Transport à fiabilité partielle d'images compresséessur les réseaux à commutation de paquets

Thomas Holl

Nancy-Université
Université Henri Poincaré

CENTRE DE RECHERCHE EN AUTOMATIQUE DE NANCY

FACULTÉ DES SCIENCES & TECHNIQUES

U.F.R. : Sciences et Techniques Mathématiques, Informatique et Automatique
École Doctorale : IAEM Lorraine
Département de Formation Doctorale : Automatique

CRAN
CNRS UMR 7039

Thèse

présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy 1

en Sciences, spécialité Automatique,
Traitement du Signal et Génie Informatique

par **Thomas Holl**

Transport à fiabilité partielle d'images compressées sur les réseaux à commutation de paquets

Soutenance prévue le 13 juillet 2007

Membres du jury :

Rapporteurs :	Marc Antonini (DR CNRS)	Université de Nice – Sophia Antipolis, I3S
	Philippe Fraisse (McF HdR)	Université de Montpellier II, LIRMM
Examinateurs :	Hervé Guyennet (Prof.)	Université de Franche-Comté, Laboratoire Informatique
	Abdelhamid Mellouk (McF)	Université de Paris XII, LIAA
	Francis Lepage (Prof.)	Université Henri Poincaré Nancy 1, CRAN
	Jean-Marie Moureaux (McF)	Université Henri Poincaré Nancy 1, CRAN
Directeurs de thèse :	Eric Rondeau (Prof.)	Université Henri Poincaré Nancy 1, CRAN
	Vincent Lecuire (McF)	Université Henri Poincaré Nancy 1, CRAN

Université Henri Poincaré, Centre de Recherche en Automatique de Nancy

Faculté des Sciences, B.P. 239, 54506 Vandœuvre-lès-Nancy

Table des matières

Table des figures

Liste des tableaux

Introduction

Depuis la fin des années 1980 et le début de l'expansion de l'Internet à l'échelle mondiale — son développement en Europe a démarré en 1989, coordonné par le RIPE (Réseaux IP Européens) — le « réseau des réseaux » a connu de profondes évolutions. En taille d'abord. Internet représentait 100 000 machines interconnectées en 1989, un million en 1992, 10 millions en 1996, 96 millions en 2000 et 439 millions aujourd'hui (2006). En capacité ensuite, tant au niveau du cœur de réseau que des boucles locales. La capacité d'Internet a augmenté avec le développement des systèmes de transmission haut-débit, en particulier ceux basés sur SDH (Synchronous Digital Hierarchy)[1], sur ADSL (Asymetric Digital Subscriber Line) [2], sur Gigabit-Ethernet et plus récemment sur 10-Gigabit-Ethernet[3]. Dans les usages enfin, avec la diversification des utilisateurs (donc des besoins) et des applications. Les premiers utilisateurs d'Internet étaient des universitaires, des chercheurs et étudiants peu rebutés par les commandes Unix, attirés par les courriels, les groupes de discussion et le transfert de fichiers. La création en 1992 du World Wide Web, un outil de navigation doté d'une interface graphique d'une grande simplicité, a favorisé l'ouverture d'Internet au grand public où l'utilisateur novice accède aux informations de son choix en quelques clics. Avec le Web, la forme des informations s'est rapidement enrichie, les contenus deviennent multimédias et associent textes, images, séquences vidéo et sons. Le transport de la voix sur Internet (VoIP[4] — Voice over IP) est une autre application importante qui a émergé au même moment que le Web, suivie par la diffusion vidéo sur Internet (mini-journaux TV, clips musicaux, etc. . .). En 1999, le succès de Napster, un logiciel établissant un système d'échange direct de fichiers entre ordinateurs connectés à Internet, a initié le déploiement des applications d'égal à égal (P2P — Peer-to-Peer). Les années 2000 marquent surtout le début de la convergence d'Internet, de la téléphonie et de la télévision avec l'offre *triple-pay* des fournisseurs d'accès. Internet est maintenant un réseau quasi incontournable pour le partage de données et les communications en général, utilisé dans des domaines aussi variés que le travail, le commerce, le divertissement, l'éducation ou la culture par exemple.

En 15 ans, les caractéristiques du trafic d'Internet ont beaucoup changé, tant en volume qu'en diversité. Ce trafic est désormais constitué des données traditionnelles correspondant

[1] SDH est une technologie de transmission longue distance sur la fibre optique qui fournit des débits jusqu'à 10 Gbits/s. Elle est utilisée pour les liaisons terrestres et sous-marines entre plaques régionales, continentales et intercontinentales qui forment les autoroutes de l'information.

[2] ADSL permet aux particuliers d'être raccordés à Internet par la paire téléphonique avec des débits jusqu'à 20 Mbits/s.

[3] 10-Gigabit-Ethernet fournit des liaisons jusqu'à 40km sur une fibre optique monomode tout en assurant la compatibilité avec un réseau SDH. Elle commence d'ailleurs à concurrencer SDH pour les liaisons entre plaques régionales.

[4] Les premiers logiciels de VoIP ont été développés entre 1992 et 1996, ce sont VIC (Video Conferencing Tool) puis VAT (Voice audio Tool) à l'université de Berkeley, NETVOT (Network Voice Terminal) à l'université de Massachusetts, FREEPHONE à l'université de Sophia-Antipolis et RAT (Robust Audio Tool) à l'université de Londres. Mais la VoIP s'est popularisée plus tard avec l'arrivée de logiciels comme Netmeeting, et aujourd'hui Skype.

aux documents textuels, mais également de données relatives à la voix, à la vidéo en téléchargement (archivage), à la demande ou en temps réel (surveillance, visioconférence). D'après une analyse effectuée en 2004 sur un lien de France Telecom interconnectant plusieurs zones ADSL à Paris, le trafic web représente 49% du volume qui y transite, 19% du trafic est de type P2P, le transfert de fichiers s'élève à 5% du volume, les applications audio/vidéo génèrent 2,5% du trafic tandis que les courriels ne représentent que 0,14% du trafic total (Owezarski *et al.*, 2004).

Toutefois, bien qu'accompagné d'une évolution des infrastructures matérielles, l'Internet campe toujours sur les même fondements en termes de routage et de transport de données. Le protocole IPv6 (Deering et Hinden, 1998) qui a été spécifié en 1998 n'est toujours pas déployé en pratique et le transport des données repose depuis plus de vingt ans sur TCP (Transmission Control Protocol) et UDP (User Datagram Protocol). TCP est un protocole orienté connexion qui assure un service de transport fiable des données alors qu'UDP est un protocole sans connexion fournissant un service non fiable. Toujours d'après l'étude menée en 2004 sur un lien d'interconnexion de plusieurs plaques ADSL de France telecom, le trafic TCP représente 87,8% en termes de volume de trafic et 49,8% du nombre de flux. TCP ne considère pas d'autre critère de qualité de service que la fiabilité, il garantit que les données sont livrées dans leur ordre et leur intégrité, sans duplication ni perte, rien de plus. TCP ne convient pas aux applications qui ont d'autres besoins de qualité de service, en particulier les applications engageant le transport d'informations multimédias.

La Qualité de Service (QdS ou QoS) dans un réseau s'exprime à l'aide de métriques pour évaluer les performances du réseau en termes de bande passante, de délais, de gigue (variation des délais), de taux d'erreur binaire et de pertes de paquets. Exprimée en bits/s, la bande passante d'un réseau correspond en réalité à la notion de débit ou de capacité. La bande passante exprime la capacité d'un lien tandis que la bande passante disponible sur un chemin indique le minimum de la bande passante de chacun des liens constituant le chemin. Le délai de transmission et la gigue sont exprimés en secondes. Le taux d'erreur binaire (BER) dépend principalement du médium (air, câble) et des technologies de couches 1 et 2. Au niveau IP, il faut adopter une approche par paquets : ceux-ci peuvent être perdus, dupliqués et peuvent également arriver dans un ordre différent de celui de l'émission. Les paquets sont considérés perdus s'ils sont effectivement perdus au niveau d'un nœud, mais aussi s'ils arrivent trop tard ou hors séquence et que le protocole de transport les écarte, ou bien encore s'ils contiennent des erreurs. Les cœurs de réseaux actuels sont très rapides et offrent un taux d'erreur binaire très faible, les pertes de paquets sont donc essentiellement dues à des congestions qui occasionnent des débordements des files d'attente au niveau des nœuds. Les réseaux sans fil, sujets à un BER importants sont hors contexte mais ils intègrent des mécanismes de correction aux niveaux 1 et 2.

Deux approches sont possibles pour traiter la QoS dans l'Internet : soit en agissant sur le réseau lui-même de manière à obtenir les métriques de QoS recherchées, soit en agissant sur les applications qui utilisent ce réseau.

La première approche consiste à traiter la QoS au niveau du réseau à l'aide de mécanismes déployés au niveau des nœuds. Intserv (Braden *et al.*, 1994) est l'un de ces mécanismes. Son but est de réserver les ressources nécessaires tout au long du chemin pour tous les paquets du flot de données généré par une application donnée. La réservation des ressources est effectuée au moyen d'un protocole de réservation comme RSVP (Resource ReSerVation Protocol), ce qui implique que tous les nœuds du chemin soient compatibles avec Intserv/RSVP. Ce type d'adaptation de la QoS fournie par le réseau en fonction des besoins n'est pas déployable dans l'Internet car la réservation de ressources ne résiste pas au facteur d'échelle d'un réseau

constitué d'une multitude de réseaux gérés par des entités différentes.

Diffserv est un autre mécanisme de traitement de la QoS sur les réseaux IP. Il s'appuie sur la classification des paquets et le traitement de ceux-ci selon trois classes de services : EF, AF et BF. Les paquets de la classe EF (Expedited Forwarding) bénéficient d'un meilleur service, avec une bande-passante garantie, un taux de perte, des délais et une gigue faibles. La classe AF (Assured Forwarding) est subdivisée en plusieurs niveaux de priorité avec une garantie de plus en plus faible du niveau de QoS. La différenciation des paquets ne se fait plus sur leur appartenance à un flot particulier mais sur une marque présente sur le champ ToS (Type of Service) de l'en-tête IP. Enfin, la classe de service BF « best-effort » qui sera traitée au mieux selon les ressources disponibles sur l'instant. Diffserv peut passer à l'échelle mais comme il ne définit aucune réservation de ressources, il n'offre par conséquent aucune garantie de QoS pour le trafic d'une application et d'un utilisateur donné.

Le problème avec ce type d'architecture de QoS, c'est que les mécanismes sont déployés dans les nœuds du cœur de réseau. Les utilisateurs lambdas n'y ont pas accès en pratique et ne peuvent donc pas obtenir une QoS garantie de bout en bout.

La seconde approche consiste à adopter des mécanismes de bout en bout pour traiter le problème de la QoS. Les applications communicantes adaptent ainsi leurs exigences de QoS à l'état du réseau par des mécanismes de bout en bout mettant en œuvre une boucle de retour (*i.e.* , le *feedback*) entre émetteur et récepteur. Ce fonctionnement en boucle fermée considère le réseau comme un élément dont on ne contrôle pas la QoS mais dont il faut mesurer les grandeurs (taux de pertes, délais) pour adapter le comportement de l'application à l'émetteur et/ou au récepteur. Ce feedback est assuré par un protocole au dessus d'IP, typiquement au niveau transport. Cette nécessité de transmettre au mieux en fonction de la QoS fournie par le réseau est la raison d'être des protocoles dits *end-to-end*. Ces protocoles peuvent être implantés au-dessus d'IP, soit au niveau transport, ce qui est intéressant si le service fourni est suffisamment générique pour convenir à toute une classe d'applications, soit au niveau application (user-level) si le service fourni par le protocole est très spécifique et n'est pas voué à être réutilisé. Ils doivent fournir les contrôles d'erreur, de flux et de débits, des procédures d'établissement et de fermeture de connexion ainsi que la remise en ordre des paquets. Le contrôle d'erreur a pour but la détection et la récupération (par retransmission) des paquets perdus. Le contrôle de flux doit limiter le débit de manière à ne pas dépasser les capacités d'admission de l'un des deux hôtes situés aux extrémités. Le contrôle de flux ne tient donc pas compte des différentes capacités des nœuds intermédiaires, d'où la nécessité d'un contrôle de débit pour éviter les congestions (dépassement de buffer) au niveau des routeurs. TCP et ISO TP1 à 4 sont typiquement des exemples de protocoles end-to-end disposant de ces contrôles. Les protocoles qui respectent cette approche de bout en bout ont l'avantage de pouvoir être déployés partout sur l'Internet, et ce malgré la diversité des technologies qui supportent les transmissions. Les traitements que ces protocoles nécessitent sont en outre effectués par définition aux extrémités, le problème du facteur d'échelle est donc résolu. De plus, l'approche de bout en bout (traitement aux extrémités) n'est pas contradictoire avec l'utilisation d'une architecture de type Diffserv (traitement par les nœuds), elles sont au contraire complémentaires.

Contribution et périmètre de la thèse

Les travaux présentés dans ce mémoire de thèse forment une contribution au transport de données multimédias sur Internet en considérant l'adaptation de la QoS de bout en bout, principalement par des mécanismes de contrôle d'erreurs et de pertes de paquets. En effet,

une des différences fondamentales entre les données multimédias (voix, vidéo, images ...) et les données classiques (documents textuels, transactions commerciales ...) réside dans leur propension à tolérer les erreurs et les pertes d'informations.

Pour les applications multimédias qui ont cette tolérance, — elles ne l'ont pas toutes — un transport fiable avec TCP constitue une solution par défaut, mais elle n'est pas optimisée : l'excès de fiabilité fourni par le protocole a mécaniquement un coût sur le temps de réponse du service de transport et la consommation des ressources du réseau.

D'une manière générale, les applications multimédias ont des besoins en termes de services de transport plus complexes que ceux fournis par TCP ou UDP. Ces besoins ont été représentés dans (Diaz *et al.*, 1995) et (Amer *et al.*, 1994) par le concept de connexion d'ordre partiel, ou POC (Partial Order Connection). Les besoins des applications sont projetés selon trois axes qui sont relatifs à la fiabilité du transport, l'ordre de délivrance des données à l'application et les contraintes de temps (échéances temporelles) comme schématisé sur la figure 1. Or ni TCP ni UDP n'adressent ces besoins spécifiques. Au contraire, une POC va fournir un service qui prend en considération les dimensions relatives à l'ordre et l'échéance temporelle des données, en faisant bien sûr des concessions sur le niveau de fiabilité.

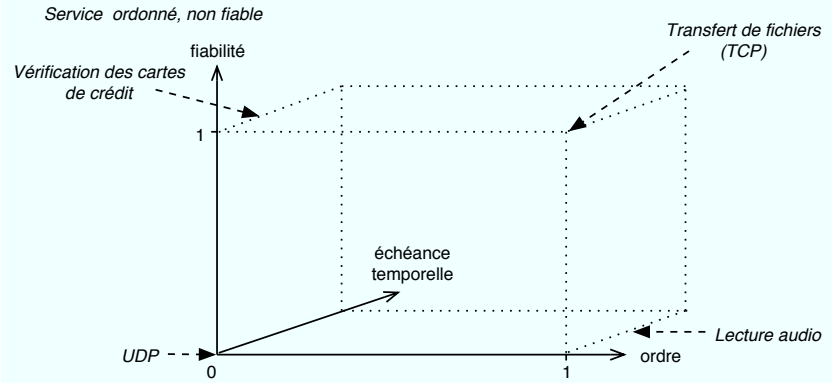


FIG. 1: Les trois axes de la QoS requise par les applications multimédias. Les services fournis par TCP (ordre et fiabilité total) et UDP (ordre et fiabilité nuls) sont indiqués.

Nos travaux peuvent s'inscrire dans la mouvance des POCs puisque nous proposons un protocole à fiabilité partielle et ordre total pour le transport d'images fixes sur Internet. Les travaux pourraient être généralisés au transport de séquences d'images pour les applications sans contrainte temporelle. Nous nous plaçons dans le cadre des applications client-serveur unicast. Le cas des applications multipoints (un serveur, une diffusion et plusieurs clients) n'est pas traité, il constitue à lui seul un sujet de recherche à part entière (remarquons qu'un contrôle partiel des erreurs serait sûrement encore plus bénéfique dans les applications multicast et pourrait réduire le périmètre d'explosion des acquittements).

Nous considérons de manière générique l'Internet en tant que grand réseau ouvert et hétérogène. Nous ne considérons pas les environnements spécifiques, par exemple les réseaux sans-fil ad-hoc ou les constellations de satellites qui impliquent des taux d'erreur binaire ou des produits (délai $\times$ bande passante) particuliers, même si ces environnements particuliers peuvent servir de prolongement de nos travaux.

Organisation du document

Ce document est organisé en deux parties de deux chapitres chacune. La première partie développe nos contributions dans le domaine des protocoles de transport à fiabilité partielle et leur positionnement par rapport à la littérature. La seconde partie développe nos contributions dans le domaine d'application, le transport d'images compressées, en particulier pour optimiser globalement toute la chaîne de traitement de l'image : compression/codage–paquétisation/classification–transport–décodage/restitution.

Le premier chapitre présente d'abord le contexte dans lequel se situent nos travaux, l'Internet comme grand réseau ouvert sans garantie de QoS, les caractéristiques de son trafic et les erreurs qui peuvent se produire. Il fournit aussi un état de l'art des protocoles de transport, ceux qui délivrent un service fiable comme ceux qui délivrent un service partiellement fiable. Il montre qu'il existe quelques protocoles à fiabilité partielle mais qu'aucun d'eux n'est parfaitement adapté au transport des images fixes compressées.

Le second chapitre décrit 2CP-ARQ, le protocole à fiabilité partielle que nous avons développé au CRAN pour fournir un service optimisé au transport d'images fixes. Ce chapitre présente en détail les spécifications du protocole et développe un modèle analytique du temps de réponse au service en fonction de l'état du réseau et du niveau de fiabilité exigé par l'application. Ce modèle est utilisé pour évaluer les performances des services fournis par le protocole 2CP-ARQ et les comparer à un protocole fiable tel que TCP.

Le troisième chapitre étudie l'intérêt et l'efficacité du protocole dans le cadre du transport d'images codées selon le standard de compression actuel JPEG2000. Ce chapitre montre en particulier que le système de codage adopté dans JPEG2000 réduit grandement l'intérêt d'utiliser un transport à fiabilité partielle pour ce type d'image.

