UNIVERSITE DE PARIS-SUD

CENTRE D’ORSAY

THESE

présentée pour obtenir le Grade de

DOCTEUR de l’Université PARIS XI

par

Walid DABBOUS

SUJET

ETUDE DES PROTOCOLES DE CONTROLE DE TRANSMISSION A HAUT DEBIT POUR LES APPLICATIONS MULTIMEDIAS

soutenue le 29 Mars 1991 devant la Commission d’examen composée de:

MM. M. DIAZ Président
R. CASTANET Rapporteur
J. BEAUQUIER
C. HUITEMA Examinateurs
F. BACCELLI Directeur
A mon père
Je remercie Monsieur Michel Diaz, Directeur de recherche au CNRS, de m’avoir fait l’honneur de présider le jury de cette thèse, et d’avoir accepté d’en être rapporteur.

Je remercie vivement Monsieur Richard Castanet, Professeur à l’ENSERB, pour avoir accepté d’être le rapporteur de cette thèse.

Je tiens à remercier Monsieur Joffroy Beauquier, Professeur à l’Université de Paris-Sud, d’avoir bien voulu participer à ce jury.

Que Monsieur Christian Huitema, Directeur de recherche à l’INRIA, pour l’intérêt continu avec lequel il a suivi la progression de cette thèse et pour son aide précieuse, reçoive le témoignage de ma sincère reconnaissance.

Je tiens également à remercier Monsieur François Baccelli, Directeur de recherche à l’INRIA, pour le soutien constant qu’il m’a toujours accordé.

Enfin, je remercie chaleureusement les membres de l’équipe du projet RODEO de l’INRIA pour leur amitié et leur aide.
Résumé

Les réseaux de communication de données ont connu une évolution rapide durant la décennie précédente. Les avancées ont porté sur plusieurs domaines de la technologie et ont abouti au développement des réseaux à haut débit ayant des vitesses de transmission pouvant dépasser les 100 Mbit/s tels que FDDI, DQDB, et le RNIS large bande (B-ISDN).

D’autre part, des applications nécessitant des hauts débits de transmission sont en cours de développement. Ces nouvelles applications supportent une intégration de plusieurs types de trafic ayant des caractéristiques différentes, comme par exemple la transmission simultanée de données, de voix numérisée ou d’images vidéo. L’implantation de telles applications et d’applications plus “classiques” comme le transfert de fichier ou la messagerie électronique sur des réseaux à haut débit pose un problème délicat à cause des besoins différents de ces applications. Le problème a plusieurs aspects selon le niveau considéré.

En ce qui concerne les couches les plus basses c’est à dire la transmission et la commutation il s’agit de réaliser des supports de transmission ayant des bandes passantes très large et de construire des commutateurs permettant de fonctionner à des débits de plusieurs millions paquets par seconde. La technique de multiplexage asynchrone ATM (Asynchronous Transfer Mode) a été recommandée par le CCITT pour le RNIS à large bande. Cette technique permettra de multiplexer plusieurs types de trafic ayant des statistiques variables dans le temps sur le même support de transmission.

Pour le contrôle de transmission il s’agit d’analyser les différents besoins des applications et de trouver un système de transport qui puisse répondre à ces besoins en propagant les augmentations des débits jusqu’aux applications. En effet, les protocoles de contrôle de transmission standards comme TP4 ou TCP ont été conçus pour assurer la transmission fiable de données sur des réseaux à faible débit en effectuant la correction des erreurs de transmission et le contrôle des flots de données. Pour cela des mécanismes complexes basés sur l’utilisation de temporisations et sur des retransmissions ont été adoptés. Ces mécanismes ralentissent la transmission sur les réseaux à haut débit actuels par des contrôles qui ne sont plus toujours indispensables à cause des progrès technologiques qui ont permis d’avoir des taux très faibles pour les erreurs de transmission. D’autre part, le développement des applications nouvelles qui supportent la transmission simultanée de flots de données ayant des caractéristiques différentes nécessitent un mécanisme efficace pour le contrôle de l’échange et la synchronisation de ces flots.

Au niveau application, le problème général de synchronisation de l’application et de gestion de la transparence des données dans un réseau de machines hétérogènes devient encore plus délicat pour les applications haut débit. Il s’agit dans ce cadre
de mettre au point des techniques qui permettraient de décrire la dynamique d’une application et la structure des données échangées. D’autre part, il faudra définir une syntaxe de transfert allégée permettant l’implémentation du protocole de présentation comme un filtre entre l’application et le transport.

Dans cette thèse, nous nous intéressons essentiellement aux protocoles de contrôle de transmission dans les réseaux à haut débit. Les travaux de recherche antérieurs sur ce problème suivaient l’une des deux lignes suivantes:

- Rechercher les points de blocage dans les protocoles classiques et introduire les améliorations nécessaires pour adapter ces protocoles aux débits élevés. Selon cette approche, les choix adéquats de l’architecture matérielle et de l’implémentation et le réglage des valeurs de paramètres permettent d’optimiser les performances des protocoles tout en conservant la fonctionnalité que fournissent ces protocoles standards.

- Concevoir de nouveaux protocoles pouvant répondre aux besoins d’une application particulière en tirant profit des avancées technologiques. La simplification du service transport pour répondre aux besoins d’une application particulière permet d’alléger les primitives de contrôle de transmission voire d’implémenter le protocole dans un chip VLSI.

Dans cette thèse, nous avons abordé le problème selon un axe intermédiaire : la conception d’un protocole compatible avec les normes OSI, intégrant les optimisations proposées pour les protocoles classiques et fournissant un nouveau service transport aux applications pour inclure le transfert de données “temps réel” i.e. nécessitant un minimum de contrôle au niveau de la couche transport. Pour cela nous sommes partis du service de transport standard de l’OSI et du protocole TP4 en proposant des modifications aux afin d’aboutir à un service de transport “typé” et un protocole de transport multimédia que nous avons appelé TP5. Ce protocole propose un moyen pour faciliter la synchronisation des différents flots de données au niveau transport. Le travail effectué fait l’objet de cette thèse.

Dans le chapitre 1 nous présentons les nouveaux changements dans les réseaux à haut débit qui ont motivé une modification des protocoles de contrôle de transmission. Les avancées technologiques qui ont offert des débits très élevés ainsi que la demande des utilisateurs vont aboutir à une prolifération des réseaux à haut débit tels que FDDI, DQDB et le RNIS à large bande. Ce chapitre examine l’architecture et les fonctionnalités de ces nouveaux réseaux afin de pouvoir ultérieurement définir les optimisations possibles des protocoles des couches intermédiaires (contrôle de transmission).

Les caractéristiques des applications multimédia et les principes de conception du service et du protocole de transport multimédia font l’objet du chapitre 2. En effet,
avec l'émergence des futurs réseaux haut débit de nouvelles classes d'applications pourront être développées. Ce chapitre est donc d'abord consacré à la spécification des besoins de ces applications et par suite de décrire les mécanismes appropriés qui vont permettre le développement d'un système fournissant les services requis.

Le chapitre 3 s'intéresse à l'étude des différents mécanismes d'optimisation de la couche transport. Nous localisons d'abord les origines des limitations des performances dont trois principales : La première ne concerne pas le protocole de contrôle de transmission mais son environnement d'implémentation i.e. l'architecture du matériel et le système d'exploitation. La puissance du processeur et la façon de gérer les interruptions et les commutations de tâches (context switching) sont des facteurs déterminants pour les performances. La deuxième concerne les choix d'implémentation et le réglage des paramètres du protocole ont aussi un impact important sur l'efficacité. La troisième provient des fonctionnalités trop complexes des protocoles “à usage général”. La solution généralement proposée à ce dernier problème consiste à concevoir des protocoles de “coûts légers” et à fonctionnalité réduite (comme XTP, VMTP ou NETBLT) afin de répondre aux besoins d’une classe d’applications.

Les analyses des chapitres précédents nous amènent à la définition d’un nouveau service transport “typé” pour les applications multimédias. Les primitives du contrôle de la transmission sont appliquées selon les caractéristiques du flot sur lequel les données sont transmises. Ce service est décrit dans le chapitre 4 où nous présentons aussi un protocole dérivé de TP4 de l’ISO qui fournit aux applications multimédias le service typé. Les points clés de la conception de ce protocole sont la gestion de la connexion et la synchronisation des différents flots de données au niveau transport. Un nouveau type de données : RTD ou données temps réel est défini et sera multiplexé avec les données standards (DT) sur la même connexion transport. Les données DT seront soumises aux contrôles de flux et d’erreurs classiques du service de transport standard alors que seul un contrôle partiel sera appliqué aux données RTD. Le but de ce contrôle est de faciliter la synchronisation des deux flots DT et RTD est alors assurée par des règles de numérotations mutuelles et de préé- dence. Deux modes de synchronisation sont alors définis : le mode séquence relative fournit à l’application le moyen de commander l’envoi de demande de synchronisation des deux flots DT et RTD. Le deuxième mode la séquence absolue décharge l’application de cette fonctionnalité en supposant que chaque unité de données DT correspond à une demande de synchronisation. Ceci revient alors à fournir le service de synchronisation complètement au niveau de la couche transport.

L’évaluation du coût de la synchronisation selon les deux modes proposés fait l’objet du chapitre 5. Il est montré en utilisant les techniques de la théories des files d’attente que le coût de la synchronisation est inférieur à celui du cas d’un service de transport standard. En effet, même si la synchronisation avec un service typé affecte la
transmission du flot temps réel en provoquant un certain pourcentage de perte de paquets dus au délai de synchronisation, les calculs montrent que la détérioration de la qualité de service globale de ce flot en ce qui concerne les délais est plus prononcée dans le cas d’un service de transport standard.

L’annexe A contient un glossaire de quelques termes figurant dans le texte. L’annexe B contient la description des formats de paquets de données. L’annexe C décrit les techniques d’implémentation du protocole TP5 dans le noyau UNIX.
# Table des matières

1 Introduction 1

2 Les réseaux à haut débit 5
   2.1 Les avancées technologiques 5
       2.1.1 Transmission 6
       2.1.2 Commutation 7
       2.1.3 ATM 8
   2.2 Les réseaux à haut débit 10
       2.2.1 FDDI 10
       2.2.2 DQDB 12
       2.2.3 Le RNIS 13
   2.3 Structure des réseaux haut débit 14
   2.4 Contrôle de transmission 14
   2.5 Développement des applications haut débit 18

3 Applications et Services de transport 21
   3.1 Les applications haut débit 21
   3.2 les flots de données 22
   3.3 Les paramètres de qualité de service 23
   3.4 Le service de transport 25
   3.5 Les besoins des applications haut débit 26
   3.6 Quel système de transport? 28

4 Le contrôle de transmission à haut débit 29
   4.1 Impact de l’environnement 30


4.1.1 Les architectures multi-processeurs .................................. 30
4.1.2 Le système d’exploitation .............................................. 32
4.1.3 Les frontaux ou Implémentation en hardware ....................... 35
4.1.4 Interfaces réseaux haut débit ....................................... 36
4.1.5 Interface Protocole-Système ......................................... 39
4.1.6 Environnement haut débit ............................................ 39
4.2 Réglage de protocoles standards ....................................... 39
4.2.1 Le contrôle d’erreurs ............................................... 40
4.2.2 Le contrôle de flux .................................................. 44
4.3 Protocoles à coût léger .................................................. 47
4.3.1 XTP ................................................................. 49
4.3.2 Autres travaux ....................................................... 55
4.4 Conclusion .................................................................. 56

5 Contrôle des applications multimédias .................................. 57
  5.1 Le service de transport multimédia ................................... 57
  5.2 La synchronisation des flots de données ............................ 59
    5.2.1 Présentation de données multimédias .......................... 60
    5.2.2 La synchronisation session .................................... 62
    5.2.3 Le multiplexage au niveau liaison ............................ 63
    5.2.4 La synchronisation transport ................................. 63
  5.3 Le protocole TPM ...................................................... 65
    5.3.1 Le service de transport typé .................................. 66
    5.3.2 La synchronisation de flots multiples ........................ 68
    5.3.3 Les modes de synchronisation d’une connexion multimédia . 70
  5.4 Conclusion ............................................................. 76

6 Évaluation des modes de synchronisation ............................. 79
  6.1 Modèle du service transport ......................................... 79
    6.1.1 Calcul du délai avec le service standard .................. 80
    6.1.2 Calcul du délai avec le service temps réel ................ 81
### TABLE DES MATIÈRES

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2 Le coût de la synchronisation</td>
<td>83</td>
</tr>
<tr>
<td>6.2.1 Calcul pour un service standard</td>
<td>88</td>
</tr>
<tr>
<td>6.2.2 Calcul pour le service typé</td>
<td>89</td>
</tr>
<tr>
<td>6.3 Analyse des résultats</td>
<td>89</td>
</tr>
<tr>
<td>6.3.1 Service de transport standard</td>
<td>89</td>
</tr>
<tr>
<td>6.3.2 Service de transport typé</td>
<td>91</td>
</tr>
<tr>
<td>6.4 Conclusion</td>
<td>91</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>93</td>
</tr>
<tr>
<td>A Glossaire</td>
<td>111</td>
</tr>
<tr>
<td>B Format de TPDUs</td>
<td>113</td>
</tr>
<tr>
<td>C Techniques d’implémentation</td>
<td>115</td>
</tr>
<tr>
<td>C.1 Environnement</td>
<td>115</td>
</tr>
<tr>
<td>C.2 Méthode adoptée</td>
<td>116</td>
</tr>
</tbody>
</table>
TABLE DES MATIÈRES
Liste des Figures

2.1 La cellule ATM ........................................ 9
2.2 Le “MAN” DQDB ...................................... 12
2.3 Interconnexion de réseaux à haut débit ..................... 15
2.4 Architecture “Switched Multimegabit Digital Services” ........ 16
2.5 L’architecture en couches: où optimiser? .................. 17
3.1 Le service de transport OSI ................................ 25
4.1 Attente pour le “scheduling” d’un processus ............... 33
4.2 Diagramme bloc de l’interface réseau haut débit .......... 38
4.3 Environnement d’implémentation de protocoles à haut débit. .. 40
4.4 Calcul adaptatif du délai de transmission .................. 46
4.5 L’architecture de la Machine de Protocole ................. 52
5.1 Multiplexage des flots au niveau de la couche N = 7, 5, 4, ou 3 ... 61
5.2 Transmission des flots de données sans multiplexage. .......... 61
5.3 Le service de transport “typé” ................................ 65
5.4 Les primitives du transport typé .............................. 67
5.5 Transfert des données DT et RTD selon le mode séquence relative . 71
5.6 Perte éventuelle de données RTD dans le mode séquence relative .. 74
5.7 Transfert des données DT et RTD selon le mode séquence absolue . 75
6.1 Le modèle des services de transport standard et typé ......... 80
6.2 Délai moyen $E(Y_{a_d})$ (en $\mu s$) des messages de données pour un service standard en fonction de $\lambda_d$, avec $\lambda_r = 100\lambda_d$, $\tau = 10\mu s$, $T = 100$ms. Les trois courbes de haut en bas correspondent à $\epsilon = 10^{-5}$, $10^{-6}$ et $10^{-7}$ respectivement. ......................... 82
6.3 Ecart type du délai des messages de données $\sigma(Y_{dd})$ pour un service standard en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu s$, $T = 100ms$. Les trois courbes de haut en bas correspondent à $\epsilon=10^{-5}$, $10^{-6}$ et $10^{-7}$ respectivement. ........................................ 83

6.4 Délai moyen $E(Y_{dd})$ (en $\mu s$) des messages de données temps réel pour un service standard en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu s$, $T = 100ms$. Les trois courbes de haut en bas correspondent à $\epsilon=10^{-5}$, $10^{-6}$ et $10^{-7}$ respectivement. ........................................ 84

6.5 Ecart type du délai des messages de données temps réel $\sigma(Y_{dr})$ pour un service standard en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu s$, $T = 100ms$. Les trois courbes de haut en bas correspondent à $\epsilon=10^{-5}$, $10^{-6}$ et $10^{-7}$ respectivement. ........................................ 85

6.6 Délai moyen $E(Y_{rr})$ (en $\mu s$) des messages de données temps réel pour un service temps réel en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu s$, $T = 100ms$ et $\epsilon=10^{-6}$ ........................................ 86

6.7 Ecart type du délai des messages de données temps réel $\sigma(Y_{rr})$ pour un service temps réel en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu s$, $T = 100ms$ et $\epsilon=10^{-6}$ ........................................ 87

6.8 Coût moyen de la synchronisation pour un service standard en fonction de $\lambda_d$. Les courbes de haut en bas représentent respectivement les variations de $P_d$, $\alpha_s E(C_2)$ et $\alpha_s E(C_1)$ en fonction de $\lambda_d$, avec $\tau = 10\mu s$, $T = 100ms$, $\epsilon = 10^{-6}$, $\lambda_r = 100\lambda_d$, $\alpha_s = 25$. ............... 90

6.9 Coût moyen de la synchronisation $P_s$ pour un service typé (en fonction de $\lambda_d$). La courbe d’en haut correspond à $\alpha_s=\lambda_d$, et celle d’en bas à $\alpha_s=25$. ........................................ 92

B.1 Format des TPDUs DT, RTD et AK ........................................ 114

C.1 Les protocoles OSI supportés par SMX V.4 ............................ 116

C.2 L’environnement d’implémentation de TPM ............................ 116
Liste des Tableaux

2.1 Temps CPU en $\mu$s pour le codage d’un entier sur 32 bit \ldots \ldots \ldots \quad 19
2.2 Comparaison des deux syntaxes BER et FTLWS \ldots \ldots \ldots \ldots \quad 19
3.1 Les paramètres QOS des applications haut débit. \ldots \ldots \ldots \quad 24
4.1 Coût des primitives \texttt{timers} fournies par Unix \ldots \ldots \ldots \quad 39
Chapitre 1

Introduction

Les réseaux de communication de données ont connu une évolution rapide durant la décennie précédente. Les avancées ont porté sur plusieurs domaines de la technologie et ont abouti au développement des réseaux à haut débit ayant des vitesses de transmission pouvant dépasser les 100 Mbit/s tels que FDDI, DQDB, et le RNIS large bande (B-ISDN).

D’autre part, des applications nécessitant des hauts débits de transmission sont en cours de développement. Ces nouvelles applications supportent une intégration de plusieurs types de trafic ayant des caractéristiques différentes, comme par exemple la transmission simultanée de données, de voix numérisée ou d’images vidéo. L’implantation de telles applications et d’applications plus “classiques” comme le transfert de fichier ou la messagerie électronique sur des réseaux à haut débit pose un problème délicat à cause des besoins différents de ces applications. Le problème a plusieurs aspects selon le niveau considéré.

En ce qui concerne les couches les plus basses c'est à dire la transmission et la commutation il s'agit de réaliser des supports de transmission ayant des bandes passantes très larges et de construire des commutateurs permettant de fonctionner à des débits de plusieurs millions paquets par seconde. La technique de multiplexage asynchrone ATM (Asynchronous Transfer Mode) a été recommandée par le CCITT pour le RNIS à large bande. Cette technique permettra de multiplexer plusieurs types de trafic ayant des statistiques variables dans le temps sur le même support de transmission.

Pour le contrôle de transmission il s’agit d’analyser les différents besoins des applications et de trouver un système de transport qui puisse répondre à ces besoins en propageant les augmentations des débits jusqu’aux applications. En effet, les protocoles de contrôle de transmission standards comme TP4 ou TCP ont été conçus pour assurer la transmission fiable des données en effectuant la correction des erreurs de transmission et le contrôle des flots de données. Pour cela des mécanismes complexes basés sur l’utilisation de temporisations et sur des retransmissions ont été adoptés. Ces mécanismes ralentissent la transmission sur les réseaux à haut débit.
actuëls par des contrôles qui ne sont plus toujours indispensables à cause des pro-
grès technologiques qui ont permis d’avoir des taux très faibles pour les erreurs de transmission.

Au niveau application, le problème général de la synchronisation de l’application et de la gestion de la transparence des données dans un réseau de machines hétérogènes devient encore plus délicat pour les applications haut débit. Il s’agit dans ce cadre de mettre au point des techniques qui permettraient de décrire la dynamique d’une application et la structure des données échangées. D’autre part, il faudra définir une syntaxe de transfert allégée permettant l’implémentation du protocole de présentation comme un filtre entre l’application et le transport.

Dans cette thèse, nous nous intéressons essentiellement aux protocoles de contrôle de transmission dans les réseaux à haut débit. Les travaux de recherche antérieurs sur ce problème suivaient l’une des deux lignes suivantes:

- Rechercher les points de blocage dans les protocoles classiques et introduire les améliorations nécessaires pour adapter ces protocoles aux débits élevés. Selon cette approche, les choix adéquats de l’architecture matérielle et de l’implémentation et le réglage des valeurs de paramètres permettent d’optimiser les performances des protocoles tout en conservant la fonctionnalité que fournisent ces protocoles standards.

- Concevoir de nouveaux protocoles pouvant répondre aux besoins des applications en tirant profit des avancées technologiques. La simplification du service transport pour répondre aux besoins d’une application particulière permet d’alléger les primitives de contrôle de transmission voire d’implémenter le protocole dans un chip VLSI.

Nous avons abordé le problème selon un axe intermédiaire: la conception d’un protocole compatible avec les normes OSI, intégrant les optimisations proposées pour les protocoles classiques et fournissant un nouveau service transport aux applications pour inclure le transfert de données “temps réel” i.e. nécessitant un minimum de contrôle au niveau de la couche transport. Pour cela nous sommes partis du protocole TP4 en proposant les modifications aux service et protocole afin d’aboutir à un protocole de transport multimédia que nous avons appelé TPM\(^1\). Ce protocole propose un moyen pour faciliter la synchronisation des différents flux de données au niveau transport. Le travail effectué fera l’objet de cette thèse.

Dans le chapitre 2 nous présentons les nouveaux changements dans les réseaux à haut débit qui ont motivé une modification des protocoles de contrôle de transmission.

\(^1\)L’ancienne appellation du protocole “TP5” a été changé d’après la proposition de Mr. Michel Diaz afin d’éviter que le protocole soit considéré une sixième classe fournissant le service de transport standard OSI.
Chapitre 2

Les réseaux à haut débit

La recherche sur les réseaux à haut débit porte essentiellement sur la technologie et les applications. Le premier axe regroupe l'ensemble des techniques nécessaires à la construction de réseaux à haut débit et notamment la transmission et la commutation de données à plusieurs Gbps. Ces techniques ont abouti à la construction de réseaux locaux, métropolitains et publics fonctionnant à des débits élevés tels que FDDI, DQDB et le RNIS à large bande. L'émergence de ces réseaux à haut débit ouvre la porte au développement d'un nouveau type d'applications réseaux nécessitant des hauts débits et intégrant la transmission de données, voix, vidéo et des images entre les entités applications. Cependant, l'existence de ces nouveaux réseaux n'est pas suffisante pour permettre aux applications de bénéficier de l'augmentation des débits déjà réalisée au niveau de la transmission à travers le réseau. Le deuxième axe de recherche, concernant le développement des applications haut débit, porte donc sur le contrôle de transmission de bout en bout et sur l'optimisation des couches hautes.

Le but de ce chapitre est de faire le point sur les technologies et l'architecture des réseaux à haut débit et de leurs caractéristiques. Les améliorations introduites par ces nouveaux réseaux nécessitent des modifications qu'il faudrait apporter aux protocoles de contrôle de transmission pour répondre aux besoins des applications.

2.1 Les avancées technologiques

Par cette dénomination, nous désignons les travaux de recherche concernant la transmission et la commutation des données à haut débit. Les recherches en transmission ont pour but la construction de systèmes capables de moduler des lasers à plusieurs Gbps. En commutation, il s'agit de concevoir et d'implémenter des commutateurs de paquets pouvant traiter plusieurs millions de paquets par seconde. Le mode de multiplexage asynchrone ou ATM (Asynchronous Transfer Mode) semble le mieux adapté pour supporter une intégration de trafics ayant des caractéristiques différentes et a été recommandé par le CCITT pour être utilisé dans le futur RNIS à large bande.
2.1.1 Transmission

Depuis leurs premières expérimentations dans les années 70, les fibres optiques ont rapidement connu des améliorations considérables avec la réalisation de fibres à faible atténuation et de lasers hautes performances fonctionnant à la température ambiante. Les systèmes à fibres ont déjà commencé à supplanter les systèmes sur support métallique, non seulement entre les nœuds d’un réseau de transmission mais aussi dans le domaine de la distribution\(^1\), il faudra s’attendre à un raccordement complet des usagers par des fibres dans la prochaine décennie. Cette évolution a été confirmée par la très grande capacité de transmission des fibres dépassant même les 30 THz. Cette bande passante permet d’enrichir les services rendus pour inclure, en plus de la communication téléphonique et la transmission de données, le transfert d’images animées.

La recherche dans ce domaine porte essentiellement sur l’augmentation de la portée et de la capacité d’une liaison par fibre optique:

- En ce qui concerne la portée, les fibres elles mêmes ayant déjà de très bonnes performances (atténuation de l’ordre de 0,2 dB/Km à 1550 nm pour une fibre unimodale), le progrès passe donc par l’amélioration des lasers d’émission et des techniques de réception. D’autre part, la technique de détection hétérodyne permet de diminuer le seuil de détection – de 1000 jusqu’à quelques dizaines de photons par élément binaire. En plus, cette technique facilite l’utilisation de l’amplification optique parce qu’elle assure un filtrage très sévère du signal reçu éliminant ainsi le bruit de l’amplificateur et permettant de simplifier les répéteurs [Mon85], [Auf89].

- Les techniques de multiplexage numérique ou fréquentiel (Wavelength Division Multiplex) permettent d’augmenter les débits de transmission, mais la mise au point de tels multiplexeurs est loin d’être maîtrisée, la technologie actuelle ne permettant pas de construire des lasers réglables assez rapidement (en quelques dizaines de ns) afin de multiplexer les paquets de différentes connexions [Maï89].

