Université de Montréal

# Ordonnancement des flots agrégés dans les réseaux IP multiservices

par

Hichem Sarraï

Département d'informatique et
de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
**Maître ès sciences (M. Sc.)**
en informatique

Décembre 2004

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé :

Ordonnancement des flots agrégés dans les réseaux IP multiservices

présenté par :

Hichem Sarraï

a été évalué par un jury composé des personnes suivantes :

Abdelhakim Hafid
(président-rapporteur)

Brigitte Jaumard
(directeur de recherche)

Rachida Dssouli
(membre du jury)

Mémoire accepté le
7 mars 2005

# Sommaire

Malgré sa popularité, l'Internet est juste capable d'offrir une seule classe de service de type *Best-Effort* et ne peut pas supporter les services temps réel avec les garanties de Qualité de Service désirées. Malgré qu'un nombre signifiant de travaux ont été effectués dans ce domaine, on croit encore que les problèmes posés par les services temps réels à travers les réseaux IP n'ont pas encore été bien résolus.

Dans notre projet, nous considérons le cas d'un ordonnancement agrégés des flots avec un choix de quatre classes de services différentes. Pour ce faire, nous explorons les apports d'un groupage des flots, particulièrement, la possibilité d'un calcul précis d'éventuelles bornes déterministes sur les délais de bout en bout. Nous enchaînons avec la conception et l'implantation d'un WFQ adaptatif. Nous évaluons ses performances en terme du niveau de respect des principales contraintes de qualité de services relatives aux applications temps réels : le délai et la gigue.

L'algorithme développé ainsi que les simulations ont été implementés et validés avec l'environnement de modélisation de réseaux OPNET Modeler.

Les expériences de calculs ont montré que les bornes déterministes sur les délais sont peu précises par rapport aux délais réelement encourus par les paquets. On ne peut donc pas concevoir un mécanisme de contrôle d'admission efficace avec de pareils calculs sur les bornes.

Cependant, les résultats de calculs obtenus à l'issu des simulations relatives à la politique adaptative montrent les bénéfices d'une telle agilité du réseau lorsqu'il s'agit du traitement d'une charge importante et variable de trafic dans un contexte multiservices.

Mots clés : Réseaux Multiservices, Ordonnancement, WFQ, Agrégation, Gestion de Trafic, QdS, bornes de délai.

# Abstract

Despite it's popularity, Internet can only offer one *Best-Effort* service class. It cannot support real-time services with desirable Quality of Services guarantees. Althought sgnificant volume of work has been done in this area, it is believed that the provision of real-time services over IP networks has not been properly addressed.

In our thesis, we consider the case of an aggregated scheduling with four different service classes. We explore the benefits of flow aggregation particularly the possibility of accurate deterministic delay bounds calculus. We conceive and implement an adaptive WFQ scheduling. We then evaluate its performances in respect to the Quality of Service constraints of real time applications, mainly delay and jitter.

The develped algorithm, and the undergone simulations were implemented and tested with the OPNET Modeler environment.

Experiences showed that deterministic delay bounds were too large in comparison with the real delays encountred by packets in a real netwroks under the same circumstances. We conclude that we can't rely on these bounds to implement an efficient admission control mechanism.

The obtained simulation results, related to the adaptive scheduling policy, show the benefits of such agility of the network when dealing with a high and variable charge on a multiservices network.

Keywords : Multiservices Networks, Scheduling, WFQ, Aggregation, Traffic Management, QoS, delay bounds.

# Table des matières

# Liste des tableaux

# Liste des figures

# Liste des sigles et abréviations

| Acronyme | Description |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| ATM | Asynchronous Transfert Mode |
| BP | Bande Passante |
| CAC | Call Admission Control |
| CL | Control Load |
| CBWFQ | Class Based Weighted Fair Queuing |
| DIFFSERV | Differentiated Services |
| DRR | Deficit Round Robin |
| DRT | Download Response Time |
| EF | Expedited Forwarding |
| FIFO | First In First Out |
| FR | Frame Relay |
| GBPS | Giga bits par seconde |
| GPS | Generalized Processor Sharing |
| GRC | Guaranteed Rate Clock |
| GRS | Guaranteed Rate Scheduling |
| GS | Guaranteed Services |
| IEC | Commission Electronique Internationale |
| IEEE | Institue of Electrical and Electronics Engineers |
| INTSERV | Integrated Services |
| IP | Internet Protocol |
| IPV4 | IP version 4 |
| IPV6 | IP version 6 |
| LAN | Local Area Network |
| MBPS | Mega bits par seconde |
| MPEG | Moving Pictures Expert Group |
| MPLS | Multi Protocol Label Switching |
| MTU | Maximum Transfer Unit |
| NTR | Non Temps Réel |

| | |
|---|---|
| NWC | Non Work Conserving |
| OPNET | Optimum Network |
| OSI | Organisation Internationale de Normalisation |
| PGPS | Packet-by-packet Generalized Processor Sharing |
| PQ | Priority Queuing |
| PRT | Page Response Time |
| QDS | Qualité de Service |
| QOS | Quality of Service |
| RED | Random Early Detection |
| RR | Round Robin |
| RNIS | Réseau Numérique à Intégration de Services |
| RSVP | Ressource Reservation Protocol |
| RTCP | Real Time Transport Control Protocol |
| RTP | Real Time Transport Protocol |
| SR | Sous Réseau |
| TCP | Transmision Control Protocol |
| TCO | Traffic Class Octet |
| TDM | Time Division Multiplexing |
| TOS | Type Of Service |
| TR | Temps Réel |
| UDP | User Datagram Protocol |
| UIT | Union Internationale des Télécommunications |
| VC | Virtual Clock |
| VOIP | Voix sur IP |
| VOATM | Voix sur ATM |
| VOFR | Voix sur FR |
| WC | Work Conserving |
| WAN | Wide Area Network |
| WFQ | Weighted Fair Queuing |
| WRED | Weighted Random Early Detection |
| WWW | World Wide Web |

# Dédicaces

À mes très chers parents,

À mes très chers parents que je ne peux jamais assez remercier pour les sacrifices qu'ils ont fait et qu'ils continuent de faire pour m'élever et m'éduquer. C'est leurs sacrifices qui me permettent de réaliser mes rêves et d'être la personne que je suis où je suis.
Je vous aime.
Je vous demande pardon pour les sacrifices auxquels vous êtes prêts à faire face rien que pour nous, vos enfants.
Avec la bénédiction de Dieu, je tiendrai mes promesses.

À tous les membres de ma famille proche, à tous ceux qui s'y reconnaissent.

# Remerciements

*Une fois ce travail accompli, je ne pouvais m'empêcher de manifester ma reconnaissance à mon directeur de recherche pour son appui durant mon projet de recherche.*

*Je tiens à remercier vivement le professeur* BRIGITTE JAUMARD, *ma directrice de recherche pour m'avoir offert l'opportunité de travailler au sein de l'équipe de la Chaire du Canada en Optimisation des Réseaux de Communications, aussi pour ses directives et son appui financier.*

*Je tiens aussi à exprimer mes remerciements à l'équipe du CRT, et spécialement à notre responsable des laboratoires informatiques,* DANIEL CHARBONNEAU, *pour sa gentillesse ainsi que sa disponibilité lors des moments critiques.*

*Je remercie toutes les personnes qui ont contribué directement ou indirectement à la réalisation de ce travail. Je penserai à* RÉDOUANE *avec qui j'interagissais lorsqu'il s'agissait de résoudre des problèmes avec OPNET.*

*Finalement, j'aimerais remercier tous mes amis de l'université et mes amis intimes pour leurs encouragements, leurs conseils, leurs soutiens ou tout simplement pour leur présence en particulier :* CHAR, DESPINA, HÉLÈNE, NOURCHÈNE, RAHA, YOSR, HICHEM, IDRISS, IKBEL, SABER, SAMI, TARIK...

# Chapitre 1

# Introduction

## 1.1 Motivations et Objectifs

Durant ces vingt dernières années, l'Internet a décidémment révolutionné le monde. Avec le grand nombre d'utilisateurs venant s'ajouter chaque année pour concurrencer l'utilisation de services tels que la navigation sur le web, le FTP, le Telnet ou le courriel, tout porte à croire que les capacités de l'Internet ne se trouverait en aucun cas compromises vu qu'on estime à 100% la croissance annuelle de ses capacités.

Le rythme accéléré de sa fulgurante vulgarisation lui a tout de même coûté cher. Il doit, en effet, répondre à de lourdes responsabilités pour lesquelles il n'était pas préalablement conçu. En effet, on voit son infrasctructure exploitée par de nouvelles applications et services exigeants un haut degré de qualité de service. Ce sont les applications multimédias et temps réel, actuellement très à la mode, et profitant de cette grande popularité. On cite la téléphonie, la vidéo sur demande, la visio-conférence, etc. Ce déploiement crée une incompatibilité entre ce que peut offrir Internet et ce que demandent ces applications. Cette incompatibilité est due au fait qu'Internet n'assure pas

une définition ainsi qu'une maintenance à priori du niveau de qualité de service des applications de ce genre et peut juste offrir une seule classe de service de type *Best-Effort*.

La panoplie de nouvelles solutions et technologies proposées a le potentiel de faire évoluer les choses. Les déploiements de nouvelles architectures telles que le DiffServ ou la combinaison de IntServ avec le protocole de réservation RSVP, s'avèrent des options intéressantes. Néanmoins, malgré le grand nombre de travaux effectués dans ce domaine, on croit encore que les problèmes posés par les services temps réels à travers l'Internet n'ont pas encore été bien résolus.

La contrainte de délai est la notion la plus importante quand il s'agit de satisfaire du trafic temps réels. Il convient alors au futur Internet de pouvoir effectuer certains traitements pour arriver à respecter cette contrainte essentiellement au niveau de l'ordonnancement des paquets. Pour assurer la qualité d'un flot temps réel, il est possible de vérifier au préalable si les délais de bout en bout seront respectés ou s'ils ne le seront pas.

Néanmoins, le nombre de flots dans l'Internet est très important et en constante croissance. Par conséquent, le fait d'effectuer des traitements rapides et adéquats par flot devient une tâche complexe, voir impossible à réaliser. L'agrégation des flots en un nombre plus restreint de flots devient une option fort intéressante. Mais il s'avère difficile à effectuer des calculs préalables sur les délais que peuvent mettre les paquets d'un même groupe de flots vu que ces derniers n'ont probablement pas les mêmes caractéristiques.

Étant donné que le délai d'attente dans les files est la composante la plus critique lors du calcul du délai de bout en bout, il faudra alors penser à trouver des solutions lors de l'ordonnancement des paquets tout au long du chemin de communication. Cet ordonnancement devrait tenir compte de l'information sur le délai et devrait, ainsi, être basé sur des indicateurs temps réel se rapportant à d'éventuels dépassements des bornes sur les délais acceptés pour pouvoir s'ajuster et, ainsi, maintenir un niveau de qualité de service acceptable.

Notre objectif pour ce mémoire était d'explorer l'opportunité de garantir la qualité de service aux applications temps réel à travers le calcul préalable des bornes déterministes sur le délai de bout en bout pour le cas des flots agrégés. Notre vision s'est ensuite orientée vers une approche plus réaliste consistant à concevoir une politique d'ordonnancement qui s'ajuste dans le cas de mise en péril de la qualité des applications temps réels.

## 1.2 Plan du mémoire

Ce mémoire est organisé comme suit :

Dans le chapitre 2, on introduit les principales infrastructures et architectures disponibles actuellement dans l'Internet, le réseau multiservices par excellence, ainsi que les principales exigences des applications temps réel.

Dans le chapitre 3, une vue d'ensemble sur les différents types de mécanismes de QdS pouvant être élaborés pour respecter ces exigences est présentée, avec une mise en valeur, dans le chapitre 4, des fonctions d'ordonnancement et d'agrégation des flots.

Le chapitre 5 est dédié à une revue de la littérature des travaux principalement effectués pour le calcul de bornes déterministes de délai pour le cas des flots agrégés.

Le chapitre 6 explique le mécanisme implanté avec OPNET pour la mesure des délais en ligne et l'ajustement, en conséquence, des poids des différentes classes de services afin d'éviter la dégradation du niveau de qualité relatif aux applications temps réel.

Dans le dernier chapitre, des résultats sont présentés quant à la non consistence des bornes déterministes quant il s'agit d'un traitement agrégé des flots. On s'intéresse par la suite à la comparaison de quelques politiques d'ordonnancement proposées dans la littérature dont celle utilisée dans l'Internet actuel (FIFO). On évalue finalement les performances réalisées par notre politique adaptative ainsi que par quelques unes de ses variantes.

# Chapitre 2

# Réseaux multiservices

## 2.1 Pourquoi des réseaux multiservices ?

Les réseaux traditionnels ont été conçus initialement sans aucune différentiation de trafic. En effet, les applications prioritaires ou sensibles à certains critères comme le délai ne sont pas traitées de façon particulière. Cela ne poserait aucun problème si la bande passante était illimitée. Mais la réalité est toute autre vu que la bande passante permet le transport d'une quantité limitée de trafic, ce qui engendre un problème de contention entre les différentes applications qui sont supportées aujourd'hui dans les réseaux.

De nos jours, on parle de liens pouvant avoir des débits en bande passante pouvant aller à des centaines de Mbps voir même quelques Gbps. C'est une évolution grandiose de la technologie des liens cablés, avec l'apparition en cours de route d'autres types de medias tout aussi performants comme la fibre optique. Cela n'a rien à voir avec les débits en bande passante offerts au début des années 80 où l'on ne pouvait même pas dépasser quelques Mbps. Cependant, le type d'applications a lui aussi suivi cette évolution puisqu'on se retrouve maintenant avec des applications multimédias très gourmandes en matière de bande passante comme : la voix, la vidéo. Voila donc qu'on se retrouve en

pésence d'applications qui s'arrachent le droit d'envoyer du trafic sur le réseau alors que les ressources ne sont pas suffisantes pour tous. Les réseaux traditionnels comme tels, sans aucune différentiation entre ces applications, ne peuvent faire grand chose face à cette diversité et montrent rapidement leur limites. On se retrouve avec un dilemne qu'est : comment faire pour permettre de traiter et satisfaire toutes ces demandes sans dégrader la qualité recherchée par chacune de ces applications ? C'est pourquoi l'interêt se porte actuellement sur un nouveau type de réseaux implantant des mécanismes capables de gérer efficacement cette panoplie d'applications et qu'on appelle les *réseaux multiservices.*

## 2.2 Infrastructure multiservices

Nous présentons ci-dessous de façon générale les réseaux multiservices à commutation de paquets, et nous posons également le problème traité en précisant les objectifs visés.

### 2.2.1 Généralités

Les réseaux multiservices sont classiquement modélisés par un graphe orienté G = (V, E) avec :

- V : ensemble des nœuds correspondants à des points de transit de l'information (routeurs, serveurs...).
- E : ensemble des arcs correspondant aux liens (exemple : câbles) assurant le transfert de l'information entre deux nœuds du réseau.

Comme mentionné ci-dessus, les nœuds peuvent être de deux types : internes ou externes. Chacun de ces nœuds assure plusieurs des fonctions suivantes pour la gestion du trafic : le contrôle d'admission, la classification des paquets, le modelage du trafic, le routage, l'ordonnancement des files d'attentes, etc. Ces nœuds sont connectés les uns aux autres

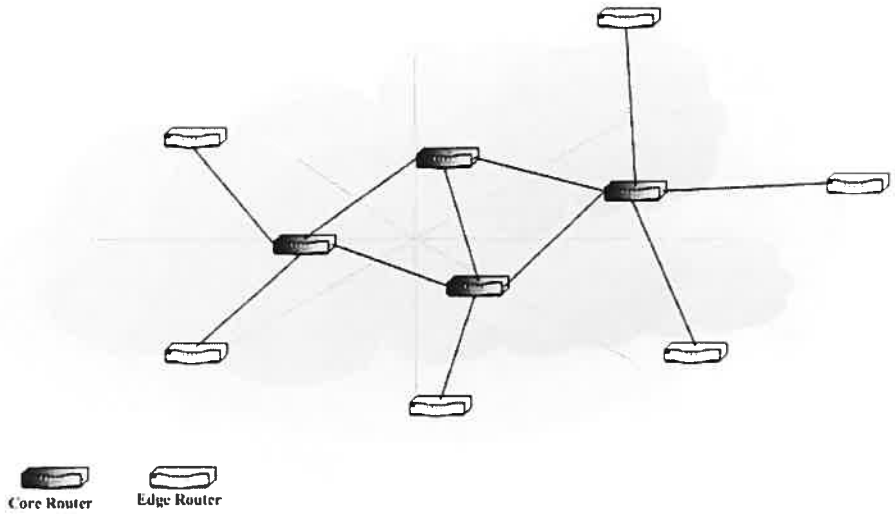



FIG. 2.1 – Architecture d'un réseau : nœuds internes et externes

par des liens qui sont généralement des liens cablés notamment de la fibre optique ou du cuivre. Puisque l'information transportée peut être de nature diverse (courriel, page web, voix, vidéo, visioconférence,...) et comme on pourrait avoir plusieurs applications de même type, alors on parle de *flux* ou *session* dont voici une définition [53] :

> *Un flux peut être défini de plusieurs manières. Une façon courante de procéder est de le définir comme une combinaison des adresses et des ports des sources et des destinations, et d'un identifiant de session. Un flux peut être plus généralement défini comme étant l'ensemble des paquets générés par une application déterminée ou reçus sur une interface entrante chez le destinataire.*

En prenant en considération la notion de flux ci-dessus expliquée, les nœuds peuvent être représentés de la manière simplifiée suivante :

A, B, C, D, E et F sont les liens cablés connectés au nœud. Les flux de 1 à 6 correspondent à de l'information transportée par les différents liens connectés au nœud. Par décision de la fonction de routage, ces flux passeront d'un lien entrant vers un autre lien appelé lien de sortie. Par exemple le flux 1 entre par le lien A et sera acheminé vers le lien D advenant la procédure de routage adéquate. Lorsqu'un lien est connecté à un

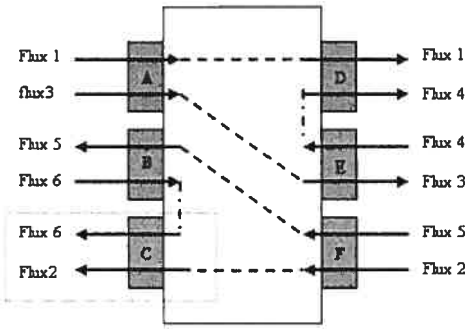



FIG. 2.2 – Passage des flots à travers un nœud

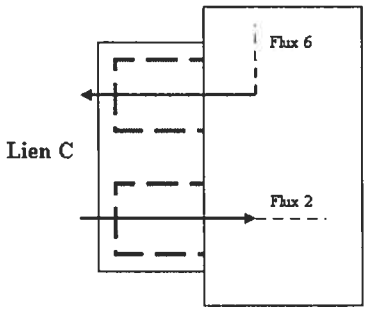



FIG. 2.3 – Passage des flots à travers une interface

nœud, deux interfaces sont créées dans ce dernier : une interface d'entrée pour recevoir les flux provenant de ce lien, et une interface de sortie pour acheminer les flux allant vers ce lien (voir figure 2.3).

### 2.2.2 Infrastructures des réseaux

Dans les réseaux multiservices, le réseau de transport peut être basé sur IP [59], FR ou ATM. Il existe aussi d'autres variantes qui combinent, par exemple, IP avec FR ou encore avec ATM. Si on considère l'infrastructure traditionnelle de téléphonie, on devrait avoir à l'extrémité d'un réseau multiservices, des composantes spéciales appelées *passerelles de voix.* Ces composantes permettront de changer l'information voix

traditionnelle, et probablement analogique, vers une autre forme adequate (numérique) pour le transport dans le réseau multiservice. Ces informations vont ensuite transiter sur le réseau pour atteindre la passerelle voix destinataire qui va à son tour effectuer le changement inverse et acheminer l'information à la partie appelée.

On se retrouve avec un choix à faire entre trois technologies principales pour le déploiement de voix : voix sur IP (VoIP), voix sur Frame Relay (VoFR) ou voix sur ATM (VoATM). VoIP est une technologie de la couche 3 du modèle OSI qui peut très bien renforcer FR et ATM dans leur rôle de la couche 2 pour le transport des paquets.

Pour VoIP il y a des protocoles de communications ou de signalisation qui entrent en compte dans le processus d'échange de paquets voix. On pourra citer quelques protocoles comme H.323, MGCP et SIP [2].

Une des raisons qui militent en faveur de VoIP contre VoFR et VoATM est l'interfaçage facile de IP avec d'autres applications voix et multimédias. FR et ATM sont des technologies de réseaux de type étendus (WAN) qui sont très robustes et efficaces en matière de bande passante. Cependant ces deux technologies sont limitées car elles ne sont pas implantables dans le cadre des réseaux locaux (LAN) ou des terminaux. Finalement VoIP est la technologie la plus utilisée pour le transport de la voix.

- IP [42] C'est un protocole non orienté connection. Il utilise des protocoles des couches supérieures comme UDP [60] et TCP [61] pour permettre à tout type de flots dont la voix de transiter sur le réseau. IP possède des protocoles de signalisation, d'adressage et de routage assez robustes. RTP [50] est un protocole très utilisé lors du déploiement d'applications de type temps réel comme la voix et la vidéo. L'un des avantages de IP est qu'il est un protocole de la couche 3, ce qui fait qu'il peut profiter des forces des protocoles de la couche 2, entre autres de FR et de ATM. En effet, on peut très bien envoyer de la voix sur IP (VoIP) en utilisant des réseaux basés sur FR ou ATM. Comme on a dit, IP ne se limite pas au WAN's mais peut tout aussi bien être supporté tout au long des LAN's et jusqu'aux terminaux. Cela facilite énormément le déploiement de nouvelles entités matérielles basées sur IP comme les téléphones IP, les PBX basés sur IP, mais aussi toute autre nouvelle application Internet se lançant à partir d'un terminal.

- FR [42] Frame Relay est un protocole WAN à hautes performances opérant au niveau des couches physique et liaison de données du modèle OSI. FR a été conçu pour être utilisé au travers des interfaces RNIS. FR est un exemple de technologie de commutation de paquets. Les réseaux à commutation de paquets permettent aux stations d'extrémité de partager entre elles le médium et la bande passante disponibles de manière dynamique. Les deux techniques suivantes sont utilisées dans la technologie de commutation de paquets :
  - longueur variable des paquets ;
  - le multiplexage statique.
- ATM [42] À l'origine, ATM a été concu pour manipuler du trafic sensible au délai comme la voix. ATM est orienté connexion. Utilisant des cellules de taille fixe au lieu des paquets de taille variable, la fonction de commutation d'ATM est optimisée pour supporter l'établissement de connections à haut débit. ATM assure bien à ce genre de connections les délais qu'elles souhaitent. ATM est généralement deployé dans l'épine dorsale (backbone) du WAN là où les liens sont à très haut débit, mais jamais deployé là où les liens sont à faible débit.

## 2.3 Exigences de conception des réseaux multiservices

La diversité des applications dans les réseaux multiservices (courriel, web, video, voix, etc) fait que chacune d'entre elles bénéficie d'un traitement particulier différent. Ainsi, les sessions peuvent se voir associer des poids qui, dans la pratique, seront employés pour les différencier ou encore caractériser leurs priorités relatives. Ces poids peuvent être arbitrairement fixés par le gestionnaire du réseau pour assouvir ses besoins de conformité à certains critères de QdS detaillés ci-dessous. Toutefois, on constate dans la littérature que le poids est souvent pris de manière à ce qu'il soit égal au débit moyen des sessions auxquelles il est associé [1] [3] [4]. Ainsi, si un flot consomme le tiers de la bande passante d'un lien, alors son poids est égal à $\frac{1}{3}$. Par ailleurs, un trafic est

caractérisé par des paramètres de QdS auxquels il doit se conformer. Nous décrivons ci-dessous les paramètres les plus fréquents.

- ➢ La limite de bande passante (bandwith bound) : permet de spécifier le minimum de bande passante que la connexion doit recevoir du réseau.
- ➢ La limite de délai (delay bound) : Cette limite peut être une limite statistique obtenue à partir de certains paramètres comme le *pire délai* (worst case) ou *le délai de bout en bout moyen* (average delay) [1]. Elle peut aussi être calculée de manière déterministe [3].
- ➢ Une limite pour la gigue (jitter) : C'est une valeur qui consiste en la différence entre le plus grand et le plus petit des délais reçus par les paquets tout au long d'une connexion donnée.
- ➢ Une limite pour la perte spécifiant une limite sur la fraction de paquets qui peuvent être perdus lors de la connexion en question.

Les services sont différenciés, et les poids sont affectés, suivant leur niveau de rigueur en QdS (QoS strictness) ; celui-ci décrit le degré auquel les services peuvent être liés à des exigences portant sur les paramètres ci-dessus.

## 2.4 Architectures Réseaux

Compte tenu des critères définis dans la section 2.3, trois niveaux de QoS de bout en bout sont généralement offerts. Les trois pourraient être présents sur un même réseau hétérogène.

- ✓ Un service best-effort. Un tel service est dépourvu de toute forme de QdS. Les connections se font toutes sans garantie de QdS et la gestion des files d'attente s'effectue conformément au modèle FIFO (premier entré premier servi) qui ne fait aucune différence entre les flux. L'Internet actuel fait partie de cette catégorie et offre un service best-effort.

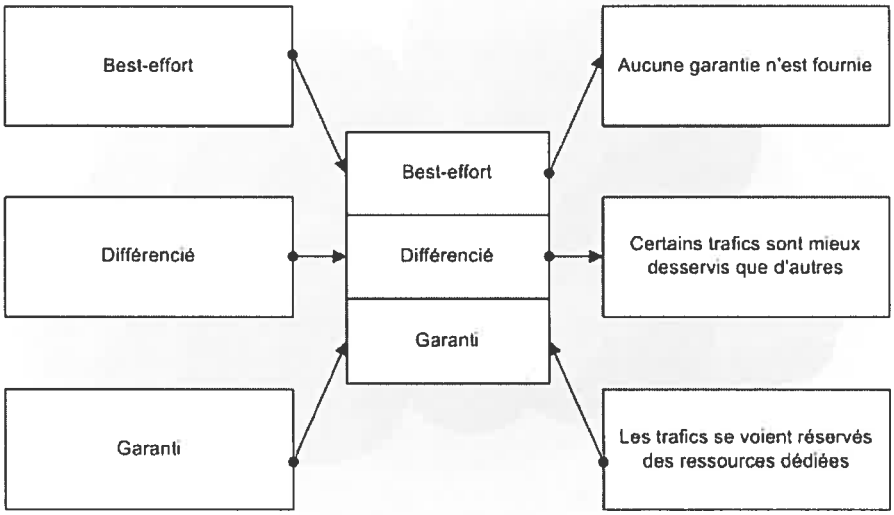



FIG. 2.4 – Trois niveaux de QoS

- ✓ Un service garanti (Hard QoS) avec une garantie sûre et rapide. Il s'agit de réserver des ressources réseau pour un trafic particulier. Ce niveau de service peut être fourni à l'aide de protocoles tels que RSVP. Une architecture basée sur ce type de services est IntServ (voir section 2.4.1).
- ✓ Un service différencié (Soft QoS). La différentiation veut que certains trafics soient mieux traités que d'autres avec, par exemple, une prise en charge plus rapide, une allocation d'une bande passante moyennement plus large, et un taux de perte inférieur. Il s'agit d'une préférence statistique et non d'une garantie sûre et rapide [53]. Ce niveau de service n'est possible qu'avec, tout d'abord, une classification des trafics, puis, la mise en œuvre de mécanismes de QdS tels que l'ordonnancement avec priorités (PQ), WFQ et bien d'autres. Une architecture basée sur ce type de services est DiffServ (voir section 2.4.2).

### 2.4.1 Architecture à services integrés (IntServ)

L'architecture IntServ [40] a été proposée par le groupe de travail IntServ de l'IETF [48] en 1997 pour être utilisée de manière complémentaire à RSVP.
Deux classes de services ont été définies :

- ➢ classe à charge commandée (CL) : fournit une garantie qualitative en évitant les conditions de congestion. Il n'y a aucune garantie explicite quant au délai ou même à la perte associés au service. Elle ne peut pas fournir des délais quantitatifs qui sont nécessaires pour des applications TR.
- ➢ classe à service garanti (GS) : fournit des garanties déterministes de QdS. Cela peut être réalisé par l'utilisation de RSVP et de certaines politiques d'ordonnancement de type GRS tels que WFQ.

Il convient de noter que GS utilise des bornes déterministes lors de l'établissement d'une connexion, ce qui peut être nécessaire pour seulement quelques classes de services TR. En effet, cela s'avère très conservateur et inefficace pour la plupart des classes de services TR. Le souci de coût et d'extensibilité (*scalability*) de RSVP ainsi que d'un ordonnancement par flot dans les réseaux à haut débits a empêché l'acceptation et le déploiement de ce modèle en pratique.

### 2.4.2 Architecture à services différenciés (DiffServ)

L'architecture DiffServ [39] a été définie par le groupe de travail DiffServ de l'IETF pour éviter le coût et le problème d'extensibilité de IntServ. DiffServ a été proposée pour fournir la différentiation de services dans l'infrastructure actuelle d'Internet sans large déploiement de RSVP et des algorithmes d'ordonnancement par flots.
Des classes de service avec différentes priorités sont définies en utilisant le champs ToS (IPv4) ou TCO (IPv6). Au niveau des routeurs à l'extrémité du réseau, chaque flot est régulé et marqué selon le profil de l'utilisateur. Au cœur du réseau, les routeurs

n'ont aucune connaissance des flots individuels et sont seulement concernés par le trafic global.

# Chapitre 3

# Mécanismes de QdS

Dans cette partie on s'intéressera à définir certaines composantes importantes utiles à l'approvisionnement et au contrôle du niveau de la QdS et qui sont appliquées aux réseaux multiservices. Les quelques outils, auxquels nous nous intéressons de plus près dans ce mémoire, ont pour la plupart fait l'objet de publications sous forme de RFCs. Il existe, en effet, bien d'autres outils d'approvisionnement, de contrôle et de gestion de la QdS.

## 3.1 Classification des paquets

La classification est un mécanisme utilisé pour sélectionner la classe de service d'un flot de trafic donné. Elle se fait en se basant sur une information contenue dans l'entête des paquets de ce flot. Le classificateur est généralement facile à implanter mais cela dépendra du modèle de service choisi par le fournisseur de services. Cette évidence vient du fait que les méthodes de classification sont propres aux modèles de services et celles-ci diffèrent d'un modèle à un autre. En général, les modèles suivants peuvent être pris en compte [40].

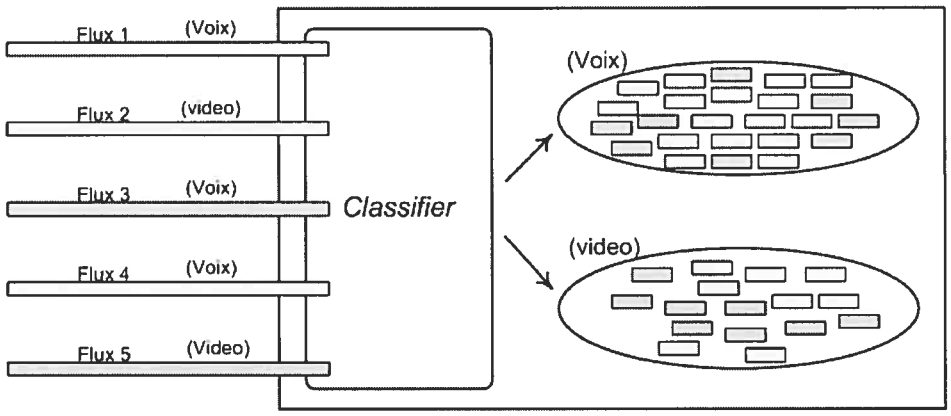



FIG. 3.1 – Procédure de classification

- ✓ L'utilisateur choisit une classe de service parmi les classes de services à sa disponibilité.
- ✓ L'application choisit automatiquement un service pour chaque agrégat de flux, pour chaque flux ou, pour chaque paquet.
- ✓ Le réseau sélectionne la classe de services en se basant sur l'information qu'il possède sur l'application.
- ✓ Le réseau choisit une classe de services en se basant sur le contrat établi entre le client et son fournisseur de services, cela sans tenir compte du type d'application.
- ✓ Une combinaison de tous les modèles précédents.

La première approche est qu'il doit y avoir une méthode pour transmettre les informations relatives à la classification entre le client et le réseau : si le client n'est pas satisfait du service fourni par cette classe pour tous les flux, alors un mécanisme de signalisation sera nécessaire pour l'échange des informations relatives à la classification. Cette méthode ne peut permettre l'utilisation simultanée de plusieurs classes sans l'implantation de mécanismes additionnels.