Le dernier chapitre associe un schéma de compression d'images et un schéma de paquétisation parfaitement adaptés à l'utilisation d'un service de transport à fiabilité partielle de type 2CP-ARQ. Les performances de 2CP-ARQ sont ensuite évaluées en termes de temps de réponse du service et de qualité des images reconstruites expérimentalement. Les résultats montrent qu'un gain de temps de 3 à 18% est envisageable lorsque le réseau est sujet à des pertes de paquets en comparaison avec un transport fiable de type TCP.

Première partie

Transport à fiabilité partielle sur l’Internet selon une approche de bout en bout

Chapitre 1

Contexte : Internet et transport de bout en bout

Introduction

Ces quinze dernières années, le développement du réseau Internet a pris une ampleur considérable tant au niveau des infrastructures que du point de vue du public qui l'utilise aujourd'hui quotidiennement. Il s'est désormais démocratisé et est devenu un outil incontournable pour la grande majorité des activités professionnelles et de loisirs parce qu'il peut véhiculer tous les types d'informations, de services, et de produits non physiques. Parmi les informations qui y transitent, les données multimédias sont devenues omniprésentes (photographie, publicité, présentation de sites web, imagerie médicale et militaire, vidéo à la demande). Elles engagent généralement de gros volumes de données. Ce constat peut être illustré par quelques exemples : le volume d'un film, même compressé, se compte en Giga octets (Go) pour une qualité cinématographique ; le poids des fichiers photographiques s'élève à plusieurs Mega octets (Mo) pour une image exploitant la résolution native des capteurs d'aujourd'hui ; la présentation des sites web est devenue très riche en images, séquences animées et sons, même compressées avec une qualité juste suffisante pour un affichage sur écran, ces données nécessitent aujourd'hui que l'utilisateur soit équipé en haut débit. L'imagerie spécifique des domaines médical, militaire ou astronomique engage des volumes colossaux par image. Ainsi les images médicales 3D se présentent sous la forme de piles d'images 2D dont les données se comptent en Go. Sachant qu'à l'ère du tout numérique, les rares données multimédias encore analogiques (TV) vont migrer, que les technologies relatives à la capture d'images ne cessent d'augmenter la densité des capteurs en photosites, que l'avènement de la Haute Définition (HD) est en cours dans les domaines vidéo et TV, nous pouvons prétendre que les données multimédias sont en première ligne sur le front des divers contenus numériques à transporter.

1.1 Caractéristiques de l'Internet

Le réseau Internet est par définition un méta-réseau issu de l'agrégation d'une multitude de réseaux appartenant à des entités diverses (opérateurs, États ...). Il s'agit d'un réseau maillé, c'est-à-dire que les nœuds y sont interconnectés pour offrir plusieurs chemins reliant la source à la destination. Ce réseau de réseaux repose sur le protocole Internet (IP) spécifié dans (Postel, 1981*a*) qui est un protocole routable. Les origines d'IP remontent à 1958 lors de

la création de l'Advanced Research Project Agency (ARPA) et sur la proposition de créer un réseau de communication robuste. Les enjeux militaires d'un tel réseau ont été clairement affichés en 1972 avec la nouvelle dénomination de cette agence qui devint la Defense Advanced Research Project Agency (DARPA) avant de retrouver sa dénomination originale en 1993. Pour satisfaire une telle contrainte de robustesse, l'Internet s'appuie sur la redondance de chemins du réseau maillé, et le rôle du protocole IP est de permettre la sélection d'un chemin par les nœuds, de proche en proche vers la destination. Le concept fondamental sur lequel repose IP consiste à découper au niveau de la source le message à transmettre en plusieurs paquets appelés datagrammes. Ces paquets sont ensuite réassemblés au niveau du destinataire pour reformer le message original. Le protocole IP est donc robuste dans le sens où il existe plusieurs chemins pour atteindre la destination. Outre la fragmentation des datagrammes, IP intègre une fonctionnalité d'adressage qui est utilisée par les routeurs constituant les nœuds intermédiaires pour aiguiller les paquets en fonction de l'adresse de destination. La notion de circuit de communication est absente du service tel qu'il est offert par IP puisque les datagrammes relatifs à un même message peuvent emprunter des chemins différents, se doubler, être dupliqués, arriver dans le désordre ou se perdre. Le rôle d'IP est donc d'aiguiller les paquets à chaque nœud du chemin vers le prochain nœud, rien ne permet à IP de garantir qu'un paquet donné arrive à destination, et s'il y parvient, IP ne fournit aucune assurance sur la date d'arrivée. Il en va de même pour tous les paquets de tous les flux de tous les utilisateurs, c'est en ce sens que l'Internet est qualifié de réseau « best-effort[1] » , le trafic est traité au mieux selon les ressources disponibles sur l'instant, sans garantie.

La prise en compte de la qualité de service (QdS – QoS) peut être réalisée de différentes manières :

– Utilisation du champ ToS par les routeurs et traitement différencié des paquets,
– Diffserv,
– Intserv.

Une première possibilité consiste à utiliser le champ inutilisé Type de Service (ToS) de l'entête IP pour marquer les paquets de certains flux dont les besoins en QoS sont critiques. Les routeurs doivent ensuite être configurés pour traiter ces paquets de façon particulière, par exemple en écartant d'abord les paquets qui ne portent pas cette marque prioritaire lorsque les files sont en passe d'être saturées. Un second moyen d'améliorer la QoS pour certains flux de paquets consiste à utiliser des mécanismes comme DiffServ (Blake *et al.*, 1998) qui fait usage du champ ToS (renommé champ Differentiated Service – DS) pour étiqueter les datagrammes selon des classes de services de priorités différentes qui seront traitées selon des politiques différenciées par les routeurs. Une limitation de ce type d'approche par classification réside dans l'impossibilité d'offrir des garanties pour tous les flux et finalement les transmissions usuelles de l'utilisateur courant continuent à être traitées en dernier sans garantie. Une troisième possibilité consiste à réserver des ressources tout au long de la liaison utilisée par un flux prioritaire à l'aide de mécanismes comme Intserv et l'utilisation de protocoles de signalisation comme RSVP, ce dernier permettant à des routeurs de réserver des ressources en termes de bande-passante (Braden *et al.*, 1997). Malheureusement, pour être efficace RSVP doit être utilisé par tous les nœuds intermédiaires qui, nous le rappelons, peuvent varier avec la modification du chemin au cours du temps, appartiennent et sont gérés par des groupes concurrents qui n'ont pas forcément les moyens ni intérêt de garantir la bande passante à des flux qui ne les intéressent en aucune façon et qui ne font que transiter par ses nœuds. L'approche RSVP n'est donc pas envisageable dans le cas du trafic habituel généré par l'utilisateur

[1]Cet anglicisme se traduit par *effort maximum*. Il exprime une classe de qualité de service dans laquelle les paquets sont traités uniformément, sans aucune indication de trafic et sans garantie de livraison.

courant (consultation de ressources web textuelles et graphiques, consultation de boîtes pour le courrier électronique ...). Dans sa version 6 spécifiée dans (Deering et Hinden, 1998), IP devrait permettre l'optimisation des communications IP car il intègre des fonctionnalités de gestion de QdS (entre autres fonctionnalités relatives à la sécurité et à la mobilité), cependant IPv6 est loin d'être généralisé sur tous les nœuds de l'Internet.

Comme l'Internet est un méta-réseau issu de l'agrégation d'une multitude de réseaux appartenant à des entités diverses (opérateurs, États ...), il est par conséquent administré par morceaux et non pas sur tout le chemin qui interconnecte deux acteurs communicants. Les chemins empruntés par les paquets des entités communicantes sont en outre variables et indéterminés *a priori* du fait du maillage du réseau, et potentiellement asymétriques. Le déploiement d'une architecture de QoS située sur les nœuds du réseau est donc rendue délicate. Pour toutes ces raisons, nous adoptons une approche de bout en bout, nous considérons en effet que les applications qui utilisent l'Internet pour transmettre des données doivent s'accommoder de la qualité de service fournie par le réseau sur l'instant.

1.1.1 Approche de bout en bout et qualité de service offerte par le réseau

Lorsque l'on considère le point de vue des applications qui utilisent l'Internet pour échanger des données, seul le trafic qu'elles génèrent et les effets que le réseau a sur les paquets de ce trafic peuvent être pris en compte. Ceci conduit à adopter une approche de bout en bout. Ainsi, le réseau peut être vu comme une boîte noire par laquelle transitent les paquets. Ces derniers peuvent alors subir des pertes, être retardés voire se doubler. La QdS offerte par le réseau s'exprime en termes de bande passante disponible, de délai de transmission unidirectionnels et aller-retour (*round-trip time*, RTT), de gigue (variations de délai) et de taux de perte de paquets.

L'internet étant un réseau fondé sur le concept de *best-effort*, les paquets sont acheminés sans aucune garantie de délai ni de fiabilité. Leur contrôle doit être pris en charge par le protocole de transport. On peut distinguer cinq types d'erreurs pendant l'acheminement des paquets : la corruption, la perte, le retard, le désordre et la duplication.

- La corruption : elle est due aux perturbations du signal physique causant des erreurs sur les bits. C'est un phénomène rare sur des liens métalliques qui devient non négligeable sur des liaisons sans-fil. Les paquets corrompus sont bien détectés et écartés par les protocoles au niveau de la couche liaison. Dans un contrôle de bout en bout, ce type d'erreur est donc assimilable à une perte de paquets.
- La perte : les équipements d'interconnexion contrôlent l'admission des paquets dans les files d'attente pour éviter leur débordement. Quand le trafic est trop élevé, des paquets doivent être écartés et sont définitivement perdus. Ces pertes de paquets constituent un indicateur sur la congestion du réseau. C'est un phénomène très fréquent dans l'Internet.
- Le retard : le délai d'acheminement des paquets dans le réseau dépend de la bande passante disponible sur leur route et de l'encombrement des nœuds d'interconnexion. Ce délai est variable d'un paquet à l'autre pour les applications qui imposent un délai de bout en bout borné, des paquets vont arriver trop tard si l'effet de gigue est trop fort. Ces paquets trop tardifs sont déclarés perdus puisque n'ayant plus d'utilité pour l'application.
- Le désordre : le routage dans l'Internet étant traité paquet par paquet, les paquets d'un même flux ne suivent pas nécessairement tous le même chemin. Il n'y a donc aucune garantie que l'ordre dans lequel les paquets ont été émis à la source soit préservé à l'arrivée. Une numérotation de la séquence des paquets dans l'ordre de leur émission suffit généralement à traiter ce type d'erreur.

– La duplication : cette erreur est produite par les protocoles qui ont la charge d'assurer la fiabilité. Ces protocoles sont programmés pour transmettre plusieurs fois un même paquet jusqu'à ce qu'il soit reçu. Ces retransmissions sont déclenchées par minuterie (timeout) et peuvent ainsi survenir à tort en cas de retard important ou en cas de perte d'acquittement. À défaut de pouvoir éviter les duplications, le récepteur doit être capable de les détecter à l'arrivée pour les écarter systématiquement. La numérotation des paquets par l'émetteur suffit pour cela.

Parmi les critères de QoS, seules les pertes de paquets peuvent être corrigées de bout en bout, par des mécanismes de recouvrement. Les délais ou retards introduits par le réseau ne peuvent pas être supprimés ou réduits lorsque l'on adopte une approche de bout en bout. Il est par contre possible de rendre la gigue invisible à l'application à l'aide d'un mécanisme d'amortissement de gigue. Son principe consiste à mémoriser les paquets en réception pour retarder leur utilisation par l'application. Cela suppose donc un buffer d'autant plus important que le produit bande passante$\times$délai et la gigue sont grands, et ce système ne réduit pas le retard effectif qui est au moins au niveau du délai maximal provoqué par le réseau. Ce type de mécanisme est intéressant dans le cadre des applications qui nécessitent une certaine cadence dans le traitement de l'information reçue, c'est par exemple le cas de la vidéo à la demande (VoD) qui oblige l'application à jouer un film cinématographique à 24 images par seconde, ou bien le cas du streaming audio qui implique la lecture des échantillons à une fréquence donnée.

Un des points clés de notre travail consistant en une correction originale des pertes de paquets, nous allons détailler la manière dont elles apparaissent et sont perçues par les deux parties communicantes. La modélisation des pertes sera un des fondements de notre contribution exposée au chapitre 2.

1.1.2 Modélisations des pertes de paquets

En règle générale les pertes de paquets dans l'Internet sont dues aux congestions. Les pertes dues aux congestions sont corrélées, c'est-à-dire que lorsqu'un paquet est perdu, le suivant a plus de chances de l'être qu'en temps normal (Zhang, 2001) (Yajnik *et al.*, 1999).

Une étude des pertes de paquets sur l'Internet a été menée d'après des enregistrements de traces réelles afin de déterminer l'adéquation de trois modèles différents pour expliquer les pertes (Yajnik *et al.*, 1999) et mettre en évidence une éventuelle dépendance entre elles. Selon la trace étudiée, trois modèles sont alternativement valables : le processus de Bernoulli (voir Fig. 1.1), le modèle markovien à deux états aussi appelé modèle de Gilbert (voir Fig. 1.2) (Gilbert, 1960), ainsi que des chaînes de Markov d'ordre n. Le modèle de Gilbert étendu du même ordre (voir Fig. 1.3) est couramment utilisé pour modéliser les pertes (Sanneck et Carle, 2000) (Jiang et Schulzrinne, 2000). La différence entre un modèle markovien d'ordre n et Gilbert étendu est une question de nuance. Alors que l'événement futur dépend des n derniers événements dans le modèle de Markov, avec Gilbert étendu il peut dépendre d'un nombre inférieur d'événements précédents (jusqu'à n).

Le processus de Bernoulli ne fait apparaître aucune dépendance entre les pertes, chaque paquet a la même probabilité p d'être perdu, quel que soit l'état (perdu ou non) des paquets précédents. En revanche, la chaîne de Markov à deux états tient compte de l'état dans lequel se trouvait le paquet précédent et un paquet a donc une probabilité p d'être perdu si le paquet précédent ne l'est pas, et une autre probabilité $1 - q$ de se perdre lorsque le précédent est perdu. Généralement $p < 1 - q$ et si $p = 1 - q$ le modèle se réduit alors au cas particulier de Bernoulli.

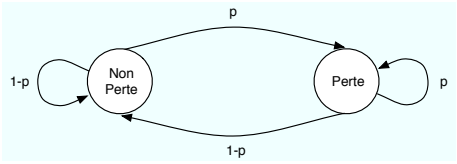


FIG. 1.1: Pertes de paquets selon Bernoulli.

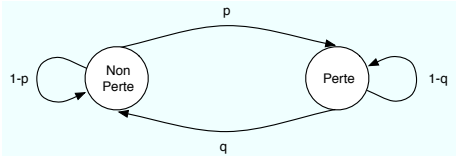


FIG. 1.2: Modèle de Gilbert pour les pertes de paquets sur l'Internet.

Les 76 heures de trafic unicast et multicast analysées par Yajnik et al. dans (Yajnik *et al.*, 1999) ont été découpées en 38 segments. Le process de Bernoulli s'est révélé pertinent pour expliquer les pertes pour 7 d'entre eux, le modèle de Gilbert explique 10 segments, et pour modéliser les pertes des 21 segments restants, il faut avoir recours à des chaînes de Markov d'ordre 2 à 42. Notons que l'utilisation d'un process de Bernoulli là où un modèle à mémoire tel que Gilbert est approprié, conduit à une sous-estimation des rafales de paquets perdus et une surestimation de pertes uniques. Cette propriété est utilisée au chapitre 2 lors de l'analyse du service 2CP-ARQ que nous proposons.

Les délais d'acheminement variables d'un paquet à un autre sont dépendants entre eux comme c'est le cas pour les pertes. Ils sont donc aussi modélisés par des modèles à mémoire et les délais sont estimés de manière probabiliste avec des fonctions de distribution (Jiang et Schulzrinne, 2000). Contrairement aux pertes, les délais imposés par le réseau ne peuvent pas être corrigés, c'est pourquoi dans la suite de ce travail nous considérons les délais comme une variable sur laquelle nous n'avons pas de moyen d'action.

Il est important de remarquer que la modélisation des pertes n'est qu'un moyen de décrire un phénomène qui ne dépend que des caractéristiques du réseau d'une part, et des caractéristiques du trafic de ses utilisateurs d'autre part. Si l'on considère que les pertes sont essentiellement liées aux congestions, il faut alors s'intéresser au comportement des protocoles de transport à qui incombe le rôle de réagir en fonction des pertes et des délais, pour adapter le débit d'une manière générale. Lorsque l'on désire développer un nouveau service de transport sur l'Internet, il convient ensuite de s'assurer que le comportement obtenu ne dévie pas significativement par rapport à celui des services existants, sans quoi la caractérisation des pertes de paquets et donc leur modélisation pourrait être modifiée.

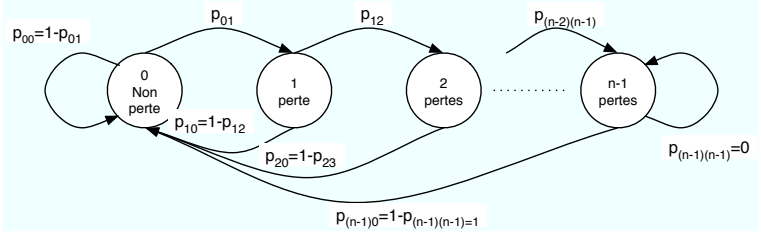


FIG. 1.3: Modèle de Gilbert étendu pour les pertes de paquets sur l'Internet.

1.2 État de l'art des services de transport sur l'Internet

Dans le modèle TCP/IP, la notion de circuit de communication logique est déléguée à la couche transport. En outre, contrairement au modèle OSI qui intègre le contrôle de flux comme fonctionnalité de couche 3, la couche réseau (IP) de la pile TCP/IP délègue aussi cette fonctionnalité à la couche supérieure. La couche transport revêt donc une importance capitale dans le bon fonctionnement de l'Internet.

La pile TCP/IP définit deux protocoles de transport qui sont utilisés par presque la totalité du trafic sur l'Internet : TCP et UDP. Plus de 99% des flux sont en effet transportés par l'un ou l'autre de ces deux protocoles (Owezarski *et al.*, 2004). TCP est très largement majoritaire avec 85 à 90% du nombre total de paquets tandis que UDP occupe les 10 à 15% restant. En toute rigueur le terme « segment » devrait être employé au niveau transport au lieu du terme « paquet », mais comme ce dernier est largement répandu dans la littérature, nous l'employons tout au long de cette thèse. En termes de volumes de données exprimés en octets, la répartition est de 94 à 98% pour TCP et 2 à 6% pour UDP (Olivier *et al.*, 2003).