D’autre part, des efforts importants sont déployés pour développer des commutateurs optiques, afin d’assurer le brassage de réseaux, le multiplexage numérique à haut débit et de minimiser les passages optique/électrique et inversement. Le but de ces recherches est d’aboutir, même si ce n’est pas dans un futur proche, au réseau tout optique [Ster90]. Dans une perspective à plus court terme, les fibres optiques offrent un milieu de transmission à très large bande, servant comme support de base

\(^1\) Quelque 30000 Km de fibres ont été installés dans un diamètre de 50 Km dans Berlin dans le cadre du projet BERKOM [Heh89], [But89].
2.1. LES AVANCÉES TECHNOLOGIQUES

pour les réseaux à haut débit avec un taux d’erreur de bit (bit error rate) de $10^{-12}$. Un réseau basé sur la technique WDM et qui permet de supporter jusqu’à 3000 canaux de 1 Gbps chacun sur une fibre optique a été proposé récemment. Ce réseau appelé WON [Ban90a], [Ban90b] (Wavelength-Division Optical Network) utilisera ce mécanisme afin d’avoir un débit élevé en transmettant plusieurs messages à la fois sur la même fibre.

2.1.2 Commutation

Deux modes étaient classiquement utilisés: la commutation de circuits et la commutation par paquets. Aucun des deux modes n’est vraiment adapté à être utilisé sur les réseaux intégrés à haut débit [Ahm89], [Haa87]. La commutation de circuits ne permet pas un partage efficace des ressources de transmission dans le cas d’une intégration de trafics ayant des débits variables. Les avantages de la commutation par paquets sont connus: une adaptation à un trafic “bursty” et à des variations de débits. Cependant, ce n’est que récemment que des commutateurs de paquets pouvant fonctionner à de très haut débits ont pu être construits. Ces commutateurs rapides ou FPS (Fast Packet Switches), basés sur l’utilisation du parallélisme et du routage câblé sont classés selon leur architecture interne en différentes catégories:

- Les Réseaux d’Interconnexions Multiétages et en particulier les réseaux Banyan permettent d’effectuer le “self routing” des paquets commutés. Le blocage interne peut être éliminé par un étage de tri des entrées résultant en un réseau appelé “Batcher-Banyan” [Hua84]. Des publications récentes ont proposé l’utilisation de réseaux Banyan en Tandem (i.e en casse) afin d’avoir plusieurs chemins entre chaque entrée et chaque sortie. Ceci permet d’éviter le blocage interne des réseaux Banyan en reroutant les paquets en conflit vers un étage supplémentaire de réseau Banyan (voir [Tob90]). Les analyses de performances effectuées dans cet article montrent que pour un réseau (1024 x 1024), il suffit d’avoir 14 réseaux Banyan en cascade ayant chacun des tampons de 60 paquets par sortie pour assurer un débit normalisé virtuellement égal à 1 et une perte de $10^{-6}$, le trafic offert à chaque entrée étant décrit par un processus de Bernoulli de paramètre $p = 0.9$. Cette architecture uniquement formée de réseaux Banyan permet de dépasser la limite théorique sur le débit des réseaux Banyan et d’éviter la complexité de l’étage de tri des réseaux Batcher-Banyan.

- Les techniques à mémoire partagée – aussi connues sous la dénomination (Asynchronous Time Division) – permettent d’atteindre des débits de l’ordre de 280 Mbits/s pour des commutateurs 16 x 16 en utilisant des chips ECL [Ahm89].
• Les architectures à milieu partagé utilisent généralement un bus ou un anneau ayant une bande passante multiple de celle des lignes d’entrées et de sorties pour commuter les paquets (par exemple le bus DQDB).

• Les commutateurs cross-bar ayant un chemin indépendent entre chaque paire entrée sortie.

L’évaluation de performances [Fdi89], [Puj89] des différentes techniques de commutation ainsi que la conception d’algorithmes d’allocation de ressources en fonction de la qualité de service requise par différents types d’utilisation font l’objet de thèmes de recherche de nombreuses équipes [Woo90], [Vir89], [Fer89].

2.1.3 ATM

Le mode de transfert asynchrone (ATM) a été recommandé par le CCITT comme support de transmission sur le RNIS large bande [Cci88]. ATM est une technique de multiplexage et de commutation de paquets permettant d’obtenir des débits de transmission élevés en simplifiant les procédures de contrôle. Les taux d’erreurs réduits des supports de transmissions éliminent le besoin de la couche de liaison de données et donc de la correction d’erreurs au niveau liaison. Dans ATM l’information à transferer est segmentée en de petites cellules (cells) de taille fixe égale à 53 octets (voir figure 2.1). La petite taille des cellules permet un partage “à grain fin” de la bande passante: ceci diminue les délais d’accès au réseau ainsi que les délais dans les nœuds et les commutateurs et facilite le réglage dynamique des débits pour les applications demandant des débits variables [Rob87], [Tho84]. Les cellules seront identifiées et commutées à l’aide d’une étiquette dans l’entête composée de deux champs: le VPI et le VCI (Virtual Path Identifier et Virtual Channel Identifier) [Sat90]. ATM est une technique “orientée connexion”: lors de l’établissement de connexion les informations de contrôle (véhiculées sur un canal de signalisation séparé ou dans l’entête du premier paquet) sont utilisées pour le choix de la route et par suite les VCI des différentes liaisons sont alloués et les ressources nécessaires pour cette connexion sont réservées. Un mécanisme de contrôle de l’admission (admission control) est responsable de l’acceptation ou du refus d’une connexion [ORe88]. Sur une connexion ATM, les cellules pourraient être altérées, ou perdues mais beaucoup plus rarement reçues hors séquence. Le modèle d’erreur de bout en bout est donc seulement celui d’un pourcentage de perte de données.

Durant la phase de transfert de données les cellules sont injectées dynamiquement par l’émetteur selon le débit dans la limite des ressources réservées. Des études sur les contrôle de l’admission de nouveaux clients sont menées actuellement dans le but de trouver la politique optimale d’allocation de ressources sans toutefois dégrader la qualité de service des connexions déjà établies [Hui88], [Woo90]. Ceci
2.1. LES AVANCÉES TECHNOLOGIQUES

Entête

<table>
<thead>
<tr>
<th></th>
<th>48 octets</th>
</tr>
</thead>
</table>

L’entête comprend:

- 4 bits pour le contrôle de flux générique (GFC)
- 3 octets pour l’identificateur de canal (VPI+VCI)
- 2 bits pour le type de données
- 2 bits réservés
- 1 octet pour le contrôle d’erreur

Figure 2.1: La cellule ATM

permettra d’éviter le contrôle de flux sur les différentes liaisons d’une connexion. ATM fournira donc une connexion de type “flot de données” ou “stream” où les cellules seront transmises en séquence avec un pourcentage d’erreur très réduit. Toutefois, un mécanisme de régulation ou de contrôle de congestion est requis pour empêcher un “dépassement de quota” par les clients: dans ce cas les cellules en surplus peuvent soit être rejetées avant le point d’accès au réseau (ou dans une passerelle intermédiaire) soit être transmises avec une priorité inférieure [McA89]. Si le taux de perte dû au contrôle de congestion devient inacceptable pour une application ayant un débit ajustable (par exemple une application vidéo avec compression adaptative), elle devra réguler son débit (en changeant par exemple le niveau de la compression et par suite la qualité de la transmission) afin d’optimiser le débit global tout en conservant un taux d’erreur réduit.

Les connexions ATM offrent aux utilisateurs un service flexible de transfert de données: les débits de transmission peuvent varier selon les besoins des utilisateurs. Ce service est fourni à tous les types de trafic qui seront transmis sur un réseau ATM. Les différents besoins des applications seront traités par une couche d’adaptation d’une façon transparente au niveau ATM. Le rôle de cette couche est donc d’adapter le service de base – flot de données ATM – aux différents profils requis par les utilisateurs. Parmi les fonctions de contrôle que devrait assurer la couche d’adaptation citons la détection et la correction d’erreur de bout en bout, la segmentation/réassemblement de paquets, le contrôle de flux, la réduction de l’impact des variations du délai de transmission des cellules sur la qualité de service fournie à l’utilisateur. La classe de service requise par l’application détermine les fonctions de contrôle à appliquer. D’autres problèmes restent à résoudre comme la conception de l’architecture de contrôle du réseau global de la façon la plus flexible. Le contrôle pourra être effectué par des nœuds intelligents connectés entre eux par les liaisons
de signalisation pour former le "plan de contrôle" (control plan) [Gec89], [Har89].
Les réseaux ATM émergeront dans le futur proche, Bellcore a déjà annoncé SMDS (Switched Multi-Megabit Data Service), un service public de communication de données (qui serait opérationnel en 1992) [Dix90] utilisant un protocole proche de celui de DQDB au dessus de la couche ATM. Un groupement d’industriels européens (Alcatel-Bell, ATEA/Siemens, Philips et autres) annoncent la sortie en 1992 de commutateurs ATM à large bande.

2.2 Les réseaux à haut débit

Des réseaux basés sur les technologies présentées dans la section précédente sont déjà opérationnels comme le réseau FDDI. D’autres sont en cours de développement comme DQDB\(^2\) ou le RNIS large bande. Les premiers produits sortiront dans quelques années.

2.2.1 FDDI

FDDI ou Fiber Distributed Data Interface est un réseau local utilisant les fibres optiques comme support de transmission et fonctionnant à un débit de 100 Mbit/s. FDDI est un anneau à jeton reconfigurable pouvant avoir un diamètre maximal de 100 kilomètres et supporter jusqu’à 500 stations [Rose89], [Ross84]. En 1987, le sous comité X3T9.5 de l’ANSI a adopté la proposition de norme concernant la couche MAC de FDDI [ANS87], et en 1989 ceci est devenu une norme ISO (9314-2).

FDDI utilise un protocole de jeton (Timed Token Rotation Protocol) pour fournir deux classes de priorités pour un service de commutation de paquets: le trafic *synchron* ayant un temps de réponse borné et le trafic *asynchron* utilisant la bande passante résiduelle et par suite n’ayant pas de garantie sur le temps de réponse. Deux types de trafic asynchrone sont définis: en mode restreint et non restreint. Le mode restreint permet l’utilisation de la bande passante disponible pour le trafic asynchrone par quelques stations privilégiées possédant le jeton “restreint” afin de permettre des “dialogues” entre ces stations.

Chaque station contient une temporisation de rotation de jeton *Token Rotation Timer* pour mesurer les inter-arrivées successives du jeton à une station. Une valeur *cible* du TRT le *Target Token Rotation Time* est négociée lors de la configuration de l’anneau. Le TTRT est choisi suffisamment petit pour satisfaire les besoins de toutes les stations en ce qui concerne le trafic synchrone pourvu que la capacité

\(^2\)Alcatel-Bell a déjà (1991) une implémentation de DQDB qui interconnecte deux réseaux locaux Ethernet.
du réseau ne soit dépassée. Si le TRT mesuré est supérieur au TTRT (jeton en retard) la station transmet seulement le trafic synchrone jus­qu’à l’expiration du temps de ré­tention du jeton (Token Holding Time). Si, par contre, le jeton est en avance (TRT < TTRT) la station s’attribue le droit d’émettre son trafic synchrone et asynchrone jus­qu’à l’expiration du THT. Dans la littéra­ture [Joh87], [Sev87] on trouve des modélisations du protocole du jeton temporisé qui montrent que le temps maximum de rotation du jeton ne peut dépasser 2.TTRT, fixant ainsi la limite du temps de réponse pour le trafic synchrone.

FDDI a été conçu initialement pour l’interconnexion de super calculateurs. Pour­tant, le champ d’applications de FDDI s’est rapidement élargi et inclut actuelle­ment l’utilisation en réseau “backbone” pour l’interconnexion de résea­ux locaux ou en réseau de stations de travail haute performance.

FDDI II est une version “améliorée” de FDDI ajoutant une troisième classe de service: le trafic isochrone pour lequel un délai de transmission fixe est assuré. La bande passante de FDDI II peut donc être partagée en 16 canaux (Wide Band Channels) pour fournir un service de commutation de circuits avec une capacité de 6.144 Mbit/s chacun.

Afin de supporter les deux modes de commutation (de paquets et de circuits) une structure de commutation hybride est utilisée: une trame de taille fixe est générée toutes les 125 µs par une station maître. Dans la trame des slots (WBC) sont réservés au trafic isochrone, l’accès au reste de la bande passante étant contrôlé par le protocole de jeton temporisé. En intégrant ces deux modes de transmission FDDI II peut supporter la transmission de données multimédias.

FDDI (I et II) est une bonne solution pour un réseau local 100 Mbit/s. Cependant, l’efficacité d’un anneau à jeton diminue avec la vitesse de transmission et avec la taille du réseau. Ceci est dû à l’augmentation du temps de transit du jeton entre les stations, temps pendant lequel aucune station ne peut émettre. Ces limitations deviennent importantes avec des vitesses de l’ordre du Gbps. Dans [Mül90], des analyses montrent que pour un débit d’un Gbps, l’utilisation maximale est de l’ordre de 0,5 de la capacité au niveau physique pour un anneau de 50 stations ayant un diamètre de 100 Km et des paquets de 1 Koctets. Il faudra donc chercher des solutions autres que FDDI pour des débits supérieurs à 100 Mb/s. Par ailleurs, le retard de FDDI II par rapport à FDDI et l’incompatibilité avec FDDI limite sérieusement le déploiement de cette deuxième version.

D’autres projets ont proposé des solutions proches de FDDI. La communauté Eu­ropéene participe à ces recherches par des projets ESPRIT dont LION [Ape88], [Leg90], [Luv87] et BWN [Dant86], [Pes87]. L’anneau Lion (Local Integrated Opti­cal Network) utilise une technique de commutation hybride pour fournir des services de commutations de paquets et de circuits similaire à FDDI II. Le débit est de 140
Mbit/s, BWN ou (Backbone Wideband Network) est un autre exemple de réseau local de recherche conçu pour un large site et fonctionnant à un débit de 134 Mbit/s. L’ONERA (Office National d’Etudes et de Recherches Aérospatiales) a lancé en Décembre 89 un projet de réseau local à 1 Gbit/s [Pel90].

2.2.2 DQDB

DQDB ou (Distributed Queue Dual Bus) fait l’objet d’une proposition de standard (IEEE 802.6) pour les réseaux métropolitains ou MANs. Cette proposition définit la structure d’un MAN comme étant une interconnexion de sous réseaux DQDB. Même si l’interconnexion des sous réseaux n’est pas définie dans le standard, les services offerts par le MAN correspondent à la spécification de DQDB.

DQDB utilise deux bus unidirectionnels (A et B) (voir figure 2.2). Chaque station peut communiquer avec les autres stations en envoyant l’information sur l’un des deux bus selon la situation de la destination. Cette solution est préférable à l’anneau quand la distance entre les stations est grande parce qu’elle permet d’éviter les délais de circulation du jeton entre les stations [Spa90]. La transmission sur les bus est discrétisée: des slots de taille fixe (53 octets) sont générés par les stations terminales sur les deux bus. Les paquets à transmettre ont généralement une taille beaucoup plus grande et seront segmentés et transmis sur plusieurs slots.

Un service “sans connexion” est déjà décrit par IEEE 802.6. Les modes “avec connexion” et “transmission isochrone” sont aussi possibles et devraient faire partie d’une autre version du standard [Bud90]. Pour le trafic isochrone, un mécanisme de réservation est utilisé afin de garantir un temps de réponse fixe.

La politique d’accès est basée sur l’émulation d’un mécanisme de file distribuée permettant à chaque station de savoir durant quel slot libre elle pourra transmettre. Pour cela DQDB utilise deux bits de contrôle par slot, le B/F (occupé/libre) et
2.2. LES RÉSEAUX À HAUT DÉBIT

REQ (requête d’émission). Une station ayant un segment à transmettre sur le bus A positionnera le bit REQ du premier slot ayant le bit REQ à zéro sur le bus B. Deux compteurs permettent l’émulation d’une file FIFO de requêtes. La station met une requête dans la file (incrémenté le compteur correspondant) à chaque passage de slot ayant REQ = 1 sur le bus B. Une requête locale ne peut être enregistrée qu’après le service de la précédente requête locale. La première requête dans la file est éliminée (le compteur est décrémenté) à chaque passage d’un slot libre sur le bus A. Si cette requête est locale le bit B/F de ce slot est mis à 1 et le segment correspondant sera transmis dans le slot. Sinon le slot sera utilisé par les stations en aval.

Ce mécanisme permet d’éviter le monopole des slots libres par les stations situées le plus en amont, pourtant il pourra y avoir des situations où des stations sont plus avantagées que d’autres à cause du délai de propagation sur le bus [Fdi90, [Won89]. D’autre part, le protocole d’accès étant distribué, il n’y a pas une limite à priori sur le nombre de stations qui pourront être attachées au réseau. Contrairement à FDDI, le débit utile maximal qui pourrait être atteint ne se dégrade pas avec l’augmentation de la distance ou de la capacité des lignes. Cependant, le débit utile est difficile à estimer et varie selon les stations surtout à charge élevée: dans [Mül90] des simulations ont montré que pour un bus dual DQDB ayant 10 nœuds, le débit utile de la dixième station ne dépasse pas 1% du débit de la première station quand le débit sur le bus dépasse 1 Gb/s.

Des recherches récentes sur l’amélioration de l’équitabilité de DQDB ont proposé un nouveau protocole d’accès DQMA permettant aussi d’avoir plusieurs requêtes d’émission locale afin de pouvoir transmettre les segments d’un même paquet dans des slots consécutifs [Mül90].

DQDB est une solution intéressante pour les réseaux MANs, toutefois, il ne pourra pas servir comme réseau backbone pour le RNIS à cause des limitations dues à l’utilisation du mécanisme de commutation par milieu partagé. Le trafic prévu sur le RNIS étant plus important et moins “bursty”, la commutation par milieu partagé est moins attrayante; le réseau backbone devra avoir dans ce cas une bande passante proportionnelle à la somme de celles des réseaux d’accès.

2.2.3 Le RNIS

Le RNIS (Réseau Numérique à Intégration de Services) à large bande est en cours de standardisation par le groupe d’étude XVIII du CCITT (série I des recommandations). Le but étant de définir un réseau capable d’intégrer différents trafics de données, de voix, ou de vidéo [Hän89, [Sta88]. Deux modes de transfert sont proposés:
CHAPITRE 2. LES RÉSEAUX À HAUT DÉBIT


- STM: le mode de transfert synchrone (Synchronous Transfer Mode) permet de fournir un service de commutation de circuits pour les trafics de téléphone ou de vidéo. Plusieurs “canaux” de capacité fixe allant de 16 kbit/s jusqu’à 140 Mbit/s pourront être réservés pour la transmission. Le système de signalisation SS7 du CCITT sera utilisé (canal D) pour fournir la signalisation pour le RNIS large bande.

Dans une première phase, des structures hybrides ATM/STM seront utilisées. Pourtant, le mode ATM est le mieux adapté pour devenir le support de base pour la transmission sur les futurs RNIS à large bande.

2.3 Structure des réseaux haut débit

Les réseaux ATM vont apparaître dans quelques années. En ce qui concerne l’accès au réseau, les utilisateurs pourraient être raccordés directement au réseau ATM. Cependant, la plupart des machines existantes sont attachées à des réseaux locaux interconnectés par des liaisons haut débit. Les réseaux ATM ont l’avantage de fournir des débits plus élevés sur des distances plus grandes. Ils seront donc utilisés en premier lieu là où ces avantages sont le plus prononcés, c’est à dire comme des réseaux backbone ou de campus assurant l’interconnexion de réseaux Ethernet ou FDDI. Le réseau DQDB peut servir comme réseau d’accès au backbone ATM. L’architecture d’interconnexion est montrée dans la figure 2.3.

Une structure générale a été proposée par Bellcore: SMDS (Switched Multi-megabit Digital Service) un service public de communication de données. Selon cette structure (voir figure 2.4) les utilisateurs seront connectés aux commutateurs métropolitains par le réseau local ou via un réseau métropolitain. DQDB a été choisi comme système de commutation métropolitain et sera donc le réseau d’accès pour les utilisateurs.

2.4 Contrôle de transmission

Les réseaux à haut débit supporteront un grand nombre de paquets circulant en même temps dans le réseau. Si on veut assurer la transmission à haut débit et à
délais réduits de ces paquets toutes les composantes du “chemin de transmission” doivent être optimisées: les protocoles de niveau MAC, réseau et transport devraient être révisés afin de permettre la propagation des améliorations de débits jusqu’aux niveaux supérieurs [Jaoq87], [Gun89], [Schw90] (voir figure 2.5)

Niveau MAC

L’augmentation de la fréquence des paquets surcharge les systèmes et limite les performances. L’adoption d’un mécanisme de priorité ne permet pas de résoudre le problème sur les stations de travail à cause du besoin d’empêcher le monopole du processeur par les applications “réseau”perme. Les stations de travail existantes peuvent traiter jusqu’à 200 paquets par seconde (à base de 68020); le progrès de la technologie des processeurs (architecture RISC) permet déjà d’améliorer les performances par un ordre de grandeur (2000 paquets par seconde). Pourtant on est loin des $10^6$ mini-paquets par seconde qui seront transmis sur un réseau ATM: si chacun des ces mini-paquets devra générer une interruption, le traitement des ces interruptions monopolisera même les processeurs les plus puissants. Les couches MAC des réseaux à haut débit devraient inclure une procédure de segmentation / réassemblage câblée, qui sert à grouper les paquets avant de les passer au système d’exploitation. Des conceptions et implémentations de la couche MAC pour des réseaux à haut débit accomplissant le groupement de paquets ont déjà été proposées [Pes87], [Sko89]. Dans le paragraphe 4.1.4 nous présentons le diagramme bloc d’une conception d’un interface réseau implémentant une couche MAC haut débit.

\footnote{Si on considère le coût moyen de traitement d’un paquet égale à 200 instructions, le traitement d’un million de paquets par seconde nécessite un processeur dédié de 200 MIPS.}
Figure 2.4: Architecture “Switched Multimegabit Digital Services”
Figure 2.5: L’architecture en couches: où optimiser?

Niveau Réseau

Les deux modes du service réseau: le mode sans connexion et le mode avec connexion vont continuer à coexister [ISO87]. L’interconnexion est généralement assurée par des passerelles fonctionnant en mode sans connexion effectuant le routage des datagrammes en se basant sur les adresses présentes dans l’entête [ISO88a], [ISO90]. Le relais des datagrammes sur des réseaux qui fournissent le service réseau avec connexion est accompli par l’établissement d’un circuit virtuel sur lequel seront transmis les paquets disponibles pour la transmission [Joh88]. L’utilisation de ce modèle pour les réseaux ATM suppose la résolution de deux problèmes:

- l’analyse des adresses de chaque paquet implique une charge importante des processeurs si le nombre des paquets est élevé
- la commutation des paquets selon le mode PAPS (Premier Arrivé Premier Servi) génère des délais d’attentes variables dans les tampons, dégradant ainsi la transmission de flots temps réel.

L’utilisation de processeurs spécialisés dans les routeurs et de la technique de groupement de paquets au niveau de la couche MAC permet de diminuer l’impact du nombre des paquets. D’autre part les applications temps réel nécessitent un mécanisme de réservation de ressources (à l’ATM) permettant l’envoi de données sans effectuer le contrôle de flux et des erreurs. Un tel protocole se situe au niveau transport du modèle OSI. Les problèmes de la couche réseau ne représentent donc pas de sérieuses limitations car ils peuvent être ramenés à ceux des couches MAC et transport.

Le problème de l’optimisation de la couche transport sera étudié avec plus de détail dans les chapitres suivants.
2.5 Développement des applications haut débit

L’optimisation des couches hautes passe essentiellement par l’étude de la synchronisation des applications et de la gestion de la transparence de données [Rose90], [Tan88], [Tan90], [Boun90], [Dog88], [Ghe90], [Cla90]. Le problème de synchronisation des applications distribuées [Ste90] fait partie de la recherche sur les modèles de communication des applications distribuées en général. Les solutions existantes utilisent un modèle client-serveur et des appels de procédures distantes (Remote Procedure Call ou RPC) pour la communication entre le client et le serveur [Bir84], [Spe82]. Le modèle RPC est adapté à une structure d’échange de type question-réponse et ne peut pas représenter une solution générale pour toutes les applications. Les applications multimédias ne suivent pas nécessairement le modèle client-serveur: une application de vidéo-conférence implique des échanges d’informations à haut débit entre les différents participants. Il s’agit donc de pouvoir décrire le mécanisme de synchronisation et d’échange des données d’une façon qui ne ralentisse pas les transferts. Les protocoles développés dans le cadre de l’architecture ISO (TP, CCR, ROS) pour les couches supérieures ne prennent pas en compte ces contraintes. Il faudrait donc que ces standards soient revoir afin de les adapter aux besoins des applications haut débit.

La gestion de l’hétérogénéité consiste à définir les mécanismes permettant d’alléger le protocole de présentation. Le coût élevé de ce protocole découle de la nécessité de “présenter” dans un format standard des données de structures incompatibles. Les différences de matériel, des langages de programmation et des compilateurs d’un même langage sont à l’origine de l’incompatibilité des structures de données entre les différentes machines d’un réseau. Dans le cadre de l’ISO, le service de présentation est basé sur le langage ASN.1 [ISO87b] (Abstract Syntax Notation One), une évolution de la recommandation X.409 du CCITT. ASN.1 sépare la partie de spécification de données c’est à dire la définition d’une syntaxe abstraite, du codage de données selon une syntaxe de transfert. Les règles de codage de ASN.1 (BER ou Basic Encoding Rules), qui décrivent la syntaxe de transfert standard, ralentissent les fonctions de codage et de décodage; les éléments de données sont codés selon un schéma récursif (Type, Longueur, Valeur), les champs étant codés sur des longueurs variables afin de réduire au minimum le nombre d’octets nécessaires pour le transfert (voir tableau 2.1).

