SiriX

Le Système d'Images Réparti Intégré à X

Lutmann Patrice,
Castanet Richard,
Cousin Bernard,
LaBRI* (Laboratoire Bordelais de Recherche en Informatique),
351 cours de la Libération, 33405 Talence CEDEX,
FRANCE

I Introduction

Les années 1970, mais surtout les années 1980, ont vu la consécration et l'explosion de l'infographie. Les trois principaux domaines concernés sont la CAO/DAO, l'imagerie et la génération d'images de synthèse. Si la CAO/DAO est arrivée à maturité avec l'apparition de normes internationales telles que GKS, PHIGS et CGM, le monde de l'image est aujourd'hui en plein essor. Avec l'avènement du multimédia dans les années 1990 et 2000, viendra un ensemble de normes et de standards qui assiéront l'image au sein d'autres médias.

Nous décrivons dans cet article un système d'images numériques réparti intégré au système X (*X Window System, MIT* [MIT 85][Scheiffler 86]). Ce système, SiriX, basé sur les concepts du prototype ITAMEX [Lutmann 92], confère à X une nouvelle dimension multimédia: l'Image. Nous présentons les divers problèmes liés à un tel système, en particulier ceux du transfert et de la visualisation d'images, fixes ou animées. Nous proposons une architecture répartie de ce système en précisant les divers services et fonctionnalités apportés. Nous détaillons les différentes stratégies possibles d'implantation et d'intégration à X, et nous développons quelques techniques originales permettant de résoudre les difficultés dues au partage des ressources du système réparti (en particulier, la gestion des couleurs, du débit, du codage, etc).

II Images, Transfert, Visualisation

L'image est aujourd'hui un support privilégié de la communication. De plus en plus d'applications informatiques font appel aux images ou sont dédiées à la manipulation d'images. Pour ces applications, les principaux besoins sont l'archivage, le transfert, le traitement et la visualisation. A partir de l'analyse de ces besoins, nous définissons les principales caractéristiques du transfert et de la visualisation des images. Les difficultés à surmonter sont alors l'hétérogénéité des matériels et des images, le débit du réseau et les délais inhérents au transfert d'images numériques [Le Pannerer 88].

II.1 Les images

II.1.1 Définitions

Une image est une donnée abstraite et spatiale, représentation visuelle d'une chose, d'un objet, d'un concept, etc. D'un point de vue informatique, une image est un ensemble d'informations ayant une signification visuelle. On définit alors une collection d'images comme étant un ensemble d'images possédant les mêmes caractéristiques structurales, c'est-à-dire ayant les mêmes valeurs pour leurs attributs de représentation. On appelle séquence d'images une collection d'images telle que toute image sauf la première découle logiquement de la précédente. On appelle suite d'images une collection d'images qui n'est pas une séquence, à savoir que chaque image est indépendante et est décrite de manière absolue. On appelle enfin séquence animée d'images une séquence d'images liées sémantiquement et suivant des contraintes temporelles. La sémantique de l'image n'est autre que la donnée abstraite et subjective qu'est l'interprétation de cette image par l'utilisateur lors de sa visualisation.

On définit deux types d'images numériques: les images photographiques (*raster*) et les images graphiques ou de modélisation (*metafile*). Une image photographique est représentée par une matrice de points élémentaires définissant la plus petite surface homogène visualisable, appelée plus communément *pixel*. Une image graphique, comme par exemple une image de synthèse, est formée d'un ensemble d'objets graphiques géométriques, élémentaires ou complexes, possédant des attributs de rendu de visualisation.

II.1.2 Représentation et codage des images

La représentation d'une image numérique dépend de l'origine de cette image. En effet, le codage d'une image numérique est différent selon la provenance de l'image (dispositif numérique de saisie tel le *scanner*, ou application de génération de graphiques, telle une image de synthèse). Il y a ainsi une multitude de façons de représenter mais aussi de coder une image [Guichard 86] et l'on ne compte plus les standards de stockage d'images, de formats d'échanges d'images (TIFF, GIF, FTCRP, RLE, etc), et les techniques de compression adaptées à des transferts donnés (comme la visiophonie [Guichard 90]). La diversité des formats de codage se retrouve autant au niveau de la structuration des images (codage de l'ordre d'apparition des pixels dans l'image) que du codage du pixel lui-même (modèle de couleur, nombre de bits, etc), voire même une combinaison des deux. De plus, des informations hors image peuvent apparaître au sein même du format de codage de celle-ci.

On ne s'intéresse ici principalement aux images de type photographique.

Si l'on considère une image comme une matrice de points de couleur, le codage de l'image porte alors à la fois sur la structure de parcours des points de l'image et sur la structure d'un point de l'image. Le balayage de l'image va ainsi pouvoir se faire ligne par ligne, colonne par colonne, par bloc, suivant une courbe de Peano, un arbre... Le pixel, lui-même, peut être codé de diverses manières: soit en indiquant directement la couleur du point, soit

---

* UNITE DE RECHERCHE 1304 ASSOCIEE AU CNRS

indirectement en indiquant une entrée dans une palette de couleurs. Là se pose le problème du modèle de couleur utilisé: RVB, YC_bC_r, Huv, HLS, etc.

II.2 Le transfert d'images

II.2.1 Besoins

Lorsqu'on parle de transfert d'images, il s'agit la plupart du temps de véhiculer un gros volume d'informations en un laps de temps assez court. En effet, de part leur représentation bidimensionnelle, ou tridimensionnelle dans le cas des séquences animées, le stockage des images requiert des médias de grande capacité. Par exemple, si l'on considère une séquence animée dont chaque image est constituée de 1 Mpixels, chaque pixel étant codé sur un octet, cela fait 1 Mo de données par image. Pour un film de 90 minutes à 25 images par seconde, cela fait 135 000 images, soit près de 132 Go de données*.

Les besoins sont alors de pouvoir transférer efficacement les images de manière à accéder rapidement à l'information afin de manipuler ces images avec souplesse. Cela est nécessaire pour assurer des temps de réponse acceptable et permettre un service interactif. La réalisation de ces besoins passe par la fourniture d'un ensemble de services [Castanet 90][Navarro 91] et de protocoles [Bertin 90] adéquats.

II.2.2 Les moyens

Pour réaliser un transfert efficace, il est nécessaire de disposer d'un réseau fiable à débit suffisant, d'un protocole de transfert d'image qui soit rapide et efficace [Tawbi 91][Wolf 91] tout en étant approprié [Coudreuse 83] pour ne transférer que l'information utile tout en conservant la sémantique et en permettant la récupération d'erreurs. Ce service de transfert devra aussi tenir compte de la qualité de service demandé par l'utilisateur et devra respecter les contraintes temporelles que l'on rencontre dans le cas du transfert d'une séquence animée d'images (délais, dérive, etc). Ce service devra également prendre en compte la compression/décompression d'image lors du transfert.