Toutefois, la deuxième approche apparaît plus pratique. C'est l'application même qui va choisir la classe de service qu'elle croit la plus appropriée pour ce genre d'application. Une application de type *vidéo conférence* peut choisir, par exemple, la classe de service ayant le plus haut niveau d'importance. Mais si le client n'est pas autorisé à utiliser ce genre de classe de services, alors le fournisseur choisit la classe définie par défaut pour

ce genre d'applications.

Le problème avec cette deuxième approche c'est qu'il faut prévoir des mécanismes pour incorporer l'information relative à la classe de services avant la transmission des données. Mais si ce genre de mécanisme n'existe pas, alors c'est le rôle des routeurs se trouvant aux frontières du réseau de faire ce choix : c'est la troisième approche [39]. Il est aussi possible qu'un client particulier n'ait accès qu'à une seule classe de services déjà prédéfinie dans son contrat avec le fournisseur d'accès. Dans ce cas, la classification est facile à faire dans le sens où tous les paquets d'un client particulier seront classifiés dans une seule classe de services. Finalement, il est possible de combiner toutes ces approches. Pour classifier les paquets dans un réseau IPv4 on utilise la *précédence IP*.

### Précédence IP

La précédence IP utilise les 3 bits de précédence du champs ToS (*Type of Service*) de l'entête d'IPv4 pour spécifier la classe de services de chaque paquet. Ces 3 bits nous

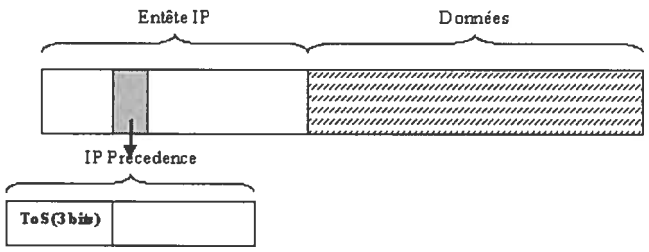



FIG. 3.2 – Paquet IPv4

permettent de bénéficier d'un champ d'action de huit valeurs. Six sont réservées pour pouvoir répartir le trafic sur six classes de services. Les deux autres sont réservées pour une utilisation interne du réseau. Compte tenu de la valeur des 3 bits de précédence IP, d'autres mécanismes de QdS tel l'ordonnanceur vont pouvoir effectuer un traitement différent sur les différentes sessions de trafic. Les trois bits correspondants aux valeurs

binaires 32, 64 et 128 du champs ToS de l'entête IPv4 constituent les bits déployés pour la précédence IP. Ces bits sont utilisés pour fournir une priorité du paquet allant de 0 à 5. Ce sont les valeurs 6 et 7 qui seront employées pour des raisons internes. Comme

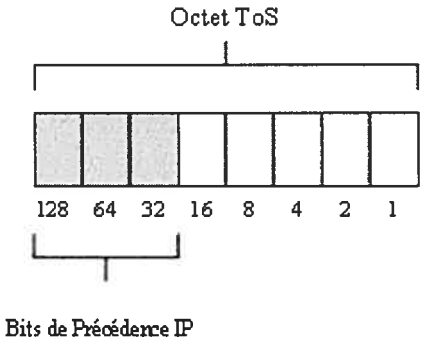



FIG. 3.3 – Champs ToS de l'entête IP

on a vu, trois des huit bits du champs ToS sont utilisées pour la précédence IP. De ce fait, il faudrait absolument distinguer ces bits là du reste des bit du champs ToS. Dans l'exemple de la figure ci-dessous, la valeur de la précédence IP (trois premiers bits) est égale à 011 et correspond à une précédence IP de 3.

Cependant, si on examine tout l'octet ToS à la fois, on aura une valeur de 96 qui

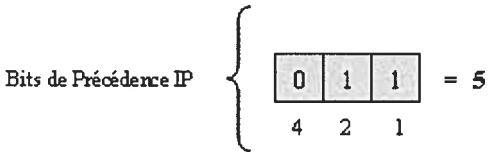



FIG. 3.4 – Exemple de trois bits de précédence IP

correspondra à 01100000.

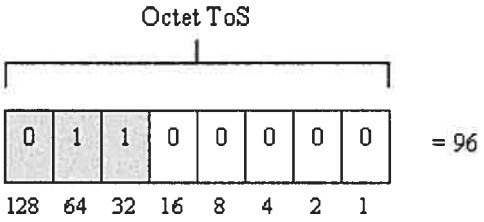



FIG. 3.5 – Exemple de champs ToS

## 3.2 Marquage des paquets

Supposons que dans une même classe, on puisse avoir différents niveaux d'importance. L'objectif du marquage est d'établir une correspondance entre les paquets d'un flux ou d'un agrégat donné avec l'un des niveaux d'importance disponibles pour la classe du flux en question. Cela veut dire qu'à la fin de ce mécanisme, le paquet appartenant toujours à la même classe, sera affecté à un autre niveau d'importance qu'il soit supérieur ou inférieur.

## 3.3 Lissage ou modelage de trafic

La principale idée du modelage est que si un paquet doit être marqué de nouveau ou ajusté à un niveau d'importance moins élevé, alors une autre alternative consiste à modeler préalablement le trafic de façon à éviter un re-marquage des paquets. Ce modelage peut être fait par l'utilisateur comme il peut être fait à l'extrémité et au coeur du réseau par l'opérateur même. Toutefois, cette technique peut être vue comme une façon de réguler le trafic et ainsi éliminer ses grandes variations. Cela voudrait dire aussi un meilleur traitement du trafic à l'intérieur du réseau. Notons que le modelage du trafic nous amène à réfléchir à une discipline de service (service discipline) de type *Sans Conservation de Travail (NWC)* qui fait que même s'il y a des paquets prêts à la transmission, ils ne seront pas transmis. La notion NWC est expliquée plus loin dans

ce mémoire. Les mécanismes les plus connus pour rendre les flos conformes sont le *Seau percé (Leacky Bucket)* ( [2] pages 372-377) et le *Seau à jetons (Token Bucket)* ( [3] pages 42-43 et [2] pages 378-379).

## 3.4 Contrôle d'admission

Le contrôle d'admission s'effectue au niveau d'un routeur placé à l'entrée du réseau. À ce niveau on doit décider de l'acceptation ou non des différentes demandes de connexions dont chacune est caractérisée par des paramètres de trafic.
Il s'agit d'accepter suffisamment de connexions pour maximiser l'utilisation des infrastructures et en même temps contrôler la charge du réseau de telle sorte que les différents niveaux de garantie de service demandés soient satisfaits. À chaque requête, on décide si l'occupation du réseau permet de garantir le niveau de qualité de service souhaité par la requête : délai, bande passante, taux de perte, etc...
Dans un contrôle d'admission, on se place généralement dans le cadre où des mécanismes de réservation de bande passante sont utilisés. L'algorithme d'acceptation au niveau du routeur doit être simple, ce qui se traduit par un minimum de calcul typiquement une comparaison avec une valeur critique calculée.

## 3.5 Prévention de la congestion et suppression des paquets

En période de congestion une option est de supprimer les paquets avant leur entrée dans le réseau. Cela se passe aussi bien pour les services utilisant des connections garanties que ceux offrant des services différenciés. Les mécanismes, les lois et les algorithmes à établir pour la réalisation de cette tâche doivent être clairs et bien définis pour ainsi éviter la suppression d'un paquet par erreur et par conséquent éviter la violation du modèle de service convenu entre client et son fournisseur [40].

### 3.5.1 Prévention de la congestion par TCP

TCP prévoit un algorithme de prévention de la congestion comme expliqué dans [63].

Cependant, le problème d'un tel mécanisme offert par TCP est la mauvaise cohabitation entre TCP et UDP. Vu que UDP ne possède pas de mécanisme de comportement adaptatif, la mise en route d'un tel algorithme de prévention de congestion de la part de TCP laisse place à une augmentation de trafic UDP. Il existe aussi un autre problème avec TCP. Lorsque des points de congestion se développent un peu partout dans un réseau chargé ; chacun des flux TCP va détecter des pertes et va declencher un algorithme de prévention de la congestion. Ainsi, de nombreux flux vont repasser en mode *slow start* [64] et tenter de réémettre les paquets perdus, ce qui ne résoudra en rien la congestion déjà présente car le redémmarrage des flux se fera de manière synchrone. Ce phénomène connu sous le nom de *synchronization globale* met en évidence l'importance de nouvelles techniques plus appropriées pour la prévention de la congestion [62].

### 3.5.2 Détection Précoce Aléatoire RED

Cette méthode a pour but de détecter la congestion avant qu'elle ne se produise. Pour se faire, on va :

- ➢ soit jeter aléatoirement des paquets par mesure préventive afin que TCP réduise sa transmission.
- ➢ soit utiliser le flag ECN pour indiquer à TCP que les paquets ont traversé un noeud congestionné.

Le seuil à partir duquel on commence à jeter les paquets, ainsi que le taux de remplissage de la file sont des paramètres configurables par l'administrateur du réseau. Dans un routeur, cela reviendrait à mesurer le taux d'occupation de la mémoire.

Dans [62], on recommande l'implantation du mécanisme RED dans les routeurs et la

définition de mécanisme de gestion des flux non régulés c'est-à-dire sans modelage du trafic au niveau des sources.

### 3.5.3 Détection Précoce Aléatoire Pondérée WRED

Cela revient au même mécanisme que RED sauf qu'en plus on permet de déterminer quel trafic on jette grâce à la prise en compte des bits de précédence IP du champs ToS de IPv4 par les routeurs. Cela permettra d'éviter la monopolisation des ressources par certaines classes de trafic. Cette pondération gérée via ce champ permet au réseau de demander aux flux de trafic moins prioritaires de s'adapter au profit des flux de trafic plus prioritaires.

## 3.6 Routage des paquets

Vu l'importance de la QdS, certains protocoles de routage, comme MPLS, traitent ce problème. MPLS, également appelé commutation par étiquette (*label switching*) ou encore commutation par marquage (*tag switching*), contient les mécanismes qui permettent d'interopérer avec la signalisation RSVP et la précédence IP. En effet, l'en-tête MPLS contient un champ sur trois bits pouvant être utilisé pour signaler la priorité du trafic. Il peut également servir à établir une correspondance entre les flux et les classes de trafic le long des chemins de commutation existants pour obtenir de la QdS sur une portion à commutation d'étiquettes d'un réseau [42] [65].

## 3.7 Autres mécanismes de QdS

### 3.7.1 Ordonnancement

Si on considère un ensemble de paquets dans la file d'attente d'un serveur, quelle sera la politique de service du serveur (ordonnanceur) pour déterminer le prochain paquet à être traité ?

En fait, les algorithmes d'ordonnancement ou de mise en file d'attente entrent en action et deviennent importants lorsque surgit le phénomène de la congestion. En effet, il faut traiter les données relatives à une connexion sans perturber les autres connections. Cela revient à essayer de respecter la QdS de chaque connexion. Evidemment, en cas de congestion cela devient peu évident et la tâche de l'ordonnanceur devient primordiale mais délicate. Les algorithmes d'ordonnancement auront le rôle de trier le trafic entrant dans plusieurs files d'attente et ensuite de déterminer une méthode qui l'ordonne selon des priorités et des contraintes de QdS sur le lien de sortie (figure 3.6). On retrouve plusieurs implémentations de ces algorithmes, chacune a été généralement conçue pour résoudre un problème de trafic réseau et un effet particulier sur les performances réseaux.

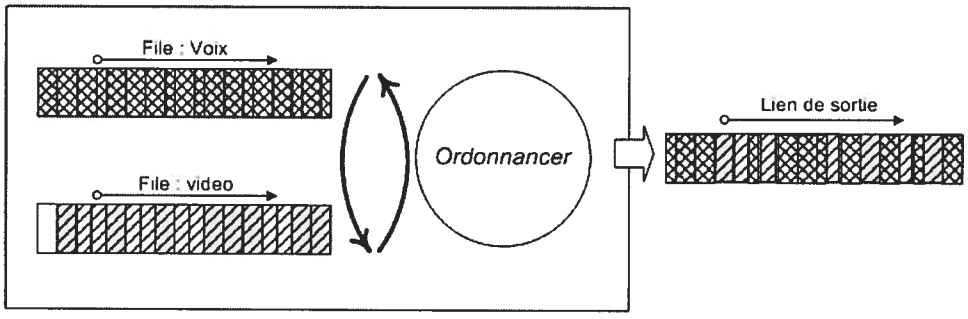



FIG. 3.6 – Procédure d'ordonnancement

### 3.7.2 Agrégation

Généralement les flots sont traités séparément ce qui pose un problème d'extensibilité (IntServ par exemple) vu l'expansion sans cesse grandissante du volume de trafic transitant dans le réseau. Ici, le trafic est divisé en un nombre restreint de classes de services et les ressources sont ainsi allouées par classe de service. On parle alors de regroupement de ces flots en un nombre restreint de classes ou tout simplement d'agrégation des flots. En jouant sur des paramètres comme les priorités ou les poids des classes, le contrôle d'admission, l'allocation de la proportion de bande passante, etc, on peut aboutir aux niveaux de performance et de QdS désirés.

# Chapitre 4

# Revue de la littérature : ordonnancement et agrégation

## 4.1 Ordonnancement

Dans le cas des réseaux multiservices, le rôle de l'ordonnancement est assez critique. Il s'agit de faire cohabiter plusieurs types de trafics à caractères différents. Par exemple le trafic *Best Effort* est un trafic dont les performances sont dites élastiques (s'adaptent aux ressources existantes). En effet, les clients de type *Best Effort* n'ont aucune QdS spécialement en terme de garanties sur le délai et la gigue. Les applications qui ont besoin de QdS forment un deuxième type de trafic aux exigences opposées.
Vu la nature concurrentielle des trafics existants dans les réseaux multiservices, il est nécessaire de concevoir puis proposer des solutions optimales pour leur cohabitation. Ceci est mieux illustré dans la section suivante à travers la panoplie de disciplines d'ordonnancement proposées dans la littérature. Nous présenterons plus loin dans ce chapitre quelques unes de ces politiques.

### 4.1.1 Attentes vis à vis d'une politique d'ordonnancement

Lors de la conception d'une politique d'ordonnancement, notre but est de disposer d'une politique satisfaisant les quatre points suivants parfois contradictoires dans le contexte des réseaux multiservices. Aucune discipline d'ordonnancement ne peut satisfaire pleinement ces propriétés. Si on favorise une propriété, c'est au détriment des autres. Ces quatre points ont été détaillés dans le chapitre *scheduling* de [1] et ce qui suit en est d'une grande inspiration.

- ➢ Implémentation aisée.
- ➢ Bornes de performances pour les connections QdS.
- ➢ Facilité et efficacité du contrôle d'admission.
- ➢ Equité entre les connections.
- ➢ Protection entre celles-ci.

**Facilité d'implémentation.** C'est un besoin demandé pour les connections de type services garantis et best-effort. Le besoin d'une politique implémentée à l'aide d'un nombre restreint d'opérations pas trop compliquées est une issue très importante lors de la conception d'une politique d'ordonnancement. En effet, dans les réseaux à haut débit, la tâche d'un nœud est de choisir lors du départ d'un paquet particulier, le prochain paquet à servir et cette décision doit être effectuée très rapidement. Une particularité désirée est que le nombre d'opérations à implémenter doit être le plus indépendant possible du nombre de connexions à servir.

**Bornes de performance.** Lors de la mise en œuvre de la politique d'ordonnancement, on doit être en mesure de pouvoir garantir certaines bornes de performance pour chaque connexion comme par exemple le délai de bout en bout moyen ou encore le pire délai. Comme déjà mentionné, les paramètres de performance les plus connus sont : la bande passante, le délai, la gigue et la perte.

Ainsi, dans le cas où le modèle de service des paquets est géré par l'opérateur, ce dernier peut garantir ces bornes de performances soit, en réservant tout simplement certaines ressources au moment même de l'établissement de la connexion (Hard QoS), ou bien, en allouant une proportion fixe de bande passante bien avant l'établissement de la connexion (Soft QoS). Quant au client, il doit accepter de limiter son trafic à une certaine borne.

**Facilité et efficacité lors du contrôle d'admission.** Une politique d'ordonnancement doit permettre un contrôle d'admission facile à effectuer. On devrait aussi être capable de décider si on peut respecter les bornes de performance de la nouvelle connexion, et ce, sans compromettre la performance des connections déjà existantes. En plus, la politique ne doit pas mener à une sous utilisation du réseau à cause des régles définies lors du contrôle d'admission.

**Equité.** Ce besoin est requis pour les connections de type best-effort. Comparée aux autres critères cités ci-dessus et couramment employés, cette notion n'est toujours pas clairement définie et sujette à des interprétations diverses. Comme on va le voir par la suite, une politique d'ordonnancement va allouer à chacune des connections ou à un groupe de connections une partie de la capacité de la bande passante ainsi qu'une partie de la capacité de stockage des paquets en sortie dont on dispose. Cette allocation doit être équitable dans le sens où elle devrait satisfaire au critère d'allocation max-min qui, selon Keshav [1], se décrit comme suit.

Le critère max-min est une technique de partage des ressources très utilisée en pratique. Une technique de partage équitable consiste à attribuer les ressources reclamées par un utilisateur ayant une petite demande. Si ces ressources ne sont pas utilisées au total, alors elles seront également distribuées sur le reste des utilisateurs ayant de plus grandes demandes. Le *max-min fair share* est défini comme suit.

- ✓ les ressources sont partagées dans l'ordre croissant de la demande.

- ✓ aucune source ne reçoit plus de ressources que ce qu'elle demande.
- ✓ les sources avec des demandes insatisfaites reçoivent, malgré tout, des portions proportionnelles à leur poids. Le poids $w_i$ se définit comme le rapport de la ressource $R_i$, attribuée par défaut à cette source $i$, sur la totalité des ressources disponibles ($R_i$ / $\sum R$).

Considérons $n$ sources $1, ...., n$ qui ont des demandes de bande passante de $d_1, d_2, ...d_n$, qui à un temps $t$ donné ne sont pas forcément égales. Supposons que ces demandes en ressources sont telles que $d_1 \leq d_2 \leq .... \leq d_n$ et que le serveur a une capacité de $C$. Le critère max-min veut qu'initialement $\frac{C}{n}$ des ressources sont allouées à la source ayant demandé le moins de ressources : la source 1. Cette fraction $\frac{C}{n}$ pourra dépasser les besoins de la source 1. Cela fait que $(\frac{C}{n} - d_1)$ des ressources reste en excès et qu'on devra partager équitablement sur les $(n-1)$ sources restantes. Ainsi les $(n-1)$ sources auront à partager $(\frac{C}{n} - d_1)$ et donc chacune aura $\frac{C}{n} + \frac{\frac{C}{n} - d_1}{n-1}$ au lieu de seulement $\frac{C}{n}$. Cela pourra être supérieur à la demande de $d_2$, mais cependant on pourra continuer le même processus jusqu'à ce qu'une source ne reçoive pas plus que ce qu'elle demande. Comme on le constate, il est très important que le processus d'allocation des ressources commence par s'occuper des sources ayant demandé le moins de ressources, vu qu'ainsi, les sources plus gourmandes restant à satisfaire se verront attribuer probablement encore plus de ressources que la fraction leur étant initialement destinée qu'est $\frac{C}{n}$.

**Protection.** Une politique qui assure la protection entre les différentes sessions est une politique où le mauvais comportement d'une connexion n'affecte pas les autres. Dans une politique sans protection, une connexion peut abuser en envoyant à un taux plus élevé que le taux convenu. Une politique qui n'assure pas de protection va accepter tous les paquets en entrée et les mettre comme par exemple FIFO dans une seule et même file. FIFO est, en effet, un exemple illustratif simple où l'on n'assure pas la protection des connections : une connexion qui envoie plusieurs paquets de suite à un taux très supérieur à celui qui lui est alloué va faire que les délais moyens des paquets suivants et appartenant à différentes autres connexions vont augmenter considérablement. Une

autre alternative pour assurer la protection est de réguler le trafic à la sortie des sources (section 3.3).

La relation entre équité et protection est qu'un ordonnanceur équitable fournit automatiquement de la protection puisqu'il limite une connexion à mauvais comportement à sa portion allouée. Cependant si on protège les connections les unes des autres, on n'est pas sûr de réaliser l'équité. En effet, si on rend conforme le trafic originaire de chaque source à un certain modèle de trafic, les connections seront protégées les unes par rapport aux autres. Ainsi, une connexion ne peut accaparer à elle seule toutes les ressources au dépend des autres connections. Cependant, cela n'assure pas que les portions des ressources allouées à chaque connexion soient équitables.

Chaque politique d'ordonnancement fait un compromis entre ces différents besoins. Dépendamment de la situation, certains de ces besoins deviennent plus importants que d'autres.

### 4.1.2 Choix fondamentaux pour la conception

D'après Keshav [1], il existe quatre principes fondamentaux lors de la conception d'une politique d'ordonnancement.

- ✓ Le nombre des niveaux de priorités.
- ✓ Si chaque niveau est de type work-conserving ou non-work conserving.
- ✓ Le degré d'agrégation des connexions dans un même niveau.
- ✓ La manière dont l'ordonnanceur sert les paquets dans un même niveau.

**Nombre des niveaux de priorités** Une manière de définir une politique d'ordonnancement est celle d'établir plusieurs niveaux de priorités : chaque connexion est associée à un de ces niveaux là. C'est en général un mécanisme facile et simple à implémenter que ce soit au niveau matériel ou logiciel. En présence de $n$ niveaux de priorités, le niveau ayant le plus grand nombre est celui qui correspondra à la connexion de plus haute priorité. Un paquet de la file $k$ n'est servi seulement s'il n'y a aucun paquet en attente

d'être servi dans les niveaux $k+1$, $k+2$, ..., $n$. En prévilégiant les paquets des files à haute priorité, le mécanisme de priorité permet de leur assurer des délais d'attente moyens moindres. Cela bien sûr au dépend des paquets de files d'attente de moindres priorités. Notons que ce nombre de niveaux de priorité est arbitraire. Néanmoins, en pratique, ce nombre dépend du nombre de classes de délais et de services que l'opérateur du réseau veut supporter.

Les classes de services généralement prises en considération sont : la voix, la vidéo, le courriel, le web et le transfert de fichier. Malheureusement, un mécanisme de priorité n'assure pas le traitement des files de basses priorités vu qu'il permet à un utilisateur de haute priorité et à mauvais comportement, d'augmenter considérablement le délai et de diminuer le bande passante restante disponible pour les autres connections de priorités plus basses. Un cas extrême de ceci est la " famine " (starvation), où l'ordonnanceur ne sert jamais les files de basses priorités puisqu'il a toujours quelque chose à servir des files qui sont plus prioritaires.

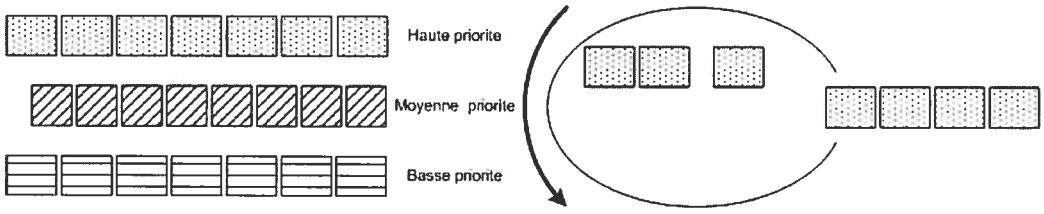



FIG. 4.1 – Ordonnancement avec priorités

**Conservation du travail (WC) versus Non conservation du travail (NWC)**
Un ordonnanceur de type WC n'est libre que s'il n'y a aucun paquet en attente d'être servi dans toutes les files d'attente. Au contraire d'un ordonnanceur de type NWC qui peut être libre même s'il y a des paquets en attente d'être servis dans certaines files : l'ordonnanceur sert une file particulière pendant une certaine période. S'il arrive que cette file soit vide avant l'écoulement de cette période alors l'ordonnanceur attendra la fin de cette période pour passer au service d'une autre file. En fait l'implantation des ordonnanceurs de type NWC a un sens du fait que le trafic devient prédictible

dans les commutateurs du coté récepteur. Du moment où le trafic devient prédictible, il est possible de réduire la taille des files d'attentes, ainsi que stabiliser la gigue d'une connexion donnée. Cependant, on se trouve dans une situation où l'on gaspille la bande passante, puisque comme on l'a mentionné plus haut, un ordonnanceur de type NWC est libre de service même s'il y a des paquets en attente de service dans d'autres files.

**Degré d'agrégation** Le troisième critère important lors de la conception d'une politique d'ordonnancement est le degré auquel les connections individuelles sont agrégées (regroupées). L'agrégation est le mécanisme permettant de regrouper un ensemble de connections en agrégats (figure 4.2). Ces agrégats partagent le même degré de qualité de service. En cas d'agrégation, l'ordonnanceur utilise un état unique pour décrire toutes les connections qui appartiennent à la même classe de service. Un état est une structure de donnée qu'on peut accéder et mettre à jour et qui permet d'avoir des informations précises sur la connexion ou l'agrégat de connections ainsi que l'historique de son asservissement. Habituellement et en absence d'agrégation, l'ordonnanceur maintient un état pour chaque connexion pour pouvoir donner aux différentes connections différentes bandes passantes et limites de délais.
L'agrégation traite, ainsi, d'un plus petit état dans l'ordonnanceur, mais réalise une différentiation de services moins fine.

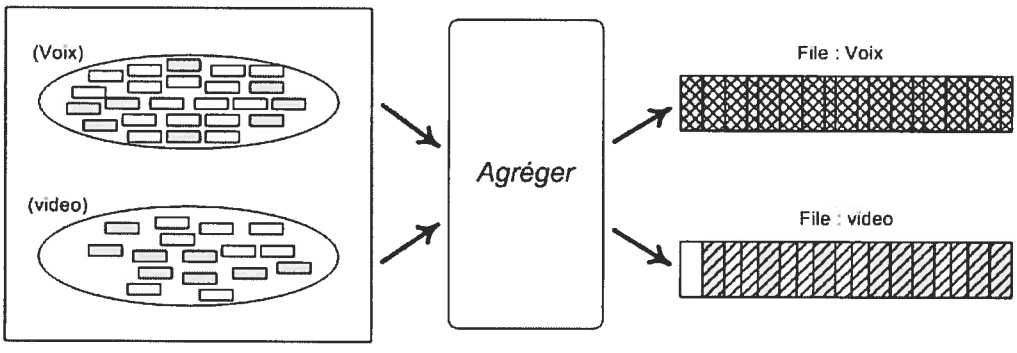



FIG. 4.2 – Formation d'agrégats

**Ordre de service à l'intérieur d'un niveau de priorité ou d'un agrégat** Ce dernier critère de liberté, lors de la conception d'une politique d'ordonnancement, représente l'ordre dans lequel l'ordonnanceur sert les paquets des connections appartenant au même niveau de priorité ou à la même classe de service. Il y a deux principaux choix :

- ✓ servir les paquets dans l'ordre de leurs temps d'arrivée.
- ✓ servir les paquets en désordre mais selon une étiquette affectée a chacun. En choisissant cette option, l'ordonnanceur se retrouve fortement dépendant de la manière dont ces étiquettes sont calculées. Cette étiquette est une information intégrée dans un ou plusieurs champs du paquet pour permettre aux nœuds suivants dans le réseau de le traiter d'une certaine manière.

### 4.1.3 Politiques classiques d'ordonnancement

Plusieurs politiques d'ordonnancement ont été proposées dans la littérature. Nous nous limiterons à présenter certaines d'entre elles. L'équité est l'une des attentes d'une politique d'ordonnancement agissant dans un contexte multiservices. Ainsi nous nous attarderons surtout sur ce critère.

#### First In First Out (FIFO)

L'une des premières politiques d'ordonnancement est le *first in first out* (premier entré premier servi). FIFO [40] est généralement prise par défaut pour les services différenciés vu sa simplicité. Généralement on ne délaissera FIFO en faveur d'une autre politique que s'il ne peut offrir les caractéristiques envisagées. Dans FIFO les paquets allant vers un lien de sortie sont placés dans une file dans l'ordre dans lequel ils ont été reçus à l'entrée. Dans la forme classique de FIFO, les paquets ne sont éliminés de la file que si elle est pleine et ne peut accepter d'autres paquets.
FIFO offre certains avantages mais aussi certains inconvénients. Il offre un bon degré d'efficacité de coût (cost-efficiency) puisqu'avec un mécanisme simple, FIFO gère effi-

cacement la capacité du buffer et du lien en sortie. Certaines politiques bien plus compliquées n'arrivent pas à faire mieux. Tant que les utilisateurs se comportent tous de façon similaire (envoient les paquets à des intervalles identiques), FIFO reste équitable. Dans ce cas, les délais et le taux de perte des paquets sont similaires pour tous les flots. Néanmoins, cela n'est pas un critère suffisant pour établir une politique équitable puisqu'un système doit offrir cette équité entre des utilisateurs ayant des profils et des comportements différents.

Le problème avec FIFO est qu'elle fournit le même traitement pour les flux à haut débit comme pour ceux à débit faible. Cela ne serait bien sûr pas un problème si les clients étaient par exemple facturés selon l'utilisation de la bande passante, et encore : les paquets qui veulent garantir un meilleur délai ne peuvent pas sauter à la tête de la queue pour être servis rapidement. De l'autre côté, une politique basée par exemple sur un service des paquets à l'aide de leur étiquette, peut très bien permettre à ces paquets de sauter, s'ils le désirent, à la tête de la file pour leur garantir des délais inférieurs. Un autre problème avec FIFO est que l'allocation de la bande passante à des connections individuelles n'est pas *max-min fair share* : les connections sont servies à peu près proportionnellement à la vitesse à laquelle ils envoient les données sur le réseau. FIFO récompense les connexions qui envoient à une grande vitesse au dépend de connections qui obéissent à des schémas de contrôle de flots ou de fenêtrage.

Même si une partie du problème d'équité peut être résolue en établissant un contrôle de trafic approprié dans les nœuds à la limite du réseau (externes), cela semblerait nécessaire que pour réaliser l'équité, il faille mettre en œuvre des politiques bien plus compliquées que FIFO. Avec un tel contrôle de trafic, le système reste vulnérable à un mauvais comportement du client ce qui fait que FIFO, même si elle est mise dans un contexte d'équité, n'est par contre pas robuste.

Ainsi, d'un point de vue services différenciés, FIFO n'est pas du tout polyvalente puisqu'elle ne peut fournir une différentiation des flots. En effet, tous les flots passant par une queue FIFO auront les mêmes caractéristiques de délai et de perte de paquets. Il est bien clair qu'une différentiation des services n'est pas possible sans l'établissement d'autres politiques d'ordonnancement plus avancées.

### Priority Queuing ou PQ

PQ peut être utilisé pour réaliser de la QdS. PQ possède un certain nombre de files d'attentes, habituellement de haut en bas. Les files d'attentes sont servies dans l'ordre strict de leur priorité : la file d'attente de haute priorité est toujours servie en premier, puis la prochaine plus basse priorité et ainsi de suite. Si une file d'attente de basse priorité est en train d'être servie, et qu'un paquet entre dans une file de plus haute priorité, alors cette dernière est servie immédiatement. Ce mécanisme est excellent quant au trafic de plus haute priorité puisqu'il lui permet de réaliser des délais des plus minimes. Cependant, l'inconvénient majeur est qu'il peut mener à la famine des files à basses priorités (4.1.2).

### Deficit Round Robin DRR

Dans RR, les paquets sont soit triés par classes pour ensuite être affectés à la file correspondante à la classe, soit affectés directement à la file correspondante au flot auquel ils appartiennent. Ensuite, un tourniquet alterne les paquets à servir parmi les files présentes(figure 4.3) [34]. Dans sa forme la plus simple, avec deux classes de paquets, on

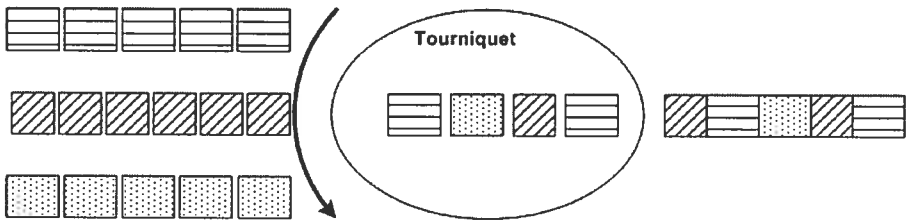



FIG. 4.3 – Round Robin

obtiendrait le trafic suivant : un paquet de la classe 1 est transmis, suivi d'un autre de la classe 2, suivi d'un de la classe 1, suivi d'un autre de la classe 2 et ainsi de suite. Le grand inconvénient de ce modèle est qu'il n'est pas équitable : les flux qui transmettent des paquets de grandes tailles sont favorisés.