1.2.1 Le service offert par UDP

UDP est spécifié dans (Postel, 1980). Ce protocole offre un service de transport simple, sans connexion. L'en-tête des paquets a une longueur de 8 octets, il contient les numéros de ports source et destination nécessaires à l'identification des applications, la longueur du datagramme ainsi qu'une somme de contrôle (checksum) pour la détection des erreurs. Les paquets ne sont pas numérotés et le récepteur ne retourne pas d'accusé de réception ni autre message de rapport. Par conséquent, UDP ne garantit pas la livraison des paquets ni leur ordre. Ce service est adapté pour les applications de type requête/réponse, particulièrement celles engageant des messages courts, typiquement inférieurs à un MTU (c'est le cas du DNS, de SNMP, ...). Mais le service minimaliste de UDP n'est pas suffisant pour transporter des données en temps réel, comme c'est le cas pour la voix, la vidéo ou bien le trafic inhérent aux jeux en réseau. Ces applications interactives requièrent le maintien d'un signal d'horloge pour les besoins de la synchronisation, et pour cette raison les paquets doivent inclure des informations d'horodatage. Elles requièrent aussi une boucle de retour du récepteur vers l'émetteur pour mettre en œuvre des mécanismes d'adaptation de débit et de correction d'erreurs. C'est pourquoi UDP peut être utilisé comme base, mais des fonctions supplémentaires vont être ajoutées via des protocoles « légers » tels que RTP et RTCP.

Les applications qui requièrent un service fiable et ordonné orienté connexion doivent utiliser TCP (Transmission Control Protocol) (Postel, 1981*b*) .

1.2.2 Le service offert par TCP

Contrairement à UDP, TCP est un protocole de type *Automatic Repeat reQuest* (ARQ), c'est-à-dire qu'il utilise un mécanisme d'acquittements et fonctionne en boucle fermée. Les acquittements servent autant à détecter les pertes de paquets (pour les besoins de la correction des pertes) qu'à estimer dynamiquement le délai aller-retour – RTT (pour les besoins de l'adaptation de débit et l'étalonnage du timeout).

Les tâches assurées par TCP sont la création, le maintien et la fermeture de connexion. TCP crée de manière logique le circuit de communication fiable entre la source et la destination, d'où la notion de connexion. Les applications disposent alors d'une connexion qui leur permet d'échanger des données de façon transparente comme si les données étaient locales à la

machine, avec un délai supérieur évidemment. Pour assurer ces fonctionnalités, TCP requière des mécanismes pour :

- recouvrir les pertes de paquets,
- remonter dans l'ordre les données à la couche supérieure (*i.e.* , remettre les paquets dans l'ordre et reformer les ADUs),
- effectuer le contrôle de flux,
- assurer le contrôle de congestions.

La récupération des pertes de paquets nécessite que l'émetteur puisse les détecter. L'émetteur envoie les données en déclenchant systématiquement une minuterie (*timeout*) à chaque émission d'un paquet. Le récepteur acquitte les données correctement reçues et c'est l'absence d'acquittement après le *timeout* qui signifie une perte pour l'émetteur. Les données sont ensuite retransmises à partir du paquet perdu. Un acquittement peut évidemment se perdre, ce qui conduirait à une fausse détection de perte par l'émetteur. Les acquittement ont cependant une valeur cumulative (ils acquittent également toutes les données précédentes), ce qui permet de limiter les fausses détections de perte de données lorsqu'un ACK se perd. Notons que pour que la détection soit possible, il est nécessaire que le *timeout* soit plus grand que le délai nécessaire à l'envoi du paquet, sa réception, l'envoi de l'acquittement et le retour de cet ACK (RTT). Cependant, si la valeur du *timeout* est trop importante, cela conduit à une perte de temps en cas de perte d'un paquet. Le retour des ACKs permet justement à l'émetteur d'estimer le RTT et par la même occasion de régler finement le *timeout* au gré des variations du retard du réseau.

Les paquets n'arrivent pas forcément au récepteur dans l'ordre d'émission. Pour les remettre dans l'ordre et reconstruire le flux avant de le remonter à la couche supérieure, le récepteur utilise le numéro de séquence des paquets.

La rapidité avec laquelle les données sont transmises dépend naturellement de la capacité du réseau à véhiculer l'information (bande passante), mais aussi des capacités de l'émetteur à les envoyer et du récepteur à les accepter. Dans l'hypothèse où le réseau n'est pas un facteur limitant, il est indispensable que l'émetteur n'inonde pas le récepteur en envoyant trop de données par unité de temps. C'est précisément la fonction du contrôle de flux que d'aligner dans le meilleur des cas le débit à l'émission sur la capacité d'admission du récepteur. Cette fonction est réalisée lorsque le récepteur indique à la source la valeur de la fenêtre de réception (*receive window – rwnd*). Cette valeur indique quelle quantité d'octets le récepteur peut accepter d'affilée sans acquitter, c'est-à-dire stocker dans son tampon. La valeur de la fenêtre *rwnd* est notifiée par le récepteur à l'établissement de la connexion, puis elle est maintenue à jour au fur et à mesure que le récepteur reçoit des données ou les remonte à la couche supérieure. La valeur courante peut être retournée à l'émetteur via les ACKs, lorsque *rwnd* atteint 0, l'émission cesse dans l'attente d'une notification de la part du récepteur.

TCP a la faculté d'adapter son débit en fonction des capacités du récepteur, mais aussi en fonction des caractéristiques du réseau à chaque instant (délais et pertes), c'est la fonction du contrôle de congestion. Le contrôle de congestion de bout en bout a pour objectif d'éviter les congestions et de contribuer à les résorber lorsqu'elles surviennent. Les algorithmes mis en jeu sont les suivants : *slow-start*, *congestion avoidance*, *fast-retransmit* explicités par Jacobson dans (Jacobson, 1988), et *fast-recovery* développé deux ans plus tard dans (Jacobson, 1990). Ces quatre algorithmes sont spécifiés dans le RFC 2581 (Allman *et al.*, 1999). Ils permettent de contrôler le débit de la source de sorte à le réduire lorsque des congestions apparaissent (détection de perte de paquet), et inversement à l'augmenter lorsque les conditions du réseau sont bonnes. Pour réaliser ce contrôle de débit, ces algorithmes font varier une valeur appelée fenêtre de congestion (*congestion window, cwnd*). La source peut ensuite calculer sa fenêtre

d'émission $swnd$ telle que $swnd = min(cwnd, rwnd)$. Il s'agit de la quantité de données qui peut être émise d'affilée par anticipation, *i.e.* , sans attendre un retour par acquittement de la part du récepteur.

TCP requiert au minimum les algorithmes *slow-start* et *congestion avoidance*. Une valeur de seuil $sstresh$ décide de l'exécution de l'un ou l'autre. Slow-start est exécuté lorsque la valeur courante $cwnd \leqslant ssthresh$, cet algorithme augmente de façon exponentielle la valeur de $cwnd$. Lorsque $cwnd \geqslant ssthresh$, c'est l'algorithme d'évitement de congestion qui est appliqué, l'augmentation de $cwnd$ est alors additive avec une majoration d'un segment (1460 octets typiquement) par RTT. Si $cwnd = sstresh$ alors TCP peut employer l'un ou l'autre des algorithmes. Quel que soit l'algorithme en cours, un *timer* est déclenché à chaque envoi de paquet. Si le timer expire (*timeout*) avant acquittement des données contenues dans le paquet qui l'a déclenché, alors le paquet est considéré perdu. La valeur du seuil $sstresh$ est alors réduite de moitié, et la fenêtre $cwnd$ est réinitialisée à la valeur lw, typiquement un à deux segments maximum, voir Fig. 1.4.

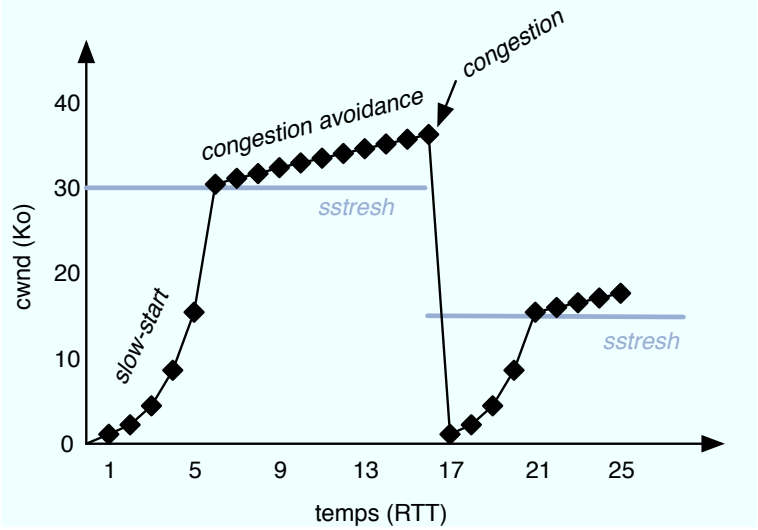


FIG. 1.4: Slow-start, congestion avoidance et évolution de la fenêtre cwnd.

Il existe plusieurs variantes de TCP, certaines sont implantées dans des équipements réseaux, d'autres ne sont encore qu'à l'état expérimental. Toutes sont caractérisées par un contrôle de congestion de bout en bout avec quelques nuances et améliorations au niveau des algorithmes utilisés.

1.2.3 Les variantes de TCP

Selon (Padhye et Floyd, 2001) et (Mathis *et al.*, 1996), les versions de TCP les plus courantes actuellement utilisées sont Tahoe sans *fast retransmit* (FR), Tahoe, Reno, New-Reno, Reno Plus, mais d'autres solutions expérimentales sont recensées, notamment les variantes basées sur les acquittements sélectifs (SACK), TCP-Vegas et TCP-Westwood. D'autres versions ont été proposées pour être utilisées dans un contexte de réseau particulier comme les réseaux ad-hoc, sans-fil ou satellite, c'est-à-dire dont les caractéristiques diffèrent de celles de l'Internet.

1.2.3.1 TCP Tahoe

TCP Tahoe existe en deux versions. La première utilise uniquement les algorithmes *slow-start* et *congestion avoidance*, c'est la version utilisée dans les systèmes d'exploitation windows 98 et 2000. Elle est la moins performante du fait que les pertes sont uniquement détectées par expiration de *timer*. Cela peut faire perdre beaucoup de temps, particulièrement lorsque le taux de pertes est élevé.

L'autre version implante l'algorithme *fast retransmit*. Dans ce cas, le récepteur qui reçoit un paquet hors séquence le signale en répétant plusieurs fois le dernier acquittement, typiquement trois fois. La duplication des acquittements (DUPACK) constitue un moyen plus rapide de détecter les pertes pour l'émetteur, en complément de l'utilisation d'un *timer*.

1.2.3.2 TCP Reno

TCP-Reno exploite un quatrième algorithme par rapport à Tahoe : *fast recovery*. Nous avons vu que *fast retransmit* permet de gagner du temps en évitant un certain nombre de *timeout* mais dès qu'une perte est détectée, la fenêtre $cwnd$ est brutalement réduite de moitié. L'algorithme *fast recovery* réinitialise la fenêtre $cwnd$ à une valeur supérieure égale à $sstresh + ndup \times SMSS$ (nombre d'octets de données par paquet à l'émetteur) lorsque la perte est détectée par *fast retransmit*. La fenêtre bénéficie alors d'un crédit bonus d'un paquet par DUPACK reçu, ce qui évite de repasser par *slow start* et tend à réduire le débit dans des proportions moins élevées. Cette version de TCP était utilisée par d'anciennes versions de BSD et d'Unix.

1.2.3.3 TCP NewReno

Cette variante modifie l'algorithme *fast recovery* de manière à réduire encore le nombre de *timeout* lorsque plusieurs pertes sont groupées dans la même fenêtre, typiquement à cause des ACKs partiels. Un ACK partiel acquitte au moins le paquet dont la perte a été corrigée avec *fast retransmit* mais pas tous les paquets transmis. C'est cet ACK qui fait quitter le mode *fast recovery* dans TCP Reno.

TCP NewReno comble cet écueil parce qu'il ne quitte le mode *fast recovery* que si l'ACK reçu acquitte tous les paquets envoyés. Lorsqu'il s'agit d'un ACK partiel, NewReno retransmet immédiatement le paquet suivant le dernier paquet acquitté, diminue la fenêtre $cwnd$ du nombre de paquets acquittés, et continue à transmettre les paquets suivants par anticipation dans la limite des crédits permis par $cwnd$. Enfin, TCP NewReno est la version couramment utilisée par les Unix/Linux actuels, notamment les serveurs web. NewReno est de ce fait considéré comme le standard sur Internet.

D'autres variantes anecdotiques de TCP Reno existent, notamment RenoPlus qui était implanté dans de vieux systèmes d'exploitation comme Solaris 2.51(SunOS) (Moraru *et al.*, 2003).

1.2.3.4 TCP SACK

Toutes les variantes qui viennent d'être présentées ont un défaut majeur : qu'il y ait un seul ou plusieurs paquets perdus dans la même fenêtre, le récepteur acquitte de façon cumulative et l'émetteur n'aura finalement la connaissance que de la première perte. L'émetteur peut décider de recommencer à retransmettre rapidement à partir du segment perdu, ce type de de reprise est appelé Go-Back-N. Dans ce cas si la fenêtre $cwnd$ avait une valeur élevée à

l'instant précédent la perte, il y a de fortes chances qu'une grande quantité d'information qui va être émise à nouveau ait été déjà reçue par le récepteur. Ce cas de figure tend à réduire l'efficacité et les performances de TCP.

Le mécanisme d'acquittement sélectif (SACK) autorise le récepteur à acquitter les paquets reçus derrière une perte. Combiné à une politique de correction d'erreur de type retransmission sélective (*selective repeat*), SACK permet à l'émetteur de renvoyer uniquement les segments incriminés. Cette technique permet de réduire le nombre de retransmissions, donc d'améliorer le débit utile (*goodput*) pour un débit à l'émission équivalent. TCP-SACK est plus qu'une variante, c'est un mécanisme qui peut être ajouté à toutes les variantes de TCP. L'utilisation des options SACK est spécifiée par Jacobson dans le RFC 1072 (Jacobson et Braden, 1988) puis précisé dans (Mathis *et al.*, 1996), mais elles n'ont jamais été déployées sur l'Internet en pratique.

Toutes les variantes de TCP présentées précédemment disposent d'un contrôle de congestion qui est effectué mécaniquement par des algorithmes qui contraignent le débit à l'émission en fonction des pertes et des retards des retours d'acquittements. Le débit en sortie de la source dépend donc implicitement du taux de pertes de paquets, du temps qui sépare l'émission des paquets de la réception des acquittements par la source, c'est-à-dire du RTT, ainsi que de la taille maximum des paquets. Il n'y a donc pas de mécanisme de mesure ou d'estimation de la bande passante et rien n'indique que TCP applique le débit de sortie optimal en fonction des caractéristiques réelles du réseau. Ce constat a conduit au développement de versions expérimentales de TCP, leur objectif est d'appliquer un contrôle de congestion qui permet d'obtenir un débit optimal en cas de congestion comme lorsque la QoS offerte sur le chemin réseau est bonne.

1.2.4 Les variantes expérimentales de TCP

1.2.4.1 TCP-Vegas

Le problème de TCP Reno c'est que lors de la phase slow-start, il y a sous-utilisation du canal. Puis le débit augmente, ce qui contribue à le sur-utiliser, d'où l'apparition de congestions et un débit de sortie en dents de scie. Finalement le débit optimum n'est jamais trouvé et c'est cela qui constitue la problématique de la recherche de TCP Vegas : avoir le bon débit. Développé à l'Université d'Arizona, TCP Vegas repose sur un algorithme de *congestion avoidance* particulier, une anticipation des congestions et une modification du *slow start* (Brakmo *et al.*, 1994). Vegas tente de détecter les congestions en comparant le débit réel au débit attendu qu'il estime à chaque cycle (RTT). L'algorithme *congestion avoidance* de TCP Vegas est caractérisé par $cwnd = (\text{débit de sortie réel}) \times (\text{base RTT}) + 2 \times SMSS$. On constate que TCP Vegas tente d'adapter un débit optimisé directement d'après les caractéristiques du réseau, c'est-à-dire le produit délai$\times$bande passante, et non plus par une augmentation/réduction purement mécanique de la fenêtre $cwnd$ avec l'effet dents de scie qui en découle. D'après (Brakmo et Peterson, 1995), TCP Vegas permet un débit de 37 à 70% supérieur par rapport à TCP Reno tout en retransmettant moins de paquets à cause des congestions. Le gain dépend de plusieurs paramètres, comme la taille des flux à transmettre par exemple. Dans le cas des petits flux, les versions classiques de TCP sont désavantagées car la totalité des données risque d'être transmise lors du slow-start, soit bien avant d'atteindre le débit de croisière, d'où une perte de temps.

1.2.4.2 TCP-Westwood

TCP-Westwood est proposé par l'Université de Californie Los Angeles (Mascolo *et al.*, 2001). Westwood propose une modification du contrôle de congestion de NewReno tout en restant équitable avec ce dernier. Le principe de Westwood consiste à surveiller constamment les retours d'ACKs pour en extraire une estimation de la QdS fournie par le réseau, notamment sur des réseaux sans-fil pour lesquels les pertes sont nombreuses, (Wang *et al.*, 2002). Cette estimation permet l'ajustement de la fenêtre $cwnd$ et le seuil $sstresh$ lorsqu'une perte est détectée. Une version modifiée appelée TCP Westwood-A (TCPW-A) est présentée dans (Wang *et al.*, 2005). TCPW-A utilise deux nouveaux mécanismes intelligents : un mécanisme qui s'appuie sur l'estimation du débit qu'il est possible d'atteindre (*agile probing*) et *persistent noncongestion detection* (PCND). Le premier ajuste la valeur du seuil $sstresh$ de façon à conserver l'augmentation exponentielle de l'algorithme slow-start plus longtemps. PCND est actif durant la phase de *congestion avoidance*, en cas d'absence de congestion PCND peut activer de nouveau *agile probing*. Si la bande passante disponible varie et peut augmenter rapidement dans de fortes proportions, PCND pourra alors permettre d'augmenter à nouveau $cwnd$ exponentiellement et atteindre rapidement le débit optimal.

