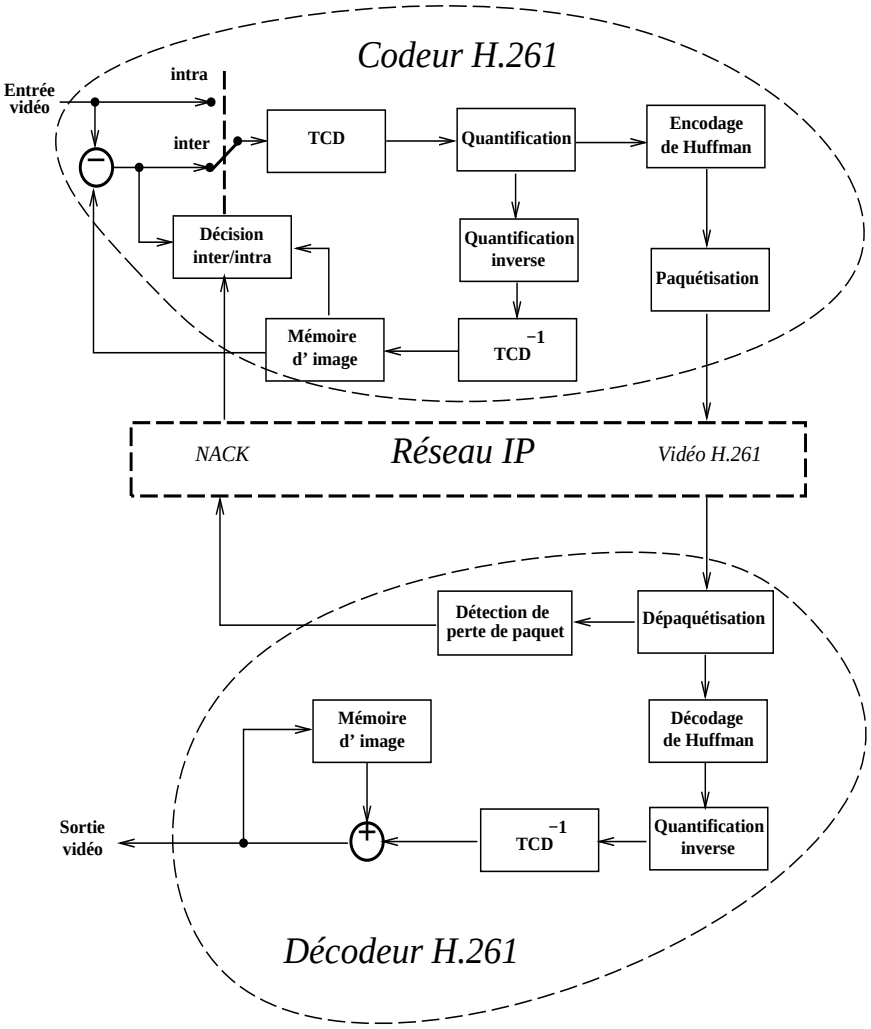


FIG. 4.1 - *Interaction entre émetteur et récepteur*

l'interaction entre récepteur et émetteur n'est possible que si le nombre de récepteurs, c'est-à-dire la taille de la conférence, n'est pas trop grand. En effet, lorsque le nombre de récepteurs est important, le feed-back des récepteurs vers l'émetteur peut engendrer une congestion du réseau connue sous le nom "d'implosion de feedback" [Yavatkar93]. Certains protocoles de transport multipoints fiables sont aussi basés sur l'utilisation de NACKs [Dabbous93] [Diot94a] [Garcia91]. Une manière d'éviter le problème "d'implosion de feedback" consiste à attendre une période aléatoire de

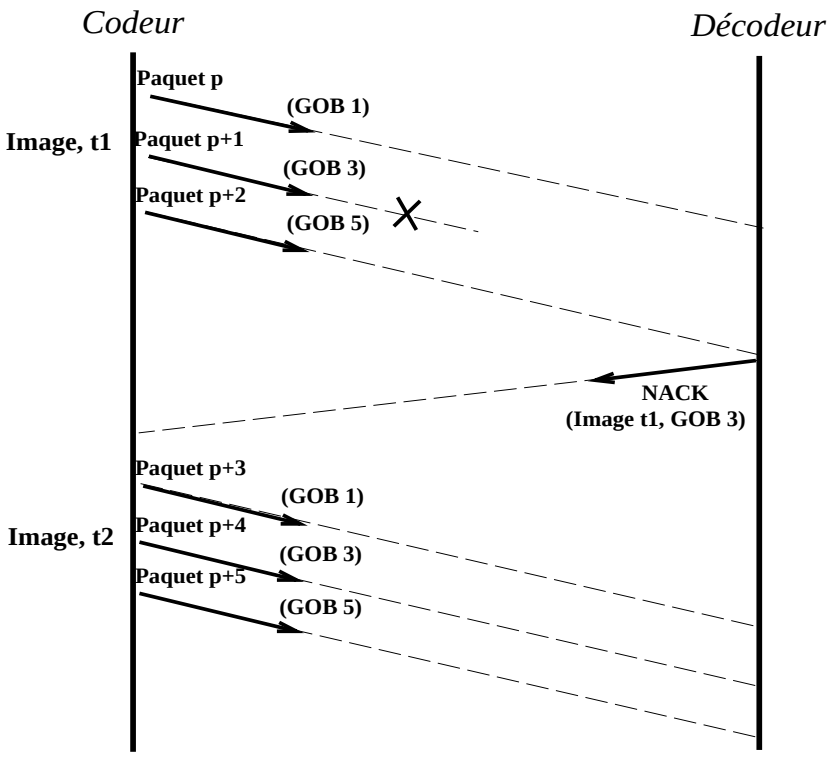


FIG. 4.2 - *Demande de rafraîchissement intra-image après une perte de paquet*

temps avant d'émettre un NACK. Dans ce cas, en transmettant en multipoint les NACKs, chaque récepteur peut vérifier si au bout du délai attendu, un autre récepteur n'a pas déjà émis le même NACK [Pingali94]. Cependant, cette méthode introduit un délai au mécanisme de correction d'erreur qui va à l'encontre de l'objectif de cette méthode qui est d'accélérer le rafraîchissement de l'image après une perte de paquet.

4.2 Méthodes renforçant la robustesse du codage

D'autres méthodes ont pour but de rendre le codage moins sensible à la perte des paquets. Elles consistent à renforcer le maillon du codeur H.261 qui est le plus sensible à la perte de paquets, c'est-à-dire, le codage inter-image. On peut distinguer deux classes de méthodes: les mécanismes à base d'ajout de redondance et le mécanismes qui rafraîchissent périodiquement la vidéo en mode intra-image.

FIG. 4.3 - *Rafraîchissement intra-image à la suite d'une perte de paquet*

4.2.1 Avec ajout de redondance

On peut constater que dans l'Internet, les pertes des paquets qui sont émis à intervalles réguliers sont le plus souvent isolées [Bolot93]. L'expérience suivante le montre pour un exemple de communication point à point entre Londres et Sophia Antipolis à des heures différentes de la journée: vers 8h00, lorsque le trafic sur le réseau Internet est assez faible (moins de 3 % de pertes) et vers 16h00, où le trafic est beaucoup plus important (environ 10 % de paquets perdus). Dans cet exemple, les paquets sont émis à intervalle régulier toutes les 20 ms et ont une taille fixe de 400 octets.

Les figures 4.4 et 4.5 représentent dans les deux cas, le taux de paquets consécutivement perdus pour 10 000 paquets transmis ainsi que la distribution du nombre de paquets consécutivement perdus. Les courbes de distribution du nombre de pertes consécutives sont en échelle logarithmique de manière à montrer la décroissance quasi exponentielle du processus de perte. On observe dans les deux expérimentations que dans plus de 75 % des cas, la perte de paquet est isolée, c'est-à-dire on ne perd pas plusieurs paquets à la suite. Dans de rares cas, on peut noter des salves de paquets perdus, e.g. 20 ou 30 paquets à la suite sont perdus. Par ailleurs, ce phénomène est souvent lié à un problème [1] dans le protocole d'échange des tables de routage entre routeurs. Dans la majorité des cas, on s'aperçoit que, si l'on corrige

1. Il arrive assez souvent que les routeurs se synchronisent entre eux, ce qui produit périodiquement des salves importantes de trafic dans le réseau [Floyd93a].

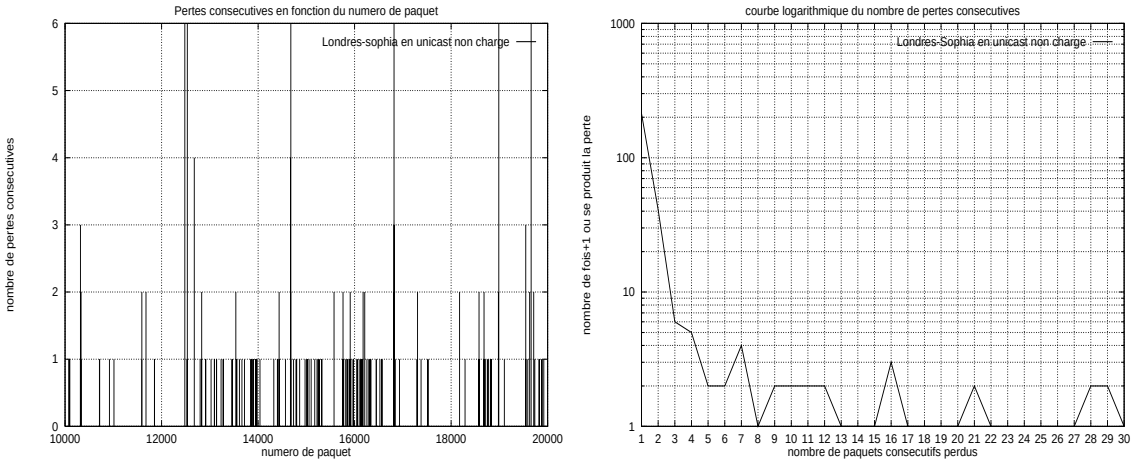


FIG. 4.4 - *Pertes consécutives pour un réseau non chargé*

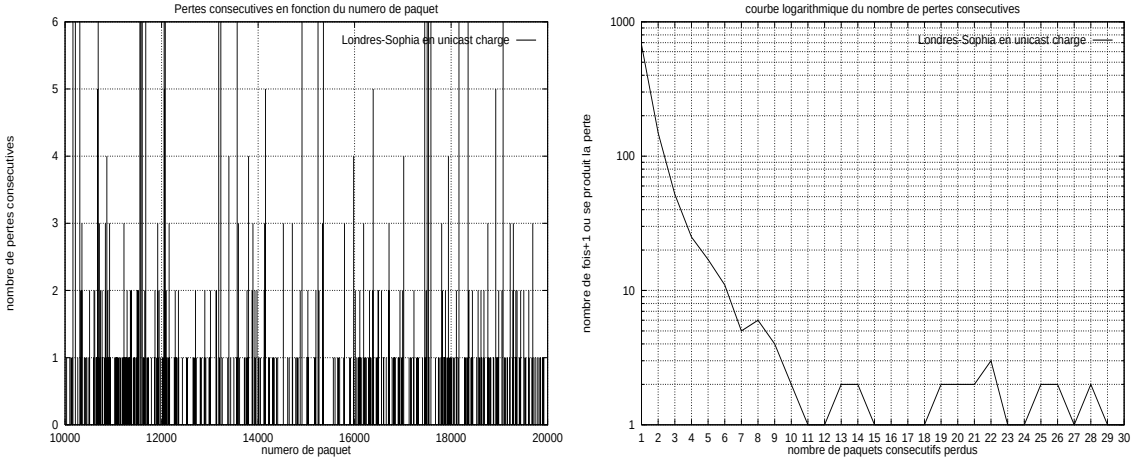


FIG. 4.5 - *Pertes consécutives pour un réseau chargé*

la perte isolée des paquets, on parvient à réduire considérablement la sensibilité du codage aux erreurs.

Pour corriger la perte isolée de paquets, il suffit d'inclure dans chaque paquet de l'information qui permet de reconstruire plus ou moins fidèlement le paquet précédemment émis.

Dans le cas extrême, on peut imaginer la duplication de l'émission de chaque paquet dans le même flot, e.g. en émettant les paquets dans l'ordre (N, N, N+1, N+1, N+2, N+2, ...). Cette méthode très simple à implanter a bien sûr l'inconvénient majeur d'utiliser deux fois plus de bande passante.

Une méthode plus coûteuse en calculs mais plus économique en bande passante

consiste à encoder dans l'en-tête de chaque paquet le paquet précédent sous une forme plus comprimée. On a ici tout un éventail de possibilités, voir section 6.2.6. On peut par exemple n'encoder que la partie la plus importante des coefficients de la TCD de chaque bloc ou bien l'ensemble des coefficients mais quantifiés de manière plus grossière. Cette technique permet donc de reconstruire plus ou moins fidèlement le paquet précédent si ce dernier a été perdu lors de la transmission sur le réseau. Cependant, le nombre de codages successifs en mode inter-image doit être suffisamment faible pour ne pas entraîner de trop grandes accumulations d'erreurs dans la reconstitution des paquets suivants. En effet, pour encoder des images en mode inter-image, un codeur H.261 suppose que le décodeur dispose d'une image de même résolution que lui. Ce n'est plus le cas à la suite d'une reconstruction imparfaite d'un paquet perdu et des erreurs vont alors s'accumuler jusqu'à ce que les blocs d'image soient recodés en mode intra-image.

On peut enfin envisager l'utilisation d'un flot différent pour transmettre l'information redondante. Dans ce dernier cas, seuls les récepteurs qui rencontrent des problèmes de congestion auront intérêt à s'abonner au flot.

4.2.2 Avec rafraîchissement intra-image périodique

La deuxième classe de solutions, plus simple dans sa réalisation, consiste à rafraîchir périodiquement l'image en mode intra-image. Le but n'est pas de reconstruire les paquets perdus mais de réduire la durée d'impact de la perte de paquets sur l'image.

La recommandation H.261 spécifie que pour limiter l'accumulation d'erreurs liées à la transformation inverse, chaque macrobloc doit être mis à jour en mode intra-image au moins une fois toutes les 132 fois où il est émis. Notons que le choix du codage intra-image/inter-image se fait à la couche macrobloc et non à la couche bloc. En effet, un codage intra-image de macrobloc implique un codage intra-image de tous les blocs contenus dans le macrobloc. En cas de perte de paquet, ce rafraîchissement n'est pas assez rapide. Il est possible avec un codec logiciel de changer la fréquence de rafraîchissement intra-image en fonction de l'état du réseau. Dans la pratique, plus le réseau est congestionné, c'est-à-dire le taux de perte des paquets est élevé et plus la fréquence de codage intra-image des macroblocs doit être grande. Dans la plupart des codecs cablés, il est toutefois possible de forcer périodiquement un codage complet en mode intra-image de l'image. Dans des cas extrêmes de congestion, il est conseillé d'utiliser le codage intra-image exclusivement.

4.3 Choix de la méthode de contrôle d'erreur

Le choix de la méthode de protection doit se faire en fonction du degré de souplesse du codec ainsi que du nombre de participants dans la conférence. Faute de temps, nous n'avons pu mettre en œuvre que deux méthodes, à savoir celle à base de feed-back et celle qui consiste à refraîchir périodiquement l'image en mode intra-image. Les méthodes à base d'ajout de redondance sont en cours d'implantation. Elles ont pour avantage de fonctionner indépendamment du nombre de récepteurs dans la conférence. Les inconvénients majeurs sont le sur-coût de bande passante nécessaire, les accumulations d'erreurs et l'incompatibilité avec les codecs standards H.261.

IVS étant un codec logiciel autorise l'utilisation des NACKs. Il permet donc une correction rapide en cas de perte de paquets à condition que le nombre de récepteurs soit faible. Nous avons fixé ce nombre à 10. Au delà, l'utilisation des NACKs est automatiquement inhibée et l'on utilise la méthode à base de rafraîchissement périodique de l'image en mode intra-image. Le nombre maximal de codages successifs en mode inter-image est fonction du taux de perte de paquets constaté sur le réseau. La manière d'obtenir l'état du réseau est discutée en section 6.2.1. Lorsque l'état du réseau est NON-CHARGE (ce qui correspond à un faible taux de paquets perdus), le nombre maximal de codages successifs en mode inter-image est fixé à 20. Ce nombre passe à 5 dans le cas d'un réseau CHARGE et à 0 (c'est à dire que lorsque le taux de perte de paquets est élevé, on n'utilise que le mode intra-image) dans le cas d'un réseau CONGESTIONNE.

Une précaution supplémentaire consiste à forcer périodiquement le codage de chaque bloc indépendamment de la détection de mouvement. On protège de cette manière les blocs d'image qui ne sont pas encodés souvent (par exemple l'arrière plan de l'image). Le rythme minimal de codage obligatoire est, de la même manière que le nombre maximal de codages successifs en mode inter-image, fonction de l'état du réseau. On associe à un réseau NON-CHARGE, un codage obligatoire des blocs toutes les 100 images. Ce nombre est de 70 pour un réseau CHARGE et de 30 pour un réseau CONGESTIONNE.

4.4 Conclusion

Dans ce chapitre, nous avons proposé plusieurs méthodes de contrôle d'erreurs qui conviennent à l'émission de données H.261 sur l'Internet [Turletti93b]. Nous les

avons regroupées en deux classes, les techniques à base de feed-back et les techniques qui renforcent la robustesse du codage. Nous avons motivé notre choix et décrit celles utilisées dans IVS.

Chapitre 5

Le contrôle de débit

Nous avons vu en section 2.1.3 que le débit généré par un codec H.261 est intrinsèquement variable et dépend en majeure partie de la quantité de mouvement et des détails dans la séquence d'images à encoder. Dans cette section, nous allons chercher à contrôler ce débit de manière à pouvoir l'ajuster en fonction des capacités disponibles du réseau. Le schéma d'un codeur H.261 est repris par la figure 5.1.

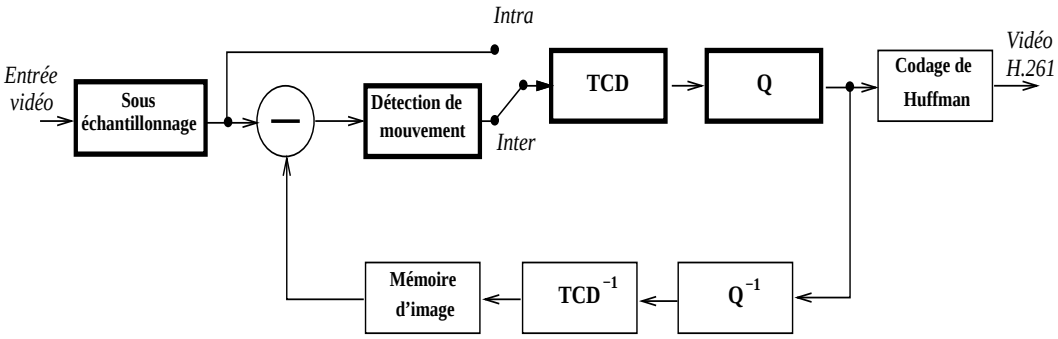


FIG. 5.1 - *Synoptique d'un codeur H.261*

Le débit d'un codeur H.261 peut être ajusté de plusieurs façons. Les procédés utilisés peuvent être regroupés en deux catégories selon qu'ils agissent sur le rythme de rafraîchissement ou sur la définition de l'image. Certains types d'applications préfèrent conserver une haute définition de l'image plutôt qu'un rythme d'image élevé, c'est le cas par exemple de la transmission de texte ou d'image bio-médicale. En revanche, d'autres applications nécessitent un rythme de rafraîchissement d'image important de manière à pouvoir suivre le mouvement des mobiles dans l'image. Nous détaillons dans la suite ces deux catégories de procédés.

5.1 Méthode favorisant la qualité d'image

La méthode la plus simple pour réduire le débit vidéo consiste à diminuer le rythme de rafraîchissement d'image. En effet, le débit à la sortie du codeur est proportionnel au nombre d'images encodées par seconde.

Le principe est très simple: on fait en sorte pour que l'intervalle de temps entre l'image courante à transmettre et la précédente soit tel que le débit reste inférieur à la valeur maximale de débit fixée V_{max}. Par exemple, pour l'image N de taille T bits, on choisira un délai t tel que $t \geq T/V_{max}$. Selon le type de carte d'acquisition d'image, on peut choisir soit d'augmenter la période d'acquisition d'image soit de n'encoder qu'une image sur N si la cadence d'acquisition ne peut pas être modifiée.

Avec une même valeur V_{max}, le rythme de rafraîchissement d'image varie en fonction de la quantité d'information à encoder par image. Ainsi, avec ce mode de fonctionnement, on constate que plus il y a de mouvement dans l'image et plus le rythme de rafraîchissement d'image est réduit.

5.2 Méthodes favorisant le rafraîchissement d'image

Lorsque le rythme de rafraîchissement d'images est plus important que la définition de l'image, il est préférable d'agir sur un ou plusieurs des paramètres suivants.

5.2.1 Le seuil de détection de mouvement

On a vu en section 2.1.3 qu'un bloc d'image est encodé si la différence entre ce bloc à l'instant T et ce même bloc à l'instant $(T - dt)$ est supérieure au seuil de détection de mouvement. Le seuil de détection de mouvement agit donc directement sur le nombre de blocs à coder dans l'image. Puisque le débit à la sortie du codeur vidéo est proportionnel au nombre de blocs encodés par image, élever le seuil de détection de mouvement revient à réduire le débit à la sortie du codeur. Augmenter le seuil de détection de mouvement a pour effet de décroître la sensibilité du codeur au mouvement. En effet, si le seuil de détection de mouvement a une valeur trop élevée, les blocs d'image au voisinage du mobile, voire même les contours du mobile risquent de ne pas être codés. A l'inverse, choisir une valeur de seuil trop faible peut entraîner le codage de zones d'image qui n'ont subi aucun changement significatif. Les graphes 5.2 montrent le rapport signal sur bruit (*Signal to Noise Ratio* (SNR)) et le nombre de bits par image pour trois valeurs de seuil de détection de mouvement.

La figure 5.3 montre l'influence du seuil de détection de mouvement pour deux

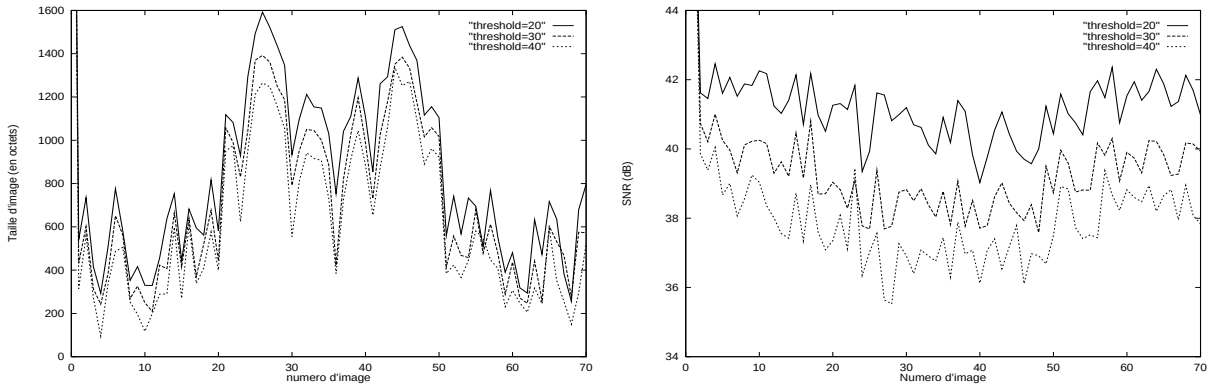


FIG. 5.2 - *Evolution de la taille d'image et du SNR en fonction du numéro d'image pour 3 valeurs du seuil de détection de mouvement*

valeurs de seuil (20 à gauche et 40 à droite). On peut remarquer que le contour du visage est moins régulier, tout comme la coiffure (particulièrement en haut à droite). L'image de gauche est plus fidèle car elle suit plus fidèlement le mouvement dans la scène, ce qui se traduit par davantage de blocs encodés par image. Notons aussi qu'en

FIG. 5.3 - *Influence du seuil de détection de mouvement*

diminuant le nombre de blocs encodés par image, on réduit en proportion le nombre de cycles cpu nécessaires pour encoder l'image. Par conséquent, cette méthode de reduction de débit peut engendrer (dans le cas où la puissance de la machine était le goulôt d'étranglement) une augmentation de la fréquence d'encodage d'images

et peut donc indirectement contribuer à une augmentation du débit de sortie du codeur.

5.2.2 Les coefficients de la TCD

Une autre solution consiste à n'encoder qu'une partie des 64 coefficients des blocs d'image. La TCD se fait sur des blocs de 8×8 coefficients. On peut choisir de ne conserver qu'une partie des coefficients transformés de manière à réduire le nombre de coefficients encodés par image et donc le débit du codeur. Ainsi, on peut se contenter de n'encoder que les coefficients BF du bloc transformé, réduisant de la même manière le coût de calcul par bloc. Pour illustrer cette solution, on a choisi d'encoder la séquence test 'Miss America" successivement en ne conservant que la composante continue, puis 2×2, 3×3, 4×4 et 8×8 coefficients des blocs. La séquence d'images est encodée au format CIF en mode intra-image pour les quatre types de transformée. La qualité d'image obtenue en n'encodant que la composante continue ou que 2×2 coefficients est mauvaise et donc inacceptable dans la pratique. Les graphes 5.4 montrent le rapport signal sur bruit (*Signal to Noise Ratio* (SNR)) et le nombre de bits par image pour les trois autres types de transformées. La figure 5.5 montre les effets pour une image de la séquence de test. On note nettement un "effet de blocs" dans l'image à 2×2 coefficients. Au pire avec un seul coefficient BF (le coefficient DC), on aurait une mosaïque de blocs uniformes dont seule la couleur où le niveau de gris serait différent.

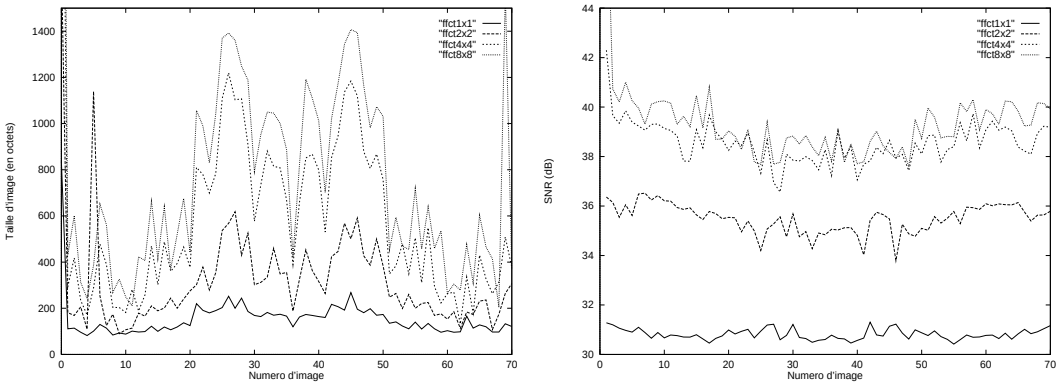


FIG. 5.4 - *Influence des coefficients de la TCD*

Notons qu'il est aussi possible de prendre un autre critère de sélection des coefficients à encoder, comme par exemple choisir les N coefficients de plus grande énergie de chaque bloc, ou tous les coefficients d'énergie supérieure à un seuil donné. Ces

FIG. 5.5 - *Image encodée avec TCD* 4×4 *et une TCD* 2×2

deux dernières méthodes correspondent aux techniques dites de "distorsion minimum" et de "seuillage d'énergie" présentées dans la section 6.2.6.