Deficit Round Robin est une amélioration de Round Robin pour atteindre un meilleur niveau d'équité. Considérons un lien en sortie dans un nœud, auquel, l'accès est contrôlé par un Ordonnanceur DRR [34] [1]. Supposons qu'on a un total de $n$ flots, associé chacun à une file en entrée, et qu'on possède une seule et même file en sortie pour toutes ces files. Un flot est dit actif durant un intervalle de temps, s'il possède toujours des paquets en attente d'être servi dans la file lui correspondant. L'ordonnanceur DRR maintient une liste chaînée des flots actifs. Au début d'une période dite active, le flot est ajouté à cette liste. Un round consiste en une itération de type *round robin* durant laquelle l'ordonnanceur DRR sert tous les flots présents dans la liste active au début du round.(Les flots qui étaient présents dans la liste active au début du round).

L'ordonnanceur DRR affecte une proportion appelée *Quantum* pour chaque flot. Cette proportion est définie comme étant la proportion de service que doit recevoir un flot durant un tour. Cependant, on peut se trouver dans le cas où un flot actif ne transmettrait pas un paquet, car s'il le transmettait, il excéderait la proportion (ou Quantum) qui lui était allouée. Dans ce cas, l'ordonnanceur va se souvenir du *reste* de la proportion qui n'a pas été utilisé (ou deficit count), et il va le stocker dans un compteur pour l'additionner au *quantum* lors de la prochaine itération. Ainsi un flot qui n'a pas reçu sa portion équitable de bande passante durant un certain tour, aura l'opportunité de voir sa portion augmenter proportionnellement lors du prochain tour.

La grande attraction de DRR est qu'il est facile à implanter.

### Generalized Processor Sharing (GPS)

Les attentes par rapport à une politique d'ordonnancement pour connections à services garantis, ainsi que les objectifs lors de leur ordonnancement, sont tout à fait différents de ceux des connections de type best-effort. Pour les connections de type best-effort, on voudrait bien que la politique d'ordonnancement fournisse une allocation qui soit conforme au critère de *max-min fair share*. Ce critère peut être achevé avec un ordonnancement idéal de type work-conserving appelé Generalized Processor Sharing, qui comme le nom l'indique, est une généralisation de Processor Sharing (PS).

Dépendamment de ce critère, GPS traite les paquets comme s'ils appartenaient à des files d'attente différentes, visitant tour à tour chaque file non vide et servant une portion infinitésimale de chaque file (quelques bits). Contrairement à PS, les connections peuvent se voir accorder des poids et du même coup recevoir un service proportionnellement à ce poids, et ce, tant qu'ils ont des données à envoyer. S'ils n'en ont pas, l'ordonnanceur saute à la file suivante. Notons que du point de vue de l'équité et precisement du critere *max-min fair share*, GPS offre de la protection. GPS est idéal puisqu'il achève exactement une allocation max-min équitable [1].

Voici une définition plus précise de GPS [3] [4] [35]. Supposons qu'à chaque session $i$ est associé un poids $\Phi(i)$. Les sessions sont au nombre de $N$. Par ailleurs, on note $r^m$ le débit du routeur $m$, $S_i^m$ la fonction de service de la session $i$ sur ce routeur : $S_i^m(\tau, t)$ désigne la quantité d'information de la session $i$ traitée par le routeur $m$ sur l'intervalle de temps $[\tau, t]$. Un routeur GPS est un routeur pour lequel :

$$\frac{S_i^m(\tau, t)}{S_j^m(\tau, t)} \geq \frac{\Phi_i^m}{\Phi_j^m}$$

pour toute session $i$ active (i.e. ayant des paquets à émettre) sur l'intervalle $[\tau, t]$. Une telle session est assurée de disposer en sortie du routeur $m$ d'un débit supérieur ou égal à $g_i^m$ avec :

$$g_i^m = \frac{\Phi_i^m}{\sum_{j=1}^{N} \Phi_j^m} r^m.$$

Intuitivement, l'idée de GPS est donc d'accorder un débit de sortie à chaque session active et qui soit proportionnel à la part de son poids parmi l'ensemble des sessions actives (et au débit du routeur). L'inconvénient réside dans le fait que cette méthode ne puisse être implémentée pour le transfert des paquets où le traitement de ceux-ci est supposé non-préemptible. En effet sous GPS, l'information (le paquet) est supposée pouvoir être infiniment divisible. À défaut de pouvoir être appliquée directement, cette politique sert de référence à diverses politiques plus récemment développées auxquelles appartient par exemple WFQ.

### PGPS ou WFQ

PGPS est aussi appelée Attente équitable proportionnelle, mais le nom le plus commun dans la littérature est WFQ [1]. Comme l'indique son nom (*fair*), WFQ est une politique d'ordonnancement équitable selon la nation déquité définie dans la section 4.1.1. Les paquets sont servis de façon circulaire comme le montre la figure 4.4.

L'idée de base de WFQ également nommé PGPS est d'imiter le comportement de GPS

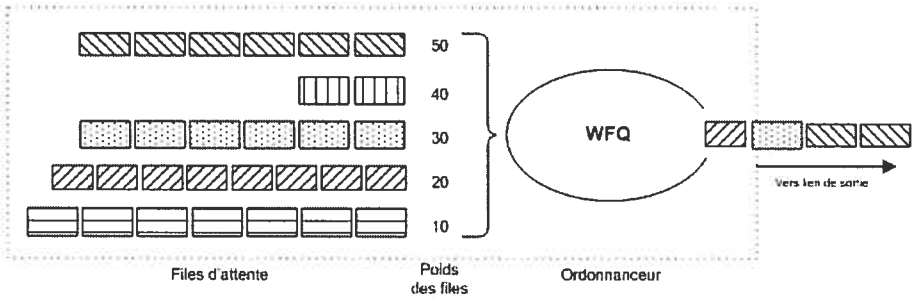



FIG. 4.4 – Weighted Fair Queuing

en traitant les paquets dans leur ordre de fin de service sous GPS. L'émulation de GPS repose sur la notion de temps virtuel. Celui-ci est employé pour le calcul d'une estampille (étiquette) associée à chacun des paquets arrivant sur le routeur et dont la valeur peut intuitivement s'interpréter comme un *indicateur sur l'ordre* dans lequel les paquets sont traités sous GPS.

La mise à jour du temps virtuel $R$ s'effectue à chaque date d'événement $t_j$, $j$ désignant le numéro de l'événement (un événement correspond à l'arrivée ou à la fin de traitement d'un paquet). Elle est donnée par les formules suivantes :

$$R(0) = 0$$
$$R(t_j) = R(t_{j-1}) + \frac{\tau}{\sum\limits_{i \in B_j} \Phi_i}, \qquad \tau \leq t_j - t_{j-1}, j = 2, 3, ...$$

où $B_j$ désigne l'ensemble des sessions actives sur $[t_{j-1}, t_j]$. Le calcul de la valeur de l'estampille $F_i^k$ du $k^{ème}$ paquet de la session $i$ sur un routeur $m$ est donné par :

$$F_i^0 = 0$$
$$F_i^k = \frac{L_i^k}{\Phi_i} + \max\{F_i^{k-1}, R(a_i^k)\},$$

où $a_i^k$ correspond au temps réel d'arrivée du $k^{ème}$ paquet de la session $i$, pour toute session $i$. Les travaux effectués par Parekh [3], et conjointement par Parekh et Gallager [4], ont en outre permis d'établir des bornes en termes de délais sur les paquets, sous certaines hypothèses (section 5.2.1).

### Horloge virtuelle (Virtual Clock)

La politique d'ordonnancement de VC [27] a été proposée presque au même moment que le *fair queuing* [29]. Le souci de VC n'était pas d'atteindre un certain niveau d'équité entre plusieurs connections partageant les mêmes ressources, mais plutôt d'émuler la technique du TDM.

En effet, dans un ordonnanceur appliquant cette technique, on alloue à chaque paquet un temps de transmission virtuel représentant le temps à partir duquel le paquet sera transmis. L'application de ce concept dans les réseaux à commutation de paquet implique [35] :

- attribuer différentes quantités en bande passante pour les flots,
- superviser la bande passante utilisée par les flots,
- isoler les flots les uns des autres,
- garder la flexibilité d'un environnement à commutation de paquets.

A la création, on assigne au flot un *taux moyen* $AR_i$ ainsi qu'une *horloge virtuelle* $VC_i$. $VC_i$ est initialisée à la valeur du temps réel de la création du paquet. Chaque fois qu'on reçoit un paquet du flot $i$, on ajoute à $VC_i$ une valeur $d$ qui n'est autre que $\frac{P}{AR_i}$, où $P$ est la taille du paquet. $VC_i$ est ensuite utilisée comme estampille pour le paquet : les paquets seront envoyés selon les valeurs de ces étiquettes. Ce calcul ressemble aux calculs pour obtenir le temps de fin, $F_i$ du WFQ. Seulement ici, l'indicateur sur l'ordre a été remplacé par le temps réel. Le VC est donc plus simple à implémenter que l'est WFQ [22] puisque le calcul du temps virtuel $R$ (voir section précédente) n'est plus

utile.

Sauf que les politiques VC et WFQ ne sont pas toujours équivalentes. Elles le sont, quand toutes les connections sont en attente d'exécution (toutes les files sont pleines). Dans ce cas, elles fournissent la même QdS ainsi que les mêmes bornes de bout-en-bout de pire cas (Worst-case-end-to-end-delay bounds) [31]. Il est donc possible de substituer WFQ par un VC dans le cas de connections à service guaranti (Hard QoS). Dans les cas du service best-effort ou du service différencié, VC n'est plus équitable tandis que WFQ le reste toujours [32]. En effet, une connexion peut se voir attribuer beaucoup plus de bande passante qu'une autre.

### 4.1.4 Architectures d'ordonnancement

#### Class-Based Queuing CBQ

Class-based queuing [36] [37] [43] [35] n'est pas tout à fait une technique d'ordonnancement, mais plutôt un ensemble de composants utilisés pour bâtir une politique générale de gestion du trafic dans les routeurs offrant une certaine qualité de service. Ces composantes sont décrites ci-dessous.

- ✓ Classificateur : à l'entrée, il dirige les paquets vers leurs files correspondantes.
- ✓ Estimateur : calcule une estimation de la bande passante allouée à une classe donnée.
- ✓ Sélecteur : cherche la prochaine classe à servir pour envoyer le paquet suivant : c'est l'équivalent de l'ordonnanceur.
- ✓ Retardateur : pour une classe qui dépasse sa portion de la bande passante allouée et qui contribue donc à la congestion, on calcule le temps où cette classe sera autorisée à envoyer des paquets de nouveaux.

Le principal objectif de CBQ est de prévenir le déni complet de ressources pour une classe particulière. Ainsi, l'objectif de CBQ est de résoudre le problème de famine (starvation) rencontré par la politique PQ. On pourrait par exemple paramétrer le sélecteur de sorte

qu'il supporte la politique CBWFQ (section 4.1.5). La principale supposition de CBQ est que le trafic est organisé dans des agrégats relativement de larges tailles en se basant sur leur classe de service (groupe de QoS) pouvant, par exemple, être identifiée par la précédence IP du champ ToS d'un paquet IP [40].

### 4.1.5 Autres variantes de la politique WFQ

#### Class Based Weighted Fair Queuing CBWFQ [CB03]

CBWFQ est une extension de WFQ et permet de prendre en compte, non plus des flots individuels comme le fait WFQ, mais des classes de QdS définies par l'utilisateur, et par conséquent permet de fournir un meilleur contrôle sur le trafic. Les paquets sont affectés à différentes files d'attente en se basant sur leur appartenance à une classe de QdS donnée. Entre autres, les paquets des différents flots se conformant aux caractéristiques d'une classe donnée vont constituer le trafic pour cette classe. A son arrivée, le paquet est diagnostiqué pour savoir sa classe de QdS, cela est possible en lisant, par exemple, la valeur de la *précédence IP* dans son champs ToS. Une fois que celle-ci est connue, le paquet sera dirigé directement vers la file d'attente réservée à cette classe. En plus d'identifier une classe de service par une valeur de la *précédence IP*, on va aussi la caractériser par un certain poids et une bande passante. Cette valeur de bande passante est une valeur à garantir à la classe en cas de congestion. En effet, en cas de congestion chaque groupe se voit alloué une proportion fixe de la bande passante proportionnelle à son poids. Par exemple, si le poids est égal à 50, les paquets appartenant à cette classe se voient alloué au moins 50% de la bande passante. Il est possible aussi de définir une limite pour le nombre de paquets à accepter par file d'attente. Si jamais il arrive qu'une file dépasse cette limite, alors les paquets superflux seront supprimés.

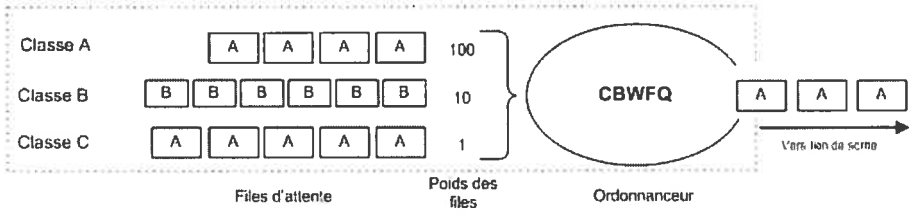



FIG. 4.5 – Class Based Weighted Fair Queuing

### 4.1.6 Politiques adaptatives

Durant ces dernières années, plusieurs politiques adaptatives, dont plusieurs spécifiques à WFQ, ont été proposées [22] [21] [20] [19] [18] [23] [24] [25].

Dans [22], on propose une manière d'ajouter un aspect dynamique à WFQ en rendant dynamique le processus de distribution de la bande passante en excès. Selon le critère *max min fair share* (section 4.1.1), lorsqu'un flot (classe) n'utilise pas la totalité de la bande passante qui lui est allouée, l'excès va être redistribué sur le reste des flots (classes) dans l'ordre croissant de la demande. Dans [22], les auteurs proposent trois redistributions différentes qui ont pour but l'amélioration des performances sur le délai.

- ➢ Favoriser les flots qui ont les plus grands risques de dépassement des délais permis.
- ➢ Favoriser les flots qui ont de plus grands ratios de violation du délai (*deadline-violation ratio*).
- ➢ Favoriser les files qui sont plus chargées que les autres.

Dans [21], on propose de faire varier les poids d'un CBWFQ. Le poids étant un paramètre qui, avec le taux d'arrivée des paquets, affectent directement la taille des files d'attentes. Le poids des files est modifié selon la formule suivante :

$$\mu_i(t) = \frac{w_i q_i(t)}{\sum_{i=1}^{n} w_i q_i(t)} \times \mu \qquad pour \quad i = 1, ..., n.$$

où $w_i$ représente le poids de la classe $i$, $\mu_i(t)$ la proportion de bande passante disponible pour $i$, $\mu$ la bande passante totale en sortie et $q_i(t)$ la taille de la file d'attente de la classe $i$ à un instant $t$. Cependant on ne mentionne pas la fréquence avec laquelle on

effectue ce changement de poids, ni à quel moment et sous quelles conditions le faire.

Dans [20], on estime servir la queue dont le débordement sera le plus proche parmi toutes les autres files pleines. Ce temps de débordement est calculé comme suit :

$$T_i(t) = \frac{Q_i^{\max} - Q_i(t)}{\phi_i.C} \tag{4.1}$$

où $Q_i^{\max}$ est la taille totale de la file, $Q_i(t)$ est l'occupation courante et $\phi_i.C$ étant la fraction de bande passante allouée à la classe $i$.

Dans le même concept du débordement le plus proche de file, [19] propose de remplacer $\phi_i.C$ de la formule (4.1) par une une valeur prédite du taux d'arrivée (predicted arrival rate). La formule présentée dans [19] est alors :

$$T_i(t) = \frac{Q_i^{\max} - Q_i(t)}{\hat{W}(t)/T_{est}}.$$

où $\hat{W}(t)$ est l'estimation du nombre de bits à recevoir durant la période $T_{est}$.

[18] propose quant à lui de faire varier les poids selon deux paramètres : le taux d'arrivée des paquets ainsi que l'occupation de la file d'attente. On précise que les périodes d'ajustement des poids, les périodes pour lesquelles on mesure le taux d'arrivée des paquets ainsi que la durrée de la période durant laquelle on calcule le nombre de bits accumulés dans les files, sont des paramètres configurables des simulations.

Le poids est modifié selon la formule suivante :

$$w_i^k = \frac{1}{\delta_i}(b_i^k + \frac{1}{2}\lambda_i^k U).$$

où $b_i^k$ représente le nombre de bits du trafic de la classe $i$ enfilés durant la période $[k-U, k]$, $\lambda_i^k$ représente, quant à lui, le taux d'arrivée d'une classe $i$ au temps $k$, et $\delta_i$, un ratio de comparaison de la classe $i$ avec les autres classes.

Dans [23], on présente une méthode pour *calibrer* les poids utilisant une estimation de la taille moyenne de la file de la classe de service ayant le plus grand poids. Cette dernière est appellée classe *premium*. Deux seuils sont aussi introduits : un seuil minimum, représentant le délai d'attente désiré dans la file, et un seuil maximum représentant

le délai d'attente acceptable. Le but est de garder la taille moyenne de la file en dessous du seuil maximum, réalisant ainsi un délai d'attente acceptable. Pour accomplir cela, le poids attribué à la classe *premium* est augmenté proportionnellement dès qu'on dépasse le seuil minimum avant même d'atteindre le seuil maximum. Ce calibrage n'est pas effectué infiniment sinon on finira par ressembler à un PQ. Ainsi on limite le poids de cette classe à une valeur maximale, protégeant ainsi les classes inférieures du risque de famine causé par PQ.

Dans [25], la décision pour la variation des poids est effectuée sur la base d'estimations réalisées sur les charges futures de trafic relative aux différentes classes de service considérées.

Comme on va l'expliquer dans le chapitre 6, nous proposons une politique adaptative qui diffère de celles présentées ci-dessus. En effet, notre politique prend en compte des mesures, en temps réel, des délais de bout en bout subis par les paquets des classes temps réel. Ces mesures seront utiles pour décider du moment opportun pour procéder à l'ajustement des poids. Cependant l'ajustement en question se base, comme dans [18], sur des paramètres comme celui du taux d'occupation dans les tampons.

## 4.2 Agrégation

### 4.2.1 Aperçus et définitions

Comme le définit Cobb dans [53] [54], un flot simple est une séquence de paquets générés par la même source et ayant la même destination. Un flot agrégé ou agrégat est un ensemble de flots simples ou d'agrégats.

La composante qui permet le regroupement de ces flots constituants en un ou plusieurs agrégats de flots est appelée *agrégateur* comme le montre la figure (4.6). Les ordonnanceurs, ou classificateurs par lesquels l'agrégat va passer ne vont pas voir les flots constituants mais plutôt l'agrégat comme tel. Les actions qu'ils vont entreprendre vont

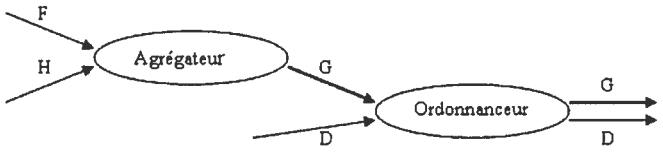



FIG. 4.6 – Regroupement des flots par l'agrégateur

se faire en tenant compte de l'agrégat et non des flots constituants [51] [52], et ce, pour les raisons suivantes.

- ✓ Lors de congestion, chaque source répond à sa manière pour éviter de futures périodes de congestion.
- ✓ Chaque flot de l'agrégat a une destination qui peut être différente des autres. Ce qui implique que chaque source va rencontrer des délais et des périodes de congestion différentes.

Cependant, pour traiter l'agrégation dans les réseaux, et particulièrement pour les réseaux multiservices, il est d'une grande utilité qu'on puisse prendre en compte non seulement le débit de l'agrégat mais aussi celui réalisé par chacun des flots constituants. Ce qui n'est pas si evident !

### 4.2.2 But et apports de l'agrégation

Les avantages les plus évidents de l'agrégation sont présentés ci-dessous [57] [51] [52] [53].

- ➢ La simplicité lors de la gestion du trafic puisqu'on est en présence d'un nombre restreint de flots. En effet, une seule file d'attente est à gérer par agrégat au lieu d'une par flot constituant.
- ➢ Amélioration de l'efficacité des ordonnanceurs vu qu'ils ne considèrent plus les files de tous les flots mais seulement celles des agrégats.

Aussi, l'agrégation réduit considérablement l'état dans l'ordonnanceur puisqu'on maintient dorénavant un état pour chaque classe au lieu d'un pour chaque connexion : L'état est la ressource la plus critique lors de l'implémentation d'une politique d'ordonnancement. L'implantation matérielle de l'état, par exemple, est en effet une tâche qui n'est pas si facile à réaliser surtout si on maintient un état pour chaque connexion. Aussi, plus l'état est grand dans l'ordonnanceur, plus il consomme de ressources (mémoire, espace disque...) et plus l'accès à l'information qu'il contient est fastidieuse.

Le dernier avantage qu'on voit à l'agrégation est qu'elle permet aux connections d'obtenir une plus petite gigue quand elles envoient une *rafale* de paquets ou *burst* [1] [8]. Supposons qu'on est en présence de dix connections partageant un ordonnanceur de type non-work-conserving qui va toutes les réguler au même taux de service. Supposons que neuf d'entre elles envoient les paquets de manière espacée, alors que la dixième les envoie sous forme d'une rafale de cinq paquets à la fois. Si les connections ne sont pas agrégées, cette dernière connexion aura une large gigue puisque le cinquième paquet de chaque rafale aura un délai beaucoup plus important que celui du premier paquet. Dans ce cas, les autres connections ne seront pas touchées par cette *burstiness.*

De l'autre coté, si on avait des connections agrégées, la rafale de la dixième connexion sera servie consécutivement aux autres connections de la même classe, ce qui fait que les gigues des autres connections se trouvent augmentées. Néanmoins, la gigue de la dixième connexion sera diminuée puisque toutes les autres connexions la partagent avec elle.

Le problème principal de l'agrégation est que les connections appartenants à la même classe d'agrégation ne sont pas protégées les unes par rapport aux autres [8]. Puisque l'ordonnanceur ne peut pas distinguer entre les connections d'une même classe, le mauvais comportement d'une connexion dans cette classe va affecter toutes les autres. Ainsi, le degré d'agrégation est inversement proportionnel à celui de la protection.

# Chapitre 5

# Notions de délais

## 5.1 Composantes du délai de bout en bout

Le délai de bout en bout expérimenté par un paquet sur les liens câblés d'un réseau à commutation de paquets est constitué de deux composantes distinctes [65] : une partie fixe due au délai de propagation, au délai de transmission sur les liens et au délai de traitement au niveau des nœuds (routeurs), et une partie variable due au délai d'attente dans les files d'attentes des différents nœuds traversés. Cette variabilité de la valeur du délai subi par un paquet est appelée *gigue* comme énoncée dans le premier chapitre. Le délai d'attente représente la composante critique du délai de bout en bout qu'il faut contrôler pour fournir des garanties de QdS aux applications surtout celles de types temps réel.

Notons $D_f$ le délai de bout en bout subi par un paquet du flot $f$ sur le réseau. Soit $h_f$ le nombre de nœuds traversés par le flot $f$ y compris la source du flot.

Soit $D_{f,m}^{prg}$ et $D_{f,m}^{trx}$ respectivement le temps de propagation et le temps de transmission sur le lien entre les nœuds consécutifs $m$ et $m+1$. Alors le délai de propagation total $D_f^{prg}$ tout au long du chemin supportant le flot $f$ dépend de la longueur des liens et est

calculé comme suit :

$$D_f^{prg} = \sum_{m=1}^{h_f} D_{f,m}^{prg}.$$

Le délai de transmission total $D_f^{trx}$ le long du chemin dépend de la taille des paquets du flot $f$ et de la bande passante allouée au flot et vaut :

$$D_f^{trx} = \sum_{m=1}^{h_f} D_{f,m}^{trx}.$$

Soit $D_f^{avg}$ le temps de traitement moyen subi par les paquets appartenant au flot $f$ dans chacun des nœuds $m$ du chemin emprunté. Le temps de traitement total $D_f^{tot}$ est :

$$D_f^{tot} = h_f \times D_f^{avg}.$$

Soit $D_f^q$ le temps d'attente d'un paquet de $f$ dans toutes les files d'attentes sur le chemin emprunté. On a alors la composition suivante du délai de bout en bout d'un réseau cablé :

$$D_f = D_f^{trx} + D_f^{prg} + D_f^{tot} + D_f^q.$$

## 5.2 Bornes de délais déterministes avec WFQ

Plusieurs travaux ont été réalisés dans le but de calculer des bornes de délais dans le cas de flots simples [3] [4] [14] [13]. Cependant peu de travaux ont été effectués pour le cas de flots agrégés [10] [8] [12]. Dans le cadre d'un réseau à services différenciés (ou même à services garantis), ces travaux ont pour but de calculer le délai avant même l'établissement de la connexion entre une même paire origine destination. La notion de borne sur le délai s'avère un outil d'aide à la decision important si celles-ci sont assez réalistes et précises. En effet, on peut, par exemple, favoriser les paquets d'une application sensible au délai, dont on sait que les délais ne seraient pas respectés. On peut aussi tout simplement refuser cette connexion vu que la qualité demandée ne sera pas assurée.

### 5.2.1 Flots simples

La classe pour les services garantis est conçue pour fournir aux applications une garantie stricte sur le délai subi par chaque paquet d'un flot depuis son émission à la source jusqu'à son arrivée à destination. Ce délai est composé de deux parties, le temps de propagation lié au support physique et incompressible ($D_f^{trx} + D_f^{prg} + D_f^{tot}$), et le temps passé dans les files d'attente, que l'on peut contrôler par divers mécanismes. Ici, nous nous intéressons uniquement au temps passé dans les files d'attente, le temps de propagation pouvant être facilement calculé et ajouté pour obtenir un délai total.

Puisque l'algorithme d'attente équitable proportionnelle isole les flots les uns des autres dans les routeurs, on peut s'attendre à ce qu'il permette d'offrir des garanties de performance aux utilisateurs du réseau. C'est effectivement le cas sous condition que le trafic généré par les utilisateurs ne soit pas trop chaotique (ce point est discuté plus en détail ci dessous). L'algorithme alloue sur chaque nœud une fraction de la bande passante disponible $r$ à chacun des flots

$$g_i = \frac{w_i}{\sum_j w_j} r \tag{5.1}$$

où $w_i$ est le poids associé au flot $i$. Cette fraction peut être arbitrairement petite si le nombre de connections actives est grand. Cependant, chaque flot est assuré d'une fraction minimum de capacité.

Les travaux de Parekh et Gallager [3] [4] ont montré que l'algorithme d'attente équitable proportionnelle permet d'offrir, en plus de cette garantie de bande passante, une garantie de délai si le trafic généré par les utilisateurs n'est pas trop chaotique. Il faut alors supposer que le trafic de chaque utilisateur se conforme à un mécanisme de contrôle d'accès de type *seau à jetons (token bucket).*

Le mécanisme est donc complètement caractérisé par ses paramètres $\sigma$ et $\rho$. Il limite le débit moyen de la source (qui est égal à $\rho$) et le nombre maximal de paquets qui peuvent être émis en rafale (qui est égal à $\sigma$ paquets).

Considérons maintenant un flot $i$ qui est passé à travers un mécanisme de contrôle de type *seau percé (leacky bucket)*de paramètres $\sigma_i$ et $\rho_i$ avant de rentrer dans le réseau.

On sait alors qu'à chaque nœud $m$ sur le chemin du flot, le débit de service minimum garanti est :

$$g_i^m = \frac{w_i^m}{\sum_j w_j^m} r^m$$

où $r^m$ est la bande passante disponible au nœud $m$ et $w_i^m$ le poids du flot $i$ au nœud $m$. Ainsi le taux de service minimum sur le chemin est :

$$g_i = \min_m g_i^m$$

Dans le cadre de l'étude, nous énonçons les résultats établis par Parekh et Gallager dans le cas des systèmes stables, c'est-à-dire pour les systèmes où

$$\sum_j \rho_j \leq r^m$$

ce qui peut être assuré par un contrôle d'admission.

Si le flot $i$ est localement stable, c'est-à-dire si $g_i \geq \rho_i$, alors les travaux de Parekh et Gallager [4] montrent que :

- la quantité de trafic en attente de service maximale à chaque nœud $m$ sur le chemin ne dépasse pas une valeur maximum $Q_i^{\max}$

$$\begin{aligned} Q_i^{\max} &= \sigma_i && GPS, \\ Q_i^{\max} &= \sigma_i + L_i^{\max} && PGPS. \end{aligned}$$

- le délai maximal des paquets de ce flot entre une source et une destination à travers un réseau de topologie arbitraire ne dépasse pas une valeur maximum $D_i^{\max}$ qui est égale à :

$$\begin{aligned} D_i^{\max} &= \frac{\sigma_i}{g_i} && GPS, \\ D_i^{\max} &= \frac{\sigma_i}{g_i} + \frac{2(h_i - 1)L_i^{\max}}{g_i} + \sum_{m=1}^{h_i} \frac{L^{\max}}{r^m} && PGPS. \end{aligned}$$

où $r^m$ est la bande passante disponible au nœud $m$, $h_i$ est le nombre de nœuds entre la source et la destination, $L_i^{\max}$ la taille maximale des paquets du flot $i$ et $L_{\max}$ la taille maximale d'un paquet dans le réseau.

Soit la condition RPPS (*Rate Proportional Processor Sharing*) $w_i = \rho_i$ à chaque nœud $m$, alors la session $i$ est localement stable. De plus, de cette manière, les coefficients $w_i$ affectés aux sessions $i$ sont alors en adéquation avec les qualités de service demandées par les sessions puisque $\rho_i$ représente le débit régulant l'émission de la source. Enfin, les travaux de Parekh et Gallager [4] montrent que sous la condition RPPS :

$$
\begin{aligned}
Q_i^{\max} &= \sigma_i && GPS,\\
Q_i^{\max} &= \sigma_i + L_i^{\max} && PGPS,\\
D_i^{\max} &= \frac{\sigma_i}{\rho_i} && GPS,\\
D_i^{\max} &= \frac{\sigma_i}{\rho_i} + \frac{2(h_i-1)L_i^{\max}}{\rho_i} + \sum_{m=1}^{h_i} \frac{L^{\max}}{r^m} && PGPS.
\end{aligned}
$$

Pour cette dernière formule en version PGPS le premier terme est souvent le terme prépondérant, en particulier si l'on considère un réseau haut débit car $r^m$ devient très grand devant $L_{\max}$ et l'on obtient alors la formule suivante :

$$
D_i^{\max} = \frac{\sigma_i}{\rho_i} + \frac{2(h_i-1)L_i^{\max}}{\rho_i} \qquad PGPS.
$$

On s'aperçoit que l'on retombe sur la formule en version GPS si $L_i^{\max}$ est très petit, soit si l'on a des paquets de petites tailles. Cette borne de délai est valide pour une topologie arbitraire du réseau, et même si le trafic généré par les autres flots n'est pas régulier du tout. La borne $D_i^{\max}$ n'est donc pas très stricte. Par exemple, des simulations faites par Kurose [28] d'un réseau avec 4 routeurs, des liens T1 (1.5 Mbps), et 48 connections à 32 Kbps, montrent qu'avec une discipline FIFO, plus de 99.9 % des paquets atteignent leurs destinations en moins de 15 ms. Mais la borne $D_i^{\max}$ obtenue par Parekh et Gallager [4] est de 65ms. Cette borne est une borne garantie, au sens que aucun des paquets ne mettra plus de 65ms pour atteindre sa destination si les routeurs utilisent WFQ, mais elle n'est pas stricte au sens où la plupart des paquets subissent des délais bien inférieurs à ce qu'elle indique. D'autre part, l'implémentation de l'algorithme WFQ est moins simple que celle de FIFO. Cependant, le résultat de Parekh [3] montre que l'algorithme WFQ permet d'offrir des garanties de bande passante (à savoir la fraction $w_i$ de la bande passante totale au flot $i$) et de délai (à savoir la valeur $D_i^{\max}$ ci-dessus).

### 5.2.2 Flots agrégés

Comme on l'a déjà mentionné peu de travaux ont été effectués pour le calcul de bornes déterministes sur le délai de bout en bout pour le cas de flots agrégés. Une particularité importante de l'agrégation est que les flots d'un même agrégat ne sont pas protégés les uns par rapport aux autres. Ceci rend le calcul des bornes difficile à réaliser.