Avec l’apparition des réseaux à haut débit, les performances des applications seront sérieusement affectées par la lenteur des procédures de codage et de décodage orientées octet: tous les octets du message doivent être traités. Des recherches sur la définition de syntaxe de transfert à coût léger (FTLWS ou Flat Tree Light Weight Transfer Syntax) ont été menées à l’INRIA [Huit89b]. Cette nouvelle syntaxe de transfert utilise des règles de codage orientées “mots” et des mappages de tampon
2.5. DÉVELOPPEMENT DES APPLICATIONS HAUT DÉBIT

<table>
<thead>
<tr>
<th>transformation</th>
<th>SPS7</th>
<th>GOULD</th>
<th>Sun 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignation (dans une boucle)</td>
<td>3.7</td>
<td>2.0</td>
<td>2.9</td>
</tr>
<tr>
<td>recopie mémoire (par ligne de 256 entiers)</td>
<td>0.9</td>
<td>2.9</td>
<td>1.1</td>
</tr>
<tr>
<td>codage suivant ASN.1 (appel de fonction)</td>
<td>18.6</td>
<td>10.9</td>
<td>25.5</td>
</tr>
<tr>
<td>codage suivant ASN.1 (code en ligne)</td>
<td>10.9</td>
<td>6.8</td>
<td>12.5</td>
</tr>
<tr>
<td>décodage suivant ASN.1 (appel de fonction)</td>
<td>20.9</td>
<td>10.7</td>
<td>24.3</td>
</tr>
<tr>
<td>décodage suivant ASN.1 (code en ligne)</td>
<td>12.2</td>
<td>6.2</td>
<td>15.2</td>
</tr>
<tr>
<td>conversion “big-endian”/“little-endian”</td>
<td>7.0</td>
<td>4.6</td>
<td>8.1</td>
</tr>
</tbody>
</table>

Tableau 2.1: Temps CPU en μs pour le codage d’un entier sur 32 bit

<table>
<thead>
<tr>
<th>Syntaxe de transfert</th>
<th>Durée (ms)</th>
<th>Taille (octets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASN1-BER</td>
<td>codage</td>
<td>8800</td>
</tr>
<tr>
<td></td>
<td>décodage</td>
<td>51280</td>
</tr>
<tr>
<td>Syntaxe allégée (FTLWS)</td>
<td>codage</td>
<td>340</td>
</tr>
<tr>
<td></td>
<td>décodage</td>
<td>20</td>
</tr>
</tbody>
</table>

Tableau 2.2: Comparaison des deux syntaxes BER et FTLWS

mémémoire et permet d’obtenir des temps d’exécution jusqu’à parfois 20 fois plus rapides que les procédures standards BER. Dans la tableau 2.2 on compare les durées en millisecondes des opérations de codage et de décodage d’une matrice de 500 x 500 entiers,\(^4\) et la taille de l’information codée selon les 2 syntaxes BER et FTLWS [Huit90b].

Par ailleurs, la lenteur des procédures de codage et de décodage standards impose une double copie des données: vue la difficulté d’être codées “en vol”, les données sont codées dans un tampon en mémoire application puis copiées dans les tampons de la couche transport. Les optimisations de la nouvelle syntaxe de transfert permettent d’envisager l’implémentation du protocole de présentation en un “filtre” entre l’application et la couche de transport. Ceci nécessite une redéfinition des mécanismes de synchronisation afin de confirmer l’interaction entre la présentation et la transmission des données (en d’autres termes inverser l’ordre des couches session et présentation) [Huit89a]. Les travaux sur ce sujet ont démarré à l’INRIA et seront poursuivis dans le cadre du projet ESPRIT OSL-95.

---

\(^4\) L’exemple a été choisi de façon à souligner l’avantage de FTLWS. En effet, avec des données de types mixtes la réduction de la durée de codage et de décodage est atténuée et on arrive parfois à des facteurs de gain de 1 à 1.5.
Chapitre 3

Applications et Services de transport

3.1 Les applications haut débit

Avec l’émergence des futurs réseaux haut débit, de nouvelles classes d’applications pourront être développées. Le groupe d’études XVIII du CCITT a proposé une classification de ces applications haut débit [Cci86] qui seraient supportées par le RNIS à large bande comme suit:

- Les applications interactives nécessitant une transmission en temps réel entre la source et la destination de différents flots de données intégrés ou séparés. Parmi ces applications on note le vidéo-téléphone, les vidéo-conférences (requérant l’échange de grandes quantités d’information dans un temps bref et par suite nécessitant un fort débit et des délais brefs), la transmission d’images fixes ou animées (par exemple graphiques générées en temps réel permettant de visualiser les résultats de gros programmes de simulation), la transmission de documents multimédias (recommendations ODA/ODIF), ainsi que les applications standards de l’OSI (transfert de fichiers, terminal virtuel, processus transactionnels, ...) qui seront utilisées à des débits plus élevés et sur des distances plus grandes.

- Les services de messagerie mono ou multimédias. Ces derniers permettent aux utilisateurs de tirer parti de différents médias (graphiques, images, voix, vidéo en plus des données textuelles) dans le transfert des documents, accroissant ainsi l’utilité des services de messagerie.

- Les services de recherche d’informations comme les systèmes vidéo-tex distribués sur des réseaux nationaux, les recherches de documents audio ou vidéo,

- Les services de diffusion TV, radio, ou de données. La vidéo sur demande (video on demand) et la vidéo-tex large bande font partie des services de diffusion où un contrôle de présentation exercé par le destinataire permet de
choisir l’information désirée soit pour sélectionner un film (video on demand) ou pour consulter une page d’un menu (dans une application vidéo tex).

- Les services de surveillance où les informations collectées de différents équipements de contrôle (capteurs, caméras) sont acheminées vers un centre de contrôle.

La mise au point de ces applications nécessite d’abord l’existence d’un moyen efficace pour effectuer la transmission de données avec un coût minimal. Il s’agit donc d’abord de spécifier les besoins transport de ces applications et par suite de décrire les mécanismes appropriés qui vont permettre le développement d’un système four-nissant les services requis.

### 3.2 les flots de données

La plupart des applications citées ci dessus supportent des flots de données variés et ayant des caractéristiques différentes; il est donc difficile de trouver un système de communication qui réponde à tous les besoins simultanément. Depuis la préhistoire, des réseaux indépendants sont utilisés pour le téléphone et la télévision; la transmission de données utilisait les liaisons téléphoniques ou telex à des débits très faibles. Chaque “réseau” était optimisé pour l’application supportée et ne pouvait pas transmettre efficacement d’autres types de trafi, Avec le RNIS, il est possible d’intégrer les flots de données sur le même réseau. Le défi reste tout de même de concevoir et de mettre en œuvre les mécanismes permettant de garantir les paramètres de “qualité de service” requis par chacune de ces applications:

- La voix numérisée (paquetisée ou sur voies MIC) a besoin d’un débit de 64 kbit/s par connexion. Le délai maximum de bout en bout permis pour une application de voix est de l’ordre de 150 ms ± 5%. D’autre part, le taux global de perte de paquets ne devrait pas dépasser 1% [Sud87], [Yin87], [Nic90], [Lei88].

- Les images vidéo nécessitent aussi un délai de transmission du même ordre que celui de la voix numérisée. Le taux d’erreurs résiduelles (c’est à dire le pour-centage de trames erronées ou perdues) est de l’ordre 10^{-3} [Woo90], [Heh89], [Mag88]. La transmission d’images comprimées impose des contraintes plus strictes en ce qui concerne les délais et le taux d’erreurs (voir tableau 3.1). Les VBR (“Variable Bit Rate encoders”) utilisent la technique du codage adaptatif

---

1 Vers les années 60-70.
différentiel: le débit sera alors lié à la variation du signal. Pour une application de type vidéo-conférence les débits vont de 34 Mbit/s à 70 Mbit/s. La transmission d’images de télévision haute définition nécessite un débit allant de 140 à 560 Mbit/s [Puj90a], [Puj90b].

- Les données standards échangées par les applications de transfert de fichiers, la messagerie, le calcul distribué, etc. Les débits varient selon les applications dans la limite de la bande passante du réseau. Ce type de données doit être sécurisé contre les erreurs afin de garantir à l’application la fiabilité de la transmission.

Les données de voix ou d’images animées seront désignées par l’appellation *flot de données temps réel* alors que les données sécurisées seront appelées *flot de données standards*.

### 3.3 Les paramètres de qualité de service

L’utilisation des paramètres de qualité de service permet d’exprimer les caractéristiques d’une connexion de transport: “Ces paramètres permettent aux utilisateurs du service de transport de disposer d’une méthode pour spécifier leurs exigences et au fournisseur du service de transport de disposer d’une base pour le choix du protocole” ([Rec84b] page 104). Après que l’application ait formulé ses besoins en terme de paramètres QOS, l’entité transport teste localement la possibilité d’assurer la qualité de service (à partir d’informations sur la qualité de service réseau). Si le réseau peut fournir les services requis, l’entité transport transmet ces paramètres à l’entité homologue afin de négocier la qualité de service avec l’application appelée. Les valeurs adoptées sont par suite utilisées pour allouer les ressources au niveau des entités transport locales et distantes. Les paramètres de qualité de service définis dans [Rec84b] sont cités ci-après:

1. Délai d’établissement de connexion de transport
2. Probabilité d’échec d’établissement de connexion de transport
3. Débit
4. Temps de transit
5. Taux d’erreurs résiduelles
6. Probabilité d’incidents de transferts
7. Délai de libération de connexion de transport

8. Probabilité d'échec de libération d'une connexion de transport

9. Protection des connexions de transport

10. Priorité d'une connexion de transport

11. Probabilité de rupture d'une connexion de transport

En ce qui concerne les applications haut débit, les paramètres les plus significatifs sont le débit, le temps de transit et le taux d'erreurs résiduelles, mais aussi la variation du temps de transit et la variabilité du débit. Toutefois, les paramètres QOS qui concernent les deuxièmes moments du délai et du débit ne sont pas définis actuellement dans le standard OSI. En plus de l'introduction de ces paramètres, il devrait y avoir un moyen de suivre l'évolution de la valeur des paramètres QOS durant toute la durée de la connexion de transport, et même de pouvoir renégocier des nouvelles valeurs suite à la demande du réseau ou de l'application.

Le tableau 3.1 résume les valeurs de ces paramètres pour différentes applications haut débit (Voir [Heh89], [Puj90b], [Sud87]).

<table>
<thead>
<tr>
<th>Paramètres</th>
<th>débit</th>
<th>délai</th>
<th>% acceptable d'erreurs de paquets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voix</td>
<td>64 Kb/s</td>
<td>150ms ± 5%</td>
<td>1%</td>
</tr>
<tr>
<td>Vidéo</td>
<td>140 Mb/s</td>
<td>150ms ± 5%</td>
<td>0.1 %</td>
</tr>
<tr>
<td>Vidéo avec compression</td>
<td>34 Mb/s</td>
<td>150ms ± 0.5%</td>
<td>(10^{-3})</td>
</tr>
<tr>
<td>Terminal virtuel</td>
<td>faible</td>
<td>faible</td>
<td>0</td>
</tr>
<tr>
<td>Transfert de masse</td>
<td>important</td>
<td>~ 1 s</td>
<td>0</td>
</tr>
<tr>
<td>Messagerie multimédia</td>
<td>important</td>
<td>1 min ~ 1h</td>
<td>0</td>
</tr>
</tbody>
</table>

Tableau 3.1: Les paramètres QOS des applications haut débit.
3.4 Le service de transport

La couche transport est responsable de la transmission fiable de données entre les systèmes finaux. Les primitives de cette couche sont nécessaires pour combler l’écart entre les services fournis par la couche réseau et les services à offrir aux utilisateurs du service de transport (voir figure 3.1).

![Figure 3.1: Le service de transport OSI](image)

Parmi les primitives de la couche transport décrites dans la recommandation du CCITT ([Rec84a]) citons:

1. Transfert d’unités de données du protocole de transport (TPDU ou Transport Protocol Data Unit),
2. Multiplexage et démultiplexage de plusieurs connexions de transport sur une même connexion réseau,
3. Éclatement et recombinaison,
4. Détection d’erreurs (perte, altération, déséquenancement ou duplication de TP-DUs),
5. Reprise sur erreur,
6. Choix du service réseau qui répond le mieux aux exigences de l’utilisateur du service transport,
7. Choix de la taille de TPDU optimale,
8. Contrôle de flux,
9. Concaténation et séparation,
10. Transfert de données exprès,
11. Segmentation et réassemblage,
Ces primitives sont classées en quatre catégories: celles utilisées en toutes circonstances au cours d’une connexion de transport (1,2,4,5), celles qui concernent l’établissement de connexion (6,7), celles qui concernent le transfert de données (3,8-11), celles qui concernent la libération de la connexion. Ces primitives sont utilisées pour améliorer les performances de la transmission (3,6-10) ou pour augmenter les fonctionnalités offertes à l’application (2,4,5,11).

La couche transport a été conçue avec la philosophie de décharger les applications du transfert de données. Il y a un seul service transport standard et les différentes classes de protocole de transport sont utilisées au dessus de réseaux ayant des caractéristiques différentes afin de fournir le même service aux applications en évitant de dupliquer une fonctionnalité déjà fournie par le service réseau. Pourtant, les applications à haut débit ont des besoins “transports” variés. La prochaine section est consacrée à la description de ces besoins.

### 3.5 Les besoins des applications haut débit

Les protocoles de transport classiques utilisent la numérotation de paquets et les accusés de réception pour détecter et corriger les erreurs de transmission comme la perte, la duplication ou le déséquenancement de paquets et pour fournir le contrôle de flux de bout en bout [Sun78]; un mécanisme de total de contrôle (checksum) est utilisé pour détecter les altérations de paquets. Ces techniques ne sont pas bien adaptées pour contrôler les flots de données des applications haut débit. En effet, ces techniques demandent d’importantes ressources de traitement et de place mémoire, et résultent en une augmentation des délais de transmission [Huit90a]:

- La correction d’erreur de bout en bout basée sur l’envoi d’accusés de réception par le destinataire oblige la source à garder une copie de tous les paquets dont la réception n’a pas été confirmée. La taille des tampons mémoires nécessaires pour garder cette copie est proportionnelle au délai de retour de l’accusé de réception et au débit du réseau (5 Mégaoctets pour un réseau de 1 Gbps et un délai de 40 ms.)

- Le traitement d’un accusé de réception demande approximativement le même temps de calcul qu’un paquet de données, alors que la vitesse du processeur est peut être un goulot d’étranglement pour les applications haut débit.

- Les flots de données temps réel comme la voix numérisée demandent des délais de transmission stables. En plus, en cas de perte de paquets il est important d’avoir toujours les derniers échantillons du signal au détriment d’un certain pourcentage de perte, lequel est toléré par l’application (on préfère passer à la
3.5. LES BESOINS DES APPLICATIONS HAUT DÉBIT

reconstitution du reste du signal sonore que d’avoir le haut parleur silencieux
en attendant une retransmission). Permettre la retransmission occasionnelle
de paquets va retarder les autres paquets par au moins le délai de transmission
d’un paquet (le contrôle de flux étant lié au contrôle d’erreur par le mécani-
isme de fenêtre de transmission). En augmentant la variabilité des délais,
les retransmissions non seulement diminuent la qualité de service fournie à
l’application (ceci est montré dans la section 6.3), mais nécessitent aussi des
tampons de reséquencement de la même taille que ceux requis pour garder
les copies pour une éventuelle retransmission (proportionnelle donc à la borne
supérieure estimée du délai de transmission et non à sa variance).

- Le calcul du total de contrôle est une opération très coûteuse car orientée
  nette amélioration du débit lors de la suppression du calcul du total de contrôle
  (de 67 % à 300 % dans des cas particuliers). Les algorithmes proposés [Coc87]
  nécessitent une implémentation “hardware” afin de tourner assez rapidement.
  D’autre part, comme le taux d’erreurs résiduel qui dépend de la qualité de
  service du réseau sera très réduit sur les futurs réseaux à haut débit, cette
  fonction risque de ne pas être très essentielle pour les paquets de voix ou
d’images à cause de la tolérance aux erreurs de ces applications.

Le service transport devrait être révisé dans le but de répondre mieux aux besoins
des applications. Ceci peut être effectué par la sélection des primitives néces-
saires pour le contrôle des flots de données. En ce qui concerne les données temps réel, on
pourrait par exemple simplifier le protocole en acceptant la perte occasionnelle de
paquets. Cette simplification est justifiée par la qualité du support de transmission
et par la tolérance aux pertes des applications échangeant des données temps réel.

On pourrait argumenter contre le changement du modèle de service de transport
en supposant qu’il suffit d’établir des connexions de transport avec des valeurs des
paramètres de qualité de service exprimant mieux les besoins des applications pour
echaque flot de données. Ceci n’est que partiellement vrai: les paramètres de qualité
de service définis dans le standard ne permettent pas de décrire tous les besoins
des applications. Par exemple il n’y a pas un moyen de mentionner la variance du
débit pour les applications ayant un débit variable. D’autre part, les paramètres de
qualité de service désignent des valeurs moyennes négociées à l’établissement de la
connexion. Les valeurs instantanées des délais et du débit ne sont pas nécessairement
egales à celles adoptées durant la phase d’établissement de la connexion de transport.
Le problème ne se limite donc pas au choix des bons paramètres de qualité de service,
mais aussi à la définition des fonctionnalités que devrait fournir la couche transport
et de la meilleure façon de fournir ces services.
3.6 Quel système de transport?

Les applications nouvelles ont des besoins variés et difficiles à être satisfaits sans une révision du service transport standard. Dans ce chapitre nous avons montré l’intérêt de la simplification du service fourni aux applications dans l’augmentation du débit de transmission. Le chapitre suivant sera consacré à la discussion des différentes approches pour la construction d’un système de transport haut débit: optimisation de l’architecture matérielle et du système d’exploitation, réglage de protocoles standards et/ou conception de nouveaux protocoles.
Chapitre 4

Le contrôle de transmission à haut débit

L’existence des réseaux à haut débit est une brique nécessaire mais non suffisante pour le développement d’applications haut débit. Il faudra optimiser en particulier la couche de transport afin d’éviter un ralentissement de la transmission. En effet, les protocoles de contrôle de transmission standards comme TCP [Pos81] et TP4 [Rec84a] ne montrent pas de bonnes performances même quand ils sont utilisés sur un réseau local [Dab87], [Hea89], [Mei87]. Dans le cas de TCP, les expérimentations effectuées dans [Ches87] et [Doe90] montrent que pour certaines implémentations le débit utile au niveau de l’application ne dépasse pas 3 Mb/s: la transmission entre deux utilisateurs puissants sur un Ethernet sera limitée par le protocole TCP en non par la vitesse de transmission sur le réseau. Cette limitation sera encore plus prononcée quand les réseaux à haut débit offrant des débits dépassant les 100 Mb/s seront disponibles. Les limitations des performances peuvent avoir plusieurs origines:

- La première cause de délai ne concerne pas le protocole de contrôle de transmission mais son environnement d’implémentation i.e. l’architecture du matériel et le système d’exploitation. La puissance du processeur et la façon de gérer les interruptions et les commutations de tâches (context switching) sont des facteurs déterminants dans les performances.

- Les choix d’implémentation et le réglage des paramètres du protocole ont aussi un impact important sur l’efficacité. En effet, deux implémentations différentes du même protocole peuvent avoir des performances qui varient d’un ordre de grandeur ou plus à cause des choix différents dans le mécanisme de mémoisation des données ou dans l’utilisation des temporisateurs [Svo89b]. D’autre part, les optimisations des algorithmes de contrôle par le réglage dynamique des paramètres tels que la taille de la fenêtre de transmission ou la valeur des temporisations permet d’éviter des retards inutiles et par suite d’améliorer le débit du protocole.

- Le complexité du service fourni par le protocole de transport détermine aussi
les performances de l’implémentation. Une autre façon d’optimiser, est de concevoir le protocole pour qu’il réponde aux besoins d’une application particulière. Le résultat est un protocole spécialisé de coût léger ou “lightweight protocol”. Ces protocoles à fonctionnalité réduite pourraient montrer des performances meilleures que les protocoles à “usage général” pour l’application pour laquelle ils ont été conçus [Svo89a].

4.1 Impact de l’environnement

Le coût du protocole de transport n’est pas la seule composante du délai introduit par la couche de contrôle de transmission. D’ailleurs, c’est loin d’être la composante la plus importante. Les analyses du coût d’un protocole standard comme TCP ont montré [Cla89] qu’une grande fraction du temps d’exécution du logiciel de communication est consacrée à des opérations qui ne sont pas spécifiques au protocole mais associées à l’environnement matériel et logiciel (la gestion des buffers et des temporisateurs, la copie des données, l’interface avec le système et avec les applications...). Les instructions spécifiques au protocole (le codage des paquets, l’exécution de la machine de protocole) ne consomment pas dans la plupart des cas plus que 20% de la charge CPU consacrée à l’application réseau. Cependant, la surcharge due à l’environnement est liée à la conception du protocole, et pourrait être diminuée par une conception appropriée du protocole qui réduit la copie des données, ou facilite l’analyse ou la génération de paquets [Cla89], [Sal89]. Toutefois, le coût de la surcharge de l’environnement reste prépondérant et nécessite l’utilisation d’un matériel et d’un système d’exploitation mieux adaptés.

4.1.1 Les architectures multi-processeurs

Le traitement d’un protocole de transport à des débits de l’ordre du Gbps nécessite des processeurs capables de servir un très grand nombre de paquets par seconde (le débit atteint par exemple 250000 paquets de 512 octets par seconde sur une ligne de 1 Gbps). Le traitement d’un paquet en temps réel doit donc se faire en 4 μs. Le nombre d’instructions nécessaires pour l’exécution du protocole du transport à la réception ou à l’envoi d’un paquet a été estimé dans l’ordre de 200-250 instructions pour le protocole TCP [Cla89]. Ceci nécessite un processeur de 50 MIPS dédié à cette tâche. Dans la plupart des cas, le coût élevé d’un tel processeur ne justifie pas

[1] Le “principe 20/80” supporté par les défenseurs de protocoles standards dit que 80% de la surcharge CPU est dû à des instructions indépendantes du protocole.
son utilisation en co-processeur réseau pour une station de travail à utilité générale.\footnote{Des processeurs 15 à 20 MIPS sont courants aujourd’hui, on pourrait aussi s’attendre à une diminution du coût d’un tel processeur.}

Les architectures multi-processeurs (par exemple à base de transputer) offrent la possibilité du partage de charge de traitement des paquets entre les différents processeurs. Dans la littérature, différentes architectures multi-processeurs ont été proposées [Zit89].

Parmi les propositions on trouve les architectures en \emph{pipeline} où plusieurs processeurs exécutent en série les fonctions du protocole qui dépendent les unes des autres. Chaque processeur effectue une partie du traitement d’un paquet et le passe au processeur suivant. Ce mécanisme ne nécessite pas une bufferisation intermédiaire des paquets permettant ainsi d’optimiser la vitesse de traitement.

D’autre architectures possibles utilisent des \emph{vecteurs} de processeurs où les fonctions indépendantes du protocole sont effectuées en parallèle par différents processeurs. Ce parallélisme peut être appliqué à des niveaux différents:

- La gestion de chaque connexion de transport pourrait être effectuée par un processeur particulier. Ce schéma est peut être facile à implémenter et mais n’apporte pas les gains désirés pour des connexions haut débit. C’est plutôt une solution générale pour l’optimisation de la couche transport.

- Le traitement parallèle peut être effectué au niveau des paquets d’une même connexion: chaque paquet est alloué à un processeur libre. La politique d’allocation des paquets aux processeurs pourrait être aléatoire (le premier processeur libre) ou déterministe (à tour de rôle). Toutefois, l’adoption d’une telle architecture soulève des problèmes de contention pour les ressources (mémoire, bus) et de séquencement des paquets traités à cause des variations des temps de traitement des paquets [Jai90], [Gia89].

- Plusieurs processeurs peuvent traiter en parallèle les différents champs d’un même paquet. Ce mécanisme n’est valable que pour un protocole spécialement conçu pour cela et ne peut donc pas être appliqué pour les protocoles comme TCP ou TP4 à cause de la dépendance entre les champs d’un même paquet [Ches87].

L’architecture multi-processeurs et la façon avec laquelle a été spécifié le protocole de communication déterminent les performances d’une implémentation parallèle de ce protocole. Il est par exemple nécessaire que tous les processeurs réseau ainsi que le processeur de la machine hôte aient accès à une copie partagée des paquets afin d’éviter des recopies qui pénalisent les performances.
4.1.2 Le système d’exploitation

Les optimisations de performances dues à l’utilisation d’une architecture matérielle multi-processeurs sont insuffisantes si on ne dispose pas d’un système d’exploitation adapté. En effet, les délais dû à l’ordonnancement de processus dans un système multi-tâches tel que Unix\(^3\) sont beaucoup plus importants que les temps d’exécution du protocole de transport. Les expérimentations que nous avons effectuées sur un réseau local de machines Unix ([Dab87] et [Huit88]) ont montré que le coût du protocole TCP est nettement inférieur à celui du système (voir aussi [Cla82]). Les paquets sont beaucoup plus retardés par l’attente de l’attribution du processeur que par le traitement. En effet le changement de contexte (context switching) nécessaire pour le service de l’application introduit des délais dû au système d’exploitation: l’algorithme d’ordonnancement de processus (“scheduler”) ne permet pas d’interrompre une tâche moins prioritaire en cours d’exécution qu’au bout d’une seconde i.e. quand les priorités sont réévaluées. En effet, le “scheduler” est appelé après:

- une interruption (réveil d’un processus),
- un top horloge (si besoin) ou
- un appel système.