5.2.3 La quantification

L'étape de quantification agit sur la précision du codage des coefficients transformés. Plus le pas de quantification est élevé et moins il faut de niveaux différents pour coder les coefficients. De plus, les coefficients de faible valeur vont s'annuler après quantification. Ainsi, à l'étape suivante de codage des coefficients, on réalise une économie de bits. Les graphes 5.7 montrent les tailles d'image et le SNR pour trois valeurs de quantificateurs: 3, 7 et 11. On observe une différence de 2 dB entre la séquence encodée avec un quantificateur 3 et celle encodée avec un quantificateur 7 et la même différence avec les quantificateurs 7 et 11. On peut aussi noter que la variabilité de la taille d'image est beaucoup plus importante avec un quantificateur faible qu'avec un quantificateur plus élevé. En effet, la taille d'une image vaut le produit du nombre de blocs encodés par la taille moyenne d'un bloc. Comme la taille moyenne d'un bloc encodé avec un quantificateur 3 est environ le double de celle obtenue avec un quantificateur 7, cela revient à doubler la variabilité de la taille d'image (d'une taille allant de 100 à 600 octets pour la séquence Q=7, on passe à une taille comprise entre 200 et 1300 octets environ pour la séquence Q=3). La figure 5.6 montre les effets d'une quantification grossière sur l'image. Les contours de l'image sont irréguliers et les blocs d'image sont légèrement "piquetés". Cepen-

FIG. 5.6 - *Image CIF encodée avec un quantificateur de 3 (à gauche) et de 11 (à droite)*

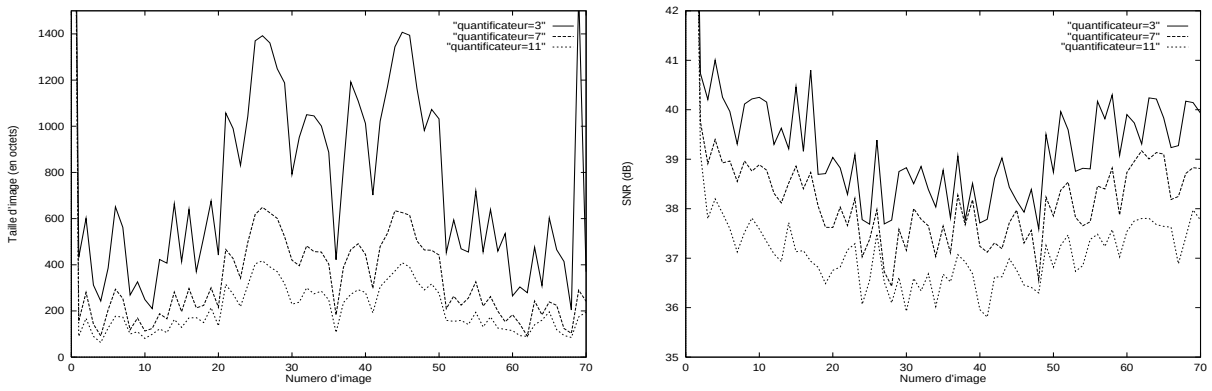


FIG. 5.7 - *Evolution de la taille d'image et du SNR en fonction du numéro d'image pour 3 valeurs de quantificateur*

dant, si l'on compare avec la méthode précédente, on remarque qu'avec un taux de réduction de débit similaire la définition d'image est nettement supérieure.

Remarque générale

Dans toutes les courbes qui montrent le SNR par rapport au numéro d'image, on peut remarquer que la première valeur du SNR est toujours supérieure aux valeurs suivantes. Ceci peut être expliqué en raison du mode d'encodage intra-image utilisé pour la totalité des blocs de la première image. La variabilité du SNR dans les

images suivantes est fonction du taux d'encodage intra-image/inter-image.

De même, on peut observer que la taille de la première image est toujours plus élevée que celle des images suivantes. En effet, la séquence test de vidéoconférence représente une personne en premier plan qui discute. Le fond de l'image, c'est-à-dire le décor derrière la personne, n'a besoin d'être encodé que rarement (pour la première image et autour de la tête de la personne pour les images suivantes).

5.3 Choix de la méthode de contrôle de débit

Nous venons d'identifier deux classes de méthodes de contrôle de débit, à savoir une méthode qui favorise le rafraîchissement d'image au détriment de sa qualité et une autre qui a les effets contraire. Nous avons laissé l 'utilisateur libre de choisir le type de la méthode de contrôle de débit en fonction du type de l'application qu'il envisage. Par exemple, dans le cas d'une transmission de séminaire, on préfère choisir la méthode qui favorise la qualité d'image pour retransmettre les transparents d'une conférence et celle qui privilégie le rafraîchissement d'image pour retransmettre le conférencier.

La méthode qui favorise la qualité d'image ne pose aucun problème particulier quant à sa mise en œuvre. Il suffit simplement de retarder l'acquisition de l'image suivante afin que le débit de sortie du codeur reste inférieur ou égal à la valeur maximale du débit souhaitée.

En revanche, la méthode qui privilégie le rafraîchissement d'image est plus complexe à mettre en œuvre. En effet, on a vu que plusieurs paramètres peuvent être utilisés pour contrôler la qualité d'image. Or, plus un schéma de contrôle comporte de paramètres et plus son élaboration est délicate. Modifier à la fois le quantificateur et le nombre de coefficients serait une erreur car on obtiendrait une trop grande diminution de la définition de l'image. Nous avons opté pour l'utilisation du quantificateur car il donne de meilleurs résultats que la méthode à base de réduction des coefficients de TCD, voir comparaison entre les images 5.5 et 5.6.

Le seuil de détection de mouvement est avantageux car son utilisation ne provoque pas de perturbations supplémentaires au niveau des blocs encodés. En revanche, les contours des objets mobiles de l'image vont être plus ou moins affectés, ce qui n'est pas trop grave dans ce mode de fonctionnement puisque la fréquence de rafraîchissement d'image est élevée.

L'algorithme de contrôle utilise donc les paramètres quantificateur et seuil de

détection de mouvement. Le seuil de détection de mouvement est utilisé lorsque l'utilisation du plus grand quantificateur ne suffit pas à réduire suffisamment la taille de l'image. Dans ce cas, on augmente la valeur du seuil de détection de mouvement afin de réduire le nombre de blocs à encoder et par conséquent la taille d'image.

La phase de détection de mouvement est réalisée avec une valeur de seuil S prise par défaut. Dans cette phase, on associe à chaque bloc deux valeurs qui sont la somme des différences ainsi que l'indicateur de marquage qui correspondant au résultat de la comparaison de la somme avec le seuil S. On dispose alors des données suivantes:

- d_{max} le débit maximal autorisé,
- N le nombre de blocs à encoder pour la nouvelle image calculé avec la valeur de seuil S,
- dt l'intervalle de temps entre l'encodage de cette nouvelle image et celui de l'image précédente au même endroit de l'algorithme de compression.

L'algorithme nécessite pour fonctionner une estimation du nombre moyen de bits par bloc encodé ($B(Q)$) et ce pour chacun des quantificateurs utilisables. Ces estimations sont initialisées dans un premier temps à des valeurs par défaut et sont mises à jour à chaque image.

On calcule ensuite la taille maximale T_{max} d'image qu'il nous est possible d'émettre à cet instant ainsi que la taille de bloc B_{max} cible pour encoder cette image.

$$T_{max} = dt \,.\, d_{max}\,; \quad B_{max} = \frac{T_{max}}{N} \tag{5.1}$$

L'algorithme cherche alors le meilleur quantificateur, ou plus exactement la plus petite valeur qui est permise, pour encoder l'image. La valeur du seuil S est modifiée dans les deux cas suivants:

- l'utilisation du plus grand quantificateur ne suffit pas à réduire suffisamment la taille de l'image. Dans ce cas, on "marque" à nouveau les blocs avec une valeur de seuil S plus élevée et on réitère l'algorithme.
- l'utilisation du quantificateur le plus faible génère une taille d'image inférieure à la taille d'image maximale permise. Dans ce cas, on "marque" à nouveau les blocs avec une valeur de seuil S plus faible.

Le coût de calcul du "marquage" est faible car le calcul de la sommation des différences entre l'image courante et l'image précédente a déjà été réalisé dans la phase de détection de mouvement. Il suffit d'effectuer une comparaison pour chaque bloc et selon les cas une affectation supplémentaire correspondant à l'opération de "marquage".

Algorithme de sélection des paramètres Q et S pour l'image courante

```
do{
   for (Q = Q_min; Q <= Q_max; Q++){
      if(B(Q) <= B_max)
         break;
   }
   if (B(Q) > B_max){
      S -= 10;
      Marquage(blocs, S, N);
      B_max = (dt * d_max) / N;
      continue;
   }
   if (B(Q) < B(Q_max))
      S += 10;
      Marquage(blocs, S, N);
      B_max = (dt * d_max) / N;
      continue;
   }
} while (true);
```

5.4 Conclusion

Evaluation du codage H.261

Nous venons de voir différentes techniques pour ajuster le débit d'un codeur H.261. Ces méthodes permettent de diminuer le débit jusqu'à quelques kbps. La valeur maximale de débit obtenue dépend de la puissance de calcul disponible de la station de travail et de la fréquence d'acquisition admissible de la carte d'acquisition vidéo. Avec les plate-formes actuelles le débit peut atteindre les 300 kbps avec

25 images par seconde dans le format QCIF. Avec le format CIF, le principal goulot d'étranglement est la limitation de la puissance de la machine: lorsque l'image contient beaucoup de mouvement, on constate que la fréquence d'encodage diminue de moitié. Si la montée en puissance des machines se poursuit au même rythme que ces dix dernières années, on pourra obtenir dans moins de deux ans le maximum de 2 Mbps prévu par le standard de compression H.261. Le principal intérêt du codage H.261 est qu'il permet d'obtenir un taux de compression élevé et facilement ajustable (environ de 10 à 100). La TCD est coûteuse en calcul mais elle permet, en plus d'obtenir un taux de compression élevé, de réaliser assez facilement du codage en sous-bandes (voir les méthodes décrites dans la section 6.2.6).

Résumé des contributions

Dans ce chapitre, nous avons regroupé les méthodes de contrôle de débit pour codeurs H.261 en deux classes, selon qu'elles favorisent la qualité ou le rafraîchissement d'image. Notons que la mise en œuvre de ces méthodes, surtout si le codec est commercialisé, n'est en général pas divulguée par les constructeurs. Nous avons ensuite présenté l'algorithme de contrôle de débit qui a été mis en œuvre dans notre logiciel de codeur vidéo H.261.

Chapitre 6

Le contrôle de congestion

6.1 Etat de l'art

La mise au point des mécanismes de contrôle de congestion et de gestion des ressources fait partie des problèmes les plus délicats à résoudre dans le domaine des réseaux. Les approches pour résoudre ce problème diffèrent en fonction du type de réseau que l'on utilise. Dans cette section, nous passons en revue les mécanismes de congestion utilisés pour les deux grandes classes d'architecture de réseau: les réseaux à *commutation de circuits* et les réseaux à *commutation de paquets* (ou *datagrammes*).

6.1.1 Réseaux à commutation de circuits

Les réseaux à commutation de circuits comme le réseau RNIS et le réseau téléphonique utilisent des liaisons de capacités fixes qui sont allouées lors de la demande de connexion. Un protocole de contrôle d'admission de connexion s'assure lors de l'établissement de la connexion que suffisamment de bande passante est disponible pour le service demandé. Le problème de congestion du réseau ne se pose donc pas: si la demande est acceptée, un chemin virtuel est crée de la source vers le récepteur avec une bande passante et un délai de "bout en bout" garantis pendant toute la durée de la transmission. Dans le cas où il n'y a pas assez de bande passante, la demande de connexion est rejetée, et il ne reste plus à l'utilisateur qu'à renouveler sa demande plus tard (comme dans l'exemple du réseau téléphonique). Les réseaux à commutation de circuits offrent des garanties de ressources (par exemple, une bande passante et/ou un délai maximal de transmission garantis). Cependant, ce type de réseau comporte quelques inconvénients. La commutation de circuits peut

conduire à une très forte sous-utilisation des liaisons lorsque le trafic généré par la source n'est pas constant, comme dans le cas de la transmission vidéo à débit variable. En effet, le canal réservé doit pouvoir supporter le trafic crête du codeur, ce qui entraîne la plupart du temps un gaspillage de la bande passante du réseau. Cependant, il est possible de réduire le taux de bande passante réservé en lissant préalablement les données à la sortie du codeur via un contrôle local [Gilge91] (figure 6.1). Une mémoire placée à la sortie du codeur vidéo joue le rôle de tampon. Elle envoit un feed-back au codeur pour qu'il ajuste son débit de manière à ce qu'il soit constant à la sortie de la mémoire. D'autre part, le service de retransmission

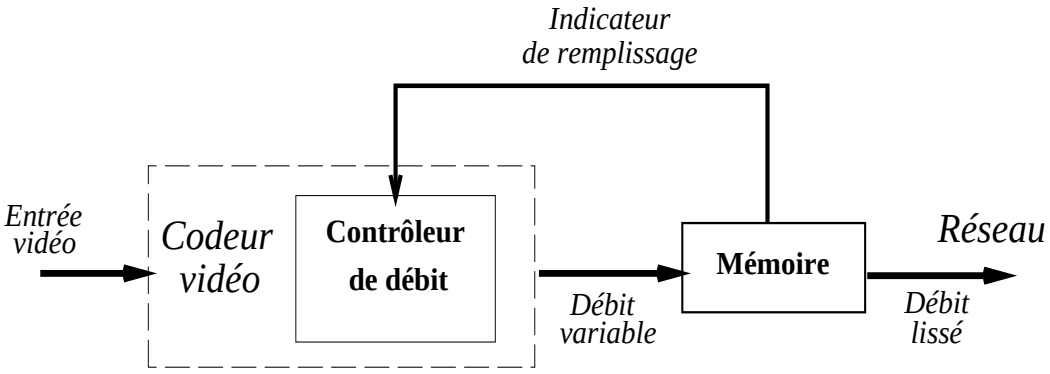


FIG. 6.1 - *Lissage du débit vidéo via un contrôle local*

automatique en cas d'erreur, ne convient pas à ce type de trafic de données car il engendre des délais qui sont incompatibles avec les contraintes temps réel requises par l'application. Enfin, la transmission multipoint vers un nombre quelconque de récepteurs n'est pas envisageable: elle ne peut se faire que par l'intermédiaire de canaux multiples qui réduisent l'efficacité de la transmission.

6.1.2 Réseaux à commutation de paquets

La deuxième classe d'architecture est la classe des réseaux à commutation de paquets. Nous nous bornons dans ce mémoire à l'étude de l'Internet car tous nos travaux ont été réalisés sur ce type particulier de réseau de datagrammes. En fait, l'Internet n'est pas vraiment un "réseau" au sens classique du terme. Il représente plutôt un réseau de réseaux, c'est à dire une interconnexion complexe de réseaux qui appartiennent à différents propriétaires [Huitema94].

L'architecture du modèle de communication Internet est très différente de celle des réseaux à commutation de circuits. Le concept même de connexion n'existe pas:

chaque paquet est transporté indépendamment des autres. Tous les contrôles ont lieu de "bout en bout", c'est-à-dire que les routeurs intermédiaires n'interviennent pas. Il n'y a pas d'établissement de circuit virtuel, les adresses sources et destination sont présentes dans tous les datagrammes IP, ce qui permet à ces datagrammes d'emprunter des chemins différents. Il n'y a aucune garantie sur le fait que les paquets atteindront bien leur destination, pas plus qu'ils ne l'atteindront dans l'ordre. Pour obtenir un transport fiable, les paquets doivent être acquittés par un protocole de transport de "bout en bout" en utilisant par exemple, le protocole TCP. En cas de non acquittement d'un paquet, l'émetteur doit réémettre le paquet non acquitté qui devra à nouveau traverser tout le réseau (les routeurs intermédiaires n'en gardant pas de copie). Les paquets peuvent aussi être reçus hors séquence ou avoir été dupliqués voire altérés par la transmission. Le protocole de transport TCP assure la remise en séquence et la retransmission des paquets en cas d'erreurs de transmission [1].

Lorsqu'un paquet arrive à un routeur (ou noeud intermédiaire) entre la source et la destination, il attend son tour dans une file d'attente avant d'être transmis. Jusqu'à présent, les files d'attente sont gérées selon une logique PAPS "Premier Arrivé - Premier Servi", (ou *FIFO "First In - First Out"* en anglais). De cette manière, les ressources sont utilisées avec une efficacité maximale: les liaisons sont utilisées dès que les paquets sont disponibles. Cependant, les connexions ne sont pas indépendantes les unes des autres, car le comportement d'une connexion affecte celui des autres connexions avec lesquelles elle partage des ressources du réseau. Il n'est donc pas possible dans l'Internet actuel d'obtenir des garanties de ressources, le réseau se contentant d'acheminer les paquets du mieux qu'il peut. Ce type de service est connu sous le nom du "meilleur effort possible" (ou "*best effort*"). Une conséquence naturelle de cette architecture est que le réseau peut être sujet à des phénomènes néfastes comme la congestion. En effet, lorsque la demande des utilisateurs excède la capacité du réseau, on observe un phénomène de saturation qui se traduit par une dégradation des performances globales du réseau. Les délais de "bout en bout" ainsi que le taux de perte des paquets augmentent alors rapidement, dégradant ainsi le débit effectif du réseau. Pour éviter ce problème, des mécanismes de contrôle de transmission sont nécessaires pour limiter le taux d'utilisation des ressources du

1. Le taux moyen d'erreurs dûes au médium est relativement faible: on l'estime à 10^{-7} pour une transmission via un faisceau hertzien et à 10^{-9} pour une transmission par l'intermédiaire de fibres optiques.

réseau en contrôlant le débit des paquets émis par les utilisateurs.

Traditionnellement, l'Internet a été utilisé pour des transmissions de données classiques à savoir les transferts de fichiers (via le protocole FTP [RFC959]), les terminaux distants (via le protocole TELNET [RFC854]) ou la messagerie électronique (via le protocole SMTP [RFC821]). Ces applications peuvent facilement utiliser le service proposé par l'Internet car elles sont de nature élastique, c'est-à-dire qu'elles tolèrent assez facilement des variations importantes de délai de transmission. De plus, elles utilisent des mécanismes de contrôle de congestion qui leur permettent de s'adapter à des pertes de paquets en réduisant leur débit d'émission dès que le réseau est congestionné.

L'arrivée des applications multimédia sur l'Internet modifie les règles du jeu. Ces nouvelles applications ont des besoins très différents des applications traditionnelles. Par exemple, la vidéoconférence nécessite que les délais de "bout en bout" soient minimes pour permettre l'interactivité entre les participants d'une conférence. La voix supporte très mal la gigue de délai. De plus, la retransmission de paquets audio ou vidéo n'est en général pas possible car une telle retransmission augmente de manière considérable les délais de "bout en bout". De plus, il faut faire attention à ce que ces nouvelles applications suivent les rêgles du jeu à savoir qu'elles ne monopolisent pas ressources du réseau. En effet, certaines applications de vidéoconférence (e.g. *VAT, NV*) ont un comportement "antisocial" vis-à-vis des applications traditionnelles de l'Internet: elles émettent leur données à un débit constant, choisi par l'utilisateur et indépendamment de la charge du réseau. Elles n'offrent dans ce cas aucune chance aux applications traditionnelles qui sont en "compétition" avec elles et contribuent fréquemment à congestionner l'Internet. Pour satisfaire ces nouveaux besoins, trois approches sont possibles: on peut adapter les applications au réseau, changer la discipline des routeurs ou encore modifier l'architecture Internet. Ces trois approches sont décrites brièvement ici et seront détaillées dans la suite du chapitre.

Adapter les applications au réseau

La première approche consiste à utiliser l'Internet sans aucune modification, en adaptant les applications aux caractéristiques du réseau. Cette adaptation revient à rendre les applications moins sensibles à la gigue de délai et à introduire des mécanismes de contrôle de congestion. On peut par exemple intégrer à l'application des mécanismes de compensation de gigue aussi appelés mécanismes de "synchronisation intra-média" [Jeffay92], [Ferrari93]. Ces mécanismes sont couramment utilisés

lors de la réception de paquets audio. Schématiquement, ces mécanismes consistent à rajouter explicitement un délai variable à chaque paquet reçu, de manière à ce que l'intervalle de temps entre chaque paquet rejoué soit respecté. On peut aussi adapter le débit de l'application en fonction de l'état du réseau (figure 6.2). Notons

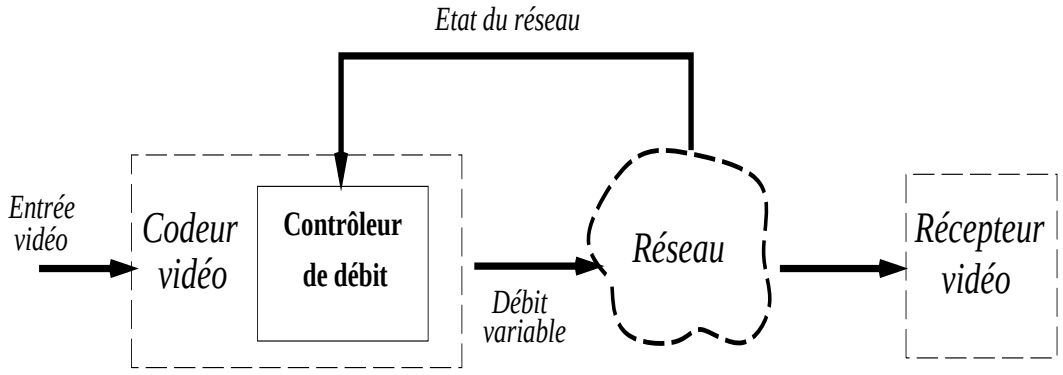


FIG. 6.2 - *Contrôle du débit vidéo en fonction de l'état du réseau*

dès maintenant que cette approche ne permet pas d'empêcher que des applications peu scrupuleuses monopolisent les ressources du réseau.

Modifier la discipline des routeurs

Pour rendre plus équitable le partage des ressources du réseau entre les utilisateurs, une méthode consiste à modifier la discipline des routeurs. Par exemple, on peut remplacer le mécanisme PAPS actuel par le mécanisme "d'attente équitable" (*fair-queuing*), décrit section 6.3.1. Ce mécanisme fait en sorte que chaque utilisateur ait une part équitable de la bande passante du réseau. Une telle modification ne nécessite pas un changement de l'architecture Internet et peut donc être envisagée à moyen terme.

Modifier l'architecture Internet

Avec l'approche précédente, la même classe de service est fournie à chaque utilisateur indépendamment des besoins de l'application et cela sans aucune garantie de service. Une autre approche consiste à changer fondamentalement l'architecture Internet en proposant tout un éventail de services aux utilisateurs (e.g. services avec ou sans garantie de délai, avec ou sans garantie de bande passante, etc...), (section 6.3.2). On peut aussi ajouter un mécanisme de contrôle d'admission afin d'empêcher l'arrivée de nouveaux utilisateurs lorsque le réseau est trop chargé. Ces approches

ne sont pas exclusives. On peut citer par exemple les expérimentations d'application multimédia faites par le groupe Tenet sur le réseau Sequoia 2000 qui combinent les trois procédés à savoir la modification de la discipline au sein des routeurs, la réservation de ressources et le contrôle d'admission afin de garantir une qualité de service excellente [Banerjea94].

Pour le moment, seul le service *best effort* est disponible sur l'Internet et vu l'attraction que suscite la vidéoconférence, il devient urgent de trouver une solution. Nous avons donc opté pour la première approche, à savoir l'adaptation de l'application aux caractéristiques du réseau. Notre approche consiste à inclure dans l'application des mécanismes de contrôle de congestion de "bout en bout". Ces mécanismes permettent d'adapter en temps réel le débit d'émission de l'application aux capacités du réseau et donc de permettre un partage plus équitable de la bande passante (cf. section suivante). Dans ce cas, ce qui est garanti c'est qu'à chaque instant, on aura la meilleure qualité possible selon les ressources disponibles du réseau.

6.2 Solution proposée dans le cas de l'Internet actuel

Pour pouvoir adapter le débit de l'application à l'état du réseau, on doit être en mesure de contrôler le débit de l'application. Ce problème a fait l'objet du chapitre 5. De plus, il faut comme dans toute adaptation de "bout en bout", pouvoir caractériser l'état du réseau sous-jacent afin de déterminer la bande passante disponible pour l'application.

Dans l'Internet, on distingue deux manières de détecter des situations de congestion, soit en demandant explicitement au réseau d'envoyer des messages de signalisation, soit en utilisant des mécanismes de "bout en bout". Dans la première méthode, les routeurs intermédiaires renvoient vers la source des messages ICMP [RFC1256] lui demandant de ralentir son débit (ces messages sont appelés *source quench* dans le protocole). Dans la deuxième méthode, des informations sont renvoyées par les récepteurs vers l'émetteur. C'est cette dernière méthode que nous avons choisi car elle correspond mieux à la philosophie Internet. Cette méthode est en fait très courante dans les protocoles de transport point à point, la plus célèbre étant sans doute l'algorithme de "démarrage lent" (*slow-start*) et "d'évitement de congestion" (*congestion avoidance*) qui a été incorporé dans TCP en 1988 [Jacobson88] à la suite de sérieux problèmes de congestion dans l'Internet. Cet algorithme a permis de rétablir

la situation en adaptant la charge aux ressources disponibles, ce qui a permis de maintenir les délais de transmisssion dans des limites acceptables. Nous en donnons ici une brève description qui sera utile pour la description de notre algorithme.

Mécanisme de "démarrage lent" et "d'évitement de congestion"

Le mécanisme de "démarrage lent" est un mécanisme de contrôle de flux à "fenêtre glissante". Il consiste à utiliser une "fenêtre de contrôle de flux" dont la taille varie en fonction de la charge du réseau. La valeur de la fenêtre est égale au nombre maximal de paquets qui peuvent être en transit sur une connexion donnée. L'état du réseau est estimé par le taux de pertes. Si aucun paque n'est perdu, le réseau est considéré comme étant peu chargé et on autorise donc la source à augmenter son débit. La "fenêtre de contrôle de flux" est égale à sa valeur minimale (1 paquet) lors de l'initialisation de la connexion TCP. On autorise alors un seul paquet à la fois à être envoyé dans le réseau et ce quelque soient les "crédits" envoyés par le récepteur. Ensuite, à chaque acquittement reçu, on augmente la taille de la fenêtre d'une unité jusqu'à ce qu'elle atteigne un "seuil". A partir de ce moment, la taille de la fenêtre est alors augmentée "linéairement" en ajoutant une unité après chaque délai d'acquittement. Le doublement périodique de la fenêtre se traduit par une croissance exponentielle du débit au début de chaque transmission [Shenker90]. En cas de perte de paquet, la fenêtre (et donc le débit de la source) est réduite à sa taille minimum. On fixe alors le nouveau seuil à la moitié de la valeur de la "fenêtre de contrôle de flux" en vigueur lors de la perte du paquet et on réitère l'algorithme. Si la fenêtre atteint le "seuil", la taille de la fenêtre est alors augmentée "linéairement" en ajoutant une unité après chaque délai d'acquittement.

Nous nous sommes inspirés de cet algorithme pour construire le mécanisme de contrôle de congestion multipoint que nous proposons dans IVS [Bolot94a] [Bolot94b]. L'information envoyée par les récepteurs ne sert pas à déclencher des retransmissions de paquets perdus mais plutôt à obtenir une estimation de l'état du réseau. Cette information n'a donc plus la fonction classique d'accusé de réception mais elle fournit à la place une indication sur la qualité de réception de chaque récepteur. La "fenêtre de contrôle de flux" qui n'a plus grand sens avec des données temps-réel du type audio ou vidéo est remplacée par le débit de transmission du codeur. L'algorithme s'appuie donc sur la possibilité de contrôler le débit de l'application, ce que nous avons montré dans la section 5. D'autre part, à la différence du mécanisme de contrôle de congestion de TCP, l'algorithme doit pouvoir fonction-

ner en multipoint, et donc avec un nombre variable voire inconnu de récepteurs, ce qui complique passablement le mécanisme. Nous décrivons dans une première section l'algorithme d'estimation (ou de caractérisation) de l'état du réseau et ensuite l'algorithme d'adaptation de débit de l'application.