Westwood, Vegas et NewReno ont été comparés par simulation et par analyse de trafics réels dans (Grieco et Mascolo, 2004). Les trois versions sont équitables dans le sens où aucune d'entre elles n'a tendance à s'accaparer toute la bande passante disponible, mais Vegas, du fait de son estimation basée sur le RTT, ne parvient pas à augmenter son débit autant qu'il le devrait, spécialement lorsque le réseau implique de grands délais.

1.2.5 Variantes associées à un contexte réseau particulier

Les variantes exposées dans la section précédente sont destinées à l'Internet, avec les caractéristiques de délais et pertes qui lui sont associées. Dans des réseaux spécifiques où les délais sont anormalement élevés (c'est le cas pour les communications satellites), ou bien si les pertes sont très fréquentes du fait d'un taux d'erreur binaire important sur le canal de transmission (c'est typiquement le cas des liaisons sans-fil), alors le TCP classique ne peut pas atteindre son débit optimum. Dans ce cas, les pertes et les retards ne signifient en effet pas forcément une congestion et des versions particulières de TCP ont fait leur apparition pour ces réseaux particuliers.

1.2.5.1 TCP Peach

Les transmissions par satellites sont caractérisées par un produit bande passante$\times$délai très important et par des pertes qui ne sont pas seulement imputables aux congestions mais aussi et surtout à cause du canal peu fiable. Il est alors inutile de réduire le débit puisque ces erreurs ne sont pas la conséquence d'une sur-utilisation de la bande-passante. De plus, la phase de slow-start prend trop de temps lorsque les RTT sont importants, cela conduit à une sous-utilisation du canal. TCP Peach a été développé pour combler cette lacune (Akyildiz *et al.*, 2001).

L'émetteur TCP Peach envoie des segments de faible priorité pour sonder le réseau. Comme ils sont de priorité faible, ces segments ne provoquent pas de congestion puisqu'ils seront les premiers écartés par les nœuds d' un lien déjà chargé. Cette technique est utilisée entre autres lorsqu'un segment de données a été perdu, cela permet de savoir si la perte est due à une congestion ou bien à une erreur sur un lien. TCP Peach emploie les mécanismes suivants :

– Sudden start remplace le slow-start, la montée en débit est beaucoup plus rapide qu'avec ce dernier.
– Congestion avoidance : bien que l'algorithme diffère légèrement, le résultat est similaire à celui de TCP Reno.
– Fast Retransmit
– Rapid Recovery : émission de paquets de bourrage dont les ACKs font remonter plus rapidement la fenêtre de congestion. Ces paquets ne sont pas pénalisants lorsque les pertes sont dues à des erreurs sur le canal et non pas à un manque de bande passante.
– Over Transmit : en l'absence de perte, l'émetteur peut outrepasser le contrôle de flux et émettre plus rapidement que la fenêtre de réception $rwnd$ déclarée par le récepteur ne le permet. L'objectif de cet algorithme est d'optimiser l'utilisation des ressources en bande passante qui sont très importantes dans ce type de réseau.

1.2.5.2 ATCP, TCP pour les réseaux sans-fil

Les réseaux sans fil sont caractérisés par un taux d'erreurs binaires très important. Les trames sont alors souvent erronées et doivent être retransmises par la couche 2, cela prend du temps. Finalement, si TCP est employé en couche 4 alors il y a des risques que ces délais supplémentaires conduisent à des *timeout* et TCP considère comme une perte ce qui n'en est pas une, d'où une fenêtre $cwnd$ diminuée et une sous-utilisation de la bande passante disponible. Dans le cas des réseaux ad hoc, les congestions peuvent varier selon la route choisie et donc la mobilité. Le paramétrage de la fenêtre de congestion est donc délicat, de même pour la détection de congestion (différencier les pertes dues à une congestion ou bien à une erreur binaire sur trame). C'est l'objectif de ATCP, une version de TCP spécifiquement développée pour ces réseaux sans-fil (Liu et Singh, 2001). ATCP est en réalité une couche rajoutée, intermédiaire entre le TCP (non modifié) et la couche réseau. ATCP utilise les informations remontées par ICMP (destination unreachable) et de notifications de congestion explicites (Explicit Congestion Notification, ECN) envoyées à l'émetteur pour ne tenir compte que des pertes dues aux congestions pour dérouler les algorithmes relatifs au contrôle de congestions.

1.2.6 Les protocoles « TCP-like »

Nous avons vu que lorsqu'une perte de paquet survient, le contrôle de congestion de TCP a tendance à réduire fortement le débit de sortie de la connexion, voire à le diviser par deux, tandis que lorsque la QdS offerte par le réseau est bonne, le débit augmente lentement. Un tel contrôle de débit est dit AIMD (Additive Increase, Multiplicative Decrease). Il peut osciller en dents de scie entre une valeur maximale qui correspond à l'instant où la fenêtre de congestion est à sa plus grande valeur, et un débit minimum qui correspond à l'instant suivant une perte. Cette perte provoque une division par deux de la fenêtre de congestion, puis le débit augmente de nouveau jusque sa valeur maximale avant d'être de nouveau diminué de moitié. Un tel comportement est également appelé *TCP-like*. La Fig. 1.5 rend compte des variations de la fenêtre (*cwnd*) selon le modèle présenté par Mathis dans (Mathis *et al.*, 1997) sous l'hypothèse de pertes périodiques.

Ce modèle suppose que la perte des paquets est périodique avec un taux de pertes p et un délai aller-retour constant (RTT). Une série de $1/p$ paquets consécutifs est alors acheminée avant qu'une perte ne survienne, et ainsi de suite. L'algorithme d'évitement de congestion fait alors varier la fenêtre de congestion *cwnd* en dents de scie entre les valeurs maximum W et sa

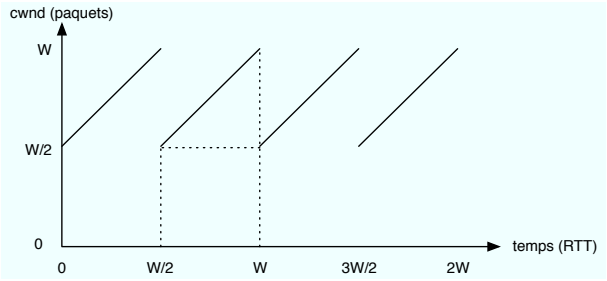


FIG. 1.5: Évolution de la fenêtre de congestion de TCP.

moitié $W/2$ paquets. Si le récepteur acquitte chaque paquet, $cwnd$ augmente alors d'un paquet à chaque RTT, d'où une période de $W/2 \times RTT$ secondes. Sur une période le nombre de paquets délivrés est alors $\left(\frac{W}{2}\right)^2 + \frac{1}{2}\left(\frac{W}{2}\right)^2 = \frac{3}{8}W^2$ paquets. D'où le calcul de W la valeur maximum de $cwnd$ et du débit moyen (noté bande passante, BW dans ce modèle), voir équations (1.1) et (1.2) respectivement. Voir (Mathis *et al.*, 1997) pour plus de détails.

$$W = \sqrt{\frac{8}{3p}} \tag{1.1}$$

$$BW = \frac{volume\ transmis\ par\ cycle}{temps\ du\ cycle} = \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \tag{1.2}$$

Nous constatons que le débit moyen du modèle fait intervenir le délai RTT et les pertes p qui sont les grandeurs de la QdS fournie par le réseau telle qu'elle est vue de bout en bout par la connexion. Le débit est inversement proportionnel au délai et à la racine carrée du taux de pertes, MSS étant le nombre d'octets transportés par un segment TCP (Maximum Segment Size) et C une constante.

Nous avons vu en section 1.2.1 que UDP est dépourvu d'un tel contrôle de congestion de bout en bout. Lorsque des communications utilisant TCP et UDP sont en compétition sur un lien, TCP a alors tendance à réduire son débit pour éviter les congestions et donc il ne peut que partager la bande passante disponible avec les autres connexions TCP. UDP n'a pas ce comportement : son débit est fixé par l'application et s'il est grand il ne restera alors qu'une petite bande passante disponible pour les flux TCP. Les congestions vont alors pénaliser les deux protocoles puisque même si le débit à l'émission d'UDP reste constant, le débit en réception (*throughput*) sera très diminué du fait des pertes. Pour pallier cette difficulté potentielle, il faut éviter de généraliser l'usage de protocoles sans contrôle de congestion pour les communications lourdes et au contraire généraliser l'utilisation de TCP ou tout autre protocole dit *TCP-friendly* ayant un contrôle de débit compatible.

Si aucun contrôle de bout en bout n'est effectué, il est aussi possible d'agir au niveau des routeurs, plus particulièrement au niveau des politiques d'ordonnancement des files d'attentes pour supprimer en priorité les paquets des flux les plus gourmands qui ne sont pas contraints par un contrôle de congestion de bout en bout, notamment avec un ordonnancement WRR (Weighted Round-Robin) plutôt que FCFS (First-come First-Served). Cette compétition déloyale est décrite par Floyd dans (Floyd et Fall, 1999). Comme nous adoptons une approche de bout en bout, nous n'étudierons pas en détail les politiques d'écartement de paquets dans les files d'attentes, ce qui imposerait l'accès aux routeurs, mais nous intégrerons un mécanisme de régulation de débit dans notre proposition au chapitre 2.

Par souci d'équité du partage de la bande passante entre les différents flux, le contrôle de congestion est donc nécessaire. Les variations brutales du débit de TCP ne sont cependant pas compatibles avec les applications qui ont un débit relativement stable, notamment les applications ayant des contraintes temporelles sévères comme la diffusion vidéo dont la qualité du rendu peut souffrir de ces variations de débit (Cen *et al.*, 1998).

D'où le développement de mécanismes de contrôle de congestion particuliers dont les réactions sont moins brutales pour les applications de diffusion de flux multimédia (*streaming*) avec par exemple le protocole Streaming Control Protocol (SCP) (Cen *et al.*, 1998). Plus généralement, c'est la notion de contrôle de débit dit *TCP-friendly* (TCP-friendly Rate Control, TFRC) qui définit un contrôle de débit à l'émission dont les variations sont plus douces que celles de TCP. Le débit TFRC est explicitement calculé par des équations en fonction des paramètres de QdS fournie par le réseau (RTT, taux de pertes). Le débit ainsi calculé correspond au débit moyen de TCP sur le long terme à paramètres de QdS équivalents. TFRC est spécifié par le RFC 3448 (Handley *et al.*, 2003) pour les communications unicast. Le contrôle de débit s'appuie sur l'équation de Floyd et Padhye dans (Padhye *et al.*, 1998) et (Floyd *et al.*, 2000), voir équation (1.3). Cette équation permet de calculer le débit à l'émission T (*send rate*) de façon inversement proportionnelle au délai aller-retour R et à la racine du taux de pertes p, tout comme dans l'équation (1.2) du modèle de Mathis. La taille en octets des paquets est notée s et t_{RTO} est la durée après laquelle le *timer* de TCP expire (timeout).

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}}p(1+32p^2)} \tag{1.3}$$

Si l'Internet « best-effort » fonctionne relativement bien malgré le très grand nombre de ses utilisateurs, c'est justement parce que la régulation de débit de bout en bout permet un équilibrage de la consommation des différents flux en ressources réseaux. Ce contrôle de débit de bout en bout ne doit donc pas être réservé aux services de transport fiables fournis par TCP. Kohler propose pour cette raison un protocole non fiable mais disposant d'un contrôle de congestion : Datagram Congestion Control Protocol (DCCP) (Kohler *et al.*, 2003). DCCP est spécifié à l'IETF (Kohler *et al.*, 2006), il intègre plusieurs contrôles de flux, notamment un contrôle TCP-like (réactions nettes aux pertes) (Floyd et Kohler, 2006), et un contrôle de débit TFRC plus doux (Floyd *et al.*, 2006). DCCP est donc utilisable lorsque la fiabilité de TCP n'est pas requise, mais aussi pour les applications qui requièrent un débit relativement stable que le contrôle de débit TFRC permet.

1.3 Le contexte multimédia

Les données numériques classiques (textes, documents, transactions ...) n'ont un sens que si elles sont parfaitement restituées au destinataire, en d'autres termes il est impératif que la transmission n'altère pas les données. Si l'on considère par exemple la transmission d'une transaction bancaire, il est évident qu'une erreur sur ne serait-ce qu'un bit peut conduire à un chiffre erroné et par conséquent à une transaction non valide. En revanche, la transmission des contenus multimédias n'a pas forcément besoin d'une parfaite restitution de tous les éléments audiovisuels au niveau binaire, mais une restitution du média acceptable qui ne soit pas dénuée de sens, *i.e.* interprétable par l'utilisateur.

La restitution de contenus multimédias conserve sa signification si certaines contraintes spécifiques à la nature du contenu sont satisfaites. Une vidéo est une séquence d'images qui doivent être jouées dans un ordre et à une cadence précis. Ces contraintes sont plus fortes

que la parfaite restitution de chaque image individuelle qui serait affichée successivement à un rythme variable. La contrainte de régularité de la cadence peut alors être satisfaite au prix de quelques erreurs sur des images individuelles, voire de la suppression de certaines images. S'il s'agit d'une transmission vidéo à la demande, les contraintes temporelles sont uniquement liées à la cadence de la restitution des images, alors que dans le cas d'une vidéoconférence une contrainte temporelle supplémentaire liée à l'interactivité apparaît. L'interactivité impose en effet une contrainte forte sur les délais séparant l'enregistrement et la restitution. Il en va de même pour des communications vocales où les contraintes temporelles liées à l'interactivité prévalent sur la parfaite restitution de chaque échantillon temporel. Pour ces communications, l'ART recommande un délai maximum de 150ms pour garantir l'interactivité (UIT-T, 2003). Dans le cas des images fixes, le sens est préservé dès lors que l'image restituée a une qualité acceptable pour l'utilisateur. Une difficulté réside dans la détermination de ce qui peut être considéré « acceptable ». Cela dépend à la fois de la nature de l'image et de l'utilisation à laquelle elle est vouée (image médicale devant permettre le diagnostic, image militaire, bandeau publicitaire, photographie), ainsi que du niveau d'exigence de l'utilisateur. Il est cependant certain que les images ont une aptitude à tolérer des pertes d'information sans que leur restitution ne soit remise en cause. Les compressions avec pertes dont elles font l'objet en sont la preuve évidente.

Les applications multimédias peuvent donc se répartir selon deux catégories :

– Les applications à forte contrainte de temps nécessitant un transport temps-réel basé par exemple sur RTP ou la protection par FEC (Forward error correction). Les applications interactives de type téléphonie ou visioconférence appartiennent à cette catégorie.
– Les applications sans contrainte de temps qui ne nécessitent pas obligatoirement un service fiable ou un transport temps réel. Ces applications peuvent alors bénéficier d'un service à fiabilité partielle basé sur un protocole fonctionnant en boucle fermée de type ARQ.

Une solution couramment employée pour la transmission temps-réel est d'utiliser le protocole *Real Time transport Protocol* (RTP) (Schulzrinne *et al.*, 2003) qui s'appuie en réalité sur UDP et rajoute un numéro de séquence et une estampille temporelle qui permettent au récepteur de jouer les informations dans le bon ordre et à la bonne cadence. Le contrôle en boucle fermé peut être effectué par *Real Time Control Protocol* (RTCP), notamment en ce qui concerne le contrôle de la qualité du service et le contrôle de flux. Cependant les pertes restent non corrigées, ce qui peut s'avérer préjudiciable lorsqu'elles portent sur des informations importantes pour la restitution.

Cette spécificité du transport multimédia a entraîné la recherche d'un compromis entre TCP et UDP, d'où la notion d'ordre et de fiabilité partiels. Dans la section suivante (1.4), un état de l'art des différents services partiellement fiables et/ou partiellement ordonnés est tout d'abord présenté. Puis, différents services de transport spécifiques à la transmission de contenus multimédias ainsi que les protocoles mis en jeu sont décrits.

1.4 Fiabilité et ordre partiels

Le transport de données sur Internet a longtemps suivi une approche tout ou rien, à travers l'utilisation des services fournis par TCP et UDP. Entre ces deux services, il existe conceptuellement tout un espace pour définir et développer de nouveaux protocoles de transport pour fournir des services à fiabilité et/ou ordre partiels. Cet espace est représenté sur la Fig. 1.6.

Les notions de fiabilité et d'ordre partiels ont été définies par Amer dans (Amer *et al.*, 1994).

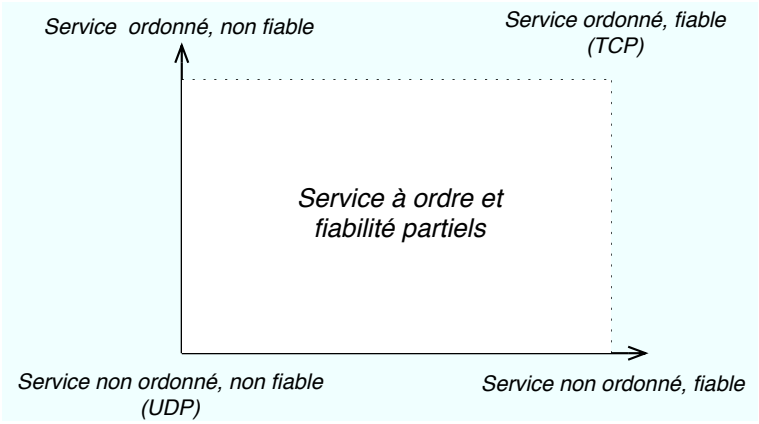


FIG. 1.6: Espace disponible pour un service à fiabilité et ordre partiels entre UDP et TCP.

Un service de transport se situe à l'intérieur de la surface dont les sommets représentent les services fiable et ordonné (TCP), fiable non ordonné, non fiable ordonné et non fiable non ordonné (UDP).

Les services non fiable non ordonné (UDP), ainsi que fiable et ordonné (TCP), ont été présentés dans les sections 1.2.1 et 1.2.2 respectivement. Un service fiable mais partiellement ordonné est adapté lorsque les applications nécessitent l'intégralité des données, mais pas forcément dans l'ordre exact de transmission. Cela suppose que les segments transmis contiennent des informations indépendantes qui peuvent être traitées dans n'importe quel ordre. Un tel service trouverait son utilité par exemple pour les télétransmissions de feuilles de sécurité sociale par les praticiens en fin de journée. Peu importe que les feuilles soient traitées dans l'ordre d'émission ou non, l'important est qu'elles le soient toutes. Les services à fiabilité partielle mais totalement ordonnés peuvent être adaptés pour les applications de type *streaming* contraintes par le temps et pour lesquelles les échantillons doivent être joués dans l'ordre, mais il est applicable plus généralement à tout contenu tolérant aux pertes. Toute combinaison des caractéristiques de fiabilité partielle et d'ordre partiel est possible et en pratique, chaque niveau de partialité peut être ajusté entre 0% et 100%.