#### Travaux de Charny et Le Boudec [6] [7] [8] [9] [10]

Dans les récents travaux de Le Boudec et Charny [7] [8] [6], certaines bornes de délai ont été démontrées dans le cas d'un réseau avec ordonnancement agrégé des flux. On s'interesse toujours au temps passé dans les files d'attente. Ces derniers travaux sont venus améliorer ceux déjà publiés par dans [9]. Ces bornes sont à prendre en compte sous certaines suppositions et conditions dans un modèle qui peut servir pour le cas d'une classe de plus haute priorité d'un PQ ou tout simplement pour une classe d'un CBWFQ. Le cas considère dans [8] est celui de la classe de service EF [16]. EF est une classe de service à haute priorité dans le contexte des services différenciés DiffServ.
La borne présentée dans [8] pour le cas de réseaux dont la topologie est arbitraire est présentée ci-bas.
Aux limites du réseau, les flux sont supposés être individuellement conformes à un seau percé de paramètres $\rho$ et $\sigma$. Par limite du réseau on veut désigner le nœud le plus proche de la source d'un flux donné. Les nœuds suivants ne verront pas le flux comme tel, mais verront un agrégat de flux de la classe de service dont fait partie ce flux. Cet agrégat à haute priorité est traité dans chaque nœud suivant une politique d'ordonnancement particulière. Soit $h$ le nombre maximum de nœuds traversés par un flot et $\beta$ la borne sur le service (*service deficit bound*). On dit qu'un nœud ou serveur servant une file à un taux $g$ (voir formule 5.1) fournit une borne sur le service $\beta$, si, dans n'importe quel intervalle de temps $(t_1, t_2)$ dans lequel la file est continuellement pleine, il est garantit qu'au moins $g(t_1, t_2) - \beta$ bits seront servis par cette file. La borne présentée dans [8]

n'est pas valide pour n'importe quel facteur d'utilisation $\alpha$ de la bande passante d'un lien par l'agrégat de la classe à haute priorité. Elle suppose que celui-ci est limité comme suit :

$$\alpha \leq \frac{C_{in}}{(C_{in} - g)(h-1) + g}. \tag{5.2}$$

Le délai est ainsi borné par

$$D = \frac{h}{1 - u\alpha(h-1)}\left(\Delta + \frac{u\alpha\sigma_{\max}}{\rho_{\min}}\right) \tag{5.3}$$

où $u = \frac{C_{in}-g}{C_{in}-\alpha g}$, $\Delta = \max_j \frac{\beta_j}{g_j}$ ($j$ représente un nœud en particulier), $C_{in}$ est la capacité totale du lien en entrée d'un nœud, $\rho_{min}$ étant le débit minimum de n'importe quel flot de la classe prioritaire, $\sigma_{\max}$ la taille maximale d'une rafale d'un des flots de l'agrégat en question.

On remarque, en plus, que pour un $C_{in}$ très supérieur à $g$ ($C_{in} >> g$), le terme $u$ devient presque égal à 1. Ainsi 5.2 peut être écrite comme suit :

$$\alpha \leq \frac{1}{(h-1)},$$

et la borne sur le délai $D$ aussi sous la forme :

$$D = \frac{h}{1 - \alpha(h-1)}\left(\Delta + \frac{\alpha\sigma_{\max}}{\rho_{\max}}\right). \tag{5.4}$$

Cependant dans [8], on confirme que cette borne explose quand $\alpha$ tend vers $\frac{1}{(h-1)}$ [6]. En fait, les bornes de délais explosent toujours pour le cas des flots agrégés [6]. Les résultats de [8] impliquent que si $\alpha > \frac{1}{(h-1)}$, la borne sur le délai n'est plus fonction de $h$ et $\alpha$. La borne sur le délai, si elle existe, devrait probablement dépendre de la taille du réseau. Cela laisse les portes grandes ouvertes à la recherche pour essayer de trouver des bornes plus précises et fiables dans le cas des flots agrégés.

### Travaux de Jiang [12] [11]

Les travaux les plus récents qui traitent de bornes sur le délai pour le cas des flots agrégés sont ceux de Jiang et Yao [12] [11]. Ces bornes sont présentées dans le cadre

d'un réseau de topologie arbitraire dont les nœuds implémentent un ordonnancement à service guaranti (GRS) [15] avec aggrégation FIFO. Dans son article [11], Jiang considère aussi la classe de traffic EF [16], à qui, le serveur GR offre une proportion $R_{\mathrm{h}}$ garantie et une latence $E_{\mathrm{h}}$. Dans chaque noeud du réseau, les paquets appartenant à la classe EF sont agrégés et enfilés dans une même file d'une manière FIFO. Chaque flot de la classe EF est conforme à un seau percé de paramètres $\sigma$ et $\rho$. On assume que la proportion du trafic de type EF ne dépasse pas une certaine proportion $\alpha$ ($\alpha \leq 1$) de la proportion $R_{\mathrm{h}}$ offerte pour le traffic EF. On impose pour tout lien $h$ du réseau que :

$$\sum_{f \in Fh} \rho_f \leq \alpha R_{\mathrm{h}}.$$

On suppose aussi une limite $\beta$ ($\beta \geq 0$) pour la sporadicité (burstiness) de tous les flux traversant le lien $h$ :

$$\sum_{f \in Fh} \sigma_f \leq \beta . R_{\mathrm{h}}.$$

Le serveur GR peut être utilisé pour modéliser plusieurs algorithmes d'ordonnencement par flux comme par exemple : Virtual Clock, WFQ, Self-Clocked Fair Queuing (SCFQ). Le GR est défini sur la base de la valeur du *taux d'horloge garanti* (GRC) d'un paquet. Le GRC d'un paquet $p^k$ est défini comme suit :

$$GRC(p^0 = 0), \tag{5.5}$$

$$GRC(p^k) = \max\{a^k, GRC(p^{k-1})\} + \frac{l^k}{R'}. \tag{5.6}$$

Pour un flux donné, on dit qu'un algorithme est de classe GR avec proportion $R$ et latence $E$, si n'importe quel paquet sera transmis durant $GRC(p^k) + E$, où E étant une constante prédéfinie pour chaque variante de serveurs GR. Pour le cas de WFQ, E est égale à $\frac{L^{max}}{C}$, où $L^{max}$ est la taille maximale d'un paquet et $C$ le débit du lien en sortie [11] [15]. Cependant, malgré que l'ordonnancement à services garantis est défini pour le cas de l'ordonnancement par flot, il peut, cependant, être directement étendu pour un algorithme d'ordonnancement par agrégats. D'où, on peut dire qu'un noeud offre à l'ensemble de la classe EF une guarantie sur la proportion $R$ avec $E$ comme latence s'il guarantit que n'importe quel paquet de l'ensemble sera transmis durant $F(p^k) + E$ avec

$F$ definie comme suit :

$$F(p^0) = 0 \tag{5.7}$$

$$F(p^k) = \max\{a^k, \min\{e^{k-1}, F(p^{k-1}\}) + \frac{l^k}{R'} \tag{5.8}$$

Dans ses travaux, Jiang [11] montre que, durant un intervalle de temps $[t, t+\tau]$, la totalité du traffic d'un flux $i$ non agrégé ou d'un ensemble de flux agrégés en un agrégat $i$, satisfait toujours la condition suivante :

$$A_i(t, t+\tau) \leq C_h\tau + L_{\max} \; ; \tau \geq 0.$$

Puis il généralise ses résultats pour montrer que si :

- on a $I_h$ liens entrants l(l=1,2,....,$I_h$) pour un lien sortant $h$,
- chaque lien entrant sera contraint par $(\sigma_l, \rho_l, C_l, L_{\max})$,
- le traffic EF de tous les liens entrants seront agrégé de manière FIFO,

alors le traffic agrégé sera contraint par $(\sigma_h, \rho_h, P_h, L_h)$, où :

$$\sigma_h = \sum_{l=1}^{I_h} \sigma_l, \quad \rho_h = \sum_{l=1}^{I_h} \rho_l, \quad Ph = \sum_{l=1}^{I_h} C_l \quad et \quad L_h = \sum_{l=1}^{I_h} L^{\max} = I_h L^{\max}.$$

Le délai expérimenté par un paquet EF dans un noeud GR est donné comme suit :

$$d_h = \frac{\sigma_h - L_h}{R_h} . \frac{(P_h - R_h)^+}{P_h - \rho_h} + \frac{L_h}{R_h} + E_h$$

Maintenant on peut dériver un délai du pire cas pour le délais de bout en bout pour le réseau considéré avec une topologie arbitraire avec une certaine condition sur le facteur d'utilisation $\alpha$ :

Si $\alpha < \min_h \frac{P_h}{(H-1)(P_h - R_h)^+ + R_h}$, alors :

$$D = \frac{H}{1-(H-1)\mu\alpha}(\mu\beta + E') \tag{5.9}$$

où

$\mu = \max_h\{\mu_h\}, \quad \mu_h = \frac{(P_h - R_h)^+}{P_h - \alpha R_h}, \quad et \quad E' = \max_h\left\{\frac{(1-\mu_h)L_h}{R_H} + E_h\right\}$

## 5.3 Conclusion

Comme mentionné dans la section 4.2.2, le degré de protection est inversement proportionnel à celui de l'agrégation. En effet, tous les flots d'une même classe ne sont pas protégés les uns par rapport aux autres.

Dans les travaux de Parekh et Gallager [3] [4], c'est l'isolation des flots ainsi que le lissage du trafic qui ont permis de calculer des bornes sur les délais de bout en bout.

Puisqu'au sein d'un même agrégat, les flots ne sont plus isolés les uns par rapport aux autres, il est très difficile de trouver des bornes déterministes précises, même dans le cas où le trafic n'est pas très variable. De plus, ces bornes sont valides pour des facteurs d'utilisation très restrictifs et restent encore peu précises. Dans le cas d'un service garanti (Hard QoS), un contrôle d'admission avec des règles basées sur ces bornes, n'est pas efficace et mène à une très grande sous utilisation du réseau.

# Chapitre 6

# CBWFQ adaptatif

## 6.1 Un CBWFQ adaptatif : pourquoi ?

### 6.1.1 Préliminaires

Du fait même de la non préemption des paquets (4.1.3), l'équité telle que définie dans la section (4.1.1) ne peut être assurée à chaque instant sur un routeur. Toutefois, l'emploi de celle-ci dans la définition d'une politique d'ordonnancement (telle que WFQ) permet d'approcher cette équité sur une échelle de temps suffisamment large [1].
Un point important est toutefois à noter concernant les politiques d'ordonnancement de type GPS ou WFQ : ces politiques prennent en compte la notion d'équité entre sessions sans tenir compte directement des délais de traitement pouvant être requis par chacun des paquets arrivant sur le routeur. En effet, sous GPS on parle d'équité relativement à la proportion des ressources allouées à une session donnée (qui découle directement du poids qui lui est associé). Ainsi en favorisant ou non l'émission de paquets de certaines sessions, les poids vont directement influer sur leur délai de bout en bout : un facteur important de qualité d'une session. Par ailleurs, comme le soulignent Sivaraman et al. [38], le fait d'employer des poids fixes peut conduire à l'obtention d'ordonnance-

ments insatisfaisants (au regard des délais), et un réajustement des poids peut s'avérer nécessaire pour améliorer les performances de GPS (WFQ) en termes de respect des délais associés aux paquets.

Dans un *Class-Based Weighted Fair Queuing*, l'allocation de la bande passante se fait en affectant un poids $w$ pour chaque file d'attente. On peut aussi parler d'une affectation de poids par classe de service. En effet, un CBWFQ est un WFQ dont le nombre de files correspond exactement au nombre de classes de services. Ce poids représente la proportion de bande passante qui lui est allouée. Étant donné un poids attribué à une classe de services particulière, le calcul de la bande passante qui lui est allouée se fait comme suit : $\frac{w_c}{\sum\limits_{c \in C} w_c} \times B_c$, où $w_c$ représente le poids actuel de la classe, $B_c$ la bande passante disponible en sortie et $\sum w$ la somme des poids de l'ensemble $C$ des classes. Cependant, un CBWFQ traite des files avec des poids statiques : dépendemment de ce qui se passe dans le réseau, ces poids ne changeront pas au cours du temps. Avec des poids statiques, il n'est pas toujours possible de respecter les critères QdS des applications temps réel. Notre but dans ce chapitre est d'essayer de respecter ces critères en utilisant une politique d'ordonnancement équitable variante de WFQ.

### 6.1.2 La congestion équivaut-elle à une dégradation de la QdS ?

Il est possible, dans le cas d'un CBWFQ, que certaines classes de services aient une proportion de bande passante allouée supérieure à ce qu'elles ont besoin. Si tel est le cas dans tous les nœuds traversés par un paquet entre une même paire de nœuds origine et destination, alors les critères QdS vont être respectés. Il est aussi possible que d'autres classes se voient attribués moins de bande passante que ce dont elles ont besoin. Pour les classes de services sensibles au délai et/ou à la gigue, vouloir émettre plus que la bande passante allouée, peut mener au non respect de certains critères QdS à travers l'augmentation des tailles des tampons et par conséquent des délais d'attentes dans ces files.

Supposons qu'on est dans le cas où le trafic temps réel est supérieur à la proportion

de bande passante allouée. Est ce que WFQ (CBWFQ) permet d'aider à garantir le respect des critères QdS des applications temps réel ? Plusieurs cas se présentent que nous décrivons ci-dessous.

- Les files NTR n'utilisent pas la totalité de la bande passante qui leur est allouée. Dans ce cas, et selon le critère *max-min-fair-share* auquel adhère WFQ, la quantité de bande passante non utilisée va être partagée sur les files temps réel. Dépendemment du taux d'utilisation du lien en sortie, plusieurs cas se présentent aussi.
  - Le lien en sortie d'un nœud est sous-utilisé : cela veut dire que la quantité de bande passante non utilisée par les files NTR peut être suffisante pour combler la demande excessive en bande passante de la part des applications TR. Dans ce cas, le critère *max-min-fair-share* aidera à la garantie de la QdS. En fait, ceci n'est plus vrai quand l'excédent de la part des classes NTR est inférieur à la portion demandée en excès par les classes TR. Dans ce cas, l'interface se trouve en état de congestion.
  - Le lien en sortie est sur-utilisé : cela veut dire que la quantité de bande passante non utilisée par les files NTR n'est pas suffisante pour combler la demande excessive en bande passante de la part des applications TR. On se retrouve dans un cas de congestion au niveau de l'interface de sortie du même lien.
- Les files NTR utilisent la totalité de la bande passante qui leur est allouée vu les demandes excessives de la part des deux types d'applications. Il y a donc sur-utilisation du lien en sortie. L'interface de sortie se retrouve en état de congestion.

Notre intérêt se porte sur les cas où l'interface se trouve en état de congestion. Cette congestion peut être juste locale au nœud et n'implique pas nécessairement un état de congestion généralisé à tout le réseau. En pratique, il existe une valeur de délai de bout en bout limite tolérée pour garder une bonne qualité de service pour les applications temps réel telles que la voix ou la vidéo. Ce délai a en général une valeur située entre $150ms$ et $200ms$. Si, dans l'un des nœuds du réseau, le délai d'attente dans une file temps réel augmente, mais que le délai de bout en bout reste en deçà de la limite per-

mise, alors le niveau de QdS reste satisfait. Aussi, si tous les nœuds traversés par un paquet sont en état de congestion, mais que le délai de bout en bout reste aussi inférieur à la limite permise, alors on ne dégrade pas le niveau de QdS.

Notre conclusion est qu'un critère de QdS comme le délai est un indice surlequel on pourrait se fier pour dépister une éventuelle dégradation au niveau de la QdS des applications TR. Mais, peut-on mesurer le niveau de congestion à partir duquel on dégrade ou on risque de dégrader les critères QdS ?

### 6.1.3 Niveau de congestion responsable de la dégradation de la QdS

Il est vrai qu'à un certain niveau de congestion bien déterminé, on ne peut plus respecter les critères QdS. Cependant, il est difficile de prédire ou de mesurer le niveau de congestion qui causera une éventuelle dégradation des critères QdS. En effet, plusieurs éléments, parfois imprévisibles, peuvent s'imbriquer pour rendre cette mesure difficile à réaliser.

- ✓ Le nombre de nœuds par lequel passe un paquet. En cas de congestion, les délais d'attente encourus dans chacun des nœuds viennent s'ajouter à la valeur du délai de bout en bout. Ces délais ne sont généralement pas négligeables et représentent une bonne proportion du délai de bout en bout. Ainsi, plus le nombre de nœuds traversés est grand, plus le risque de dépasser la limite du délai est élevé.
- ✓ La durée des profils des applications multimédias. Si un nœud est en état de congestion et qu'il continue à recevoir la même quantité de trafic qui a causé cet état, alors la taille des files d'attentes va continuer à augmenter et on se retrouve dans un état de congestion de plus en plus forte. Par contre, supposons que la durée des applications temps réel est très courte. Si, à un moment donné, on a noté l'entrée dans le réseau de plusieurs applications de ce genre, et qu'on se retrouve en état de congestion, alors on devrait s'attendre à la non persistence d'une éventuelle augmentation du délai de bout en bout pour ces applications.

- ✓ La quantité d'applications non temps réel présentes dans le réseau. Si ce type d'applications utilise un faible pourcentage des ressources qui lui sont allouées, alors l'excédent va être redistribué sur les applications temps réel.

### 6.1.4 Ajustement des poids

Le volume du trafic temps réel diffère d'une période à une autre. Si le volume de trafic de l'une de ces classes augmente au delà de la bande passante allouée, on pourrait décider de favoriser cette classe en haussant son poids. Le but est d'essayer de respecter au mieux les critères QdS de chaque classe, et surtout ceux des classes temps réel. Pour ces dernières, on va essayer de diminuer les délais d'attentes dans les files qui représentent la composante la plus critique lors du calcul du délai de bout en bout auquel les applications multimédias sont particulièrement sensibles.

Une allocation statique des ressources avec WFQ ou CBWFQ n'est certainement pas désirable. Notre priorité a été de penser à respecter les critères QdS des applications TR précisément en termes de délai et de gigue tout en optimisant l'utilisation des ressources disponibles et en respectant l'équité au sens de WFQ, soit d'un partage équitable de la bande passante au prorata des bandes passantes requises pour chaque classe d'applications.

Nous avons pensé à présenter une solution permettant d'adapter les ressources allouées et de satisfaire les classes temps réel quand ces dernières en ont besoin tout en gardant un seuil minimal de ressources disponibles pour les applications non temps réel. La variabilité des poids est une notion qui nous permettrait de favoriser les applications temps réel sensibles aux délais en augmentant leurs poids au détriment des applications non temps réel. Il faut s'attendre à voir augmenter la taille des files non temps réel et dépendemment de la sensibilité des applications non temps réel à la perte des paquets, on doit prévoir des tampons de plus grandes tailles pour éviter les pertes de paquets.

Étant donné que la cause de la dégradation du service est le non respect d'un critère QdS et que les applications temps réel sont sensibles au délai et à la gigue, nous nous baserons sur un de ces critères pour déclencher la procédure d'ajustement de poids.

La valeur de la gigue n'est pas très significative sans une idée sur la valeur du délai de bout en bout. En effet, on peut avoir simultanément une petite gigue et un grand délai. C'est le cas où la valeur du délai est grande mais qu'elle ne varie pas au cours d'un intervalle de temps $[t, t+n]$.

La valeur du délai de bout en bout est plus utile et nous utiliserons donc ce délai comme critère pour décider de l'instant où on procédera au changement de poids. L'idéal serait de pouvoir calculer à priori le délai de bout en bout que prendra un paquet pour arriver à destination et de vérifier que cette valeur nous permettra de rester en dessous de la limite sur le délai. Ceci est possible avec les calculs de bornes sur le délai présentés dans le chapitre 5.

Mais comme on l'a dit, ces bornes sont souvent peu précises. Au niveau d'un nœud, le seul moyen direct pour prendre une décision qui soit basée sur les valeurs réelles du délai est de mesurer en temps réel le délai de bout en bout encouru par un paquet arrivé à destination.

Dans ce qui suit nous allons principalement expliquer, dans un premier lieu, la façon avec laquelle nous avons procédé pour implémenter ce mécanisme, et dans un deuxième lieu, la procédure utilisée pour le changement des poids. Le CBWFQ est la politique d'ordonnancement prise en compte par tous les nœuds du réseau, qu'ils soient des nœuds extrêmes ou appartenant au coeur du réseau. Ce CBWFQ fait la différentiation entre quatre classes de services : deux classes temps réel (vidéo et voix) et deux autres non temps réel (courriel, web).

## 6.2 Algorithme de routage

Comme le type de service choisi est un service en mode sans connexion, les paquets ou datagrammes sont routés indépendemment les uns des autres. Dans chaque nœud du réseau, une table de routage est définie. Ainsi, lors du passage d'un paquet par un nœud particulier, on va consulter cette table et décider du chemin que suivra le paquet pour arriver à sa destination. La table de routage contient des entrées dont chacune est

composée d'une adresse de destination et une ligne ou interface de sortie à emprunter pour atteindre cette destination. Dans la plupart des cas, un nœud n'est pas directement lié à la destination d'un datagramme particulier. Dépendemment de l'algorithme de routage, le nœud décidera de la ligne de sortie qu'empruntera le datagramme pour se rapprocher de sa destination et ainsi du nœud directement lié à la destination.
Considérons les tables de routages de deux nœuds B et C dans la figure 6.1 et le réseau utilisé dans la figure 6.2. Un paquet envoyé par $S1$ et destiné à $S2$ va emprunter le

**Table de routage de B**

| Destination | Ligne de sortie |
|---|---|
| A | A |
| B | - |
| C | C |
| C | F |
| D | C |
| D | F |
| E | C |
| E | F |
| F | F |
| F | C |
| G | F |
| G | C |

**Table de routage de C**

| Destination | Ligne de sortie |
|---|---|
| A | B |
| A | F |
| B | B |
| B | F |
| C | - |
| D | D |
| D | F |
| E | D |
| E | F |
| F | F |
| F | D |
| F | B |
| G | F |
| G | D |
| G | B |

FIG. 6.1 – Tables de routage statique de deux nœuds

chemin spécifié dans la figure 6.2. Nous supposons que les tables de routage respectives des nœuds A, D, E, F et G sont déjà définies. Les tables de routages sont spécifiées dès le début : il existe une ligne de sortie possible pour chaque nœud destination du réseau. Nous considérons tous les nœuds comme des nœuds destination puisque chacun peut jouer le rôle d'un nœud d'extremité. Nous avons alors défini, dans chaque table de routage, plusieurs lignes de sorties permettant chacune d'atteindre une destination potentielle dans le réseau autre que le nœud lui même. En effet, l'algorithme de routage aura à emprunter un même chemin à partir d'une source pour arriver à une destination particulière durant toute la durée d'une simulation. Dans notre étude, les tables de routages sont *statiques*. Considérant une source $S1$ envoyant du trafic vers $S2$, les paquets envoyés de $S1$ vers $S2$ passeront toujours par les mêmes nœuds au cœur du

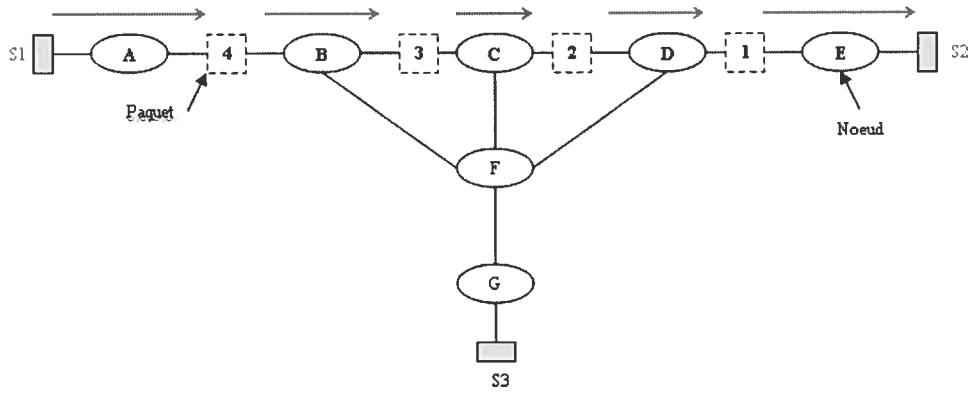



FIG. 6.2 – Routage statique des paquets entre une source et une destination

réseau. Nous avons fait l'hypothèse dans notre étude qu'un paquet allant de $S2$ vers $S1$ prenne un chemin inverse à celui pris par un paquet allant de $S1$ vers $S2$. Ainsi, qu'il soit issu d'une paire source destination $S1/S2$ ou $S2/S1$, un paquet $p$ passera toujours par les mêmes nœuds.

### Mise en œuvre dans OPNET

Pour implémenter un tel routage, on doit concevoir un schéma d'adressage complet de tout le réseau. L'attribution des adresses IP à chaque terminal ainsi qu'aux interfaces de chaque nœud est effectuée manuellement. La classe utilisée pour le masque de réseau est la classe C [65] et le masque de sous réseau est 255.255.255.0. Avec l'environnement OPNET et les outils visuels offerts, il est possible de configurer les adresses IP pour chaque terminal/serveur ainsi que pour chaque interface de n'importe quel nœud du réseau. Grâce à ces outils visuels, il est aussi possible de facilement configurer la table de routage dans chaque nœud.

STRUCTURE D'UN NŒUD

Un niveau d'abstraction inférieur intitulé *modèle de nœud (node model)* est employé pour décrire la structure interne d'un nœud. Celui-ci comporte :

- ➤ des modules, chacun caractérisé par un comportement défini par le modèle de processus ;
- ➤ des liens entre modules (unidirectionnels ou bidirectionnels), assurant le transport de l'information entre modules ;
- ➤ des interfaces de deux types : des interfaces de type série et d'autres de type *Ethernet*. Comme on le voit dans l'exemple présenté dans la figure 6.3, le nœud dispose de deux interfaces Ethernet et de huit interfaces série.

Comme on le remarque, les différentes fonctions sont réparties sur plusieurs couches (modules) pour ainsi suivre un modèle dont le concept est basé sur celui du modèle TCP/IP. Chaque nœud IP possède un module de routage qui contient un *répartiteur*

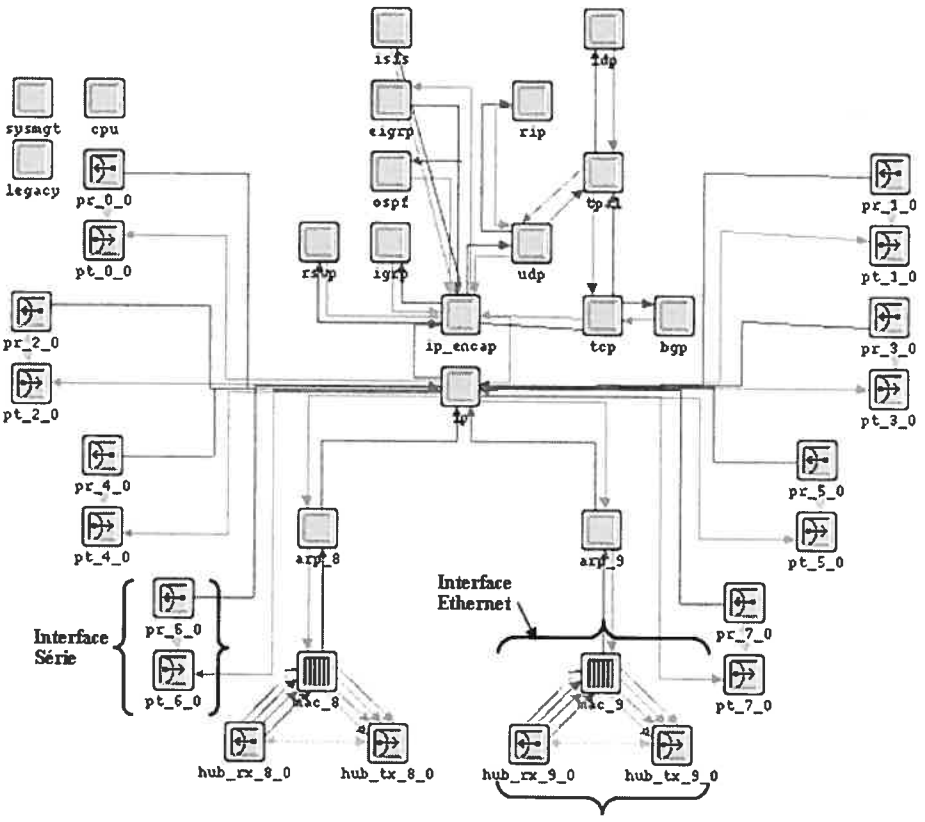



FIG. 6.3 – Structure d'un nœud IP dans OPNET [46]

ou *dispatcher*. Celui-ci génère les différents processus de routage. OPNET offre trois types de modules de routage : centralisé (centralized), basé sur des slots (slot-based),

nuage (cloud). Nous utilisons le routage centralisé à travers le processus correspondant qu'est *ip-rte-central-cpu*. Quant au *répartiteur*, il est réalisé à travers le processus *ip-dispatch*.

A chaque module présent dans un nœud est associé un modèle de processus. Celui-ci peut être décrit par un ensemble d'attributs et de fonctions et se trouve composé d'états et de transitions auxquels sont associés des codes sources. Le comportement du modèle de processus réel est simulé par son passage dans une succession d'états et de transitions. Ce comportement est implémenté au moyen du langage de programmation Proto-C, analogue au langage C. *ip-dispatch* est le modèle de processus associé au module IP. Il a comme fonctions principales le routage, la fragmentation et le réassemblage. Les paquets IP arrivant dans n'importe quelle interface du nœud sont routés vers la bonne interface de sortie en se basant sur leurs adresses de destination. Une vue d'ensemble du processus IP est présentée à travers la figure 6.4.

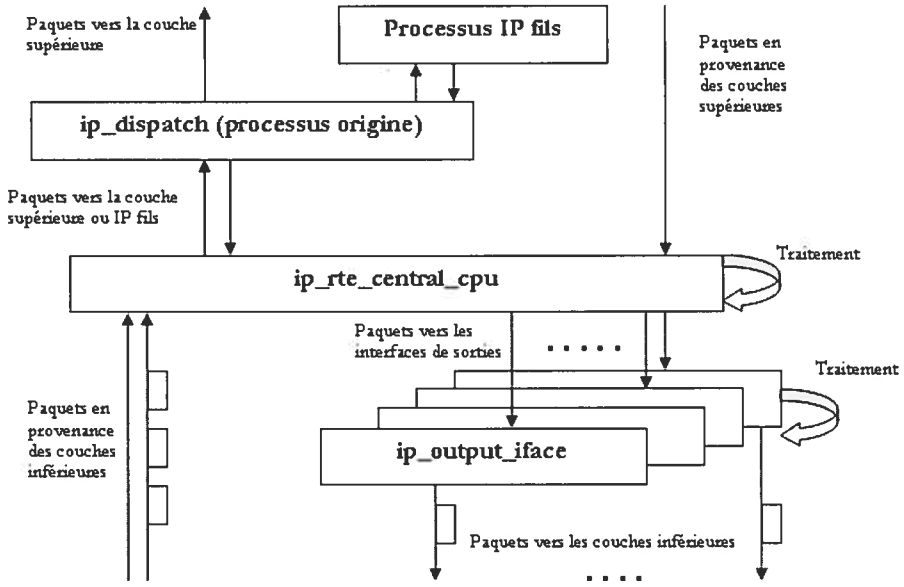



FIG. 6.4 – Le module IP comme défini dans OPNET [46]

## 6.3 Mesure du délai

Le calcul du délai de bout en bout s'effectue au niveau de la station réceptrice du flot. Sachant que l'information sur le moment de création d'un paquet est incorporée dans son entête, le délai de bout en bout $d$ se définit comme suit.

$$d = t_{current} - t_{start}.$$

où $t_{current}$ étant le temps courant et $t_{start}$ le temps de création du paquet. Au niveau de la station réceptrice, et pour chaque session, on crée une liste (liste chainée, tableau statique ou vecteur) pour l'application considérée (classe de service). Chaque élement créé de la liste est initialisé à zéro. Pour le cas d'un tableau statique, on initialisera, dès le début, tous ses éléments à zéro. La liste permet, peu importe le flot reçu de cette même application, d'y stocker les valeurs des délais mesurés. Au début, cette liste est vide puis elle sera remplie au fur et à mesure de la réception des paquets. Trois stratégies sont possibles pour le remplissage de la liste.

- Stocker les $n$ dernières mesures des délais encourus par les $m$ derniers paquets du flot agrégé où $n$ et $m$ sont des paramètres configurables avec $n \leq m$, $m = p \times n$ et $p \geq 1$ où $n$ est une valeur constante. $p$ représente le nombre de paquets à attendre avant de stocker de nouveau une valeur de délai dans la liste.