6.2.1 Estimation de l'état du réseau

Pour pouvoir caractériser le réseau, l'application elle-même, et donc dans notre cas le codeur vidéo, doit utiliser des mécanismes spécifiques pour estimer la charge instantanée du réseau. Pour cela, elle utilise un mécanisme en boucle fermée (ou à contre-réaction ou encore à base de feed-back) dans lequel les récepteurs se chargent de renvoyer une indication sur la qualité du signal vidéo qu'ils reçoivent. L'idée de base est d'utiliser ces rapports de qualité de réception pour obtenir une estimation de la qualité globale de réception, elle-même fonction de l'état du réseau. En effet, plus la liaison entre émetteur et récepteur est chargée et plus l'image reçue est de mauvaise qualité.

Le mécanisme de feed-back doit pouvoir permettre de réagir assez rapidement aux variations de la charge du réseau. Pour cela, il est nécessaire de sonder périodiquement tous les récepteurs, ce qui introduit de nouveaux problèmes. Par exemple, dans le cas de la diffusion vers un grand nombre de récepteurs, il est essentiel de limiter l'émission des messages de feed-back: ces derniers contribueraient alors à la congestion du réseau. Ce phénomène est connu dans la littérature sous le nom "d'implosion de feed-back" [Yavatkar93]. Il est donc impératif que le mécanisme d'estimation de l'état du réseau puisse s'adapter à un nombre variable de récepteurs.

Plusieurs solutions existent pour éviter le problème d'implosion de feed-back: on peut utiliser une approche où les réponses sont émises avec une certaine probabilité, ou avec un retard aléatoire, ou encore utiliser une approche à base de "diffusion progressive".

Approche probabiliste

Dans l'approche probabiliste, un récepteur envoie son message de feed-back à la demande de l'émetteur avec une probabilité donnée. Si le message est perdu, la demande est renvoyée de nouveau après un délai donné. Ce schéma a l'avantage d'être simple mais comporte plusieurs limitations. Par exemple, l'émetteur n'est pas

sûr de recevoir une estimée de l'état du réseau dans un intervalle de temps donné et il peut donc réagir trop tard à une variation de la charge du réseau. De plus, lorsque la taille du groupe est inconnue, il est délicat de trouver une valeur appropriée pour initialiser la probabilité de réponse.

Approche avec retard aléatoire

Dans la seconde approche, chaque récepteur retarde l'émission de son message de feed-back d'un intervalle de temps choisi de façon aléatoire. Il est clair que cette approche ne permet pas d'empêcher le problème d'implosion si l'intervalle dans lequel est choisi le retard est trop petit. En revanche, ce mécanisme est intéressant car il permet à tous les récepteurs du groupe de répondre, à condition que le délai soit adapté à la taille du groupe. Ce schéma, plus connu sous l'appelation *slotting and damping* fait partie du protocole de transport XTP [XTP92]. Il est aussi utilisé dans l'application d'audioconférence *VAT* [Jacobson92] pour transmettre les messages qui permettent à des participants de joindre et de quitter une audioconférence. Le délai entre deux émissions de messages de session est alors sélectionné dans un intervalle donné de manière à ce que la bande passante utilisée pour transmettre ces messages ne représente que 1% de celle du flot audio.

Approche à "diffusion progressive"

Avec l'approche à "diffusion progressive", la portée des messages de demande de feed-back émis par l'émetteur est graduellement augmentée. Ces paquets sont donc transmis de plus en plus loin dans l'arbre de transmission multipoint, jusqu'à ce qu'un nombre suffisant de réponses soient reçu. Ce type de schéma comporte plusieurs inconvénients. Tout d'abord, il ne permet pas de trouver le récepteur qui reçoit la plus mauvaise qualité car ce dernier a peu de chances de se situer près de l'émetteur. De plus, il n'est pas efficace lorsque le nombre de récepteurs est élevé. En effet, en raison des décalages horaires entre les régions du globe, il est probable qu'une grande proportion des récepteurs soit localisée au voisinage de l'émetteur, empêchant ainsi de limiter le nombre de réponses des récepteurs.

Approche hybride

Le mécanisme de caractérisation du réseau proposé ci-après [Bolot94b] a été élaboré par Ian Wakeman. Nous en donnons ici le principe; les détails et l'évaluation

de l'algorithme se trouvent en Annexe B.

Ce mécanisme essaie de tirer partie des avantages des trois approches précédentes. Il effectue des séries de sondages sur les récepteurs du groupe multipoint de manière à évaluer:

- la pire qualité de réception perçue par le groupe,
- le nombre de récepteurs dans le groupe multipoint,
- la valeur maximale du temps d'aller-retour *rtt* (*round trip time*) dans le groupe; cette valeur servant à déterminer l'intervalle de temps entre deux sondages consécutifs.

Pour répondre à tous les cas de figure, on fait l'hypothèse que la taille du groupe multipoint est inconnue à la fois par l'émetteur et par les récepteurs. Lorsque la taille du groupe est connue, elle peut être utilisée pour initialiser l'algorithme avec de meilleurs paramètres. Cependant, il n'est pas indispensable de connaître le nombre de récepteurs car l'algorithme permet d'estimer à la fois la taille du groupe et la proportion des récepteurs qui perçoivent une même qualité de réception.

De plus, le fait de placer la détection de congestion au niveau des récepteurs donne la possibilité d'utiliser une métrique qui convienne à chaque type de récepteur, que ce soit la mesure du taux de pertes ou du délai de "bout en bout". De cette manière, des signaux de congestion hétérogènes peuvent être utilisés à l'intérieur d'un même arbre de distribution multipoint. Afin de permettre la conception d'algorithmes génériques, on utilise les variables suivantes pour caractériser l'état du réseau perçu par un récepteur: *NON-CHARGE, CHARGE, CONGESTIONNE.*

Principe

Dans la suite, on appelle une "époque" une série de sondages. Au début de chaque époque, l'émetteur et tous les récepteurs génèrent aléatoirement des mots de 16 bits[2] appelés "clefs". Lorsque l'émetteur désire recevoir des messages de feedback, il envoie un paquet REQUETE aux récepteurs en spécifiant sa clef ainsi que le nombre de bits significatifs de sa clef. A l'initialisation de l'algorithme, les 16 bits sont significatifs. Si les bits significatifs de la clef d'un récepteur sont identiques aux

2. Cette taille est justifiée en Annexe B

bits significatifs de la clef de l'émetteur, et si une des deux conditions suivantes est remplie, alors le récepteur répond à la requête en envoyant un paquet REPONSE.

- L'indicateur TAILLE-DEMANDEE dans l'en-tête du paquet REQUETE est mis à 1. Dans ce cas, la réponse du récepteur permet à l'émetteur d'estimer la taille du groupe. En effet, il existe une relation simple entre le nombre de récepteurs qui répondent, le nombre de bits significatifs et le nombre de récepteurs, voir Annexe B.

- L'état perçu par le récepteur est pire que celui envoyé dans le champ ETAT du paquet REQUETE. Le récepteur indique l'état qu'il perçoit dans le champ ETAT du paquet REPONSE. Le champ ETAT du paquet REQUETE est initialisé au début de chaque époque avec l'état NON-CHARGE.

L'exemple de la figure 6.3 montre le test d'identité de la clef du récepteur n avec celle de l'émetteur pour 5 bits significatifs. Les deux clefs sont différentes, le récepteur n

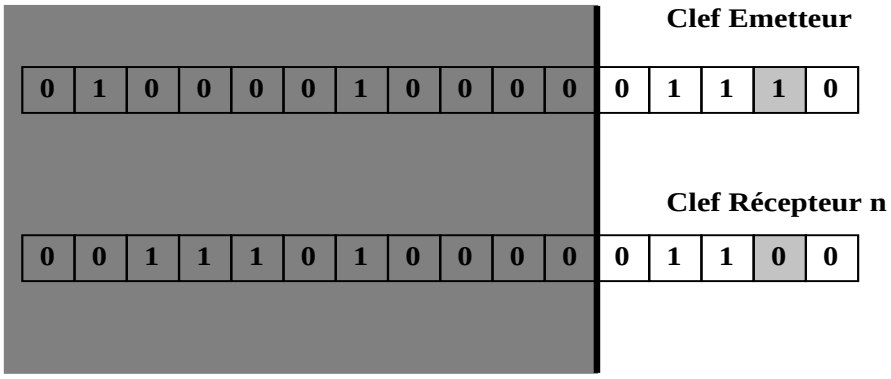
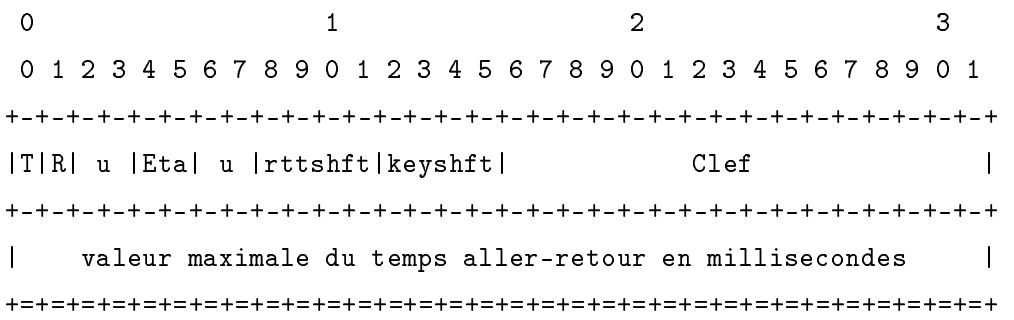


FIG. 6.3 - *Exemple de comparaison de clefs pour 5 bits significatifs*

ne peut en aucun cas envoyer de paquet REQUETE dans cette époque.

Les figures 6.4 et 6.5 donnent une description des paquets REQUETE et REPONSE. Si l'émetteur ne reçoit aucune réponse pendant un intervalle de temps correspondant à deux fois la valeur maximale du rtt calculée dans le groupe, alors l'émetteur décrémente de un bit le nombre de bits significatifs dans sa clef, puis il réitère sa demande avec la même clef que précédemment. L'algorithme continue ainsi jusqu'à ce que le nombre de réponses escompté soit reçu ou qu'il ne reste plus un seul bit significatif dans la clef de l'émetteur. Dans les deux cas, l'époque se termine.



Avec: T = TAILLE-DEMANDEE; R = RTT-DEMANDE; Eta = ETAT; u sont des bits inutilisés; rttshft spécifie le délai 2^rttshft en secondes; keyshft indique le nombre significatif de bits dans la clef.

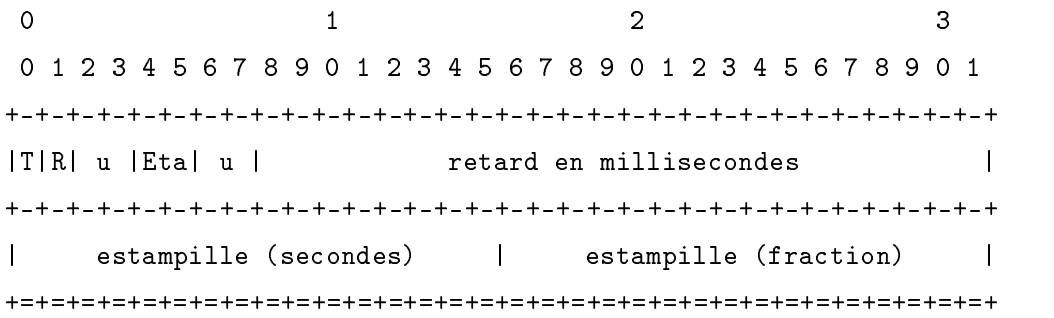
FIG. 6.4 - *Format du paquet REQUETE généré par l'émetteur*

Si un récepteur a répondu au cours d'une époque, il ne répondra plus pendant une période correspondant au temps d'aller-retour maximal calculé par l'émetteur. Après cette période, il est de nouveau autorisé à répondre si il remplit encore les conditions requises. Lorsque l'émetteur reçoit une réponse qui contient l'état CHARGE, il met à jour sa variable ETAT et continue de solliciter des réponses. Dès que le nombre de réponses indiquant l'état CONGESTIONNE dépasse un seuil[3] donné, le codeur vidéo est supposé réduire son débit. Dans ce cas, l'époque courante est interrompue et une nouvelle époque commence avec comme initialisation, l'état NON-CHARGE et le bit TAILLE-DEMANDEE à 1. Dans le cas où l'émetteur réussit à terminer une époque avec l'état NON-CHARGE, le codeur a la possibilité d'augmenter son débit.

Les époques doivent se suivre de manière continue afin que l'émetteur soit averti rapidement de tout changement concernant les conditions du réseau. Une fois que la taille du groupe est estimée, on peut réduire le temps de réponse du mécanisme en choisissant au début de chaque époque un nombre de bits significatifs qui soit plus approprié à la conférence.

La valeur maximale de rtt est calculée de la manière suivante. L'émetteur envoie un paquet REQUETE aux récepteurs en mettant à 1 l'indicateur RTT-DEMANDE. Il spécifie aussi un délai moyen de réponse T secondes. Dès qu'un récepteur reçoit

3. Le choix de ce seuil est discuté dans l'annexe B.



Avec: T = réponse à un TAILLE-DEMANDEE; R = réponse à un RTT-DEMANDE Eta = ETAT; u sont des bits inutilisés; retard est le retard d'une réponse à RTT-DEMANDE; estampille est l'estampille du paquet REQUETE qui a engendré la réponse.

FIG. 6.5 - *Format du paquet REPONSE généré par un récepteur*

la requête, et si il n'est pas déjà en attente pour émettre une réponse, il sauvegarde l'estampille du paquet REQUETE puis attend un délai T_s. Le délai T_s est calculé à partir d'une distribution uniforme sur l'intervalle $[0, T]$. Une fois ce délai écoulé, le récepteur envoie au codeur un paquet REPONSE en spécifiant le délai qu'il a attendu et l'estampille du paquet REQUETE qui a provoqué la demande. L'émetteur, en recevant les réponses des récepteurs, peut en déduire la valeur maximale du rtt pour la conférence. Le délai moyen T, fonction de la taille du groupe, est fixé de manière à obtenir un débit raisonnable des réponses.

A la fin de chaque époque, la source va pouvoir caractériser l'état du réseau en fonction des paquets REPONSE qu'elle a reçus et ensuite ajuster son débit selon l'état estimé. Le but est de réduire le débit de la source si plus d'un certain pourcentage des récepteurs sont congestionnés et de l'augmenter si l'état du réseau est estimé comme NON-CHARGE. Le pourcentage de récepteurs congestionnés peut se déduire en fonction de la différence de numéro de sondage entre un sondage dans lequel la première réponse à une demande TAILLE-DEMANDEE est reçue et le sondage dans lequel on reçoit une réponse CONGESTIONNE (l'équation B.1 de l'Annexe B).

6.2.2 Adaptation du débit du codeur à l'état du réseau

Une fois que l'on a caractérisé le réseau, il reste à adapter le débit de la source. C'est ce que nous étudions dans cette section.

L'estimation de l'état du réseau est mise à jour à la fin de chaque époque. L'algorithme de contrôle de congestion vise à maximiser le nombre de récepteurs qui reçoivent une bonne qualité vidéo. Pour cela, il va ajuster le débit de la source en fonction de l'état estimé du réseau.

Nous avons vu que dans le protocole TCP, la taille de la "fenêtre de contrôle de flux", approximativement proportionnelle au débit avec lequel les paquets sont émis dans le réseau, était ajustée via un algorithme d'accroissement linéaire et de décroissance multiplicative. Cet algorithme est utilisé pour assurer la stabilité et fournir à la fois efficacité et équitabilité dans le sens où n sources qui partagent le même lien recevront chacune en moyenne une fraction $1/n$ de la bande passante de la liaison [Chiu89].

Etant donné que les applications de vidéoconférence sont sensées partager la capacité du réseau avec un grand nombre d'applications qui utilisent TCP, il est nécessaire que notre algorithme ait un "niveau d'agressivité" équivalent. Nous nous sommes donc inspirés de l'algorithme de contrôle de congestion de TCP et utilisons le même type d'accroissement linéaire et de décroissance multiplicative. L'objectif de l'adaptation est de conserver le réseau dans l'état CHARGE.

Dans l'algorithme d'adaptation du débit du codeur à l'état du réseau que Jean Bolot et moi-même avons élaboré, $debit_{MAX}$ (qui représente le débit maximal que la source vidéo est autorisée à émettre) est diminué de moitié lorsque la fraction des récepteurs congestionnés $frac_{CONG}$ est supérieure à un seuil donné, appelé $seuil_{CONG}$. Le paramètre $debit_{MAX}$ est augmenté d'un incrément fixe $taux_{INC}$ lorsque tous les récepteurs considèrent le réseau comme NON-CHARGE. L'algorithme s'assure aussi que le débit de la source est toujours supérieur au débit minimal $debit_{MIN}$ qui correspond à la qualité minimale que l'on désire.

Algorithme de contrôle de congestion

if ($frac_{CONG} > seuil_{CONG}$) {

$debit_{MAX} = \max(debit_{MAX}/2, debit_{MIN})$;

} else if ($frac_{NON-CHARGE} == 100\%$) {

$debit_{MAX} += taux_{INC}$;

}

Une évaluation de cet algorithme est donnée dans la section 6.2.5.

Nous venons de décrire un algorithme de contrôle de congestion qui effectue un contrôle de "bout en bout", c'est-à-dire qui ne fait pas intervenir les routeurs intermédiaires entre la source et le récepteur. L'approche de "bout en bout" a toutefois certaines limites. En particulier, elle suppose que tous les utilisateurs du réseau coopèrent, en ce sens qu'ils utilisent des algorithmes de contrôle de congestion similaires. Malheureusement, il est impossible d'obliger l'ensemble des utilisateurs de l'Internet à utiliser ce genre d'algorithmes. En effet, rien n'empêche les utilisateurs de développer des mécanismes de contrôle de débit plus agressifs et de ce fait, de s'approprier une part plus importante des ressources du réseau. Ces utilisateurs peu scrupuleux peuvent de cette manière monopoliser les ressources, entraînant alors pour tous les autres usagers du réseau des taux de perte de paquets et des délais élevés. Toutefois, on peut éviter l'apparition de ces problèmes en apprenant aux utilisateurs certaines "règles de bonne conduite" mais cela suppose une coopération de leur part.

Dans une précédente version de l'algorithme de contrôle élaboré avec Jean Bolot [Bolot94a], la source diminuait son débit lorsque plus de la moitié des récepteurs étaient congestionnés. Dans la version actuelle de l'algorithme, i.e. celle qui intègre le mécanisme de caractérisation de réseau de Ian Wakeman [Bolot94b], nous avons porté ce seuil à 1.4 % (ce qui correspond à une différence de 6 numéros de sondage, voir l'équation B.1). Evidemment, le premier choix pouvait engendrer des situations où la moitié des récepteurs recevaient un flot vidéo de mauvaise qualité. Les choix des valeurs de seuil 50 % et 1.4 % sont *ad hoc* et il n'est pas sûr que ces valeurs conviennent à un réseau étendu dans lequel les liaisons sont hétérogènes: les taux de pertes observés à chaque récepteur peuvent dans ce cas être très différents selon la liaison qui est utilisée dans l'arbre de distribution multipoint. Néanmoins, des solutions sont possibles pour résoudre ce type de problème. Elles sont présentées dans la section suivante.

On s'assure ensuite que le feed-back est encore consistant au moment où il parvient à la source. Enfin, on donne plusieurs exemples d'adaptation du débit du codeur vidéo IVS aux conditions du réseau.

6.2.3 Initialisation des paramètres de l'algorithme

Dans la section 6.2.2 nous avons décrit un mécanisme utilisé par les récepteurs pour caractériser le réseau. Cependant, nous n'avons pas spécifié comment étaient assignés les paramètres de l'algorithme.

Le paramètre $rate_{MIN}$ est la valeur de débit que l'application est autorisée d'émettre lorsque le réseau est congestionné. Cette valeur correspond en fait à la qualité minimale de l'application permise. Nous l'avons empiriquement initialisée à 10 kbps. Cette valeur peut être associée à la fenêtre minimale de congestion de TCP qui est de taille 1.

Le paramètre $debit_{MAX}$ est la valeur maximale de débit que l'application peut émettre. Evidemment, elle ne sera choisie que lorsque le réseau est estimé comme NON-CHARGE. Cette valeur n'est pas indispensable, par exemple l'algorithme "d'empêchement de congestion" de TCP ne spécifie aucune taille de fenêtre maximale.

Le paramètre $taux_{INC}$ est l'incrément de débit qui est rajouté au débit maximal autorisé à chaque fois que l'état du réseau est NON-CHARGE. Ce taux est fixé empiriquement à 10 kbps.

Revenons à présent aux variables d'états (NON-CHARGE, CHARGE, CONGESTIONNE). Rappelons nous que le but de l'algorithme consiste à maximiser la qualité vidéo perçue par les récepteurs tout en minimisant la bande passante utilisée par la transmission vidéo. Il est donc nécessaire de pouvoir mesurer la qualité perçue par les récepteurs. Nous donnons ici une brève description des différentes mesures de qualité vidéo qu'il existe dans la littérature.

Le rapport signal sur bruit

Le rapport signal sur bruit (*Signal to Noise Ratio*) (SNR) correspond à la puissance moyenne de la distorsion entre les images originelles, c'est-à-dire avant compression, et les images obtenues aux récepteurs après décompression. Le SNR est obtenu à partir de la formule 6.1.

$$SNR = 10 \log \frac{V_{cc}^2}{V}; \quad V = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (P_u(i,j) - P_o(i,j))^2 \qquad (6.1)$$

où V_{cc} est l'amplitude maximale crête-à-crête du pixel, V la variance de l'erreur, $N \times N$ la taille de l'image, $P_o(i,j)$ la valeur du pixel (i,j) avant compression et $P_u(i,j)$ sa valeur après décompression.

Le SNR permet de comparer entre eux différentes techniques de codage en utilisant des séquences d'images test, (par exemple, la séquence "Miss America", utilisée dans la section 5.2.2). Cependant le SNR ne peut pas être utilisé pour estimer la qualité vidéo reçue par un récepteur. En effet, le calcul du SNR utilise la séquence d'images non comprimées qui n'est pas disponible au récepteur. De plus, il est coûteux en cycles cpu car il nécessite une multiplication et une soustraction par pixel.

L'opinion moyenne des utilisateurs

Comme le but de l'algorithme de congestion est de maximiser la qualité vidéo reçue par les récepteurs, on peut imaginer de demander périodiquement aux récepteurs ou plus exactement aux membres de la conférence de donner leur jugement sur la vidéo qu'ils reçoivent. Par exemple, on peut classer la qualité dans une des catégories suivantes: (mauvaise, passable, assez bonne, bonne, excellente). Cette méthode subjective est connue sous la dénomination "d'opinion moyenne des utilisateurs (ou *Mean Opinion Score* (MOS)) [Jayant93].

Le MOS reflête la véritable opinion des utilisateurs. Ces derniers jugent la qualité de la vidéoconférence en fonction de plusieurs critères: la fréquence d'émission des images, la définition de l'image qui est fonction du taux de perte de paquets mais aussi de la netteté de l'image. A ces critères s'ajoutent le degré d'intéractivité entre les membres de la conférence. Ce dernier point est fonction du délai de transmission de "bout en bout" de la transmission. Si l'application de vidéoconférence permet une fréquence de codage élevée, la synchronisation entre la vidéo et l'audio pourra aussi faire partie des critères pour mesurer voire comparer la qualité de plusieurs systèmes de vidéoconférence.

Bien entendu, cette méthode n'est pas envisageable dans notre algorithme de contrôle de congestion car elle suppose un sondage périodique qui serait pénible pour les membres de la conférence. De plus, le MOS est fonction de certains paramètres qui sont indépendants de l'état du réseau, e.g. la fréquence d'émission vidéo ou la netteté de l'image.

Le taux de perte de paquets

Il peut arriver dans le MBONE actuel que le taux de perte avoisine les 20 %. L'expérience montre que les paquets perdus sont la cause principale de la dégradation de l'image en réception. Il est donc légitime d'utiliser cette mesure pour caractériser le réseau. De plus, le coût en calcul est très faible. Nous avons donc décidé d'utiliser l'information de taux de perte de paquets pour assigner les variables d'états du réseau. Nous avons choisi la variable générique d'état du réseau de la manière suivante:

- 0 à 5 %: NON-CHARGE,
- 5 à 15 %: CHARGE,
- plus de 15 %: CONGESTIONNE.

Ce choix est empirique. Bien entendu, ces valeurs ne conviennent pas à tous les cas de figure. Par exemple, dans un réseau local ou un réseau très peu chargé, il semble plus souhaitable d'associer à l'état NON-CHARGE, un taux de perte nul. En effet, il est alors souhaitable de n'augmenter le débit de l'application que lorsqu'une perte de paquet est constatée. A l'inverse, il est possible que le réseau soit congestionné avant même que l'application ne commence d'émettre. Principalement, on peut distinguer deux causes de congestion:

- un trop grand nombre d'utilisateurs se partage la bande passante du réseau.
- une ou plusieurs applications qui utilisent des protocoles de contrôle de congestion plus "aggressifs" voire même aucun protocole de contrôle de congestion.

Dans le premier cas, il est raisonnable de remettre à plus tard la vidéoconférence, ou si cela n'est pas possible, de choisir un débit d'émission assez faible pour éviter d'agraver la congestion du réseau. C'est ce qui arrive avec l'algorithme que nous proposons: si le réseau est CONGESTIONNE, le débit d'émission du codeur diminue jusqu'à ce qu'il atteigne la valeur minimale $rate_{MIN}$. Dans le deuxième cas, on peut imaginer de rendre notre algorithme plus aggressif en augmentant la valeur de $rate_{MIN}$. Mais choisir un débit d'émission plus important ne serait pas équitable pour les autres applications et contribuerait à congestionner davantage le réseau. De plus, on est souvent dans l'incapacité de déterminer la cause de la congestion du réseau. Il n'est donc pas souhaitable de rendre notre algorithme plus agressif

à l'exception de cas très particuliers (par exemple, si l'on sait à l'avance que les applications avec lesquelles on partage la bande passante du réseau utilisent des algorithmes de contrôle de congestion plus agressifs).

6.2.4 Consistance du feed-back à l'arrivée à la source

Un algorithme de contrôle de "bout en bout" ne peut être efficace que dans la mesure où il réagit assez vite aux variations du signal d'entrée, c'est-à-dire dans notre cas aux variations de la charge du réseau. Il faut donc s'assurer que l'information du feed-back est consistante au moment où elle parvient à la source. Si les variations de la charge du réseau sont plus rapides que le mécanisme de feed-back, alors, le contrôle de "bout en bout" devient impossible. Nous avons évalué dans la section précédente le temps maximal de réaction de l'algorithme. Il reste à le comparer avec la variabilité de la charge du réseau.

Une manière de montrer que le feed-back est encore consistant au moment où il atteint la source est de s'assurer que l'autocorrélation de l'information de feed-back décroît lentement en fonction du temps. L'expérience suivante a été menée sur le MBONE au mois d'octobre 1993 entre une source (i.e. codeur vidéo) localisée à l'INRIA Sophia Antipolis en France et un récepteur sur un site distant à UCL (University College London) en Grande Bretagne [Bolot94a]. A cette époque, la liaison multipoint entre la France et la Grande Bretagne passait par les Etats Unis, ce qui impliquait une valeur de rtt très élevée (de l'ordre de la seconde) et nous mettait donc dans les pires conditions d'expérience. Nous avons mesuré le taux de pertes de paquets correspondant à une vidéoconférence à bas débit (entre 10 et 100 kbit/s) et ce pour un réseau peu chargé (taux moyen de paquets perdus inférieur à 5 %) et pour un réseau congestionné (taux de perte voisin de 20 %). Le récepteur calculait alors périodiquement (toutes les T = 15 secondes) le taux de perte de paquets puis l'envoyait à la source. Les courbes de la figure 6.2.4 montrent l'autocorrélation du taux de perte en fonction du temps (multiple de T), dans le cas d'un réseau non chargé et d'un réseau chargé.