Il examine alors la file des processus activables et attribue le processeur au plus prioritaire parmi eux. La priorité d’un processus est initialisée au réveil du processus et est mise à jour toutes les secondes en se basant sur la “consommation CPU”. Les tâches en mode privilégié ont la plus grande priorité, toutes les autres tâches sont retardées dans l’attente de la fin du traitement de ces tâches. Si cet algorithme d’ordonnancement semble “équitable”, il ne permet pas d’assurer l’attribution du processeur en temps utile à une tâche devenue prioritaire. Un processus \(a\) (voir figure 4.1 qui se réveille à l’instant \(t_1\) avec une priorité inférieure à celle du processus “servi” \(b\) devrait attendre la prochaine mise à jour des priorités (à \(t_3\) avant d’être servi, même si \(a\) est supposé devenir plus prioritaire que \(b\) à partir de \(t_2\).

Nous avons introduit dans le noyau Unix un mécanisme de priorité avec préemption permettant à une application prioritaire\(^4\) d’arrêter le service d’un autre processus utilisateur et d’appeler instantanément l’algorithme de réévaluation des priorités. Le temps maximum de fonctionnement en “mode prioritaire” (appelé fenêtre temps réel) est limité afin d’empêcher le monopole du processeur par les tâches réseaux.

\(^3\)Unix est une marque déposée des laboratoires Bell des AT&T.

\(^4\)par exemple une tâche réseau qui devrait traiter des paquets de voix ou d’images reçus selon une certaine cadence.
4.1. IMPACT DE L’ENVIRONNEMENT

Figure 4.1: Attente pour le “scheduling” d’un processus

Le réglage de la fenêtre temps réel permet de satisfaire les besoins des applications et d’éviter l’attente du “scheduler” dans la limite de la puissance maximale du processeur.

L’optimisation décrite ci-dessus permet d’exécuter une seule tâche en temps réel sur le système. Dans le cas où plusieurs applications réseaux tournent en parallèle sur la même machine, le changement de contexte entre ces applications est effectué selon l’algorithme standard d’ordonnancement de tâches Unix. On est donc loin d’avoir une solution générale pour un système d’exploitation temps réel. En effet, l’implémentation d’un algorithme d’ordonnacement basé sur des priorités dynamiques (la tâche ayant le deadline le plus proche d’abord) avec préemption est aussi importante que les bonnes performances matérielles obtenues par un système d’exploitation permettant d’effectuer le “fast context switching”. Alors que la conception d’un tel algorithme distribué est un sujet de recherche aujourd’hui (voir [LeL90]), une bonne partie des travaux sur l’architecture des systèmes a été focalisée sur l’optimisation des performances en introduisant le concept de processus de coût léger (lightweight processes) [McJ89], [Wat80], [Wat88]. Le principe de l’utilisation des processus de coût léger a été aussi proposé par des systèmes d’exploitation de recherches tels que THOTH [Cher79], V [Cher84a], [Cher84b] et par le système Mach ([Acc86], [Tev87], [Ras86]) afin de tirer profit du parallélisme sur les architectures
multi-processeurs.

Mach


L’intérêt d’avoir ces deux abstractions orthogonales réside dans le gain de performances obtenu pour les applications “serveurs” (par exemple chaque nouveau client peut être servi par un flot) et pour les applications parallèles. En particulier, une tâche qui représente une implémentation parallèle du protocole de transport pourra utiliser plusieurs flots pour servir des connexions différentes ou même effectuer des traitements en parallèle sur les paquets d’une même connexion. Les flots étant des objets de faible coût, leur utilisation permet de tirer profit du parallélisme avec une surcharge minimale sur le noyau. (Par exemple les mesures effectuées dans [Tev87] montrent que le coût d’une création/terminaison d’un flot est de 784 instructions sur Micro VAX II alors que le fork/wait3/exit d’un processus nécessite 7425 instructions)

Mach fournit aussi d’autres facilités: un nouveau mécanisme de mémoire virtuelle permettant le partage de mémoire entre les tâches en minimisant les copies (copie-sur-écriture) et une facilité de communication interprocessus basée sur les capabilités (ou messages de droit d’accès) qui, intégrée avec le mécanisme de mémoire virtuelle, permet d’effectuer le transfert de large blocs de données entre deux tâches locales (tout l’espace d’adressage d’une tâche) via la technique de copie-sur-écriture. L’IPC (InterProcess Communication) de Mach est extensible d’une façon transparente à travers le réseau par l’utilisation des port réseaux tout en préservant la protection des capabilités par l’utilisation d’un mécanisme de cryptage.

Les concepts introduits par le système d’exploitation Mach facilitent l’implémentation “haut débit” de la couche transmission des applications réseaux car il fournit des mécanismes efficaces de partage de ressources et un support mieux adapté pour les communications interprocessus. Mach est actuellement la base du standard OSF/1.

5Dans ce paragraphe, nous traduisons par flot le mot thread.
4.1.3 Les frontaux ou Implémentation en hardware

En ce qui concerne les couches basses (physique et MAC), leur optimisation est indépendante de l’OS car elles sont implémentées en hardware dans les interfaces réseaux.

La VMP NAB

La “NAB” (Network Adapter Board) est une carte implémentant le protocole de transport VMTP [Cher88a] sur la machine multi-processeurs VMP. Les points clefs de la conception de la carte sont la réduction du nombre d’interruptions, du transfert de données sur le bus système et des délais d’attente des paquets dans les files. Une architecture en pipeline permet d’effectuer le traitement des paquets (le calcul du total de contrôle, le cryptage, l’accès au réseau) à la vitesse de transmission. Les Vidéo RAM permettent un accès simultané sans contention à la mémoire, (pour l’émission/réception et pour le traitement des paquets) réduisant ainsi le délai total du traitement des paquets. Un processeur gère les tampons mémoires, la copie des données, et le traitement des paquets: ces fonctions complexes sont difficilement “siliconisables”. Un premier prototype de la NAB utilisant un M68020 a permis d’atteindre des débits de 100 Mbit/s sur des liaisons point à point [Kan88].

L’expérience NADIR

Dans le cadre du projet NADIR 6 ont été développés des protocoles de communications permettant aux machines (des Mini6 ayant une puissance de 0.5 MIPS) de mieux tirer profit des débits élevés sur les liaisons satellites (2 Mbit/s sur TELECOM1). Le protocole de transport et une partie du protocole réseau NADIR étaient implémentés dans les Mini6. Pour pallier à la non disponibilité de coupleurs haut débit sur Mini6, un frontal THOMTIT a été utilisé en guise de coupleur haut débit (900 Kb/s!). Même si les coupleurs permettaient l’échange de données à haut débit, les limitations apparaissaient au niveau du système d’exploitation du Mini6 et de l’interface Mini6-coupleur; en ce qui concerne l’OS, le coût élevé de la commutation de tâches (7.5 ms) a conduit à réécrire un transport monotâche avec un mécanisme d’entrées/sorties non bloquantes afin de conserver un pseudo-asynchronisme entre l’accès des données entre mémoire secondaire et la transmission. D’autre part, le Mini6 était lié au coupleur par une liaison semi-duplex sur laquelle transistent les

6Le projet pilote NADIR (81-86) avait pour but l’étude expérimentale des possibilités apportées par l’utilisation informatique des systèmes de communication par satellite.
données et les acquittements", ce qui provoquait des retournements très pénalisant en performance. Pour limiter le nombre de ces retournements, deux optimisations ont été implantées:

- Passer les données en blocs de 16Koctets entre le Mini6 et le frontal,
- Minimiser le nombre des acquittements en ne les émettant que sur erreur ou sur "time out".

L’expérience NADIR permet de tirer la conclusion suivante: l’approche simpliste qui consiste à déporter dans un frontal les fonctions du protocole contrôle de transmission afin de "décharger" le processeur central de ce travail doit faire face aux problèmes de l’interface entre le coupleur et l’OS de la machine hôte.

**Implémentation câblée**

Une implémentation du protocole de transport en hardware (comme proposée par les concepteurs du protocole XTP [Ches87]) devrait, en principe, apporter une amélioration de performances. En effet, les limitations dues à la puissance insuffisante du processeur, à l’accès mémoire, à la surcharge de la gestion des tampons et au formatage des paquets pourraient être dépassées avec une implémentation en hardware [Kri89]. Pourtant cette solution ne résoud pas le problème de l’interface avec le système d’exploitation qui représente à notre avis le goulot d’étranglement de l'implémentation de protocoles à haut débit.

### 4.1.4 Interfaces réseaux haut débit

Une alternative au déport des fonctions dans un processeur frontal consiste à optimiser l’interface réseau comme proposé dans [Ade87], [Alb88], [Dang87]. Dans le but de réduire la charge des processeurs, les interfaces réseaux doivent minimiser le nombre des interruptions au système d’exploitation. Une station de travail ne devrait pas être monopolisée par le traitement des paquets reçus ou à transmettre. L’interface réseau haut débit devrait donc inclure une fonction de segmentation et de réassemblage câblée qui groupe les paquets avant de les passer au système d’exploitation (et vice versa). Selon le modèle IEEE 802, cette fonction est accomplie par les protocoles de niveau MAC. En plus de la segmentation/réassemblage des trames, d’autres fonctions sont généralement déportées sur l’interface réseau:

---

7Par abus de langage, nous utilisons dans le reste de la thèse le mot "acquittement" pour signifier " accusé de réception", de même "acquitter" sera utilisé à la place de "accuser la réception de".
• Le calcul du total de contrôle,

• La copie des données en/de la mémoire durant les entrées/sorties (module DMA).

Nous avons participé à la conception d’un tel protocole de segmentation/réassemblage. Dans le but d’effectuer le groupement de paquets arrivant simultanément sur différents canaux virtuels ATM sur la même machine, nous avons proposé d’utiliser une combinaison d’un module DMA, de mémoires associatives et des FIFOs. L’architecture de l’interface est représentée dans la figure 4.2.

Chaque mini-paquet ATM porte dans l’entête un identificateur de canal virtuel (VCI). Nous avons proposé d’insérer dans la partie données un “pointeur d’adresse” qui désigne la position du mini-paquet dans le paquet. Lors de la réception d’un mini-paquet, le VCI et le pointeur d’adresse seront utilisés pour calculer l’adresse mémoire à laquelle le segment sera copié. Le calcul peut être accompli rapidement en utilisant un mémoire associative (CAM). Le segment sera directement sauvé à l’adresse correspondante (par le module DMA).

La délimitation de paquets est un problème critique dans les procédures de segmentation: il s’agit de s’assurer que la recomposition ne va pas fusionner des paquets successifs, même en cas de perte de segments. Une solution très élégante à ce problème est de transmettre les segments en ordre inverse, c’est à dire transmettre le dernier segment du paquet global dans le premier mini-paquet et terminer par le premier segment qui sera caractérisé par une valeur nulle du champ “pointeur d’adresse”. Dans ce cas, il n’y a pas besoin de dédier des champs spéciaux pour les délimiteurs de début de paquet ou de fin de paquet. La réception du mini-paquet “0” indiquera que le paquet global peut être passé aux couches supérieures. Si ce mini-paquet est perdu, les segments du prochain paquet remplaceront le contenu du tampon mémoire et ainsi le paquet sera reçu correctement. Pour plus de sécurité des champs spéciaux pourraient être insérés au début d’un paquet: en particulier un champ obligatoire longueur de paquet et un champ optionnel total de contrôle. Le calcul du total de contrôle peut être combiné avec la copie du paquet comme les deux opérations sont “orientées octet”. Dans le cas où les données sont altérées, la copie du paquet est suspendue.

Cette procédure devra être implémentée dans les interfaces des machines connectées directement à des réseaux ATM. Pour les machines connectées à des réseaux locaux FDDI cette fonction sera accomplie par les passerelles LAN/WAN. L’inconvénient de cette technique réside dans l’absence d’un mécanisme de retransmission de bas niveau: la perte d’un mini-paquet implique la perte d’un paquet.
Figure 4.2: Diagramme bloc de l’interface réseau haut débit
4.2  RÉGLAGE DE PROTOCOLES STANDARDS

<table>
<thead>
<tr>
<th></th>
<th>get timer</th>
<th>set timer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITIMER_REAL</td>
<td>120-140 µs</td>
<td>140 µs</td>
</tr>
<tr>
<td>ITIMER_VIRTUAL</td>
<td>120-150 µs</td>
<td>140 µs</td>
</tr>
</tbody>
</table>

Tableau 4.1: Coût des primitives timers fournies par Unix

4.1.5 Interface Protocole-Système

Les performances d’une implémentation du protocole de transport dépend de l’optimisation des fonctions systèmes les plus appelées.

La gestion des tampons mémoires à l’interface avec l’utilisateur est une fonction complexe qui peut limiter les performances si sa conception ne correspond pas aux besoins de l’application. D’autre part, ce mécanisme doit faciliter la construction des paquets en évitant les copies multiples des données à la traversée des couches de protocoles (en utilisant par exemple le chainage de données).

La gestion des temporisateurs est une opération de base dans tous les protocoles de transport (TP4 utilise 7 types différents de temporisateurs). Le système doit fournir un mécanisme efficace de mise et de remise de temporisateurs. En effet, le nombre de temporisateurs servis en même temps pourrait être très grand et par suite les pénalisations sur les performances très importantes. La table 4.1 contient des mesures sur le coût de l’utilisation des appels systèmes Unix gettimer et settimer sur un SUN3/60 (68020).

4.1.6 Environnement haut débit

La figure 4.3 résume les fonctions à optimiser afin de minimiser l’impact de l’environnement sur les performances du protocole de transport.

4.2 Réglage de protocoles standards

Les protocoles de contrôle de transmission “généraux” tels que TCP ou TP4 utilisent des mécanismes complexes pour gérer la connexion de transport, rectifier les erreurs de transmission et contrôler le flux sur des réseaux hétérogènes. En effet, la diversité des réseaux interconnectés (réseaux locaux haut débit, liaisons commutées,
Figure 4.3: Environnement d’implémentation de protocoles à haut débit.

canaux satellites à délai de propagation élevé, réseaux publics etc) nécessite la mise en œuvre d’algorithmes sophistiqués afin d’assurer les fonctionnalités requises pour le contrôle de la transmission de bout en bout. Ces protocoles peuvent être utilisés par un large spectre d’applications ayant des besoins différents; le prix à payer pour ces avantages (portabilité, service transport standard) est la limitation des performances du protocole. En effet, la plupart des implémentations de protocoles généraux ne permettent pas d’assurer la transmission des données à haut débit\(^8\) [Boud85]. Outre les limitations dûes à l’environnement (voir section 4.1), la complexité des mécanismes de contrôle de flux et des erreurs pénalisent souvent les performances [Huit84]. L’analyse de ces protocoles montre trois types de difficultés à résoudre si l’on veut améliorer les performances:

- Des problèmes intrinsèques dus à l’utilisation d’informations insuffisantes dans un environnement distribué (temporisateurs),

- Des points de blocage provenant d’une conception inadaptée des algorithmes de contrôle (acquittements, retransmissions),

- Un mauvais choix des paramètres de contrôle (fenêtre d’émission, taille des messages, taille des buffers dans les systèmes finaux) [Col85].

Le réglage de ces protocoles consiste à apporter les modifications nécessaires aux algorithmes utilisés et à choisir les bonnes valeurs des paramètres [Wat87].

### 4.2.1 Le contrôle d’erreurs

La transmission fiable nécessite la correction des erreurs causées par les paquets perdus, altérés, dupliqués ou reçus hors séquence. Pour cela, les protocoles de trans-

\(^8\)En contre exemple, Ultra Network Technologies a effectué une implémentation de TP4 tournant à 480 Mbit/s entre deux supercalculateurs.
port utilisent les accusés de réception avec un mécanisme de temporisateurs pour la détection de la perte de paquets. L’émetteur conserve une copie des paquets transmis jusqu’ils soient acquittés. Si l’acquittement d’un paquet n’est pas reçu au bout d’une temporisation ou “time-out” il y a une retransmission. Les algorithmes de retransmissions (“goback-n” et “selective repeat”) ont été analysés et leur performances comparées [Huit85]. La stratégie de retransmission ainsi que le choix de la valeur de la temporisation sont des facteurs déterminants dans les performances des protocoles.

**Les acquittements**

Les acquittements au niveau transport pénalisent les performances globales comme le service d’un acquittement est presque aussi coûteux que celui d’un paquet de données (la charge la plus grande provenant de l’ordonnancement de tâches dans l’OS plutôt que du traitement du paquet). Les protocoles standards (TP4, TCP) utilisent un mécanisme d’acquittements cumulatifs: l’accusé de réception du paquet “N” acquitte aussi tous les paquets précédents, ainsi le récepteur peut acquitter plusieurs paquets en un seul message. La stratégie de retransmission “goback-n” est aussi souvent utilisée: si l’acquittement du paquet “N” n’est pas reçu au bout du time-out, “N” et tous les paquets suivant sont retransmis, Ce mécanisme minimise la taille des tampons de mémoire à la réception, mais surcharge le réseau avec des retransmissions inutiles dans le cas d’erreur.

Dans TCP chaque octet a un numéro de séquence, les acquittements indiquent le numéro du prochain octet attendu. Dans le but de limiter le nombre des acquittements Clark a proposé dans [Cla82] de retarder dans certains cas le renvoi de l’acquittement d’un segment dans l’espoir d’en cumuler plusieurs. Pour éviter une retransmission inutile provoquée par un retard excessif, un temporisateur sera armé et un acquittement est renvoyé à l’expiration du time-out. Ce mécanisme n’est efficace que si l’acquittement est envoyé avant l’expiration du time-out (le service d’un temporisateur étant aussi coûteux que celui d’un acquittement).

Avec des débits de transmission très élevés et des réseaux à grande étendue géographique (par exemple un réseau ATM) le nombre d’octets “en transit” dans le réseau (égal au produit de débit et du délai de propagation) peut dépasser 10\(^6\). La perte d’un paquet peut avoir un effet catastrophique sur le débit à cause du mécanisme des acquittements cumulatifs. Jacobson et Braden ont proposé une extension du protocole TCP basée sur l’utilisation d’acquittements selectifs pour les réseaux “LFN” (“Long and Fate pipe Networks”) ayant un produit débit par délai de propagation important [Jaco88b] et [Jaco90]. En renvoyant de tels acquittements, le récepteur peut informer l’émetteur de tous les paquets correctement reçus, l’émetteur aura alors à retransmettre seulement les paquets perdus. Cette amélioration a été
aussi introduite dans plusieurs protocoles que NADIR, NETBLT, XTP et RDP [Par87]. Des simulations effectuées dans [Sal89] ont montré que l’ajout de ce mécanisme à TP4 apporterait un gain de performances de 20%. Brodd et Donnan avaient aussi proposé un mode de fonctionnement de la procédure HDLC: CPM ou (Check Point Mode) où seuls les numéros de trames à retransmettre sont indiqués dans les trames de synchronisation (CP-frame) transmises périodiquement [Bro83], [Bux88].

En bref, le groupement des acquittements et la retransmission selective permettent de réduire le nombre des acquittements et des retransmissions inutiles et par suite d’améliorer le débit global.

Les temporisateurs

Les protocoles de transport utilisent les temporisateurs pour détecter la perte de paquets ou la coupure de la connexion de transport. Le protocole TP4 emploie plusieurs temporisateurs pour la gestion de la connexion de transport, parmi lesquels: le délai de réexpédition de l’entité locale (T1), le délai d’inactivité (I) et le délai de réexpédition d’information de contrôle de fenêtre (W)...

Outre le coût de mise et de remise des temporisateurs – non négligeable si leur nombre est très grand – le problème essentiel est de trouver la “bonne” valeur de la temporisation. Cette difficulté est particulièrement prononcée pour le temporisateur de retransmission: une valeur trop petite provoque des retransmissions inutiles alors qu’une grande valeur retarde la retransmission des paquets perdus. Il faudrait donc trouver un algorithme qui permette d’ajuster dynamiquement la valeur de la temporisation afin de satisfaire deux buts incompatibles: la détection rapide des pertes de paquets et la réduction du nombre de fausses alarmes.

La dépendance de la valeur optimale de la temporisation du délai aller-retour entre l’émetteur et le récepteur (Round Trip Time ou RTT) pose un problème critique: le RTT dépend de plusieurs facteurs comme la charge du réseau, les routes suivis par les paquets, la longueur des paquets (si des liaisons à faible débit sont traversées) ce qui rend l’estimation de la valeur du RTT très difficile. D’autre part, une mesure exacte du RTT est impossible s’il y a eu des erreurs de transmission car si un acquittement est reçu après n retransmissions on ne sait pas quelle copie du paquet est acquittée. Ce problème pourra être contourné si le protocole identifie chaque copie du message retransmis, l’acquittement designera alors l’identificateur de la copie acquittée.

Après chaque nouvelle estimation du RTT, la valeur obtenue est filtrée pour atténuer l’effet des changements brusques et transitoires. La valeur filtrée est alors utilisée pour le calcul de la temporisation de retransmission (Retransmission Time Out ou RTO). Le gain du filtre (par exemple le paramètre α dans le cas de TCP)
determine la rapidité de réponse de l’algorithme adaptatif: Mills [Mil83] a proposé que l’adaptation soit plus "sensible" aux mauvaises nouvelles (i.e. le délai augmente) qu’aux bonnes nouvelles (i.e. le délai diminue), afin d’éviter de saturer le réseau par des retransmissions prématurées en cas de congestion. Une autre amélioration passe par une contre réaction exponentielle sur la valeur du RTO (similaire à l’algorithme de retransmission sur Ethernet) en cas de retransmissions successives (proposé dans [Jaco88a]). Elle permet de conserver la stabilité9 mais provoque des délais plus grands.

D’autre part, une bonne estimation du RTT moyen ne suffit pas à elle seule: l’introduction de la variance du RTT dans le calcul du RTO permettra aussi d’éviter des retransmissions inutiles. De toutes façons, le déphasage entre le temps d’estimation du RTT et le calcul du RTO pourrait rendre le problème encore plus difficile sur les réseaux étendus.

Les difficultés de l’utilisation de temporisateurs proviennent du fait qu’ils sont un moyen local pour estimer des événements extérieurs. Le manque d’informations extérieures sur la nature des erreurs (perte de paquets, machine en panne, ligne coupée, réseau partagé, etc..) pourra provoquer des retransmissions inutiles (à cause de l’expiration du RTO) dans beaucoup de situations [Zha86]:

- Le paquet n’a pas encore été transmis (interface réseau bloquée),
- une mauvaise estimation du RTT a donné un RTO inférieur à la valeur actuelle du RTT,
- le paquet a été reçu, mais l’acquittement perdu,
- la machine destination est en panne.

Pour pallier à ces difficultés intrinsèques dans l’utilisation des temporisateurs on pourrait:

- Utiliser des informations extérieures (par exemple des NACKs selectifs) de façon à ne pas compter sur les temporisateurs pour détecter les erreurs que quand les autres moyens échouent.
- Concevoir des algorithmes adaptés aux besoins des applications, comme celui du protocole NETBLT (voir 4.3.2)

9En théorie, pour une population infinie, même un “back off” exponentiel ne suffit pas pour assurer la stabilité (voir [Akl87]). En pratique, on arrive à conserver l’équilibre par cette contre réaction vu le nombre fini de paquets à transmettre [Jaco88a].
Il existe un algorithme utilisé par des implémentations de TP4 qui permet de réduire les délais de retransmissions dans le cas d'une surestimation du RTT; il s'agit de considérer les "doubles acquittements". Le protocole TP4 suggère d'émettre un ACK après chaque paquet reçu, même s'il est "hors séquence". Dans le cas "hors séquence", résultant en général de la perte d'un paquet, l'ACK répète le même numéro que l'ACK précédent. La détection d'un tel ACK correspondrait donc à une "perte de paquet", et on peut sans trop de risque répéter la paquet manquant, même si le RTT estimé n'est pas écoulé.

Certains protocoles utilisent les temporisateurs pour la gestion de la connexion comme le protocole Delta-t [Wat81], [Fle78], [Wat81]. Ceci permet de réduire l'échange de paquets de contrôle entre les deux systèmes finaux et par suite d'améliorer les performances globales. Ceci n'est qu'en partie vrai, car des valeurs majorées des temporisations doivent être choisies afin d'éviter les confusions dues à l'estimation des événements extérieurs. Toutefois, ce type de protocole est particulièrement adapté aux applications transactionnelles car il permet de minimiser le nombre de d'échanges entre le client et le serveur.

**Le total de contrôle**

Le but du total de contrôle est de détecter des erreurs éventuelles de transmission. C'est pourtant une opération très coûteuse car chaque octet du message devrait être traité. Plusieurs algorithmes ont été proposés pour améliorer l'efficacité du calcul du total de contrôle [SkI89], [Bra88] notamment en utilisant dans procédures orientées mots. La plupart des protocoles fournissent la possibilité de négocier la non utilisation de cette facilité afin d'optimiser les performances pour les applications qui tolèrent un certain pourcentage d'erreurs.

### 4.2.2 Le contrôle de flux

Le contrôle de flux au niveau transport assure essentiellement deux fonctions:

- Une adaptation de vitesse de transmission entre l'émetteur et le récepteur,
- La protection du réseau contre la congestion causée par un surnombre de paquets en transit.