  Au fur et à mesure de la réception des paquets, on procède à l'insertion de la valeur du délai à la tête de cette liste et on calcule une moyenne $\overline{d}$ sur les $u$ éléments nons nuls de la liste (avec $u \leq n$). Cette liste prend le rôle de *fenêtre coulissante* de taille fixe à laquelle on insère et retire des éléments.
- Stocker les $n$ mesures des délais encourus pendant l'intervalle de temps $[t_{current} - t, t_{current}]$ avec $n \leq N$ et $N = p \times n$, $p \geq 1$ où $N$ ést le nombre total de paquets reçus durant $[t_{current} - t, t_{current}]$. $t$ étant un laps de temps à définir. Dans ce cas, $n$ n'est plus constant.

  Au fur et à mesure de la réception des paquets, on procède à l'insertion du délai à la queue de cette liste tout en calculant la moyenne arithmétique $\overline{d}$ des $u$ éléments

nons nuls de la liste ($u \leq n$).

- Une combinaison des deux stratégies précédentes. C'est-à-dire stocker une mesure par $k$ paquets reçus durant un intervalle de temps $[t_{current} - t, t_{current}]$.

La valeur de $\overline{d}$ sera incorporée dans le champs *Option* de l'entête $IP$ du prochain paquet à envoyer dans l'autre sens.

Nous avons choisis de stocker les $n$ valeurs des délais dans un tableau de taille égale à 5 et dont chaque élément est initialisé à zéro. Un décalage des $n - 1$ premières valeurs du tableau est effectué pour chaque valeur insérée à la tête du tableau. A la fin de la session, on procède à la suppression de ce tableau pour éviter des problèmes de mémoire.

### Mise en œuvre dans OPNET

Le délai de bout en bout est calculé dans le terminal/serveur destination du flot. Les structures d'un terminal et d'un serveur sont très semblables. Voici celle d'un terminal. Ce délai est calculé au niveau du module application équivalent à la couche application du modèle TCP/IP par l'intermédiaire du modèle de processus principal *gna-clsvr-mgr*. Dépendemment de la situation d'un terminal, qu'il soit émetteur ou récepteur d'un flot en particulier, mais aussi du type d'application, ce processus va faire explicitement ou implicitement appel à un processus fils bien spécifique : le *calling-process-party* et le *called-process-party* (figure 6.6). Chacun de ces deux processus existe sous deux instances différentes dont l'une est appelée lors du traitement d'une session voix et l'autre lors du traitement d'une session de type vidéo. Par exemple le processus *gna-clsvr-mgr* d'une station réceptrice d'un flot vidéo va appeler directement le processus *video-called-process-party*. Le *called-process-party* et le *calling-process-party* sont les deux processus fils où l'on procède à des changements dans le code source d'OPNET pour le calcul de $\overline{d}$.

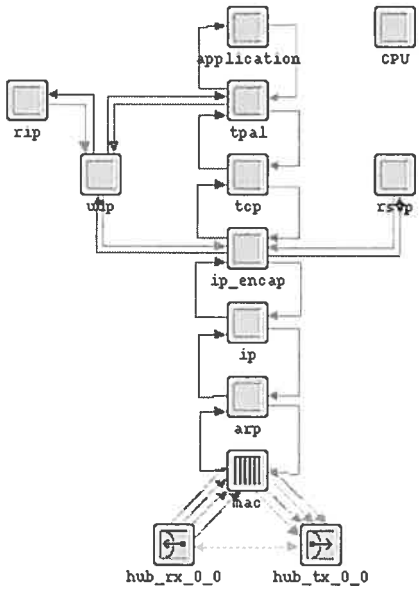



FIG. 6.5 – Structure d'un terminal dans OPNET [46]

## 6.4 Transport du délai dans le réseau

Les applications temps réel sont des applications caractérisées par un échange d'informations entre deux ou plusieurs parties. Généralement, c'est l'une des parties qui fait la demande pour l'établissement d'une communication et ainsi réaliser un échange de flux dans les deux sens. Ceci est bien illustré dans la figure 6.7. Pour les applications temps réel choisies (*voix*, *vidéo conférence*), nous avons choisi de modéliser le trafic de sorte que la quantité d'information contenue dans le flot montant soit exactement la même que celle contenue dans le flot du sens descendant. Les applications temps réel sont très sensibles au délai et à ses variations. Pour mieux répondre à ces exigences nous avons choisi de procéder à la mesure des délais en temps réel et de faire des changements, en conséquence, juste pour ce type d'applications. Comme on l'expliquera plus tard avec détail, nous avions besoin de récupérer le délai encouru par un paquet appartenant au flot montant (ou descendant) et arrivant à destination, et ce, par tous les nœuds qu'il avait traversé durant son chemin avant d'atteindre cette destination. Autrement dit nous

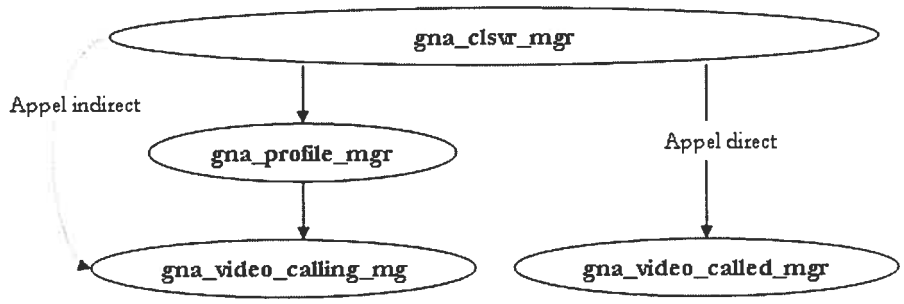



FIG. 6.6 – Relations entre différents modèles de processus pour le cas de la vidéo

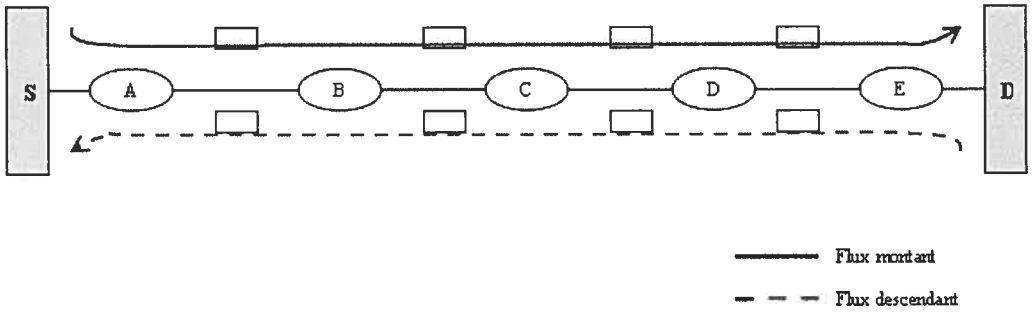



FIG. 6.7 – Flots montants et descendants entre deux stations

voulions que ces nœuds intermédiaires puissent avoir accès, en temps réel, aux délais encourus par des paquets y ayant déjà transité. Chacun de ces nœuds utilisera cette information pour procéder, rapidement, à des changements à sa politique d'ordonnancement au niveau de la proportion de bande passante allouée en sortie pour chaque type d'applications via l'attribution de valeurs de poids supérieures. Ces nouvelles valeurs vont favoriser les applications temps réel en leur allouant plus de bande passante, ce qui aidera à la réduction des délais d'attente dans les files, et par conséquent, des délais de bout en bout.

Soit S une station échangeant de l'information avec D. S ne peut pas savoir instantanément le délai qu'a mis un paquet pour arriver à D.

Deux moyens sont possibles pour communiquer ces délais à la source tout en passant par les mêmes nœuds que ceux qu'avait pris le paquet responsable de ce délai (sections 6.4.2 et 6.4.3).

### 6.4.1 RTCP comme moyen d'amélioration de la QdS

Le protocole RTCP [50] est basé sur des transmissions périodiques de paquets de contrôle par tous les participants (source et destination) dans une session de type temps réel. C'est un protocole de contrôle des flux RTP [50], permettant de véhiculer des informations de base sur les participants d'une session et sur la QdS (délai, gigue, paquet perdus...).
Cependant, RTCP n'est pas retenu pour servir de support de transport de $\overline{d}$ pour les raisons décrites dans la section 6.4.4

### 6.4.2 Mécanisme avec utilisation de flux existants

Une hypothèse déjà faite dans notre étude, est celle de disposer d'un flot dans le sens descendant utilisant la même route que dans le sens montant. Cette hypothèse nous permet d'incorporer l'information sur le délai relatif à un flot montant, au niveau du flot descendant correspondant. La structure d'un paquet IP n'a cependant pas été changée. Sachant maintenant que l'information $\overline{d}$ va être utilisée par les modules *IP* des nœuds intermédiaires pour faire des changements au niveau de leurs ordonnanceurs WFQ, la question d'ordre pratique posée est : à quel niveau incoporer $\overline{d}$ pour qu'il soit disponible dans la couche IP de chaque nœud ? On a pensé à incorporer le délai, qu'on calcule dans la couche application, dans la même couche. Le fait que les données soient encapsulées lorsqu'elles passent d'une couche à une autre, nous obligerait à faire la décapsulation des paquets au niveau de chaque nœud pour pouvoir récupérer cette information. Or, au niveau de la source, la couche IP procède à la fragmentation des segments reçus de la couche supérieure avant de les envoyer vers la couche inférieure. La fragmentation,

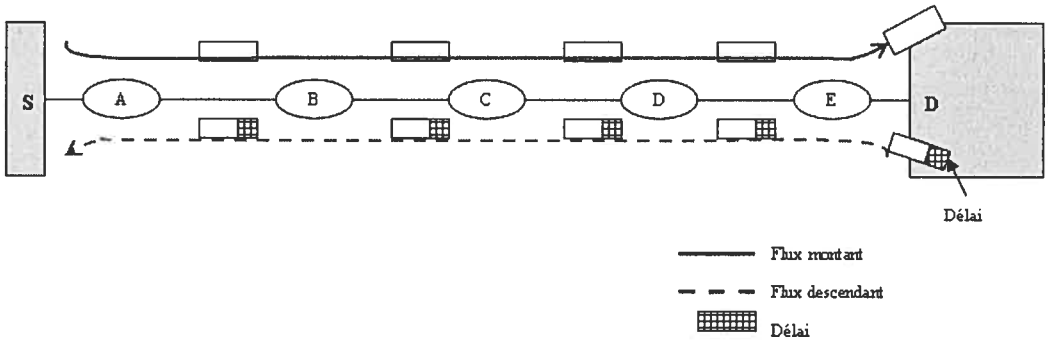



FIG. 6.8 – Délai de bout en bout incorporé dans un flot descendant

propriété du protocole IP, sert à garder une taille fixe des paquets. Ceci pose donc un problème vu que l'information sur le délai, incorporée au niveau application, ne sera plus disponible si on défait l'encapsulation du paquet au niveau de la couche IP des nœuds intermédiaires. La meilleure solution est d'incorporer $\overline{d}$ au niveau de la couche IP de la source. Il faudrait donc trouver une façon pour que la couche application puisse informer la couche IP de cette donnée sans l'incorporer dans une structure qui sera fragmentée au niveau IP. Pour le reste, transporter $\overline{d}$ de la couche *IP* de la source à celle de la destination en passant par les nœuds intermédiaires est facile à faire puisque toutes les couches *IP* parlent le même langage qu'elles appartiennent à un terminal ou à un nœud.

### 6.4.3 Mécanisme avec création de nouveaux flux

Une autre possibilité n'imposant pas aux flux montants et descendants d'emprunter le même chemin, tout en ayant la valeur de $\overline{d}$ disponible pour certains nœuds, est la création de nouveaux flux pour le transport de $\overline{d}$.

Pour, ainsi, transporter les informations sur les délais calculés à la destination d'un ou plusieurs paquets appartenant à un flux $f$, il est possible de créer de nouveaux flux dans le sens contraire qui auront la tâche d'aviser un groupe de nœuds en particulier.

Ces flux prennent la forme d'accusés de réception de la valeur du délai calculé, et ce,

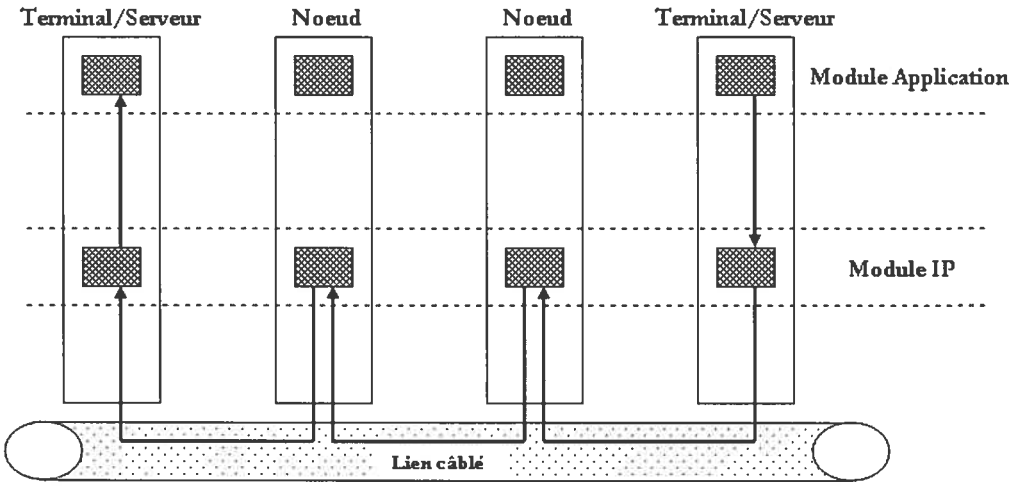



FIG. 6.9 – Cycle de vie d'un paquet entre une source et une destination

jusqu'au nœud destination. Ces flux traverseront plusieurs nœuds, en particulier ceux par lesquels le flux montant est déjà passé.

L'idée est de créer un flux séparé qui va s'occuper seulement du transport du délai tout en s'assurant d'emprunter la même route que le paquet reçu. Le nœud d'extrémité auquel est rattaché la station de destination devra être en mesure de reconnaître la route empruntée par le flux montant et s'assurer que le flux descendant en prenne chemin. Sinon ce sera à la station destinataire d'ajouter cette information au paquet sortant. Cela nécessite la conception de tout un protocole de transport des valeurs des délais et une traçabilité au niveau des chemins empruntés par les paquets de type temps réel. Cependant nous avons opté pour une autre méthode moins complexe.

### 6.4.4 Discussions

Le protocole RTCP n'est pas retenu pour servir de support de transport de $\overline{d}$ pour les raisons décrites ci-dessous.

- ✔ RTP et RTCP sont des protocoles de la couche application. Les entêtes RTP et RTCP ne sont pas disponibles au niveau des nœuds intermédiaires (routeurs), vu

que les paquets routés au niveau d'un nœud ne vont pas monter plus haut que la couche réseau du modèle TCP/IP.

- ✔ Le but de RTCP est d'envoyer des paquets de contrôle aux participants d'une même session. Les informations sur la QdS ne seront disponibles et utilisées que par les participants d'une session.

Notons aussi qu'OPNET ne contient pas de bibliothèques spécifiques permettant de simuler les protocoles RTP et RTCP.
Il aurait été, certes, préférable d'implémenter le mécanisme décrit dans la section 6.4.3 mais certaines contraintes nous en ont empêché.

- ➢ Notre but est d'explorer les performances d'un CBWFQ adaptatif en se basant sur des mesures de délais, mais pas d'implémenter une solution complète de gestion de réseau.
- ➢ Cela prendrait énormément de temps à implanter sous OPNET. Pour cela il faudrait générer de nouveaux flux et s'assurer de bien les gérer. Il faudrait recréer nos propres protocoles de transport pour permettre, par exemple, à un paquet transportant la valeur $\overline{d}$ dans le sens descendant, d'emprunter le même chemin que celui pris par les paquets dans le sens montant.

Nous avons, donc, implanté le mécansime de la section 6.4.2.

### 6.4.5 Mise en œuvre dans OPNET

La valeur moyenne du délai $\overline{d}$ sera acheminée de bout en bout entre deux couches applications en passant par les différents nœuds intermédiaires. Lorsqu'on est dans un terminal source, les données émises à partir du module *application* passeront à travers le module *tpal*, *udp*, *ip-encap* pour arriver à *IP*. Lorsqu'on est dans un terminal destination, les données arrivant au niveau du module *IP* emprunteront un chemin inverse pour arriver à la couche application. Notons que pour les applications non temps réel les paquets passeront par le module *TCP* au lieu de *UDP*.
Pour communiquer entre eux, les processus OPNET se transmettent des messages grâce

à des entités de simulation appelés *ICI* (Interface Control Information). Les *ICI*s sont des collections structurées de données tranférables entre deux ou plusieurs processus. Quand un *ICI* est installé au niveau d'un processus, il devient associé à la prochaine interruption. Si cette interruption fait appel à un autre processus, alors cet *ICI* sera disponible au niveau de celui-ci.

Les données sont envoyées d'un module à un autre par l'intermédiaires d'interruptions. Par exemple, un processus déclenche une interruption pour envoyer des données vers un module inférieur. Le processus directement associé à celui-ci sera donc activé et sera capable de recupérer les données envoyées ainsi que l'*ICI* associé à cette interruption. Nous avons utilisé ce mécanisme pour le transport de $\overline{d}$ entre plusieurs modules. Il suffit d'incorporer $\overline{d}$ dans un *ICI*, et installer ce dernier juste avant le déclenchement de l'interruption. Au niveau d'un terminal destination, $\overline{d}$ n'est pas une valeur utile, c'est pourquoi l'utilisation d'*ICI*s va être limitée au terminal/serveur source entre les modules *application* et *IP*.

Reste à se préoccuper de savoir dans quelle structure sera incorporée la moyenne $\overline{d}$ pour être acheminée entre les modules *IP*. Comme nous l'avons déjà mentionné dans la section 6.3, $\overline{d}$ fait partie du champs *Option* de l'entête du paquet $IP$ (*IpT-Dgram-Fields*).*IpT-Dgram-Fields* est une structure prédéfinie dans OPNET permettant de spécifier toutes les informations utiles à la description d'un paquet $IP$ y compris son entête en vue de son transport dans le réseau. Comme mentionné précédemment, une meilleure solution aurait été de créer un mécanisme de flux séparé permettant aux différents modules *IP* de s'envoyer les valeurs de délais.

## 6.5 Traitement au niveau d'un nœud supportant CBWFQ

Lorsqu'un nœud reçoit un paquet au niveau d'une interface de la couche physique, ce paquet va ensuite traverser différentes couches avant d'atteindre la couche réseau qui est, entre autres, responsable de la fonction de routage. Dans un schéma de traitement usuel et dépendemment de l'interface de sortie à emprunter, le paquet va être enfilé dans

une file d'attente en attendant son traitement. Cependant, le paquet reçu contient une information importante (délai) servant, éventuellement, à ajuster les poids attribués à certaines files d'attente d'une même interface.

Dans ce qui suit, nous expliquons la procédure de lecture et de traitement du délai. Cette procédure se déclenche juste après l'enfilement d'un paquet dans la bonne file d'attente. Cependant, l'interface à laquelle appartient la file d'attente où ce dernier paquet a été stocké n'est pas celle où on va procéder à un éventuel ajustement de poids (figure 6.10). Dans ce qui suit, nous avons décrit la procédure permettant de vérifier certains critères avant de procéder à un ajustement.

### 6.5.1 Conditions nécessaires pour ajuster les poids

On considère un paquet $p$ du flot $f$, arrivant à un temps $t$ à l'entrée d'une interface $I$ d'un nœud $N$ et appartenant à un flux descendant. Selon la figure suivante, on note cette interface $I_{in}^{down}$. Ce paquet va être dirigé vers la file d'attente correspondante située dans l'interface $I_{out}^{down}$.

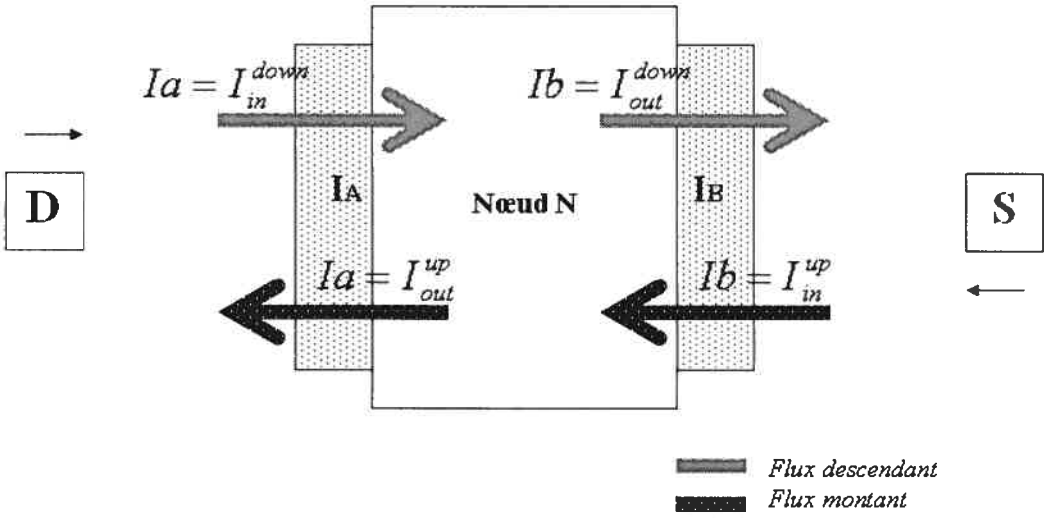



FIG. 6.10 – Flux montants et descendants vs interfaces d'entrées et de sorties

On procède, tout d'abord, à la lecture de l'information sur le délai $\overline{d}$ contenue dans le paquet. Après l'enfilement, on vérifie si $\overline{d}$ est supérieure à $d_{max}$, où $d_{max}$ est la valeur au delà de laquelle on estime qu'il y a un risque important de dégradation ou dégradation du niveau de la qualité de service des applications temps réel ($d_{max}$ choisie entre 150 et 200ms). Si c'est le cas, alors on cherchera à identifier l'interface $I_{out}^{up}$ où l'on doit procéder à des changements de poids pour réduire les délais. Sachant que $p$ appartient au flux montant, $I_{out}^{up}$ correspond à l'interface par laquelle est sorti un paquet $p'$ appartenant à un flux montant. Si, au niveau de ce nœud, $I_{out}^{up}$ est identifiée comme une interface responsable du dépassement de la borne $d_{max}$, alors on appellera la procédure d'ajustement des poids. Le passage du paquet en question par le reste des nœuds dans le sens descendant permettra d'identifier les autres interfaces responsables du dépassement de $d_{max}$.

On convient qu'une interface est en partie responsable d'un dépassement de la borne $d_{max}$ pour une certaine classe, si la taille moyenne du tampon alloué à celle-ci pendant un intervalle de temps $[t-\varepsilon, t]$ est supérieure à une valeur $Q_{min}$. Le choix de la valeur de $\varepsilon$ est argumenté dans la suite de cette section. Pour pouvoir calculer la taille moyenne du tampon pendant un intervalle de temps $[t-\varepsilon, t]$, on crée un tableau contenant les tailles des $k$ paquets reçus pendant un intervalle de temps $[t-T, t]$. On choisit la durée $T$ de sorte qu'elle soit supérieure ou égale à la plus grande valeur possible de $\varepsilon$.

Intuitivement, si le tampon est presque vide ($\leq Q_{min}$) pendant $[t-\varepsilon, t]$, alors cette interface n'est pas responsable du dépassement de $d_{max}$. La file de l'interface en question est presque vide, si chaque paquet stocké dans la file y reste pour un laps de temps très court. En effet, même si le paquet est enfilé dans une file vide, il doit attendre le service de l'ordonnanceur (occupé à servir les autres files). Ce délai est, certes, très court, mais le tampon s'est tout de même trouvé non vide pendant une brève période. On convient, alors, que $Q_{min}$ représente la taille d'un ou d'au plus quelques paquets.

Décrivons maintenant ce que doit représenter $\varepsilon$ et comment s'effectue son paramêtrage. Au temps $t$, l'interface $I_{in}^{down}$ voudrait bien connaître l'état exact (tailles des tampons) de l'interface $I_{out}^{up}$ par laquelle est passé le paquet $p'$ du flot montant pour avoir la taille exacte des tampons. Le passage de $p'$ par $I_{out}^{up}$ correspond au temps $(t-\varepsilon)$.

La principale raison qui rend le calcul de $\varepsilon$ difficile à effectuer c'est qu'à son arrivée, $p'$ sert juste au calcul du délai de bout en bout entre sa source S et sa destination D. Le terminal D ne sait pas le délai exact qu'a encouru le paquet $p'$ depuis son depart de $I_{out}^{up}$. Cependant $I_{in}^{down}$ peut très bien calculer le délai qu'a encouru $p$ en partance de D en se basant sur la date de création de ce paquet contenue dans son entête IP. Il faut, donc, trouver un estimé de $\varepsilon$ en plus du calcul d'une moyenne de la taille des tampons sur l'intervalle $[t-\varepsilon, t]$.

En supposant que le délai maximum toléré pour un paquet, afin d'effectuer le chemin d'aller-retour entre deux stations, est de $2 \times d_{max}$, le choix de la valeur estimée de $\varepsilon$ s'est fixé comme étant une moyenne du délai maximum d'aller-retour. $\varepsilon$ est, donc, choisie égale à $d_{max}$. $T$ est choisie supérieure ou égale à $\varepsilon$, et donc, supérieure ou égale à $d_{max}$

La procédure qui suit permet de résumer ce qui se passe avant le declenchement de la procédure d'ajustement des poids.

**Procédure *décision-ajustement(p,I)***

Pour chaque paquet $p$ entrant sur l'interface $I_{in}^{down}$ du nœud $N$ faire
Récupérer le $ToS$ de $p$.
Si ce $ToS$ correspond à une classe temps réel et si $\overline{d} \geq d_{max}$
    Identifier l'interface $I_{out}^{up}$ ainsi que $Q_{out}^{up}$ utilisé par le flux descendant ;
    Si $Q_{out}^{up} < Q_{min}$ pendant $[t-\varepsilon, t]$ alors sortir ;
    Appeler la procédure d'ajustement *changement-poids*($Q_{out}^{up}$, $I_{out}^{up}$)

Le $ToS$ (voir section 3.1) est la valeur contenue dans l'entête du paquet $p$ et qui permettra de le classifier.

La procédure *décision-ajustement* est déclenchée à chaque fois qu'un paquet de type voix ou vidéo est reçu sur une interface d'un nœud du réseau. Deux conditions principales ont été considérées pour procéder à l'ajustement.

- La valeur de $\overline{d}$ est supérieure à $d_{max}$. En effet, on ne changera pas les poids tant qu'on n'a pas dépassé la limite $d_{max}$.
- La taille moyennne de la file $Q_{out}^{up}$ durant l'intervalle de temps $[t-\varepsilon, t]$, est supérieure à un nombre de bits minimum $Q_{min}$.

### 6.5.2 Ajustement des poids

L'algorithme qui suit explique en détail la procèdure de changement de poids. Dans un intervalle donné, on calcule l'occupation relative pondérée de la file temps réel $Q$ par rapport au poids qui lui est associé. Le but est d'ajouter cette proportion de poids au poids $w$ que possède déjà la même file temps réel $Q$ tout en la retranchant proportionnellement aux files non temps réel. Du fait on favorise la file temps réel par rapport aux files non temps réel lorsque les contraintes de délais deviennent critiques. Cependant, notre algorithme ne permettra pas de défavoriser indéfiniment ces files. En effet, il s'assurera que les poids respectifs des files non temps réel ne descendent pas en deça d'un seuil minimum $w^{thres}$ qu'on choisira égal à 5 ($\sum w$=100).

#### Notations

Dans ce qui suit la signification des notations utilisées dans la procédure ci-dessus :

- $w(Q)$ le poids de la file $Q$.
- $w(I)$ le poids total des files de l'interface $I$ : $w(I) = \sum_{Q \in I} w(Q)$.
- $w^{NTR}(I)$ le poids total des files NTR de l'interface $I$ : $w^{NTR}(I) = \sum_{Q \in NTR} w(Q) = W^{Mail}(I) + W^{Web}(I)$.
- $w^{cut}(Q)$ le poids à enlever à la file non temps réel $Q$.
- $w^{thres}(Q)$ le poids minimal d'une file non temps réel $Q$
- $w^{add}(Q)$ le poids ajouté à la file TR $Q$ à la fin de la procédure de changement de poids

- $\overline{\ell}(Q)_{[t-\varepsilon,t]}$ la taille moyenne du tampon de la classe temps réel durant la période $[t-\varepsilon, t]$.
- $PR(Q)$ représente l'occupation relative pondérée de $\overline{\ell}(Q)_{[t-\varepsilon,t]}$ par rapport à $w(I)$.
- $BP(I)$ la bande passante totale au niveau de l'interface $I$ ou, tout simplement, le débit du lien à la sortie.

**Procédure *changement-poids(Q,I)***

On calcule $PR(Q)$ : $PR(Q) \leftarrow \left(\overline{\ell}(Q)_{[t,t-\varepsilon]}/BP(I)\right) \times w(I)$
On initialise $w^{add}(Q)$ à zéro : $w^{add}(Q) \leftarrow 0$ ;
Pour toute file non temps réel $Q'$ dans l'interface $I$ faire :

$w^{cut}(Q') \leftarrow PR(Q) \times \left(w(Q')/w^{NTR}(I)\right)$ ;
Si $\Big(\left(w(Q') - w^{cut}(Q')\right)$ est inférieur à $w^{thres}(Q')\Big)$ alors
$w^{cut}(Q') \leftarrow w(Q') - w^{thres}(Q')$;
$w(Q') \leftarrow w(Q') - w^{cut}(Q')$;
$w^{add}(Q) \leftarrow w^{add}(Q) + w^{cut}(Q')$;

Fin Pour
$w(Q) \leftarrow w(Q) + w^{add}(Q)$.

Les intuitions, quant aux choix effectués dans cet algorithme se décrivent comme suit. $\overline{\ell}(Q)_{[t-\varepsilon,t]}$ représente, comme décrit, la taille moyenne du tampon de la classe TR durant la période $[t-\varepsilon, t]$. Elle représente, entre autre, l'excès qu'il ne fallait pas avoir pour ne pas provoquer de longs délais d'attente dans $Q$ et risquer, ainsi, de dépasser $d_{max}$ (section 6.5.1). En supposant que cette moyenne sera la même durant les périodes à venir, et qu'un paquet devra, donc, subir les mêmes délais d'attentes que précédemment, il sera utile d'ajouter une valeur, ayant rapport avec $\overline{\ell}(Q)_{[t-\varepsilon,t]}$, au poids de la file en question $Q$. C'est ce qu'on a noté plus haut comme étant $PR(Q)$. La classe correspondant à la file $Q$ sera favorisée, puisqu'en en théorie, $Q$ sera vide.
Ce qui reste à faire, c'est d'enlever, proportionnellement aux poids des deux files NTR, la

valeur $\overline{\ell}(Q)_{[t-\varepsilon,t]}$ à la mesure du possible. En effet, si la proportion calculée de $\overline{\ell}(Q)_{[t-\varepsilon,t]}$, à enlever du poids d'une file NTR $Q'$, fait qu'on dépasse la limite minimale permise $w^{thres}(Q')$, alors on se contente de donner $w^{thres}(Q')$ comme nouvelle valeur de poids de $Q'$. Enfin, on ajoute la somme de toutes les proportions enlevées $w^{add}(Q)$ à la valeur du poids de la file TR $Q$.

### 6.5.3 Mise en œuvre dans OPNET

Pour chaque interface d'un nœud, OPNET crée une instance différente de procesus. Ce processus appelé *ip-output-iface* s'occupe entre autres de la gestion des files d'attentes dans cette interface. La figure (6.11) permet de visualiser certaines interactions qui peuvent exister entre ce processus et les fichiers auxquels ce processus fait appel dont les plus importants sont *oms-qm*, *ip-qos-support* et *oms-buffer*. Ces fichiers comportent les principales fonctions permettant de manipuler et de gérer les files que ce soit à la réception ou à l'envoi des paquets.