Pour être plus convivial, le service de transfert permettra à l'utilisateur d'interagir avec le transfert, de pouvoir en sélectionner le mode et d'en modifier les paramètres au cours du temps. Il s'agira ici de dissocier le transfert de données proprement dit de la commande de ce transfert. Cela implique au niveau de l'application deux associations d'application, l'une de données, l'autre de commande.

II.3 Visualisation

II.3.1 Compréhension de l'image

La visualisation d'une image ne peut se faire que si l'on connaît la représentation de codage de l'image et du périphérique de visualisation. Pour l'utilisateur cela signifie reconnaître le format de l'image et pouvoir le décoder, savoir comment afficher un point d'une couleur donnée sur le périphérique de visualisation, et enfin effectuer la correspondance entre les deux.

II.3.2 Utilisation et partage des ressources

La visualisation d'une image nécessite d'adapter les caractéristiques et attributs de cette image à ceux du périphérique de visualisation. Les deux problèmes principaux sont alors de faire coïncider les modèles de couleur de l'image et du périphérique, et de faire tenir l'image sur le périphérique en conservant la sémantique et les proportions de l'image. Dans le cas d'affichage multiple d'images apparaît un nouveau problème qui est le partage des ressources disponibles au niveau du périphérique de visualisation (plus exactement du service d'image en charge de l'affichage sur le périphérique de visualisation). Il s'agit alors de gérer le chevauchement des images et la saturation de la palette de couleurs du périphérique.

II.3.3 Contraintes de temps

La visualisation de séquences animées d'images est assujettie au respect d'un certain nombre de contraintes temporelles: l'accès aux images, l'instant d'affichage et la durée d'affichage de chaque image. Il faut dans ce cas disposer d'un service d'horloge pour contrôler le temps et d'un service d'accès instantané à l'image. De plus, pour pouvoir réaliser un affichage à une cadence imposée, il est nécessaire de disposer d'un matériel et d'un système d'exploitation adéquat [Hanko 91][Northcutt 91]. Pour le service de visualisation, cela signifie posséder les données images à temps, pouvoir se synchroniser avec le service d'images en charge de l'affichage effectif et utiliser un service d'horloge. Associée au transfert, la visualisation d'images sous contraintes exige donc d'aller vite [Blum 90], en tout point de la chaîne.

II.3.4 Transfert et visualisation

Les diverses combinaisons de transfert et de visualisation se résument à:

- un transfert seul (c'est-à-dire un simple échange de données entre deux applications),
- une visualisation seule (l'image est déjà disponible sur le site de visualisation),
- un transfert immédiatement suivi d'une visualisation (avec un couplage ou non entre les deux actions).

Cela correspond à deux opérations: transférer pour visualiser ou transférer pour éditer puis visualiser. Il s'agit dans le premier cas d'une chaîne (il peut y avoir plusieurs sites acteurs du transfert, comme les opérateurs) de transfert de données (les images) dont le dernier site est le site de visualisation, tandis que dans le second cas, il y a une étape supplémentaire où le demandeur effectue une transformation locale sur l'image, auquel cas il y a d'abord un transfert simple d'images, puis un transfert pour visualisation.

---

* 141 557 760 000 OCTETS

II.3.4.1 Transférer pour visualiser

Les applications nécessitant un transfert d'images sont utilisées principalement pour sauvegarder ou visualiser des images. Dans le cas d'un transfert d'image où le seul besoin pour l'application est la visualisation de cette image, l'application n'a pas besoin de disposer effectivement des données de l'image. Seul l'élément de service qui aura en charge la visualisation de l'image en a réellement besoin. Ainsi, le service de transfert d'images et le service de visualisation peuvent être directement reliés et coopérer par un service optimum. Dans le cas d'une application répartie, ces deux services ne sont pas liés directement et peuvent se situer sur des sites physiquement distincts séparés du corps de l'application.

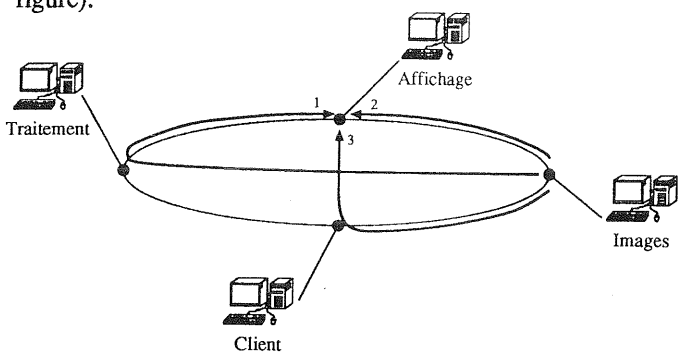
II.3.4.2 Transférer pour éditer puis visualiser

Dans le cas d'un transfert d'images avec traitement de l'image avant visualisation, ou même sans visualisation, l'application a directement besoin des données effectives de l'image. Au service du transfert et de visualisation s'ajoute alors un service de présentation des données directement lié à l'application. Dans le cas où le service de visualisation n'est pas sur le site de l'application, le service de transfert d'image doit alors jouer un double jeu. Il s'agit de transférer l'image depuis la source vers l'application demandeur d'image avec négociation des paramètres de profil, puis de transférer l'image depuis l'application (après traitement) vers le site de visualisation, avec renégociation optionnelle des paramètres de profil.

Il en est de même dans le cas d'un transfert pour visualisation où l'application a besoin des données effectives de l'image (par exemple pour archivage).

II.3.4.3 Flux de données

En fonction du type de transfert et de visualisation, il peut y avoir plusieurs configurations possibles entre les divers sites participant au transfert et à la visualisation (cf figure).



1: Transfert coopératif pour Visualisation

2: Transfert direct pour Visualisation

3: Edition avant Visualisation

Transfert et Visualisation d' Images: divers flux de données

Pour chacun de ces flux, il est nécessaire de faire correspondre la représentation de codage de l'image afin que chaque site soit en mesure de comprendre cette image. C'est le rôle de la négociation des paramètres de profil. Une image n'est ainsi transférée d'un site A vers un site B que si le site B peut restituer cett image (exception faite de l'archivage).

II.4 Environnement et objectifs de réalisation

La réalisation de SiriX suppose l'existence réelle ou virtuelle d'une base de données répartie d'images où l'ensemble des sites possède un système d'images en relation avec un service d'image et un système de visualisation en relation avec un service de visualisation. Le service d'image et le service de visualisation réalisent les opérations de base qui interviennent dans la chaîne de manipulation des images.

En fonction des besoins de l'application, les services offerts par SiriX permettent de déterminer les paramètres de profil optimum à utiliser pour minimiser le volume de données à transférer. Ils prennent en compte les services d'image existants pour réaliser l'adéquation de l'image avec les besoins de l'utilisateur (conversion de codage et traitement d'image). Ils présentent une interface générique homogène de visualisation permettant de faire abstraction des caractéristiques hétérogènes des divers matériels. Ils exploitent les ressources de chacun des acteurs participant à la réalisation du service en fonction des capacités relatives de ceux-ci. Ils gèrent les différentes contraintes temporelles nécessaires au transfert et à la visualisation acceptable de séquences animées d'images.