La plupart des protocoles standards appliquent un contrôle basé sur le mécanisme de fenêtre de transmission, l'allocation de nouveaux crédits à l'émetteur étant liée à la disponibilité de tampons mémoires à la réception [Ber87], [Ger86]. Les limitations de l'efficacité d'un protocole de fenêtre proviennent d'une valeur insuffisante pour
“remplir le canal de transmission” haut débit. En effet, si la vitesse de transmission est très élevée, une petite fenêtre va imposer des périodes de silence dans l’attente de nouveaux crédits de la destination. Comme il faut au moins un délai d’aller-retour pour recevoir l’acquittement la taille de la fenêtre devrait donc vérifier:

\[ W > \frac{2D.C}{L} \]

où \( C \) est la capacité de la ligne en bps, \( L \) la taille des packets en bits et \( D \) le délai de transmission de bout en bout d’un paquet ou de son acquittement. L’augmentation de la taille de la fenêtre a été adoptée ou proposée par plusieurs concepteurs de protocoles [Bro83], [Bux88], [Jaco90], [Fox89] dans le but de permettre une meilleure utilisation du “canal” de transmission. Par ailleurs, cette augmentation devrait être accompagnée par un mécanisme de numérotation étendue afin d’éviter des erreurs de séquencement [McK89].

Cette amélioration est valide tant que la charge du réseau est faible, à condition que le récepteur dispose de ressources suffisantes (mémoires et processeur) pour traiter les paquets reçus. Toutefois, si plusieurs connexions de transport ayant chacune une large fenêtre partagent une liaison du réseau (par exemple un canal satellite) la lisaion sera vite saturée et les délais dans les passerelles risquent de déclencher des retransmissions inutiles, ce qui augmenterait de plus en plus les délais. Cette contre-réaction positive provoquerait une congestion si aucun contrôle n’est effectué. La solution de ce problème varie selon le niveau auquel on introduit le contrôle: une solution “réseau” tenterait d’obtenir des informations sur la liaison en congestion par le retour de messages de contrôle des passerelles intermédiaires [Fin89], [Nag84], alors qu’une solution “transport” utiliserait un mécanisme de fenêtre adaptative comme expliqué dans le paragraphe suivant.

**Adaptation de La fenêtre de transmission**

Le but d’un mécanisme d’adaptation de la fenêtre est de ne pas limiter la vitesse de transmission tant que la congestion pourrait être évitée, en augmentant la taille de la fenêtre. Dans le cas où les délais augmentent, une réduction de la taille de la fenêtre s’avère utile pour réduire le temps de réponse global.

Le problème est posé dès le démarrage d’une connexion de transport: si l’on permet une grande fenêtre de transmission à plusieurs connexions en même temps, l’absence du contrôle du débit de transmission “à l’intérieur” de la fenêtre, pourrait provoquer la congestion d’une ligne avant même la réception d’un acquittement. Pour pallier à ce problème il a été suggéré dans [Jaco88a] que la fenêtre soit ouverte graduellement, avec une augmentation exponentielle à la réception d’un acquittement, jusqu’à atteindre la valeur maximale de la fenêtre. Cette technique appelée
Figure 4.4: Calcul adaptatif du délai de transmission

*snow start* permettrait d’éviter la congestion à l’ouverture, mais pénalise les performances des connexions qui nécessitent un court échange de paquets.

Le contrôle de la taille de la fenêtre est aussi nécessaire durant toute la phase de transfert de données. Le principe de l’algorithme de contrôle pour une liaison satellite est expliqué dans la figure 4.4. Le débit effectif de la liaison, ainsi que les délais de bout en bout sont mesurés. Pour cela, on mesure le délai d’acquittement $D_a$ des paquets "tête de fenêtre" ($X_1, X_2, \ldots, X_i, X_{i+1}, \ldots$). $D_a$ a deux composantes: un délai de transmission $D_t$ (délai de propagation inclus) et un délai d’attente dans les files du réseau $D_q$. Pour deux intervalles successifs, nous pouvons faire l’hypothèse que:

$$D_t^i = D_t^{i+1} = D_t$$

c’est à dire que le délai de transmission reste stable. En observant que $D_q^{i+1}$ inclut les délais d’attente causés par l’envoi de $W^i - 1$ paquets (entre $X_i$ et $X_{i+1}$), le délai $D_a^{i+1}$ représente le temps nécessaire pour écouter $W^i$ paquets sur la liaison. Le même débit est atteint avec une fenêtre:

$$W = \frac{W_i*D_t}{D_a^{i+1}}$$
La valeur estimée de $D^i$ sera le minimum des délais d’acquittement mesurés:

$$D^{i+1}_t = \min\{D^i_t, D^{i+1}_a\}$$

La nouvelle fenêtre de transmission sera:

$$W^{i+2} = \begin{cases} 
  \frac{W^i D^{i+1}_t}{D^{i+1}_a} & \text{si } D^{i+1}_a > D^i_t \\
  W^i + 1 & \text{autrement}
\end{cases}$$

c’est à dire la taille de la fenêtre est augmentée de 1 si tous les paquets ont été éoulés sans attente et diminuée d’un facteur multiplicatif afin de réduire le temps de réponse dans le cas contraire. Une meilleure estimation du délai de transmission serait peut être nécessaire pour les réseaux datagrammes où les paquets peuvent emprunter des chemins différents (par exemple en faisant la moyenne du délai sur une minute). Dans son article, Jacobson a proposé une valeur fixe du facteur multiplicatif égal à $\left(\frac{1}{2}\right)$ [Jac88]. La valeur que nous avons proposée ci-dessus $\left(\frac{D^i_t}{D^{i+1}_a}\right)$ a l’avantage d’adapter la diminution de la taille de la fenêtre à la “gravité” de la congestion. Cet algorithme peut être introduit dans une implémentation du protocole TP4, ainsi que d’autres protocoles utilisant un mécanisme de fenêtre pour le contrôle du flux.

**Les tampons mémoires**

Les besoins haut débit imposent la transmission continue de l’information. La taille des tampons mémoires nécessaires à l’émetteur pour une retransmission éventuelle est proportionnelle au produit (débit×RTT). Cette taille atteint 5 Moctets pour un réseau à 1 Gbps et un délai de 40 ms. Ceci limite le nombre d’applications tournant en même temps sur un système final. Cette difficulté ne peut être résolue que par le changement du modèle de service en ce qui concerne la correction d’erreurs comme proposé dans la section 3.5.

### 4.3 Protocoles à coût léger

La recherche de nouveaux protocoles a principalement deux motivations: (1) les performances souvent “moyennes” des implémentations des protocoles standards, (2) la recherche de nouvelles fonctionnalités. Les concepteurs de ces nouveaux protocoles à coût léger ont choisi de réduire les fonctionnalités fournies par le protocole ou sa “portabilité” afin d’améliorer les performances [Dab89], [Sab89], [San90], [The82].

La complexité des protocoles standards nécessitait souvent une “lourde” implémentation au niveau utilisateur. Les simplifications des primitives de transport allant
jusqu’à la description des éléments de protocole par des bits figés dans l’entête des messages permet de faciliter l’implémentation voire même de micro-coder le protocole dans un chip VLSI.

D’autre part, la diminution des erreurs de transmission a permis de concevoir des protocoles faisant des “suppositions” sur le milieu: le protocole URPI suppose un service réseau de type “stream” i.e. pas d’octets reçus hors séquence [Fra89a], [Fra89b], XTP a une option (NOERR) pour ne pas effectuer le contrôle d’erreurs.

Il est parfois possible d’optimiser les différents coûts de transmissions (allocations de ressources, contrôle de transmission) pour une application particulière, alors qu’une telle optimisation serait très difficile à mettre en œuvre pour un protocole “à usage général”.

Toutefois, ces nouveaux protocoles devraient être considérés avec précautions pour les raisons suivantes:

- Un protocole très “léger” peut avoir des performances médiocres: la simplicité de conception n’étant pas nécessairement synonyme d’améliorations de performances.

- Le goulot d’étranglement pourrait ne pas être dans le protocole mais dans son environnement. Dans ce cas les protocoles à coût léger font face aux mêmes problèmes des protocoles standards.

- Les simplifications des primitives peuvent conduire à des états aléatoires. Par exemple, l’utilisation du “2-way handshake” implicite pour l’établissement d’une connexion réduit le nombre de paquets échangés mais n’assure pas la protection contre une ancienne duplication du paquet de demande de connexion.

- Le coût d’implémentation ajouté par un protocole de transport additionnel.

En plus, plusieurs mécanismes proposés pour ces protocoles peuvent être introduits dans les protocoles généraux. Pourtant, les gains de performances attendus pourraient justifier l’utilisation de tels protocoles. Le “task group” X3S3.3 de l’ANSI (couches réseau et transport) étudie plusieurs propositions de modifications des standards OSI dans le but d’intégrer les optimisations des nouveaux protocoles en ce qui concerne la transmission haut débit [X3S90a], [X3S90b], [X3S90c], [X3S90d], [X3S90e], [X3S90f]. XTP est l’un des protocoles les plus importants qui sont considérés dans cette étude.
4.3.1 XTP

Le protocole XTP (eXpress Transfer Protocol) a été conçu pour une nouvelle architecture matérielle pour l’implémentation de protocoles “la Machine de Protocoles” MP (ou Protocol Engine) [Xtp90], [San90], [Ches88b]. L’idée de base de la MP consiste à avoir un processeur spécialisé, implémenté dans un ou plusieurs circuits VLSI, afin de réduire la charge de traitement des paquets. Le protocole XTP représente un compromis entre simplicité de conception d’une part et fonctionnalité d’autre part dans le but de démontrer la possibilité d’atteindre des débits de transmission de 1 Gbps en l’implémentant au dessus de la MP.

La Machine de Protocole

Selon les concepteurs de la MP, les limitations de performances au niveau application sont dues en grande partie à l’architecture même de la couche transport et non seulement à des choix d’implémentation ou de paramètres de protocoles [Ches87]. La nouvelle architecture proposant une implémentation “câblée” d’un protocole de transport simplifié a été motivée par deux aspects des avancées technologiques:

- l’augmentation des débits de transmission au niveau des couches physique et MAC
- l’amélioration des caractéristiques des nouveaux réseaux en ce qui concerne les erreurs de transmission.

Le traitement des paquets reçus est généralement plus coûteux que l’envoi de paquets (génération d’interruption, tests, recherches)\(^\text{10}\). Pour cela, plusieurs protocoles ont proposé de simplifier le “récepteur” [Fra89a], [Xtp90]. Deux simplifications du récepteur ont été proposées par Chesson [Ches87]: (1) la minimisation du nombre des temporisateurs (un seul temporisateur qui sert pour la fermeture d’un contexte) et (2) la génération des messages de contrôle uniquement sur la demande de l’émetteur. La partie “récepteur” de la PM devrait donc supporter un débit de 1 Gbps entre la machine hôte et le réseau et inversement (donc avoir une bande passante de 2 GHz).

La MP devrait permettre de traiter les paquets en temps réel c’est à dire durant la transmission ou la réception de ces paquets (pour un débit de 1 Gbps et des paquets de 1280 octets, le temps de transmission est de 10\(\mu\)s). La résolution d’adresse,

\(^{10}\)Les résultats cités dans [Cla89] semblent contredire cette affirmation: Clark et al. ont estimé le coût de la réception d’un paquet à 186 instructions alors que le coût d’envoi est de 235. Toutefois, ces analyses ont porté sur les instructions spécifiques au protocole et n’incluent pas la surcharge due au traitement de l’interruption et à la recherche du TCB (Transmission Control Block).
le routage, la construction des paquets, le réassemblage des messages, le contrôle d’erreurs et le contrôle de flux ainsi que l’accès aux tampons de données devraient donc être accomplis durant le temps de réception du paquet. Une implémentation en plusieurs sous modules fonctionnant en parallèle et effectuant chacun une partie du traitement du paquet serait plus convenable. (figure 4.5(b)).

La conception du protocole

La complexité des fonctions d’un protocole de transport (codage des paquets, choix des options) rend plus difficile le câblage du protocole dans un circuit VLSI. Pourtant, le protocole le plus “léger” est loin d’être le plus utile ou le plus performant. XTP se propose d’accomplir les fonctions les plus utiles (contrôle de flux, contrôle d’erreurs, routage...) avec toutefois une implémentation simplifiée. Le service fourni par le protocole correspond en gros à celui des couches Réseau et Transport de l’architecture OSI, d’où le nom de couche de transfert11.

L’un des points clés de la conception de XTP est d’éviter la complexité en décrivant les éléments de protocole par des champs de bits de taille et d’emplacement fixes dans l’entête des paquets12 et en laissant le soin à l’application au dessus de XTP de “commander” la MP en positionnant les valeurs pour la plupart de ces champs. Chaque paquet XTP contient un entête et une “terminaison” (trailer) de tailles fixes qui permettent de simplifier le traitement du paquet par les différents modules en parallèle. Une autre simplification consiste à aligner le début des paquets (ainsi que les champs de données ou de contrôle et la terminaison) sur les frontières de mots de 32 ou 64 bits afin de minimiser la copie de données au niveau “octet”. Il s’agit là de faire un compromis entre l’augmentation de la longueur du paquet due au bourrage et la réduction du coût de copie de données: avec des débits de transmission élevés, la composante liée au traitement du paquet devient de plus en plus prépondérante.

XTP a été conçu de façon à pouvoir être décomposé en différents “modules” fonctionnant en parallèle ou en pipeline et effectuant le contrôle de flux, le contrôle d’erreurs, le calcul du total de contrôle13, la gestion de la connexion de transport, la translation d’adresses ou le contrôle du débit de transmission. Lors de la réception d’un paquet, la recherche du contexte correspondant à la connexion est effectué en même temps que la bufferisation des données dans des tampons de rétention dans l’attente de la décision d’accepter ou de rejeter le paquet. L’architecture matérielle de chaque

11Cette appellation a été introduite par les concepteurs du protocole GAM-T-103 [Gam87] pour désigner une fusion des couches 3 et 4 du modèle OSI.
12Dans TP4 les champs optionnels dans l’entête d’une TPDU entraînent une longueur variable et nécessitent des opérations “orientées octets” afin de décoder la TPDU.
13Le “checksum” est placé à la fin du paquet afin de permettre son calcul en cours de transmission sur le réseau.
4.3. PROTOCOLES À COÛT LÉGER

“sous système” comprend un chip VLSI sur lequel sont câblés un ou plusieurs “modules” et deux circuits logiques pour l’interface avec le réseau (couche MAC) d’une part et la machine hôte d’autre part. L’utilisation de deux circuits d’interface permettra d’éviter les commutations de contexte dans le cas de traitement simultané de paquets reçus et émis. Cette modularité dans la conception permettrait de bâtir des MP très performantes comprenant plusieurs “sous systèmes”, ainsi que des implémentations moins onéreuses (et moins rapides) dans lesquelles XTP est câblé sur un seul chip (figure 4.5(b) et (a)).

XTP utilise un bit dans la terminaison du paquet pour demander à la destination de renvoyer un paquet de contrôle décrivant son état (les intervalles d’erreurs pour une retransmission selective, la fenêtre allouée pour le contrôle de flux, et les paramètres du contrôle du débit de transmission...). Le paquet de contrôle pourrait être généré automatiquement par le récepteur dans le cas de la détection d’une erreur de transmission. Ceci fournit aux applications un moyen d’ajuster la fréquence des messages de contrôle selon leurs besoins. Le contrôle d’erreurs dans XTP est choisi pour chaque connexion à part parmi l’un des trois modes supportés (“go-back-n”, “selective repeat”, ou pas de retransmission) selon les caractéristiques du sous réseau et de l’application.

Le routage câblé de XTP devrait s’appliquer sur des architectures différentes de sous réseaux: réseaux ATM avec le modèle FPS, ou interconnexions de réseaux locaux. L’utilisation du champ “route” facilite l’adoption d’une stratégie de routage homogène sur les deux types de sous réseaux. Alors que sur un réseau local sa valeur est nulle, ce champ est utilisé par les passerelles comme une étiquette désignant le triplet <source, destination, porte de sortie>. Cette étiquette est semblable au VCI des réseaux ATM et facilite le routage en “temps réel”.

Dans le but de minimiser le temps pour effectuer le routage plusieurs connexions pourraient partager la même route vers une destination. Ceci suppose l’existence d’un protocole <hôte, passerelle> pour gérer l’allocation et la suppression de routes. D’autre part, chaque entité XTP transmet à l’entité homologue une clé qui désigne de façon unique le contexte XTP correspondant à chaque connexion. Ceci permet de réduire la recherche du contexte à une seule consultation de table. Toutefois, les informations sur le routage dans les nœuds XTP sont basées sur des tables statiques car le volume des informations à échanger entre les nœuds afin d’implémenter un protocole de mise à jour dynamique des tables de routage (comme RIP, GGP ou IS/IS) est énorme.

XTP utilise un mécanisme nouveau pour contrôler le flot de données: le contrôle de cadence (rate control). Pour cela, les passerelles XTP allouent une cadence maximale de transmission pour chaque nouvelle route\textsuperscript{14}. La source devra respecter cette limite

\textsuperscript{14}La combinaison des fonctionnalités des deux couches réseau et transport dans XTP permet à
Figure 4.5: L’architecture de la Machine de Protocole
en envoyant des groupes (burst) de $N$ paquets toutes les $(\frac{1}{s})$ secondes. $N$ ("burst size") et $s$ ("burst rate") étant déterminés par la cadence allouée.

Ce mécanisme permet de "découpler" le contrôle de flux et le contrôle d’erreurs: ces deux fonctions étant liées dans les protocoles à fenêtres où l’allocation de nouveaux crédits est conditionnée par la réception correcte de la fenêtre transmise. Cette dépendance est nuisible pour les applications temps réel sur des réseaux étendus ayant un délai de propagation élevé parce que la correction d’une erreur par une retransmission va entraîner le retard de tous les paquets suivants.

Le contrôle de cadence par route (et non par connexion) a plusieurs avantages:

- éviter la saturation d’une route par plusieurs connexions en même temps,
- permettre des débits de transmission plus élevés par l’allocation de nouvelles routes,
- les passerelles pourront participer à la régulation de la cadence,
- la gestion dans les passerelles sera basée sur les couples (source/destination) actifs et non sur le nombre des connexions ouvertes.

L’ouverture de la connexion est assurée par une procédure “2-way handshake” implicite: l’attribution d’une nouvelle référence (identificateur de connexion) est effectuée après l’échange de deux messages entre la source et la destination. Toutefois, la transmission des données utilisateur peut être entamée dès le premier paquet (FIRST) avant même la réception d’une réponse de la destination. Ceci réduit le délai d’établissement d’une connexion, et améliore le débit global au niveau de l’application surtout quand le produit bande passante par délai de bout en bout augmente d’une façon significative (sur les LFN). La simplification des procédures d’établissement et de libération de la connexion, ainsi que celles d’adressage et de routage permet d’une part de minimiser les échanges et d’autre part d’effectuer le traitement des paquets en temps réel.

Cependant, il n’y a pas actuellement dans XTP un moyen de détecter la réception d’une demande de connexion dupliquée après la fermeture de cette même connexion: les références ne sont pas gelées avant une nouvelle utilisation. XTP suppose en fait que le service de sous réseau préserve l’ordre des messages et ne les duplique pas. Cette condition n’est pas vérifiée sur tous les types de sous réseaux, et pourrait causer des confusions dues à la réception d’un ancien paquet dupliqué ou hors séquence [Wat89]. Pour pallier à ce problème, les protocoles Delta-t et VMTP (qui adoptent la même procédure que XTP pour l’ouverture de connexion) retiennent un système intermédiaire i.e. une passerelle de participer au contrôle de la congestion dans le réseau en limitant la cadence des émetteurs.
l’information sur l’état d’une connexion après sa terminaison afin de s’assurer que toutes les duplications éventuelles de paquets ont été détruites.

XTP a été conçu pour répondre aussi aux besoins des applications distribuées nécessitant des échanges de types RPC ou question/réponse. Pour cela un service de “datagramme fiable” est fourni.

En ce qui concerne le service de transmission fiable en multipoint (reliable multicast) XTP fournit un mécanisme de bufferisation pour la collecte et le traitement des acquittements reçus par les destinataires d’un paquet “multicast”. Cet algorithme appelé “bucket algorithm” permet d’effectuer une transmission multipoint presque fiable\(^ \text{15} \) pour toutes les stations actives c’est à dire participant à la connexion multicast. Toutefois, ce mécanisme n’assure pas la fiabilité de la transmission à tous les membres du groupe des stations destinataires d’un paquet multipoint: ceci nécessiterait un protocole de gestion de l’appartenance au groupe afin de connaître les membres actifs à tout moment, chose qui n’existe pas actuellement en XTP.

Les systèmes temps réel ont besoin de contrôler l’ordonnancement des événements afin de garantir les délais. XTP fournit un mécanisme de priorités statiques pour les paquets reçus ou émis. Chaque contexte est associé à un niveau de priorité (il y a 2\(^32 \) niveaux différents) hérité de l’application par la commande d’envoi de données et qui peut être modifié à chaque nouvel envoi de messages par l’application. Chaque paquet porte le niveau de priorité dans le champ SORT de l’entête. Cette valeur ne pourra pas être modifiée lors du transit du paquet sur le réseau. Ceci permet de traiter le paquet selon sa priorité à l’émission et à la réception en établissant un ordonnancement entre les différents contextes. Cependant, le protocole XTP ne contient pas tous les mécanismes nécessaires pour le développement des applications temps réel distribuées puisqu’il n’effectue pas l’ordonnancement des paquets d’un même contexte\(^ \text{16} \). En plus, XTP laisse à l’application le soin de modifier la priorité d’un contexte et ne fournit pas (dans la version 3.5) un mécanisme de modification dynamique du niveau de priorité d’un paquet en fonction du “deadline” correspondant à ce paquet. Enfin, le traitement du champ SORT étant optionnel, la sémantique de l’ordonnancement ne serait pas nécessairement propagée à travers le réseau.

Le protocole XTP est soumis à l’ANSI pour standardisation et fait actuellement l’objet de recherches pour sa validation par les techniques de description formelles (FDT) dans le cadre du projet ESPRIT OSI95 [Hig90]. Toutefois, de nombreuses modifications pourront être apportées à ce protocole avant qu’il soit adopté comme standard international.

\(^ \text{15} \) Il s’agit d’un compromis dépendant de la taille des buffers réservés pour la connexion multipoint. Si les participants à une telle communication sont nombreux, la gestion automatique des retransmissions sera très coûteuse en tampons mémoire.

\(^ \text{16} \) Ceci pourrait être introduit dans des versions ultérieures de XTP.
4.3.2 Autres travaux

**NETBLT**  (NETwork BLock Transfer) est un protocole à coût léger de la famille TCP-IP conçu au MIT et optimisé pour le transfert de données en masse sur des réseaux étendus ou locaux à haut débit [Cla87a], [Cla87b], [Lam87]. L'idée clef de la conception de NETBLT est la dissociation des mécanismes de *synchronisation des états* entre l'émetteur et le récepteur (contrôle d'erreurs) et du *contrôle de flux*. Un client NETBLT transmet des blocs complets de données sans l'attente de l'accusé de réception. À la fin de la transmission d'un bloc le récepteur demande la retransmission des paquets non ou mal reçus. Les avantages de cette technique sont les suivants:

- Minimiser les attentes sur un réseau ayant un délai de propagation important.
- Pallier aux difficultés intrinsèques de l'adaptation de la valeur des temporisations (4.2.1).
- L'utilisation de NACKs selectifs en fin de transmission permet un meilleur rendement.\(^\text{17}\)

La régulation de débit instantanné est effectuée par le mécanisme de contrôle de cadence (rate control). La temporisation de retransmission de blocs au récepteur\(^\text{18}\) sont basées sur le temps d'inter-arrivée des paquets et sur la taille des blocs négociée durant l'établissement de la connexion. Toutefois, le protocole ne spécifie pas comment la cadence d'envoi des paquets pourrait être adaptée à l'état du réseau afin d'éviter une congestion. En plus, les passerelles intermédiaires risquent de perturber la cadence de transmission des paquets. La mise en œuvre de ce protocole suppose l'existence d'un contrôle intelligent du réseau qui permettrait d'informer l'application quand la charge augmente afin qu'elle réduise son débit.

**VMTP**  (Versatile Message Transfer Protocol) est un protocole de transport spécialisé pour le support des applications transactionnelles (appels de procédures distantes ou RPC "Remote Procedure Call"), des applications multipoints, ainsi que des communications temps réel entre les entités d'un système distribué [Cher88a], [Cher86], [Cher89]. Dans VMTP la réponse à une requête sert aussi d'accusé de réception de cette requête, afin de minimiser le nombre de messages échangés pour une transaction. La confirmation de la réception réponse par le client est effectuée par la requête suivante, par l'envoi d'un accusé de réception explicite ou par

\(^\text{17}\)Ceci avait d'ailleurs été constaté dans les experimentations effectuées dans le cadre du projet NADIR [Nad85], [Az083].

\(^\text{18}\)Pas de temporisateurs à l'émetteur dans NETBLT.
l'expiration d'un temporisateurs. Il n'y a donc pas de phase d'établissement ou de libération de connexion. La gestion des associations est effectuée par l'utilisation de temporisateurs: les identificateurs $T$-stables ayant une seule signification pendant $T$ secondes seront gelés pendant la même durée ($T$) après leur utilisation. Ceci permet d'éliminer les duplications en évitant d'avoir un très large espace pour l'allocation des identificateurs. Un mécanisme de "2-way handshake" explicite pourrait aussi être utilisé dans certains cas afin d'accélérer la libération du "contexte" de la transaction. Le protocole fournit un mécanisme de retransmission selective pour les paquets perdus d'un message$^{19}$. Le contrôle de cadence remplace l'algorithme de fenêtre pour le contrôle de flux. Il existe trois implémentations du protocole VMTP: une au dessus du système V [Cher87], l'autre sous Unix [Nor89b], et la NAB [Kan88] une implémentation câblée.