La première courbe montre que l'autocorrélation reste au dessus de la valeur 0.4 pendant plus de 40 T, ce qui correspond à environ 10 minutes. Au bout du même temps, la valeur correspondante pour un réseau chargé est encore plus élevée (environ 0.7). Nous avons montré dans la section précédente que le temps maximal d'obtention de l'état du réseau était de 32 rtt, c'est-à-dire dans notre exemple d'environ 32 secondes. On constate donc dans ces deux exemples, que l'algorithme de

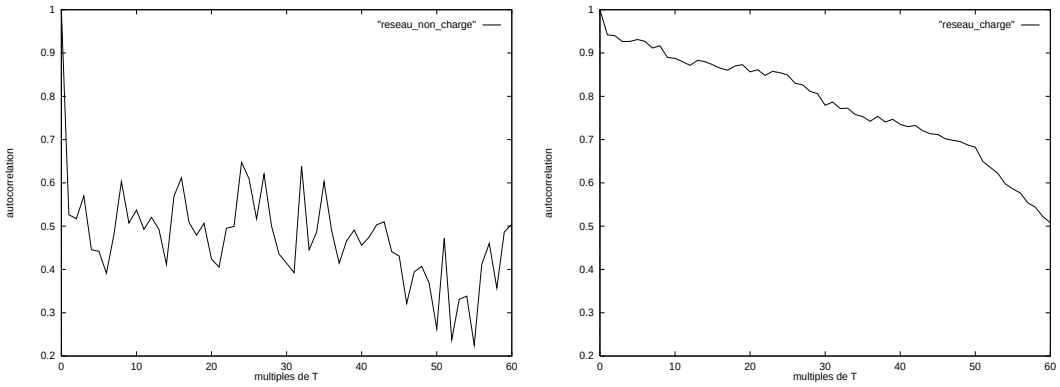


FIG. 6.6 - *Autocorrélation du taux de perte en fonction du temps pour un réseau non chargé et congestionné*

contrôle de congestion réagit donc environ 20 fois plus vite que les variations de la charge du réseau.

6.2.5 Résultats

On présente dans cette section les résultats d'expériences réalisées sur le MBONE avec le logiciel IVS qui utilise cet algorithme. Ces expériences ont pour but d'illustrer l'impact de l'algorithme de contrôle de congestion à la fois sur la bande passante utilisée et sur la qualité vidéo en réception. Pour simplifier, on se place dans le cas où un seul récepteur est abonné au groupe multipoint. Cela nous permet de tester si l'application parvient à contrôler efficacement le débit qu'elle émet en fonction de l'état du réseau. Effectuer une expérience avec plusieurs récepteurs n'apporte pas beaucoup d'intérêts puisque les résultats obtenus dépendraient surtout de l'hétérogénéité des récepteurs.

Comme dans l'expérience précédente, on utilise ici une source vidéo localisée à l'INRIA Sophia Antipolis et un récepteur sur un site distant à UCL. Notons cependant que bien qu'il n'y ait qu'un seul récepteur, on se place dans les conditions du multipoint en envoyant les paquets via le MBONE. Cette expérience remonte à janvier 1994, et à cette époque, la liaison multipoint entre l'INRIA et UCL ne passait plus par les Etats Unis. La liste des routeurs multipoints utilisés par la transmission est donnée dans la table 6.1. Remarquons que cette table ne retrace qu'une partie du chemin physique qu'empruntent les paquets, par exemple le chemin entre Paris et Amsterdam passait alors via le CERN à Genève en Suisse ce qui n'apparaît pas dans la table.

Nom de machine	Institution et/ou pays
sobone.inria.fr	INRIA, France
fmroute11.exp.edf.fr	Paris, France
test-RS.ripe.net	Amsterdam, Pays-Bas
broodjeham.surfnet.nl	Amsterdam, Pays-Bas
noc.ulcc.ja.net	University London, GB
laphroaig.cs.ucl.ac.uk	UCL, GB

TAB. 6.1 - *Principaux routeurs multipoints entre Sophia Antipolis et Londres*

La figure 6.7 montre les évolutions du débit maximal du codeur en fonction du temps. La séquence vidéo envoyée pendant les tests a duré un peu plus de 11 heures. La figure 6.8 montre pour la même expérience, l'évolution du taux de perte de paquets mesuré au récepteur. Comme prévu, à un grand taux de perte correspond

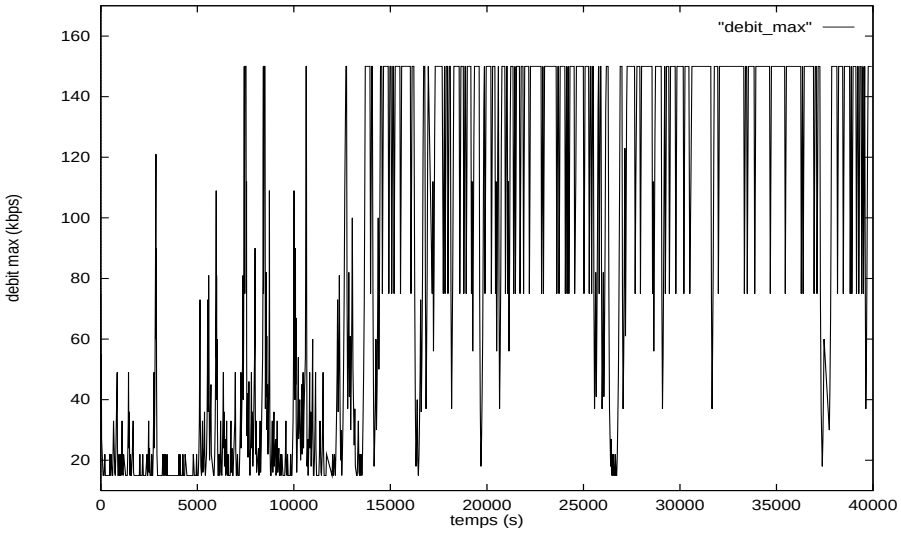


FIG. 6.7 - *Evolutions du $debit_{MAX}$ en fonction du temps*

une petite valeur de $debit_{MAX}$. Si l'état du réseau est congestionné (si le taux de perte est supérieur à 5 %), alors la valeur de $debit_{MAX}$ décroît jusqu'à sa valeur minimal $rate_{MIN} = 10$ kbps. Ceci est visible pour $t > 15000\,s$. La figure montre clairement qu'en cas de congestion, le mécanisme de contrôle empêche la source de surcharger le réseau en limitant son débit à la valeur minimale permise.

La figure 6.9 montre les évolutions du débit maximal du codeur et du taux de

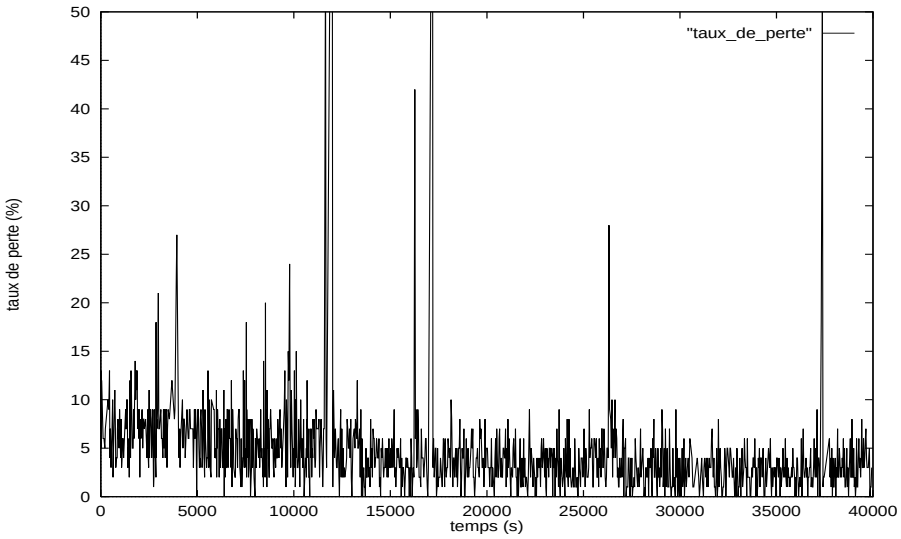


FIG. 6.8 - *Evolutions du taux de perte en fonction du temps*

perte observé par le décodeur pour une expérience de durée plus courte (une heure). La même source et le même récepteur que précédemment sont utilisés. On remarque

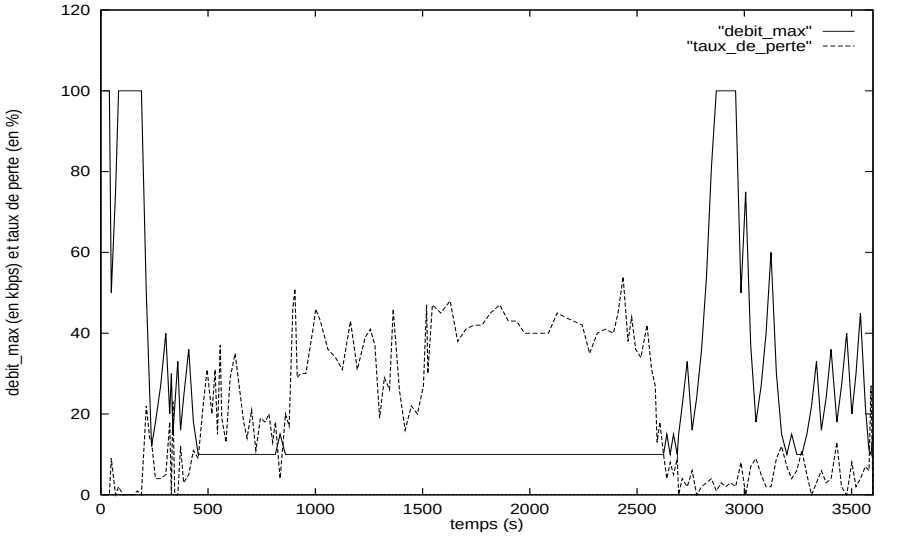


FIG. 6.9 - *Evolutions du debit$_{MAX}$ et du taux de perte en fonction du temps*

que le taux de perte observé par le récepteur est voisin de 40 % pendant environ 30 minutes. Cette congestion a été provoquée par une autre vidéoconférence en provenance de l'Université de Caroline du Nord aux Etats Unis. La source vidéo, de type NV, émettait alors à un débit constant de 128 kbps. Notre algorithme de contrôle de congestion a utilisé alors la valeur minimale de débit admissible (i.e. 10

kbps).

Notre algorithme de contrôle de congestion est basé sur une estimation du taux de perte calculé par chaque récepteur. Dans la majorité des cas, les pertes constatées ont pour origine la congestion du réseau. Cependant, dans certaines conditions, il peut arriver que les paquets soient perdus parce qu'ils n'ont pas eu le temps d'être traités par le récepteur. Ce cas arrive rarement car le coût de l'algorithme de compression H.261 est au moins deux fois plus élevé que celui de l'algorithme de décompression. Toutefois, si la puissance de la machine émettrice est beaucoup plus grande que celle du récepteur, ou si la machine qui effectue la décompression est très chargée, les paquets vidéo vont arriver au récepteur avec un rythme trop élevé pour pouvoir être tous décodés. Si le récepteur se contentait de ne décoder qu'une image sur N selon la puissance de calcul qu'il dispose, la qualité d'image s'en ressentirait du fait de l'utilisation du codage inter-image dans H.261. On note alors le besoin d'un contrôle de flux pour adapter le rythme d'émission des paquets et non plus le débit du codeur, au rythme de décodage du récepteur. Notre algorithme de contrôle de congestion utilise les méthodes de contrôle de débit du codeur H.261 décrites dans le chapitre 5. Seule la méthode qui favorise la qualité d'image peut résoudre le problème précédent car, dans ce cas, la réduction de débit agit directement sur le rythme d'émission des paquets. En revanche, si le codeur a choisi d'utiliser la méthode qui favorise le rafraîchissement d'image, la réduction de débit n'agit que sur la taille des paquets émis (la vidéo étant plus comprimée) et non sur le rythme d'images émises. Une solution à ce problème consiste à ajouter dans le feed-back des récepteurs un indicateur sur l'origine de la perte des paquets (du réseau ou du récepteur). L'émetteur peut alors forcer, si nécessaire, l'utilisation de la méthode qui favorise la qualité d'image.

6.2.6 Le problème de l'hétérogénéité des récepteurs

La transmission multipoint de la vidéo sur l'Internet soulève un problème délicat. Dans la pratique, un arbre de transmission multipoint (c'est-à-dire d'un émetteur vers N récepteurs) est constitué de branches (ou liaisons) qui ont en général des caractéristiques différentes car elles ont soit des bandes passantes différentes soit un taux d'utilisation différent.

Cette non-homogénéité rend particulièrement délicat le contrôle de congestion du réseau. Le problème pour l'émetteur est de choisir son débit de sortie en fonction de l'état du réseau. Le choix du débit aura un impact sur le niveau de congestion

du réseau, exprimé par exemple en terme de taux de pertes des paquets ou de retard de "bout en bout" le long de l'arbre de transmission multipoint. En raison de l'hétérogénéité du réseau, on peut s'attendre à des niveaux de congestion différents selon le trajet du flot vidéo dans l'arbre entre l'émetteur et les récepteurs, c'est-à-dire sur différentes branches de l'arbre. C'est pour cette raison qu'il n'existe pas de valeur optimale de débit qui satisfasse aux besoins de tous les participants. Par exemple, si l'on choisit de s'adapter au récepteur qui expérimente le taux de perte le plus élevé, on diminue en conséquence la qualité vidéo de tous les récepteurs. Si l'on choisit d'adapter le débit du codeur de manière à ce que la majorité des récepteurs ne soit pas congestionnée, on maximise alors le nombre de récepteurs satisfaits en ignorant les problèmes de congestion d'une minorité de récepteurs.

De façon idéale, on aimerait être capable de séparer l'arbre de distribution multipoint en autant de groupes que de récepteurs qui partagent les mêmes caractéristiques. Le traitement de ces différents groupes serait ainsi adapté aux capacités de chacun. Nous étudions par la suite différents procédés pouvant être utilisés pour résoudre ce problème, à savoir le codage hiérachique, le codage simulcast et l'utilisation de passerelles vidéo.

Le codage hiérarchique

La manière la plus élégante d'offrir plusieurs niveaux de qualité vidéo aux récepteurs consiste à utiliser un codage hiérarchique, aussi connu sous le nom de codage multi-résolution, codage en couches ou encore codage en sous-bandes [Garcia85] [Ghanbari89] [Gharavi91] [Gonzales92] [Vetterli90] [Westerink88]. Le codage en sous-bandes s'est développé grâce aux progrès réalisés dans le domaine des techniques de représentation multi-résolution du signal audio et vidéo [Burt83] [Woods86].

Il regroupe une famille de techniques de représentation de signaux dans laquelle le flot émis par la source vidéo est partitionné en plusieurs couches ou sous-flots de données. Les couches sont classées par ordre décroissant d'importance, la première couche (flot de base) comportant les éléments les plus importants du signal original. Les couches suivantes (flots complémentaires) améliorent progressivement la qualité vidéo du flot de base jusqu'à retrouver la qualité du flot original. La décomposition en couches peut se faire par analyse fréquentielle ou à l'aide de filtres spéciaux qui permettent d'obtenir une stratégie d'analyse hiérarchique. L'analyse couche par couche contribue à une interprétation plus pertinente de l'image en se rapprochant

de celle du système visuel humain. Ce dernier possède une sensibilité spatiale qui est fonction du contenu fréquentiel [Bourguignat93]. Le codage en sous-bandes appliqué à la compression d'images permet ainsi de mieux exploiter les propriétés statistiques et psychovisuelles de chaque sous-bande en leur appliquant une compression adéquate des données comme la quantification vectorielle [Vanderdop91].

Une autre application du codage hiérarchique est la transmission progressive d'images fixes à bas débit. Elle consiste à émettre chaque image en plusieurs temps: d'abord, une image de basse résolution correspondant au flot de base, puis des flots complémentaires qui accroissent progressivement la définition de l'image, [Wang88] [Miran90].

Le codage hiérarchique peut aussi être utilisé comme technique d'encodage pour transmettre différents niveaux de qualité vidéo vers des récepteurs hétérogènes. C'est cette dernière application qui nous intéresse ici. L'idée de base est de transmettre l'ensemble des flots sur les liaisons non congestionnées de l'arbre de distribution multipoint et de réduire la bande passante utilisée sur les liaisons congestionnées en n'y transmettant que les flots de base. Un récepteur peut ainsi faire le choix de ne recevoir que les flots de bas niveau ou l'ensemble des flots en fonction des ressources dont il dispose (figure 6.10). La figure 6.11 représente un arbre de transmission multipoint pour un émetteur et neuf récepteurs. On remarque que le flot de base atteint l'ensemble des récepteurs mais que seul les récepeturs des réseaux locaux $r1$ et $r2$ sont abonnés au flot complémentaire.

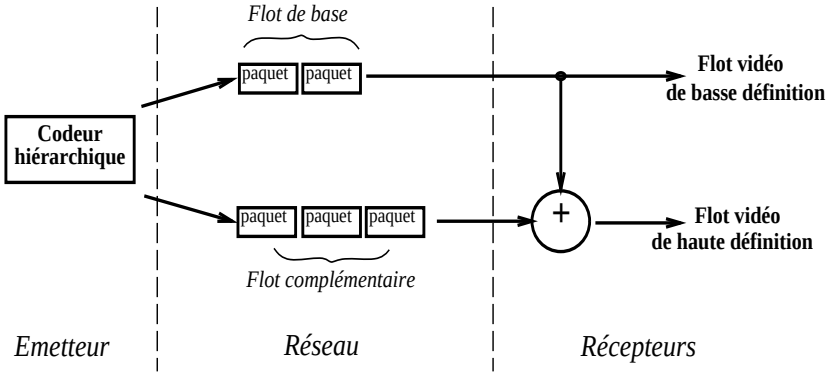


FIG. 6.10 - *Exemple de codeur en sous-bandes*

Le schéma de compression H.261 ne fait pas partie de la famille des schémas de codage hiérachique. Cependant, on peut profiter de la transformation orthogonale qu'il utilise pour générer différents niveaux de qualité vidéo. Rappelons que

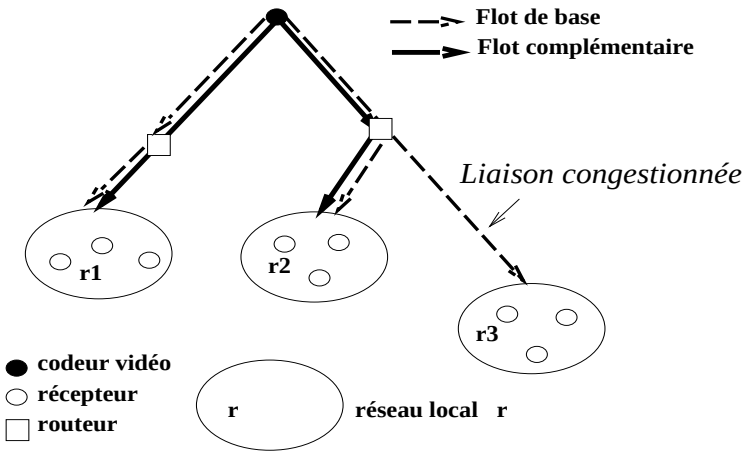


FIG. 6.11 - *Arbre de transmission multipoint pour un codage hiérarchique*

le schéma de compression H.261 utilise une transformée en cosinus qui peut permettre une analyse fréquentielle de l'image. En accord avec l'importance visuelle des composantes fréquentielles, on peut transmettre une partie des coefficients dans le flot de base et le reste des coefficients dans des flots complémentaires qui améliorent la définition de l'image décodée. Plusieurs techniques de segmentation des composantes fréquentielles de l'image existent [DeCleene94] [Wang91]. Prenons pour simplifier l'exemple d'un codage hiérarchique H.261 en deux sous-bandes. Une manière peu coûteuse de segmentation consiste à séparer l'émission des coefficients basse-fréquence (BF) de l'émission des coefficients haute-fréquence (HF) en les parcourant dans l'ordre "zigzag". Les N premiers coefficients parcourus font partie du flot de base et les coefficients restants, du flot complémentaire (figure 6.12). Cette méthode dite par "troncation en fréquence" est basée sur l'observation que les coefficients basse-fréquence ont une importance visuelle plus grande que les coefficients haute-fréquence. Une autre méthode dite de "distorsion minimum" consiste à transmettre les N coefficients de plus grande énergie dans le flot de base et les coefficients restants dans le flot complémentaire. Cette technique est plus coûteuse en calculs que la précédente mais elle donne de meilleurs résultats. Enfin, une autre technique fonctionnant par "seuillage d'énergie" consiste à transmettre dans le flot de base tous les coefficients dont l'énergie excède un seuil donné, sans tenir compte du nombre de coefficients. Cette méthode a un coût de calcul intermédiaire entre les deux précédentes techniques et donne de meilleurs résultats que la technique par troncation en fréquence [DeCleene94].

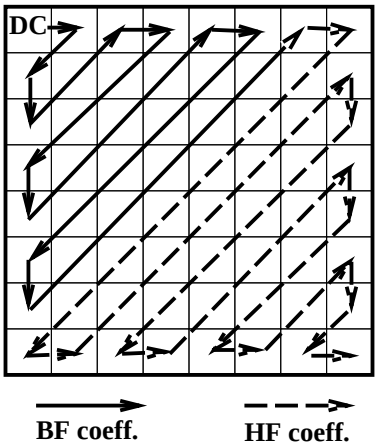


FIG. 6.12 - *Segmentation des coefficients BF et HF dans un bloc de TCD*

Afin d'assurer la compatibilité avec des codecs H.261, il est nécessaire de fusionner préalablement les flots vidéo avant leur décodage. Prenons par exemple, le cas d'un codage hiérarchique à segmentation de coefficients, où les N premiers coefficients (BF) sont émis dans le flot de base et les coefficients restants (HF) dans le flot complémentaire. Sans aucune modification, le flot de base est le seul à pouvoir être décodé par un codec H.261. Pour décoder les deux flots, il est nécessaire avant décodage de fusionner les coefficients BF et HF de chaque bloc d'image. Pour accéder aux coefficients des blocs, le multiplexeur doit au préalable effectuer un décodage de Huffman des flots BF et HF pour chaque image. Après avoir regroupé l'ensemble des coefficients de chaque bloc, il faut effectuer un encodage de Huffman afin d'obtenir un flot vidéo conforme au standard H.261. Ces opérations sont récapitulées dans la figure 6.13. On a vu que le codage hiérarchique permet d'offrir différents niveaux de qualité vidéo en fonction du nombre de sous-bandes décodées. Cependant, les flots complémentaires ne permettent d'améliorer la définition d'image que lorsque le flot de base a été correctement reçu. Le codage hiérarchique n'est donc efficace que lorsqu'il est possible de garantir une bonne réception du flot de base. Cela pose un problème dans l'Internet actuel, et oblige l'application à utiliser des méthodes de contrôle d'erreurs (comme par l'exemple, les méthodes à base d'ajout de redondance, voir section 4.2.1) pour reconstituer tous les paquets perdus du flot de base. En revanche, si l'on modifie la discipline des routeurs en leur permettant d'écarter sélectivement les paquets non prioritaires, il est possible de favoriser la transmission du flot de base. Un mécanisme de priorité associé aux paquets émis

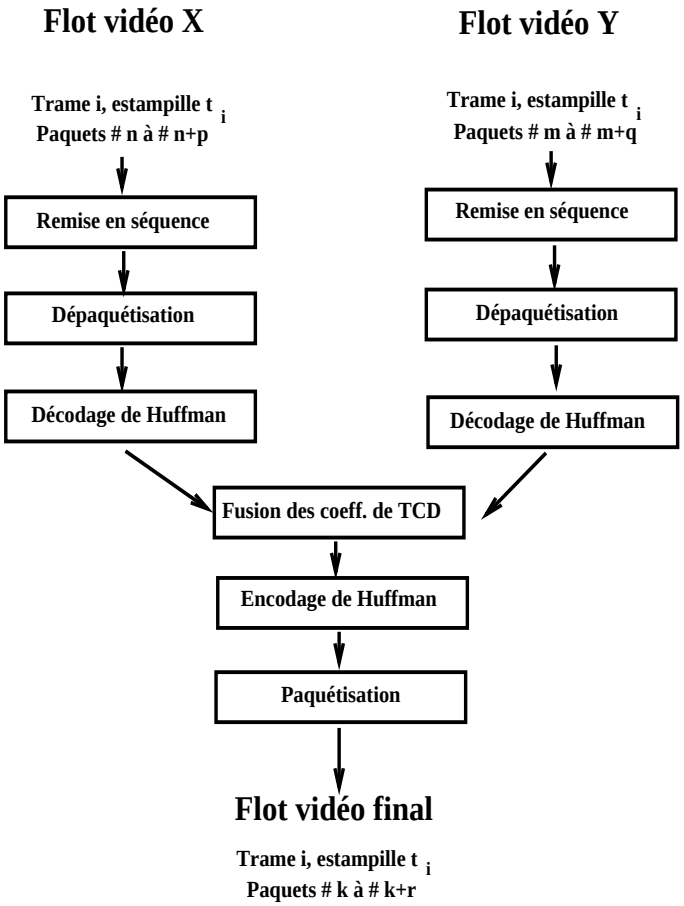


FIG. 6.13 - *Multiplexage de deux flots vidéo H.261*

à partir d'une même source est d'ailleurs prévu dans la nouvelle génération de IP (*IPng*), cf. [RFC1752]. Dans l'en-tête de IPng, un champ de 4 bits appelé *TCLASS* est utilisé pour typer les données émises. De cette manière, il sera possible de choisir la politique adéquate à adopter en cas de congestion du réseau selon la nature des applications utilisées. On distingue deux grandes classes d'application selon qu'elles utilisent un algorithme de contrôle de congestion ou non. Les valeurs de 0 à 7 sont recommandées pour les applications avec contrôle de congestion intégré. Le tableau 6.2 récapitule les différents types de trafic associés au champ TCLASS. Les valeurs de 8 à 15 doivent être utilisées pour le trafic sans contrôle de congestion, comme la plupart des applications temps-réel. La valeur 8 est réservée pour les paquets dont la priorité est maximale, la valeur 15, pour ceux dont la priorité est minimale. Par exemple, pour un codage en 4 bandes, on attribuera la plus petite valeur (i.e. 8) pour les paquets du flot de base et les valeurs 9, 10 et 11 pour les paquets des flots

TCLASS	Type de trafic
0	Trafic indéfini
1	Trafic de faible priorité (e.g. netnews)
2	Trafic imprévisible (e.g. e-mail)
3	(Réservé)
4	Trafic prévisible (e.g. FTP, NFS, HTTP)
5	(Réservé)
6	Trafic intéractif (e.g. telnet, X)
7	Trafic de contrôle d'Internet (e.g. SNMP)
8 - 15	Trafic sans contrôle de congestion

TAB. 6.2 - *Attribution des valeurs du champ TCLASS*

complémentaires suivants.

Si de plus, une bande passante minimale peut être réservée (les mécanismes qui le permettront sont décrits dans la section 6.3.2), on peut alors garantir la bonne réception du flot de base et le codage hiérarchique revêt alors tous ses avantages pour ce genre d'application.