III Architecture du système

Nous définissons les multiples acteurs du service (site de stockage, de production, de visualisation, de traitement; émetteur, récepteur, utilisateur, demandeur, accepteur). Nous présentons les diverses associations permettant le contrôle du transfert, la commande du transfert et la visualisation. Nous analysons les flux de données et la localisation des traitements, et en particulier leurs influences sur la réalisation de SiriX, notamment ce qui concerne les contraintes d'implantation.

Nous détaillons le modèle en couches du système d'images réparti et nous montrons les différentes configurations possibles de l'architecture répartie de SiriX en fonction de la localisation des acteurs. Nous illustrons les différents comportements du système en fonction des besoins spécifiques de l'application.

III.1 Les acteurs du système

Le service de transfert et de visualisation d'image met en jeu au moins deux acteurs: le site demandeur d'image, le site détenteur d'image (un site de stockage ou de production), le site récepteur d'image. Un site est dit demandeur s'il est à l'initiative du service, il est dit accepteur sinon. Un site est dit émetteur s'il est origine de la transmission, récepteur sinon. Un site est dit opérateur si sur ce site est effectué un traitement d'image intervenant

dans le transfert de cette image. Le site de visualisation est le site où est affichée l'image. L'utilisateur est le bénéficiaire du service, qu'il soit demandeur ou accepteur. On appelle enfin *source* le maillon terminal de la chaîne de manipulation de l'image (ie le site qui fournit l'image, généralement le premier émetteur), et *puits* le maillon terminal qui réceptionne cette image (pratiquement, le site de visualisation).

III.2 Complexité et fonctionnalités

SiriX propose plus qu'un simple service de transfert [Degan 90] et de visualisation. Il permet un transfert coopératif d'images ou de séquences d'images, animées ou non, coordonné avec leur visualisation. Pour se faire, on exploite au maximum les fonctionnalités des services du système d'images sous-jacent. Les services proposés offrent à l'utilisateur la possibilité de transférer et de visualiser des images sans avoir à se préoccuper de la réalisation de toute conversion de codage et d'adaptation d'image préalable à l'affichage. Pour réaliser ceci, chacun des services est découpé en modules fonctionnels répartis entre les divers sites et interopérant pour fournir le service demandé par l'utilisateur. Ces modules sont regroupés logiquement et offrent les bases nécessaires au transfert, à la visualisation, au traitement et aux diverses commandes de transfert et de visualisation. On a ainsi plusieurs associations d'applications responsables d'une partie bien définie du service.

L'association de commande permet le contrôle et la gestion de la réalisation du service de transfert et de visualisation d'image. Elle est fonctionnellement découpée en deux sous-associations, l'une dédiée au transfert, l'autre dédiée à la visualisation, regroupées en une méta-association. Elles ont pour rôle les diverses identifications et initialisations des acteurs du service, leur mise en route, leur coopération, leur terminaison et leur abandon éventuel. Elles servent aussi à l'exécution d'un service interactif supportant un dialogue avec l'utilisateur.

L'association de transfert met en place et réalise le transfert effectif des données de l'image.

L'association de visualisation effectue l'affichage de l'image sur le périphérique de visualisation au travers du service d'image sous-jacent associé à ce dernier.

L'association de commande et l'association de transfert autorisent un transfert négocié entre deux sites avec la participation d'un ou de plusieurs sites intermédiaires (les opérateurs). Ces sites intermédiaires effectuent des opérations de traitement d'image ou de conversion de format réalisant l'adéquation de l'image entre les différents sites terminaux de la chaîne de transfert. Il s'agit dans ce cas d'un *pipeline* d'opérations effectuées sur l'image entre chaque couple de sites participant au transfert. La négociation des paramètres de profil à chaque étape du transfert permet de ne transférer que l'information utile. On a ainsi localement un transfert optimum d'image au sens du volume d'informations échangé. Cette propriété locale est globale si le transfert ne met en jeu aucun opérateur.

L'association de commande et l'association de visualisation servent au contrôle de l'affichage des images sur le périphérique de visualisation, ce dernier n'étant pas à priori limité qu'au seul écran *bitmap*. Elles prennent en charge les paramètres nécessaires à la visualisation et offrent les mécanismes indispensables à un rendu acceptable des images fixes et animées.

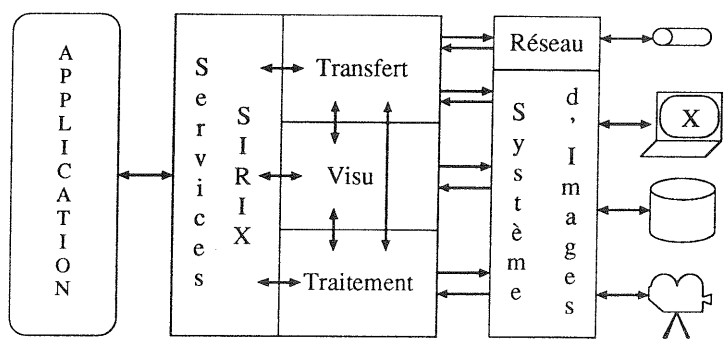
La méta-association de commande et les deux associations de transfert et de visualisation permettent un transfert et une visualisation synchronisés d'images en couplant la visualisation avec le transfert de celles-ci. Il s'agit ici de coordonner les actions entre les divers sites et les divers éléments de service pour fournir à temps les données images à destination du service de visualisation d'images sous-jacent. La visualisation d'une image est alors asservie à son transfert. Dans le cas d'une séquence animée obéissant à des contraintes temporelles précises, un service d'horloge est alors nécessaire pour respecter ces contraintes. Il faut dans ces conditions garantir d'une part la disponibilité des données (à la charge du service de transfert) et d'autre part le cadençage demandé par l'utilisateur (à la charge du service de transfert et du service de visualisation).

III.3 Structuration du système

La réalisation de SiriX se situe à deux niveaux: locale à un site, et répartie sur les sites intervenant dans l'exécution du service.

III.3.1 Structure locale de l'application

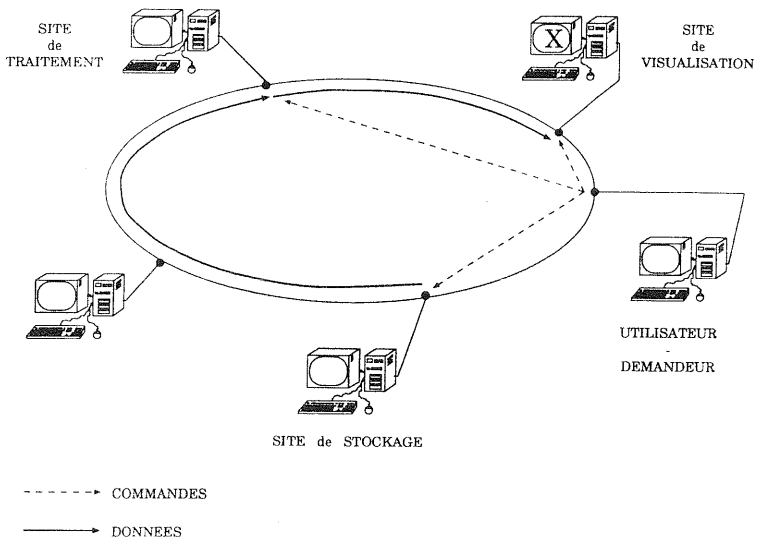
L'architecture de SiriX, s'intégrant dans le modèle OSI, est alors décrite conformément à la figure ci-dessous. L'utilisateur qui bâtit son application en utilisant SiriX dispose d'un ensemble de services de niveau application accessibles au travers d'une interface application.