4.4 Conclusion

Dans ce chapitre nous avons présenté des mécanismes pour l'amélioration des performances de la couche transport. L'optimisation du matériel, des systèmes d'exploitation ainsi que des interfaces réseaux devrait faciliter le développement des protocoles de transport dans un environnement haut débit. En ce qui concerne les protocoles "généraux", la détection des points de blocage et l'adoption des choix adéquats pour les valeurs des paramètres permet d'améliorer les performances de ces protocoles sans toutefois réduire les fonctionnalités fournies à l'application. D'autre part, les protocoles spécialisés à coût léger fournissent un service plus réduit à un coût inférieur. Cependant, la majorité des améliorations proposées par les protocoles à coût léger peuvent être appliquées aux protocoles généraux afin d'obtenir des protocoles très performants.

$^{19}$Un message est formé d'un ou de plusieurs paquets de 512 octets, jusqu'à 16 Koctets.
Chapitre 5

Contrôle des applications multimédias

Les protocoles présentés dans le chapitre précédent offrent chacun une solution pour le contrôle de transmission pour une application ou un environnement particuliers. Même les protocoles “généraux” optimisés ne suffisent pas pour répondre aux besoins des applications temps réel: les paramètres de qualité de service ne permettent pas de décrire complètement les besoins de ces applications (voir 3.3). Les applications multimédias nécessitent en plus la satisfaction de besoins variés ce qui rend encore plus difficile la conception d’un système de transport adapté à ce type d’applications. Dans ce chapitre nous présentons un protocole de transport adapté aux applications multimédias. Les points clés de la conception de ce protocole sont la fourniture d’un nouveau service de transport et la synchronisation des différents flots de données au niveau transport.

5.1 Le service de transport multimédia

Parmi les applications cités dans 3.1 nous nous intéressons à celles qui supportent la transmission simultanée de plusieurs flots de données ou médias. Une première façon pour satisfaire les besoins des différents flots de données, est que l’application utilise un protocole de transport spécialisé pour chaque flot. Cette approche a l’inconvénient d’aboutir à une implémentation très coûteuse de la couche transport. En plus, la contrainte de synchronisation des différents flots ne serait réalisable que par des mécanismes appropriés de la couche application. Nous optons donc pour l’idée d’avoir un seul protocole de transport multimédia. Ce protocole devrait permettre la transmission efficace de chaque type de données d’une application multimédia. Néanmoins, aucun protocole ne pourrait à notre avis satisfaire tous les types d’applications. Notre objectif n’est donc pas de définir LE protocole de transport, mais un protocole de transport pour les applications multimédias.

L’architecture en couches du modèle de référence OSI a pour but de permettre l’interopérabilité des applications distribuées dans un environnement hétérogène. Dans le but d’uniformiser l’interface entre les couches hautes (session, présenta-
tion et application) concernées par le traitement des données et les couches basses (jusqu’à la couche transport) concernées par la transmission des données, un seul service transport orienté connexion a été standardisé par l’ISO. Afin de fournir ce service au dessus de réseaux de nature différentes cinq classes du protocole de transport ont été définies :

- La classe 0 correspond à un protocole de transport presque “vide” utilisé au dessus des réseaux X.25 afin de fournir un mécanisme de sous adressage par le T-sel ou “transport selector”.

- Les classes 1, 2 et 3 sont des variantes plus complexes pour les réseaux X.25 onéreux ou ayant un taux inacceptable d’incidents signalés. La classe 2 fournit les moyens de multiplexer plusieurs connexions de transport sur une connexion de réseau, la classe 1 permet une reprise sur erreur du réseau, alors que la classe 3 présente les caractéristiques des classes 1 et 2 regroupées.

- La classe 4 inclue en plus des primitives de la classe 3 la détection et la reprise sur des erreurs de transmission. TP4 a été conçu pour les réseaux datagrammes qui permettent une interconnexion plus fiable des réseaux mais qui ont l’inconvénient de dupliquer, perdre, altérer ou remettre hors séquences des paquets.

Le choix d’une classe (0 à 4) ne correspond donc pas au choix d’une fonctionnalité particulière fournie aux applications mais aux caractéristiques du service réseau sous-jacent. Selon la philosophie de l’OSI une application qui nécessite l’échange simultané d’informations de nature différentes sur un même réseau devra établir une connexion de transport particulière avec des paramètres de qualité de service différents pour chaque type de données. Il n’y a pas un moyen de négocier la non utilisation des primitives telles que le contrôle de flux ou le contrôle d’erreurs pour tel ou tel type de données.\footnote{Il est possible de négocier la non-utilisation du mécanisme du total de contrôle, mais ça sera pour toutes les TPDUs transmises sur la connexion de transport correspondante.}

Nous proposons l’extension du service transport standard de l’OSI afin de supporter la transmission d’un nouveau type de données: le flot de \textit{données temps réel} ou “Real Time Data”. Le protocole de transport fournissant ce service utilisera les primitives de transport correspondantes à la nature du flot de données :

- Les données standards (ou DT)
- Les données exprès (ou ED)
- Les données temps réel (ou RTD)
En ce qui concerne les données standards ou exprès le contrôle de flux par le mécanisme de fenêtre adaptative, ainsi que le contrôle d’erreur avec la mise à jour dynamique de la temporisation de retransmission seront toujours appliqués. Pour les données ED l’utilisation de files indépendantes dans les systèmes finaux pour la bufferisation des données au niveau transport permet de leur attribuer une priorité plus grande. En ce qui concerne les autres aspects du service transport, les primitives appliquées aux TPDU DT et ED sont les mêmes. Pour cela nous allons citer dans la suite seulement les types DT et RTD.

Pour les données temps réel, l’hypothèse d’accepter un certain pourcentage d’erreurs permet de ne plus effectuer des retransmissions au niveau transport (en supposant le réseau très fiable). Un mécanisme de numérotation sera utilisé pour détecter les erreurs de transmission; un tampon de réception servira à égaliser les variations des délais de transmission et de corriger d’éventuelles (très rares) réceptions de cellules hors séquence; la négociation de la cadence de transmission (rate control) éliminera le besoin du contrôle de flux. Les acquittements au niveau transport seront remplacés par un mécanisme de contrôle partial dont le but est d’assurer un “acquittement global” des données temps réel, ou en d’autres termes un suivi de la qualité de service indiquant que l’application accepte toujours la qualité de service offerte par le réseau jusqu’au numéro de séquence précisé.

Un seul flot RTD suffit pour supporter tous les types de données temps réel: les images et le son vidéo, les images fixes, les paquets de voix numérisée, etc. Les besoins particuliers de chacune de ces applications seront exprimés par les paramètres de qualité de service.

D’autre part, même les applications “temps réel” nécessitent un flot de données sécurisées: les paquets effectuant le contrôle partiel de la transmission d’images devraient être protégés par les retransmissions.

Un service de transport assurant le transfert de données avec les trois types (DT, ED, RTD) répond mieux aux besoins d’applications multimédias et temps réel. La nécessité de multiplexer ces flots sur une même association au niveau application pose le problème du choix de la couche au niveau de laquelle le multiplexage devrait être accompli. Ce problème sera analysé avec plus de détail dans la section suivante.

5.2 La synchronisation des flots de données

Le transfert de données pour les applications multimédias interactives sur un réseau nécessite, outre la satisfaction des contraintes exprimées par les paramètres de qualité de service pour chacun des flots de données, un mécanisme assurant la gestion simultanée de ces différents flots. Citons deux exemples de telles applications:
une conférence multimédia où les interlocuteurs s'échangent des images, de
la voix et des données. Ces flots devraient être transmis à tous les autres
participants à la conférence d'une façon synchrone c'est à dire en effectuant
simultanément l'affichage des images et le décodage des paquets de voix ou
l'impression d'un texte de commentaire correspondant à l'image.

La transmission simultanée de fenêtres d'images et de texte par le protocole X-
windows [Sche86], pourrait présenter les mêmes contraintes de synchronisme,
Selon la relation entre le contenu des deux fenêtres.

La problème de la synchronisation des différents flots de données pourrait avoir
plusieurs solutions selon le niveau choisi pour "multiplexer" les flots de données: application, session, réseau ou transport (voir figure 5.1). Le multiplexage est donc
effectué afin de préserver l'ordre relatif des composantes d'une association multimé-
dia. Les deux cas extrêmes étant de multiplexer à tous les niveaux de la hiérarchie
(c'est à dire depuis la couche application) ou de transmettre les flots de données sur
des réseaux différents. (voir figure 5.2)

5.2.1 Présentation de données multimédias

Le multiplexage des flots de données au niveau application signifie qu'ils seront
transmis comme étant un "document" multimédia unique, Le problème de trans-
mission des différents flots se ramène donc à la définition d'une architecture et d'un
format d'échange de documents. Des normes ISO (ODA/ODIF) [ISO88b] et des
solutions constructeurs (DCA/DIA d'IBM et DHA/DPA de Bull) plus ou moins
conformes aux normes ISO ont été développées. En fournissant des règles de struc-
turation de documents de façon compréhensible par des équipements de fournisseurs
différents, la norme ODA permet de "préparer" le document à la transmission. Le
format d'échange de documents (ODIF) est alors représenté par une séquence de
descripteurs figurant les attributs d'un profil de document et d'unités de textes
représentant les éléments graphiques d'une portion du contenu. La transmission de
ces IDEs (pour Interchange Data Elements) pourrait être effectuée en série (sur un
même support de transport) ou en parallèle (sur différents supports). Dans tous les
cas, l'ordre de transmission est spécifié par le standard; la synchronisation entre les
différents flots est automatique. D'autre part, le mécanisme standard des points de
synchronisation au niveau de la couche session permettra de gérer la dynamique de
cadais des échanges de données.

L'approche ODA/ODIF élimine donc le besoin de synchroniser les flots de données
par l'adoption d'un ordre préétabli dans la transmission des IDEs. Cependant, il n'y
a aucun moyen de spécifier des contraintes temporelles sur l'échange des données: les
délais variables sont introduits d'une part par les procédures de codage et de
Figure 5.1: Multiplexage des flots au niveau de la couche N = 7, 5, 4, ou 3.

Figure 5.2: Transmission des flots de données sans multiplexage.
décodage et d’autre part par les temps de réponse imprévisibles sur les associations d’application. D’autre part, la selection du système de transport montre un autre aspect du problème: si le système est supposé fournir le service standard de l’OSI, il pourrait ne pas assurer les contraintes temps réel d’une transmission interactive; l’utilisation d’un protocole spécialisé pour chaque flot de données pourrait suffire pour répondre aux besoins spécifiques de ce type de données, mais le problème de la synchronisation reste posé.

ODA/ODIF présente alors une solution ouverte pour l’échange de “documents” multimédias statiques. Les applications multimédias interactives ont besoin d’un mécanisme plus efficace pour la synchronisation.

D’autre part, les mécanismes de “synchronisation” intégrés aux principaux éléments de service de la couche application (TP-SE, ROSE, CCR) concernent la gestion de la dynamique de l’application distribuée (comme le 2-phase commit) et ne correspondent pas au problème traité: la synchronisation de plusieurs flots de données entre deux systèmes finaux.

5.2.2 La synchronisation session

La solution la plus “OSI” est d’effectuer la synchronisation seulement au niveau session.² Cette solution consisterait à établir sur une même association de session plusieurs connexions de transport “typées” par les données à transmettre et à effectuer la synchronisation de chaque connexion à part. Si l’on voulait prendre en compte les améliorations proposées dans la section 3.5 c’est à dire l’utilisation du transport typé chaque connexion devrait fournir un service de transport correspondant aux besoins du flot de données supporté: DT ou RTD. Les connexions transport seraient associées à des canaux réseaux différents afin de profiter d’une utilisation efficace des ressources tout en séparant les flots. Une extension du protocole de session serait alors nécessaire pour gérer ces connexions multiples.

Toutefois, cette solution risque de ne pas répondre aux besoins de synchronisme des applications interactives à cause des délais variables et imprévisibles sur chacune des connexions transport. D’autre part, la connexion fournissant un service de transfert de données temps réel n’étant pas sécurisée par les retransmissions, le protocole de session ne pourrait même pas garantir la terminaison d’une opération de synchronisation (par exemple une connexion de données attendant indéfiniment une trame d’image perdue sur la connexion RTD).

²Des débats techniques ont même eu lieu au sein du groupe de travail WG6 de l’ISO lors de l’étude du traitement transactionnel (TP pour Transaction Processing) à cause des ressemblances avec le protocole de session [Ghe90].
5.2.3 Le multiplexage au niveau liaison

Si le multiplexage des flots de données sur une même connexion d’une couche de niveau inférieur permet de réaliser plus facilement leur synchronisation, il pour-rait en revanche dégrader les performances globales de la transmission par l’effet d'interférence entre les différents connexions multiplexées, ou par les délais vari-
able introduits lors du multiplexage/démultiplexage.

Le problème de l’interférence complique la garantie de la qualité de service de chacune des connexions de niveau supérieur [Fel90] surtout quand une application ne respecte pas les paramètres de qualité de service négociés: il n’y a aucun moyen de contrôler le flot de données concerné si le multiplexage est effectué avant l’admission dans le réseau (c’est à dire s’il est effectué dans les systèmes finaux). L’adoption d’un mécanisme de priorités ne résoud pas le problème s’il n’y a pas un moyen de le propager jusqu’aux couches les plus basses, i.e. de “typer” les unités de données pour conserver la qualité de service, ce qui nécessiterait des opérations “par cellule” qui seraient très coûteuses dans un réseau ATM.

Pour ces raisons, il a été proposé dans [Ten89] d’effectuer le multiplexage des associ-ations “application” seulement au niveau de la couche liaison sur des réseaux ATM. Cela revient à la solution de la figure 5.2 si l’on considère un réseau ATM comme une “intégration” de réseaux chacun adapté à supporter un type de traffic. Chaque association nécessite alors une instanciation de la pile de protocoles, gérée indépendamment par le système par l’utilisation des “threads” de contrôle.

Cette solution permettrait une utilisation efficace des ressources du réseau, ainsi qu’une réduction des délais de transmission et de leurs variations. Cependant, elle ne prévoit aucun mécanisme pour la synchronisation des différentes associations: celle ci devrait être assurée “hors bande” par l’établissement d’une connexion entre les entités de gestion de la couche LLC.

5.2.4 La synchronisation transport

La synchronisation au niveau transport nous semble un compromis convenable. D’une part, le marquage du séquencement temporel des unités des données des différents flots sera plus facile à la transmission. En effet, au niveau des couches hautes il n’y a pas d’informations précises et garanties sur les délais à travers le réseau alors que ces informations pourront être accessibles au niveau de la couche transport, ce qui favorise la solution “transport” par rapport à la solution “session” ou “application”.

D’autre part, l’absence d’une synchronisation “in band” dans le cas d’un multi-
plexage au niveau réseau nécessite un contrôle “hors bande” qui serait difficile à
appliquer. La solution "transport" permet un ordonnancement implicite des unités de données à la transmission et à la réception pourvu que le service réseau soit comme décrit dans le paragraphe (Service réseau) suivant.

Nous proposons donc d'enrichir le service de transport standard de l'OSI pour supporter le mélange des flots de données normales et temps réel et pour faciliter leur synchronisation.

Selon cette approche, les différentes "sources" de traffic seront considérées comme étant des "utilisateurs" du service transport, qu'elles soient des applications multimédias échangeant des données entre systèmes finaux ou des dispositifs générant un traffic temps réel (une caméra vidéo ou une prise de voix paquetisée). Dans tous les cas, le protocole de transport fournira les mécanismes qui permettront à l'application de synchroniser les deux flots. Une seule connexion de transport sera utilisée pour véhiculer les différents flots de données. Ce choix sera justifié dans la section 5.3.2.

**Service réseau**

Le protocole de transport proposé dans la suite suppose que le service réseau sus- jacent soit du type ATM; grâce à l'efficacité du multiplexage et de la commutation des cellules, la technique ATM permet d'établir des canaux virtuels de coût léger entre les systèmes finaux. Les cellules seront transmises avec un pourcentage d'erreur très faible. On peut surtout supposer que sur le réseau ATM le taux d'erreur dûs à des duplications de cellules ou à une réception hors séquence d'une cellule est négligeable. Selon le modèle adopté, les délais de transmission des cellules peuvent fluctuer sans pourtant causer la réception de cellules hors séquence.

Du moment où le multiplexage des flots de données sera effectué au niveau transport, une connexion transport devrait être associée à une connexion ATM. L'approche qui consiste à écarter la connexion transport sur plusieurs connexions ATM afin de profiter du partage efficace des ressources et d'assurer la qualité de service pour les différents flots ferait perdre à la solution transport son avantage essentiel: l'ordonnancement temporel des unités de données transmises. On suppose donc dans la suite qu'**une connexion transport sera associée à une connexion ATM**. Selon cette approche, il sera plus compliqué d'exprimer les besoins de l'application car ceci devrait être fait d'une façon collective (voir section 5.3.2) pour tous les flots de données. Toutefois, cette difficulté existe aussi dans le cas de connexions de transport séparées pour les flots de données parce que l'application multimédia nécessite un mécanisme pour effectuer la demande des connexions d'une façon atomique.
5.3 Le protocole TPM

Le protocole que nous proposons [Da91] est une variante de la classe 4 du protocole standard de l’ISO. Deux idées simples ont servi de base pour la conception de ce protocole:

- Le typage des TPDUs (DT et RTD), et l’application des primitives du service de transport selon le type des TPDUs.
- Fournir au niveau transport les mécanismes pour effectuer la synchronisation des flots de données.

Contrairement aux classes 0 à 4, le protocole TPM fournit un nouveau service de transport selon le type des données à transmettre: le service transport standard défini dans [Re84b] pour les données DT et ED, et le service de transport de données temps réel pour les données RTD. Pour cela, TPM ne doit pas être considéré comme une nouvelle classe du transport car selon la terminologie OSI toutes les classes fournissent le même service. Cette approche viole le modèle de référence OSI par le non respect de la relation service-protocole: un seul protocole est utilisé pour fournir “deux” services de transport différents (voir la figure 5.3).

![Diagramme de l'architecture du protocole TPM](image)

Figure 5.3: Le service de transport “type”

D’une part, ce protocole intègre les optimisations proposées dans le chapitre précédent pour les protocoles standards (l’adaptation de la temporisation de retransmission et de la valeur de la fenêtre) afin de fournir un service standard optimisé, et d’autre part il fournit un nouveau service pour le transfert de données temps réel.
5.3.1 Le service de transport typé

Les trois phases d’une connexion de transport telles que décrites dans [Rec84b] sont conservées: la phase d’établissement de connexion, la phase de transfert de données et la phase de libération de connexion. Cependant, des paramètres et même des primitives seront modifiées afin de fournir le nouveau service. Le tableau 5.4 donne une liste des primitives du transport typé.

Durant l’établissement de la connexion transport, l’utilisateur du service transport appelant précise qu’il s’agit d’une connexion multimédia par le paramètre option “RTD” dans la primitive de demande de connexion de transport. Le protocole traduira cette demande par l’envoi d’une TPDU de demande de connexion (CR) spécifiant le service typé comme “service préféré”. Dans le cas où le service de transport typé n’est pas supporté par l’entité transport appelée, la TPDU CC correspondante proposera une classe de repli (de 0 à 4). Si c’est la classe 4, il appartient à l’entité de transport de décider de la continuation ou de la rupture de la connexion, en se basant sur les informations fournies par l’utilisateur du service de transport appelant; sinon l’entité de transport appelante émet une demande de déconnexion de transport afin de libérer la connexion. Si l’entité appelée supporte TPM, la décision d’accepter ou de refuser la connexion devrait être basée sur les paramètres de qualité de service.

Durant la phase de transfert de données le service de transport typé fournit, en plus des primitives de transfert de données normales et exprès définies dans le standard, une primitive de transfert de données temps réel. Il existe une différence fondamentale entre les primitives de transfert de données exprès et de données temps réel:

- Le transfert de données exprès est soumis aux mêmes contrôles que les données normales, un niveau de priorité est pourtant garanti par l’adoption d’un contrôle de flux séparé de celui des données normales. Les données ED pourront être remises à un moment où l’utilisateur du service de transport n’accepte pas de données normales. Le même service est donc fourni séparément\(^3\) pour les deux flux de données avec une priorité pour les données exprès.

- Les données temps réel seront soumises au contrôle partiel, c’est à dire sans le contrôle d’erreurs ni le contrôle de flux. La relation entre les deux flux DT et RTD varie selon le mode de synchronisation adopté, séquence relative ou séquence absolue, comme décrit dans 5.3.2. Deux services de transport seront appliqués aux flux DT et RTD intégrés sur la même connexion.

Le service de transport typé correspond mieux aux besoins des applications multi-

\(^3\)Les numérotations de TPDUs DT et ED sont indépendents.
<table>
<thead>
<tr>
<th>Phase</th>
<th>Service</th>
<th>Primitves</th>
<th>Paramètres</th>
</tr>
</thead>
<tbody>
<tr>
<td>Etablissement de connexion de transport</td>
<td>Etablissement de connexion de transport</td>
<td>DEMANDE DE CONNEXION DE TRANSPORT&lt;br&gt;INDICATION DE CONNEXION DE TRANSPORT&lt;br&gt;RÉPONSE À UNE DEMANDE DE CONNEXION DE TRANSPORT&lt;br&gt;CONFIRMATION DE CONNEXION DE TRANSPORT</td>
<td>(Adr. entité appelée, adr. entité appelante, option &quot;ED&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfert de données</td>
<td>Transfert de données normales</td>
<td>DEMANDE DE TRANSFERT DE DONNÉES DE TRANSPORT&lt;br&gt;INDICATION DE TRANSFERT DE DONNÉES DE TRANSPORT</td>
<td>(Données utilisateur du service de transport) (=)</td>
</tr>
<tr>
<td></td>
<td>Transfert de données exprès</td>
<td>DEMANDE DE TRANSFERT DE DONNÉES ED DE TRANSPORT&lt;br&gt;INDICATION DE TRANSFERT DE DONNÉES ED DE TRANSPORT</td>
<td>(Données utilisateur du service de transport) (=)</td>
</tr>
<tr>
<td></td>
<td>Transfert de données temps réel</td>
<td>DEMANDE DE TRANSFERT DE DONNÉES RTD DE TRANSPORT&lt;br&gt;INDICATION DE TRANSFERT DE DONNÉES RTD DE TRANSPORT</td>
<td>(Données utilisateur du service de transport) (=)</td>
</tr>
<tr>
<td>Libération de connexion de transport</td>
<td>Libération de connexion de transport</td>
<td>DEMANDE DE DÉCONNEXION DE TRANSPORT&lt;br&gt;INDICATION DE DÉCONNEXION DE TRANSPORT</td>
<td>(Données utilisateur du service de transport) (Cause de la déconnexion, données utilisateur du service de transport)</td>
</tr>
</tbody>
</table>

Figure 5.4: Les primitives du transport typé
médias qu’un service de transport standard ayant un système de priorité entre les différents flots de données (par exemple par l’utilisation de files multiples). En effet, le changement du modèle de service afin de réduire les contrôles est plus bénéfique que d’effectuer les contrôles (non nécessaires) avec une priorité plus grande.

L’entité transport dispose des paramètres de qualité de service négociés durant la phase d’établissement de connexion et d’un modèle du temps du réseau, dérivé des hypothèses sur le service réseau. En utilisant ces informations, l’entité transport traite les données utilisateur correspondant à chacune des primitives (DT et RTD) en effectuant la segmentation éventuelle de longues TSDUs ainsi que la numérotation et la transmission des TPDUs.

Les primitives de la phase de libération de connexion sont identiques à celles définies dans le standard.

5.3.2 La synchronisation de flots multiples

La représentation du flux de données temps réel par un nouveau type de TPDUs signifie implicitement l’utilisation d’une seule connexion de transport pour véhiculer les deux flux de données DT et RTD. A priori, il n’est pas évident que l’utilisation d’une seule connexion de transport pour l’envoi des flots DT et RTD soit la meilleure solution. On pourrait en effet argumenter pour l’envoi des différents flots de données sur des connexions ATM séparées : ceci permet de mieux définir les paramètres de qualité de service transport pour chaque connexion. Comme l’éclatement au niveau réseau est à exclure, cette solution revient à établir des connexions de transport séparées pour les différents flots. Il s’agit donc d’analyser le coût de la synchronisation sans et avec le multiplexage au niveau transport afin de justifier le choix de l’utilisation d’une ou de plusieurs connexions de transport pour véhiculer les différents flots de données.

Les connexions de transport multiples

Deux procédés sont possibles pour effectuer la synchronisation de connexions multiples en évitant le multiplexage au niveau transport : la synchronisation par typage de message ou par canal de synchronisation.

La première approche consiste à transmettre sur chaque connexion des marques de synchronisation comme proposé dans [Sal89]. Ces marques serviront de délimiteurs des parties à synchroniser dans les flots. Elles sont transmises simultanément sur les différentes connexions lors de l’émission d’une demande de synchronisation par l’utilisateur du service de transport, et serviront à resynchroniser les flux à l’arrivée avant le passage à l’application. Cette approche est simple à implémenter mais elle
a l’inconvénient de ne pas permettre de tirer effectivement profit de l’absence du multiplexage des flots au niveau transport; les marques de synchronisation sont des messages typés qui nécessitent un service de transport sécurisé et qui seront multiplexés avec le flot temps réel; d’autre part, la nécessité d’insérer ces marques dans les flots de données empêche le raccordement direct de sources extérieures (comme les caméras ou les sources vocales) au système de transport.

L’autre approche consiste à transmettre les informations relatives aux séquencement des flots de données (par exemple la numérotation relative des données dans les flots, ou leur ordre d’affichage) sur un canal de synchronisation séparé afin d’éviter la modification du contenu des flots par l’insertion des marques de synchronisation. Cette approche permet de fournir des mécanismes de synchronisation plus complexes comme la présentation des données en séquence, simultanément, ou dans un ordre quelconque et s’adapte très bien à la transmission de messages multimédias, comme proposé dans [Pos88] et [Lit90].