Le codage simulcast

A la différence du codage hiérarchique, le codage simulcast [Hoffman93] partitionne la source vidéo en plusieurs flots vidéo indépendants les uns des autres (figure 6.14). Avec ce type de codage, les récepteurs ne s'abonnent qu'à un seul

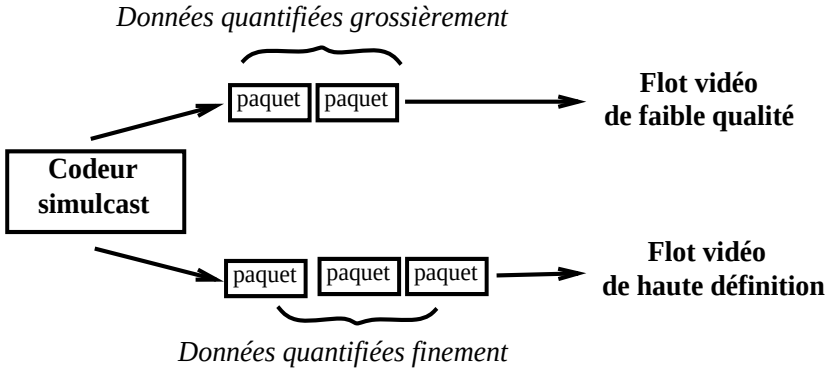


FIG. 6.14 - *Exemple de codeur simulcast*

des flots vidéo disponibles. L'indépendance de décodage inter-flots est obtenue par

le biais d'un recouvrement d'information entre les flots. Prenons pour exemple le cas du codage H.261. Deux types de codage simulcast sont possibles: si le rythme de rafraîchissement d'image est important, une solution consiste à encoder chaque image avec deux valeurs différentes de quantificateur. Le flot d'images encodé avec le quantificateur le plus grand (et donc le flot le plus comprimé) correspond au flot vidéo de faible qualité. Si la définition des images est plus importante que le rythme de rafraîchissement d'images, une solution consiste à forcer périodiquement un codage complet d'image en mode intra-image [Turletti94a]. Ces images encodées exclusivement en mode intra-image peuvent être à la fois émises avec les autres images dans un flot dit de haute qualité et émises séparément dans un autre flot dit de basse qualité. Pour cela, il suffit de les stocker en mémoire après encodage et de les envoyer dans un autre flot avec un débit plus faible, par exemple avec 20% du débit du flot de haute qualité. Afin de se prémunir contre la perte des paquets, la même image peut être continuellement réémise jusqu'à ce que la prochaine image INTRA soit disponible.

L'avantage du codage simulcast est qu'il ne nécessite pas de mécanisme de réservation de ressources. De plus, le flot de basse qualité est moins sensible aux pertes de paquets que ne l'est le flot de base du codage hiérachique pour deux raisons: il n'utilise que le codage intra-image et la réémission continue des images assure un rafraîchissement des zones d'images erronées en cas de perte de paquets. Cette méthode peut donc être utilisée avantageusement dans l'Internet actuel. D'autre part, à la différence du codage hiérarchique, les flots de basse et haute qualité sont des flots standards H.261 et peuvent donc être décodés par des codecs conformes au standard H.261 sans transformation préalable. Cependant, cette méthode nécessite plus de bande passante que le codage en sous-bandes, car l'information qui circule est en quelque sorte dupliquée dans certaines parties de l'arbre de transmission multipoint. La figure 6.15 représente un arbre de transmission multipoint pour une source S et neuf récepteurs. Les récepteurs des réseaux locaux $r1$ et $r2$ sont abonnés au flot vidéo de haute qualité tandis que ceux du réseau local $r3$ reçoivent le flot vidéo de faible qualité. On remarque que les deux flots vidéo circulent sur la liaison a-c et que la somme des flots est supérieure au flot initial.

Les passerelles vidéo

Une autre solution consiste à introduire des passerelles vidéo à certains endroits de l'arbre de transmission multipoint (figure 6.16). Une passerelle vidéo peut être

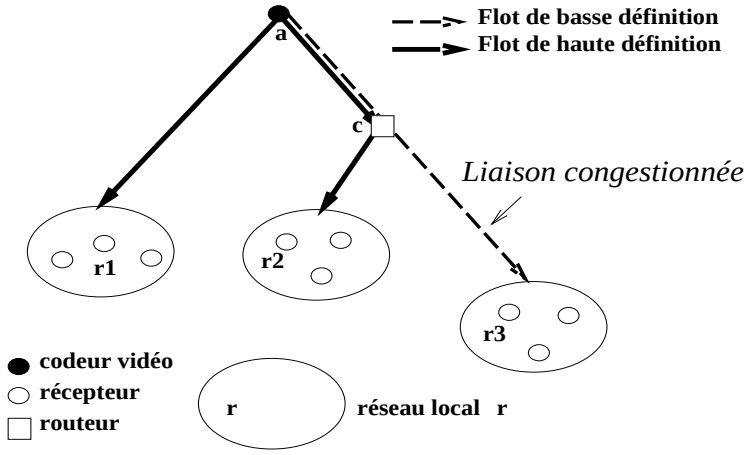


FIG. 6.15 - *Arbre de transmission multipoint pour un codage simulcast*

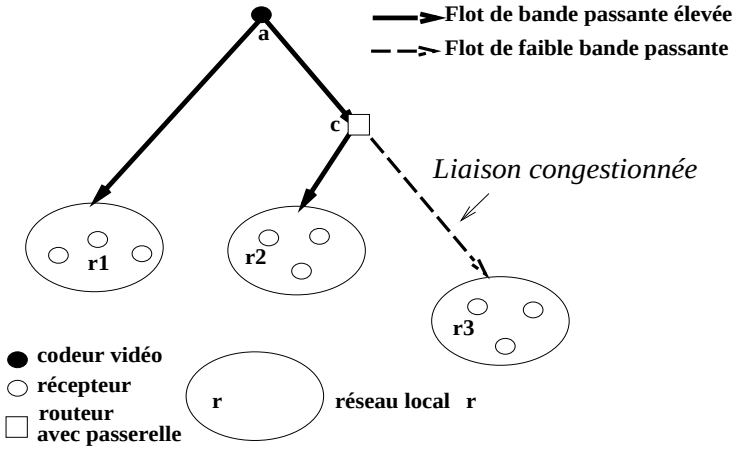


FIG. 6.16 - *Mise en place d'une passerelle vidéo*

schématisée par une boite noire qui transforme un flot vidéo A en un flot vidéo B de débit inférieur. Les schémas de compression des flots vidéo A et B peuvent être identiques ou différents. Par exemple, une passerelle vidéo peut transformer un flot MPEG de haute qualité à 1 Mbps en un flot MPEG de qualité inférieure à 256 kbps ou encore en un flot H.261 à 128 kbps. Le délai engendré par la passerelle vidéo dépend en grande partie du fonctionnement intrinsèque de la passerelle. Le délai est minimal dans le cas où la passerelle se contente de filtrer le flot vidéo, par exemple en ne laissant passer que les paquets d'un flot MPEG encodés en mode intra-image. Ce délai est en revanche beaucoup plus important si la passerelle modifie le codage. Dans ce dernier cas, elle doit au préalable décoder le flot A puis le recoder soit

avec un autre schéma de compression soit en changeant des paramètres de codage comme le quantificateur ou le nombre de bits par pixel. Il est possible de réaliser des passerelles entre n'importe quels types de schémas de compression. Une solution universelle consiste à décoder chaque image du flot vidéo d'entrée et à n'encoder qu'une image parmi N en fonction du débit de sortie désiré.

Dans le cas de la transmission vidéo multipoint dans l'Internet, l'utilisation de passerelles permet de fournir plusieurs niveaux de qualité et donc d'augmenter la qualité de réception de l'ensemble des récepteurs du groupe. Mais pour cela, il faut être en mesure de placer les passerelles aux endroits adéquats, ce qui peut s'avérer parfois difficile. En effet, il est nécessaire de connaître les liaisons du réseau où se produit la congestion afin d'y introduire en amont la passerelle appropriée. Son emplacement idéal se situe à l'intérieur même ou au voisinage du routeur multipoint où se situe le goulot d'étranglement. J.C. Pasquale a proposé un protocole de mise en place automatique de passerelles dans un arbre de distribution multipoint dans lequel les routeurs multipoint ont la possibilité d'introduire différentes sortes de passerelles [Pasquale93]. Ces passerelles se mettent en place automatiquement dans le réseau et se propagent en chemin inverse du trafic vidéo, c'est-à-dire des récepteurs vers la source d'émission.

Remarques générales

Les trois techniques que nous venons de décrire (à savoir le codage hiérarchique, le codage simulcast et l'utilisation de passerelles vidéo) supposent toutes l'utilisation du mécanisme de *pruning*[4]. Les flots vidéo de faible bande passante et de bande passante plus élevée doivent être transmis avec des adresses multipoint différentes pour permettre le *pruning*.

Les récepteurs ne doivent pas s'abonner ou se désabonner aux flots trop souvent car ces actions sont coûteuses et peuvent engendrer des oscillations indésirables dans les mécanismes de contrôle de congestion du réseau. Ce problème fait partie des discussions en cours dans le groupe de travail IDMR (*Inter-domain Multicast Routing*) de l'IETF [IDMR].

4. Le *pruning* est un mécanisme qui permet de limiter la transmission des flots de données aux branches de l'arbre de transmission multipoint qui contiennent au moins un récepteur (cf. [Deering91]).

6.3 Autres solutions

6.3.1 En changeant la discipline des routeurs

Rappelons que la plupart des routeurs de l'Internet gèrent aujourd'hui leurs files d'attente selon la discipline de service FIFO. Cette discipline est facile à utiliser mais comporte deux inconvénients importants:

- les délais sont dépendant de la charge du réseau, ce qui ne convient pas aux applications qui ont des contraintes temps-réel sévères.
- les applications peu scrupuleuses peuvent s'approprier injustement une trop grande part de la bande passante du réseau.

Un grand nombre de disciplines de service ont été développées pour offrir des garanties de service plus ou moins complètes: par exemple les algorithmes "d'attente équitable" (*Fair Queuing FQ*) et "d'attente équitable proportionnelle" (*Weighted Fair Queuing WFQ*) [Nagle87], [Demers89] et l'algorithme "d'Horloges Virtuelles" (*VirtualClock*) [Zhang90]. Plusieurs comparaisons et analyses sont disponibles dans la littérature [Kurose93], [Aras94] [Huitema94]. Nous donnons ici les principes des disciplines FQ et WFQ et introduisons le modèle "d'attente organisée par classe".

L'idée de l'algorithme FQ est de séparer le trafic en un ensemble de flots bien identifiés et d'allouer ensuite à chaque flot la même part de la capacité de transmission. Ce mécanisme permet de s'affranchir des comportements "antisociaux" d'une ou de plusieurs connexions afin qu'elles n'aient pas d'impact sur les connexions "raisonnables". Le principe est schématisé par la figure 6.17.

L'allocation de bande passante peut se faire de façon relativement simple, en servant les paquets en tête de chaque file (et donc de chaque flot) à tour de rôle. Ainsi, lorsqu'une file est vide, on passe son tour et on sert la suivante. Cet algorithme a cependant des limitations. En particulier, il ne tient pas compte de la taille des paquets ce qui provoque des disparités en bande passante allouée selon la taille des paquets que l'application utilise. De plus, les paquets qui arrivent juste après que la file d'attente ait été examinée sont pénalisés par rapport aux autres.

Pour remédier à ces problèmes, une nouvelle version de l'algorithme a été développée en utilisant une approximation "fluide" de la discipline FQ. Dans ce nouveau schéma, les files sont servies à tour de rôle, et un seul bit de chaque file est envoyé lors de chaque tour. Cette discipline est plus équitable que la précédente car elle prend en compte la taille des paquets, et le bit en tête d'une file attend au plus

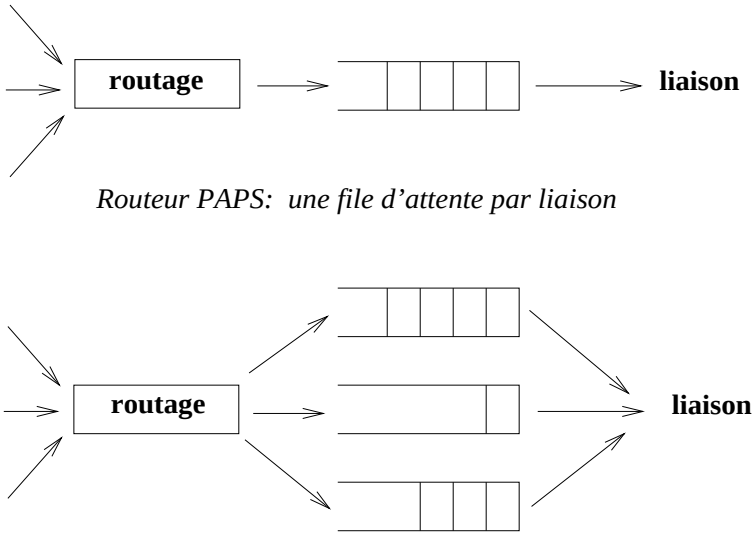


FIG. 6.17 - *Schéma de principe des routeurs FIFO et FQ*

$(n-1)$ fois le temps de transmission d'un bit (si n est le nombre de files actives) avant d'être servi. Si l'on associe un poids ϕ_i à un flot i, qui reçoit une fraction $\frac{\phi_i}{\sum_j \phi_j}$ de la bande passante, on obtient alors la discipline WFQ.

L'implantation des algorithmes FQ et WFQ part du principe que les flots sont identifiables pour permettre la classification du trafic du réseau. Il est donc nécessaire de pouvoir donner une définition d'un flot. Un grand nombre de définitions existe. Chaque définition a des avantages et des inconvénients. Par exemple, un flot peut être caractérisé par une paire d'adresses source et destination, ou un type de trafic (audio, vidéo, données textuelles). Le classement par adresse source IP qui parait au premier abord équitable, revient en fait à donner la même part au trafic généré par un PC qui est utilisé par une seule personne qu'au trafic généré par un Cray probablement utilisé par un grand nombre de personnes.

On aimerait donc que le modèle prenne en compte des classifications multiples. Le modèle "d'attente organisée par classe" (*Class Based Queuing* (CBQ)) propose une organisation hiérarchique des classes [Floyd93b] [Jacobson93]. Par exemple, on peut avoir la hiérarchie suivante:

- par organisation (ou agence),
- selon le type de protocole (e.g. TCP/IP ou IPX de Novell ou SNA d'IBM),

– avec un protocole spécifique (e.g. IP),

– selon l'application (e.g. transmission d'audio, de vidéo ou de fichiers).

Une fois que la division en classes est faite, on peut spécifier des prévisions d'utilisation: par exemple 40% pour IP, 30 % pour IPX et 30% pour SNA. Enfin, à l'intérieur de la classe IP, on peut prévoir 50% pour le transfert de fichiers, 25% pour les applications temps réel, et 25% pour les applications interactives. Ce classement par classe donne de meilleurs résultats que l'utilisation de priorités. En effet, il est par exemple dangereux de toujours donner la priorité aux applications temps réel sur les transferts de fichiers car alors on permet au trafic temps réel d'exclure toutes les applications de transmission de fichiers.

Puisque l'algorithme WFQ isole les flots les uns des autres, on peut s'attendre à ce qu'il permette d'offrir des garanties de performance aux utilisateurs du réseau. C'est effectivement le cas si le trafic généré par les utilisateurs n'est pas trop chaotique. Notons tout d'abord que l'algorithme alloue une fraction de la bande passante disponible C à chacun des flots. Dans le cas de l'algorithme FQ, la fraction allouée à chaque flot est C/n, où n est le nombre de flots actifs. Dans le cas de l'algorithme WFQ, la fraction allouée au flot i est $\phi_i\, C/n$, où ϕ_i est le poids associé au flot i. Notons que cette fraction peut être arbitrairement petite si la valeur de n est grande. Toutefois, chaque flot est assuré d'une fraction minimum de capacité. Des travaux récents ont montré que l'algorithme WFQ permet d'offrir en plus de cette garantie de bande passante une garantie de délai si le trafic généré par les utilisateurs est suffisamment stable [Parekh93].

Nous venons de voir que l'algorithme WFQ permet à un réseau à commutation de paquets d'offrir des services avec certaines garanties de qualité. Ce résultat est à la base d'une réflexion en cours dans l'IETF au sujet de la définition d'une nouvelle architecture pour les "réseaux de paquets à intégration de service". Cette nouvelle architecture permettra d'offrir en plus du service *best effort* actuel, un service avec des garanties strictes de performances (qui mettrait en œuvre la discipline WFQ ou le protocole de réservation de ressource RSVP, protocole que nous décrivons dans la section suivante) et un autre service avec garanties "statistiques" de performance (qui mettrait en œuvre des mécanismes encore en cours d'étude) [Braden93].

6.3.2 Vers une nouvelle architecture d'Internet

Pour expliquer cette nouvelle architecture, on base notre raisonnement sur des fonctions de satisfaction c'est-à-dire des fonctions qui donnent une indication de la qualité de l'application en fonction du débit disponible sur le réseau.

Les fonctions de satisfaction

Pour illustrer notre propos, supposons que le réseau utilise une combinaison de protocoles de "bout en bout" et d'attente équitable au sein des routeurs, pour assurer un partage efficace de la bande passante entre les utilisateurs. S'il n'existe qu'un seul utilisateur, il disposera de toute la capacité disponible du réseau. Avec deux utilisateurs, chacun aura la moitié de la bande passante, et ainsi de suite. La qualité observée est donc variable en fonction du nombre d'utilisateurs du réseau. Dans la plupart des cas, la "loi des rendements décroissants" s'applique, c'est-à-dire que la qualité s'accroît lorsque davantage de débit devient disponible, mais pas de manière linéaire [Huitema94]. Si la fonction de satisfaction est concave, alors la satisfaction globale augmente avec le nombre d'utilisateurs. On peut s'en convaincre avec l'exemple suivant: supposons qu'il y ait un seul utilisateur et qu'il dispose du débit maximal d associé à la qualité maximale q. Si un autre utilisateur arrive et qu'il utilise la même application, les deux utilisateurs se partagent alors équitablement la bande passante d et chacun obtient alors un débit $d' = d/2$ associé à la qualité q' telle que q' est supérieure à $q/2$ (en raison de la concavité de la courbe (figure A, 6.18). Dans ce cas, la réservation de ressources n'est pas souhaitable puisque la

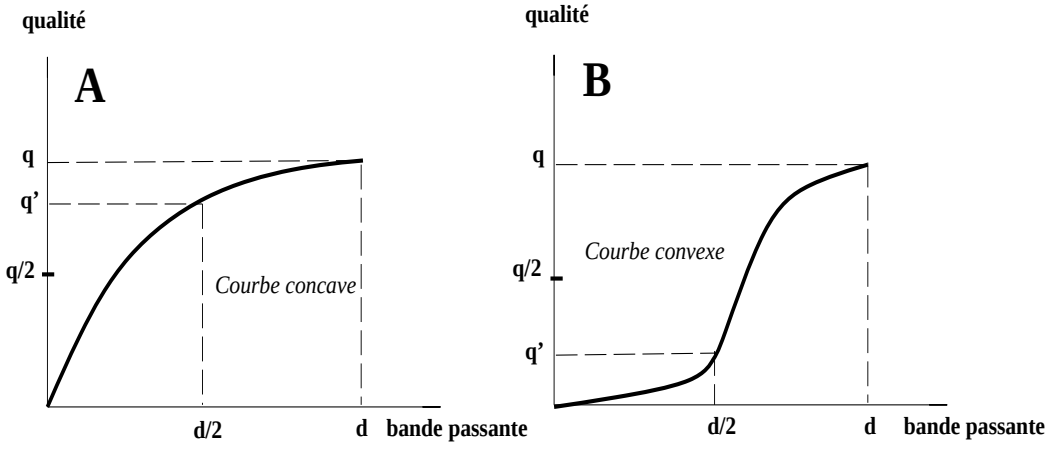


FIG. 6.18 - *Fonction de satisfaction concave et convexe*

satisfaction globale $2\,q'$ est supérieure à celle où il n'y avait qu'un seul utilisateur, c'est-à-dire q. En revanche, si la fonction de satisfaction est convexe, la réservation de ressources permet de maximiser la satisfaction globale des utilisateurs (courbe B, figure 6.18). Ce résultat peut s'étendre facilement avec n utilisateurs: si n utilisateurs disposent d'un débit d/n associé à une qualité q', alors si un nouvel utilisateur arrive, la réservation de ressource n'est souhaitable que si $n\,q' < (n+1)\,q''$ où q'' est la qualité associée au débit $d/(n+1)$.

Pour pouvoir maximiser la satisfaction globale des utilisateurs, il est donc nécessaire de connaître la forme des fonctions de satisfaction de chacune des applications de l'Internet. Scott Shenker a classé les applications utilisées dans l'Internet en quatre catégories [Shenker94] (figure 6.19).

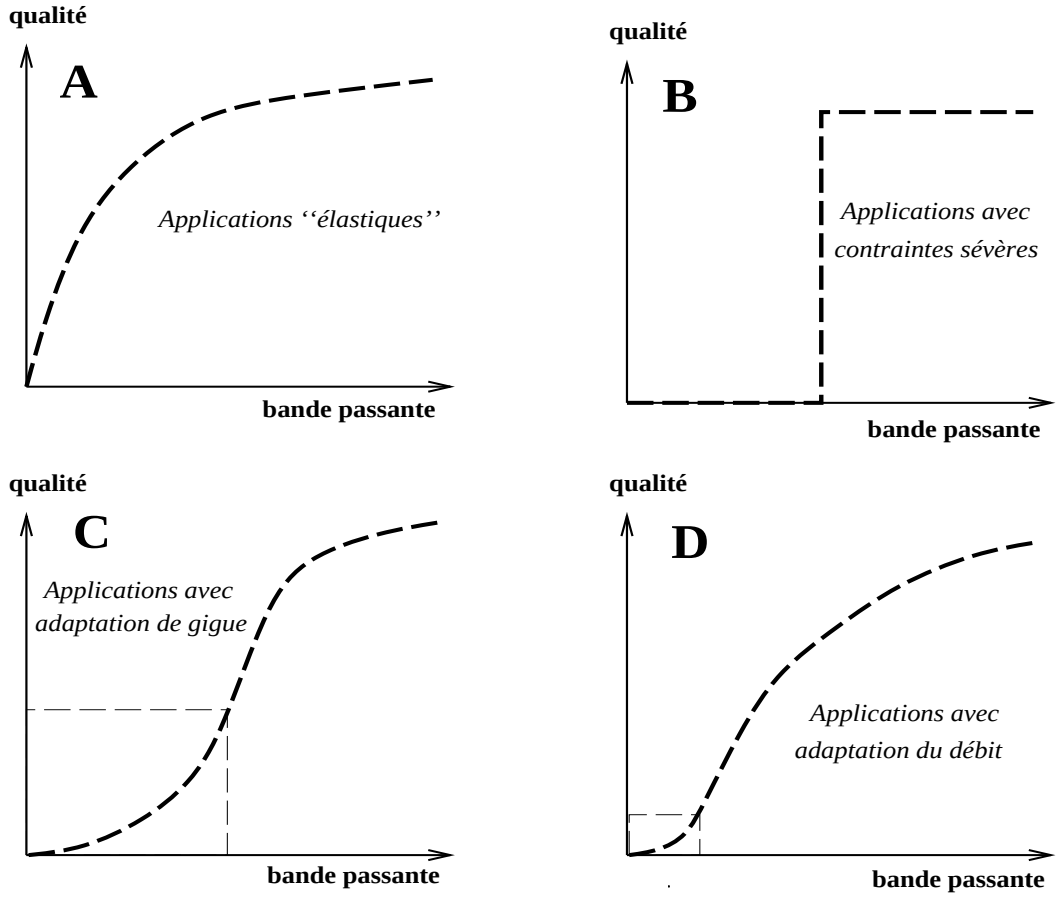


FIG. 6.19 - *Fonctions de satisfaction selon les caractéristiques de l'application*

Les applications traditionnelles, comme la transmission de fichiers, plus tolérantes aux variations de délais que les applications multimédia, sont souvent qua-

lifiées "d'applications élastiques". Leur fonction de satisfaction est concave, ce qui signifie que la satisfaction globale augmente avec le nombre d'utilisateurs (courbe A, figure 6.19). L'admission de contrôle n'a aucune place dans ce cas de figure, ce qui légitimise le choix de conception de l'architecture de l'Internet.

A l'opposé, d'autres applications ont des contraintes temps réel très sévères. Il s'agit par exemple des applications couramment utilisées dans les réseaux à commutation de circuits, comme le téléphone traditionnel. Au delà d'une valeur limite de délai, ces applications ne fonctionnent plus. Leur courbe de satisfaction prend la forme d'une marche d'escalier. Dès que le partage de la bande passante du réseau est tel que les délais dépassent le délai maximal autorisé par l'application, celle-ci ne peut alors plus fonctionner et la qualité s'annule (courbe B, figure 6.19).

Toutefois, on peut faire en sorte que ces mêmes applications utilisent des mécanismes pour s'adapter à la bande passante disponible. Par exemple, on peut utiliser des algorithmes à l'intérieur de l'application pour s'adapter aux variations de délai, e.g. transmission audio et mécanismes de "synchronisation intra-média" [Ferrari93]. En diminuant légèrement l'intéractivité de l'application, les paquets qui arrivent en retard peuvent de cette manière être pris en compte. Cependant, si ces applications ne peuvent pas adapter leur débit en fonction de la bande passante disponible, les performances atteintes se dégradent rapidement lorsque la bande passante disponible devient inférieure au débit de l'application. La fonction de satisfaction est convexe puis concave (courbe C, figure 6.19). On peut aussi utiliser des algorithmes pour adapter le débit de l'application en fonction de la bande passante disponible [Bolot94a]: la convexité de la courbe peut alors se réduire à une région étroite proche de l'origine (courbe D, figure 6.19).

On peut en déduire que l'utilisation au sein des applications de mécanismes d'adaptation à la congestion du réseau permet de repousser la plage d'utilisation du contrôle d'admission. Ce dernier ne devient alors souhaitable que dans les rares cas de congestion où le nombre d'utilisateurs est très grand, ou pour des applications pour lesquelles l'adaptation est impossible.

En résumé, le service nécessaire dépend de la fonction de satisfaction, ce qui implique un impact sur l'architecture des réseaux. Le problème est que, dans le cas général, on ne connaît pas les fonctions de satisfaction des applications (sauf exception, e.g. la courbe concave pour le transfert fiable de fichiers). L'idée serait donc de mettre à la disposition des applications toute une panoplie de services correspondant à une panoplie de fonction de satisfaction. Notons que cette idée est

également au coeur de l'approche ATM qui cherche aussi à fournir une panoplie de services qui soit adaptés aux différents types d'application. La technique ATM permet de réunir les avantages de la commutation de circuits et de la commutation de paquets en particulier en permettant le transport de sources "mixtes" (i.e. a débit variable avec des contraintes de temps réel) comme la vidéo. Pour cela, différents types de services sont offerts:

- Service à débit fixe CBR (*Constant Bit Rate*), e.g pour le téléphone.
- Service à débit variable VBR (*Variable Bit Rate*), pour la vidéo.
- Services "meilleur effort possible" appelé ABR (*Available Bit Rate*) et UBR (*Unspecified Bit Rate*).

La différence principale entre les approches Internet et ATM vient de leurs points de départ. Dans l'approche Internet, le service initial est un service *best effort* auquel on cherche à ajouter d'autres services, en particulier des services avec garantie de performance. Dans l'approche ATM, le service initial est un service CBR prévu pour la transmission de parole auquel ont été ajoutés des services *best effort* et VBR (notons que le service VBR n'est pas encore spécifié).

Une panoplie de services

En passant d'un service unique *best effort* à différents types de services (par exemple, données traditionnelles, données temps réels, etc...), on augmente l'adéquation entre les besoins de l'application et les services offerts et, par conséquent, la satisfaction des utilisateurs. Le choix du service à utiliser peut être soit implicite (à l'initiative du réseau) soit explicite (à l'initiative de l'application).

Dans le cas où le choix du service est implicite, l'application ne nécessite aucune modification car c'est le réseau qui se charge de lui associer implicitement [5] le service requis. Cependant, "l'approche implicite" suppose que le réseau peut identifier le type d'application et en déduire le service approprié. De plus, un seul service est proposé à l'application, ce qui peut parfois s'avérer insuffisant. Par exemple, la même application d'audioconférence peut servir à la fois à une retransmission de séminaire et à une réunion entre plusieurs participants. La première application est

5. L'association peut se faire par inspection du numéro de port que l'application utilise ou par une signature des flots.

beaucoup plus tolérante au délai de "bout en bout" que la deuxième application car elle est moins interactive.