Une taxonomie des protocoles ARQ à fiabilité partielle a été présentée par K-J. Grinnemo, A. Brunstrom et J. Garcia dans (Grinnemo *et al.*, 2004), où les différents protocoles sont classés selon le mode de décision de la fiabilité, la granularité de cette décision (message ou flux), et selon le mode de décision des retransmissions. K-J. Grinnemo *et al.* ont présenté deux classifications des services à fiabilité partielle, l'une en fonction de la définition de la fiabilité par le protocole, l'autre selon le mode de décision de la récupération des pertes.

Les principaux protocoles de transport ARQ à fiabilité partielle sont recensés dans la table 1.1. Ils seront détaillés dans la suite de ce chapitre.

1.4.1 Classification des services selon leur spécification du niveau de fiabilité

La première classification range les services selon trois critères : la spécification du niveau de fiabilité, la granularité de cette spécification et l'adaptativité. La spécification de la fiabilité est alors soit implicite, soit explicite. Elle est implicite lorsqu'elle est spécifiée indirectement en fonction de contraintes de QoS ou de transport comme par exemple le nombre de retransmissions autorisé ou un retard maximum autorisé. La spécification est explicite dans le cas où la fiabilité est décidée directement en fonction d'un niveau de pertes autorisé au niveau des

TAB. 1.1: Les principaux protocoles à fiabilité partielle, leur spécification de la fiabilité et la localisation du contrôle des retransmissions.

Protocole	*Spécification de la fiabilité*	*Correction*
POCv2	classes F/NF/PF, échéance	récepteur
HPF	classes F/NF/PF, échéance	émetteur
XUDP	classes F/NF/PF, échéance	émetteur
VDP	classes NF/PF, échéance	récepteur
SLACK ARQ	échéance	récepteur
MSP	échéance	récepteur
PR-SCTP	échéance	émetteur
TLTCP	échéance	émetteur
PECC	fenêtre glissante, échéance	récepteur
k-XP	nombre de retransmissions	émetteur
AOEC	fenêtre glissante, pertes	récepteur
SRP	estimation des pertes, délai	récepteur
PRTP-ECN	estimation des pertes	récepteur
CUDP	priorité	émetteur
SR-RTP	priorité, échéance	récepteur

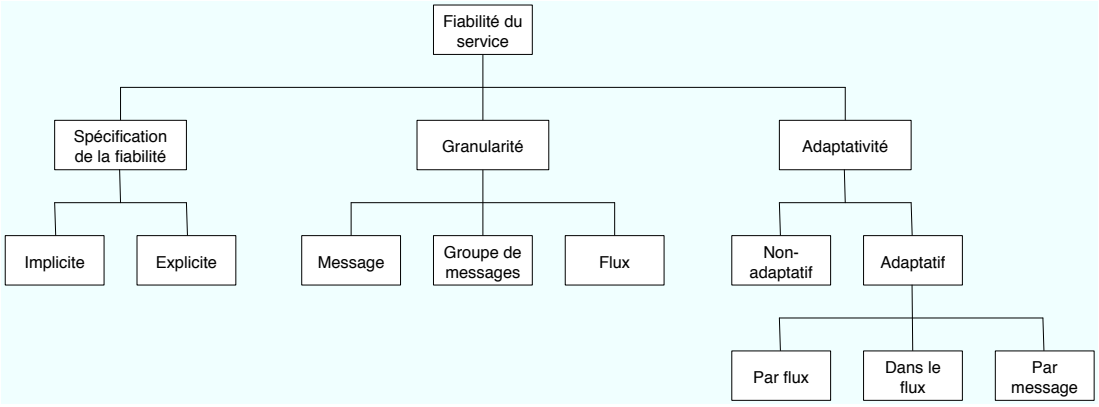


FIG. 1.7: Classification des services selon leur définition de la fiabilité.

messages, des flux ou des paquets, ou bien lorsque la fiabilité est directement marquée sur les messages/paquets/flux à l'aide de classes de fiabilité.

La granularité de la fiabilité indique quelle est la plus petite unité sur laquelle la fiabilité est contrôlée par le protocole. La figure 1.7 montre que les services peuvent être classifiés selon trois niveaux de granularité qui sont le message, le groupe de message et le flux. La spécification de la fiabilité par message signifie que la récupération vise à assurer la complétion d'un message entier quand un de ses paquets est manquant. Cette solution est appropriée pour le transport de l'audio et de la vidéo qui génèrent des blocs de données de longueurs fixes ou semblables pour lesquels l'importance — et donc la fiabilité requise — peut différer. On peut citer dans ce cas les codages vidéo MPEG-1 et 2 qui produisent des images I, P et B dont l'importance est différente. En cas de perte d'un paquet relatif à une image I, le protocole pourra corriger l'ensemble du message relatif à l'image, le niveau de fiabilité pour ce message étant élevé. Une spécification par groupe de messages indique que le niveau de fiabilité et par conséquent la retransmission éventuelle sont définis sur un ensemble de messages. Cette approche est moins fine que la précédente mais elle peut se révéler utile pour transmettre des contenus vidéo et audio si l'on considère que chaque message constitue un échantillon temporel. On peut alors décider qu'un groupe de messages correspond à un intervalle de temps supérieur à celui de l'échantillon et que la perte de $x\%$ des données est autorisée sur cet intervalle. Si le groupe est plus sévèrement altéré alors sa retransmission sera engagée. La granularité par groupe de messages est surtout intéressante lorsque les messages et groupes de messages ont une importance semblable, c'est typiquement le cas des flux produits par les codeurs audio. Une définition de la fiabilité sur le flux est évidemment moins fine mais plus simple à mettre en œuvre que les précédentes. La décision de retransmission à un instant donné est fondée sur l'ensemble des événements du flux depuis le départ, de manière à assurer la conformité du flux en accord avec le niveau de fiabilité spécifié. Ce type d'approche globale est intéressante si on se place du point de vue d'un fournisseur avec un nombre de flux important, la diffusion de streaming audio dans le cadre des application radio sur Internet par exemple. Dans ce cas, la spécification du niveau de fiabilité par flux permet de contrôler que chaque flux corresponde au niveau de fiabilité spécifié, et donc que chaque utilisateur bénéficie au moins de la QoS correspondante.

Le troisième critère de cette classification est la faculté à s'adapter. La famille des protocoles non adaptatifs sont conçus pour offrir un certain niveau de fiabilité quelle que soit la communication et ne le modifie pas au cours du temps. TCP et UDP par exemple ne sont pas adaptatifs puisque le niveau de fiabilité de leur service est fixe. Les autres protocoles sont dits adaptatifs, *i.e.* le niveau de fiabilité de leur service n'est pas pré-configuré. Le degré d'adaptation peut là encore se scinder en trois groupes : par flux, dans le flux et par message. Une adaptation par flux implique que le niveau de fiabilité soit configuré à la création du flux et ne varie pas tout au long de sa durée de vie. Le gel du niveau de fiabilité dès la création du flux est compatible avec un trafic régulier et non élastique comme celui généré par les applications audio par exemple. Lorsque le trafic est élastique et caractérisé par une importance variable selon l'instant, comme c'est le cas avec les applications de vidéo, il est nécessaire que le protocole soit plus flexible en permettant la renégociation du niveau de fiabilité. L'adaptation peut enfin être réalisée au niveau du message, cette approche est très semblable à la précédente si ce n'est que le flux est subdivisé en messages entre lesquels le niveau de fiabilité peut être renégocié.

Cette classification ne tient cependant pas compte de la manière selon laquelle les retransmissions sont décidées. C'est pourquoi K-J. Grinnemo *et al.* en ont proposé une seconde qui fait apparaître la manière dont les protocoles calculent un niveau de fiabilité et retransmettre le

cas échéant. La figure 1.8 illustre cette classification. Elle montre par exemple que la décision de retransmission peut s'effectuer par l'une ou l'autre des parties communicantes et que la retransmission peut être décidée en accord avec des mécanismes de priorité, par l'intermédiaire de métriques pour évaluer le niveau de fiabilité courant, ou encore selon des classes de fiabilité prédéterminées.

1.4.2 Classification des services selon le mode de contrôle des erreurs

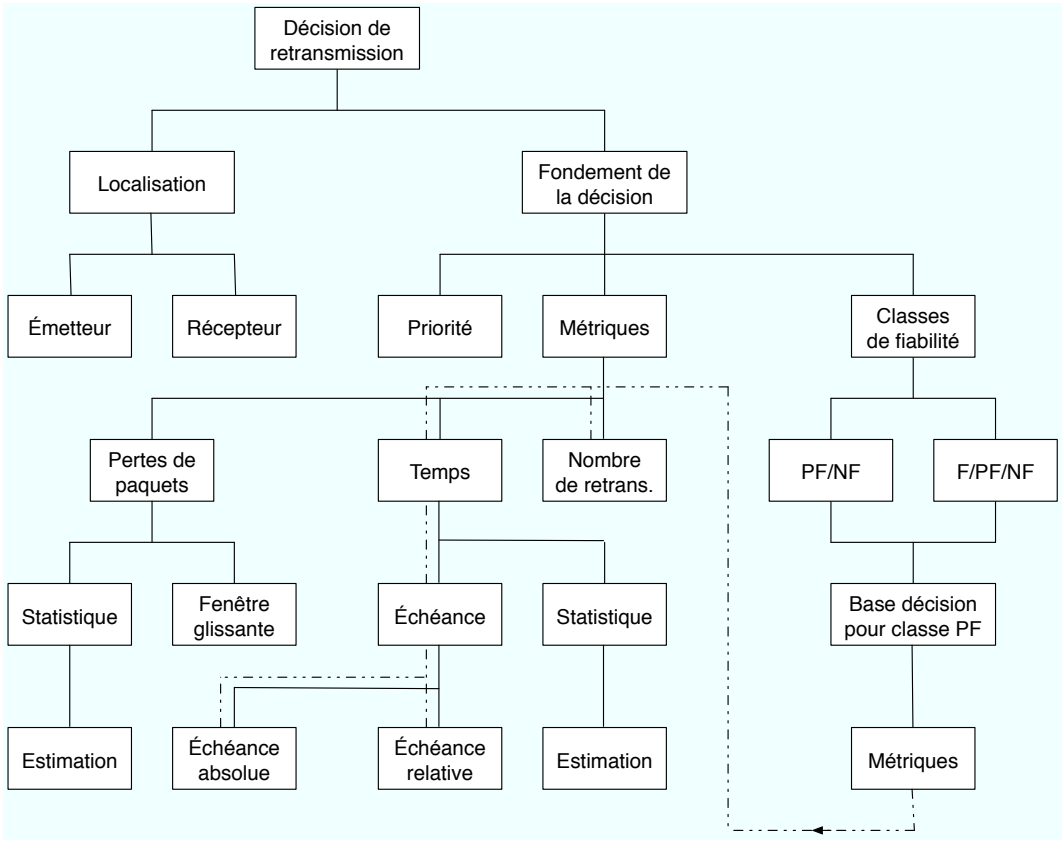


FIG. 1.8: Classification des services selon leur mode de décision pour la récupération des pertes.

L'étude du mode de décision des retransmissions révèle la manière dont les protocoles définissent et évaluent le niveau de fiabilité. Les métriques sur lesquelles la décision de retransmission est fondée sont étroitement liées aux contraintes de l'application pour laquelle le protocole a été conçu. Cette approche permet donc de faire le lien entre les contraintes de l'application, la définition du niveau de fiabilité et les mécanismes mis en jeux pour y parvenir. C'est la raison pour laquelle nous l'avons retenue pour présenter les différents services à fiabilité partielle.

La correction partielle des pertes implique leur détection. La correction des pertes peut ensuite être décidée soit par l'émetteur, soit par le récepteur. La plupart des protocoles à fiabilité partielle définit les classes de fiabilité explicites fiable (F) et non fiable (NF), et la classe partiellement fiable (PF) qui peut être explicite ou implicite selon les critères sur lesquels la retransmission se fonde. Selon le niveau de granularité de l'unité marquée par la classe, un

paquet, un groupe de paquets ou un flux sera retransmis coût que coût, *i.e.* , avec autant de retransmissions que nécessaires, lorsqu'il est marqué F. Inversement, si l'unité est marquée NF, elle ne sera transmise qu'une seule fois et dans l'hypothèse de non duplication des paquets, elle sera reçue au plus une fois. La classe PF laisse toute latitude à l'émetteur ou au récepteur pour décider si la perte doit être corrigée en fonction de métriques telles que le taux de pertes, les délais de transmissions et la date de validité des informations en question.

1.4.3 Décision de retransmission fondée sur l'échéance temporelle

La décision de retransmission sur critère temporel est pertinente pour la transmission de flux soumis à des contraintes temporelles comme c'est en le cas des flux continus audio et vidéo.

1.4.3.1 Partial Order Connection (POC)

La famille des protocoles *Partial Order Connection* (POC et POCv2) assure un service partiellement fiable dont la décision de retransmission est basée sur la validité éventuelle des retransmissions au temps où elle sont reçues. POC a été développé conjointement par l'Université du Delaware et au Laboratoire d'Analyse et d'Architecture des systèmes (LAAS) à Toulouse (Amer *et al.*, 1993) puis de nouveau à l'Université du Delaware (Conrad *et al.*, 1996) pour transmettre des flux multimédias. Les objets cruciaux de la vidéo (*e.g.* , images I d'un flux MPEG) ou plus généralement les points de synchronisation sont transportés par des paquets de classe F. Pour ces objets, la fiabilité prévaut sur toute forme de contrainte temporelle puisque sans ces objets, le média lui-même n'existe pas (impossibilité de reconstruire un flux MPEG sans image I). Les données les moins significatives pour la qualité de restitution du contenu sont de classe NF. Les données restantes peuvent améliorer la qualité de restitution mais leur acheminement ne doit pas retarder celui des informations plus importantes, elles sont donc transportées par des paquets PF. Le récepteur fait explicitement la demande de correction de ces pertes uniquement lorsque le temps pris par la retransmission ne rendra pas l'information caduque, c'est-à-dire lorsqu'il a assez d'objets antérieurs mémorisés qui peuvent être joués en attendant la correction.

1.4.4 TCP-POC

Connolly *et al.* ont proposé à l'IETF une solution expérimentale pour ajouter le support de la fiabilité partielle à TCP. Il s'agit d'une extension qui intègre le concept POC à TCP (Connolly *et al.*, 1994). Cela suppose une modification de TCP sur les deux parties (émetteur et récepteur) qui doivent définir les paramètres du service à l'établissement de la connexion. Cette proposition d'un TCP-POC définit trois classes de fiabilité :

- BART-NL (Buffer ACKs Retransmissions Timers, No Loss). Les objets de cette classe sont requis, ils doivent être mémorisés et retransmis tant qu'ils ne sont pas acquittés.
- BART-L, la perte des objets de cette classe est tolérable, néanmoins il est concevable de les retransmettre si le temps le permet (à la discrétion de l'émetteur).
- NBART-L, ces objets ne sont pas mémorisés par l'émetteur car ils ne sont en aucun cas retransmis. Leur perte est tolérable.

L'hypothétique classe NBART-NL imposerait l'utilisation de mécanismes de correction en boucle ouverte (FEC) pour garantir la fiabilité sans retransmettre. Il s'agirait alors d'un protocole hy-

bride qui sort du cadre de la correction d'erreurs par retransmissions, c'est pourquoi une telle classe n'est définie par aucun des protocoles présentés dans ce chapitre.

TCP-POC précise également que les applications peuvent définir des profils pour indiquer une table des objets transmis, en précisant la classe de chaque objet mais aussi leur tolérance relative à l'ordre. La couche transport en réception peut alors remonter les objets vers la couche application dans la mesure où les contraintes indiquées par le profil sont satisfaites. En supposant une transmission de n objets, les contraintes d'ordre sont indiquées par une matrice (n, n) où un 1 à la ligne i et à la colonne j signifie que l'objet i doit être délivré impérativement avant l'objet j, un 0 marque au contraire l'absence de contrainte. En réalité, seule la partie de la matrice située au-dessus de la diagonale est requise, ce qui porte le nombre de bits nécessaires à $n \times (n-1)/2$ pour la définition des contraintes d'ordre. La spécification de la classe de fiabilité des objets ne nécessite pour sa part qu'un vecteur de n éléments, la définition d'un tel profil n'impose pas une surcharge démesurée, du moins pour les flux comprenant un nombre restreint d'objets. Notons que cette classification porte sur des objets, ce qui suppose des frontières pour les délimiter. TCP ne définit cependant que des flux d'octets, sans possibilité de délimiter des objets tout au long de la durée de vie d'un flux. Cette subdivision du flux signifie donc une profonde modification de TCP, les mécanismes pour y parvenir ne sont toutefois pas présentés par le RFC 1693 (Connolly *et al.*, 1994).

1.4.4.1 Heterogeneous Packet Flow (HPF)

La décision de la retransmission des pertes peut aussi être localisée sur l'émetteur. c'est le cas du protocole HPF, ordonné et à fiabilité partielle (Li *et al.*, 1999). La fiabilité est décidée *a priori* par l'application d'émission, sur la base du message. L'application spécifie la classe fiable (F), non fiable (NF) ou partiellement fiable (PF) pour chaque message. La date de validité du message est également spécifiée dans le cas de la classe PF. HPF se charge ensuite d'appliquer cette classe sur tous les paquets constituant le message au niveau transport. L'émetteur décide alors de retransmettre un paquet perdu selon la classe. Les paquets marqués F sont retransmis autant de fois que nécessaire jusqu'à leur acquittement, les paquets de la classe NF ne sont jamais retransmis et les paquets PF ne le sont que si l'estimation du délai aller-retour (RTT) suggère que l'échéance temporelle sera satisfaite.