La structure de l'application est une structure en couche découpée en unités fonctionnelles de service. Pour l'utilisateur, la couche accessible est la couche méta-commande (services SiriX) au travers de laquelle il peut utiliser et contrôler les diverses fonctionnalités de transfert, de visualisation et de traitement d'image. Cela est rendu possible grâce à une interface générique homogène permettant de faire abstraction des services sous-jacents offerts par le système d'exploitation ou d'autres applications.

III.3.2 Structure répartie de l'application

La réalisation de SiriX mettant en jeu tous les acteurs n'est autre qu'une application répartie de transfert et de visualisation d'image entre tous les sites y participant. Sur chacun des sites, seule une partie des fonctionnalités est nécessaire à l'exécution du service. Le modèle typique d'une application répartie utilisant SiriX est alors donné par la figure ci-dessous.



Dans l'exemple ci-dessus, l'application utilisateur pilote un transfert coopératif d'images couplé avec leur visualisation. Le transfert effectif des données est pris en charge par les divers sites accepteurs. Le site émetteur se charge de récupérer les images, de les envoyer à destination de l'opérateur qui, après traitement, les envoie au site de visualisation pour affichage. On voit bien dans cette configuration que tous les éléments du service ne sont pas indispensables sur chacun des sites.

Chaque élément de service d'application disponible sur ces divers sites peut être vu comme un sous-système client ou serveur suivant son rôle dans la réalisation du service, par analogie aux *daemons* UNIX.

IV Intégration à X

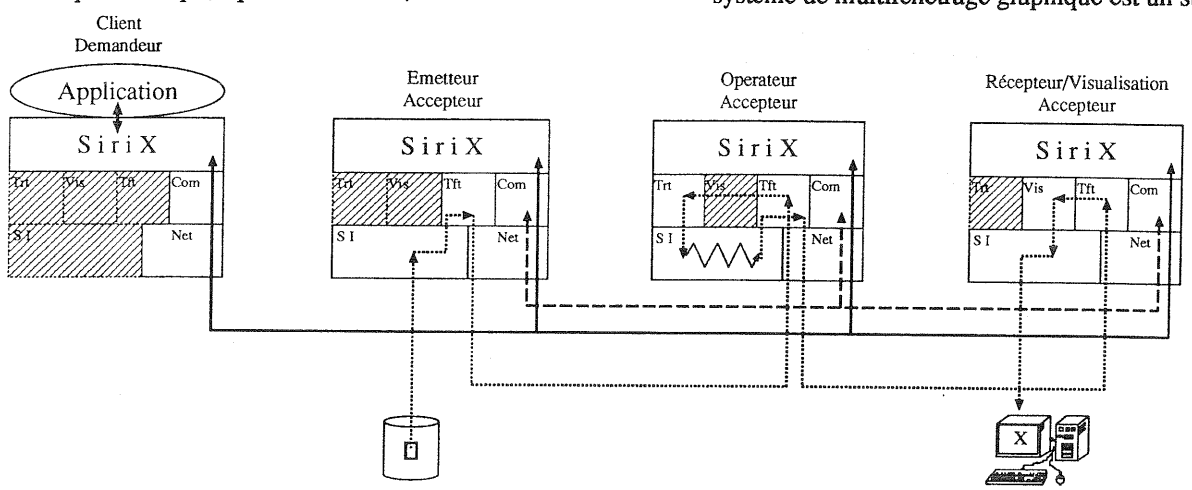
En tenant compte de l'impact de la philosophie de X sur SiriX, nous étudions les diverses possibilités d'implantation des services offerts par Sirix et leur intégration au système X.

Sans exposer en détail le coeur du système X *(core X, server and Xlib)*, les classes du protocole X, les mécanismes d'extensions et les concepts fondamentaux, nous précisons les diverses ressources manipulées par X et partagées par tous les clients qui lui sont rattachés. Nous évoquons également les problèmes liés à l'affichage d'images et au partage des couleurs entre les clients X.

Nous comparons enfin les divers modèles réalisés tant du point de vue des fonctionnalités que des performances.

IV.1 Choix du système X

Le choix de X comme base de SiriX en temps que service d'image de visualisation repose sur le fait que X est aujourd'hui l'environnement graphique le plus répandu. Ce système de multifenêtrage graphique est un standard de l'industrie qui offre la possibilité de bâtir des applications réparties possédant une interface utilisateur graphique portable. Son attrait réside aussi dans son architecture unique et indépendante du matériel. X permet ainsi à une application d'afficher des fenêtres contenant texte et graphiques sur tout type de matériel supportant le



Trt	Module de contrôle des Traitements
Vis	Module de contrôle de la Visualisation
Tft	Module de contrôle du Transfert
Com	Module de Communications et de Commandes
S I	Système d' Images
Net	Interface Réseau

- Commandes de Transfert et de Visualisation
- Commande locale de Transfert
- Flux des Données
- Traitement d' Images
- Module optionnel non utilisé

protocole X, et ce sans avoir à recompiler l'application. Cette double richesse offre la possibilité aux applications basées sur X de pouvoir fonctionner efficacement dans un environnement hétérogène regroupant de nombreux matériels graphiques informatiques équipés pour la visualisation d'images.

De plus, l'architecture désormais classique client/serveur du système X se prête aisément à la réalisation de SiriX et à la construction d'applications réparties.

IV.2 Modèles d'implantation

L'utilisation du système X comme service de visualisation d'images au sein du service de transfert et de visualisation d'images peut se faire de plusieurs manières [Lutmann 92]. L'intégration peut se faire extra-X ou intra-X, en réunissant les divers services d'application en un ou plusieurs modules d'applications (architecture monolithique "tout intégré" ou modulaire).

IV.2.1 Intégration intra-X

Dans ce modèle, les services de visualisation et de transfert d'image du site de visualisation sont directement intégrés à X. Dans ce cas, le serveur X a en charge à la fois le transfert et la visualisation des images, ce qui présente l'avantage de disposer immédiatement, au niveau du serveur, des données à afficher. Cela permet entre autre de supprimer un échange de données images supplémentaire entre un client X potentiel et le serveur, évitant ainsi une redondance inutile d'information. Néanmoins, cette manière de procéder va un peu à l'encontre de la philosophie de X car celui-ci se retrouve monopolisé par le client demandeur pendant le temps du transfert et de la visualisation. Cela complique aussi la gestion des ressources mémoire et logiques du serveur et impose une extension du protocole rendue nécessaire pour prendre en compte le transfert optimisé des données.