Dans le cas où le choix du service est explicite, le réseau propose un ensemble de services et c'est à l'application de choisir le service le plus approprié à ses besoins, par l'intermédiaire d'une interface de service. Cependant, en laissant le choix du service à l'utilisateur, on prend le risque que ce dernier choisisse toujours le service qui offre les meilleures garanties indépendamment de l'application utilisée. Si tout le monde agit de la sorte, on se retrouve avec un seul type de service, ce qui revient au service offert actuellement par l'Internet. Il ne serait pas raisonnable de baser la fiabilité du système sur le "bon comportement" des utilisateurs du réseau. Pour obliger les utilisateurs à ne pas choisir par défaut le service qui offre les meilleures garanties, une approche consiste à associer un coût à chaque service. Cependant, l'idée de tarification suppose des changements aussi bien dans la mentalité des utilisateurs qu'au sein des applications. Si l'Internet a connu un si grand succès depuis sa création, c'est en partie parce qu'un grand nombre de ressources sont (ou tout au moins semblent aux utilisateurs) disponibles gratuitement. Il n'est pas dit que l'Internet survive un tel bouleversement. Toutefois, dans un premier temps, on peut envisager que seuls les services qui nécessitent le plus de ressources soient payants. D'autre part, le mécanisme de tarification est complexe car il suppose toute une infrastructure associée à la réservation de ressources ainsi qu'une authentification des utilisateurs. Enfin, si l'utilisateur a le choix du service, cela suppose qu'il connaisse les caractéristiques de chacun des services offerts par le réseau et, aussi, que ces services restent inchangés au cours du temps. En effet, si l'on modifie un service qu'une application avait l'habitude d'utiliser, des problèmes de compatibilité peuvent apparaître.

Nous venons de voir comment associer un service particulier à une application. Intéressons nous maintenant au problème d'obtenir ces nouveaux services. Si l'on veut être en mesure d'offrir des services avec des garanties strictes de performance, il est nécessaire d'effectuer une réservation de ressources. Deux principaux protocoles de réservation de ressources ont été développés à ce jour. Il s'agit de ST-II (*Stream Protocol version 2*) [Topolic90], et de RSVP (*Resource reSerVation Protocol*) [Zhang93].

Protocoles de réservation de ressources

Ces deux protocoles comportent plusieurs points communs [Delgrossi93] [Mitzel93]. Tout d'abord, ils permettent de transmettre des données à un ensemble de récepteurs hétérogènes. L'addition ou la suppression d'un récepteur se fait sans devoir réétablir l'ensemble des connexions. Enfin, ils ont tous les deux besoin de connaître les ressources disponibles du réseau afin de prendre les décisions de routage appropriées. Ce dernier point implique des modifications préalables au niveau des routeurs.

La principale différence entre ces deux protocoles est leur position respective dans le support de communication. ST-II se situe au niveau de la couche IP (couche réseau), ce qui le rend incompatible avec l'architecture Internet. En revanche, RSVP s'utilise en conjonction avec IP et ne s'occupe que du contrôle de la transmission et non de la transmission elle-même. D'autre part, ST-II est "orienté-connexion", c'est-à-dire que c'est l'émetteur qui propose un service aux récepteurs en leur transmettant une description du flot de données. Les routeurs intermédiaires ainsi que les récepteurs renvoient ensuite à la source la description originale du flot après l'avoir ajustée en fonction des ressources disponibles. Au contraire, RSVP est un protocole "orienté-récepteur" dans lequel chaque récepteur décrit ses besoins en ressources dans une description de flot, et cette description remonte en direction de l'émetteur. L'hétérogénéité des récepteurs est aussi gérée différemment dans les deux protocoles. Avec le modèle ST-II, l'émetteur a une vue homogène de l'ensemble des récepteurs. En effet, les données émises par l'émetteur doivent pouvoir être supportées par toutes les connexions, ce qui oblige les autres participants à recevoir un service de qualité inférieure à celui qu'ils pourraient théoriquement supporter. D'autre part, comme RSVP est "orienté-récepteur", il permet de construire un arbre multipoint des récepteurs vers l'émetteur en réservant sur chaque liaison le minimum de ressources pour satisfaire le service demandé par chaque récepteur. Il fournit ainsi un support pour un codage hiérarchique (cf. section 6.2.6). Enfin, le protocole ST-II associe la réservation de ressources avec l'établissement d'un circuit virtuel entre source et récepteur, ce qui n'est pas compatible avec la transmission multipoint où le nombre de récepteurs peut dans la pratique être très grand. Le protocole ST-II convient mieux pour des applications sécurisées où l'émetteur a besoin de connaître et contrôler l'ensemble de ses récepteurs. Dans la section suivante, nous détaillons un peu plus les principes du protocole RSVP car il correspond mieux aux besoins des applications de vidéoconférence.

Dans le protocole RSVP, la réservation des ressources est prise en charge par les récepteurs, ce qui permet au protocole de s'adapter à un nombre variable et important de récepteurs. Ainsi, lorsqu'un récepteur veut recevoir un ou plusieurs flots, il doit s'abonner au groupe multipoint correspondant. Les messages de réservation de ressources sont envoyés en parallèle avec le flot de données de l'utilisateur. Ces messages contiennent une description du profil des datagrammes pour lesquels la réservation est faite. D'autre part, le protocole RSVP n'associe pas la réservation au routage: la route empruntée par les paquets d'un flot est découverte automatiquement et les paquets de réservation sont émis sur cette même route. RSVP convient parfaitement aux applications multipoint de type "vidéo à la demande" car, dans ce cas, le flot de données peut être encodé de manière hiérarchique (par exemple en utilisant du sous-bande, cf. section 6.2.6). La source transmet ses données en multipoint et en parallèle avec des messages de chemins qui contiennent une description de tous les flots de données émis. Les routeurs décident quels flots de données doivent être transmis pour chaque récepteur. Le récepteur choisit alors, à partir des messages de chemin qu'il reçoit, le sous-ensemble de flots qu'il souhaite recevoir. Il envoie alors des messages de réservation qui contiennent un filtre ainsi qu'une description de flot. Le message de réservation se propage du récepteur vers la source dans le sens inverse de l'arbre multipoint établi par les messages de chemin. Le filtre peut être soit nul (la totalité du flot est demandée), soit statique (le même filtre pour toute la durée de la connexion), soit dynamique (le filtre est fonction du trafic au cours du temps).

Le protocole RSVP utilise des "états transitoires" pour mettre à jour la liste des réservations en cours. En d'autres termes, cela signifie que les sources doivent périodiquement retransmettre les messages de chemin pour rappeler aux routeurs que les flots existent et pour se faire connaître auprès de nouveaux récepteurs potentiels. De plus, ces retransmissions renforcent la robustesse du protocole contre les pannes temporaires du routeur, de la source ou des récepteurs.

La réservation de ressources est indispensable pour atteindre un niveau de garantie fiable. Mais tout comme la commutation de circuit, elle peut conduire à une sous-utilisation des ressources réservées (e.g. avec réservation du débit crête de l'application [Tenet94]), voire à des abus (e.g. les réservations "antisociales" qui consistent à demander plus de ressources que nécessaire risquent d'entraîner des performances dégradées pour tous les autres utilisateurs). Nous avons déjà mentionné ce problème au début de cette section lors de la description de l'approche explicite

du choix de service. Un moyen de prévenir ce comportement consiste à facturer les utilisateurs en fonction des ressources réservées. Cependant, dans un réseau qui n'appartient pas à un seul fournisseur, la facturation est un problème complexe. En effet, la facturation implique l'authentification des demandes de réservation ainsi que l'établissement d'un contrat entre l'utilisateur et les différents fournisseurs de service. Un mécanisme de facturation pour l'Internet qui utilise un coût variable en fonction de la charge du réseau est décrit dans [Mackie94]. Enfin, si l'on veut garantir la réservation de ressources, il faut lui associer un protocole de contrôle d'admission.

6.4 Conclusion

Pour le moment, IVS est la seule application de vidéoconférence qui intègre un algorithme automatique de contrôle de congestion. Nous avons proposé une solution qui fonctionne dans l'Internet actuel sans qu'aucune modification ne soit nécessaire dans les routeurs ou dans l'architecture de l'Internet. Cependant, cette approche ne résout pas les problèmes relatifs a l'hétérogénéité des récepteurs ou à l'utilisation abusive de la bande passante par des applications trop agressives. Nous avons proposé des solutions immédiates (le codage en sous-bandes, le codage simulcast et l'utilisation de passerelles vidéo) pour le premier problème. Faute de temps, nous n'avons malheureusement pas encore pu les mettre en œuvre, ce qui explique l'absence d'évaluation quantitative dans le chapitre. La non-équité du partage des ressources sera résolue à moyen terme avec l'introduction de changements de discipline au sein des routeurs. L'approche que nous avons proposée n'est donc pas incompatible mais bien au contraire complémentaire avec les changements escomptés à moyen terme dans l'architecture du réseau. La date à laquelle ces changements seront effectifs n'est bien sûr pas fixée. On notera cependant qu'au moins un fabricant de routeurs, à savoir ACC, offre des routeurs qui utilisent la discipline WFQ. L'utilisation généralisée de cette discipline va permettre de se prémunir contre les applications qui sont trop agressives. A plus long terme, on peut imaginer par exemple avec un modèle CBQ et RSVP la configuration suivante. Trois types de services offerts à l'utilisateur:

- un service *best effort* pour les applications traditionnelles,
- un service temps-réel avec réservation de bande passante,

– un service temps-réel sans réservation de bande passante.

Au sein des routeurs, est associé à chaque service un pourcentage de la bande passante disponible (e.g. 40 % pour le *best effort*, 20 % pour le service temps-réel avec réservation de ressources et 40 % pour le service temps-réel sans réservation de ressources).

Le codage en sous-bandes pourra être utilisé pour gérer l'hétérogénéité des récepteurs. Tous les récepteurs pourront s'abonner au flot vidéo de base dont la bande passante pourra être réservée par les récepteurs et qui sera transmis à l'aide du service temps-réel (avec réservation de bande passante). Les flots complémentaires seront transmis à l'aide du service temps-réel mais sans réservation de bande passante et seuls les récepteurs qui pourront se le permettre s'abonneront à ces flots.

En résumé

Nous avons posé le problème du contrôle de congestion pour les applications multimédia. Nous avons apporté une première solution qui utilise du feed-back de la part des récepteurs [Bolot94a]. L'utilisation de l'algorithme de caractérisation de réseau de Ian Wakeman nous a permis de résoudre le problème de l'implosion de feed-back lié à un nombre important de récepteurs [Bolot94b].

Nous avons posé le problème de la transmission vidéo vers un ensemble hétérogène de récepteurs et proposé des solutions [Turletti94a]. Ces solutions sont applicables dans l'Internet actuel: le codage simulcast, les passerelles vidéo et le codage en sous-bandes (sous réserve de pouvoir garantir la bonne réception du flot de base).

Enfin, on peut ajouter que les applications d'audio et de vidéoconférence comme IVS, NV ou VAT ont contribué indirectement à la réflexion actuelle sur les impacts des applications multimédia sur l'architecture des réseaux.

Chapitre 7

Conclusion

Résumé de nos travaux

A l'époque où nos travaux ont débuté (en fin 1991), la vidéoconférence semblait réservée à une élite. Elle était souvent associée à l'idée de garanties de ressources strictes, de liaisons à très grande bande passante, et à de coûteux codecs cablés.

Nos travaux ont contribué à montrer que la technologie actuelle permet de réaliser des codecs vidéo en logiciel et donc à moindre coût. Le codec IVS a été l'un des premiers logiciel de vidéoconférence élaborés dans le monde.

Nous avons montré qu'une application de vidéoconférence ne nécessite pas obligatoirement des débits très élevés et peut s'adapter à des réseaux qui n'offrent aucune garantie de service.

Les mécanismes d'adaptation de l'application aux conditions du réseau sont décrits pour un type particulier de codage vidéo. La transmission de sources vidéo nécessite une bande passante relativement importante, et il est donc indispensable pour des raisons économiques, d'utiliser des algorithmes de compression numérique de l'information. Nous avons choisi le standard de compression H.261 car ce dernier représentait l'état de l'art des techniques de compression de données à bas débit. Son utilisation sur l'Internet pose un certain nombre de problèmes car ce standard a été à l'origine conçu pour être utilisé sur des réseaux à commutation de circuits de type RNIS. Principalement, nous avons été amenés à développer les algorithmes suivants:

- découpage en paquets du flot de bits H.261,

- contrôle d'erreurs contre la perte des paquets,

- contrôle de débit pour codeur H.261,
- contrôle de congestion pour adapter le débit du codeur à la bande passante disponible du réseau.

Enfin la dernière contribution porte sur l'intégration des mécanismes que nous avons proposés étant donné l'évolution probable de l'Internet. Nous avons montré que ces mécanismes gagneront en efficacité lorsque les routeurs intégreront de nouvelles disciplines.

Comme nous l'avons noté dans l'introduction, la vidéoconférence sur l'Internet regroupe un grand nombre de domaines et seuls ont été abordés les problèmes liés à la transmission de sources vidéo H.261. Les schémas que nous avons décrits peuvent facilement s'étendre à d'autres algorithmes de compression vidéo (par exemple MPEG). L'algorithme d'estimation de l'état du réseau que nous avons décrit peut aussi servir à adapter le débit de sources audio.

Quelques précisions

Tout au long du manuscrit, le terme "nous" apparait pour désigner les personnes qui ont collaboré à mes travaux. Pour être plus précis, je les nomme à la suite ce qui me permettra par la même occasion de les en remercier.

Dans le deuxième chapitre, il est question du codage H.261 et de sa mise en œuvre logicielle. Précisons que lorsque j'ai commencé ma thèse, je ne disposais d'aucun exemple de logiciel de vidéoconférence. Il a donc fallu que j'étudie et que je mette en oeuvre toutes les composantes d'un tel outil, à savoir, le codec lui-même, l'interface avec le réseau, l'interface avec la carte d'acquisition vidéo, l'interface avec l'utilisateur, le protocole de contrôle de session entre les participants de la conférence, etc. La première version du logiciel a été mise dans le domaine public en un peu moins d'un an. A partir de là, j'ai reçu beaucoup de contributions de personnes extérieures à l'INRIA qui se sont portées volontaires pour effectuer l'interfaçage avec d'autres cartes d'acquisition vidéo. Un autre groupe à l'INRIA Rocquencourt a plus spécifiquement travaillé sur les problèmes d'affichage de couleurs, de librairie bas-niveaux avec la carte d'acquisition vidéo PARALLAX et aussi sur une réarchitecture du code. La plupart de leurs travaux ont été intégrés dans IVS; ce même groupe a créé à partir d'IVS leur propre logiciel de vidéoconférence (nommé TELESIA [1]). J'aimerais remercier Pierre Delamotte, Andrzej Wozniak et tous ceux qui

1. Ce logiciel incorpore en outre un protocole de session de type maître-esclave.

ont participé à l'écriture de mon logiciel.

Dans le troisième chapitre, nous abordons le problème de la mise en paquets du flot vidéo H.261. Ce schéma de mise en paquets a été écrit en collaboration avec Christian Huitema. Il est actuellement soumis à discussion dans le groupe de travail AVT (*Audio Video Transport*) de l'IETF [Turletti95]. Je tiens à remercier en particulier Steve Casner, Steve Mac Canne et Mark Handley pour leurs commentaires judicieux.

Les chapitres 4 et 5 traitent du contrôle d'erreur et du contrôle de débit. Les algorithmes qui ont été mis en œuvre sont les miens.

Enfin, le chapitre 6 concerne le contrôle de congestion. Jean Bolot et moi avons étudié et mis en œuvre un premier algorithme de contrôle de congestion [Bolot94a]. Nous avons ensuite utilisé l'algorithme de caractérisation du réseau de Ian Wakeman qui présente l'avantage de fonctionner quelquesoit le nombre de participants [Bolot94b]. Enfin, pour le problème de distribution de vidéo à un ensemble de récepteurs hétérogènes, les différentes solutions ont été proposées par Jean Bolot et moi-même [Turletti94a].

Critique de l'existant

Nous récapitulons dans ce paragraphe les principales critiques des travaux effectués et introduisons les travaux à venir.

- ré-architecture du codec: nous avons vu que l'architecture actuelle d'IVS utilisait autant de processus décodeurs que de flots vidéo (voir section 2.2.1). Des travaux sont en cours pour permettre à un seul processus de décoder l'ensemble des flots vidéo de la session permettant d'augmenter les performances du codec.

- la méthode de détection de mouvement actuellement mis en œuvre dans IVS est sensible aux variations de luminosité de l'image (voir section 2.2.2). Nous comptons expérimenter d'autres algorithmes dans un proche avenir.

- la TCD utilisée dans le codage H.261 a plusieurs inconvénients. Les algorithmes rapides de TCD ont tout de même un coût de calcul important qui peut s'avérer prohibitif pour des stations de travail peu puissantes. D'autre

part, la TCD se révèle être peu adaptée à la transmission de données textuelles, par exemple, les "transparents" et autres tableaux projetés pendant les conférences. Leurs "lettres" comportent davantage de composantes hautes fréquences que les scènes classiques de vidéoconférence. Une solution consiste à utiliser une autre type de transformée comme la transformée de Haar. Cette transformée est d'ailleurs utilisée par le logiciel de vidéoconférence NV (voir Annexe C.2.

- nous n'avons pas encore expérimenté les méthodes de correction d'erreurs à base d'ajout de redondance (voir section 4.2.1). Ces méthodes devraient permettre de réduire de manière significative l'effet de la perte de paquets sur la vidéo. En effet, l'ajout de redondance dans le codage permet de reconstituer la plupart des paquets perdus. Ainsi l'affichage erroné transitoire dû à la perte de paquets qui subsiste jusqu'au rafraîchissement intra-image suivant peut être réduit.

- pour permettre de résoudre le cas des récepteurs surchargés, le protocole de contrôle de congestion actuel nécessite un nouvel indicateur. Ce dernier servira à indiquer que la cause de la perte des paquets provient d'une surcharge du récepteur et non d'une congestion du réseau. Le codeur pourra alors prendre les mesures adéquates et passer automatiquement dans le mode qui convient (voir section 6.2.5).

- nous n'avons pas encore mis en œuvre les passerelles vidéo et le codage simulcast proposés dans le chapitre 6. Nous espérons le faire dans un proche avenir.

- la version actuelle 3.4 d'IVS utilise la version 1 de RTP. La version 2 de RTP est sur le point d'être standardisée. Les principales différences sont une valeur de timestamp dont la fréquence d'horloge dépend du média utilisé, l'ajout d'un champ de 32 bits dans l'en-tête de RTP pour identifier de manière unique les différentes sources d'émission dans la session et surtout l'ajout des paquets RTCP de rapport d'émission et de réception. Ces rapports sont émis en multipoint par tous les émetteurs et récepteurs de la session, ce qui permet aussi aux récepteurs d'avoir une estimée de l'état du réseau. Pour éviter le problème d'implosion de feed-back, la fréquence d'émission de ces rapports est fonction du nombre de participants dans la session. L'algorithme proposé

dans la version 2 du protocole RTP remplacera celui utilisé actuellement dans IVS (i.e. l'algorithme d'estimation de l'état du réseau de Ian Wakeman). Nous espérons l'intégrer dans la nouvelle architecture d'IVS prévue pour juin 1995.

Perspectives de recherche et travaux futurs

Nos travaux servent de base à d'autres travaux et réflexions qui ont par ailleurs déjà commencé.

L'hétérogénéité du réseau et l'utilisation prochaine de mécanismes de réservation de ressources donnent un regain d'intérêts aux techniques de codage vidéo hiérarchique. Ces techniques de codage font l'objet de nombreuses recherches récentes (cf. par exemple les articles dans la sixième conférence *Packet Video* qui s'est tenue à Portland en septembre 1994). Nos travaux s'orientent vers l'utilisation de techniques à base d'ondelettes [Antonini92].

Nous proposons également d'étendre la plage de variation des débits du codeur vidéo à la fois dans le domaine des haut débits (vidéo de haute qualité) et des très bas débits. Nous souhaitons porter le logiciel IVS sur un environnement de mobiles. Ce nouvel environnement dont les caractéristiques sont différentes de l'environnement standard Internet (par exemple des liaisons à très bas débit) permettra d'étendre l'adaptabilité de l'application au réseau en accord avec le modèle NCA (*Network Conscious Applications*) [Diot95].

Les stations de travail actuelles sont assez puissantes pour encoder et décoder des sources vidéo à plus de 20 images par seconde. L'ajout de techniques de synchronisation entre les flots audio et vidéo permettra d'améliorer la qualité de réception [Escobar92] [Ramanathan92].

Aujourd'hui, la vidéoconférence connaît un essor important et de nombreux instituts de recherche s'y intéressent de près. Ainsi, en parallèle à nos travaux sont abordés les problèmes relatifs au contrôle de la session et aux problèmes de sécurité de la transmission.

Les problèmes liés au contrôle de session des conférences concernent la gestion et la coordination des conférences multimédia qui comportent un nombre quelconque de participants. Ces problèmes sont en cours d'étude au sein du groupe de travail MMUSIC (*Multiparty MUltimedia SessIon Control*) de l'IETF. Un des buts étant

de spécifier un protocole de contrôle de session commun à tous les logiciels de vidéoconférence sur l'Internet.

Les problèmes relatifs à la sécurité des flots audio et vidéo transmis sur l'Internet sont en cours d'étude dans le cadre du projet MICE II[2]. Ces problèmes concernent la politique de contrôle d'accès à la conférence ainsi que les mécanismes de cryptage de l'information et l'échange des clefs de cryptage entre les participants de la conférence.

2. Le projet Européen MICE II a pour objectif d'améliorer les outils de travail en commun développés dans le projet MICE (en particulier, il est prévu de les utiliser sur des réseaux à haut débit et d'y intégrer des mécanismes de sécurité).

Annexe A

La transmission multipoint

Le besoin de la transmission multipoint, c'est-à-dire d'une ou plusieurs sources vers un nombre variable de récepteurs, est apparu récemment avec la venue de nouvelles applications distribuées comme l'audioconférence et la vidéoconférence.

La transmission multipoint d'une source vers N récepteurs comporte deux avantages principaux comparé à N transmissions point à point distinctes entre la source et chacun des N récepteurs.

Elle permet tout d'abord de rendre plus facile la gestion d'un nombre variable de récepteurs. En effet, dans le cas de transmissions point-à-point, chaque arrivée et chaque départ d'un participant dans le groupe doit s'accompagner de messages de signalisation explicites du participant vers tous les émetteurs du groupe. Dans le cas de transmissions multipoint, l'abonnement et le désabonnement à un groupe se fait de la manière suivante. Quand un hôte joint un groupe, il envoie un message IGMP [1] à une adresse multipoint réservée afin de notifier sa participation au groupe. L'appartenance à un groupe étant dynamique, les routeurs multipoint s'échangent périodiquement des requêtes pour mettre à jour la liste des membres de chaque groupe.

La transmission multipoint permet d'autre part de réduire le trafic total échangé entre la source et les récepteurs. Dans le cas d'une transmission point-à-point, il peut y avoir jusqu'à N copies du même paquet en transit sur une liaison (si cette liaison est commune aux N connexions). En revanche, dans le cas de la transmission multipoint, chaque liaison est utilisée au plus une fois pour le même paquet. La figure A.1 montre un exemple de transmission point-à-point et multipoint avec une

1. IGMP *Internet Group Membership Protocol* est défini dans le RFC [RFC1112].

source **S** et quatre récepteurs. Il apparaît que dans le cas de la transmission point-à-point, 3 copies de chaque paquet transitent via la liaison **a** pour atteindre les récepteurs **R1**, **R2** et **R3**. En revanche, une seule copie est transmise via la liaison **a** dans le cas de la transmission multipoint, le routeur r1 se chargeant d'envoyer une copie de chaque paquet sur les liaisons **b** et **c**.

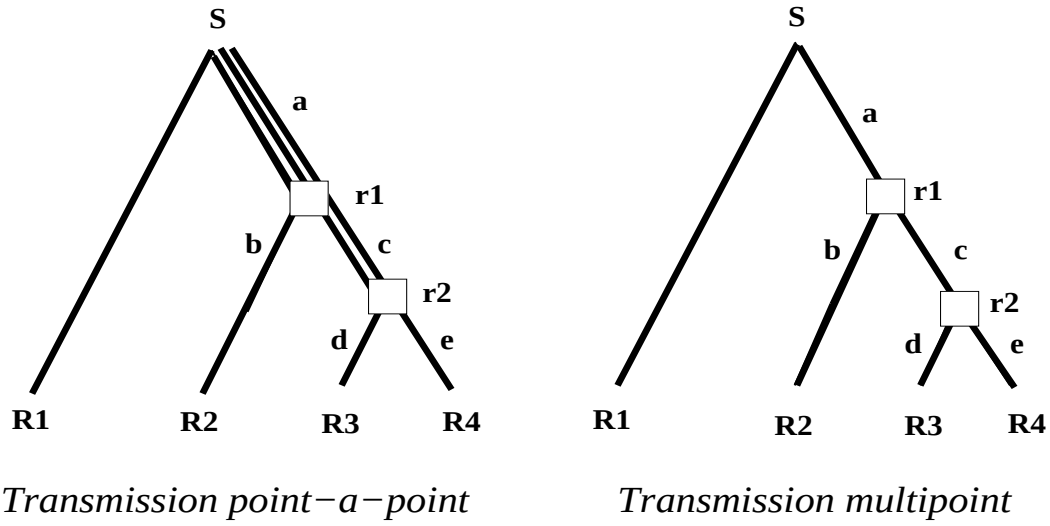


FIG. A.1 - *Comparaison entre transmission point-à-point et multipoint*

Pour joindre une conférence émise en multipoint, un participant doit lancer l'application de vidéoconférence avec l'adresse multipoint et le numéro de port où se tient la conférence. Cette adresse multipoint, appelée adresse de groupe, appartient à une classe d'adresses réservées, appelée classe D. Les quatre premiers bits d'une adresse de classe D contiennent la valeur 1110 et les 28 bits restants spécifient l'adresse multipoint. Ainsi, la gamme d'adresses multipoint s'étale entre 224.0.0.0 et 239.255.255.255.

Les travaux sur la transmission multipoint sur Internet sont mis en pratique dans le cadre du MBONE [Huitema94] [Macedonia94] ou "Multicast backBONE" ("réseau d'interconnexion multipoint" en français). Le MBONE est un réseau virtuel qui s'étend au dessus de l'Internet afin de permettre l'établissement de transmissions multipoint.

Le MBONE était encore récemment un réseau expérimental qui ne concernait qu'une toute petite fraction des routeurs de l'Internet. Il s'étend aujourd'hui à tous les continents et inclut une trentaine de pays. Les routeurs, distribués sur tout le réseau utilisent leurs propres protocoles de routage, e.g. DVMRP (*Distance Vector*

Multipoint Routing Protocol) [Deering91], en attendant que tous les routeurs de l'Internet acceptent d'acheminer des adresses de classe D. La solution adoptée consiste à utiliser des "îlots multipoint", c'est-à-dire des zones où le routage multipoint est supporté par des liaisons "virtuelles" passant par le réseau IP point-à-point classique. Ces liaisons sont appelées des "tunnels". La figure A.2 donne une vue d'ensemble du MBONE au mois de mai 1994.

L'îlot multipoint le plus simple est un réseau local qui supporte le multipoint. Un standard [RFC1112] spécifie les extensions nécessaires à IP pour émettre et recevoir en multipoint des paquets sur une interface IP. Prenons le cas d'un réseau Ethernet: le passage entre une adresse de groupe Internet et une adresse de groupe Ethernet se fait en plaçant les 23 bits de poids faible de l'adresse multipoint dans les 23 bits de poids faible de l'adresse Ethernet multipoint qui est 1.0.94.0.0.0. Par exemple, l'adresse IP multipoint 224.8.8.8 devient l'adresse Ethernet 1.0.94.8.8.8. Notons que puisqu'il y a 28 chiffres significatifs dans une adresse de groupe Internet, plusieurs adresses de groupe Internet peuvent avoir la même adresse de groupe Ethernet, ce qui peut générer des conflits entre plusieurs groupes multipoint.