1.4.5 eXtended User Datagram Protocol (XUDP)

XUDP a été proposé dans (Andrews, 1997). Implanté dans l'espace utilisateur au-dessus de UDP plutôt que dans l'espace du noyau, XUDP est dédié aux applications multimédias. Ce protocole fournit un service partiellement fiable, orienté connexion et assure les contrôles de flux et de congestions. Contrairement à TCP, XUDP utilise une fenêtre glissante exprimée en nombre de paquets et non pas en octets. Le récepteur XUDP adopte une stratégie d'acquittements sélectifs pour les paquets et les messages, ce qui permet à l'émetteur de retransmettre les paquets efficacement de manière sélective. Les ACKs retournés sont utilisés par l'émetteur pour évaluer le RTT et comptabiliser les pertes, et donc pour réduire ou augmenter la profondeur de la fenêtre et assurer le contrôle de congestions. Le niveau de fiabilité de XUDP est encore défini selon les trois classes F/NF/PF pour les unités de données appelées *parcel* fournies par l'application. La fiabilité est donc marquée sur des messages, ces derniers peuvent avoir une taille comprise entre un octet et 32Ko qui est bien adaptée pour correspondre à une fraction de seconde voire une seconde de flux audio ou bien pour une image ou un groupe d'images. XUDP marque à l'identique la classe sur tous les paquets constituant le message.

Les paquets F perdus sont retransmis autant que nécessaire alors que les paquets NF ne le sont jamais. Les pertes PF sont quant à elles corrigées uniquement si tous les paquets du message PF peuvent être émis sans compromettre l'échéance temporelle du message. Un numéro de séquence sur les paquets permet à la couche transport réceptrice de reconstruire les messages avant de les remonter à l'application (pas forcément dans l'ordre cette fois).

1.4.5.1 PR-SCTP

R. Stewart *et al.* ont proposé dans (Stewart *et al.*, 2004) un standard pour implanter le concept de fiabilité partielle dans le protocole SCTP[2] (Stream Control Transmission Protocol). Cette extension transforme SCTP en PR-SCTP en ajoutant la possibilité d'autoriser des pertes de messages. La décision de retransmission appartient à l'émetteur qui se fonde sur la validité de l'information dans le temps pour engager les retransmissions. En pratique, chaque message est étiqueté par un numéro (Transport Sequence Number – TSN). Si un message n'a pas été acquitté dans le temps imparti et que l'émetteur juge qu'il est inopportun d'engager sa retransmission parce que ce serait trop coûteux en temps et nuirait à la cadence d'un flux vidéo par exemple, l'émetteur peut alors émettre un message appelé Forward-TSN (spécifique à PR-SCTP) pour indiquer au récepteur que ce dernier doit considérer que tous les messages antérieurs sont considérés acquittés. Ce mécanisme permet à l'émetteur de poursuivre la transmission sans retransmettre certains messages. Il permet également au récepteur de remonter les messages à la couche supérieure et libérer son buffer car aucune retransmission d'un message précédent le nouveau TSN n'est à attendre. Contrairement à HPF, PR-SCTP ne spécifie pas la fiabilité de manière figée dès le départ, mais avec plus de flexibilité. Cela revient à dire que tous les messages sont de classe FP et que l'échéance temporelle n'est pas fixée non plus puisque le protocole peut décider de retransmettre ou de sauter un message à tout moment.

Les protocoles qui basent les retransmissions des paquets PF perdus sur le respect d'une échéance temporelle se scindent en deux catégories : certains définissent une classe fiable (TCP-POC, HPF, XUDP) et d'autres non (PR-SCTP).

Lorsqu'une classe fiable est spécifiée, le récepteur disposera alors dans le pire des cas au moins des paquets de cette classe. Cela implique un certain déterminisme et permet par exemple d'assurer la transmission des informations clés comme les images I des flux MPEG sans lesquelles le flux n'a plus de sens. La classe F peut cependant être une source de retard du fait du nombre inconnu *a priori* des retransmissions qu'elle engendre. Elle peut donc provoquer des points bloquants qui peuvent nuire à la fluidité du média continu. Tout est affaire de compromis : selon que le niveau d'importance que l'on accorde à la nécessité de recevoir les informations clés d'un côté et la nécessité d'éviter autant que possible les points bloquants pour assurer la fluidité de la vidéo, la pertinence de la classe F est remise en question. C'est pourquoi ont émergé des protocoles à fiabilité partielle qui ne garantissent pas la fiabilité sur un sous-ensemble du flux. C'est par exemple le cas du protocole *Video Datagram Protocol*.

1.4.5.2 Video Datagram Protocol (VDP)

Ce protocole a été proposé par Z. Chen *et al.* dans (Chen *et al.*, 1995) spécialement pour les applications de transport vidéo et audio sur Internet. VDP ne propose plus que les classes NF et PF, la retransmission des paquets PF perdus est cette fois commandée par le récepteur

[2]SCTP est un protocole fiable et ordonné. Contrairement à TCP qui est orienté flux d'octets, SCTP transporte plusieurs flux de messages sur une connexion, comptabilise les pertes, et permet le *multihoming*.

si le temps le permet. Le fonctionnement de VDP repose sur la création de deux canaux entre le client et le serveur. Un canal fiable sert à transporter les rapports de contrôle (feedback) du client vers le serveur du média continu tandis que le canal non fiable est utilisé pour transmettre les données audio-vidéo ainsi que les rapports de moindre importance.

Le canal de contrôle fiable transporte les commandes du client telles les fonctions magnétoscope (lecture, stop, avance, retour) et de fermeture de connexion. Lorsque le client demande la lecture du média, le serveur émet les paquets relatifs aux images avec un débit qui correspond à la fréquence des images du contenu original. Le serveur émet en outre périodiquement des paquets spéciaux qui indiquent au client combien de paquets de données ont été émis depuis le début de la transmission de façon à permettre au client d'évaluer le taux de pertes. Le client peut rencontrer deux difficultés : soit il ne dispose pas des ressources suffisantes pour jouer le film avec la cadence imposée et il écarte dans ce cas des images pour garantir que les images suivantes soient jouées dans les temps, soit les pertes sont trop importantes. Dans les deux cas, le client émet un rapport au serveur via le canal de contrôle pour indiquer le taux de pertes de paquets (message de type *packet drop rate*) ou le taux d'écartement des images (*frame drop rate*). Suite à ces rapports, le serveur dégrade volontairement la transmission en écartant périodiquement des images pour limiter l'utilisation du réseau et donc les congestions (contrôle de congestions), ou pour limiter le flux reçu par le client dont les ressources sont insuffisantes (contrôle de flux). L'implémentation de VDP précise que les messages de contrôle sont émis toutes les 30 images. La dégradation ou au contraire l'amélioration du débit est décidée par le serveur lorsque les pertes sont supérieures à 15% et inférieures à 5% respectivement.

Le client VDP a aussi la possibilité d'envoyer une requête explicite de retransmission lorsque certains paquets relatifs à des données cruciales sont manquantes (*e.g.* , les images I d'un flux MPEG). La fiabilité partielle de VDP est alors commandée par le client qui est finalement le mieux placé pour savoir si son buffer d'admission est suffisamment dimensionné pour permettre la réception de l'image dans les temps. VDP est donc un protocole dédié à un média particulier, en l'occurrence les flux MPEG. Il est étroitement lié à l'application car il suppose la connaissance du codage du média pour que l'écartement des paquets à la source ainsi que les demandes expresses de retransmissions soient efficaces.

Quand les données du contenu à transmettre ont toutes la même importance, il n'y a pas de raison pour différencier leur niveau de fiabilité à l'émission. Il n'y a alors plus de notion de classe, ou plutôt tous les paquets sont de la classe FP et leur perte est corrigée uniquement si le récepteur estime que les contraintes temporelles le permettent.

1.4.5.3 Slack-ARQ

Le cas des flux audio segmentés en échantillons temporels est un exemple où les paquets sont considérés tous égaux en importance (résultats des modulations d'impulsions codées de type MIC ou PCM). Le protocole Slack ARQ a été proposé dans ce sens par Dempsey pour le transport de la voix (Dempsey *et al.*, 1993). Le client Slack-ARQ contrôle le niveau de fiabilité des paquets en effectuant des requêtes de retransmission s'il reste des échantillons à jouer avant la perte dans le buffer d'amortissement de gigue (un tampon mémoire qui délaye la lecture des échantillons reçus). Aucune perte ne fera l'objet de demande de retransmission si elle concerne un échantillon qui aurait déjà du être traité.

1.4.6 Time Lined TCP (TLTCP)

Lorsqu'au contraire, les objets du contenu multimédia sont dépendants (typiquement les images I, P et B d'un flux MPEG), tous les objets dépendants doivent avoir la même échéance temporelle. C'est l'approche présentée par Mukherjee avec le protocole TLTCP présenté dans (Mukherjee, 2000) et dans (Mukherjee et Brecht, 2000). TLTCP se base sur TCP mais ajoute la notion de validité temporelle de l'information. Il s'appuie donc sur les mécanismes de contrôle de congestion bien connus de TCP pour transporter des médias continus sous contrainte de temps. Le débit de TLTCP est par conséquent de type TCP-friendly pour rendre équitable le partage de la bande passante avec les traditionnels flux TCP concurrents tout en relâchant le niveau de fiabilité pour satisfaire des contraintes d'ordre temporel.

Pour y parvenir, TLTCP ne traite pas un flux d'octets mais un flux de messages délivrés par l'application, ces derniers sont appelés *sections* par les auteurs. En plus de fournir les données des messages et leur taille, l'application indique aussi leur date de validité. Cette estampille temporelle est relative à l'émetteur, TLTCP tente alors de corriger les pertes non encore périmées en retransmettant les paquets concernés. Notons qu'il est possible dans ce cas qu'un paquet soit retransmis avant sa date limite mais parvienne trop tard. L'échéance relative au récepteur peut alors être prise en compte par le protocole à l'aide de l'estimation du RTT.

TLTCP a été conçu pour le transport de médias continus, et plus précisément pour le transport de flux MPEG-1. TLTCP adopte dans ce cas un mécanisme particulier pour diviser le contenu en messages et leur assigner l'échéance temporelle. Les séquences vidéo MPEG sont composées d'images I, P et B. Les images I sont décodables indépendamment tandis que les images P dépendent des images I ou P précédentes, les images B dépendent en revanche des images I ou P qui l'encadrent. Dans un fichier MPEG, les images sont stockées dans l'ordre où elles ont été codées, par exemple $\{I_1^1, P_1^1, P_2^1, I_1^2, B_1^1, P_1^2, I_1^3, B_1^2 \ldots\}$ mais les images seront jouées dans un autre ordre qui serait $\{I_1^1, P_1^1, B_1^1, P_2^1, I_1^2, P_1^2, B_1^2, I_1^3 \ldots\}$. Les messages comprennent une ou plusieurs images et sont créés dans l'ordre de présence des données dans le fichier MPEG. Les frontières du messages sont dictées par un changement d'échéance temporelle entre deux images consécutives du fichier. TLTCP considère que toutes les images interdépendantes portent l'échéance temporelle de l'image qui doit être lue en premier. De cette manière et en reprenant le même flux MPEG que ci-dessus, les messages seraient formés de la manière suivante en notant d leur échéance temporelle (deadline) : $\{\{I_1^1, P_1^1, P_2^1 : d^1\}, \{I_1^2 : d^2\}, \{B_1 : d^1\}, \{P_1^2 : d^2\}, \{, I_1^3, B_1^2 : d^3\}, \ldots\}$. Les messages sont transmis dans l'ordre de leur création, les pertes étant récupérées en fonction de la valeur d^i. Lorsque l'émetteur décide de ne pas récupérer une perte il doit informer le récepteur au moyen d'un paquet qui notifie ce dernier que le numéro de séquence attendu a changé, sans quoi le récepteur rejetterait les paquets suivants en attendant une récupération impossible. Ces numéros de séquence mis à jour sont indiqués au moyen d'un champ de 32 bits appelé `seq_update` situé dans le champ option de TCP.

TLTCP définit aussi la possibilité tout comme VDP de dégrader volontairement la transmission en cas de congestion et donc de pertes. Dans ce cas les images B sont supprimées des messages, puis si cela ne suffit pas à résorber les pertes dues aux congestions, les images P sont supprimées à leur tour.

Bien que s'appuyant sur les mécanismes de TCP, TLTCP est donc dédié à un usage particulier qui est la transmission de flux MPEG. Son principal intérêt est de proposer un service avec un contrôle de débit de bout en bout de type TFRC pour le transport de médias continus alors que la majorité des services de transport multimédias ne régulent pas leur débit pour as-

surer l'équité du partage de la bande passante dans l'Internet. Ceci est d'autant plus important que le fonctionnement *best-effort* de l'Internet est justement permis par le contrôle de débit effectué de bout en bout par TCP, le principal protocole utilisé sur le web.

1.4.6.1 Media Streaming Protocol (MSP)

Le protocole MSP est dédié au transport des flux audio et vidéo. Il s'agit encore d'une variation du principe selon lequel la perte est préférable à la retransmission si l'échéance temporelle ne peut être satisfaite. Comme pour VDP, MSP permet au client de piloter le niveau de fiabilité en fonction du taux de remplissage du buffer en réception qui est réduit en cas de congestion. Le récepteur fait explicitement une demande au serveur dans le but que ce dernier cesse volontairement d'émettre certains paquets. La nuance par rapport à VDP réside dans l'objectif de cette demande qui est de privilégier le flux audio par rapport au flux vidéo pour MSP (Hess, 1998).

Toutes les propositions présentées précédemment définissent la fiabilité partielle en relation avec la validité temporelle de la retransmission. Les protocoles ayant recours à ce mode de décision sont donc essentiellement dédiés à la transmission de contenus vidéo et/ou de l'audio. Il est cependant envisageable de fonder la décision de retransmission sur d'autres métriques, notamment le nombre de retransmissions, le taux de pertes de paquets ou des mécanismes de priorités.

1.4.7 Décision de retransmission fondée sur le nombre de retransmissions

Plus le niveau de fiabilité requis est élevé, et plus il faudra retransmettre. C'est sur ce principe que *k transmit* (k_XMIT) définit une fiabilité partielle dont la décision de retransmission est prise par l'émetteur sur la base d'un nombre maximum de transmissions k pour chaque paquet (Marasli *et al.*, 1996), (Marasli, 1997). Selon cette règle, si un paquet a déjà fait l'objet de k transmissions, sa perte sera alors acceptée. Dérivé de ce principe, un protocole *k transmit protocol* (k-XP) peut simuler la notion de classes de fiabilité avec une classe NF ($k_{NF} = 1$) et la classe F ($k_F = \infty$).

1.4.8 Décision de retransmission fondée sur le taux de perte des paquets ou sur la combinaison des pertes

Bien que tolérables lorsqu'elles sont considérées individuellement, les pertes peuvent devenir gênantes quand elles sont trop proches temporellement.

1.4.8.1 Application-Oriented Error Control (AOEC)

AOEC a été proposé par Gong et Parulkar dans (Gong et Parulkar, 1992). Ce protocole définit le niveau de fiabilité et permet certaines pertes de paquets uniquement en comptabilisant les pertes et leur situation. Le niveau de fiabilité à satisfaire est exprimé par un triplet (W, E, B). Cette spécification autorise au plus une série de E pertes sur une fenêtre glissante de W paquets et une rafale de pertes B (burst) sur cette même fenêtre.

1.4.8.2 Partially Error-Controlled Connection (PECC)

Fonctionnant sur un principe anologue de contrôle des pertes sur une fenêtre glissante, le contrôle d'erreur PECC a été proposé par Dempsey (Dempsey *et al.*, 1992). À la différence de AOEC, PECC exprime la fenêtre et les rafales de pertes en octets, et le récepteur est soumis à la condition d'un seuil minimal de remplissage de buffer en deçà duquel il ne peut pas demander la retransmission. Implicitement, cela revient à fonder la décision de retransmission sur un critère d'échéance temporelle.

AOEC et PECC mesurent directement les pertes à l'intérieur d'une fenêtre glissante. Cette métrique est ensuite directement exploitée pour définir si le critère de fiabilité est atteint et si la retransmission est nécessaire ou non. Si l'objectif est de maintenir un niveau de qualité suffisant pour l'application, le critère peut alors prendre une forme plus évoluée avec un calcul statistique pour estimer le niveau courant de qualité. La décision de retransmission se base alors sur des équations.

1.4.8.3 Selective Retransmission Protocol (SRP)

Contrairement à PECC qui retransmet d'abord si le temps le permet et ensuite si les pertes ne sont pas acceptables, le protocole SRP proposé par Piecuch et al. (Piecuch *et al.*, 2000) met en balance ces deux métriques pour élaborer un critère de retransmission. Le récepteur évalue le délai à la moitié du RTT qu'il mesure avec des paquets sondes. Les pertes sont comptabilisées lorsqu'un paquet n'est pas arrivé à temps. Le récepteur évalue le quotient noté r_{loss} du taux de pertes courant sur le taux de pertes maximum admissible ainsi que le rapport du délai de transmission courant sur le délai maximum tolérable noté r_{delay}. Deux algorithmes de décision de retransmission sont implantés au récepteur dans le but d'établir l'équilibre entre les pertes et les délais nécessaire aux applications multimédias :

- *Equal Loss Latency* (ELL) qui met directement en balance le temps perdu par l'éventuelle retransmission et l'amélioration du taux de pertes. ELL tente de minimiser l'équation 1.4.
- *Optimum Quality* (OQ) qui tente de minimiser le critère quadratique de l'équation 1.5.

$$\Delta = |r_{loss} - r_{delay}| \quad (1.4)$$

$$\Delta = r_{loss}^2 + r_{delay}^2 \quad (1.5)$$

1.4.8.4 (PRTP-ECN)

Pour les applications plus faiblement contraintes par le temps (temps-réel mou), Grinnemo et Brunstrom ont proposé PRTP-ECN (partially reliable transport protocol TCP compliant), une extension de TCP qui réduit le nombre de retransmissions (Grinnemo et Brunstrom, 2001). PRTP-ECN ne modifie que le client TCP (le récepteur), ce dernier peut décider de ne pas récupérer certaines pertes en les acquittant quand même, ce qui finalement dupe l'émetteur TCP qui continue à émettre la suite comme si rien ne s'était passé. Pour que l'émetteur puisse tout de même appliquer le contrôle de congestion en cas de perte non déclarée, le récepteur utilise les drapeaux ECN pour signaler la congestion à la source. PRTP-ECN définit le niveau courant de fiabilité du service en calculant un critère en permanence. Ce critère, appelé $crl(n)$, est évalué sur l'historique des n derniers paquets précédant la perte. L'aspect pénalisant des pertes peut être finement réglé par des coefficients de pondération, voir équation 1.6. Le paramètre af pondère plus ou moins l'impact des pertes récentes, p_k vaut 1 lorsque le paquet k

est reçu, 0 sinon et b_k précise la quantité de données en octets du paquet k.

$$crl(n) = \frac{\sum_{k=0}^{n-1} af^k \times p_k \times b_k}{\sum_{k=0}^{n-1} af^k \times b_k} \tag{1.6}$$

Le critère est ensuite comparé au niveau de fiabilité requis rrl. Pour demander la retransmission, le récepteur acquitte comme le fait normalement TCP lorsque $crl(n) < rrl$. Dans le cas contraire, le récepteur trompe l'émetteur en acquittant quand même le segment perdu et la perte n'est pas récupérée.