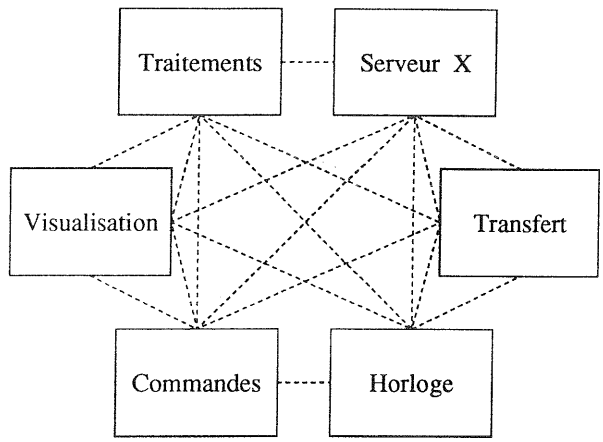
IV.2.2 Intégration extra-X

Dans ce modèle, les services de visualisation et de transfert d'images du site de visualisation sont indirectement intégrés à X. Seule une extension d'affichage et d'échange de données est nécessaire. Un processus indépendant du serveur prend en charge le transfert des données images. L'affichage de ces images est alors effectué par le serveur à la demande de ce processus, les données images étant partagées au travers d'une ressource commune (*shared memory*) pour des raisons évidentes d'efficacité. Cela permet d'éviter de surcharger de travail le serveur X pour la partie transfert. Malheureusement, on perd ainsi la possibilité d'afficher directement et à la volée l'image reçue sur l'écran. En contrepartie, la réalisation de SiriX est grandement simplifiée et ne nécessite que peu de modifications du système X.

IV.3 Choix d'une architecture d'implantation

La réalisation de SiriX impose l'implantation des divers services d'application qui sont (cf figure):

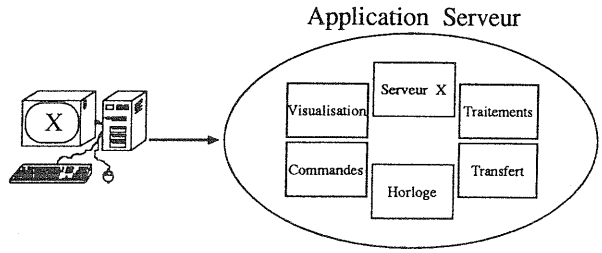
- un service de traitement pour les conversions de format d'image et de traitement d'image,
- un service de visualisation pour l'affichage de l'image,
- un service de transfert pour l'échange de données (ie les images),
- un service de commande et de communication pour coordonner et piloter les divers services,
- un service d'horloge pour gérer les contraintes temporelles et les attributs temporels de présentation,
- un service d'image de base, le système X.



Ces six services sont matérialisés au sein d'unités fonctionnelles ou modules d'application, qui sont les modules de base *Commandes*, *Transfert*, *Visualisation*, *Traitements*, *Horloge* et *Serveur X*.

IV.3.1 Configuration "tout intégré"

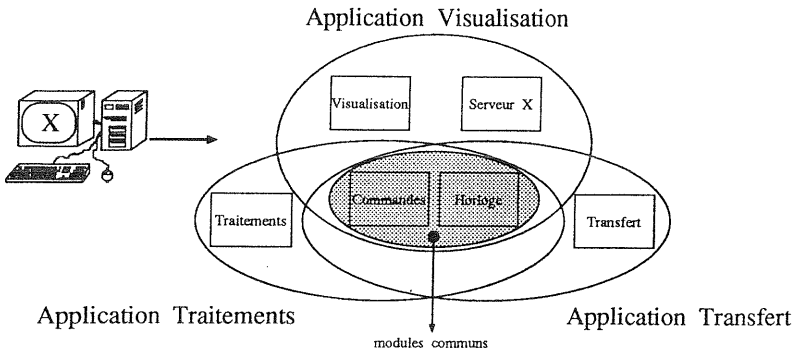
Les divers services présentés ci-dessus peuvent être intégrés en une seule application monobloc (cf figure ci-dessous) ou en une application modulaire comportant une partie commune (cf figure ci-contre).



Configuration "Tout intégré"

La configuration "tout intégré" regroupe en une seule application toutes les fonctionnalités de transfert et de visualisation. Si cette architecture offre l'avantage de pouvoir tout gérer à la fois, elle présente néanmoins de nombreux inconvénients. Outre la lourdeur de cette application et son aspect figé quant à ses possibilités d'évolution, cette configuration est trop complète car toutes

les fonctionnalités ne sont pas utiles en même temps. De plus, la cohabitation de tous ces modules réduit d'autant leur portabilité et leur mise au point.



Configuration "Tout intégré" Modulaire

La configuration "tout intégré" modulaire palie aux principaux défauts de la configuration précédente. L'application SiriX est découpée en trois applications coopérantes dédiées aux traitements, à la visualisation et au transfert. Les modules communs à ces trois applications sont l'*Horloge* et les *Commandes*. Ainsi, sur un site donné, seule une, deux ou trois de ces applications seront nécessaires à la réalisation des services demandés. Les deux modules communs servent ici à assurer la coordination de ces diverses applications et à la communication inter-processus. Le module *Commandes* a aussi en charge le dialogue avec l'utilisateur pour la réalisation d'un service interactif. Le module *Horloge* sert à la synchronisation des trois applications pour la gestion de l'affichage sous contraintes temporelles et le séquencement des opérations.

IV.3.2 Configuration "répartie"

Cette configuration correspond aux divers cas de figure que l'on rencontre au niveau d'une application répartie SiriX et qui se rapportent au rôle que peut jouer chacun des sites, à savoir la source, le puits, l'opérateur et le client (cf figure en bas de page).