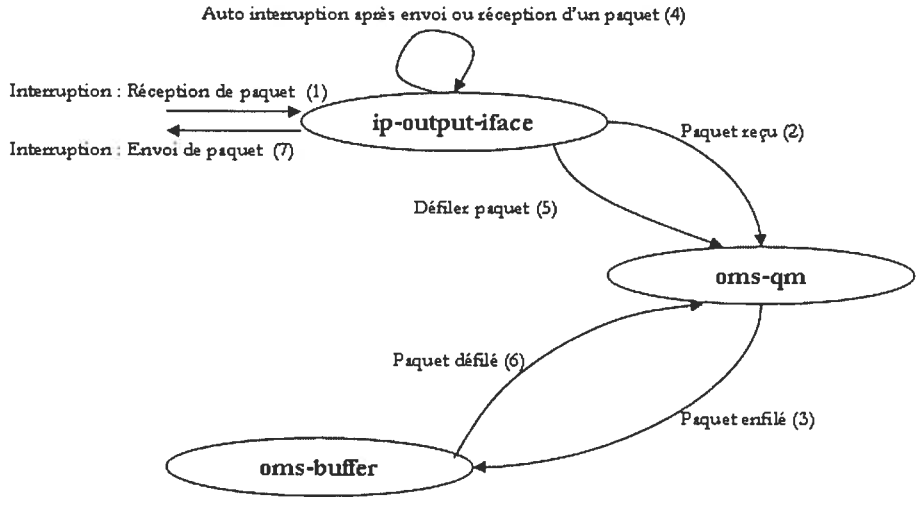



FIG. 6.11 – Modèle de gestion des files d'attentes dans OPNET [46]

Quand un paquet arrive dans le *ip-output-iface* cela déclenche une interruption (voir étape (1) dans la figure (6.11)). Ce processus va appeler la fonction de classification pour reconnaître la bonne file à choisir pour enfiler le paquet.
Une fois déterminée, le paquet sera enfilé dans la bonne file d'attente. C'est juste à ce niveau et après cet enfilement que nous ajoutons la fonction permettant de choisir la bonne interface et, éventuellement, d'y adapter les poids.

# Chapitre 7

# Résultats

Notre objectif consiste à étudier quelques politiques d'ordonnancement existantes notamment FIFO, WFQ et CBWFQ pour ensuite en élaborer d'autres variantes. Toutes ces politiques seront comparées selon des critères bien définis afin d'évaluer leurs performances lors de la gestion des flux de données que ce soit en cas de congestion ou non. La démarche adoptée est guidée par des contraintes importantes typiques aux réseaux multiservices se manifestant essentiellement par la diversité de la typologie des applications. Ces dernières ont des critères de QdS très différents. Les ressources restreintes, dont dispose en général un opérateur de réseau, telles que la taille des tampons (buffers), la capacité des liens, etc, sont aussi des contraintes à prendre en compte.

Notre étude expérimentale portera donc sur les caractéristiques et les performances des politiques d'ordonnacements utilisées. En effet, FIFO qui est une politique d'ordonnancement traditionnelle largement utilisée dans les réseaux actuels, n'est malheureusement pas capable dans certains cas de garantir les critères de QdS pour chaque application. Comme on l'a déjà expliqué, le problème majeur est que FIFO ne possède pas de mécanismes de différenciation entre paquets selon un ou plusieurs critères. Le but de cette étude comparative est de prouver que d'autres politiques alternatives existent, qu'elles peuvent être efficaces en matière de garantie de certains critères de QdS, et

qu'elles sont bien placées pour remplacer FIFO dans les réseaux multiservices. Cette étude permet aussi de constater des différences entre les différentes variantes de WFQ pour pouvoir ainsi déceler les points forts et faibles de chacune.
Nous allons donc analyser les performances de chaque politique selon des critères de QdS tels que le délai et la gigue. Nous aurons les objectifs de simulation suivants :

- élaborer les graphes des statistiques utiles pour chaque politique,
- analyser les graphes de chacune de ces politiques,
- déceler les différences probables et donner un avis sur ces résultats.

## 7.1 Description du simulateur

Pour mettre en œuvre nos propres mécanismes de QdS et réaliser nos simulations nous avons utilisé le logiciel de modélisation de réseaux OPNET MODELER [46]. Ce logiciel est un environnement de conception et de simulation offrant une gamme d'outils permettant de créer et simuler différents types de réseaux de communications qu'ils soient basés sur les technologies ATM, IP ou autres.
Certains mécanismes standards sont déjà définis et intégrés par défaut et donc facilement configurables par l'utilisateur. On peut citer par exemple, les mécanismes de routage (RIP, OSPF, MPLS...), de réservation (RSVP), de transport (TCP, UDP), de signalisation (SIP), de gestion de la congestion (RED, WRED, WFQ).
Certaines politiques d'ordonnancement telles, par exemple FIFO, PQ ou CBWFQ, sont déjà implantées. Le WFQ simple ne l'est par contre pas, et est donc supplée par le CBWFQ.
De plus, OPNET laisse le champ ouvert aux utilisateurs pour modifier ces mécanismes puisqu'il laisse un accès libre au code source. L'utilisateur peut aussi créer ses propres mécanismes à travers la création de nouveaux modèles de processus, modèles de nœud, ou autres. Il peut aussi s'assurer de leur bonne integration avec la plateforme existante en tirant profit des outils de correction (*debugging tools*) offerts.

Notons que le langage utilisé est le *PROTO C* ressemblant au langage C mais disposant de librairies propres à OPNET.

## 7.2 Réseau simulé

Le modèle du réseau utilisé pour générer notre trafic est illustré à travers la figure 7.1. Ce réseau est caractérisé par des nœuds au cœur du réseau et d'autres à son extrémité appelés *nœuds d'accès*. On suppose que tous ces nœuds ont des tampons de tailles illimitées. Les nœuds au cœur sont *router 8*, *router 9*, *router 10* et *router 11*. Les nœuds d'accès sont *router 0*, *router 1*, *router 2*, *router 3*, *router 4*, *router 5*, *router 6* et *router 7*. Chacun de ces derniers est connecté à un commutateur qui est finalement connecté au différents serveurs et terminaux du sous réseau correspondant, comme le montre notamment la figure 7.2.
Dans notre réseau, chaque SR est configuré de sorte qu'il envoie du trafic vers un seul autre SR. Ainsi, on définit les couples (SR, SR) suivants : (SR0, SR4), (SR4, SR0), (SR1, SR7), (SR7, SR1), (SR2, SR6), (SR6, SR2), (SR3, SR5), (SR5, SR3). Ainsi tout le trafic généré par SR1 a pour destination SR7.
Le but est d'envoyer du trafic entre SR1 et SR7 (généré par SR1 et SR7) passant par les routeurs 8, 9, 10 et 11. Dans chacun de ces routeurs, ce trafic sera en concurrence lors du partage des ressources avec d'autres trafics échangés entre le reste des couples (SR, SR) définis ci-dessus. Ainsi par exemple, la disponibilité des ressources dans chacun des routeurs tout au long du chemin entre SR1 et SR7, ne dépendra pas seulement du trafic entre SR1 et SR7, mais aussi du trafic concurrent.

Les liens cablés entre des stations et un switch, mais aussi entre un commutateur et un nœud d'accès, sont des liens 1000BaseX de type full-duplex. Ce lien représente une connexion Ethernet opérant à 1 Gbps. Quant aux liens entre les nœuds, ils opérent à 2.048 Mbps. Ce débit est très faible par rapport à ce qu'offre les infrastructures ac-

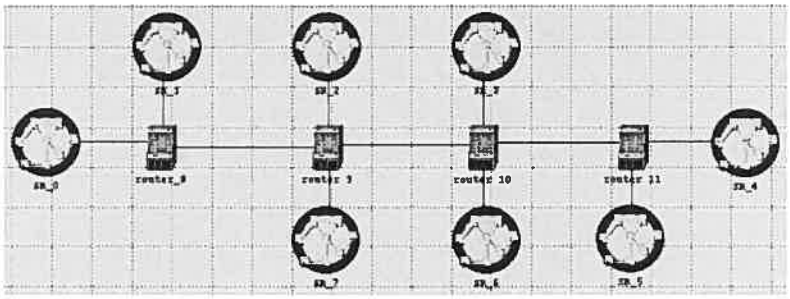


FIG. 7.1 – Topologie du réseau simulé

tuelles. On aurait pu choisir des liens de capacité supérieure pouvant supporter un débit de 44.736 Mbps ou même de 594.43 Mbps tels qu'offre OPNET MODELER dans la palette de choix des liens disponibles. Cependant utiliser au maximum la capacité de ce genre de liens en simulant des périodes de congestion signifie un nombre beaucoup plus grand de sessions à générer et donc des temps de simulation fastidieusement longs pouvant être comptés en semaines !

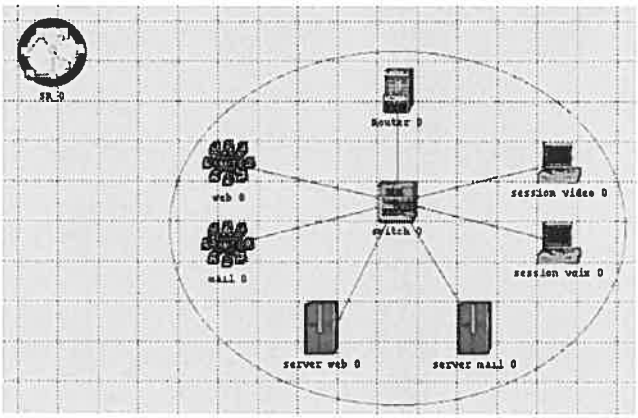



FIG. 7.2 – Composantes matérielles de chaque SR

## 7.3 Modèles de trafic

### 7.3.1 Caractéristiques des applications

Les réseaux multiservices ont la caractéristique de transporter à la fois des applications NTR et d'autres TR notamment la voix et la vidéo. Ce genre d'applications a la particularité de générer du gros trafic ce qui crée des problèmes de stockage, de sur utilisation, et ainsi des difficultés au niveau de la gestion des réseaux de communications. On a donc tendance à coder et à compresser les données émises par chacune de ces applications lors de l'émission avant leur encapsulation sous forme de paquets IP. Il existe plusieurs types de codeurs ayant des caractéristiques et des qualités différentes que ce soit pour la voix ou la vidéo. Les codeurs se basant sur des méthodes sans perte (Huffman, LZW...) ont généralement un taux de compression très faible par rapport à d'autres méthodes qu'on verra par la suite.

L'UIT [45] est une organisation internationale dans laquelle les compagnies du secteur privé, mais aussi des institutions publiques et gouvernementales, coordonnent leurs efforts pour la définition de normes touchant les réseaux et les services de télécommunication. L'UIT est donc une source d'informations sur la normalisation, la réglementation et la technologie des télécommunications. La famille des codeurs standards pour l'audio qu'on a pris en compte et présentés dans le tableau 7.1 ont été déposés par l'UIT [45].

| Standard | Débit(kbps) | Taille d'une trame(ms) | Année |
|---|---|---|---|
| G.711 - audio codec | 64 | 0.125 | 1972 |
| G.726, G.727 - audio codec | 16, 24, 32, 40 | 0.125 | 1990 |
| G.722 - audio codec | 48, 56, 64 | 0.125 | 1988 |
| G.728 - speech codec | 16 | 0.625 | 1992/1994 |
| G.729 - speech codec | 8 | 10 | 1995 |
| G.723 - speech codec | 5.3 et 6.4 | 30 | 1995 |

TAB. 7.1 – Standards audios recommandés par l'UIT

Quant à MPEG [66], c'est le nom d'une famille de standards utilisés pour le codage de l'information de type audio-visuel (musique, vidéo) dans un format digital compressé. L'avantage majeur de MPEG par rapport aux autres formats de codage est que les fichiers MPEG sont plus petits tout en gardant la même qualité de l'image. En effet, MPEG utilise des techniques de compression assez sophistiquées. La famille des standards MPEG inclut MPEG-1, MPEG-2 and MPEG-4, formellement connus comme ISO/IEC-11172, ISO/IEC-13818 et ISO/IEC-14496 et a été réalisée conjointement sous l'égide de l'IEC et de l'OSI.

H.261 [45] est un autre type de codage vidéo faisant partie d'une famille de standards crées par l'UIT. Cependant le dernier standard, le H263 [45] va remplacer le H.261 dans plusieurs applications. Les principaux codeurs standards utilisés actuellement pour la vidéo sont présentés dans le tableau 7.2.

| Standard | Taux de compression | Application | Année |
|---|---|---|---|
| H.261 | p × 64 Kbps | Vidéo conférence | 1988-1990 |
| H.263 | ¡64 Kbps à plus de 2 Mbps | Vidéo conférence | 1995/1996 |
| MPEG-1 | de 1.0 à 1.5 Mbps | Stockage d'images animées sur des systèmes de sauvegarde(CD-ROM) | 1992/1993 |
| MPEG-2 | de 4 à 20 Mbps dépendant de la qualité | télévision digitale | 1994/1995 |
| MPEG-4 | 9.6 à 1024 Kbps | Applications multimédia (vidéo conférence) | 1999 |

TAB. 7.2 – Standards vidéos recommandés par l'UIT

Pour la voix, les trames de données audio sont compressés puis transmises au réseau à débit constant. La vidéo est la succession d'une série d'images de taille fixée envoyées à des intervalles de temps réguliers. Le système de compression compresse chaque image qui sera par la suite transmise sur le réseau. Les codeurs permettent ainsi de réduire la bande passante utilisée en réduisant l'espace mémoire (en bits) utilisé par chaque image. Cependant, en conservant une qualité donnée, chaque image est plus ou moins compressible. Ainsi, il existe deux types de codeurs : les codeurs à taux fixe remplissent simplement un espace mémoire qui est ensuite écoulé à un débit constant sur le réseau.

Il s'agit alors, étant donné le débit choisi, de compresser chaque image de manière à ce que la mémoire ne soit jamais vide et ne déborde jamais. Cela implique éventuellement une réduction de la qualité des images qui ne peuvent être assez bien compressées. En revanche, les codeurs à taux variable choisissent une qualité d'image donnée, et étant donnée cette qualité, ils compressent chaque image au maximum possible et il en résulte des images compressées de tailles variables et par conséquent un trafic à débit variable. Les codeurs à taux variables sont appelés à être plus utilisés dans le futur puisqu'ils ne dégradent pas la qualité de l'image.

Pour les futurs terminaux multimédias, les codeurs principalement utilisés seront le codeur H.263 ou MPEG4 à taux variable pour la vidéo et le codeur G.729 pour la voix. Ainsi, les applications voix générées dans notre simulateur utilisent le codeur G.729 disponible dans OPNET, et les applications de vidéo à la demande sont générées avec un débit variable par la donnée d'une taille d'image (en bits) qui suit une loi non constante (exponentielle) avec un temps d'interarrivée constant des images. Toutes les applications types mises en œuvre sont des instances des modèles d'applications proposés par OPNET Technologies Inc. Les paramètres nécessaires à la configuration des instances d'applications sont définis dans les manuels de configuration fournis par OPNET Technologies Inc [46].

Dans notre étude nous avons choisi de définir quatre types d'applications. deux applications de type temps réel (voix et vidéo) et deux autres de type non temps réel (mail et navigation web). Chacune de ces applications appartient à une classe de service donnée et est caractérisée par un certain nombre de paramètres. Les paramètres de configuration

| Classe | Application | Description | Hiérarchie |
|---|---|---|---|
| Tâche de fond | Mail | Envoi et réception de courriels | 0 |
| Standard | Web | Navigation sur le web | 1 |
| Multimédia Interactive | Vidéo | Vidéo streaming | 2 |
| Conversationnelle | Voix | VoIP | 3 |

TAB. 7.3 – Les quatre types d'applications

de chacune des quatre applications ont été définis comme mentionné dans la table 7.4. On a choisi de s'inspirer du paramètrage déjà prédéfini dans OPNET MODELER pour

chaque type d'application. Ce paramètrage s'effectue aussi bien pour le flot montant que pour le flot descendant.

| Application | Paramètre | Valeur sens montant | Valeur sens descendant |
|---|---|---|---|
| Mail | Temps d'interarrivées(en s) | exponentiel(400) | exponentiel(400) |
| | Nombre d'emails | constant(5) | constant(5) |
| | Taille d'un email (en bytes) | constant(1000) | constant(1000) |
| Web | Temps d'interarrivées(en s) | exponentiel(60) | |
| | Nombre d'objets | constant(6) | |
| | Taille de l'objet (en byte) | constant(1000) | |
| Vidéo | Temps d'interarrivées des images (en s) | constant(0.1) | constant(0.1) |
| | Tailles de l'image (en bytes) | exponentiel(828) | exponentiel(828) |
| Voix | Longueur du silence (en s) | exponentiel(0.65) | exponentiel(0.65) |
| | Longueur de la conversation (en s) | exponentiel(0.352) | exponentiel(0.352) |
| | Type de codeur | G.729 | G.729 |
| | Nombre de trames par paquets | 1 | 1 |

TAB. 7.4 – Paramètres de configuration des applications

### 7.3.2 Profils de trafic

Dans notre réseau, chaque terminal est une source de trafic. Ce trafic est représenté globalement par une ou plusieurs sessions (applications de différents types). Mais, pour décrire ce trafic et en particulier le comportement d'une source avec OPNET, on est amené à créer un profil pour exécuter une ou plusieurs sessions. Le profil lui même va se répéter un certain nombre de fois. On pourrait choisir préalablement le nombre de répétitions du profil, sinon on pourrait choisir de l'exécuter jusqu'à la fin de la simulation. La figure 7.3 montre pour un terminal donné, la succession en série du même profil qui a été défini statiquement. La description d'une telle succession de profils est possible grâce aux paramètres cités ci-dessous.

- Début du profil : représente le moment où le premier profil commence. Il suit une distribution statistique.

- Durée du profil : la durée d'un seul profil. Il suit une distribution statistique.
- Temps d'inter-répétition : représente le temps qui sépare deux profils consécutifs. Il suit aussi une distribution statistique.
- Nombre de répétition : nombre maximum de profils possibles avant la fin de la simulation.

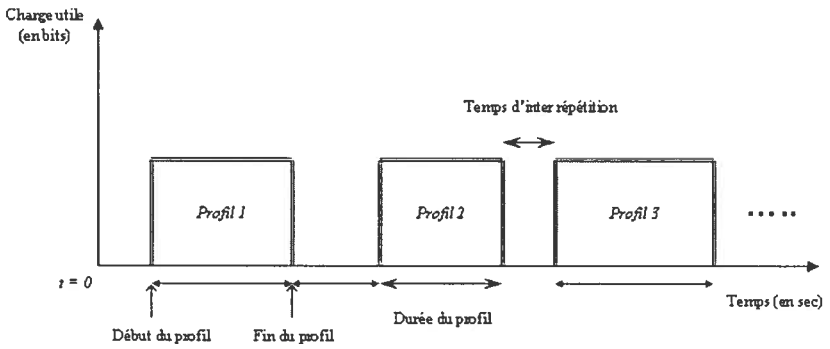



FIG. 7.3 – Succession temporelle des profils de trafic

La description de la succession des sessions dans un même profil, comme le montre la figure 7.4, est elle même possible grâce aux paramètres suivants.

- Décalage : représente le décalage en secondes entre le début du profil et le début de la première session de ce profil. Il suit une distribution statistique.
- Durée : temps maximum que peut durer une session avant qu'elle se termine. Ce paramètre suit une distribution statistique.
- Temps d'inter-répétition des sessions : suit une distribution statistique. Les sessions sont générées en ordre. Cependant, les sessions peuvent être générées de manière concurrente. Ainsi, pour calculer le début de la session suivante, OPNET va ajouter le temps d'inter-répétition au temps correspondant au début de la session précédente et non pas au temps de fin de la session précédente comme on le fait pour le calcul de sessions en série. Cette explication est aussi valide pour le cas d'inter-répétition des profils présentés ci-haut.
- Nombre de répétition : nombre maximum de sessions possibles avant la fin du profil.

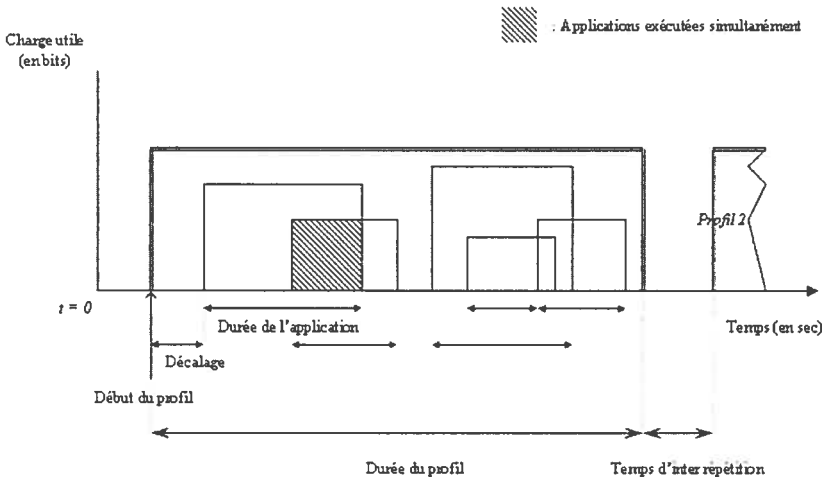



FIG. 7.4 – Succession temporelle des sessions à l'intérieur d'un profil

Les profils utilisés pour la génération du trafic sont présentés dans les tables suivantes.

| Profils | Début Profil (s) | Durée Profil (s) | Inter-répétition Profil (s) | Nombre Répétition | Répétition |
|---|---|---|---|---|---|
| Mail | exponential(30) | fin simulation | constant(300) | 0 | Série |
| Web | constant(0) | fin simulation | constant(300) | 0 | Série |
| Vidéo | constant(20) | constant(300) | constant(300) | 0 | Série |
| Vidéo 2 | constant(110) | constant(70) | constant(300) | 0 | Série |
| Vidéo 3 | constant(240) | constant(50) | constant(300) | 0 | Série |
| Voix | constant(20) | constant(300) | constant(300) | 0 | Série |
| Voix 2 | constant(20) | constant(110) | constant(300) | 0 | Série |
| Voix 3 | constant(160) | constant(100) | constant(300) | 0 | Série |

TAB. 7.5 – Paramètres de configuration des profils dans le simulateur

| Applications | Décalage (s) | Durée (s) | Inter-répétition (s) | Nombre répétition | Répétition |
|---|---|---|---|---|---|
| Mail | exponentiel(60) | Fin de tâche | constant(500) | Illimité | Série |
| Web | exponentiel(60) | Fin de tâche | constant(300) | Illimité | Série |
| Vidéo | exponentiel(5) | constant(250) | exponentiel(300) | constant(0) | Série |
| Vidéo 2 | constant(5) | constant(60) | exponentiel(300) | constant(0) | Série |
| Vidéo 3 | constant(5) | constant(40) | exponentiel(300) | constant(0) | Série |
| Voix | exponentiel(5) | constant(250) | exponentiel(300) | constant(0) | Série |
| Voix 2 | exponentiel(5) | constant(100) | exponentiel(300) | constant(0) | Série |
| Voix 3 | exponentiel(5) | constant(90) | exponentiel(300) | constant(0) | Série |

TAB. 7.6 – Paramètres de configuration des sessions dans les profils

## 7.4 Scénarios pour la génération du trafic

Comme déjà mentionné dans la section 7.2, on a défini les couples (SR, SR) suivants : (SR0, SR4), (SR4, SR0), (SR1, SR7), (SR7, SR1), (SR2, SR6), (SR6, SR2), (SR3, SR5), (SR5, SR3). Ainsi tout le trafic généré par SR1 a pour destination SR7.

La durée de chaque simulation étant de 330 secondes.

La période d'exécution du profil Voix 2 correspond à la *Phase A*, celle de Vidéo 2 à la *Phase B*,celle de Voix 3 à la *Phase C* et celle de Vidéo 3 à la *Phase D*. Une phase s'écoule entre le moment du début et de fin du profil (voir table 7.5). il se peut que deux phases se chevauchent pendant une courte période, comme les différentes phases présentées dans la table 7.5. Les tables suivantes décrivent le type ainsi que le nombre de profils utilisés par scénario.

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Voix 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Vidéo | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 |

TAB. 7.7 – Profils par SR pour le scénario A

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 |
| Voix 3 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 |
| Vidéo | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 |

TAB. 7.8 – Profils par SR pour le scénario B

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Voix 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Vidéo | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 |

TAB. 7.9 – Profils par SR pour le scénario C

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Voix 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Vidéo | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 |

TAB. 7.10 – Profils par SR pour le scénario D

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| Voix 3 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| Vidéo | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 |

TAB. 7.11 – Profils par SR pour le scénario E

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 |
| Voix 3 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 |
| Vidéo | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 |

TAB. 7.12 – Profils par SR pour le scénario F

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Voix 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Vidéo | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 110 | 110 | 110 | 110 | 110 | 110 | 110 | 110 |

TAB. 7.13 – Profils par SR pour le scénario G

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 |
| Voix 3 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 3 |
| Vidéo | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 37 | 37 | 37 | 37 | 37 | 37 | 37 | 37 |
| Web | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

TAB. 7.14 – Profils par SR pour le scénario H

| Profil\SR | SR 0 | SR 1 | SR 2 | SR 3 | SR 4 | SR 5 | SR 6 | SR 7 |
|---|---|---|---|---|---|---|---|---|
| Voix | 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Voix 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Voix 3 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Vidéo | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Vidéo 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Vidéo 3 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mail | 130 | 130 | 130 | 130 | 130 | 130 | 130 | 130 |
| Web | 330 | 330 | 330 | 330 | 330 | 330 | 330 | 330 |

TAB. 7.15 – Profils par SR pour le scénario I

Pour chaque scénario, dépendemment du nombre et du type des profils générés, on aura une demande différente en matière de bande passante. Dans la partie gauche de la table 7.16, on définit pour chaque scénario et pour chaque classe, le pourcentage de bande passante à utiliser à travers tout le réseau à certains moments. Ces moments correspondent aux moments où les profils *Voix 2* (*Phase A*) et *Voix 3* (*Phase D*) sont actifs, alors que les profils *Vidéo 2* et *Vidéo 3* ne le sont pas. Pour le cas où les profils *Voix 2* et *Voix 3* sont inactifs et les profils *Vidéo 2* (*Phase B*) et *Vidéo 3* (*Phase D*) sont actifs, le pourcentage de bande passante est présentée dans la partie droite de la table 7.16.

| Scénario\Classe | Voix | Vidéo | Web | Mail | % BP | Voix | Vidéo | Web | Mail | % BP |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 43 | 37 | 13 | 7 | 100 | 31 | 43 | 13 | 7 | 94 |
| B | 46 | 40 | 13 | 7 | 106 | 31 | 43 | 13 | 7 | 94 |
| C | 49 | 43 | 13 | 7 | 112 | 31 | 43 | 13 | 7 | 94 |
| D | 37 | 40 | 13 | 7 | 97 | 31 | 43 | 13 | 7 | 94 |
| E | 40 | 43 | 13 | 7 | 103 | 31 | 43 | 13 | 7 | 94 |
| F | 46 | 31 | 13 | 7 | 97 | 31 | 43 | 13 | 7 | 94 |
| G | 49 | 34 | 13 | 7 | 103 | 31 | 43 | 13 | 7 | 94 |
| H | 46 | 40 | 12 | 2 | 100 | 31 | 43 | 13 | 7 | 94 |
| I | 34 | 31 | 39 | 7 | 111 | 31 | 43 | 13 | 7 | 94 |

TAB. 7.16 – Trafic généré par les classes de chaque scénario

Dans le cas d'un CBWFQ, les poids des différentes classes sont les mêmes pour tous les scénarios et sont présentés dans la table 7.17.

Dans un PQ les classes ont le même ordre d'importance que celles dans CBWFQ. La

| Voix | Vidéo | Web | Mail | Total |
|---|---|---|---|---|
| 43 | 37 | 13 | 7 | 100 |

TAB. 7.17 – Poids des classes dans un CBWFQ

Voix étant la classe à plus haute priorité.

## 7.5 Evaluation de la précision des bornes déterministes

Dans ce qui suit on va comparer les bornes déterministes, calculées en se basant sur les travaux présentés dans la section 5.2.2, avec les bornes mesurées par simulation sous les mêmes conditions. On suppose qu'on calcule le délai pour la classe voix dont le poids $w$, dans un contexte CBWFQ, représente une fraction de 0.4 de la somme totale des poids. Les délais calculés sont effectués entre une source voix rattachée au routeur 0 et une destination reliée au routeur 4.

### 7.5.1 Bornes de Charny et Le Boudec

On suppose que dans le *Seau percé*, $\sigma$ est égale à 1 unité (section 3.3). D'où, $b_{max} = MTU$ (taille d'un paquet). Les paramètres de réseau, déjà définis dans la section 5.2.2 et liés aux formules 5.2 et 5.3 sont détaillés comme suit :

- $w = 40/100$.
- $C_{in} = 1Gbps$ ou $10Mbps$.
- $C_{out} = 1Gbps$ ou $10Mbps$.
- $g = w \times C_{out}$.
- $g_j = w \times C_{out}$.
- $h = 6$.
- $b_{max}$ = taille d'un paquet = 48 Octets = 384 Bits(20 Octets Données + 20 Octets entête IP + 8 Octets entête UDP).
- $r_{min} = 19.2Kbps$. Pour la voix, on est présence d'un codec G.729 ($8Kbps$). En ajoutant les différentes entêtes (IP et UDP), cette source enverrait à un débit de $19.2Kbps$.
- $\beta_j$ = taille d'un paquet = 384 Bits.

Avec ses paramètres, le facteur d'utilisation maximum toléré et calculé à partir de l'inégalité 5.2 est : $\alpha \leq 0.2941$. Donc, on ne peut pas effectuer un calcul du délai pour

une utilisation de plus de 29.41% de la bande passante disponible $g_j$. En se basant sur le théorème 5.3, on présente dans les tables 7.19 et 7.18 les délais calculés avec différents facteurs d'utilisation $\alpha_i$ tels que $\alpha_i < \alpha$. On choisit les valeurs $\alpha_i$ suivantes : $\alpha_i \in 0.2, 0.0923, 0.0462, 0.0153$. La différence entre les deux tables est la valeur maximale qu'un nœud peut avoir comme débit d'entrée. Pour la table 7.19, on a utilisé un SR où le lien vers le nœud d'entrée au réseau possède une capacité de 1Gbps. Pour la table 7.18, on a utilisé un SR où le lien vers le nœud d'entrée a une capacité de 10Mbps.

| | $\alpha = 0.2$ | $\alpha = 0.0923$ | $\alpha = 0.0462$ | $\alpha = 0.0153$ |
|---|---|---|---|---|
| Délai calculés (ms) | 46.658 | 10.493 | 4.617 | 1.76 |
| Délai réels (ms) | 0.00006761 | 0.0000525 | 0.0000007 | 0 |

TAB. 7.18 – Délais calculés (Charny) vs Délais mesurés ($C_{in} = 10Mbps$)

| | $\alpha = 0.2$ | $\alpha = 0.0923$ | $\alpha = 0.0462$ | $\alpha = 0.0153$ |
|---|---|---|---|---|
| Délai calculés (ms) | 45.016 | 9.692 | 3.953 | 1.168 |
| Délai réels (ms) | 0.00006207 | 0.00004915 | 0.00000024 | 0 |

TAB. 7.19 – Délais calculés (Charny) vs Délais mesurés ($C_{in} = 1Gbps$)

### 7.5.2 Bornes de Jiang

De même que dans la section 7.5.1, on suppose que dans le *Seau percé*, $\sigma$ est égale à 1 unité. D'où, $\sigma = L_{max}$ (taille maximale d'un paquet). Les paramètres de réseau, déjà définis dans la section 5.2.2 et liés à la formule 5.9 sont détaillés comme suit :

- $w = 40/100$.
- $H = 6$.
- $C_h = 1Gbps$ ou $10Mbps$, représente le débit du lien $h$.
- $L_{max} = 384Bits$.
- $R_h = w \times C_h$.
- $E_h = \frac{L_{max}}{C_h}$, pour le cas d'un ordonnanceur de type WFQ.
- $P_h$ est égale à $1 \times C_h$.

- $L_h$ est égale à $1 \times L_{max} = 384Bits$.
- $I_h$ est égale à 1.
- $\beta = \frac{3 \times L_{max}}{C_h}$.

Avec ses paramètres, le facteur d'utilisation maximum toléré et calculé est : $\alpha \leq 0.2941$. En se basant sur le théorème 5.9, on présente dans les tables 7.21 et 7.20 les délais calculés avec différents facteurs d'utilisation $\alpha_i$ tels que $\alpha_i < \alpha$, avec $\alpha_i \in 0.2, 0.0923, 0.0462, 0.0153$. Pour la table 7.21, on a utilisé un SR où le lien vers le nœud d'entrée au réseau possède une capacité de 1Gbps. Pour la table 7.20, on a utilisé un SR où le lien vers le nœud d'entrée a une capacité de 10Mbps.

| | $\alpha = 0.2$ | $\alpha = 0.0923$ | $\alpha = 0.0462$ | $\alpha = 0.0153$ |
|---|---|---|---|---|
| Délai calculés (ms) | 3.6862 | 1.8419 | 1.5423 | 1.3969 |
| Délai réels (ms) | 0.00006761 | 0.0000525 | 0.0000007 | 0 |

TAB. 7.20 – Délais calculés (Jiang) vs Délais mesurés ($P_h = 10Mbps$)

| | $\alpha = 0.2$ | $\alpha = 0.0923$ | $\alpha = 0.0462$ | $\alpha = 0.0153$ |
|---|---|---|---|---|
| Délai calculés (ms) | 1.9756 | 0.9368 | 0.7681 | 0.6861 |
| Délai réels (ms) | 0.00006207 | 0.00004915 | 0.00000024 | 0 |

TAB. 7.21 – Délais calculés (Jiang) vs Délais mesurés ($P_h = 1Gbps$)

### 7.5.3 Conclusions

Les bornes déterministes présentées par Jiang sont meilleures que celles déterminées dans le cadre des travaux de Charny et Le Boudec. Elles sont nettement meilleures quand il s'agit d'un plus grand facteur d'utilisation.