1.4.9 Décision de retransmission fondée sur des priorités

Pour cette famille de protocoles, chaque paquet se voit attribuer une priorité en fonction du niveau d'importance des données qu'il transporte. En cas de pertes, les paquets les plus prioritaires sont retransmis d'abord. Le mécanisme de priorités est souvent utilisé en conjonction avec un critère temporel. Si le temps le permet (taux de remplissage du buffer en réception) alors les pertes sont récupérées dans l'ordre des priorités, mais si le temps fait défaut il n'y a pas retransmission. Finalement les paquets les plus prioritaires ont donc une meilleure probabilité d'être acheminés. Ce type de protocole est typiquement destiné au transport de flux continus pour lesquels RTP est généralement utilisé.

1.4.9.1 Cyclic UDP (CUDP)

CUDP a été proposé par Brian Smith dans (Smith, 1994) pour assurer le transport dans le cadre des applications audio-vidéo, notamment pour les flux MPEG et motion JPEG. CUDP émet les paquets dans l'ordre des priorités décroissantes. Dans le cas d'un flux MJPEG (séquences vidéo constituées d'images JPEG, qui peuvent donc être traitées indépendamment), CUDP le divise en cycles de x images, qui sont numérotées de 0 à $(x-1)$. Supposons un cycle de 15 images, numérotées de 0000 à 1110 en binaire. La priorité est alors affectée à chaque image selon l'ordre binaire inverse (IBO) comme le montre le tableau 1.2. Les images sont donc ordonnées dans le buffer d'émission dans l'ordre $\{0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7\}$ et sont transmises dans cet ordre. Lorsque les congestions apparaissent et que plusieurs paquets consécutifs sont perdus, ce mécanisme de priorité évite que les pertes se traduisent par un groupe d'images manquantes. Tout se passe comme si la cadence avait diminué, avec une image passée de temps en temps.

Avec CUDP, les paquets ne sont pas acquittés, par contre le récepteur émet un acquittement négatif pour demander la retransmission des pertes lorsqu'elles sont détectées à la réception de paquets hors séquences. L'émetteur retransmet ces paquets tant qu'il n'a pas changé de cycle. Les paquets les plus prioritaires sont donc ceux qui sont émis en début de cycle, ce qui donne au récepteur plus de temps pour détecter leur perte et demander leur retransmission. De son côté, l'émetteur peut les retransmettre un nombre de fois plus important que les paquets moins prioritaires qui peuvent suivre.

1.4.9.2 Selective Reliability RTP (SR-RTP)

SR-RTP a été conçu pour le transport de vidéo MPEG sur l'Internet. Par rapport à RTP, il peut s'accommoder des variations de délais, de bande passante et des pertes tout en disposant d'un contrôle de débit TCP-friendly et de la fiabilité partielle définie à l'aide d'un mécanisme de priorités (Feamster, 2001).

TAB. 1.2: Attribution des priorités selon l'ordre binaire inverse IBO.

Numéro image	Ordre binaire inverse	Priorité
0	0000	0
1	1000	8
2	0100	4
3	1100	12
4	0010	2
5	1010	10
6	0110	6
7	1110	14
8	0001	1
9	1001	9
10	0101	5
11	1101	13
12	0011	3
13	1011	11
14	0111	7

Une ADU correspond à une image que SR-RTP découpe en paquets. Par rapport à RTP, SR-RTP permet au récepteur de détecter les pertes dans une ADU à l'aide des informations additionnelles présentes dans l'en-tête (voir figure 1.9). Le numéro d'ADU, sa longueur et l'offset du fragment au sein de celle-ci sont suffisants pour détecter les pertes complètes d'une ADU aussi bien que la perte de fragments quelles que soient leurs situations. La priorité est marquée selon l'importance des données transmises, les paquets relatifs aux images I ayant la plus haute priorité. Le récepteur pilote les retransmissions par envoi d'acquittements négatifs lorsque la priorité des données manquantes est supérieure au seuil défini par la politique de fiabilité choisie par l'utilisateur. La politique de fiabilité peut par exemple requérir systématiquement les corrections relatives aux images I et uniquement les images I, ou bien demander aussi la correction des images P qui leur succèdent immédiatement. Les pertes totales d'ADU ne sont pas corrigées car elles seraient trop consommatrices de temps. Une perte de tous les fragments d'une ADU n'est en général pas préjudiciable car étant donnée la taille conséquente des images I devant celle des autres, la probabilité que tous ses fragments soient perdus est par conséquent très faible.

Ces services de transport partiellement fiables et leur spécification de la fiabilité sont récapitulés dans la table 1.1. Pour la plupart, ils sont destinés au transport des flux continus caractérisés par des contraintes de type temps-réel, notamment l'audio et/ou la vidéo. Par opposition à la vidéo, le transport des images fixes n'est en général pas contraint par le temps. Cependant ce type de transmission fait l'objet d'investigations particulières pour proposer des alternatives partiellement fiables à TCP.

1.5 Les services dédiés au transport d'images fixes

Le trafic relatif aux images JPEG qui transitent sur l'Internet était déjà considérable en 1997 : 31% du volume total et 16% du nombre total de fichiers échangés d'après une analyse

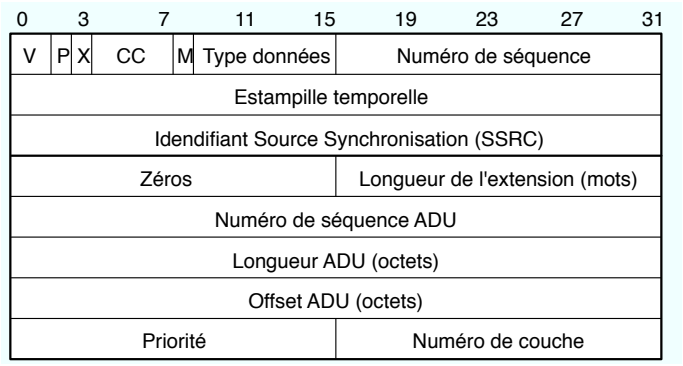


FIG. 1.9: L'en-tête SR-RTP permet la fiabilité partielle.

de trace d'un client à l'Université de Berkeley (Gribble et Brewer, 1997). Actuellement, la grande majorité des images échangées sur l'Internet est constituée par les agréments graphiques des pages web (boutons et autres bandeaux), la publicité sans cesse croissante, ainsi que la publication de photographies que les accès hauts débits et les appareils photo numériques ont démocratisée. Il s'agit donc de trafic HTTP (Fielding *et al.*, 1997), qui repose sur le transport fiable et ordonné de TCP. Cette solution est facile à mettre en oeuvre, cela permet à la destination de disposer du contenu intégral des fichiers des images compressées et donc de les reconstruire comme si ces ressources étaient locales.

1.5.1 Ordre partiel et progressivité de l'affichage

Lorsque le réseau est sujet à des pertes de paquets, le caractère ordonné de TCP entraîne un délai d'attente avant de remonter les unités de données à l'application (*Application Data Unit*, ADU) qui suivent une perte jusqu'à ce que celle-ci soit corrigée. Ce délai empêche l'application de traiter les données de l'image au fur et à mesure qu'elles sont reçues. Or les données contenues dans un segment reçu hors séquence peuvent éventuellement être traitées voire être affichées sans attendre par l'application si les informations peuvent être traitées indépendamment. Cette approche a été fréquemment traitée, son objectif est un affichage progressif que permet le traitement des données par l'application immédiatement après réception. Pour y parvenir, elle s'appuie sur la notion d'ordre partiel et doit redéfinir le schéma de compression de l'image en ayant à l'esprit que le produit sera transmis par l'intermédiaire de paquets, *i.e.* , le codage doit produire des unités décodables indépendamment suffisamment petites pour être transportées chacune dans un paquet.

1.5.1.1 GIF-Network Conscious (GIF-NC)

Cette approche a été menée avec succès par Amer dans (Amer *et al.*, 1999) avec la spécification d'une compression GIF-NC dite « consciente du réseau » (*network conscious*, NC). Avec cette approche, l'image n'est pas compressée pour produire le plus petit fichier possible pour une qualité de reconstruction donnée en vue du stockage ou de l'archivage. Cette compression a vocation d'optimiser la progressivité de l'affichage de l'image par le récepteur au fur et à mesure que les paquets lui parviennent. Le transport utilisé est fiable mais non ordonné, ce qui suppose que chaque segment remonté forme une ADU exploitable indépendamment par l'application et donc l'emploi du concept d'application level framing (ALF) (Clark et Tennenhouse, 1990). Habituellement les images codées selon le standard GIF89a sont transportées

par TCP, mais par souci d'équité, Amer *et al.* ont comparé le transport de GIF-NC avec celui du standard GIF89a en utilisant un transport fiable, non ordonné et sans contrôle de congestions dans les deux cas. Lorsque le réseau n'est pas sujet à des pertes de paquets, l'image GIF-89a est affichée plus rapidement que l'image GIF-NC, cela s'explique par la surcharge de GIF-NCa par rapport à GIF-89a. Par contre, dès que les pertes apparaissent, l'image GIF-NC montre un affichage plus complet que GIF89a à temps de service égal : avec 5% de pertes de paquets, l'image GIF89a est affichée en moyenne à 16%, 57%, 95%, 99% et 100% au temps 5, 10, 15, 20 et 25 secondes respectivement. GIF-NC permet un affichage des images à 34%, 67%, 96%, 100% et 100% dans les mêmes conditions. En présence de pertes, l'affichage partiel de GIF-NC est donc plus rapide. Par contre, le temps nécessaire à la transmission complète de l'image et donc à son affichage complet n'est pas réduit par l'emploi d'un transport non ordonné.

Pour chaque schéma de compression d'images, cette approche impose de modifier le standard de compression pour obtenir les qualités requises pour la taille des ADUs et la possibilité de traitement indépendant. Ce travail a été effectué pour le schéma de compression *Set Partitioning in Hierarchical Trees* (SPIHT) dans (Iren et Amer, 2000) avec les mêmes qualités de progressivité que GIF-NC.

1.5.1.2 Image Transport Protocol (ITP)

Le même principe a été utilisé pour le transport des images JPEG avec la proposition du protocole *Image Transport Protocol* (ITP) dans (Raman *et al.*, 2000). ITP ne modifie pas le standard de compression JPEG, mais il n'en fait pas abstraction pour autant puisqu'il utilise les propriétés de codage en blocs de JPEG et conserve le principe ALF. Chaque ADU correspond à une zone spatiale de l'image, c'est-à-dire un certain nombre de blocs qui forment une bande. Chaque bande est transmise par un paquet qui identifie de quelle ADU il s'agit. L'ordre partiel autorise l'application à afficher les bandes dans le désordre plus progressivement que si elle devait attendre et afficher l'image dans l'ordre (de haut en bas). Lorsqu'un paquet est perdu, l'application peut continuer d'afficher les bandes suivantes avant que la perte ne soit corrigée. ITP intègre un raffinement qui vise à dissimuler temporairement les bandes non affichées du fait des pertes grâce à un mécanisme de dissimulation d'erreur (*error concealment*). Ce mécanisme recrée les pixels de la bande manquante en fonction des bandes adjacentes. Cette dissimulation est ensuite remplacée à partir des vraies données du paquet perdu corrigé mais il est tout à fait possible d'imaginer que les pertes ne soient pas toutes corrigées et que la dissimulation d'erreurs soit permanente, typiquement avec un service de transport partiellement fiable.

Nous remarquons que toutes ces approches lient étroitement la compression de l'image et le transport. Dès lors que le transport n'est pas fiable et/ou pas ordonné, il n'est en effet pas possible de faire abstraction de la compression de l'image. L'explication vient de l'objectif même de la compression d'image avec pertes. Quel qu'il soit, tout schéma de compression est conçu dans l'optique d'obtenir une suite de bits la plus courte possible tout en permettant une reconstruction d'image avec la plus grande fidélité possible après décodage. La compression d'image avec pertes est avant tout prévue pour produire des flux de bits qui sont stockés dans des fichiers les plus légers possibles. Que ce soit en local ou à distance via TCP, le décodeur dispose alors de l'intégrité des données et dans le bon ordre. Dans le cas contraire, rien n'indique que le fichier tronqué ou désordonné puisse être décodé. Les modifications des données d'images compressées conduisent d'ailleurs en général à un fichier invalide, corrompu qui prévient en l'état toute reconstruction future. Il s'agit d'un problème récurrent que nous

exposerons au chapitre 3 dans le cas du standard de compression JPEG2000.

Conclusion

Nous avons vu que TCP ralentit la mise à disposition des données à la couche application du fait de l'ordre total. L'ordre partiel permet d'éliminer cette attente mais si le service reste fiable alors le temps de réponse du service *i.e.* , le temps nécessaire à la transmission de la totalité de l'image est comparable à ce que permet TCP. Notre objectif n'est pas l'affichage progressif mais le gain de temps sur la durée de transmission des images. Nous savons que les images peuvent tolérer des pertes contrairement aux données classiques. Cette propriété n'est pourtant pas mise à profit lorsque le service de transport générique fourni par TCP est utilisé. L'emploi de la fiabilité partielle dans le but de transmettre plus rapidement les images compressées a déjà été traité dans (Danskin *et al.*, 1995). Ce protocole repose sur un mécanisme de type FEC avec une redondance d'informations qui augmente systématiquement la charge du réseau. Cette approche n'est pas très efficace lorsque les pertes du réseau sont dues à des congestions car plus on ajoute de paquets, plus on provoque de pertes.

Si nous désirons réduire le temps de service du transport des images, c'est aussi pour réduire son utilisation et sa charge. Nous écartons donc les solutions de type FEC et optons pour un protocole de type ARQ. La solution ARQ (fonctionnement en boucle fermée, avec acquittements et retransmissions) est en effet moins onéreuse en termes de rendement du protocole, tout en étant bien adaptée en l'absence de contrainte temporelle de type temps-réel.

En section 1.4, nous avons vu que la fiabilité partielle pouvait être spécifiée sur la base d'un délai, d'un taux de perte ou de rafales de pertes admissibles. La transmission des images fixes n'est pas contrainte par le temps, et la décision de retransmission est liée à l'importance de l'information du point de vue de la qualité de restitution, importance qui peut être connue *a priori*. Pour cette raison, nous proposons dans la suite de ce document 2CP-ARQ, un service de transport semblable à POC, qui s'appuie donc sur des classes de paquets. De façon à ne pas perturber le fonctionnement *best-effort* de l'Internet, 2CP-ARQ doit en outre être équitable avec TCP, c'est-à-dire qu'il doit intégrer un contrôle de congestion. La présentation de 2CP-ARQ, ainsi que son analyse, font l'objet du chapitre 2.

Notre proposition doit également permettre la garantie d'un niveau minimal de la qualité des images qu'il transporte dans le pire des cas. Cette garantie impose d'une part que l'image doit pouvoir être reconstruite en toute circonstance, malgré les pertes, et d'autre part que cette reconstruction ait le niveau de qualité requis. Les données de l'image sont transportées par des paquets dont certains sont fiables, donc dans le pire des cas le récepteur dispose uniquement des données transportées par ces segments. La reconstruction de l'image à partir de ces seules données doit être rendue possible et doit correspondre au niveau de qualité minimal requis par l'application. Cela suppose soit la définition d'une interface entre un schéma de compression existant et la mise en paquets de l'information, soit la définition d'un mécanisme de compression d'images qui tienne compte de la transmission avec pertes sur le réseau. La première approche impose une étude du standard de compression considéré de façon à mettre en paquets l'information et à attribuer la classe de fiabilité à ces paquets tandis que la seconde rejoint le concept de *network conscious* présenté par Amer avec une modification de la compression GIF dans (Amer *et al.*, 1999). Cette mise en paquets doit bien sûr conserver la pertinence et la décodabilité des données transportées même lorsque d'autres informations viennent à manquer. Nous détaillerons la première approche pour le standard de compression JPEG2000, puis la seconde pour un schéma de compression fondé sur la transformée en on-

delettes discrètes et la quantification vectorielle algébrique, respectivement dans les chapitres 3 et 4.

Chapitre 2

2CP-ARQ : un service de transport à deux classes de fiabilité

Introduction

Le transport des images fixes est souvent considéré sans contrainte de temps, le protocole de transport employé est alors naturellement TCP. Dans ce cas la qualité d'image prime sur le temps de service. Dans ce chapitre, nous proposons 2CP-ARQ, un service de transport partiellement fiable et totalement ordonné dont la vocation est le transport de contenus tolérant certaines pertes. L'objectif de ce service est de réduire le temps de service comparativement à un service totalement fiable lorsque le réseau perd des paquets, non pas pour satisfaire des contraintes temporelles de l'application, mais pour solliciter moins fortement le réseau. La fiabilité partielle permet d'y parvenir en limitant les retransmissions consécutives à la perte des paquets contenant les informations les moins préjudiciables pour l'application. Si le critère de performance ne repose plus entièrement sur la qualité de restitution des images mais aussi sur le temps de service, 2CP-ARQ peut alors optimiser le compromis entre la réduction du temps de transport et la dégradation de la reconstruction des images impliqués par la fiabilité partielle. Cette analyse du transport d'images par 2CP-ARQ, notamment en termes de gain de temps et de la dégradation de qualité d'image, a fait l'objet de publications notamment aux Journées Doctorales d'Informatiques et Réseaux (JDIR) (Holl *et al.*, 2004) et au Colloque Francophone sur l'Ingénierie des Protocoles (CFIP) (Holl *et al.*, 2005).

Ce chapitre est organisé avec le plan suivant : 2CP-ARQ est présenté en section 2.1. Son modèle analytique est introduit dans la section 2.2 puis développé en 2.3. Ce modèle a pour but l'évaluation des performances de 2CP-ARQ, notamment au niveau du gain de temps potentiel qu'il permet par rapport à un service fiable. Cette évaluation fait l'objet de la section 2.4.