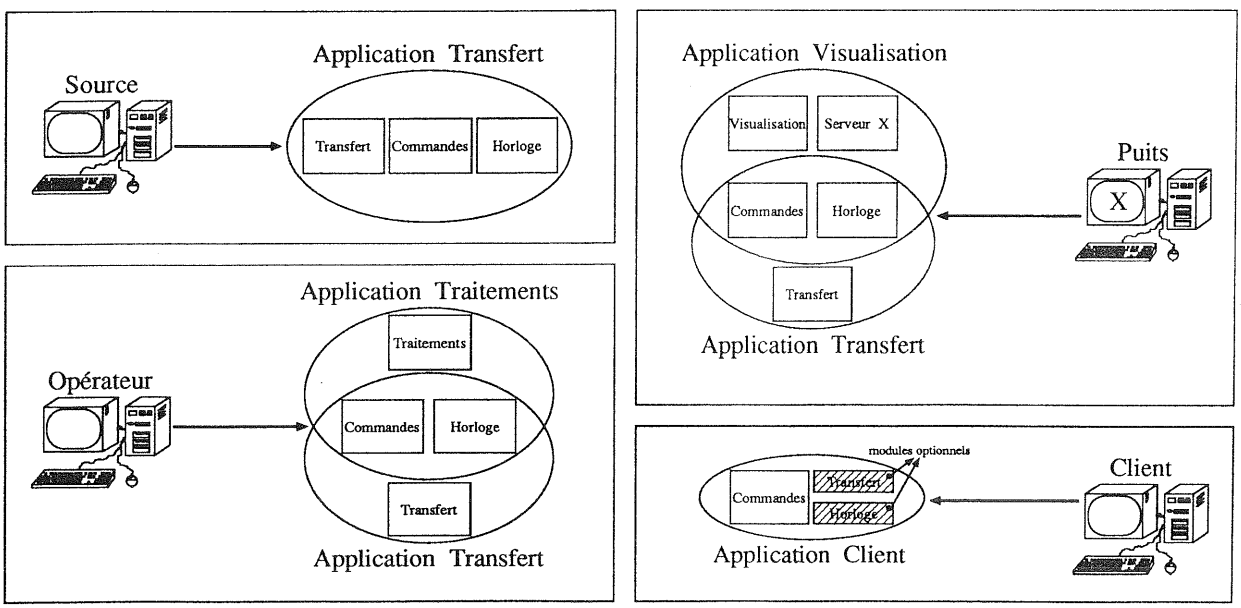
Ici on retrouve au niveau de chaque site le même découpage fonctionnel par module décrit dans la configuration "tout intégré", à la différence près que seuls les modules indispensables à un site sont présents sur celui-ci. Entre autre, le serveur X apparaît seulement sur le site de visualisation. Le besoin du système X, et de fait le protocole X, ne se manifeste qu'en bout de chaîne. Ainsi, les applications de transfert et de traitement sont indépendantes de X. Il en est de même pour un client ne faisant que du transfert.

Note: Les modules *Transfert* et *Horloge* de l'application client ne sont requis que si celle-ci désire disposer des données image en local (édition avant visualisation, *hardcopy*, stockage, etc).

IV.4 Contraintes d'implantation

La réalisation effective de SiriX pose plusieurs problèmes au niveau de l'implantation des divers services de transfert et de visualisation d'image. Les principales difficultés concernent les échanges de données entre les applications locales à un site et entre les applications réparties, le respect des contraintes temporelles et le partage des ressources graphiques du serveur X, spécialement la palette de couleurs du périphérique.

La résolution de ces problèmes passe par le choix d'un format d'images adapté à la fois au transfert et à la visualisation, par la définition d'un protocole de transfert de données efficace (au sens du nombre de messages échangés), par la mise en place d'un moyen rapide d'échange de données inter processus, par une stratégie d'allocation et de partage des couleurs et par un *handler* pour gérer les interruptions pseudo temps réel.



Configuration "Répartie"

IV.4.1 Le transfert des images

Le transfert des images, coopératif ou non, utilise le protocole TCP/IP disponible en standard sur les plateformes UNIX. Si les associations de commandes utilisent le mode connecté TCP (socket de type *STREAM*), le transfert de données proprement dit se fait en mode non connecté UDP (socket de type *DATAGRAM*). En effet, le protocole de transport fiable TCP est trop verbeux pour permettre une exploitation optimale de la bande passante offerte par le réseau sous-jacent (en particulier Ethernet) [Boggs 88][Joubert 92]. En dépit du fait qu'UDP ne garantit ni le séquencement ni l'acheminement des paquets, le protocole utilisé par SiriX réalise un transport fiable (*reliable data transfer* à-la-TCP) lorsqu'il s'agit de transférer une image fixe, sinon un transport avec une tolérance de perte pour les images animées. La perte d'information pour la visualisation d'une séquence animée d'images est acceptable car dans ce cas l'impact de l'erreur est relativement limité compte tenu des capacités physiologiques de l'oeil humain.

L'architecture d'un transfert bipoint est simplifiée à l'extrême (système *GUN*). Sur chaque site, deux processus sont impliqués dans le transfert, l'un gère le transfert des données en relation avec son pair, l'autre à en charge l'acquisition des données à transférer sur le site émetteur, et la restitution des données pour l'application sur le site récepteur. Ces deux processus distincts échangent leurs données via une mémoire partagée et se synchronisent par sémaphores. Un troisième processus virtuel, en *pipeline* avec les deux autres, est mis en place pour une éventuelle compression/décompression des données images.

IV.4.2 Le format des images

La gestion des couleurs disponibles sur le périphérique par le serveur X est généralement de type LUT (*LookUp Table*, opposé au type DCA, *Direct Color Access*). Le modèle de couleurs le plus simplement utilisé est le modèle additif RVB.

A l'instar du système *X-Movie* [Lamparter 91], on ne s'intéresse pour la visualisation qu'aux images RVB. Chaque pixel de l'image est alors décrit par un triplet (r,v,b) codé sous la forme d'une indirection dans une palette de couleurs associée à l'image. Une image fixe ainsi transmise est structurée en un entête descriptif, une palette de couleurs RVB, triée dans l'ordre décroissant d'occurrences du pixel correspondant dans l'image, et la matrice d'index correspondant aux pixels de l'image. A ces données s'ajoutent des informations spatio-temporelles de présentation relatives à la visualisation de l'image, particulièrement utilisées pour les animations.

IV.4.3 Partage des couleurs

Au niveau du serveur X, SiriX suppose l'existence d'une palette de 256 couleurs simultanées parmi N. L'affichage d'une image couleur pose alors deux problèmes: ajuster l'image dont la palette peut contenir plus de 256 entrées et tenir compte de l'environnement d'affichage, c'est-à-dire accorder sa palette avec celle déjà active au moment de l'affichage (pour éviter un désagréable effet "technicolor" sur l'écran X).

Pour satisfaire ses besoins, SiriX propose à l'utilisateur deux modes d'affichage. L'un, exclusif (*private mode*), alloue à l'utilisateur toutes les ressources de couleur, à savoir 256 entrées réellement disponibles. L'autre, partagé (*shared mode*), alloue virtuellement à l'utilisateur toutes les ressources de couleurs effectivement disponibles: une seule palette de couleurs est alors partagée par toutes les applications.

IV.4.3.1 Modes d'affichage et stratégie d'allocation

Pour limiter les interférences avec les autres applications X au niveau de la palette de couleurs installée par le serveur X, la stratégie employée par SiriX pour l'allocation des couleurs est la suivante:

- création d'une palette vierge en lecture/écriture (avec un flag d'extension propre à SiriX),
- recopie des entrées utilisées de la palette active (*installed colormap*) dans cette nouvelle palette,
- substitution de la palette active par celle-ci (l'*ID* de ressource est conservé, seule la gestion de la palette active et de l'allocation des entrées est changée),
- allocation des nouvelles entrées SiriX en partant de la fin (valeur décroissante des pixels).