Reste que ces bornes sont peu précises. Ceci s'ajoute à une forte restriction quant au degré d'utilisation de la bande passante disponible. Dans le cas d'un service garanti, un contrôle d'admission basé sur ces bornes, n'est pas efficace et mène à une grande sous utilisation du réseau.

## 7.6 Comparaison des politiques FIFO, PQ, CBWFQ

Les politiques d'ordonnancement étudiées sont FIFO, PQ, et CBWFQ. Le choix s'est porté sur FIFO puisque c'est la politique actuellement utilisée dans l'Internet. On a aussi choisi PQ car elle reste une politique intéréssante vu qu'elle favorise les classes de plus hautes priorités alors que les classes de basses priorités risquent la famine. PQ a aussi été utilisé dans [21] pour le comparer à une politique adaptative. Pour étudier les performances de chacune de ces politiques et réussir à les comparer, on va tout d'abord construire sous OPNET plusieurs instances de réseaux identiques. La seule différence entre ces instances concerne la politique d'ordonnancement implantée dans tous les nœuds (routeurs) du réseau. Evidemment, dans une même instance de réseau, on va choisir un même processus d'ordonnancement pour tous les types de nœuds qu'ils soient des nœuds à l'extémité ou des nœuds au cœur du réseau. On parlera alors d'un réseau tout FIFO, tout PQ, ou tout CBWFQ. Notre étude va se baser sur la comparaison de ces politiques par rapport aux principaux critères de QdS connus.

- ✓ Le niveau de respect du délai de bout en bout des applications temps réel. Ce délai étant principalement calculé à partir du temps passé par les paquets d'une application donnée dans les files de chaque nœud qu'elle traverse.
- ✓ La valeur de la gigue réalisée par ces différentes politiques. La meilleure entre ces dernières est celle qui réalise la plus petite valeur de la gigue.
- ✓ Temps de réponse d'une page web.
- ✓ Temps de téléchargement d'un message électronique (courriel).

Nous avons essayé d'évaluer la politique WFQ simple (par flots), mais cela était impossible à réaliser. Déjà et à travers les différentes lectures effectuées [1] [42], on dit bien que WFQ simple n'est pas implantable dans les réseaux à haut débit du fait du grand nombre de flots à gérer en plus de la complexité du calcul des valeurs des estampilles $F_i^k$ (voir section 4.1.3). Malgré cela, nous avons essayé d'évaluer WFQ mais nous avons rencontré les contraintes suivantes ce qui nous a obligé à renoncer aux tests sur les performances du WFQ simple.

- OPNET implante un WFQ où la configuration des files d'attentes dans les nœuds se fait manuellement et statiquement préalablement au lancement des simulations. Ceci est utile pour implanter un CBWFQ. Cependant il est possible d'imaginer la création au préalable d'une file pour chaque session allant être établie durant la simulation. Ceci est possible grâce au paramètres dont on dispose pour la caractérisation d'une session dans le panneau de configuration de WFQ. Ces paramètres sont : TOS, Protocole, adresse source, adresse destination, port source, port destination, interface d'entrée.
- OPNET se limite aux 100 premières files d'attentes créées par un nœud donné. OPNET ignore le reste des files. On ne peut donc pas avoir plus que 100 sessions à gérer par nœud, ce qui ne répond pas à nos objectifs pour évaluer un WFQ simple dans des cas de congestion comme dans celui du scénario C. En fait nous générons plus de 100 sessions dans tous les scénarios.
- OPNET n'implante pas un vrai WFQ. C'est un VC qui est implémenté à la place. Bon nombre de simulateurs de réseaux tel que *NS2* font de la sorte et implantent VC au lieu de WFQ. Ceci découle du fait que le calcul des $F_i^k$ d'un WFQ est très complexe alors que le calcul des temps virtuels de VC est beaucoup plus simple à effectuer (section 4.1.3). Aussi, VC se comporte identiquement à un WFQ dans le cas où toutes les files sont pleines (section 4.1.3, section 9.5.2 de [1] et [31]). Les concepteurs de ce genre de simulateurs se permettent alors d'échanger un vrai WFQ par un simple VC ce qui limite le champs de tests à des cas où toutes les files sont pleines. Nous ferons en sorte de nous intéresser aux cas où toutes les files sont pleines pour éviter de tomber dans de fausses interprétations.

Les scénarios choisis pour l'évaluation prennent en considération différents cas de figure où la quantité de trafic influe sur les résultats et les performances de chaque politique. Les délais de bout en bout exprimés dans les figures sont des moyennes et ne représentent pas une mesure des délais entre une paire origine destination particulière. Dans le scénario A, on a pris en compte du trafic utilisant à peu près 100% des ressources disponibles (table 7.16). On en utilise un peu plus au début, vu que le profil *Mail* utilise un peu plus que 7% des ressources, mais se stabilise vers le milieu de la simulation. Ceci

explique le fait que dès le début, les délais pour FIFO dépassent les 200 ms et atteignent 1.2 secondes vers la fin du profil *Voix 2* (*Phase A*) (figure 7.5). 200 ms est la valeur à partir de laquelle on considère qu'on dégrade la QdS des applications TR. Dès lors, les performances de FIFO s'améliorent jusqu'au début de *Vidéo 3* (*Phase D*) vu que les ressources sont utilisées à peu près à 94%(< 100%). Le fait que *Voix 3* commence à peu près à 160sec et que *Vidéo 3* finit à peu près à 180sec fait que sur le graphique, et durant ce petit intervalle, les délais pour FIFO augmentent de nouveau.

Cependant, PQ et CBWFQ réalisent comme prévu des délais en dessous de 200 ms tout au long de la simulation. Les délais de bout en bout réalisé par un réseau *tout CBWFQ* sont légèrement supérieurs à cause des plus longs délais de traitement que cause un ordonnanceur VC par rapport à un ordonnanceur PQ (figure 7.7). Cela est dû, d'une part, à la simplicité de gestion des files de PQ, et de l'autre, aux calculs plus complexes effectués par VC. On aurait sûrement eu de plus grands délais de bout en bout si OPNET implantait un vrai WFQ.

Pour la vidéo, FIFO dépasse de même la limite des 200 ms durant toute période où les ressources sont utilisées à un peu plus de 100%. PQ réalise toujours d'excellents résultats quant aux délais de bout en bout. Quant à CBWFQ, il se comporte mieux que FIFO. Les deux remontées dans la courbe FIFO de la figure 7.6 sont dues au fait que *Voix 2* et *Vidéo 2*, ou *Voix 3* et *Vidéo 3*, sont actifs au même moment et donc représentent des demandes en ressources supérieures aux ressources disponibles. Ceci représente des petites périodes d'à peu près 20 sec. Durant ces périodes les files d'attentes des classes vidéo vont augmenter de taille. Juste après ces périodes les performances de CBWFQ commencent à s'améliorer de nouveau. Reste que CBWFQ doit vider petit à petit les files déjà remplies, ce qui explique la lenteur dans la diminution des délais de bout en bout après les deux pics (130 sec et 180 sec).

PQ traite la classe voix comme la classe de plus haute priorité et la classe vidéo avec une priorité moindre. Dans nos scénarios, puisque la somme des trafics voix et vidéo est toujours inférieure à la totalité de la bande passante disponible en sortie, alors on aura toujours les mêmes résultats pour des réseaux *tout PQ*. C'est ainsi qu'on omettra les résultats de PQ pour certaines figures dans le reste de cette section.

Pour la gigue, les trois politiques réalisent de bons résultats. Les variations du délai de bout en bout sont minimes (voir figure 7.8).

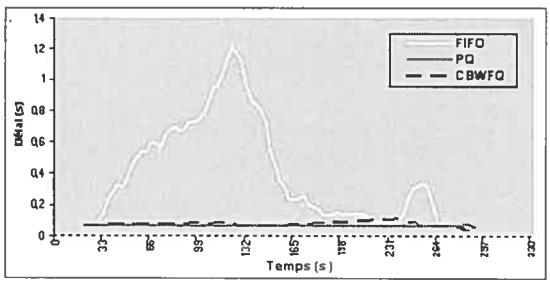



FIG. 7.5 – Délai bout en bout voix : Scénario A

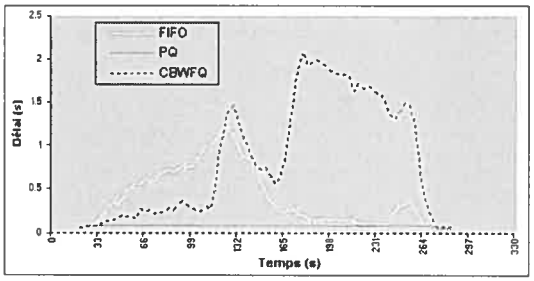



FIG. 7.6 – Délai bout en bout vidéo : Scénario A

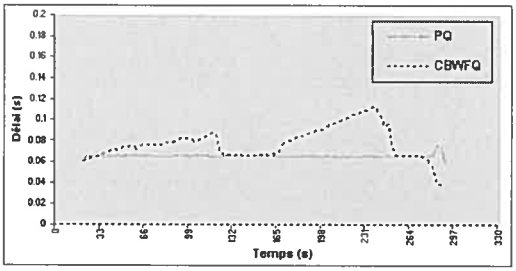



FIG. 7.7 – Délai bout en bout voix : Scénario A (sans FIFO)

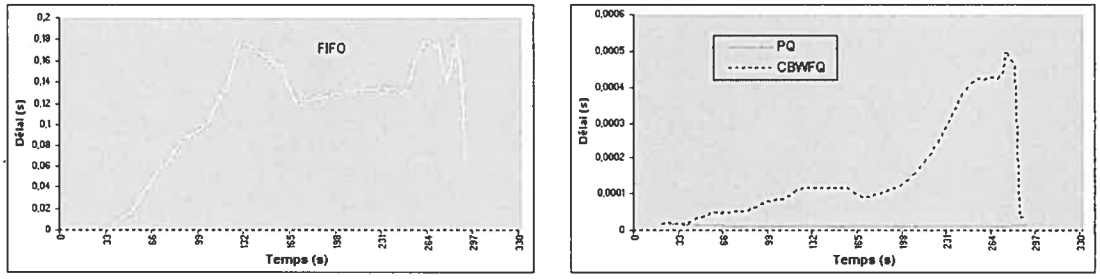



FIG. 7.8 – Gigue voix : Scénario A

Dans le scénario B, on a choisi le trafic de sorte qu'il soit supérieur aux ressources disponibles (table 7.16). FIFO réalise de mauvais résultats partout. Quant à CBWFQ il est aussi mauvais vu que les proportions de bande passante allouées aux classes voix et vidéo sont inférieures à la demande. Cela résulte en une grande accumulation de trafic dans les files correspondantes. Durant la *Phase A*, les délais de bout en bout augmentent linéairement. Cela résulte en une large gigue aussi bien pour la voix que pour la vidéo, contrairement à la petite gigue réalisée par PQ (voir figure 7.11).
Sachant que dans ce scénario, les trafics générés par les classes *Mail* et *Web* ne dépassent pas les proportions de bande passante allouées par le CBWFQ, le mauvais comportement des classes *Voix* et *Vidéo* ne peut être absorbé par les classes NTR. On voit que les délais de bout en bout dépassent ceux réalisés avec FIFO. Ceci est dû au fait que dans le cas de FIFO (ordonnancement agrégé), ce mauvais comportement est aussi absorbé par le trafic des classes *Mail* et *Web* ce qui rend les délais de bout en bout pour *Voix* et *Vidéo* un peu moindres. Ceci vient du fait que le degré d'agrégation est inversement proportionnel au degré de protection (voir section 4.2.2).

Dans le scénario C, de même que dans B, on a choisi le trafic de sorte qu'il soit très supérieur aux ressources disponibles. Cependant, ici, on se retrouve dans le cas de plus forte congestion si on comparait le trafic généré par B et celui généré par C. Ainsi les mauvais résultats de FIFO et CBWFQ sont plus accentués que ceux trouvés avec B. On peut constater cela à travers les figures 7.9 pour B et 7.12 pour C. Cependant les mêmes conclusions faites sur B sont applicables avec C. La gigue est aussi plus large.

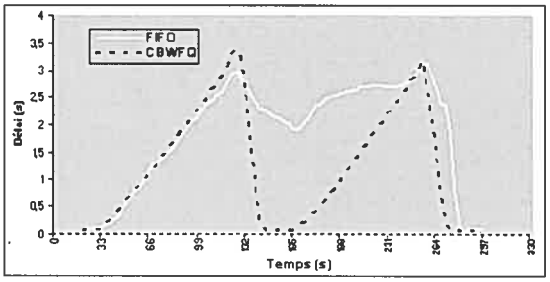



FIG. 7.9 – Délai bout en bout voix : Scénario B

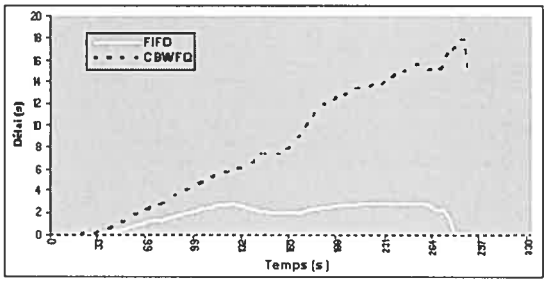
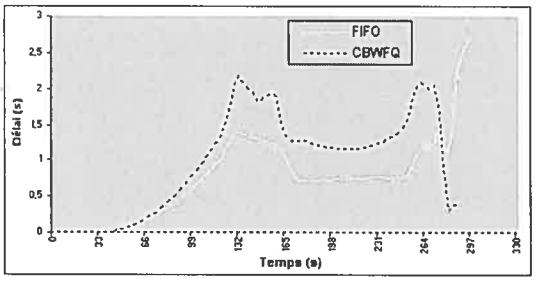



FIG. 7.10 – Délai bout en bout vidéo : Scénario B



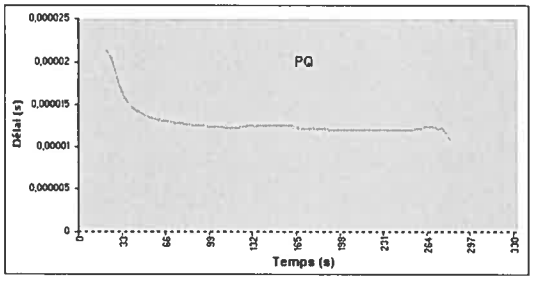




FIG. 7.11 – Gigue voix : Scénario B

La particularité de CBWFQ par rapport à FIFO est qu'il différencie entre les classes (agrégation moins rigide) ce qui fait qu'il alterne entre bon et mauvais dans le cas de la voix (7.12). En effet les délais de bout en bout diminuent dès que le trafic généré est inférieur à aux ressources disponibles (fin des profils *Voix 2* et *Voix 3*). Dans la *phase A*

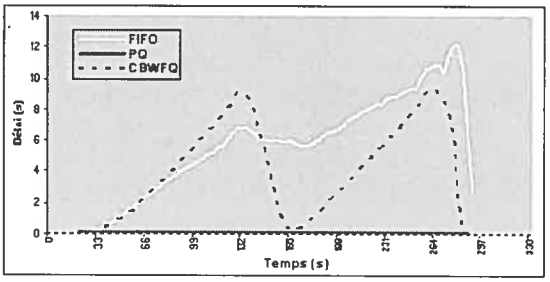



FIG. 7.12 – Délai bout en bout voix : Scénario C

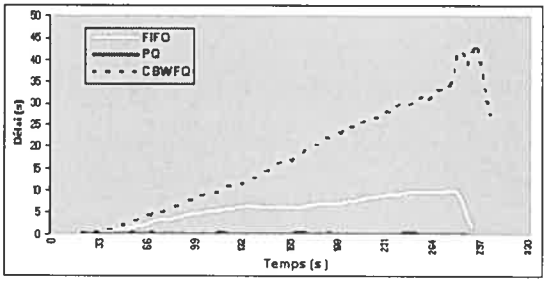



FIG. 7.13 – Délai bout en bout vidéo : Scénario C

du scénario D, on utilise un peu moins que la totalité des ressources. Pour la voix, FIFO réalise d'assez bons résultats au début, mais dépasse malgré tout la barre des 200 ms

avant même le début de *Vidéo 2*. Quant à CBWFQ, et vu que le trafic voix (37%) est inférieur à la quantité de bande passante allouée ($w = 43$), les résultats sont excellents et se situent en dessous de 75 ms (figure 7.14). Pour la vidéo, FIFO se comporte de même que pour la voix à cause d'absence de différentiation. Pour sa part, CBWFQ à du mal à toujours rester sous la barre des 200 ms durant toute la *phase A*. On note, en effet, quelques dépassements si on se réfère à la figure 7.15. Malgé que le trafic vidéo généré est supérieur à la bande passante allouée, CBWFQ arrive généralement à respecter les délais et assure souvent une assez bonne qualité pour la vidéo. Ceci est dû au fait que, durant la *phase A* on n'utilise pas la totalité des ressources.

Avec CBWFQ et au début de la *Phase C*, les files vidéo sont remplies, à cause du trafic vidéo élevé généré durant la *phase B*. Cependant ces délais vont diminuer peu à peu pour atteindre des valeurs inférieures à 200 ms.

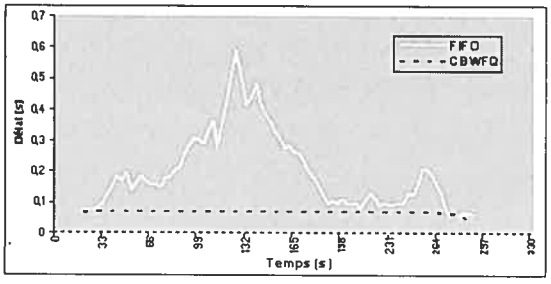



FIG. 7.14 – Délai bout en bout voix : Scénario D

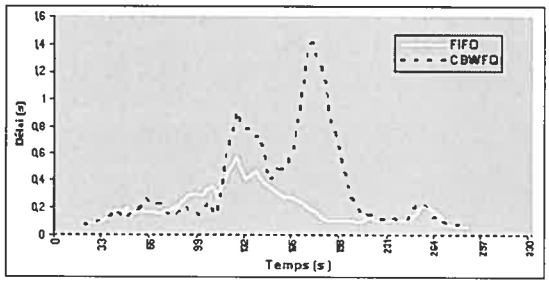



FIG. 7.15 – Délai bout en bout vidéo : Scénario D

Pour le scénario E, on génère du trafic qui sollicite à peu près 103% des ressources (sur utilisation). FIFO n'arrive pas à satisfaire les besoins des classes temps réels en terme de délais et de gigue. Dans le cas de CBWFQ, le fait que trafic voix généré soit inférieur à la bande passante lui étant allouée ($w = 43$), fait que les délais pour la voix sont excellents (voir figure 7.18). La gigue est aussi très petite et est comparable à celle de PQ (figure 7.19).

La demande du trafic vidéo est, quant à elle, très supérieure à la bande passante allouée ($w = 37$). Vu qu'on est en période de congestion avec une demande de 103%, une partie du surplus du trafic vidéo est accumulée dans les files de sorte que les délais de bout en bout pour la vidéo augmentent rapidement (figure 7.17).

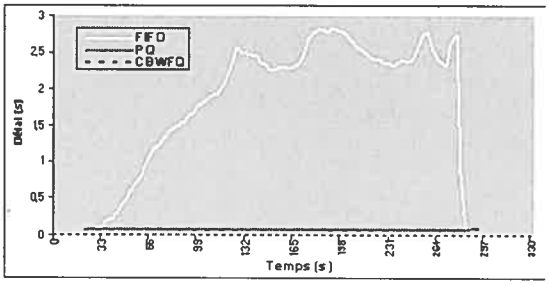



FIG. 7.16 – Délai bout en bout voix : Scénario E

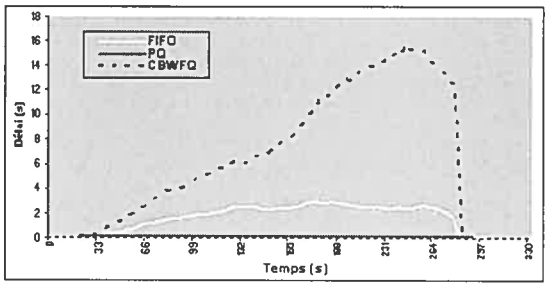



FIG. 7.17 – Délai bout en bout vidéo : Scénario E

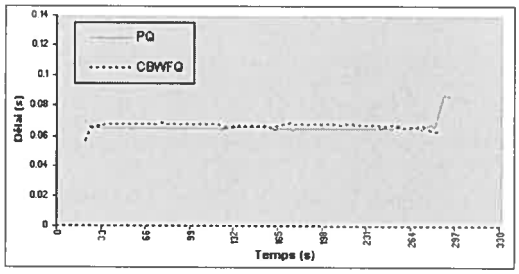
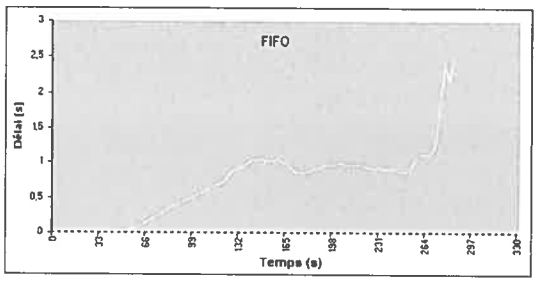



FIG. 7.18 – Délai bout en bout voix : Scénario E (sans FIFO)



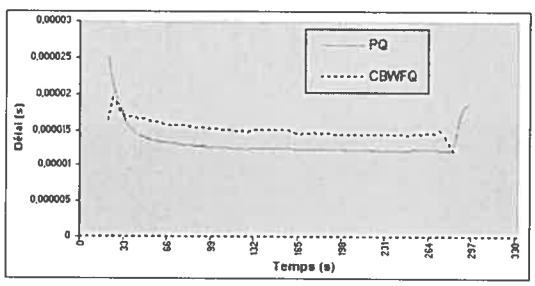




FIG. 7.19 – Gigue voix : Scénario E

Dans le scénario F, on génère du trafic qui sollicite à peu près 97% des ressources disponibles. Dans la figure 7.21, on voit que FIFO maintient un délai pour la voix le plus souvent supérieur à 2 sec. Pour la vidéo, là où l'utilisation ne dépasse pas les 31% des ressources, FIFO arrive à réaliser difficilement de bons résultats et maintient assez souvent des délais inférieurs à 200 ms.

CBWFQ, réalise de très bons résultats avec délais comparables à ceux de PQ. Pour la vidéo, les délais sont autour de 100 ms dans les périodes où le trafic vidéo généré est inférieur à la bande passante allouée ($w = 37$). Pour la voix, malgré que la bande passante allouée est inférieure à la demande, la proportion non utilisée par la vidéo sert à la voix pour réaliser de très bons délais (voir figure 7.20 et 7.22). CBWFQ est équitable en matière de bande passante à cause notamment du critère *max-min-fair-share*.

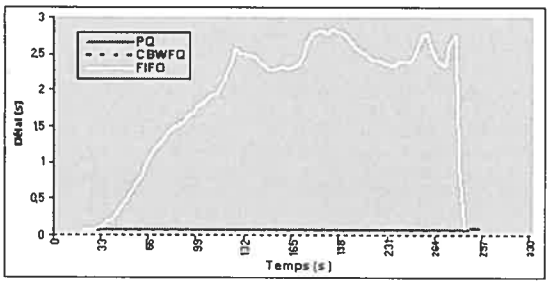



FIG. 7.20 – Délai bout en bout voix : Scénario F

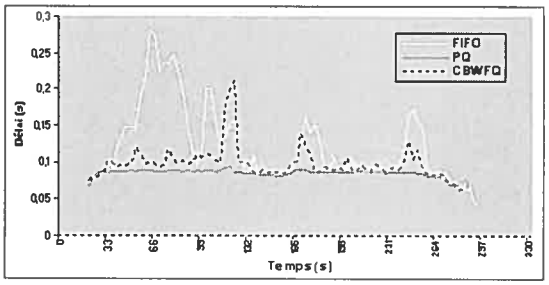



FIG. 7.21 – Délai bout en bout vidéo : Scénario F

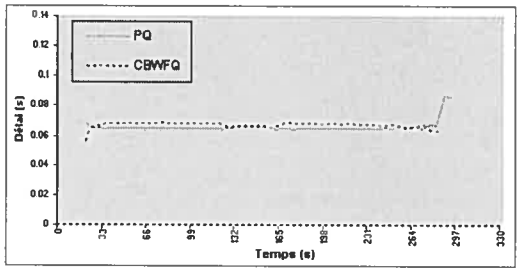



FIG. 7.22 – Délai bout en bout voix : Scénario F (sans FIFO)

Pour le scénario G, comme dans le scénario E, on génère du trafic qui sollicite à peu près 103% des ressources. Dans le cas de CBWFQ, la demande du trafic voix (49%) est très supérieure à la bande passante allouée ($w = 43$). Vu qu'on est en période de congestion avec une demande de 103%, une partie du surplus du trafic voix est accumulée dans les files de sorte que les délais de bout en bout augmentent au dessus de la barre de 200 ms (figure 7.23).

Pour la vidéo, le fait que le trafic généré soit inférieur à la bande passante lui étant allouée ($w = 37$), fait que les délais pour la vidéo sont très satisfaisants durant les phases A et C (figure 7.24).

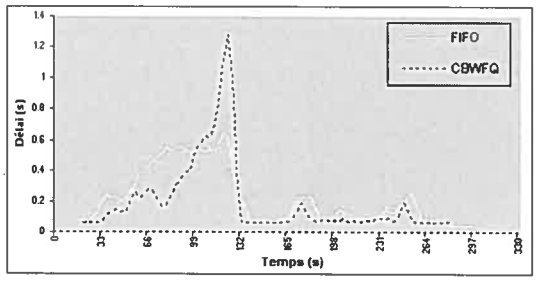



FIG. 7.23 – Délai bout en bout voix : Scénario G

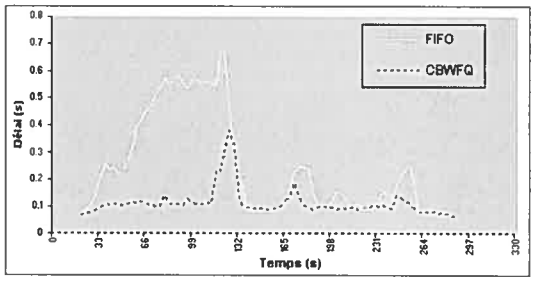



FIG. 7.24 – Délai bout en bout vidéo : Scénario G

Dans le scénario I, on a choisi le trafic de sorte qu'il soit très supérieur aux ressources disponibles (table 7.16). Sauf que, dans le cadre d'un CBWFQ, les trafics voix et vidéo générés sont très inférieurs à la bande passante disponible. CBWFQ, en donnant de gros poids aux classes TR, arrive à satisfaire les exigences des trafics TR. Les délais de ceux-ci sont comparables à ceux de PQ (voir figures 7.27 et 7.28).

FIFO réalise évidemment de mauvais résultats comparables à ceux réalisés dans le scénario A (figures 7.25 et 7.26).

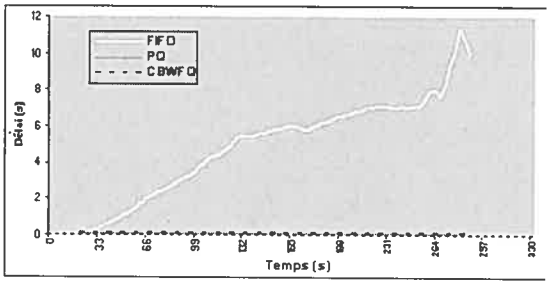



FIG. 7.25 – Délai bout en bout voix : Scénario I

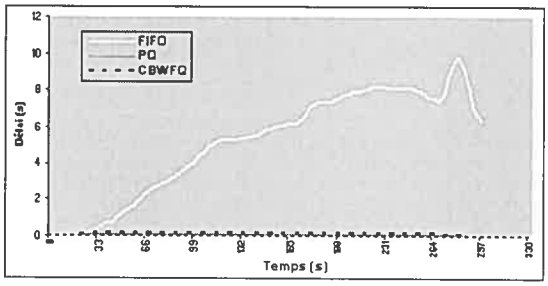



FIG. 7.26 – Délai bout en bout vidéo : Scénario I

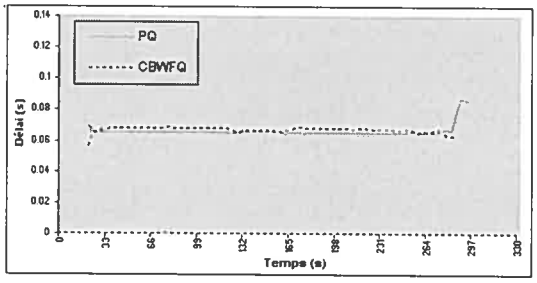



FIG. 7.27 – Délai bout en bout voix : Scénario I (sans FIFO)

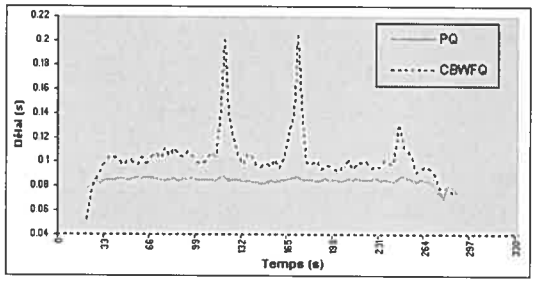



FIG. 7.28 – Délai bout en bout vidéo : Scénario I (sans FIFO)

A travers ces interprétations, on voit que, par rapport aux applications TR, FIFO se comporte très mal, CBWFQ arrive à satisfaire dans pas mal de cas alors que PQ réalise des résultats irréprochables. Les grandes variations des délais mesurés avec FIFO montrent que la gigue est plus large.

Les résultats de PQ pour les classes prioritaires TR est néanmoins source de bien des tracas pour le cas des classes de priorité inférieures ou NTR. On s'attend sûrement à avoir de mauvaises surprises quant au traitement de ces classes. En effet, si on regarde les temps de réponse (DRT) lors du téléchargement d'un message électronique (*Mail* étant la classe de plus basse priorité) dans le cas du scénario A, on voit qu'ils sont plus grands que ceux réalisés avec CBWFQ ou FIFO (figure 7.29). Pour le scénario C, ces écarts sont encore plus accentués (figure 7.30).

Ces dernières performances de PQ font apparaître le risque de suppression des paquets

au niveau des nœuds où les tampons sont de tailles limitées, voir même le risque de famine pour les classes NTR, spécialement la classe *Mail*.

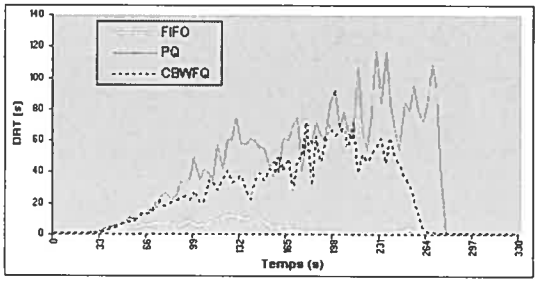



FIG. 7.29 – DRT : Scénario A

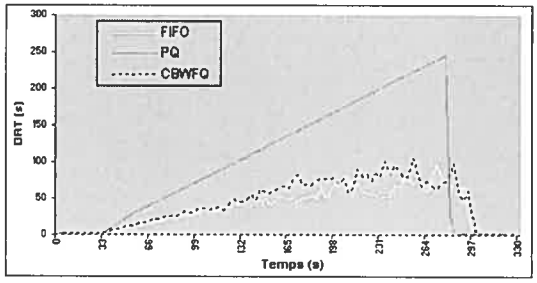



FIG. 7.30 – DRT : Scénario C

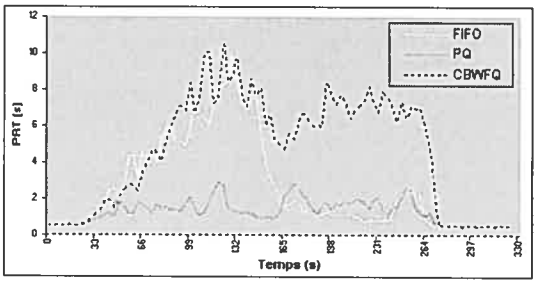



FIG. 7.31 – PRT : Scénario A

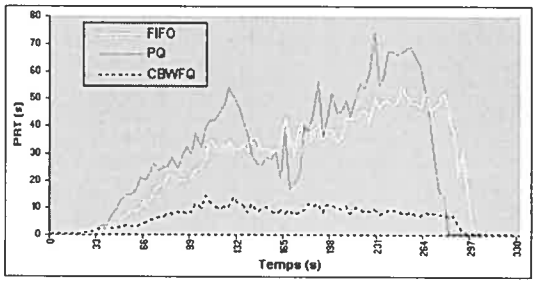



FIG. 7.32 – PRT : Scénario C

Etant donné que la classe *Web* est de priorité plus élevée que *Mail* (mais de moindre priorité que *Vidéo*), ses temps de réponse (PRT) pour le téléchargement d'une page web restent comparables à ceux de CBWFQ dans le cas du scénario A (figure 7.31). Cependant ses performances se dégradent plus vite que les autres politiques dans le cas du scénario C où la congestion est plus accentuée (*Phase A* et *Phase C* de la figure 7.32). Cela est dû au fait que la valeur de $w^{add}(Q)$ (section 6.5.2) peut au maximum être de 10 $((w^{Mail}+w^{Web})-(w^{thres}_{Mail}+w^{thres}_{Web}) = (7+13)-(5+5))$(voir table 7.16), alors que les classes TR ont besoin, intuitivement, d'un $w^{add}(Q)$ de valeur 12 représentant la demande excédentaire du trafic TR par rapport à la bande passante disponible (voir table 7.16).