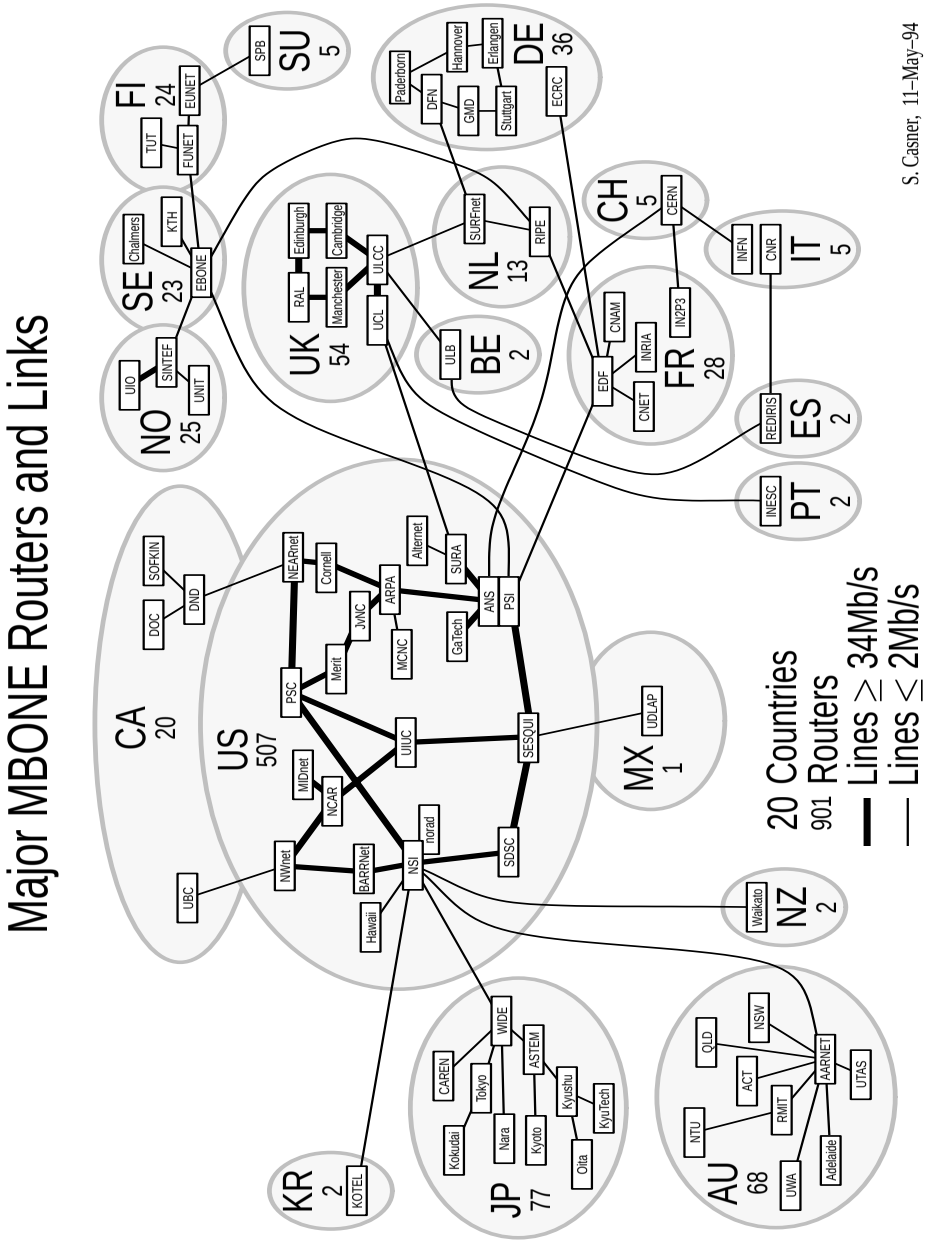


FIG. A.2 - *Vue d'ensemble du MBONE au mois de mai 1994*

Annexe B

Evaluation de l'algorithme de sondage

Nous donnons dans cette annexe une évaluation du mécanisme de sondage décrit en section 6.2.1. On vérifie tout d'abord que le mécanisme de caractérisation du réseau n'engendre pas de problème "d'implosion de feed-back". On calcule ensuite le temps maximal au bout duquel le pire état du réseau est reçu par la source.

B.1 Le problème "d'implosion de feed-back"

Commençons par calculer le nombre moyen de sondages nécessaires dans une époque pour que l'émetteur reçoive une première réponse. Cela revient à calculer le nombre de sondages utiles pour que les bits significatifs de la clef d'un récepteur soient identiques aux bits significatifs de la clef émise par l'émetteur.

Soient n le nombre de récepteurs dans l'arbre multipoint et i la longueur de la clef exprimée en bits qui est envoyée par l'émetteur. Le premier sondage d'une époque porte le numéro 0. Soient r_j le nombre de réponses renvoyées à l'émetteur pendant le sondage j et p_j la probabilité pour qu'un récepteur réponde au sondage j. On suppose dans cette démonstration que les paquets de contrôle (e.g. REQUETE, REPONSE) ne sont pas perdus dans le réseau. La probabilité d'obtenir au moins une réponse dans une époque quand aucune réponse dans le sondage précédent n'a été reçue est donnée par:

$$\Pr(r_j > 0 \,|\, r_{j-1} = 0) = 1 - (1 - p_j)^n$$

où

$$p_j = \begin{cases} 2^{-i} & \text{si } j = 0 \\ \frac{2^{j-1}}{2^i - 2^{j-1}} & \text{si } j > 0 \end{cases}$$

En effet, au j-ième sondage, on avait déjà essayé 2^{j-1} nombres dans l'espace de recherche. Le nombre moyen E de sondages pour recevoir la première réponse est donc:

$$\begin{aligned} E &= \sum_{j=1}^{i} j(1-(1-p_j)^n)(1-2^{j-1-i})^n \\ &\quad +1-(1-2^{-i})^n \end{aligned}$$

La figure B.1 montre le graphe de logarithme (E), en fonction de n pour n compris entre 0 et 10000. Si l'on approxime la courbe par une droite, on obtient:

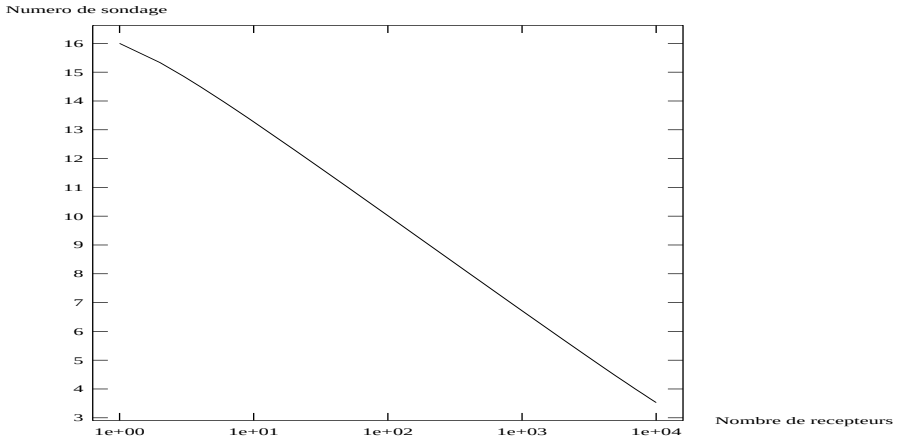


FIG. B.1 - *Numéro de sondage pour lequel on reçoit la première réponse en fonction du nombre de récepteurs*

$$n \sim e^{16.25 - E/1.4} \tag{B.1}$$

On peut montrer par une simple manipulation de l'équation B.1 qu'en moyenne, à une différence de 6 sondages correspond un rapport de 1 sur 72, c'est-à-dire à environ 1.4%.

Le modèle ci-dessus montre que l'algorithme élimine le problème d'implosion de feed-back. Considérons à présent un algorithme simplifié dans lequel une époque se termine dès qu'une réponse est reçue par l'émetteur. Dans ce cas, nous avons:

$$\Pr(r_j = m) = \Pr(r_j = m | r_{j-1} \neq 0) \Pr(r_{j-1} = 0) \tag{B.2}$$

La probabilité conditionnelle $\Pr(r_j = m | r_{j-1} \neq 0)$ est distribuée selon la loi binomiale de paramètre $p = 2^{j-1}/(2^i - 2^{j-1})$ et de nombre n. La table ci-dessous montre la probabilité pour qu'un émetteur reçoive plus de 10 réponses au cours d'une époque, en fonction du numéro de sondages. Les valeurs sont calculées pour 1000 récepteurs. Les sondages pour lesquels la probabilité est inférieure à 10^{-10} ne sont pas montrés. Nous voyons que la probabilité pour qu'un émetteur reçoive plus de 10 réponses au cours d'une époque est très faible et atteint une valeur maximale de 10^{-6} au sixième sondage d'une époque.

Numéro de sondage	4	5	6	7	8
Pr[réponses > 10]	$2\ 10^{-8}$	$4\ 10^{-6}$	$9\ 10^{-5}$	$2\ 10^{-5}$	$3\ 10^{-9}$

Les valeurs numériques indiquent clairement que la probabilité de recevoir un grand nombre de réponses pendant le premier sondage est très faible.

B.2 Temps maximal de réaction de l'algorithme

L'adaptation de l'algorithme à un nombre quelconque de participants est illustrée par l'équation B.1, c'est-à-dire par la relation logarithmique qui existe entre la probabilité de recevoir une réponse et le nombre de récepteurs.

Comparons à présent cet algorithme avec un schéma à retard aléatoire (cf. section 6.2.1) pour un groupe de 10000 récepteurs. Si l'émetteur désire limiter le taux de réponses à ses requêtes à 10 réponses par seconde, chaque récepteur retarder l'émission de son message de feed-back d'un temps aléatoire tiré au sort entre 0 et 1000 secondes. Cette valeur de 1000 secondes (environ 15 minutes) ne permet pas de suivre assez rapidement les variations de condition du réseau. Avec notre algorithme, le temps maximal de réponse est dans le pire cas égal à 32 fois la valeur maximale du rtt du groupe. Par exemple, en prenant une valeur relativement élevée de 500 ms pour le rtt, le codeur connaîtra le pire état du réseau en moins de 16 secondes. Un autre avantage est notre algorithme donne la préférence d'émission de feed-back à un récepteur congestionné. De ce fait, plus le nombre de récepteurs congestionnés est élevé et plus vite l'algorithme peut rendre compte du problème à la source qui pourra alors réagir plus vite à une situation de congestion.

Annexe C

Etat de l'art

Cette annexe décrit brièvement les principaux standards de compression vidéo et les principaux logiciels de vidéoconférence qui existent aujourd'hui dans le domaine public.

C.1 Les standards de compression vidéo, hormis H.261

C.1.1 CellB

CellB n'est pas un vraiment un standard mais un algorithme de compression vidéo à faible coût de calcul développé par Sun Microsystems [Sunvideo93]. L'algorithme de compression vidéo CellB effectue un découpage des images en cellules (ou blocs) de 16 pixels (4×4). Chaque cellule est encodée par un nombre constant de 4 octets. Les deux premiers octets correspondent à un masque de 16 bits qui correspond aux 16 pixels de la cellule. Les deux octets suivants spécifient les deux couleurs (C0 et C1) qui seront utilisées dans la cellule courante. Un bit à zéro (à un) dans le masque signifie que l'on associe la couleur C0 (resp. C1) au pixel correspondant dans la cellule. Cet algorithme est intégré dans la carte d'acquisition (et de compression) de données SunVideo[1] et est aussi utilisé par les applications de vidéoconférence NV et VIC.

C.1.2 JPEG

JPEG (*Joint Photographic Experts Group*) est le nom d'un groupe d'experts en normalisation dont le rôle est d'établir un standard de compression pour les images

1. La carte SunVideo est développée par Sun Microsystems.

fixes. Le nom officiel du groupe est ISO[2]/IEC[3] JTC1[4] SC29[5] WG 10[6]. L'algorithme de compression vidéo est voisin de celui de H.261. Il utilise tout comme H.261, la TCD, la quantification scalaire et le codage de Huffman mais il n'utilise que le mode intra-image. D'autre part, la méthode de quantification est différente: JPEG n'utilise pas le même quantificateur pour encoder tous les coefficients à l'intérieur d'un même bloc. De cette manière, il est possible d'accorder plus de précisions aux coefficients basse-fréquence des blocs. De plus, avec JPEG, l'utilisateur a la possibilité de choisir ses propres tables de quantification.

JPEG a été défini pour des applications d'archivage d'images fixes. Cependant, il est possible de l'utiliser pour encoder les images animées en les considérant comme une suite d'images fixes. L'inconvénient majeur est que l'on ne tient alors pas compte de la redondance temporelle qui existe entre les images successives. Récemment, avec l'arrivée des codecs JPEG cablés sur le marché (par exemple dans la carte d'acquisition et de compression vidéo PARALLAX[7]), on trouve plusieurs applications de vidéoconférence qui utilisent la compression JPEG pour les images animées (e.g. VideoTool[8], NV et VIC).

C.1.3 MPEG

MPEG (*Moving Picture Experts Group*) est le nom d'un groupe d'experts en codage vidéo qui a pour objectif de définir des standards de compression pour les signaux audio et vidéo. MPEG n'est en fait qu'un surnom, le nom officiel du groupe d'experts est ISO/IEC JTC1 SC29 WG 11[9].

Chaque norme MPEG spécifie la représentation codée d'un signal vidéo, ce qui définit implicitement la procédure de décodage associée. La seule contrainte imposée au codeur est d'être compatible avec les décodeurs qui sont spécifiés.

Actuellement, on compte quatre standards MPEG: MPEG 1 et 2 sont normalisés, MPEG 3 est obsolète et MEPG 4 est en cours de spécification.

---

2. ISO: *International Organization for Standardization*
3. IEC: *International Electro-technical Commission*
4. JTC1: *Joint Technical Committee 1*
5. SC29: *Sub-committee 29*
6. WG10: *Working Group 10* (images fixes).
7. PARALLAX Graphics, Inc.
8. VideoTool est un outil de vidéoconférence développé par Parallax Graphics, Inc.
9. WG11: *Working Group 11* (images animées avec audio).

Le standard MPEG 1

Le standard MPEG 1 a été finalisé dans le courant de l'année 1993. La représentation proposée autorise une lecture avant à vitesse normale, ainsi que des fonctions "magnétoscopes" classiques telles que la lecture arrière à vitesse normale, la lecture avant/arrière accélérée, l'accès aléatoire à une image, la pause et l'arrêt sur image. Ce standard est compatible avec les formats de télévision actuels 525 lignes / 60 Hz ou 625 lignes / 50 Hz. En fait, MPEG-1 peut utiliser n'importe quelle taille d'image inférieure à 4095×4095 pixels et jusqu'à une fréquence maximale de 60 images par seconde.

A l'origine, MPEG 1 a été optimisé pour des applications de type CD-ROM (disque optique) à 1.5 Mbps

Tout comme le standard H.261, MPEG 1 utilise un codage hybride à prédiction-transformation (cf. section 2.1.3). Cependant, la boucle de prédiction temporelle de MPEG 1 est plus complexe car elle intègre une prédiction bi-directionnelle. Il existe donc trois sortes d'images: celles encodées en mode intra-image (I), celles prédites (ou encodées en mode inter-image) (P) et enfin les images bi-directionnelles (B) qui sont prédites à la fois par rapport à l'image précédente et l'image qui suit (de type I ou P). H.261 n'utilise que les deux premiers types d'image (I et P). Une séquence d'image MPEG 1 peut prendre la forme suivante:

IBBPBBPBBPBBIBBPBBPBBPBBI...

Notons que les images B améliorent sensiblement [10] le SNR pour des débits inférieurs à 2 Mbps. Cependant les images B apparaissent moins utiles pour des débits plus élevés et augmentent dans tous les cas la complexité de calcul de l'algorithme et la taille mémoire nécessaire. De plus, elles ajoutent un retard supplémentaire au codage et au décodage, ce qui nuit à la contrainte d'intéractivité des applications de vidéoconférence.

On peut ajouter que MPEG 1 utilise une méthode de quantification différente de celle de H.261 pour les images encodées en mode intra-image (I). Cette méthode se rapproche de celle de JPEG. D'autre part, la précision des vecteurs de mouvements est moins grande dans le standard H.261 car les composantes des vecteurs de mouvement ne peuvent pas excéder 15 pixels d'amplitude [11].

---

10. Les images B augmentent le SNR de 2 dB lorsque le débit est de 1.15 Mbps.

11. Ce choix est légitime pour les applications cibles de H.261 (vidéoconférence) où les mouvements dans l'image sont généralement de faible amplitude.

Le standard MPEG 2

En novembre 1994 ont été finalisés les standards ISO/IEC 13818-1 (Systèmes MPEG 2), ISO/IEC 13818-2 (Vidéo MPEG 2) et ISO/IEC 13818-3 (Audio MPEG 2). Les concepts de MPEG 2 sont similaires à ceux de MPEG 1, mais incluent des extensions pour couvrir un plus grand nombre d'applications. L'application cible de MPEG 2 est la transmission numérisée des signaux télévisés à des débits compris entre 4 et 9 Mbps. Cependant, la syntaxe de MPEG 2 convient parfaitement à des applications de TéléVision à Haute Définition (TVHD). L'amélioration la plus significative par rapport à MPEG 1 est l'addition d'une syntaxe efficace pour le codage de la vidéo entrelacée[12]. Comme autres améliorations, on peut noter la possibilité d'encoder la composante continue des blocs de coefficients de TCD avec un nombre variable de bits (de 8 à 11), une quantification non linéaire et de nouvelles tables de codage de Huffman.

Le standard MPEG 3

Le standard MPEG 3 avait pour cible les applications TVHD avec des formats d'image pouvant aller jusqu'à 1920×1080 pixels à une fréquence de 30 images par seconde et des débits compris entre 20 et 40 Mbps. Il s'est avéré plus tard que la syntaxe de MPEG 2 convenait aussi pour les applications TVHD. Aussi, la spécification de MPEG 3 est aujourd'hui abandonnée.

Le standard MPEG 4

Le standard MPEG 4 spécifiera un codage audio et vidéo à très bas débit (de 4.8 à 64kbps). Les travaux ont commencé officiellement avec la réunion MPEG de Bruxelles de septembre 1993 et la spécification est attendue dans le courant de l'année 1997. Le champ d'application est varié et comprend les mobiles, le vidéophone, le courrier multimédia électronique, les bases multimédia intéractives, etc.

C.2 Les logiciels de vidéoconférence, hormis IVS

Actuellement, sur l'Internet, on trouve quatre principaux logiciels de vidéoconférence disponibles dans le domaine public: CU-SeeMe, IVS, NV et VIC. Nous dé-

12. Les signaux vidéo utilisés actuellement en télévision sont entrelacés (i.e. une image contient les lignes paires et la suivante les lignes impaires).

crivons brièvement dans cette section les algorithmes utilisés dans ces différents logiciels.

C.2.1 CU-SeeMe

CU-SeeMe[Dorcey95] a été conçu spécialement pour des plate-formes peu puissantes de type Macintosh et PC/Windows. Il a été écrit par Tim Dorcey à l'Université de Cornell, Etats Unis. CU-SeeMe autorise deux résolutions d'image, une taille de 320×240 pixels (qui se rapproche de CIF (352×288) et une taille de 160×120 pixels (QCIF vaut 176×144). Il n'utilise pas la transmission multipoint mais fait appel à des réflecteurs[13] pour des conférences dont la taille excède deux participants. L'algorithme de codage vidéo n'est pas standardisé et son coût de calcul est très faible. Chaque pixel est représenté par un niveau de luminance codé sur 4 bits. L'image est ensuite découpée en blocs de 8×8 pixels et un algorithme de détection de mouvement est appliqué à ces blocs. Ensuite, CU-SeeMe applique un algorithme de compression de données sans perte d'information aux blocs détectés. Cet algorithme utilise la propriété de redondance qui existe entre les pixels voisins d'un bloc de manière à les coder avec le moins de bits possible. Le taux de compression de cet algorithme est de l'ordre de 40 %.

C.2.2 NV

Le logiciel de vidéoconférence NV[Frederick94] a été écrit par Ron Frederick à Xerox PARC, Etats Unis. Il intègre plusieurs codecs vidéo comme JPEG, CellB et aussi un algorithme non standardisé qui porte son nom (NV). L'algorithme NV découpe tout d'abord l'image en blocs de 8×8 pixels. Ensuite, une détection de mouvement est appliquée aux blocs de l'image et les blocs détectés subissent une Transformée de Haar (cf. section 2.1.3). Les coefficients obtenus sont seuillés de manière à éliminer les termes de basse énergie. Enfin, ces coefficients sont encodés via un algorithme de compression de données sans perte d'information. Pour améliorer la qualité d'image, NV retransmet les zones d'image qui sont stationnaires avec plus de précision (en utilisant le même algorithme que précédemment mais sans l'opération de seuillage des coefficients). NV permet de régler manuellement le débit de sortie du codec.

13. Un réflecteur est un programme dont le rôle est de dupliquer redistribuer les flots de paquets qu'il reçoit à l'ensemble des participants de la conférence.

C.2.3 VIC

Le logiciel de vidéoconférence VIC[McCanne94] a été écrit par Steven Mac Canne et Van Jacobson à Lawrence Berkeley Laboratory (LBL), Etats Unis. VIC est le plus récent des logiciels cités, il a été mis dans le domaine public au mois de décembre 1994. Il intègre plusieurs codecs vidéo: il est capable d'encoder au format H.261, au format NV mais aussi au format CellB et au format JPEG. A la différence d'IVS, VIC n'utilise que le mode intra-image. Il ne met pas en œuvre de méthode de correction d'erreurs à base de feed-back comme IVS. VIC permet tout comme NV de régler manuellement le débit de sortie du codec. Il utilise la nouvelle version 2 de RTP mais est aussi compatible avec la version précédente de RTP, ce qui lui permet l'intéropérabilité avec IVS et NV.

C.3 Comparaison des logiciels IVS, NV et VIC

Il est difficile de comparer équitablement les différents logiciels pour plusieurs raisons:

- les différents logiciels subissent très souvent des améliorations, ces comparaisons ne peuvent être valides qu'un laps de temps très court.
- il faudrait pouvoir comparer chaque module des codeurs séparément. Par exemple, les taux de compression obtenus dépendent fortement de l'opération de détection de mouvement qui est différente dans chacun des logiciels.
- les séquences d'images de test doivent être les mêmes pour chacun des logiciels. Cependant, cette contrainte est difficile à satisfaire en raison du rythme d'acquisition des images qui peut différer selon la puissance disponible des machines. En effet, le rythme d'acquisition des images peut dépendre du coût de l'algorithme de codage utilisé lorsque la machine est peu puissante ou de la politique de contrôle de débit utilisé (par exemple selon que l'on utilise le mode qui favorise la qualité des images ou le mode qui favorise la fréquence d'image dans IVS).

Nous comparons ici les versions 3.4 d'IVS, 3.3 de NV et 2.6 de VIC sur une plateforme constituée d'une station de travail SUN SS20 bi-processeur avec la carte d'acquisition vidéo SunVideo. Les courbes de la figure C.1 montrent la fréquence d'image en fonction du débit utilisé pour les logiciels NV, VIC en mode H.261 avec

un quantificateur de 3 et 10 et IVS avec le mode qui favorise la qualité et celui qui favorise la fréquence d'image. La vidéo est en couleur et l'image est au format QCIF.

On note que le taux de compression obtenu avec le logiciel NV est plus faible que celui d'IVS et celui de VIC: celui de NV est environ 1.5 moins élevé que celui obtenu avec le codage H.261 et un quantificateur de 3. On parvient à encoder avec le même débit à un rythme environ 4 fois plus élevé avec le codage H.261 et un quantificateur de 10 qu'avec le codage NV. Ceci se fait bien sûr au détriment de la qualité de l'image, c'est-à-dire son SNR. Le mode qui favorise la fréquence d'image d'IVS utilise un quantificateur compris entre 3 et 13 selon le taux de compression à effectuer. Ce sont les valeurs obtenues avec un quantificateur de 13 qui sont montrées sur la courbe, figure C.1.

L'algorithme de codage H.261 de VIC est moins coûteux que celui d'IVS car il n'utilise pas le mode INTER. Le codeur se simplifie et on évite par exemple les opérations inverses de quantification, la TCD et la soustraction des pixels. VIC permet ainsi d'obtenir un encodage plus rapide sur les machines peu puissantes.

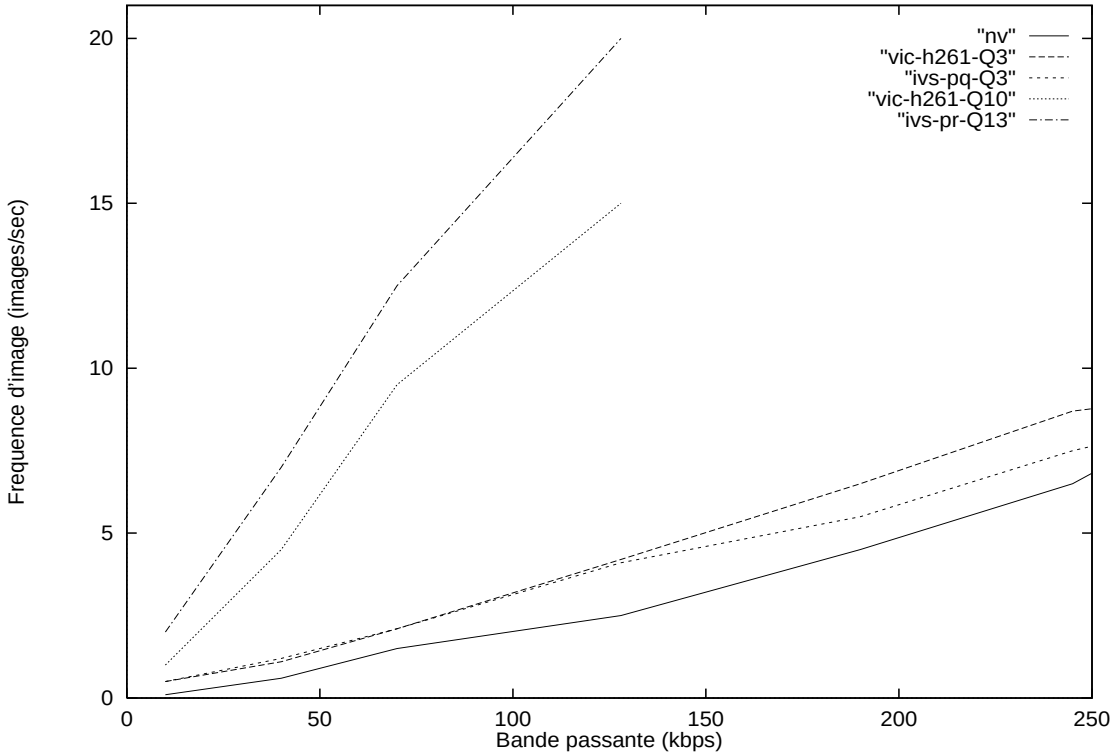


FIG. C.1 - *Comparaison des logiciels IVS, NV et VIC*

Annexe D

Glossaire

Les termes suivants, classés dans l'ordre alphabétique, sont utilisés dans le texte.

- **ADU:** une ADU (*Application Data Unit*) est la plus petite unité de donnée qu'une application peut traiter hors séquence.
- **application multimédia:** application échangeant des flots de données temps réel de nature différentes, (e.g. audio, vidéo, texte).
- **application temps réel:** application interactive échangeant des données qui ont de fortes contraintes temporelles, (e.g. faible délai de "bout en bout").
- **application traditionnelle:** application de données traditionnelles, c'est-à-dire de type transmission de fichiers, courrier électronique, terminal distant ...
- **BCH:** le code BCH (Bose, Chaudhuri Hocquengham) est un code de correction d'erreurs.
- **bloc:** plus petite couche hiérarchique du codage H.261; un bloc contient en particulier les coefficients de la TCD.
- **CCITT:** Comité Consultatif International Télégraphique et Téléphonique, aujourd'hui remplacé par l'Union Internationale des Télécommunications (UIT).
- **chrominance:** signal vidéo ne contenant que les informations de couleur.
- **CIF:** (*Common Intermediate Format*) est le Format Intermédiaire Commun défini par le CCITT. Il correspond a des images de 288 lignes de 352 pixels.

- **codage inter-image (INTER):** algorithme de codage vidéo qui encode uniquement la différence entre la nouvelle image et l'image précédente. Le codage inter-image est souvent complété d'un codage intra-image pour rafraîchir périodiquement l'image.