Ainsi, les couleurs déjà utilisées par les clients existants sont toujours valides au moment où la palette SiriX est activée. Les clients X peuvent toujours allouer des entrées dans cette nouvelle palette, la valeur des pixels s'incrémentant. Suivant le mode d'affichage, les clients SiriX peuvent s'allouer des entrées prioritairement (*private mode*) ou non (*shared mode*) sur les clients X ordinaires. La présence d'un client SiriX commute automatiquement l'utilisation d'une palette SiriX, de manière transparente pour les autres clients X. Au niveau du serveur, il s'agit en fait d'une palette *cache*.

IV.4.3.2 Affichage d'images et gestion de couleurs

L'affichage d'une image fixe SiriX passe toujours par une palette SiriX (*S_Cmap*). Si la palette de couleurs associée à l'image comporte plus d'entrées (soit I_p ce nombre) qu'il n'y en a de réellement disponibles dans la *S_Cmap* (soit I_c ce nombre), les I_p-I_c couleurs seront approximées en fonction des I_c correspondantes déjà utilisées. Le tramage (*dithering*) n'est utilisé que si I_c est faible et que la taille de l'image résultante n'excède pas trop les dimensions de l'écran.

L'affichage de séquences animées d'images se fait à l'aide d'une palette de couleurs réduite à 127 entrées pour chaque image de la séquence. L'image affichée par rapport à l'image originale est celle obtenue après approximation des couleurs de l'image, mais sans tramage. La perte de qualité de rendu due à la réduction du nombre de couleurs employées n'est pas sensible à l'oeil humain (en fait, à peine plus d'une cinquantaine de nuances distinctes suffit). Les 127 entrées utilisées correspondent aux 127 dernières entrées disponibles dans la *S_Cmap*.

IV.4.4 Respect des contraintes temporelles

La visualisation de séquences animées pose plusieurs difficultés dont le débit du réseau de transport, les phénomènes de délais et de dérive, la puissance des machines et la disponibilité des ressources mémoire, la synchronisation des acteurs et un système de gestion du temps fiable [Bulterman 91].

Le cadre de réalisation de SiriX comprend un réseau Ethernet à 10 Mbits/s sous TCP/IP. Pour pouvoir effectivement exécuter une animation satisfaisante sous X [Chevalier 92] avec une machine de puissance raisonnable, la taille des images a volontairement été limitée afin de garantir la portabilité de SiriX sur des matériels généralistes sans extensions dédiées au multimédia. La dimension des images est restreinte à 64 Kpixels, ce qui, pour une cadence de 18 images/s, représente un volume d'environ 1.2 Mo de données (soit 9.5 Mbits). Pour véhiculer ceci à travers un réseau à 10 Mbits/s, on utilise une technique de compression de données élémentaire peu coûteuse en temps de calcul: la compression RLE (*Run Length Encoding*). D'autres techniques, comme JPEG, MPEG, ou plus simplement des techniques dérivées de codages différentiels, peuvent être utilisées en fonction de la puissance croissante des machines.

L'amortissement des phénomènes de dérive est rendu possible par une anticipation sur l'affichage. Plusieurs images sont ainsi mises en attente avant de démarrer l'animation. Le délai ainsi engendré est la résultante du délai intrinsèque du réseau sous-jacent et du temps de transfert de ces images. Cette bufférisation d'images permet d'absorber en partie les variations de taux de compression des images et donc des temps de transfert de celles-ci. Elle est aussi rendue nécessaire par le protocole de transfert de données SiriX pour la gestion des erreurs et des retransmissions [Joubert 92]. L'inconvénient de cette méthode est de devoir disposer de ressources mémoire suffisantes pour stocker les images compressées en attentes (quelques Mo).

L'application de visualisation est découpée en trois processus: le serveur X étendu, le processus de présentation de données et le processus en charge du transfert. Ce dernier réceptionne les données en provenance du site émetteur et les stocke dans une mémoire partagée. Le processus de présentation décompresse les images reçues dans cette même mémoire partagée et les met dans un format directement utilisable par X (*pixmap+colormap*). Le serveur X récupère ensuite les images dans la mémoire partagée et les affiche à la cadence voulue. L'affichage de chaque image est asservie à un *timer* temps réel (ou qu'on assimilera comme tel) et exécuté par une procédure de traitement d'interruption du serveur. Ce mode de fonctionnement permet au serveur X de ne pas être monopolisé par un seul client. Il peut ainsi continuer à traiter les requêtes des autres clients X.

V Réalisation et Utilisation

Sur la base de ce qui est réalisé, nous présentons les deux aspects du système Sirix: le côté utilisateur et le côté développeur. Nous montrons l'intérêt de ce système dans le cadre de l'utilisation d'images dans les réseaux locaux ou métropolitains, et dans la création de *PACS* (*Picture Archiving and Communication System*).

V.1 Le système SiriX

L'architecture modulaire et répartie de SiriX confère à un tel système souplesse, extensibilité et portabilité. Il ne se limite pas à l'affichage d'images sous X; il offre en plus un ensemble de services couvrant les besoins de transfert et de visualisation d'images. Pour l'utilisateur, SiriX est vu comme un support d'application permettant d'une part l'échange de données image entre applications, et d'autre part la visualisation d'images fixes ou animées. Pour le développeur, il s'agit d'un ensemble de services et de fonctionnalités accessibles via une interface application (*API*), doublée d'une architecture modulaire extensible autorisant l'ajout de nouvelles fonctionnalités.

V.2 Services et fonctionnalités offerts

Les services SiriX permettent le transfert d'images, la visualisation d'images et la combinaison coordonnée des deux, particulièrement pour les séquences animées d'images.

Pour le transfert, il s'agit de la mise en place de la communication entre les divers sites, de la gestion des ressources, du contrôle du transfert et le transfert effectif des données, de l'accès aux images, de la commande de l'équipement de production d'images, de la gestion de la qualité de transmission, de la possibilité de transfert sélectif, progressif, interactif, de la négociation de la représentation de transfert, des transformations d'images, de la synchronisation des sites et de la gestion des erreurs.

Pour la visualisation, il s'agit de la mise en place de la communication entre les divers sites, de la gestion des ressources, de la sélection et du contrôle du périphérique de visualisation (qui ne se limite pas qu'au seul écran X), de la gestion de l'affichage des images (qui ne se limite pas qu'aux seules images photographiques RVB), de la récupération d'images dans leur environnement de visualisation (*hardcopy*), de l'animation d'images et de la gestion de séquences d'images et de la qualité de rendu d'affichage. Néanmoins, il ne s'agit pas ici d'une extension vidéo du système X, celle-ci existant déjà [Brunhoff 90][Brunhoff 91] mais destinée à un usage différent.

Pour le transfert et la visualisation, il s'agit de la coordination des divers sites en vue de l'affichage des images transmises.

V.3 Utilisation et évolution

SiriX est principalement destiné au monde UNIX. Il offre un système réparti de transfert d'images comprenant la compression, le stockage, la restitution et les échanges de tous types d'images. Son architecture répartie découpée en modules fonctionnels le rend aisément portable et permet l'intégration sur un site des seuls services nécessaires. Le module de visualisation est conçu pour permettre

l'affichage d'images sur divers types d'équipements, matriciels ou vectoriels. Utilisé en tant que tel, il suffit à la partie communication et visualisation d'images d'un *PACS* [Binding 91][Gibaud 83].