Les résultats obtenus à travers ces différents scénarios viennent confirmer les points annoncés dans la section 6.1.2 pour le cas de la politique CBWFQ. Aussi, cette politique

reste la politique la plus équitable. Les résultats obtenus pour FIFO et PQ confirment leurs statuts de politiques non équitables.

CBWFQ se comporte assez bien même si les demandes des classes TR sont excessives, mais que la congestion n'est pas forte. C'est le cas de la *Phase A* du scénario A.

Sous CBWFQ et durant certaines phases de certains scénarios (*Phase A* des scénarios A, D ou G), les délais de bout en bout pour les classes TR dépassent de peu la barre de 200 ms. Il est, ainsi, intéressant de disposer d'une façon permettant de favoriser ce type de classes.

Dans la section suivante on présente les résultats de la politique adaptative du chapitre 6. Celle-ci fait la différentiation entre ces quatre classes et qui essaye de donner, sans abuser et quand il le faut, plus de priorité aux classes TR.

## 7.7 Évaluation du CBWFQ adaptatif

Pour les besoins des simulations et de la procédure *décision-ajustement*, nous avons choisi $d_{max} = \varepsilon = 150$ ms et $Q_{min} = 1000$ bits.

Pour les classes TR et dans la plupart des scénarios, la politique adaptative améliore significativement les délais de bout en bout obtenus par CBWFQ. Ceci est valable pour tous les scénarios sauf pour le cas du scénario C (figures 7.33, 7.34, 7.35, 7.36, 7.37, 7.38, 7.39, 7.40, 7.41 et 7.42).

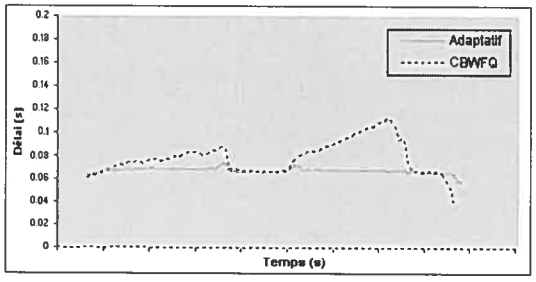



FIG. 7.33 – Délai bout en bout voix : Scénario A

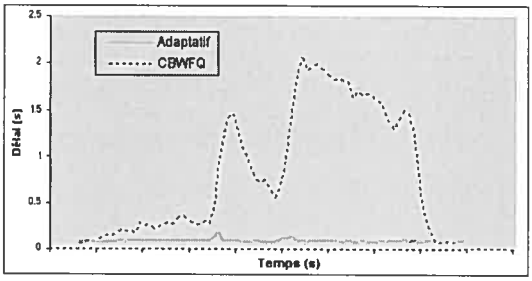



FIG. 7.34 – Délai bout en bout vidéo : Scénario A

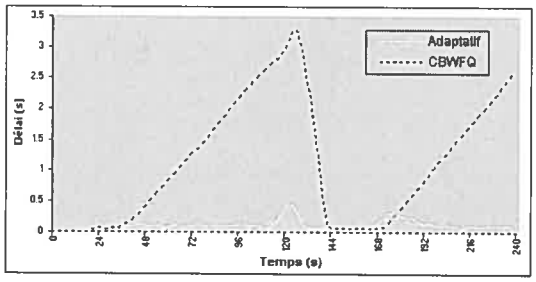



FIG. 7.35 – Délai bout en bout voix : Scénario B

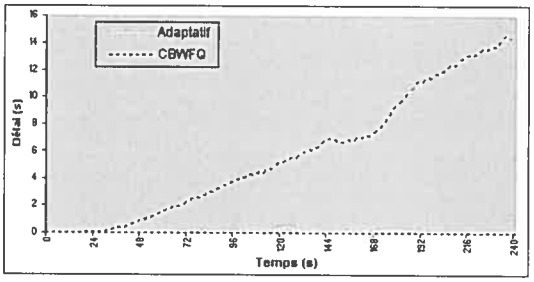



FIG. 7.36 – Délai bout en bout vidéo : Scénario B

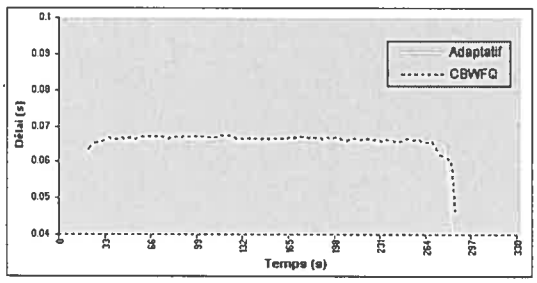



FIG. 7.37 – Délai bout en bout voix : Scénario D

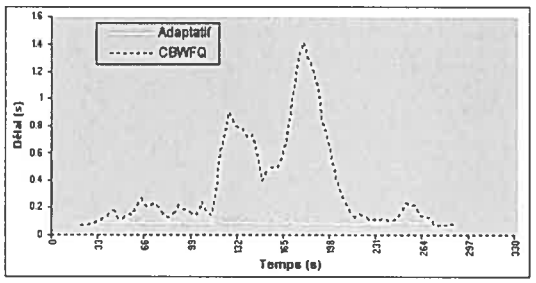



FIG. 7.38 – Délai bout en bout vidéo : Scénario D

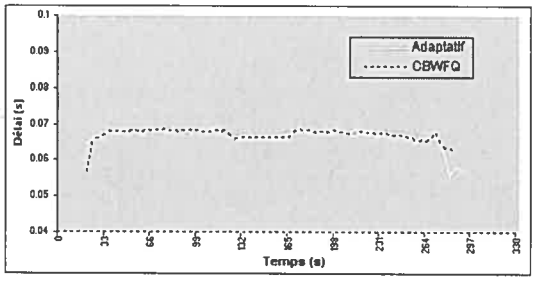



FIG. 7.39 – Délai bout en bout voix : Scénario E

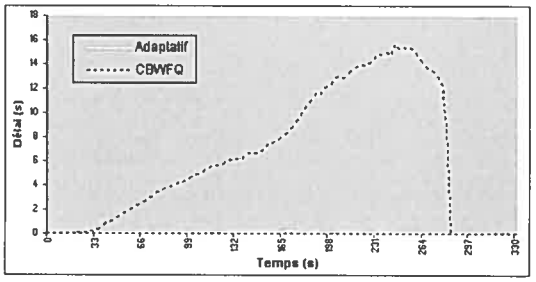



FIG. 7.40 – Délai bout en bout vidéo : Scénario E

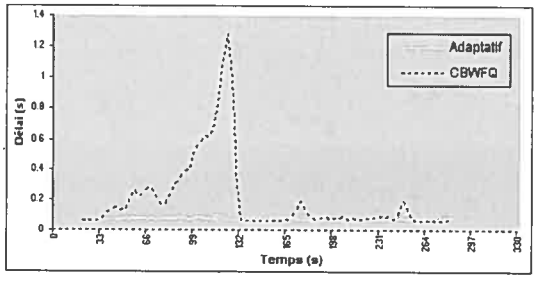



FIG. 7.41 – Délai bout en bout voix : Scénario G

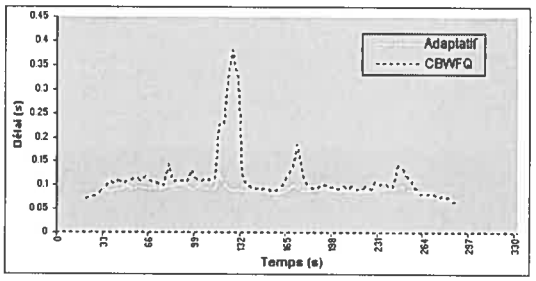



FIG. 7.42 – Délai bout en bout vidéo : Scénario G

En effet, les proportions en excès demandées de la part des classes TR dépassent les proportions que peut leurs offrir les classes NTR. L'algorithme adaptatif peut soustraire au maximum un poids de 10 $((w^{Mail} + w^{Web}) - (w^{thres}_{Mail} + w^{thres}_{Web}) = (7 + 13) - (5 + 5))$ de $w^{NTR}$. Mais cela ne suffit pas à la demande supérieure des classes TR du scénario C. Ainsi, d'après les figures 7.43 et 7.44, la voix va rester insatisfaite.

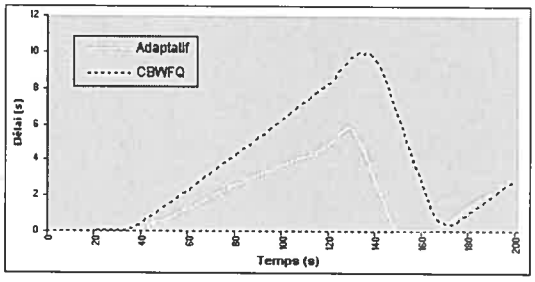



FIG. 7.43 – Délai de bout en bout voix : Scénario C

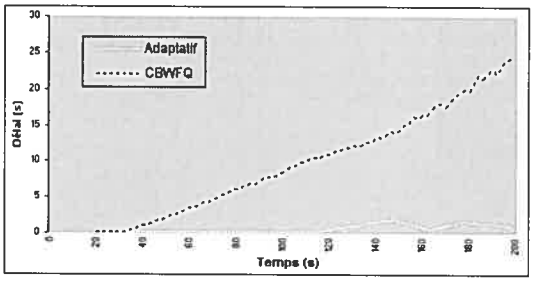



FIG. 7.44 – Délai de bout en bout vidéo : Scénario C

On pourrait se demander pourquoi c'est la voix qui se trouve insatisfaite dans ce cas (scénario C), et non pas la vidéo ?

Au niveau de chaque nœud du réseau, les trafics TR envoient toujours plus que la proportion allouée en sortie. On devrait rapidement s'attendre à recourir à un ajustement quasi simultané des poids des classes TR. Cependant, pour ce scénario, les tailles des tampons des files TR sont très importantes durant $\varepsilon$ (150 ms). Dans le cas où les deux classes TR ont dépassé les délais, il suffit que trois paquets vidéo arrivent avant un paquet voix pour qu'ils raflent la majorité des 10 unités de poids que peuvent offrir les classes NTR. Ainsi, la fréquence de réception des paquets influe grandement sur la procédure de redistribution des poids.

Cela est dû au fait que notre procédure d'ajustement satisfait toujours, à la mesure du possible, celui qui demande de hausser son poids sans se préoccuper de combien de fois il l'a déjà haussé.

Dans le cas où $PR(Q)$ (section 6.5.2) est supérieur à 3, la procédure d'ajustement des poids haussera de 3 unités le poids de la file vidéo, à chaque paquet vidéo reçu à un

temps $t$. La valeur 3 représente la demande excédentaire du trafic vidéo par rapport à la bande passante disponible durant la période $[t-\varepsilon, t]$.

Pour le cas du scénario C, nous avons changé la valeur de $w^{thres}$ de 5 à 3. Ceci est pour permettre à la demande des classes TR d'être satisfaites. En effet, dans ce cas, l'algorithme adaptatif peut soustraire au maximum un poids de 14 $((w^{Mail}+w^{Web})-(w^{thres}_{Mail}+w^{thres}_{Web})=(7+13)-(5+5))$ de $w^{NTR}$ qui va, effectivement, suffire à satisfaire toutes les demandes des classes TR (voir figures 7.45 et 7.46).

Le fait qu'on essaye, au mieux, de satisfaire les classes TR, fait que les délais vont

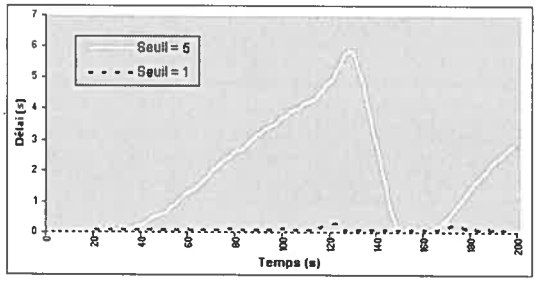



FIG. 7.45 – Délai bout en bout voix : Scénario C ($w^{thres}=3$)

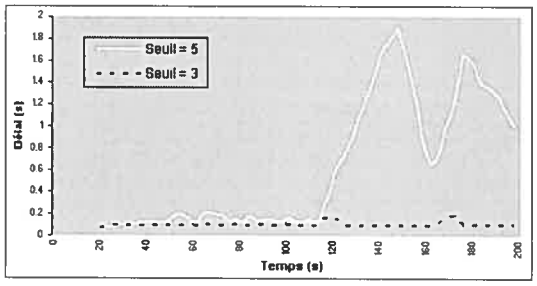



FIG. 7.46 – Délai bout en bout vidéo : Scénario C ($w^{thres}=3$)

diminuer et leurs variations aussi. En effet, les gigues réalisées avec un CBWFQ sont nettement améliorées. Dans la figure 7.47, on montre les résultats réalisés à ce niveau par CBWFQ et par notre politique adaptative.

Pour le cas du scénario C, cette gigue est aussi améliorée mais reste très large.

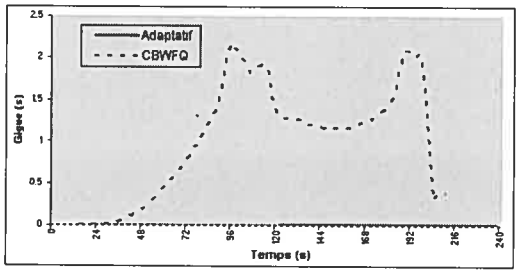



FIG. 7.47 – Gigue voix : Scénario B

**Conclusions**

Dans le cas de certaines phases de certains scénarios (*Phase A* des scénarios A, B, D, E, ou G), notre politique adaptative respecte les délais de bout en bout pour les classes TR qui ne dépassent pas 200 ms. Ceci se fait sans trop dégrader la qualité des classes NTR (PRT pour la classe Web et DRT pour la classe Mail). Les variations sur les délais sont aussi bien maîtrisées et les gigues sont, par conséquent, petites.

Pour le cas de fortes congestions (plus accentuées que celle du scénarios C), notre politique ne garantit plus les délais pour les classes TR, en plus de dégrader considérablement la qualité des classes NTR.

Dans la section suivante on explique une variante de notre politique adaptative, qui au contraire et dépendemment du choix de $\varepsilon$, favorisera la voix.

## 7.8 Variantes du CBWFQ adaptatif

La variante présentée dans cette section a pour but de montrer l'influence du choix de $\varepsilon$ sur la procédure de redistribution des poids, s'il y a forte congestion. Au lieu d'un $\varepsilon = 150$ ms, on le choisira égal à 0.0001 ms pour simuler un ajustement des poids de moindre ampleur. En effet, en regardant les résultats du scénario E (voir figure 7.48), on remarque que les délais de bout en bout pour les deux classes TR ont été aussi améliorés. Cependant, on remarque que la réaction se fait plus lente. La période d'ajustement est plus longue que dans le cas précédent. Cela est dû au fait que les proportions de $w^{cut}$ des classes NTR sont plus petites. Dans le cas du scénario C, ce choix mène à plusieurs dépassements du délai permis (200 ms). La réaction de la procédure d'ajustement se fait tardive.

Dépendemment du trafic, le choix de $\varepsilon$ peut, soit signifier une réaction lente, soit une réaction rapide. L'idéal c'est de disposer d'un $\varepsilon$ qui mène à une diminution du délai à la limite inférieure de 200 ms, sans toutefois dépasser cette limite. Reste que pour le cas de

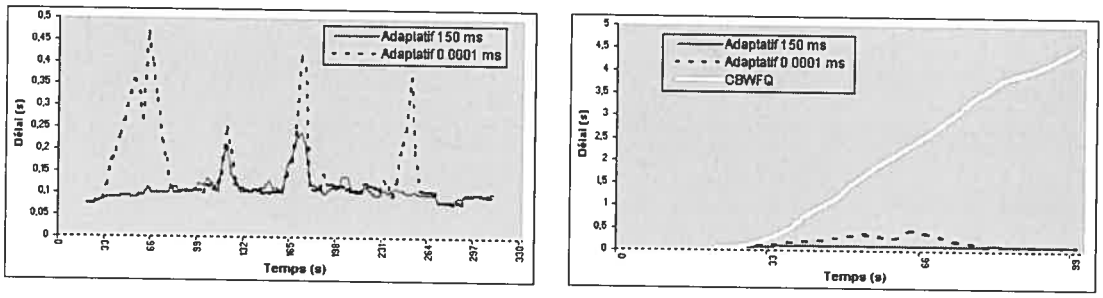



FIG. 7.48 – Délai bout en bout vidéo : Scénario E

fortes congestion, la nouvelle valeur de $\varepsilon$ change la manière avec laquelle la procédure redistribue les poids. Sachant qu'un plus grand nombre de paquets de type voix circule dans le réseau (table 7.4), la chance que donne notre procédure d'ajustement à ce type de paquets est beaucoup plus grande que celle donnée aux paquets vidéo. Dans le cas du dépassement de $d_{max}$ par les deux types de paquets TR, il y aura plus de paquets voix qui viendront reclamer la hausse des poids de leurs files respectives.

Le choix de $\varepsilon$ s'avère crucial quant à la définition du comportement exigé de la part de la procédure d'ajustement des poids.

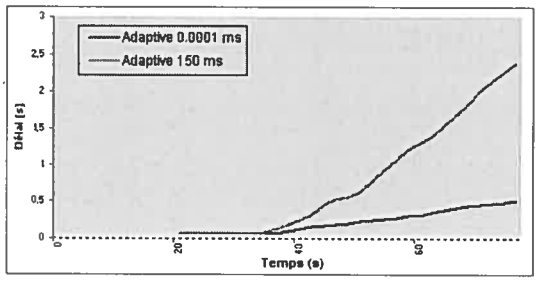



FIG. 7.49 – Délai bout en bout voix : Scénario C

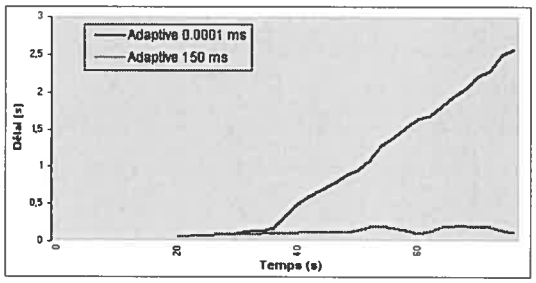



FIG. 7.50 – Délai bout en bout vidéo : Scénario C

# Conclusion et Perspectives

Plusieurs points dans ce mémoire ont été considérés. Notre but était de satisfaire la QdS des applications temps réel essentiellement en terme de délai de bout en bout et de gigue.

Nous avons tout d'abord évalué des bornes déterministes pour les cas de flots agrégés, puis quelques politiques d'ordonnancement existantes. Nous avons finalement proposé une politique d'ordonnancement adaptative.

Lors de l'évaluation des bornes déterministes, nous avons remarqué que les bornes trouvées sont peu précises. Les bornes sont généralement utilisées pour effectuer un contrôle d'admission afin d'accepter ou refuser une session. Réaliser un tel contrôle d'admission, en se basant sur ces bornes, peut mener à une sous utilisation des ressources du réseau et donc à l'inefficacité de sa gestion.

Dans la littérature, les politiques d'ordonnancement les plus intéressantes sont celles qui traitent les paquets avec équité. Cette équité est, dans l'exemple de WFQ, réalisée à travers le critère *max min fair share*. Ces politiques sont dites équitables, mais elles ne considèrent pas certaines informations importantes telles que le degré de respect des bornes sur le délai.

Nous avons conçu une politique basée sur WFQ avec traitement agrégé des flots. Cette politique récupère, en temps réel, les délais subis par les paquets en cours de traitement. Dépendemment du niveau de respect du délai, notre politique va adapter les poids pour, ainsi, satisfaire les classes temps réel. Cette politique a été implémentée et validée avec le simulateur Opnet Modeler et a été comparée avec CBWFQ. Les résultats

obtenus montrent que la politique adaptative améliore nettement, dans le cas de faible ou moyenne congestion, la gigue et les délais de bout en bout des paquets des classes temps réel.
Cependant, les choix de certains paramètres relatifs à la procédure d'ajustement des poids influent grandement sur le comportement de la politique adaptative.

### Travaux futurs

Les bornes déterministes actuelles sont peu précises et il serait intéressant de mieux les explorer. Le fait de disposer de meilleures bornes peut aider à la procédure de prise de décision pour l'acceptation ou le refus d'une session. Nous pensons qu'il est peu probable que de tels travaux aboutissent dans le cadre des flots agrégés.
Il serait intéressant de penser à d'autres variantes de cette politique adaptative. Nous avons remarqué que le choix de certains paramètres peut influer la procédure de distribution des poids. La taille des paquets est aussi un paramètre qu'il faudrait prendre en compte pour l'amélioration de notre politique en terme du degré d'équité lors de la distribution des proportions de poids.
Vu qu'on a évalué notre politique dans le cadre de nœuds ayant des tampons de tailles illimitées, et donc avec aucun risque de perte de paquets (spécialement pour les applications NTR), il serait aussi intéressant d'évaluer la politique adaptative dans un contexte de QdS se rapportant aux applications non temps réel. Ces applications étant sensibles à d'autres critères tel que la perte des paquets.
Finalement, on pourrait concevoir un algorithme de contrôle d'admission dont les décisions d'acceptation ou de rejet de nouvelles connections sont basées sur des informations ayant rapport avec l'état du réseau. Ensuite, on pourrait comparer les performances (nombre de connections acceptées ou rejetées) de cet algorithme dans différents contextes comme, par exemple, le cas où tous les noeuds du réseau implantent un CBWFQ simple ou le cas ou ils implantent tous un CBWFQ adaptatif.

# Bibliographie


[1] S. Keshav, *An Engineering Approach to Computer Networking : ATM Networks, the Internet, and the Telephone Network*, Addison-Wesley, Chapitre 9, 1997.

[2] W. Stallings, *High-Speed Networks and Internets : Performance and Quality of Service*, 2ème édition, Prentice Hall, 2002.

[3] A. K. Parekh, *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*, PhD thesis, Department of Electrical Engineering and Computer Science, MIT, Février 1992.

[4] A. K. Parekh et R. G. Gallager, *A generalized Processor Sharing approach to flow control in integrated services networks - the single node case*, ACM/IEEE Transactions on Networking, 1 :344-357, Juin 1993.

[5] J. C. R. Bennett, H. Zhang, *Why WFQ Is Not Good Enough For Integrated Services Networks*,

[6] J. Y. Le Boudec, *A Proven Delay Bound for a Network with Aggregate Scheduling*, EPFL-DSC Rapport Technique DSC2000/002, http ://ica1www.epfl.ch/PS_files/ds2.pdf, 10 Janvier, 2000.

[7] A. Charny et J. Y. Le Boudec, *Delay Bounds in a Network With Aggregate Scheduling*. Proc. 1st Int Workshop Quality of Future Internet Services (QofIS'2000), pp. 1-13, Berlin, Allemagne, Septembre 2000

[8] A. Charny, *Delay Bounds in a Network With Aggregate Scheduling*, Draft, 29 Février, 2000.

[9] A. Charny, *Delay bounds in a network with aggregate scheduling*, ftp ://ftpeng.cisco.com/ftp/acharny/aggregate_delay.ps, Cisco, 1998.

[10] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney, J. Y. Le Boudec et al., *Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding*. IEEE/ACM Transactions on Networking, Vol. 10, NO. 4, Août 2002.

[11] Y. Jiang, *Delay Bounds for a Network of guaranteed Rate Scheduling Servers with FIFO Aggregation*, Computer Networks, vol 40, No.6, pp 683-694, Décembre, 2002.

[12] Y. Jiang et Q. Yao, *Impact of FIFO Aggregation on Delay Performance of a Differentiated Services Network*, The International Conference on Information Networking ICOIN 2003, Février 12-14, 2003.

[13] R. L. Cruz, *A Calculus for Network Delay, Part II : Network Analysis*, IEEE Transactions on Information Theory, Vol 37, no. 1, Janvier 1991.

[14] R. L. Cruz, *Quality of service guarantees in virtual switched netwroks*, IEEE J. Select. Areas Commun., pp.1048-1056, Août 1995.

[15] P. Goyal et H. M. Vin, *Generalized Guaranteed Rate Scheduling Algorithms : A Framework*, IEEE/ACM Transactions on Networking, 5(4) :561-571, Août 1997.

[16] V. Jacobson, K. Nichols et K. Poduri, *An Expedited Forwarding PHB*, IETF, RFC 2598, Juin 1999.

[17] M. F. de Castro, D. Gaïti, A. M'hamed et M. Oliveira, *Comparing Application Performance on Distinct IP Packet Scheduling Configurations.*

[18] C. C. Li et al, *Proportional Delay Differentiation Service Based on Weighted Fair Queuing*, IEEE, 2000.

[19] J. Gallardo et D. Makrakis, *Dynamic Predictive Weighted Fair Queuing for Differentiated Services*, IEEE, 2001.

[20] N. G Duffield, T. V. Lakshman, D. Stiliadis, *On adaptive bandwidth sharing with rate guarantees*, IEEE INFOCOM, pages 1122-1130, 1998.

[21] M. F. Horng et al, *An Adaptive Approach to Weighted Fair Queue with QoS Enhanced on IP Network*, IEEE, 2001.

[22] K. Zhu, Y. Viniotis et Y. Zhuang, *Guaranteed Rate Scheduling with Adaptive Excess Bandwidth Distribution*, IEEE, 2000.

[23] H. Wang, C Shen et K. G. Shin, *Adaptive-Weighted Packet Scheduling for Premium Service*, Mitsubishi Electric Information Technology Center America, 2001.

[24] N. Natchimuthu et J. Y. Khan, *Adaptive Priority Traffic Management Algorithm for IP-Based Diffserv WAN.*

[25] P. Xu, M. Devetsikiotis et G. Michailidis, *Adaptive Scheduling using Online Measurements for Efficient Delivery of Quality of Service*, Rapport Technique, 29 Mars 2004.

[26] K. Kumaran et al, *Novel Tehniques for the Design and Control of Generalized Processor Sharing Schedulers for Multiple QoS Classes*,

[27] L. Zhang, *VirtualClock : A New Traffic Control Algorithm for Packet Switching Networks*, In Proceeedings of ACM SIGCOMM'90, pages 19-29, Août 1990.

[28] J. Kurose, *Open issues and challenges in providing QoS guarantees in high speed networks*, CCR, vol. 23(1), pages 6-15, Janvier 1993.

[29] A Demers, S Keshav et S Shenker, *Analysis and Simulation of a Fair Queueing Algorithm*, Journal of Internetworking Research Experience, pages 3-26, Octobre 1990.

[30] H. Zhang et S. Keshav, *Comparison of rate-based service disciplines*, Proceeding of ACM SIGCOMM'91, pages 113-122, Septembre 1991.

[31] P. Goyal, S. Lam et H. Vin, *Determining end-to-End Delay Bounds in Heterogeneous Networks*, Proceedings of Fifth Workshop on Network and Operating System Support for Digital Audio and Video, pages 287-298, Avril 1995.

[32] S. Stiliadis et A. Verna, *Latency-Rate Servers : A General model for Analysis of Traffic Scheduling Algorithms*, Proceedings of IEEE INFOCOM'96, 1996.

[33] J. Néto, *Développement et évaluation de politiques d'ordonnancement pour des réseaux multiservices*, Rapport de stage, 5 Octobre, 2002.

[34] S. S. Kanhere et H. Sethu, *Fair, Efficient and Scalable Scheduling Without Per-Flow State*, 20th IEEE International Performance Computing and Communications Conference, Avril 4-6, 2001.

[35] C. Deleuze, *Scheduling.*

[36] S. Floyd, *Notes on CBQ and Guaranteed Service*, 12 Juillet, 1995.

[37] S. Floyd et M. F. Speer, *Experimental Results for Class-Based Queuing*, 11 Novembre, 1998.

[38] V. Sivaraman et al, *End-to-end statistical delay service under GPS and EDF scheduling : A comparison study*, INFOCOM, pages 1113-1122, 2001.

[39] S. Blake, D. Black, M. Carlson, E, Davies, Z. Wang et W. Weiss, *An Architecture for Differentiated Services*, IETF, RFC2475, Décembre 1998.

[40] K. Kilkki, *Differentiated Services For The Internet*, Macmillan Technology Series, 1999.

[41] Cisco Systems, *Class-Based Weighted Fair Queuing*, http ://www.cisco.com/ univercd/cc/td/doc/product/software/ios120/120newft/120t/120t5/cbwfq.htm.

[42] Cisco Systems, *Technologies des interconnexions réseaux*, Cisco Press, 2001.

[43] S. Floyd et V. Jacobson, *Link-sharing and Resource Management Models for Packet Networks*, IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, Août 1995.

[44] *Some Issues and Applications of Packet Marking for Differentiated Services*, http ://www.globecom.net/ietf/draft/draft-blake-diffserv-marking-00.html.

[45] UIT, *Union Internationale des Télécommunications*, http ://www.itu.int.

[46] OPNET, *Optimum Network*, http ://www.opnet.com/support/home1.html.

[47] NS2, *Network Simulator*, http :/www.isi.edu/nsnam/ns/ns-documentation.html

[48] IETF, *Internet Engineering Task Force*, http ://www.ietf.org/rfc.html.

[49] 3GPP, *3rd Generation Partnership Project, Technical Specification Group Services and Systems Aspects ; QoS Concept and Architecture (Release 2004)*, Rapport Technique TS 23.107 v6.1.0, mars 2004, http ://www.3gpp.org.

[50] H. Schulzrinne, S. Casner, R. Frederick et V. Jacobson, *RTP, A Transport Protocol for Real-Time Applications*, IETF, RFC 3550, Juillet 2003.

[51] J. A. Cobb, *Providing Quality of Service Guarantees without Per-Flow State*, IEEE, 2001.

[52] J. A. Cobb, *An In-Depth Look at Flow Aggregation for Efficient Quality of Service*, Proccedings of the IEEE International Conference on Network Protocols, 1999.

[53] J. A. Cobb, *Preserving Quality of Service Guarantees in Spite of Flow Aggregation*, Proceedings of the IEEE International Conference on Network Protocols, 1998.

[54] J. A. Cobb et M. Gouda, *Flow Theory*, IEEE/ACM Transactions on Networking, Octobre 1997.

[55] J. M. Blanquer, B, Özden, *Fair Queuing for Aggregated Multiple Links*, SIGCOMM'01, Août 2001.

[56] K. Dolzer, W. Payer, M. Eberspacher, *A Simulation Study on Traffic Aggregation in Multi-Service Networks.*

[57] T. Worster et A. Doria, *Levels of Aggregation in Flow-Switching Networks*, IEEE, 1997.

[58] R. A. Guerin et V. Pla, *Aggregation and Conformance in Differentiated Service Networks A case Study.*

[59] Information Sciences Institute, *Internet Protocol*, IETF, RFC 791, Septembre 1981.

[60] J. Postel , *User Datagram Protocol*, IETF, RFC 768, Août 1980.

[61] Defense Advanced Research Projects Agency, *Transmission Control Protocol*, IETF, RFC 793, Septembre 1981.

[62] S. Floyd, K. Ramakrishnan, S. Shenker et al, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, IETF, RFC 2309, Avril 1998.

[63] J. Nagle, *Congestion Control in IP/TCP Internetworks*, IETF, RFC 896, 6 Janvier 1984.

[64] W. Stevens, *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, IETF, RFC 2001, Janvier 1997.

[65] A. Tanenbaum, *Réseaux*, 3ème édition, Dunod, 1996.

[66] www.mpeg.org.