**Une connexion multimédia**

La synchronisation de connexions de transport multiples consiste à bufferiser les données “en avance” dans l’attente des données correspondant aux marques de synchronisation. Outre la taille importante de tampons requises, cette approche n’est pas bien adaptée aux applications multimédias interactives où les attentes de resynchronisation pourraient impliquer une dégradation importante de la qualité de service.

Dans le protocole TPM, nous proposons l’utilisation d’une connexion de transport unique pour véhiculer les flots DT et RTD. Cette approche permet de synchroniser efficacement les flots transmis, c’est à dire sans surcharger le réseau par des informations additionnelles sur le séquencement des données. Elle suppose (1) un ordonnancement des flots de données à l’émission selon le séquencement voulu et (2) un réseau qui fournisse un service de circuit virtuel de coût léger c’est à dire ayant un taux négligeable d’erreurs de réception de paquets hors séquence. La protection contre des erreurs éventuelles sera assurée par la numérotation mutuelle des deux flots selon les deux modes de synchronisation proposés dans la section suivante.

La synchronisation dans le cas d’une connexion unique est effectuée par l’écartement (ou rejet) des données temps réel hors séquence au lieu de leur bufferisation. Cette approche se base sur la modification proposée du service de transport, et peut être justifiée comme suit:

- Le rejet de données temps réel selon le service de transport typé permet d’éliminer la contrainte intrinsèque sur la taille des buffers dans les réseaux à
haut débit. En plus, le service de transport typé permet un traitement beaucoup plus rapide des données temps réel.

- La synchronisation de connexions multiples n’est possible que par l’introduction de messages sécurisés dans le flot temps réel, ce qui serait incompatible avec le service de transport typé.

- La fréquence des messages temps réel est supposée être supérieure d’un ou deux ordres de grandeur à celle des messages de données standards. L’effet des données standards sur la qualité de service globale ne devrait pas être très important.

- L’utilisation d’une connexion unique permet de détecter plus facilement les erreurs de séquencement quand elles ont lieu.

La qualité de service

L’intégration des différents flots sur une même connexion transport présente un aspect dual du problème de la satisfaction de la qualité de service: les demandes des différents flots seront regroupées ensemble ce qui permet d’avoir une atomicité de la connexion multimédia en acceptant ou en refusant la connexion pour tous les flots ensemble. Cependant, ceci complique la recherche d’une politique efficace d’allocation des ressources dans le réseau et empêche de profiter des caractéristiques du multiplexage asynchrone de l’ATM jusqu’au niveau de la couche liaison. Ce coût est allégé par les avantages des nouveaux réseaux à haut débit en ce qui concerne les débits et les délais.

5.3.3 Les modes de synchronisation d’une connexion multimédia

Cette section est consacrée à la description de deux modes de mise en œuvre de l’intégration de flots multiples sur une même connexion de transport: la séquence relative et la séquence absolue.

La séquence relative

La synchronisation au niveau transport des flots DT et RTD peut être assurée en adoptant une règle de précédence, c’est à dire établir une séquence relative dans la transmission des deux flots. La règle peut être énoncée comme suit: les données temps réel “devraient être plus rapides que les données standards”. Autrement dit,
une RTD TPDU qui a été transmises avant une DT TPDU devrait être reçue par l'utilisateur de service transport distant avant cette même DT TPDU, sinon elle sera écartée et non pas délivrée en retard.

La justification de cette règle provient de l'existence de contraintes temporelles pour le flot temps réel: les statistiques sur les délais d'interarrivée des RTD TPDUs sont connues (par la négociation de la cadence de transmission), l'utilisateur de service transport ne devrait donc pas “attendre” une RTD TPDU, si elle arrive en retard c'est à dire après une date limite ou après la première DT TPDU transmise après elle. Cette deuxième condition assure un certain ordre relatif entre les flots; les marques de synchronisation sont “imitées” par les DT TPDUs. La figure 5.5 montre un transfert de données DT et RTD selon le mode séquence relative: la retransmission d'une TPDU DT n'affecte pas la réception des données RTD.

Figure 5.5: Transfert des données DT et RTD selon le mode séquence relative

Pour implémenter ce mode de synchronisation, les DT TPDUs devront contenir le numéro de séquence de la prochaine RTD TPDU à transmettre sur la connexion de transport. Un paramètre obligatoire sera ajouté alors dans les DT TPDUs pour les connexions TPM pour désigner cette information de séquenancement: le paramètre $N(RT'S)$. Du côté récepteur, l'entité de transport sauvegarde ce paramètre dans une variable d'état et la met à jour lorsqu'une nouvelle DT TPDU est reçue. Ceci va permettre aux DT TPDUs d'accomplir le “contrôle partiel” sur les RTD TPDUs.
sans pourtant stopper la transmission du flot temps réel.

Le scénario d’un transfert de données dans une application interactive (par exemple transfert d’images vidéo) sera comme suit :

**À l’émetteur**

L’utilisateur du service transport émet une demande d’envoi de données de transport (T-RT-DATA) quand une trame est prête à être envoyée. La trame (TSDU ou unité de donnée du service de transport) sera segmentée en plusieurs RTD TPDUs, et sera suivi par une ou plusieurs DT TPDUs dont une contiendrait éventuellement une demande de synchronisation qui est émise par l’application. Comme le service réseau n’est pas fiable, l’une quelconque de ces TPDUs pourrait ne pas atteindre la destination. La perte de RTD TPDUs va impliquer une dégradation de la qualité de service, c’est l’application qui décidera si la transmission devra être stoppée ou pas. Les DT TPDUs seront par contre retransmises à l’expiration d’une temporisation si un accusé de réception n’a pas été renvoyé. Le protocole assure la réception correcte de RTD TPDU même dans le cas de perte d’une DT TPDU. La TPDU retransmise portera alors le numéro d’une “ancienne” RTD TPDU. Toutefois, les RTD TPDUs transmises avant la réception correcte de la TPDU DT ne seront pas écarteres parce que la règle de précédence n’empêche pas les données RTD de “doubler” les données DT.

**Au récepteur**

Une RTD TPDU sera écartere si elle est “dépassée” par une DT TPDU. Comme la perte d’une DT TPDU est relevée au récepteur seulement après une retransmission réussie, les RTD TPDUs continueront à être acceptées dans le tampon de compensation (pour compenser le “jitter” du réseau) puis délivrées à l’application. La réception d’une DT TPDU “en retard” n’entrainera pas la perte de RTD TPDUs: l’information portée par le paramètre \( \text{\textit{N(RTS)}} \) sera considérée comme étant une indication d’état périmée. Le contrôle partiel est repris par les DT TPDUs suivantes. Si la DT TPDU contient une demande de synchronisation émise par l’application, la partie correspondante du buffer de réception des données RTD est passée à l’application en même temps que la partie correspondante des données DT. Les RTD TPDUs reçues “en retard” après la DT TPDU contenant la demande de synchronisation seront écarteres par le protocole.

Le fonctionnement correct d’un protocole de transport fournissant le service de synchronisation décrit ci dessus nécessite un mécanisme de numérotation approprié. Deux conditions devraient être vérifiées afin de garantir la réception correcte des RTD TPDUs: la première concerne le “modulus” de numérotation des RTD TPDU.
et la deuxième est une condition sur la cadence minimale des DT TPDU nécessaire pour garder l’alignement des données dans les deux flots.

Dans le but d’expliquer le problème des numéros de séquence considérons le scénario suivant: une DT TPDU qui a déjà été retransmises $N$ fois est reçue par l’entité de transport à la destination. Le paramètre $N(RTS)$ devrait être considéré comme faisant référence à une “ancienne” RTD TPDU afin d’éviter une perte importante de RTD TPDU. En effet, si la DT TPDU porte un paramètre $N(RTS)$ désignant une “très ancienne” RTD TPDU, elle pourra être considérée comme référençant une RTD TPDU plus récente. Dans ce cas là, toutes les RTD TPDU suivantes ayant un numéro de séquence “inférieur” au paramètre $N(RTS)$ seront considérées “en retard” et pursuite écartées.

Soient $N$ le nombre maximum de retransmission, $\tau$ l’intervalle minimale entre la transmission de deux RTD TPDU successives et $T$ la temporisation de retransmission. Lorsqu’une DT TPDU ayant $N(RTS) = I$ est retransmises $N$ fois, le nombre maximum $M$ de RTD TPDU qui pourraient être transmises avant la réception réussie de la DT TPDU à la destination devrait vérifier (voir figure 5.6 (a)):

$$ I + M > I. $$

les opérations sont effectuées modulo $F = 2^f$, $f$ représentant la taille du champ de numéro de séquence des RTD TPDU en bits. L’équation précédente se réécrit $M < 2^{(f-1)}$. En remplaçant $M$ par $NT/\tau$ on obtient:

$$ f > \log_2(2NT/\tau) + 1. $$

Pour un réseau de 1 Gbps, une taille de TPDU de 1 Ko, $N = 10$ et $T = 2s$ on a $f > 27$.

Une autre condition devra être vérifiée afin d’éviter un problème de séquencement: l’émetteur ne devrait pas transmettre plus que $F/2$ RTD TPDU sans informer la destination de la valeur actuelle du paramètre $N(RTS)$. Ceci pourrait être accompli en envoyant des ACK TPDU (pour Unité de Données du Protocole de Transport pour accusé de réception) portant la valeur incrémentée du $N(RTS)$. Cependant, comme les ACK TPDU pourraient se perdre, l’émetteur devra transmettre une ACK TPDU après chaque $F/2N$ RTD TPDU. Dans ce cas, la probabilité d’avoir une erreur de séquencement est réduite au dessous de celle qui cause la libération de la connexion de transport.

Le délai de réexpédition d’informations de contrôle de fenêtre $W$ devra alors vérifier:

$$ W < F\tau/2N $$

Si les deux conditions (sur $f$ et $W$) sont vérifiées, les DT TPDU assureront le contrôle partiel des RTD TPDU. Ce mode de synchronisation est un mode “touche
Figure 5.6: Perte éventuelle de données RTD dans le mode séquence relative
pas au temps réel", le seul cas où il pourrait y avoir une perte de données RTD étant la réception d'une RTD TPDU en retard sur une DT TPDU.

La séquence absolue

Le mode de synchronisation précédent est basé sur la règle "RTD est plus rapide que DT" afin de permettre la réception correcte de RTD TPDUs, même dans le cas de retransmissions de DT TPDUs. Il existe un moyen d’assurer une synchronisation plus stricte des deux flots: une autre règle de précédence est d’accepter les RTD TPDUs seulement quand elles arrivent en séquence par rapport aux DT TPDUs. En d’autre termes, si une RTD TPDU est transmise avant une DT TPDU elle devrait être reçue avant cette DT TPDU, sinon elle sera écartée et si une RTD TPDU est transmise après une DT TPDU elle devrait être reçue après cette DT TPDU, sinon elle sera écartée.

Ce mode peut être implémenté en incorporant les informations concernant la synchronisation dans les deux types de TPDUs DT et RTD: chaque TPDU contiendra deux paramètres de numérotations, l’une correspondant au numéro de séquence de cette TPDU et l’autre indiquant le numéro de séquence de la prochaine TPDU de l’autre type: N(DT) et N(RTS) pour les TPDUs DT et N(RT) et N(DTS) pour les TPDUs RTD. Cet arrangement assure la synchronisation mutuelle des deux flots selon le scénario suivant:

A l’émetteur

L’entité de transport segmentera l’image (TSDU) et la transmettra sur plusieurs RTD TPDUs suivies par des DT TPDUs contenant des données relatives à l’image transmise. L’entité de transport transmet les RTD TPDUs en incrémentant le paramètre N(DTS) à chaque envoi d’une DT TPDU. Cet arrangement correspond
alors à une séquence absolue de DT TPDUs avec des RTD TPDUs insérées entre elles. Les RTD TPDUs “dépassant” des deux côtés seront écartées à la réception. Selon ce mode de synchronisation, si une DT TPDU est perdue toutes les RTD TPDUs suivantes seront écartées, jusqu’à la retransmission réussie de la DT TPDU.

Au récepteur

Lorsqu’une RTD TPDU est reçue, le paramètre $N(DTS)$ est comparé à la variable d’état désignant la prochaine DT TPDU à recevoir, soit $N(DTR)$:

- si $N(DTS) < N(DTR)$, la RTD TPDU est en retard, elle a été transmise avant la dernière DT TPDU reçue, elle sera donc écartée.

- si $N(DTS) = N(DTR)$, la RTD TPDU est reçue “en séquence absolue”, elle sera copiée dans le tampon de compensation. Si le paramètre $N(RT)$ ne correspond pas à la prochaine RTD TPDU une perte de paquet RTD est détectée.

- si $N(DTS) > N(DTR)$, une DT TPDU a été alors perdue; deux alternatives sont théoriquement possibles: copier la RTD TPDU dans le tampon de compensation ou l’écarter. Si la DT TPDU pourrait être retransmise rapidement, les RTD TPDUs bufferisées seront alors remises à l’application selon la même séquence d’émission. Sinon, si le délai dépasse un seuil fixé par la contrainte de temps des RTD TPDUs, elles seront écartées.

Lorsqu’une DT TPDU est reçue, le paramètre $N(RTS)$ permet de détecter une perte éventuelle de la TPDU RTD correspondante. L’arrangement en séquence absolue permet une synchronisation plus stricte des deux flots en n’acceptant les RTD TPDUs que si elles arrivent “en séquence” par rapport aux DT TPDUs. Les données RTD devraient “se synchroniser” au flot des données DT. En effet, chaque DT TPDU correspond à une demande de synchronisation: le passage des données à l’application s’effectue à la réception d’une DT TPDU. Toutefois, le coût de cette synchronisation est une perte plus grande de RTD TPDU: dans l’absence de perte de DT TPDUs, les RTD TPDUs arrivant en avance ou en retard seront écartées. En plus, si une DT TPDU est perdue, toutes les RTD TPDUs reçues avant la retransmission réussie de cette DT TPDU seront aussi écartées.

5.4 Conclusion

La synchronisation au niveau transport correspond à un compromis entre l’efficacité de transmission et la fonctionnalité offerte à l’application. Le service de transport typé permet de mieux répondre aux besoins des applications, mais nécessite le multiplexage des flots de données sur une même connexion de transport. Le protocole
5.4. CONCLUSION

TPM présenté dans ce chapitre permet de fournir les deux optimisations: il fournit un service de transport typé, en effectuant la synchronisation des différents flots intégrés sur la même connexion. Pour cela, deux modes de synchronisation peuvent être utilisés: la séquence relative correspond au multiplexage des données DT et RTD avec une contrainte “en moyenne” sur le déphasage entre les deux flots. Selon ce mode, c’est la tâche de l’application ou de l’utilisateur d’envoyer occasionnellement les demandes de synchronisation dans des DT TPDUs particulières. Ceci permet d’assurer une synchronisation “sur demande” basé par exemple une demande d’affichage synchronisé d’une image et d’un texte lors de l’activation d’un bouton par l’utilisateur. La fréquence $\alpha_4$ de ces messages de synchronisation dépend de la nature de l’application. Le cas extrême où tous les messages DT sont des demandes de synchronisation correspond au mode séquence absolue dans lequel l’application est dégagée de la tâche de la gestion de la synchronisation. La séquence absolue permettant d’assurer une synchronisation plus stricte pour un coût plus élevé. Ce mode correspond à une synchronisation continue des flots transmis, donc de type lip-sync (comme par exemple dans la transmission de film vidéo où la parole devrait être reconstituée en même temps que l’affichage des images correspondantes).

Le chapitre suivant traite les aspects quantitatifs de la comparaison des deux modes de synchronisation selon le service typé et de la synchronisation par l’utilisation des messages marqués sur des connexions de transport standard.
Chapitre 6

Evaluation des modes de synchronisation

Ce chapitre traite les aspects quantitatifs permettant l’évaluation du coût de la synchronisation au niveau transport. Il s’agit donc d’établir et d’évaluer un modèle simplifié des différentes approches permettant d’effectuer la synchronisation transport:

- les connexions de transport multiples, avec l’utilisation de marques de synchronisation et le service de transport standard sur les différentes connexions,

- le service de transport typé: une connexion unique pour les données standards et les données temps réel, la synchronisation étant effectuée selon l’un des deux modes séquence relative ou séquence absolue.

La première approche suppose que le service de transport sur les différentes connexions soit le service standard c’est à dire en effectuant le contrôle d’erreurs afin d’assurer le transfert fiable de données sécurisées y compris les marques de synchronisation. La deuxième approche, fournie par le protocole TPM, permet de synchroniser des flots et de fournir le service temps réel pour les données RTD et le service standard pour les données DT. Nous allons commencer par la présentation du modèle simplifié des services de transport, ensuite nous évaluerons le coût de la synchronisation pour chacune des approches citées ci dessus et l’impact de la synchronisation sur la qualité de service globale des flots DT et RTD.

6.1 Modèle du service transport

Chaque connexion transport est modélisée par une file d’attente et un serveur. Les “clients” représentent les paquets ou TPDUs à transmettre. Le temps d’attente d’un client dans la file correspond au temps de service des clients antérieurs. Nous supposons que les arrivées des paquets des données RTD et DT forment des processus
Figure 6.1: Le modèle des services de transport standard et typé

de Poisson de paramètres $\lambda_r$ et $\lambda_d$ respectivement\(^1\) et que le débit moyen du flux temps réel $\lambda_r$ est largement supérieur à celui des données standards $\lambda_d$. Le "service" rendu à chaque paquet correspond à la transmission réussie de ce paquet; dans le cas d'un transport standard, un paquet est retransmis autant de fois qu'il est nécessaire pour assurer une transmission réussie\(^2\), alors que pour le transport temps réel un paquet RTD sera considéré "servi" dès la première transmission. Une connexion sera donc modélisée par une file M/G/1, avec une loi de service $B(x)$ différente selon le type de la connexion (voir figure 6.1).

Les variables aléatoires $Y_{dd}$, $Y_{dr}$ et $Y_{rr}$ représentent respectivement le délai d'un message DT avec un service standard, le délai d'un message RTD avec un service standard et le délai d'un message RTD avec un service temps réel.

### 6.1.1 Calcul du délai avec le service standard

Il est possible de calculer des statistiques sur le temps de transmission fiable d'un paquet si l'on connaît la loi de service: $B_d(x) = Pr[X < x]$, $X$ étant la variable aléatoire représentant le temps de service d'un paquet. Soit $\epsilon$ le pourcentage de

\(^1\)L'hypothèse d'un flux d'arrivée poissonnier n'est pas très exacte; les interarrivées varient selon les besoins de l'application en étant corréllées pendant les périodes d'émission. Pourtant, cette approximation nous permettra de faciliter les calculs afin d'obtenir des valeurs approchées des délais de transmission pour les flux DT et RTD.

\(^2\)Le nombre maximum de retransmission sera supposé assez élevé.
6.1. MODÈLE DU SERVICE TRANSPORT

paquets erronés, c'est à dire la probabilité qu'un paquet transmis ne soit pas reçu correctement. On a
\[ P_r[X = \tau + nT] = e^n(1 - \epsilon). \]

\( \tau \) étant le temps de transmission d’un paquet et \( T \) le délai de retransmission (temporisation de retransmission + délai de transmission). La transformée de Laplace-Stieljes (TLS) \( B_d^*(s) \) de \( B_d(x) \) sera donc:
\[ B_d^*(s) = E[e^{-sx}] = \int_0^\infty e^{-sx}dB_d(x) = \sum_{n=0}^{\infty} e^{-s(\tau+nT)}.e^n(1 - \epsilon) = \frac{(1 - \epsilon)e^{-s\tau}}{1 - \epsilon e^{-Ts}} \]

La TLS du délai de transmission \( Y \) d’un paquet pourra donc être écrite\(^3\)
\[ S_d^*(s) = B_d^*(s) \frac{s(1 - \rho)}{s - \lambda + \lambda B_d^*(s)} \]
d'où:
\[ S_d^*(s) = \frac{(1 - \rho)(1 - \epsilon)se^{-s\tau}}{\lambda(1 - \epsilon)e^{-s\tau} + (s - \lambda)(1 - \epsilon e^{-sT})} \]
\[ S_{dd}^*(s) = S_d^*(s) \mid_{\lambda = \lambda_d, \rho = \rho_d} = E[e^{-sY_{dd}}] \quad (6.1) \]
\[ S_{dr}^*(s) = S_d^*(s) \mid_{\lambda = \lambda_r, \rho = \rho_r} = E[e^{-sY_{dr}}] \quad (6.2) \]

On pourra alors calculer la moyenne et la variance du délai de transmission en utilisant la formule suivante\(^4\):
\[ S_d^{(k)}(0) = (-1)^k E[Y_d^k] \quad (6.3) \]

Les variations de la moyenne et de l’écart type pour \( Y_{dd} \) et \( Y_{dr} \) sont tracées dans les figures 6.2, 6.3,6.4 et 6.5 respectivement. Le seul paramètre différent entre ces deux figures étant dans le débit: le débit pour les données temps réel \( \lambda_r \) est supposé 100 fois supérieur à celui des données normales \( \lambda_d \).

6.1.2 Calcul du délai avec le service temps réel

Dans le cas d’un service transport temps réel le temps de service d’un paquet est égal à \( \tau \) quelque soit le pourcentage d’erreur sur la connexion, puisqu’un paquet RTD sera transmis une seule fois. on a donc:
\[ P_r[X = x] = \begin{cases} 1 & \text{si } x = \tau \\ 0 & \text{si } x \neq \tau \end{cases} \]

\(^3\)D’après la formule (5.100) page 199 dans [Kle75].
\(^4\)De [Kle75], page 185.
Figure 6.2: Délai moyen $E(Y_{dd})$ (en µs) des messages de données pour un service standard en fonction de $\lambda_d$, avec $\lambda_r = 100\lambda_d$, $\tau = 10\mu$s, $T = 100$ms. Les trois courbes de haut en bas correspondent à $\epsilon=10^{-5}$, $10^{-6}$ et $10^{-7}$ respectivement.

alors,

$$B^*_r(s) = e^{-s\tau}$$

et

$$S^*_r(s) = \frac{(1 - \rho)s e^{-s\tau}}{s - \lambda + \lambda e^{-s\tau}}$$

$$S^*_{rr}(s) = S^*_r(s) \mid_{\lambda - \lambda_r - \rho_r} = E[e^{-sY_{rr}}]$$

On pourra donc calculer $E(Y_{rr})$ et $\sigma(Y_{rr})$ (voir figures 6.6 et refmvarrrb).
6.2. LE COÛT DE LA SYNCHRONISATION

Figure 6.3: Écart type du délai des messages de données $\sigma (Y_{dd})$ pour un service standard en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu s$, $T = 100ms$. Les trois courbes de haut en bas correspondent à $\varepsilon=10^{-5}$, $10^{-6}$ et $10^{-7}$ respectivement.

6.2 Le coût de la synchronisation

La primitive “synchronisation” consiste à envoyer avec une fréquence $\alpha_s$ des messages particuliers et à attendre leur réception au destinataire avant qu’une action soit accomplie. La valeur de $\alpha_s$ peut être

- égale à 0 : aucune demande de synchronisation par l’application,
- comprise entre 0 et $\lambda_d$ : une fraction des messages DT portent une demande de synchronisation par l’application ce qui correspond au mode “séquence relative” dans le cas d’un service typé,
- égale à $\lambda_d$ : tous les messages DT contiennent des demandes de synchronisa-
Figure 6.4: Délai moyen $E(Y_{dr})$ (en µs) des messages de données temps réel pour un service standard en fonction de $\lambda_d$, avec $\lambda_v=100\lambda_d$, $\tau=10\mu s$, $T=100\text{ms}$. Les trois courbes de haut en bas correspondent à $\epsilon=10^{-5}$, $10^{-6}$ et $10^{-7}$ respectivement.

Ceci permet d’effectuer un seul calcul pour les deux modes du service typé : “séquence relative” et “séquence absolue”. Les résultats étant obtenus par les valeurs différentes de $\alpha_s$.

Dans le cas des connexions multiples (utilisation des marques de synchronisation avec un service de transport standard sur les connexions) le coût “d’une synchronisation” $C_1$ est défini comme étant le délai d’attente entre la réception de la marque de synchronisation sur la connexion temps réel et la réception de la marque de synchronisation sur la connexion de données standards. Ce délai donne une indigation de la détérioration de la qualité de service du flot temps réel dû à la synchronisation.
Figure 6.5: Écart type du délai des messages de données temps réel \( \sigma(Y_{dr}) \) pour un service standard en fonction de \( \lambda_d \), avec \( \lambda_r = 100 \lambda_d \), \( \tau = 10 \mu s \), \( T = 100 \text{ms} \). Les trois courbes de haut en bas correspondent à \( \epsilon = 10^{-5}, 10^{-6} \) et \( 10^{-7} \) respectivement.

On peut écrire:

\[
C_1 = (Y_{dd} - Y_{dr})1(Y_{dd} > Y_{dr}) = max(0, Y_{dd} - Y_{dr})
\]  \( (6.4) \)

Le retard de la marque de synchronisation sur la connexion temps réel par rapport à celle de la connexion de données standards ne devrait pas être pris en compte si l'on évalue seulement l'influence de la synchronisation sur le flux temps réel. Toutefois, ce retard indique une détérioration proportionnelle de la qualité de service du flux temps réel due aux retransmissions. On a donc

\[
C_2 = Y_{dr} - \tau
\]  \( (6.5) \)

où \( C_2 \) représente le “coût” du service standard sur la connexion de données temps réel. La “perte” totale de messages temps réel due à la synchronisation selon ce mode
Figure 6.6: Délai moyen $E(Y_{rr})$ (en $\mu$s) des messages de données temps réel pour un service temps réel en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu$s, $T = 100\text{ms}$ et $\epsilon=10^{-6}$.

de service sera donc:

$$P_d = \alpha_d(C_1 + C_2)$$  \hspace{1cm} (6.6)

Le coût de la synchronisation sera exprimée en “délai” parce qu’un retard sur une connexion temps réel signifie une perte d’informations temps réel.