L'évolution multimédia de SiriX viendra avec l'utilisation de MPEG comme standard de compression d'images et de sons. La dimension audio sera bientôt accessible avec l'apparition d'interfaces audio de qualité présentes sur les machines. L'utilisation de protocoles (comme XTP [Hienrichs 92]) et de réseaux hauts débits (comme ATM ou FDDI) augmentera sensiblement les capacités du système et permettra d'accroître l'interactivité des services. L'intégration dans un environnement multimédia avec des techniques propres au multimédia [Blair 90][Coulson 91][Miyamoto 90] permettra de tirer parti au mieux des nouveaux matériels et systèmes d'exploitation répartis de la génération multimédia.

VI Références et Bibliographie

* [ACSE] ISO 8649, 8650, Association Control Service Element
* [Bertin 90] C. Bertin, Y. Sureur, "Protocols for Multimedia Interactive Applications", 3rd IEEE Com. Int. Workshop on Multimedia Communications (Multimedia'90), Bordeaux, Novembre 1990
* [Binding 91] Binding C., "Demands on Network Technology from Medical Applications", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991
* [Blair 90] G.S. Blair, "Engineering Support for Multimedia Application in Open Distributed Processing", 3rd IEEE Com. Int. Workshop on Multimedia Communications (Multimedia'90), Bordeaux, Novembre 1990
* [Blum 90] C. Blum, "Fast Image Communication and Presentation in a Distributed LAN Environnement", 3rd IEEE Com. Int. Workshop on Multimedia Communications (Multimedia'90), Bordeaux, Novembre 1990
* [Boggs 88] Boggs David R., Mogul Jeffrey C., Kent Christopher A., "Measured Capacity of an Ethernet: Myths and Reality", ACM Comm. 1988, pp 222-234
* [Brunhoff 90] Brunhoff T., "VEX, Video Extension to X, Version 5.9", Tektronix Inc., 1990
* [Brunhoff 91] Brunhoff T., "MVEX, Minimal Video Extension to X, Version 6.2", Tektronix Inc., 1991
* [Bulterman 91] Bulterman D.C.A, van Liere R., "Multimedia Synchronisation and UNIX", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991
* [Castanet 90] R. Castanet, X. Navarro, "ITAM: un service de transmission d'images numériques", 1st Int. Conference Dedicated to Professional Image Chains (Image'Com 90), Bordeaux 1990
* [CCR] ISO 9804, Concurrency, Commitment and Recovery
* [Chevalier 92] Chevalier Roger, Bry Christophe, "Animation d'Images sous X", Rapport de projet, ENSERB, juin 1992
* [Coudreuse 83] Coudreuse J.P., "Les réseaux temporels asynchrones: du transfert de données à l'image animée", L'écho des RECHERCHES, No 112, 2ème trimestre 1983, pp 33-48
* [Coulson 91] Coulson G., Garcia F., Sheperd D., Hutchison D., "Protocol support for distributed multimedia applications", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991
* [Degan 90] N. Dal Degan, P. Migliorati, S. Pozzi, V. Trecordi, "Still Images Retrieval from a Remote Database", 1st Int. Conference Dedicated to Professional Image Chains (Image'Com 90), Bordeaux 1990
* [FDDI] ANSI X3.139, Fiber Distributed Data Interface
* [FTAM] ISO 8571, File Transfert Access and Management
* [Gibaud 83] Gibaud B., Scarabin J.M., De Certaines J., Coatrieux J.L., "Projet SIRENE: Serveur d'Imagerie médicale accessible via un REseau Numérique Expérimental", Rapport de projet, Université de Rennes I, juin 1983
* [Guichard 86] Guichard J., Nasse D., "L'image numérique et le codage", L'écho des RECHERCHES, No 126, 4ème trimestre 1986, pp 21-36
* [Guichard 90] Guichard J., Eude G., "Visages", L'écho des RECHERCHES, No 140, 2ème trimestre 1990, pp 3-12
* [Hanko 91] Hanko G., Northcutt J.D., Kuerner E.M., Wall G.A., "Workstation Support for Time-Critical Applications", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991
* [Hienrichs 92] Heinrichs B., Rupprecht M., "XTP - Efficient Parallel Software Implementation based on a Petri Net Specification Technique", Workshop on Broadband Communications, janvier 1992, pp 342-352
* [Hughes 90] L. Hughes, "A Hypermedia System With Educational Applications: The Edmund Hypermedia Research Project", 3rd IEEE Com. Int. Workshop on Multimedia Communications (Multimedia'90), Bordeaux, Novembre 1990
* [ICCCM] InterClient Communications Conventions Manual, Documentation X11 MIT
* [Joubert 92] Joubert Laurent, "Transfert d'images: une architecture efficace pour Ethernet/TCP-IP", Rapport de fin d'études, ENSERB, juin 1992
* [JPEG] ISO 10918, Joint Photographics Expert Group, Rev. 9.6, Janvier 1991
* [JTM] ISO 8831, 8832, Job, Transfert and Management
* [Lamparter 91] Lamparter B., Effelsberg W., "X-Movie: Transmission and Presentation of Digital Movies under X", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991
* [Le Pannerer 88] Le Pannerer Yves-Marie, "La transmission numérique des images", La Recherche, No 196, février 1988, vol 19, pp 174-181
* [Lutmann 92] Lutmann P., Cousin B., Castanet R., "ITAMEX: Image Transfert Access and Management Extension to X", Convention UNIX 92, mars 1992, pp 131-140
* [MIT 85] X Window System, Trademark of the Massachusetts Institute of Technology
* [Miyamoto 90] T. Miyamoto, "Multimedia Server Architecture for Broadband Networks", 3rd IEEE Com. Int. Workshop on Multimedia Communications (Multimedia'90), Bordeaux, Novembre 1990
* [Navarro 91] X. Navarro, "Un service d'application de transfert d'images", Thèse, Université de Bordeaux, Mars 1991
* [Northcutt 91] Northcutt J.D., Kuerner E.M., "System Support for Time-Critical Applications", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991
* [OSI] ISO 7498, Open Systems Interconnection (OSI) Reference Model
* [ROS] ISO 9072, Remote Operation Service
* [Scheiffler 86] R.W. Scheiffler, J. Gettys, "The X Window System", ACM Trans. on Graphics, Vol 5, No 2, Avril 1986, pp 79-109
* [Tawbi 91] Tawbi W., Horlait E., Dupuy S., "High Speed Protocols: State of Art in Multimedia Applications", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991
* [TP] ISO 2093, 2094, Transaction Processing
* [VT] ISO 9040, 9041, Virtual Terminal
* [Wolf 91] Wolf L.C., "A Runtime Environment for Multimedia Communication", 2nd Int. Workshop on Network and OS Support for Digital Audio and Video, novembre 1991