- **codage intra-image (INTRA):** algorithme de codage vidéo qui, à la différence du codage inter-image, ne tient pas compte des images précédentes.

- **codec:** abbréviation de codeur-décodeur.

- **conférence:** association entre les participants d'une communication temps réel multimédia. Une conférence est composée d'un contrôle de session, une association de médias et d'une politique de conférence.

- **flot de données:** flux unidirectionnel de données transmises en séquence.

- **GOB:** un GOB (*Group of blocks*) est constitué de 33 macroblocs dans une image au format H.261.

- **groupe:** ensemble des émetteurs et récepteurs identifiables grâce à une adresse multipoint qu'ils utilisent pour diffuser des informations.

- **IETF:** (*Internet Engineering Task Force*), forum où sont définis les standards de l'Internet (cf. [RFC1602]).

- **IVS:** (*Inria Videoconferencing System*) est le nom de baptême de notre logiciel de vidéoconférence. Il peut être récupéré par ftp anonyme via *ftp://zenon.inria.fr:rodeo/ivs/last_version.*

- **luminance:** signal vidéo ne contenant que les informations monochrome.

- **macrobloc:** un macrobloc (MB) est constitué de 6 blocs dans une image au format H.261.

- **MBONE:** réseau multipoint virtuel au dessus d'IP, voir annexe A.

- **MICE:** (*Multimedia Integrated Conferencing for European researchers*) est un projet Européen dont l'objectif est d'offrir aux chercheurs Européens des outils multimédia pour leur permettre de travailler ensemble et à distance depuis leur station de travail.

- **Miss America:** séquence vidéo souvent utilisée pour tester les applications de vidéoconférence. La séquence, au format CIF, représente une scène classique (*Head and shoulder*), c'est-à-dire la tête et les épaules d'une personne qui parle au premier plan et un décor fixe en arrière plan de l'image.

- **NDU:** un NDU (*Network Data Unit*) représente l'unité d'échange du réseau (e.g. un paquet IP ou une cellule ATM).

- **QCIF:** (*Quarter Common Intermediate Format*) est le Quart du Format Intermédiaire Commun défini par le CCITT. Il correspond a des images de 144 lignes de 176 pixels.

- **session:** association de membres pour le contrôle; par exemple, pour contrôler une conférence composée de plusieurs participants.

- **RTP:** (*Real Time Protocol*), protocole en cours de développement au sein du groupe de travail Audio Video Transport (AVT) de l'IETF.

- **SNR:** (*Signal to Noise Ratio*) est le rapport signal sur bruit, voir section 6.2.3.

- **TCD:** Transformée en Cosinus Discrète, ou *Dicrete Cosine Transform* (DCT).

- **UIT:** Union Internationale des Télécommunications, remplace le CCITT depuis le 1er mars 1993.

Bibliographie

[Ahmed74] N. Ahmed, T. Natarajan, K.R. Rao, "Discrete Cosine Transform", *IEEE Transactions on Computers*, January 1974.

[Ahuja88] S.R. Ahuja, J.R. Ensor, D.N. Horn, "The rapport multimedia conferencing system", *Proceedings of ACM conferencing on Office Information*, pp. 1-8, 1988.

[Antonini92] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, "Image coding using Wavelet Transform", *IEEE Trans. on Image Processing*, Vol. 1, No. 2, Apr. 1992.

[Aras94] C. Aras, J. Kurose, D. Reeves, H. Schulzrinne, "Real-time communication in packet-switched networks", *Proc. of the IEEE*, Jan. 1994.

[Aravind93] R. Aravind, G.L. Cash, D.L. Duttweller, H.M. Hang, B,G. Haskell, A. Puri, "Image and Video Coding Standards", *AT&T Technical Journal*, January-February. 1993, pp. 66-89.

[Banerjea94] A. Banerjea, E. Knightly, F. Templin, H. Zhang, "Experiments with the Tenet Real-Time Protocol Suite on the Sequoia 2000 Wide Area Network", *Technical Report TR-94-020*, International Computer Science Institute, Berkeley, CA, April 1994.

[Bolot93] J-C. Bolot, "End-to-end packet delay and loss behaviour in the Internet", *Proc. ACM/SIGCOMM'93*, Sep. 1993, pp. 289-298.

[Bolot94a] J-C. Bolot, T. Turletti, " A rate control for packet video in the Internet", *Proc. IEEE INFOCOM '94*, Toronto, Canada, pp. 1216-1223.

[Bolot94b] J-C. Bolot, T. Turletti, I. Wakeman, "Scalable feedback control for multicast video distribution in the Internet", *Proc. ACM/SIGCOMM'94*, Vol. 24, No 4, October 1994, pp. 58-67.

[Bourguignat93] E. Bourquignat, T. Alpert, "Caractérisation psychophysique de l'image de télévision", *L'écho des Recherches*, No 151, 1er trimestre 1993, pp. 43-50.

[Braden93] R. Braden, D. Clark, S. Shenker, "Integrated services in the Internet architecture", RFC 1633, June 1994.

[Burt83] P.J. Burt, E.H. Adelson, "The Laplacian pyramid as a compact image code", *IEEE Trans. Comm.*, Vol. 31, No 4, Apr. 1983.

[Casner92] S.Casner, S. Deering, "First IETF Internet Audiocast", *ACM Computer Communication Review*, Vol. 22, No 3, July 1992.

[CCIR601] "CCIR Recommendation 601-2: Encoding parameters of digital television for studios", *International Telecommunication Union*, 1990.

[Chan91] S. C. Chan and K. L. Ho, "A new two-dimensional fast cosine transform algorithm", *IEEE Transactions on Signal Processing*, vol 39, No 2, February 91, pp. 481-485.

[Chang92a] Y.H. Chang, J. Whaley, "Remote Conferencing Architecture", *MCNC Center for communications Research Note*, July 1992.

[Chang92b] W.T. Chang, N. Chang, D.G. Messerschmitt, "Call processing and signaling in a desktop multimedia conferencing system", *IEEE GLOBECOM '92*, Orlando, Florida, Dec. 1992, pp. 225-229.

[Chen90] Y. C. Chen, K. Sayood, D. J. Nelson, "A robust low-rate coding scheme for packet video", *Communication, Control and Signal Processing*, E. Arikan Elsevier Science Publishers B.V., 1990, pp. 1490-1498.

[Chiu89] D.M. Chiu, R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", *Computer Networks and ISDN Systems*, vol. 17, pp. 1-14, 1989.

[Clark90] D. D. Clark, D. L. tennenhouse, "Architectural considerations for a new generation of protocols", *Proc. ACM SIGCOMM '90*, Sep. 1990, Philadelphia, pp. 200-208.

[Crowcroft] J. Crowcroft, S. Hailes, M. Handley, A. Jena, D. Lewis, I. Wakeman, "Some Multimedia Traffic Characterisation", *IEEE ICT 93*, Manchester, England.

[Dabbous91] W. Dabbous, "Etude des protocoles de contrôle de transmission à haut débit pour les applications multimédias", *Thèse de Doctorat*, Université de Paris-Sud - Centre d'Orsay, mars 1991.

[Dabbous93] W. Dabbous, B. Kiss, "A Reliable Multicast Protocol (RMP)", *Rapport de recherche INRIA*, No 2100, Nov. 1993.

[Deering91] S. Deering, "Multicast routing in a datagram internetwork", *Phd Thesis*, Stanford University, California, U.S.A., 1991.

[DeCleene94] B. DeCleene, P. Pancha, M. E. Zarki, H. Sorensen, "Comparison of priority partition methods for VBR MPEG", *Proc. INFOCOM '94*, pp. 689-696.

[Delgrossi93] L. Delgrossi, R. G. Herrtwich, C. Vogt, L. C. Wolf, "Reservation protocols for internetworks: A comparison of ST-II and RSVP", *4th International Workshop on Network and Operating System Support for Digital Audio & Video*, Nov. 1993, pp. 199-207.

[Demers89] A. Demers, S. Keshav, S. Shenker, "Analysis and simulation of a fair queuing Algorithm", *Proc. ACM SIGCOMM'89*, Austin, Texas, Sep. 1989.

[Diot94a] C. Diot, "Reliability in multicast services and protocols: a survey", *In: International conference on local and metropolitan communication systems*, IFIP, Chapman and Hall, Kyoto, Dec. 1994.

[Diot95] C. Diot, C. Huitema, T. Turletti, "Network Conscious Applications", *soumis à HPCS'95*, Aou. 95, Mystic, Connecticut.

[Dorcey95] T. Dorcey, "The CU-SeeMe Desktop Videoconferencing Software", *ConneXions - The Interoperability Report*, Vol. 9, No 3, March 1995, pp. 42-45.

[Escobar92] J. Escobar, D. Deutsch, C. Partridge, "Flow synchronization protocol", *IEEE GLOBECOM '92*, Orlando, Florida, Dec. 1992, pp. 1381-1387.

[Ferrari93] D. Ferrari, "Distributed delay jitter control in packet-switching internetworks", *Journal of Internetworking: research and Experience*, 4, 1993, pp. 1-20.

[Floyd93a] S. Floyd, V. Jacobson, "The synchronization of periodic routing messages", *Proc. ACM SIGCOMM '93*, San Francisco, Aug. 1993, pp. 33-44.

[Floyd93b] S. Floyd, "Link-sharing and resource management models for packet networks", *unpublished memorandum*, Sep. 1993.

[Frederick94] R. Frederick, "Experiences with real-time software video compression", *Sixth International Workshop on Packet Video*, Portland, Oregon, Sep. 26-27, 1994, pp. F1.1-F1.4.

[Garcia85] N. Garcia, C. Munoz, A. Sanz, "Image compression based on hierarchical encoding", in *SPIE Image Coding*, Vol. 594, 1985, pp. 150-157.

[Garcia91] H. Garcia-Molina, A. Spauster, "Ordered and reliable multicast communication", *ACM Trans. on Computer Systems*, Vol. 9, No 3, Fev. 1991, pp. 242-271.

[Ghanbari89] M. Ghanbari, "Two-layer coding of video signals for VBR networks", *IEEE JSAC*, Vol. 7, No 5, Jun. 1989, pp.771-781.

[Ghanbari93] M. Ghanbari, C. J. Hugues, "Packing Coded Video Signals into ATM Cells", *IEEE/ACM Trans. on Networking*, Vol. 1, No 5, Oct. 1993, pp. 505-509.

[Gharavi91] H. Gharavi, "Multilayer subband-based video coding", *IEEE Trans. on Comm.*, Vol. 39, No 9, Sep. 1991, pp. 1288-1291.

[Gilge88] M. Gilge, "A high quality videophone coder using hierarchical motion estimation and structure coding of the prediction error", *Visual Communication and Image Processing*, (SPIE), Cambridge/Massachussets, Nov. 1988.

[Gilge91] M. Gilge, R. Gusella, "Motion video coding for packet-switching networks - An integrated approach", *Visual Communication and Image Processing III*, (SPIE), Boston, 10-13 Nov. 1991.

[Gonzales92] R.C. Gonzales, R.E. Woods, *Digital Image Processing*, Addison-Wesley, 1992.

[Guichard86a] J. Guichard, D. Nasse, "L'image numérique et le codage", *L'écho des Recherches*, No 126, 4e trimestre 1986, pp. 21-36.

[Guichard86b] J. Guichard, G. Eude, "Intra and Inter frame transform coding for moving pictures transmission", *Proc. ICC '86*, pp. 381-384.

[Guichard90] J. Guichard, G. Eude, "Visages", *L'écho des Recherches*, No 140, 2e trimestre 1990, pp. 3-12.

[Guichard91] J. Guichard, G. Eude, N. Texier, "State of the art in picture coding for low bitrate applications", *Proc. ICC '90*, pp. 120-125.

[Haendel94] R. Haendel, M.N. Huber, S. Schroeder, "ATM networks: concepts, protocols, applications", 2nd ed.,. Addison-Wesley, 1994.

[Haque85] M. A. Haque, "A two-dimensional fast cosine transform",*IEEE Transactions on Acoustic, Speech and Signal Processing*, Vol. ASSP-33, No 6, Dec. 1985, pp. 1532-1539.

[Handley93] M. J. Handley, P. T. Kirstein, M. A. Sasse, "Multimedia Integrated Conferencing for European researchers (MICE): piloting activities and the Conference Management and Multiplexing Centre", *Computer Networks and ISDN Systems*, No 26, 1993, pp. 275-290.

[Handley95] M. J. Handley, "The use of plain text keys for encryption of multimedia conferences", *Internal Report*, draft V1.3, Feb. 23th, 1995, UCL, London.

[Hellemans92] P. Hellemans, M. Mampaey, "A signaling protocol supporting multimedia and supplementary services", *IEEE GLOBECOM '92*, Orlando, Florida, Dec. 1992, pp. 1388-1394.

[Heybey92] A. T. Heybey, "Video coding and the application level framing protocol archtecture", *Master of Science Report at the MIT*, Jun. 1991.

[Hoffman93] D. Hoffman, M. Speer, G. Fernando, "Network Support for Dynamically Scaled Multimedia Data Streams", *4th workshop on NOSSDAV*, 1993.

[Hou87] H.S. Hou, "A fast recursive algorithm for computing the discrete cosine transform", *IEEE Transactions on Acoustic, Speech and Signal Processing*, Vol. ASSP-35, No 10, Oct. 1987.

[Huang93] H.C. Huang, J.H. Huang, J.L. Wu, "Real-time software-based video coder for multimedia communication systems", *Multimedia Systems 1993*, pp. 110-119.

[Huitema92] C. Huitema, T. Turletti, "Software codecs and work station video conferences", *Proc. of INET '92*, Kobe, Japan, pp. 501-508.

[Huitema94] C. Huitema, "Le routage dans l'Internet", *Edition Eyrolles*, 1994.

[H221] "Recommandation H.221 : Structure de trame pour canal à 64 kbps", *Recommandation de l'Union Internationale des Télécommunications (UIT)*, 1993.

[H261] "Recommandation H.261 : Codec vidéo pour services audiovisuels à $p * 64$ kbit/s" *Recommandation de l'Union Internationale des Télécommunications (UIT)*, 1993.

[IDMR] "idmr@cs.ucl.ac.uk mailing list", l'archive des messages échangés dans ce groupe de travail est disponible par ftp anonyme via cs.ucl.ac.uk:/darpa/idmr-archive.Z.

[Jacobson88] V. Jacobson, "Congestion avoidance and control", *Proc. ACM SIGCOMM '88*, Stanford, CA, pp. 314-329, Aug. 1988.

[Jacobson92] V. Jacobson, "vat - X11-based audio conferencing tool", *Unix Manual Pages*, Feb. 1993.

[Jacobson93] V. Jacobson, S. Floyd, "Class based queuing for policy based resource sharing", *Presentation IETF*, 1993.

[Jain89] A. K. Jain, "Fundamentals of Digital Image Processing", *Prentice-Hall*, 1989.

[Jayant93] N. Jayant, "High quality networking of audio-visual information", *IEEE Communications Magazine*, Sep. 1993, pp. 84-95.

[Jeffay92] K. Jeffay, D. L. Stone, T. Talley, F.D. Smith, "Adaptive, best-effort delivery of digiatl audio and video across packet-switched networks", *3rd Workshop on NOSSDAV*, San Diego, CA, Nov. 92, pp. 1-12.

[Kanakia93] H. Kanakia, P. P. Mishra, A. Reibman, "An adaptative congestion control scheme for real-time packet video transport", *Proc. ACM SIGCOMM '93*, San Francisco, Sep. 1993, pp. 20-31.

[Kim92] R.H. Kim, P. Joon-Seek, "A Fast Feature-Based Block Matching Algorithm Using Integral Projections", *IEEE JSAC*, Vol. 10, No 5, Jun. 1992, pp. 968-971.

[Kirstein93] P. T. Kirstein, M. J. Handley, M. A. Sasse, "Piloting of Multimedia Integrated Communications for European researchers (MICE)", *Proc. INET '93*, Aug. 1993, pp. DCA.1-DCA.12.

[Koga81] T. Koga, Y. Iijima, K. Iinuma, T. Ishiguro, "Statistical Performance Analysis of an Interframe Encoder for Broadcast Television Signals", *IEEE Trans. on Comm.*, Vol. 29, No. 12, Dec. 1981, pp. 1868-1876.

[Kunt84] M. Kunt, "Traitement Numérique des Signaux", *Traité d'Electrécité*, Vol. 20, Presses Polytechniques Romandes, 1984.

[Liou91] M. Liou, "Overview of the $p * 64$ kbit/s video coding standard", *Communication of the ACM*, No 4, Apr. 1991, pp. 60-63.

[Lynch85] T. J. Lynch, Data Compression : Techniques and Applications, *Lifetime Learning Publications*, Belmont, California, 1985.

[Macedonia94] M.R. Macedonia, D.P. Brutzman, "MBone provides audio and video across the internet", *IEEE COMPUTER magazine*, Apr. 1994, pp. 30-36.

[Mackie94] J. K. MacKie-Mason and H. Varian, "Pricing the Internet", *Public access to the Internet*, B. Kahin and J. Keller (Eds.), Prentice-Hall, New Jersey, à paraître.

[McCanne94] S. McCanne, V. Jacobson, "vic - video conference", *Unix Manual Pages*, Nov. 1994.

[Minoli79] D. Minoli, "Optimal packet length for packet voice communication", *IEEE Trans. on Com.*, vol. COM-27, pp. 607-611, Mar. 1979.

[Minoli94] D. Minoli, R. Keinath, "Distributed multimedia through broadband communications", *Artech House, Inc.*, Bellcore, 1994.

[Miran90] M. Miran, K. R. Rao, "Fast progressive reconstruction of images using the DCT", *Signal processing V: theories and applications*, 1990, pp. 897-900.

[Mitzel93] D. J. Mitzel, D. Estrin, S. Shenker, L. Zhang, "An architectural comparison of ST-II and RSVP", *Proc. IEEE INFOCOM '93*, Jun. 1993.

[MPEG] "Coding of moving pictures and associated audio (MPEG).", *International Organization for Standardization / International Electro-technical Commission Joint Technical Committee 1, Sub-committee 29, Work Group 11*, ISO/IEC JTC1 SC29.

[Nagle87] J. Nagle, "On packet switches with infinite storage", *IEEE Trans. on Comm.*, Apr. 1987.

[Ohta94] N. Ohta, "Packet video: Modeling and Signal Processing", *Artech House, Inc.*, 1994.

[Pagani93] D.S. Pagani, W.E. Mackay, "Bringing Media Spaces into the Real World", *Proceedings of ECSCW '93*, Milano, Sept. 13-17, 1993.

[Pancha92] P. Pancha, M. el Zarki "A look at the MPEG video coding standard for variable bit rate video transmission", *Proc. INFOCOM '92*, Florence, Italy, May 1992, pp. 85-94.

[Parekh93] A. Parekh, R. G. Gallage, "A generalized processor sharing approach to flow control in integrated services networks", *Proc. INFOCOM '93*, San Francisco, CA, Mar. 1993, pp. 521-530.

[Pasquale93] J.C. Pasquale, "Filter propagation in dissemination trees", *4th Workshop on NOSSDAV*, 1993.

[Pearl92] A. Pearl, "System Support for Integrated Desktop Video Conferencing", *Sun Microsystems Laboratories Inc. Internal Report*, SMLI TR-92-4, Dec. 1992.

[Petajan92] E. Petajan, "Digital video coding techniques for US high-definition TV", *IEEE MICRO*, Vol. 12, No 5, Oct. 1992, pp.13-21.

[Pingali94] S. Pingali, D. Towsley, J. F. Kurose, "A comparison of sender-initiated reliable multicast protocols", *Proc. IEEE GLOBECOM '94.*

[Puri89] A. Puri, R. Aravind, "An Interframe Coding Scheme for Packet Video", *SPIE Visual Communications and Image Processing IV*, AT&T Bell Laboratories, 1989, pp. 1610-1619.

[Kurose93] J. Kurose, "Open issues and challenges in providing QoS guarantees in high speed networks", *CCR*, Vol. 23, No 1, Jan. 1993, pp. 6-15.

[Ramanathan92] S.Ramanathan, V. Rangan, "Continuous media synchronization in distributed multimedia systems", *3rd Workshop on NOSSDAV*, San Diego, CA, Nov. 1992.

[Rao90] K. R. Rao, P. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications, *Academic Press Inc*, 1990.

[Reibman91] A. R. Reibman, "DCT-based embedded coding for packet video", *Signal processing: Image Communication III*, 1991, pp231-237.

[RFC768] J. Postel, "User Datagram Protocol (UDP)", *RFC 768*, Aug. 1980.

[RFC793] J. Postel, "Transmission Control Protocol (TCP) specification", *RFC 793*, Sep. 1981.

[RFC821] J. Postel, "Simple Mail Transfert Protocol (SMTP)", *RFC 821*, Aug. 1982.

[RFC854] J. Postel, "Telnet Protocol specification", *RFC 821*, May 1993.

[RFC959] J. Postel, "File Transfer Protocol (FTP)", *RFC 959*, Oct. 1985.

[RFC1602] C. Huitema, P. Gross, "The Internet standards process - Revision 2", *RFC 1602*, Mar. 1994

[RFC1112] S. Deering, "Host Extensions for IP Multicasting", *RFC 1112*, Aug. 1989.

[RFC1256] S. Deering, "ICMP Router Discovery Messages", *RFC 1256*, Sep. 1991.

[RFC1752] S. Bradner, A. Mankin, "The Recommendation for the IP Next Generation Protocol", *RFC 1752*, Jan. 1995.

[RM8] "Description Reference Model 8", *Specialists group on coding for visual telephony, COST 211-bis*, Paris, May 89.

[Sasse94] M.A. Sasse, U. Bilting, C-D Schulz, T. Turletti, "Remote Seminars through MultiMedia Conferencing: Experiences from the MICE project", *Proc. INET'94/JENC5*, Prague, June 1994, pp. 251/1-251/8.

[Schooler91] E.M. Schooler, S. L. Casner, J. Postel, "Multimedia Conferencing: Has it come of age?", *Proceedings 24th Hawai International Conference on System Sciences*, Vol. 3, Jan. 1991, pp. 707-716.

[Schooler93] E.M. Schooler, "Case Study: Multimedia Conference Control in a Packet-switched Teleconferencing System", *Internetworking: Research and Experience*, Vol. 4, 1993, pp. 99-120.

[Schulzrinne92] H. Schulzrinne, "Voice communication across the Internet: A Network Voice Terminal", *University of Massachussetts Technical Report*, June 1992.

[Schulzrinne93] H. Schulzrinne, S. Casner, "RTP: A transport protocol for real-time applications", *INTERNET-DRAFT*, Sep. 1993.

[Schulzrinne95] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A transport protocol for real-time applications", *INTERNET-DRAFT*, Mar. 13, 1995.

[Scotton93] P. Scotton, "Compression et transmission de signaux vidéo sur des réseaux haut débit avec contrôle de congestion du flux de données", *Thèse de Doctorat*, Université de Nice, Octobre 1993.

[Shenker90] S. Shenker, L. Zhang, D. Clark, "Some observations on the dynamics of a congestion control algorithm", *ACM CCR*, pp.30-39.

[Shenker94] S. Shenker, "Fundamental design issues for the future Internet", *Internal Report*, 1994.

[Stevens90] W. R. Stevens, "Unix network programming", *Prentice-Hall*, 1990.

[Sullivan92] G.J. Sullivan, R.L. Baker, "Rate-distortion Optimized Motion Compensation for Video Compression Using Fixed or Variable Size Blocks", *IEEE Globecom*, Dec. 1991, pp.85-90.

[Sunvideo93] "SunVideo 1.0 User's Guide", *Sun Microsystems, Inc.*, 1993, pp. 81-84.

[Tang92] J.C. Tang, "Why do users like video? Studies of multimedia-supported collaboration", *Sun Microsystems Laboratories Inc. Internal Report*, SMLI TR-92-5, Dec. 1992.

[Tawbi93] W. Tawbi, F. Horn, E. Horlait, J.B. Stefani, "Video Compression Standards and Quality of Service", *The Computer Journal*, Vol. 36, No. 1, 1993.

[Tenet94] The Tenet Group, "Recent and Current Research", University of California, Berkeley and International Computer Science Institute, *Internal Report*, Apr. 1994.

[Topolic90] C. Topolic, "Experimental Internet Stream Protocol, version 2 (ST-2)", *Request For Comment (RFC) 1190*, Oct. 1990.

[Turletti93a] T. Turletti, "A H.261 software codec for videoconferencing over the Internet", *INRIA Research Report*, No. 1834, January 1993.

[Turletti93b] T. Turletti, C. Huitema, "Videoconferencing in the Internet", *submitted to IEEE/ACM Transactions on Networking*, Dec. 1993.

[Turletti94a] T. Turletti, J-C. Bolot, "Issues with multicast video distribution in heterogeneous packet networks", *Sixth International Workshop on Packet Video*, Portland, Oregon, Sep. 26-27, 1994, pp. F3.1-F3.4.

[Turletti94b] T. Turletti, "The INRIA Videoconferencing System (IVS)", *ConneXions - The Interoperability Report*, Vol. 8, No 10, October 1994, pp. 20-24.

[Turletti95] T. Turletti, C. Huitema, "Packetization of H.261 video streams", *INTERNET-DRAFT*, Mar. 1995.

[Vanderdop91] L. Vanderdop, "Optimized quantization for image subband coding", *Signal Processing: Image Communication*, Vol. 4, No 1, Nov. 1991.

[Vetterli90] M. Vetterli, K. M. Uz, "Multiresolution techniques with application to HDTV", *4th International Colloquium on Advanced Television Systems*, Vol. 1, Ottawa, Jun. 1990.

[Wakeman92] I. Wakeman and J. Crowcroft, "A combined admission and congestion control scheme for variable bit rate video", *UCL Research Note*, RN/92/93, October 1992.

[Wakeman93] I. Wakeman, "Packetized Video - Options for interaction between the user, the network and the codec", *The Computer Journal (BCS)*, vol. 36, No. 1, Feb. 1993.

[Wallace91] G. K. Wallace, "The JPEG still picture compression standard", *Communications of the ACM*, Apr. 1991, Vol. 34, No 4, pp. 30.

[Wang88] L. wang, M. Goldberg, "Progressive image transmission by transform coefficient residual error quantization", *IEEE Trans. on Comm.*, Vol. 36, No 1, Jan. 1988, pp. 75-87.

[Wang91] Z. Wang, "Pruning the fast discrete cosine transform", *IEEE/ACM Trans. on Comm.*, Vol. 39, No 5, May 1991, pp. 640-643.

[Westerink88] P. H. Westerink, D. E. Boekee and J. W. Woods, "Subband coding of image using vector quantisation", *IEEE Transactions on Communications*, Jun. 1988, pp. 713-719.

[Wilson93] F. Wilson, I. Wakeman, W. Smith, "Quality Of Service parameters for commercial application of videotelephony", *Human Factors in Telecommunication Conference, HFT'93*, Darmstadt, Germany.

[Woods86] J.W. Woods, S.D. O'Neil, "Subband coding of images", *IEEE Trans. ASSP*, Vol. 34, No 5, Oct. 1986.

[Wu92] S.W. Wu, A. Gersho, "Improved decoder for transform coding with application to the JPEG baseline system", *IEEE/ACM Trans. on Comm.*, Vol. 40, No 2, Feb. 1992.

[XTP92] "XTP protocol definition - Revision 3.6", *Protocol Engines Incorporated*, Jan. 1992.

[Yavatkar93] R. Yavatkar, L. Manoj, "Optimistic strategies for large-scale dissemination of multimedia information", *Proc. ACM Multimedia '93*, Anaheim, CA, pp. 1-8, Aug. 1993.

[Zhang90] L. Zhang, "VirtualClock: A new traffic control algorithm for packet switching networks", *Proc. ACM SIGCOMM '90*, Sept. 1990, pp. 19-29.

[Zhang93] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource ReSerVation Protocol", *Proc. IEEE Network*, Sept. 1993.

[Zimmermann80] "OSI reference model - the ISO model of architecture for Open Systems Interconnection", *IEEE Trans. on Comm.*, Vol. 28, pp. 425-432, 1980.