Pour le service de transport typé, les messages de synchronisation sont transmis seulement sur le flot DT. Toutefois, le coût d’une synchronisation est défini dans ce cas comme étant le délai d’attente entre la réception d’un message “fictif” transmis sur le flot RTD au même instant de l’issue du message de synchronisation et la réception réussie de ce message de synchronisation

$$C_1 = \max(0, Y_{dd} - Y_{rr})$$
Figure 6.7: Écart type du délai des messages de données temps réel $\sigma(Y_{rr})$ pour un service temps réel en fonction de $\lambda_d$, avec $\lambda_r=100\lambda_d$, $\tau = 10\mu s$, $T = 100\text{ms}$ et $\epsilon = 10^{-6}$.

En plus, selon le protocole TPM tous les messages RTD reçus “en retard” par rapport à un message DT seront écartés. Le coût additionnel sera proportionnel à

$$C_2 = \max(0, Y_{rr} - Y_{dd})$$

La “perte” totale de messages RTD s’écrit donc

$$P_t = \alpha_s C_1 + \lambda_d C_2$$  \hspace{1cm} (6.7)

La première composante correspond à l’attente de messages RTD pour l’arrivée des messages DT contenant une demande de synchronisation, alors que le deuxième terme représente les messages RTD écartés parce qu’ils sont reçus “en retard”.

Cette expression est valable pour les deux modes séquence relative et séquence absolue qui sont alors représentés par le même modèle, la fréquence des messages
de synchronisation étant différente pour chaque mode: la séquence absolue est un cas particulier de la séquence relative correspondant à α_s = λ_d. Dans ce cas, nous retrouvons dans l’équation 6.7 les deux termes correspondant aux pertes de messages RTD arrivant en avance ou en retard par rapport aux messages DT.

Les fonctions de répartition (FR) des variables aléatoires Y_{dd} et Y_{dr}, connues par leur TLS, correspondent toutes les deux à un service standard dans le cas de connexions multiples (S_{dd}^*(s) et S_{dr}^*(s) obtenues de S_d^*(s) en remplaçant ρ et λ par ρ_d et λ_d pour la première et par ρ_r et λ_r pour la deuxième), alors que Y_{dd} et Y_{rr} correspondent chacune à un type de service différent dans le cas du service de transport typé (S_{dd}^*(s)etS_{rr}^*(s)).

6.2.1 Calcul pour un service standard

Nous allons nous contenter du calcul du premier moment de C_1 à partir de C_1^*(s) la TLS de la loi de distribution de C_1 par la formule 6.3.

Soient Y = Y_{dd} - Y_{dr} et R^*(s) la TLS de la FR de Y. On a

\[ R^*(s) = E[e^{-sY}] = E[e^{-sY_{dd}}]E[e^{-(s)Y_{dr}}] \]

donc pour les connexions standards multiples:

\[ R^*(s) = S_{dd}^*(s),S_{dr}^*(-s) \]

\[ C^*(s) = E[e^{-sC_1}] \] est donnée par la formule suivante\(^5\) qui permet de calculer la TLS de la FR de max(0,Y) connaissant la TLS de la FR de Y:

\[ C_1^*(s) = -\lim_{\Re{z} \to 0 \atop \Im{z} \neq 0} \frac{1}{2i\pi} \left( \int_{i\mathbb{R}} R^*(s + y) \frac{dy}{y - z} + \int_{i\mathbb{R}} R^*(y) \frac{dy}{y - z} \right) \]

où \(\Re{z}\) désigne la partie réelle d'un nombre complexe \(z\).

On a donc

\[ E(C_1) = -C^*(0) = \lim_{z \to 0 \atop \Re{z} > 0} \frac{1}{2i\pi} \left( \int_{i\mathbb{R}} R^*(y) \frac{dy}{y - z} \right) \]

Un théorème démontré dans [Jea87] page 133 permet de se débarasser de la limite:

\[ E(C_1) = \frac{1}{2i\pi} \int_{i\mathbb{R}} (R^*(y) - R^*(0)) \frac{dy}{y} + sgn(\Lambda) R^*(0) \]

6.3 ANALYSE DES RÉSULTATS

où \( \text{sgn}(\lambda) \) est la fonction “signe” qui prend la valeur \(-\frac{1}{2}\) pour le domaine \( \Lambda \) correspondant à \( R(z) > 0 \).

\[
E(C_1) = \frac{1}{\pi} \int_{\mathbb{R}} \left( \frac{R^*(iy) - R^*(0)}{2i} \right) \frac{dy}{y} + \frac{1}{2} (E(Y_{dd}) - E(Y_{dr}))
\]
d'où

\[
E(C_1) = \frac{1}{\pi} \int_{0}^{\infty} \Im(R^*(iy)) \frac{dy}{y} + \frac{1}{2} (E(Y_{dd}) - E(Y_{dr})) \tag{6.8}
\]
où \( \Im(R^*(iy)) \) désigne la partie imaginaire de la fonction complexe \( R^*(iy) \). L’expression obtenue est intégrée numériquement (en utilisant Maple [Cha88]) pour différentes valeurs du débit moyen des flux de données standards et temps réel.

On a donc

\[
E(P_d) = \alpha_s (E(C_1) + E(Y_{dr}) - \tau) \tag{6.9}
\]

6.2.2 Calcul pour le service typé

Soient \( Y_1 = Y_{dd} - Y_{rr}, Y_2 = -Y_1, R_1^*(s) \) et \( R_2^*(s) \) les TLS des FRs de \( Y_1 \) et \( Y_2 \) respectivement.

\[
R_1^*(s) = S_{dd}^*(s).S_{rr}^*(-s)
\]
et

\[
R_2^*(s) = S_{rr}^*(s).S_{dd}^*(-s)
\]

On pourra alors calculer \( E(C_1) \) et \( E(C_2) \) en utilisant la formule 6.8

\[
E(C_1) = \frac{1}{\pi} \int_{0}^{\infty} \Im(R_1^*(iy)) \frac{dy}{y} + \frac{1}{2} (E(Y_{dd}) - E(Y_{rr}))
\]
et

\[
E(C_2) = \frac{1}{\pi} \int_{0}^{\infty} \Im(R_2^*(iy)) \frac{dy}{y} + \frac{1}{2} (E(Y_{rr}) - E(Y_{dd}))
\]

Ces intégrales seront de même évaluées numériquement pour différentes valeurs du débit. On obtient

\[
E(P_t) = \alpha_s E(C_1) + \lambda_d E(C_2)
\]

6.3 Analyse des résultats

6.3.1 Service de transport standard

L’évaluation du coût moyen de la synchronisation dans le cas d’un service standard est montrée dans la figure 6.8.
Figure 6.8: Coût moyen de la synchronisation pour un service standard en fonction de $\lambda_d$. Les courbes de haut en bas représentent respectivement les variations de $P_d$, $\alpha_s E(C_2)$ et $\alpha_s E(C_1)$ en fonction de $\lambda_d$, avec $\tau = 10 \mu s$, $T = 100 \text{ ms}$, $\epsilon = 10^{-6}$, $\lambda_r = 100 \lambda_d$, $\alpha_s = 25$.

La valeur de $\alpha_s$ correspond au nombre de demandes de synchronisation par seconde envoyées sur le flux $DT$. La courbe montre que le coût moyen dû à la synchronisation diminue quand les débits augmentent. En effet, ce coût est proportionnel à l’attente des messages de la connexion temps réel entre la réception de la marque de synchronisation sur cette connexion et la réception de la marque sur la connexion de données standards. Comme le service est le même pour les deux connexions (avec les mêmes valeurs des paramètres $\tau$, $T$, $\epsilon$), les délais dépendent alors des débits $\lambda_d$ et $\lambda_r$. Le débit $\lambda_r$ étant supposé largement supérieur à $\lambda_d$, le délai moyen sur la connexion temps réel est plus important que sur la connexion standard (voir figures 6.2, 6.3, 6.4 et 6.5), ce qui explique le fait que l’attente moyenne des messages temps réel pour l’arrivée de la marque de synchronisation sur la connexion de données standards soit faible. Ce coût diminue avec l’augmentation des débits, car l’écart entre les délais moyens sur les deux connexions augmente, réduisant ainsi le coût moyen de la synchronisation.

Cependant, la qualité de service globale pour le flux temps réel n’est pas satis-
6.4. CONCLUSION

...faisante. En effet, le coût dû au service de transport standard \( \alpha_x E(C_2) \) est plus important et augmente avec le débit du flux temps réel. À très haut débit cette mauvaise qualité de service l’emporte; la variation du délai sur la connexion temps réel perturbe la reconstitution des images animées ou du signal vocal même si le coût additionnel de la synchronisation est faible devant celui du service de transport standard pour le flux temps réel.

6.3.2 Service de transport typé

L’intérêt de ce mode de service est de réduire considérablement la moyenne et la variance du délai des messages RTD (voir figures 6.6 et 6.7). Dans ce cas le coût moyen dû à la synchronisation augmente comme le montre la figure 6.9. En effet, le délai des messages RTD augmente beaucoup plus lentement avec \( \lambda_d \) que dans le cas d’un service standard. La moyenne des “dépassements” du délai des messages DT par rapport aux messages RTD et inversement est plus importante et augmente avec le débit. Toutefois, d’après les figures 6.8 et 6.9 la détérioration de la qualité de service dû au coût de la synchronisation avec le transport typé est inférieure à celle dûe aux délais de retransmissions des messages temps réel dans un service de transport standard. L’adoption du service de transport typé permet donc d’améliorer les performances globales du flux temps réel, le coût de la synchronisation provenant de l’attente des messages de données étant compensé par la réduction de la valeur moyenne et de la variance des délais des messages RTD.

Les deux courbes de la figure 6.9 correspondent aux modes séquence relative \( (\alpha_s = 25) \) et séquence absolue \( (\alpha_s = \lambda_d) \). Dans les deux cas la valeur de \( P_t \) correspondant à une certaine valeur de \( \lambda_s \) est plus importante que \( \alpha_x E(C_1) \), mais inférieure à la valeur correspondante de \( P_d \) pour le même \( \lambda_d \). Ceci confirme que l’adoption du service de transport typé permet d’améliorer les performances globales pour la synchronisation des deux flots de données. D’autre part, en comparant les deux courbes de la figure 6.9, on remarque que le coût de la synchronisation en séquence absolue est plus important qu’en séquence relative et que la différence augmente avec \( \lambda_d \). Toujours est il que pour les valeurs pratiques de \( \lambda_d \) (jusqu’à 200) et de \( \lambda_r \) (jusqu’à 20000) la valeur du coût de la séquence absolue dans le cas d’un service typé reste inférieur à celui du cas d’un service standard.

6.4 Conclusion

Le protocole TPM permet d’offrir aux applications multimédias un service de transport typé et les facilités pour effectuer la synchronisation des flots DT et RTD. Si le coût de la synchronisation pour les deux modes séquence relative et séquence ab-
Figure 6.9: Coût moyen de la synchronisation $P_t$ pour un service typé (en fonction de $\lambda_d$). La courbe d’en haut correspond à $\alpha_s=\lambda_d$, et celle d’en bas à $\alpha_s = 25$.

En solue est plus important que celui de la synchronisation de connexions de transport multiples, l’amélioration globale de la qualité de service pour le flot RTD montre l’intérêt du changement du modèle de service de transport.
Chapitre 7

Conclusion

L’émergence des nouveaux réseaux à haut débit et à faible taux d’erreurs de transmission permet de lever la limitation qui provenait de la bande passante du réseau de transmission. Avec l’introduction des HSLANs, des MANs et des réseaux ATM le goulot d’étranglement est passé du réseau au traitement requis pour l’exécution des protocoles de communications dans les stations. Les protocoles de contrôle de transmission conçus pour corriger les erreurs sur les réseaux à faible débit (64 kb/s jusqu’à 1 Mb/s) ne sont plus adaptés à ce nouvel environnement: ces protocoles minimisent les coûts de transmission au détriment d’une complexité dans les procédures de contrôle nécessitant une importante consommation CPU. D’autre part, de nouvelles applications telles que la visualisation graphique haute résolution, l’affichage interactif d’une simulation en cours ou la téléconférence nécessitent que l’augmentation des débits soit propagée au niveau des applications. En plus, ces applications intègrent l’échange de plusieurs flots de données ayant des caractéristiques différentes en ce qui concerne les délais et la fiabilité de la transmission. Ceci nécessite une révision du service de transport fourni aux applications afin de pouvoir satisfaire les besoins des nouvelles applications. Les avancées technologiques et les nouveaux réseaux sont décrits dans le chapitre 2, les besoins des nouvelles applications et le service de transport requis sont présentés dans le chapitre 3.

Plusieurs améliorations ont été proposées dans le but d’optimiser les performances des protocoles de contrôle de transmission standards. Ces améliorations concernent soit l’optimisation de l’environnement et/ou des choix de l’implémentation, soit le réglage des paramètres des protocoles afin de minimiser les délais de transmission ou de maximiser le débit, soit les primitives de base fournies par le protocole allant donc jusqu’à la conception d’un nouveau protocole. Nous présentons dans le chapitre 4 une analyse des techniques d’amélioration des performances des protocoles de contrôle de transmission.

Dans le chapitre 5 nous montrons la nécessité d’un mécanisme efficace de synchronisation des différents flots de données pour les applications multimédias. Nous proposons ensuite d’enrichir le service de transport standard OSI afin de fournir
aux applications des primitives permettant d'effectuer la synchronisation des flots multiples. En particulier, nous choisissons de multiplexer les flots données d’une application sur une même connexion de transport “multimédia” et d’effectuer le contrôle de transmission selon le type des données: les primitives standards de contrôle de flux et d’erreur seront appliquées aux données standards alors que seul un contrôle partiel qui sert à détecter les erreurs de synchronisation sera appliqué aux données temps réel. Nous proposons ensuite une extension du protocole TP4 (appelée TPM) qui fournit le service “typé” ainsi que les mécanismes de synchronisation des flots multiples. Deux modes de synchronisation sont décrits: l’un appelé (séquence relative) permet à l’application de garder le contrôle sur les demandes de synchronisation alors que le deuxième (séquence absolue) impose une stricte synchronisation au niveau transport et dégage l’application la tâche d’envoi des requêtes de synchronisation.

Le chapitre 6 est consacré à l’évaluation des deux modes de synchronisation et à leur comparaison avec d’autres mécanismes proposés pour effectuer la synchronisation au niveau transport (en gardant le service de transport standard et des connexions de transport multiples pour les différents flots). L’analyse d’un modèle simplifié montre que le “coût” de la synchronisation est inférieur au “coût” du service standard (en délai): cette comparaison a un sens parce que des délais de transmission importants et variables sont équivalents à une perte des paquets ayant des contraintes temporelles.

Les techniques d’implémentation du protocole TPM dans le noyau Unix sont présentées dans l’annexe C.

Ce travail sera poursuivi dans le cadre du projet ESPRIT OSI 95 par une évaluation des fonctions des couches supérieures du modèle OSI dans le but de proposer une architecture optimisée pour les applications OSI multimédias (gestion de la synchronisation et de l’hétérogénéité à haut débit).
Bibliographie


[X3S90b] Modifications to the transport layer service description in support of very high speed networking. Draft X3S3.3/90-204, CBEMA, June 1990.


[X3S90d] Modifications to the network layer service description in support of very high speed networking. Draft X3S3.3/90-206, CBEMA, June 1990.


[Xtp90] XTP Protocol Definition, Revision 3.5. PEI 90-120, xtp-request@pei.com, Protocol Engines Incorporated, Santa Barbara, CA 93101, USA.


BIBLIOGRAPHIE


Annexe A

Glossaire

Les termes suivants, listés dans l’ordre alphabétique, sont utilisés dans le texte.

application haut débit : application réseau nécessitant l’échange d’un large volume de données dans un laps de temps réduit non nécessairement en temps réel par exemple le transfert de gros fichier, mais aussi la transmission d’images vidéo.

application multimédia : application réseau échangeant des flots de données différents (standards et temps réel).

application réseau : logiciel tournant sur un système final et échangeant des données avec une entité homologue dans le même ou dans un autre système final.

application temps réel : application haut débit interactive ou pas échangeant des données temps réel. (Exemple: transmission d’images animées).

contrôle de cadence : une technique qui consiste à limiter le débit de transmission d’un émetteur en imposant un délai d’attente après chaque paquet et/ou chaque groupe de paquets.

données standards : données transmises avec un pourcentage de perte nul. En cas de perte, ces données seront retransmises par la source afin d’assurer la fiabilité de la transmission. (Exemple: contenu d’un fichier, courrier électronique).

données temps réel : données ayant des contraintes strictes sur les délais de transmission. Si ces données ne sont pas reçues à la destination avant un délai donné elles seront considérées comme perdues (non valides) et ne seront pas retransmises. (Exemple: voix paquetisée ou images vidéo).

écartement de données: rejet de données par une entité transport (discard).
flot de données : flux unidirectionnel de données transmises en séquence.

modèle stream : mode de transmission dans lequel toutes les données sont délivrées à l’utilisateur de service transport destination dans le même ordre dans lequel elles étaient transmises. Pour cela, les données altérées ou perdues sont retransmises, les données dupliquées sont écartées et les données reçues hors séquence sont remises en séquence.

réseau haut débit : réseau permettant un débit supérieur ou égal à 100 Mbit/s au niveau de la couche physique.

synchronisation de flot de données : assurer le passage simultané de données spécifiques appartenant à des flots distincts à une couche supérieure.

système d’exploitation temps réel : système d’exploitation permettant de satisfaire des contraintes de temps spécifiées en ce qui concerne le lancement et la terminaison des processus. Ceci nécessite que l’ordonnancement des processus soit effectué selon des algorithmes basés sur des mécanismes de priorités dynamiques avec préemption.

système final : système implémentant les couches supérieures du modèle OSI (transport, session, présentation et application).
Annexe B

Format de TPDUs

Signification des champs:

LI : Longueur de l’entête (Length Indicator);

DT: code de données standards (0x 1111 0000);

RTD: code de données temps réel (0x 0100 0000);

AK: code d’accusé de réception de données (0x 0110 0000);

Référence destination: contient l’identificateur de la connexion transport à l’entité réceptrice;

N(DT): numéro de séquence de TPDU DT envoyée, (format étendu);

N(RT): numéro de séquence de TPDU RTD envoyée, (format étendu);

N(YR−TU): numéro de séquence de la prochaine TPDU DT attendue, (format étendu);

Fin de TSDU: indique que la TPDU courante est la dernière d’une suite de TPDUs formant une seule TSDU;

N(RTS): numéro de séquence de la prochaine RTD TPDU qui sera transmise après la DT TPDU sur la même connexion de transport;

N(DTS): numéro de séquence de la prochaine DT TPDU qui sera transmise après la RTD TPDU sur la même connexion de transport;

CDT: valeur du crédit;

Partie variable: contient le paramètre de total de contrôle (TPDU DT) et éventuellement le paramètre N(RTS) (dans les AK TPDUs).
Figure B.1: Format des TPDUs DT, RTD et AK
Annexe C

Techniques d’implémentation

C.1 Environnement

Nous avons décidé d’implémenter le protocole TPM dans le noyau Unix afin de respecter un certain niveau de performance et de bénéficier du support pour la communication inter-processus déjà fourni par l’interface socket [Lef82], [Lef83]. D’autre part, nous avons voulu avoir une implémentation complète des couches basses du modèle OSI, alors nous sommes partis d’une version du système SMX V [Gip85] qui supportait déjà la famille de protocoles ISO (voir figure C.1) en lui ajoutant les protocoles qui n’avaient pas été implémentés (voir figure C.2). L’insertion de TPM dans le noyau comme étant un nouveau protocole de la famille ISO est préférable à l’approche de type “driver” à cause du principe de découpage en couches des protocoles ISO. En effet, outre la limitation par les appels systèmes standards (open, close, read, write, ioctl), chaque protocole implémenté suivant le schéma “driver” est traité comme un contrôleur spécifique et indépendant ce qui ne prend pas en compte les concepts de hiérarchie et de coopération entre protocoles induits par le modèle OSI et conduit à une redondance de code.

La version du système SMX V.4 fournissait des “sockets transport” et des “sockets session”\(^1\). Nous avons ajouté le support du service IEEE 802.2 LLC-1 au dessus d’Ethernet pour tous les protocoles des couches supérieures (et non seulement pour X.25). Nous avons aussi implémenté les protocoles CLNP et ES/IS de la couche réseau (à la place de la partie IS “Inactive Subset” de ISO-IP dans SMX V.4). La structure en couche nous permet d’avoir à chaque niveau une prise “socket”: les sockets IEEE permettent de tester les performances maximales qu’on peut atteindre sur le réseau local et les sockets CLNP fournissent un mécanisme de test de l’adressage NSAP et du routage ISO.

\(^1\)Nous n’avons pas utilisé les sockets session et nous pensons qu’il ne fallait même pas les implé- menter: d’une part, les mécanismes de la couche session doivent être accessible à l’application et d’autre part le code est trop lourd pour qu’il soit introduit dans le noyau Unix.
### C.2 Méthode adoptée

L'implémentation de TPM n'a pas posé beaucoup de problèmes parce que nous sommes partis du code TP4 fourni dans la version SMX V.4. Il a fallu effectuer les opérations suivantes:

- ajouter la négociation du service typé à l'établissement de la connexion,
- changer la structure des pcb's (protocol control blocks) afin d'introduire les nouvelles variables d'état,
- ajouter un appel ioctl afin de choisir le mode de synchronisation (séquence relative ou absolue),
- modifier la machine de protocole afin d'appliquer les fonctions de contrôle selon le type de données (DT ou RTD) qui est spécifié par un "flag" du send. Par exemple, lors de l'envoi de données RTD les variables d'états seront mises à jour sans toutefois bufferiser les TPDUs transmises, alors que les TPDUs DT sont bufferisées et éventuellement retransmises même en cours de transmission de TPDUs RTD.
C.2. MÉTHODE ADOPTÉE

Le passage des données RTD à l’application s’effectue dès que le contenu des buffers de réception dépasse un certain seuil. Dans le cas de perte d’une DT TPDU et selon le mode de synchronisation choisi, les RTD TPDUs reçues sont soit mises dans le buffer de réception (séquence relative), soit écartées (séquence absolue).

Les modifications introduites n’affectent pas beaucoup la structre du noyau Unix. Une première version représente environ une extension de 80 Koctets du noyau de la version SMX V.4 d’origine. Ceci inclue l’ajout des sockets IEEE et CLNP et d’autres modifications présentées dans le paragraphe 4.1.2. Si l’essentiel de l’architecture est maintenant réalisé, il faudra tester les performances du protocole TPM sur un réseau haut débit tel que FDDI, chose qui sera effectuée dans le cadre du projet OSI95.
Etude des protocoles de contrôle de transmission à haut débit pour les applications multimédias.

Les avancées technologiques dans les domaines de la transmission et de la commutation a conduit au développement de réseaux à haut débit tels que FDDI, DQDB et le RNIS large bande. L’augmentation des débits de transmission au delà de 100 Mbits par seconde permet de considérer de nouvelles applications réseau (dites applications multimédias) supportant une intégration de plusieurs types de flots ayant des caractéristiques différentes, comme les données textuelles, la voix numérisée et les images vidéo. Cependant, l’implantation de telles applications sur les réseaux à haut débit pose un problème au niveau du contrôle de la transmission des différents flots et de leur synchronisation. Cette thèse s’intéresse à la conception et à l’implantation de protocoles de transport à haut débit et en particulier ceux qui sont adaptés aux applications multimédias. Nous analysons les mécanismes utilisés par les protocoles de transport afin de détecter les points de blocages qui empêchent l’amélioration des débits de se propager jusqu’aux applications, en prenant en compte les avancées technologiques et les besoins des applications. Ceci nous amène à la définition d’un nouveau service de transport “typé” selon la nature du flot de données à transmettre. Nous présentons ensuite un protocole de transport optimisé (appelé TP5) qui fournit aux applications le service typé et les facilités pour la synchronisation des différents flots de données au niveau transport. Nous analysons ensuite le coût de deux modes de synchronisation possibles offerts par TP5 et nous montrons que le service de transport typé permet d’améliorer la qualité de service fournie aux applications multimédias.

Protocoles à Haut Débit
Protocoles de Communication
Applications Multimédias
Synchronisation

Réseaux Normalisés
Réseaux à Haut Débit
Contrôle de Transmission
Files d’Attente
High Speed Transport Protocols for Multimedia Applications.

New technologies in both transmission and switching led to the development of high speed networks such as FDDI, DQDB or B-ISDN. The increase in the transmission speed up to the gigabit per second range allows the consideration of new applications called multimedia applications integrating the transmission of several data flows with different characteristics like text, voice and video. However, the implementation of such applications over high speed networks require an efficient mechanism to control the transmission of the different data flows and to synchronize them. This thesis is concerned with the design and implementation of high speed transport protocols and in particular those which are suitable for multimedia applications. We analyze the transmission control mechanisms in order to detect the source of the overhead which prevent the bandwidth increase to be available for the applications, taking into consideration the technological improvements and the applications needs. This lead us to the definition of a new transport service “typed” according to the nature of the flow to be transmitted. We also present an enhanced transport protocol (called TP5) which provide for the applications this typed transport service as well as mechanisms for the synchronization of the different data flows at the transport level. We then analyse the cost of two possible synchronization modes in TP5 and we show that the typed transport service allows for the enhancement of the global quality of service for multimedia applications.

High Speed Protocols
Communication Protocols
Multimedia Applications
Synchronization

Open Networks
High Speed Networks
Transmission Control
Queues