2.1 Présentation du protocole 2CP-ARQ

2CP-ARQ est un protocole de type *Automatic Repeat reQuest* (ARQ) qui fournit un service de transport à fiabilité partielle pour les images fixes. Il a été défini en considérant une approche Application Level Framing (ALF) (Clark et Tennenhouse, 1990), c'est-à-dire en faisant l'hypothèse que l'application découpe les images en unités de traitement indépendantes, les ADUs (Application Data Unit), qui forment aussi les unités de transport. Cette condition est réalisée en limitant la taille des ADUs de manière qu'elles puissent, chacune, être embarquées

entièrement dans un paquet et acheminées à destination sans subir de fragmentation.

Dans cette hypothèse, la qualité de service associée aux données peut être spécifiée finement sur chaque paquet. Le protocole 2CP-ARQ distingue deux classes de paquets :

- les paquets primaires (notés p-paquets) : ces paquets sont indispensables pour pouvoir reconstruire l'image avec la qualité minimum requise. Ils doivent être délivrés à tout prix. Les pertes de p-paquets sont donc toujours corrigées, quel que soit le nombre de retransmissions nécessaires.
- les paquets secondaires (notés s-paquets) : ces paquets sont optionnels car ils apportent un certain raffinement de la qualité d'image, mais leur perte est tolérée par l'application. Lorsqu'un s-paquet est manquant, le transfert peut continuer comme si de rien n'était.

La classification des paquets est le rôle de l'application côté émetteur, qui va mettre en œuvre une politique de fiabilité définie par l'utilisateur. Précisons que la classe d'un paquet est spécifiée une fois pour toute (il n'y a pas de classe « flottante », et un paquet ne peut changer de classe entre deux retransmissions). La classification des paquets va donc dépendre exclusivement des préférences de l'utilisateur, du type de codage des images, et du MTU du réseau. Pour que l'indépendance entre l'application et le système de transport soit préservée, la classification des paquets sera effectuée à la volée. Le séquencement des p- et s-paquets n'est donc pas connu *a priori*. Comme l'engagement d'une retransmission n'est pas systématique pour toutes les pertes de paquets, émetteur et récepteur doivent être capables d'identifier sans ambiguïté la classe des paquets perdus pour interfonctionner correctement. Cela ne pose pas de difficulté du côté de l'émetteur qui prend connaissance de la classe des paquets au moment où l'application sollicite une requête de service. Par contre, le récepteur est lui complètement aveugle car il ne sait jamais d'avance le type de paquet que l'émetteur va envoyer. Le protocole 2CP-ARQ résout ce problème en marquant chaque paquet de sa classe. Cette marque a la propriété d'être persistante puisqu'elle sera reconnaissable par le récepteur en l'absence du paquet lui-même. Les principes de ce système de marquage des paquets sont développés dans la section ci-dessous.

2.1.1 Un schéma original pour le marquage de la fiabilité

La classe de fiabilité d'un paquet, p- ou s-, est marquée directement sur les paquets par l'émetteur. Selon la spécification des ADUs : requises ou optionnelles, 2CP-ARQ définit deux types de paquets.

2.1.1.1 Marquage de la classe à l'émetteur

La marque de la classe de fiabilité est produite sur chaque paquet au moyen d'un couple de compteurs (i, j). Le compteur i trace l'évolution de la séquence des p-paquets tandis que le compteur j trace celle des s-paquets. Ces deux compteurs sont utilisés à des fins de reconnaissance de la classe des paquets par le récepteur, ils ne se substituent pas au numéro de séquence classique. La manière dont évolue le couple (i, j) respecte les cinq règles suivantes :

i i et j sont des compteurs cycliques, tels que $0 \leqslant i < k$ et $0 \leqslant j < p$, où (k, p) est la valeur maximum de la plage de numérotation (*e.g.,* $k = p = 2^{16}$ en pratique).

ii la classe du paquet est indiquée par le compteur j, $j = 0$ désigne un p-paquet, tandis que $j \neq 0$ marque un s-paquet.

iii les deux compteurs i et j sont initialisés à zéro à l'établissement de la communication.

iv lorsque l'application requiert la transmission fiable d'une ADU, le p-paquet qui la transporte est marqué par l'incrémentation de i et la remise à zéro de j. En supposant que le dernier paquet porte la marque (i, j) et que le prochain soit un p-paquet, alors il sera numéroté $(i + 1, 0)$ si $i + 1 < k$, ou $(0, 0)$ si $i + 1 = k$.

v lorsque l'application requiert la transmission d'une ADU sans garantie, le s-paquet correspondant sera marqué par l'incrémentation du compteur j. En supposant que le dernier paquet est numéroté (i, j) et que le prochain soit un s-paquet, alors il portera la marque $(i, j + 1)$ si $j + 1 < p$, mais $(i + 1, 0)$ si $j + 1 = p$ et $i + 1 < k$, voire $(0, 0)$ si $j + 1 = p$ et $i + 1 = k$. Dans ces deux derniers cas la remise à zéro du compteur j peut prêter à confusion en faisant passer un s-paquet pour un p-paquet. Plus les valeurs de k et p sont élevées, et moins cette situation apparaît. Bien que possible, le déguisement d'un s-paquet en p-paquet reste néanmoins marginal et ne constitue pas un problème, finalement c'est la substitution inverse qui serait rédhibitoire.

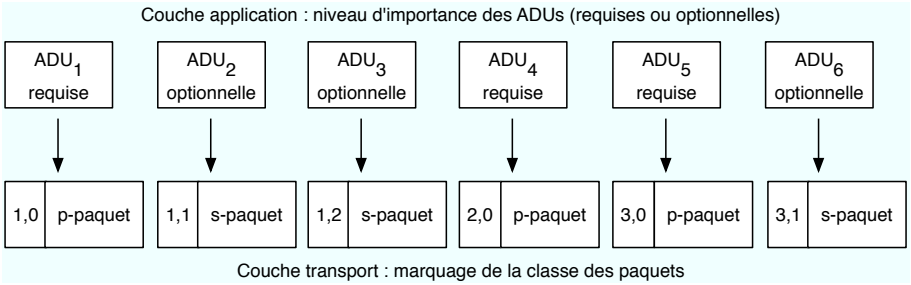


FIG. 2.1: Exemple de marquage de la fiabilité des paquets à l'aide du couple (i, j). La classe fiable impose $j = 0$ pour tout p-paquet, autrement il s'agit d'un s-paquet.

Le fonctionnement pratique de ce système de numérotation est présenté sur la figure 2.1, sur l'exemple d'une séquence de six ADUs. On estime que la communication vient d'être établie et que les compteurs ont été initialisés à 0 (règle 3). Considérons que la première ADU a été classifiée comme primaire (requise) par l'application. Elle devra donc être transportée sous la forme d'un p-paquet. La numérotation du paquet est ici déterminée par la règle 4 : incrémentation du compteur i et remise à zéro de j, ce qui donne le couple $(1, 0)$. Regardons maintenant la deuxième ADU classifiée optionnelle. La numérotation du paquet fait ici intervenir la règle 5 selon laquelle le compteur j est incrémenté et i est inchangé, ce qui donne le couple $(1, 1)$. La troisième ADU est encore optionnelle, la numérotation du paquet correspondant suit la même règle 5, ce qui produit le couple $(1, 2)$. Les ADUs 4 et 5 sont requises, le marquage des p-paquets correspondants est dicté par la règle 4 (incrémentation de i et remise à zéro de j), leurs numérotations respectives sont donc $(2, 0)$ $(3, 0)$ respectivement. La sixième ADU est optionnelle, d'après la règle 5 sa numérotation est logiquement $(3, 1)$. L'algorithme général du marquage des paquets est donné dans la figure 2.2.

2.1.1.2 Identification de la classe des paquets par le récepteur

Le récepteur ne dispose bien évidemment pas des couples de numérotation des paquets manquants, par contre le marquage des paquets reçus est connu. Ce sont les trous dans l'évolution du compteur i qui indiquent la perte des p-paquets au récepteur. La perte d'un p-paquet implique forcément un saut dans l'évolution du compteur i de part et d'autre du p-paquet

```
Initialisation :
  prec_i ← 0
  prec_j ← 0
pour chaque paquet
  si le paquet est primaire
    courant_j ← 0
    si (prec_j + 1 < borne_k)
      courant_i ← prec_i + 1
    sinon
      courant_i ← 0
  sinon si (prec_j + 1) < borne_p
          courant_j ← prec_j + 1
        sinon
          courant_j ← 0
          si (prec_i + 1) < borne_k
            courant_i ← prec_i + 1
          sinon
            courant_i ← 0
```

Variables mémorisées :

prec_i : valeur du compteur i du dernier paquet traité.
prec_j : valeur du compteur j du dernier paquet traité.
courant_i : valeur du compteur i calculée pour le paquet courant.
courant_j : valeur du compteur j calculée pour le paquet courant.

constantes :

borne_k : valeur maximum du compteur cyclique i.
borne_p : valeur maximum du compteur cyclique j.

FIG. 2.2: Algorithme du marquage de la classe des paquets par l'émetteur.

perdu, et réciproquement, un trou dans la séquence du compteur i signifie immanquablement qu'un p-paquet a été perdu. Toute perte de p-paquet est donc détectable par le récepteur. Inversement, un trou dans l'évolution du compteur j signifie la perte d'un s-paquet. Ce principe est illustré par un exemple en Fig. 2.3. Lorsqu'aucun paquet n'est perdu, le récepteur ne détecte aucun trou dans le marquage des paquets et tous les paquets sont délivrés à l'application (voir Fig. 2.3(a)). La figure 2.3(b) illustre la perte d'un s-paquet suivie de la réception du s-paquet suivant. Dans ce cas de figure, le récepteur détecte un trou dans la séquence des compteurs j et par conséquent la perte du s-paquet $(1, 1)$. Si par contre un ou plusieurs s-paquets sont perdus alors que le paquet suivant est un p-paquet, la remise à zéro du compteur j du p-paquet empêche le récepteur de détecter un trou dans la numérotation j et donc la détection des s-paquets perdus n'est pas possible. Ce cas de figure est illustré sur la figure 2.3(c), le récepteur n'est pas en mesure de détecter la perte des s-paquets $(1, 1)$ et $(1, 2)$ et continue de délivrer les paquets suivants à l'application (comportement qui aurait été identique même si la perte des s-paquets avait eu lieu). Le dernier cas de figure concerne la perte de p-paquets, sur l'exemple de la figure 2.3(d), le récepteur détecte le trou dans la séquence i provoqué par la perte du p-paquet $(2, 0)$, les paquets suivants ne sont pas délivrés à l'application tant que le paquet $(2, 0)$ n'est pas corrigé et reçu.

Finalement, ce système de numérotation fournit au récepteur les moyens de détecter toute perte de p-paquets et la plupart des pertes de s-paquets. Il existe un cas de figure où des pertes de s-paquets ne sont pas détectées, précisément quand ces pertes sont suivies d'un p-paquet. Mais en pratique, cette faiblesse n'apparaît pas puisque les paquets embarquent un numéro de séquence distinct du couple (i, j). Il suffit donc de surveiller l'évolution du numéro de séquence pour détecter toutes les pertes de s-paquets.

Si le récepteur détecte une perte de p-paquet, il doit conserver tous les paquets arrivant en mémoire mais ne doit pas les délivrer à l'application tant que la correction n'a pas eu lieu. Inversement, une absence de perte ou un s-paquet perdu autorisent la couche transport réceptrice à remonter les ADUs à l'application. La réaction de la couche transport dépend de l'évolution du couple (i, j) conformément à l'algorithme de la Fig. 2.4.

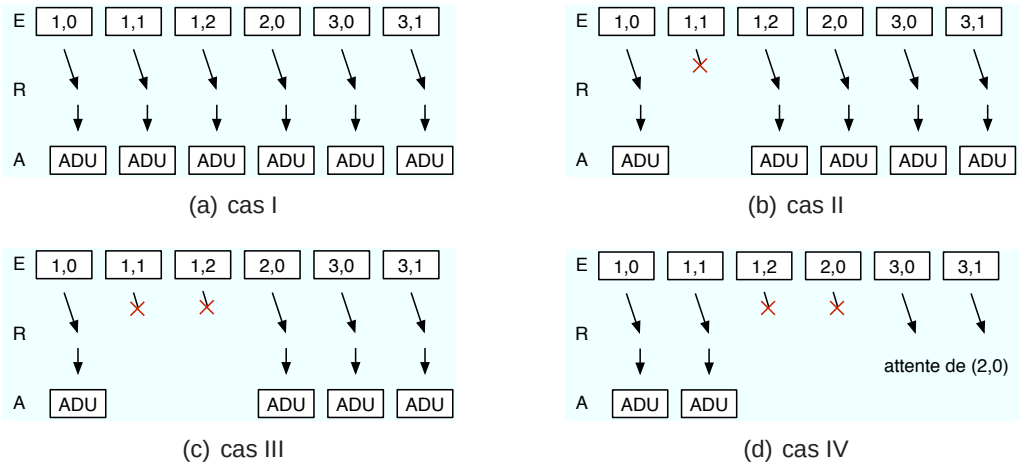


FIG. 2.3: Identification de la classe des paquets perdus par le récepteur.

On remarque qu'une perte de p-paquet est bloquante en réception, aucune ADU n'est remontée à l'application tant que la perte n'est pas corrigée. Ce comportement est lié à l'ordre total qui caractérise notre service.

```
Initialisation :
  prec_i ← 0
  prec_j ← 0
pour chaque arrivée de paquet
  lire le couple (count_i ; count_j) relatif au paquet
  si (courant_i = prec_i)
    délivrer le contenu du paquet à la couche application
    mémoriser le couple (courant_i ; courant_j) :
    prec_j ← courant_j
  sinon si ((prec_i + 1) < borne_k) ET (i = prec_i + 1) ET (j = 0))
        délivrer le contenu du paquet à l'application
        mémoriser le couple (i ; j) :
        prec_i ← courant_i
        prec_j ← 0
  sinon si ((prec_i + 1) = borne_k) ET (i = 0) ET (courant_j = 0))
        délivrer le contenu du paquet à l'application
        mémoriser le couple (i ; j) :
        prec_i ← 0
        prec_j ← 0
  sinon
    écarter le paquet (attente de correction)
```

Variables mémorisées :

prec_i : valeur du compteur i du dernier paquet traité.
prec_j : valeur du compteur j du dernier paquet traité.
courant_i : valeur du compteur i calculée pour le paquet courant.
courant_j : valeur du compteur j calculée pour le paquet courant.

constantes :

borne_k : valeur maximum du compteur cyclique i.

FIG. 2.4: Algorithme de détection de la classe des paquets en réception pour leur délivrance à la couche application.

2.1.2 Mode d'acquittement des paquets

Nous avons vu à la section 2.1.1 que le récepteur 2CP-ARQ est en mesure d'identifier toutes les pertes de paquets et d'identifier s'il s'agit de p-paquets. La manière dont l'émetteur 2CP-ARQ va lui-même détecter les pertes dépend du mode d'acquittement des paquets qui peut être positif (ACK) ou négatif (NACK).

Lorsque le mode d'acquittement positif est adopté, le récepteur envoie un ACK chaque fois qu'un paquet est correctement reçu (*i.e.* sans erreur et en ordre). Côté émetteur, une technique à base de temporisations est utilisée pour détecter les pertes. Quand il envoie un paquet, l'émetteur déclenche un temporisateur pour une durée légèrement supérieure à un RTT. S'il ne reçoit pas l'acquittement avant la fin de la temporisation, il estime que le paquet a été perdu. Cette technique implique que l'émetteur soit doté d'une instrumentation pour l'estimation du RTT. Lorsque le mode d'acquittement négatif est adopté, le récepteur envoie un NACK pour signaler les paquets qu'il a détecté perdus. La détection des pertes peut être basée sur l'évolution du numéro de séquence des paquets ou sur des temporisations. L'utilisation du numéro de séquence est le moyen le plus simple, mais après une perte, il faut nécessairement attendre qu'un nouveau paquet arrive pour s'en rendre compte (la perte de plusieurs paquets consécutivement va augmenter le retard de réaction). De plus, le récepteur est facilement abusé si des paquets arrivent dans le désordre, comme cela est possible dans les réseaux IP. L'utilisation de temporisations évite ces problèmes mais elle ne fonctionne bien que si l'intervalle d'envoi (et donc d'arrivée) des paquets est régulier. Quel que soit le mode d'acquittement adopté, la perte d'acquittements va induire l'émetteur en erreur. Lorsqu'il s'agit de la perte d'un ACK, l'émetteur va engager inutilement quelques retransmissions de paquets, ce qui n'est pas grave en soi. Les pertes de NACK sont plus préjudiciables et elles doivent être récupérées par le protocole, qui sera donc plus complexe. Le mode d'acquittement positif est le plus simple à mettre en œuvre.

Dans ses travaux de thèse (Marasli, 1997), Marasli a comparé les performances d'un service de transport partiellement fiable selon qu'il utilise le mode ACK ou NACK. Bien que les performances des deux approches soient extrêmement proches en termes de délai et de probabilité d'acheminement des paquets, l'approche par acquittements positifs (ACK) est légèrement devant.

2.1.3 Stratégie de retransmission des paquets

Les paquets peuvent être émis avec ou sans anticipation. Le mode de transmission le plus simple est dit *Stop & Wait*. Les paquets sont alors émis un par un, toute nouvelle émission est déclenchée par le bon acquittement du paquet précédent, soit au maximum un paquet par RTT. Les retransmissions sont quant à elles déclenchées par des temporisations (voir Fig. 2.5). Le mécanisme *Stop & Wait* est d'une complexité faible, l'émetteur doit uniquement mémoriser le dernier paquet transmis. Ses performances sont à la mesure de sa complexité, avec seulement un paquet transmis par RTT. Le débit est extrêmement limité, et ce d'autant plus que le temps de propagation du réseau est grand.

Si le réseau est caractérisé par une bande passante et un délai importants (produit bande passante $\times$ délai élevé), il est nécessaire d'émettre par anticipation pour atteindre des débits proches de ce que permet la capacité des liens d'interconnexion. C'est le mode d'émission adopté par la plupart des flux sur l'Internet : TCP émet par anticipation une quantité de données variables qui correspond à plusieurs paquets. Le récepteur n'acquitte pas forcément tous les paquets et un ACK valide forcément toutes les données antérieures.

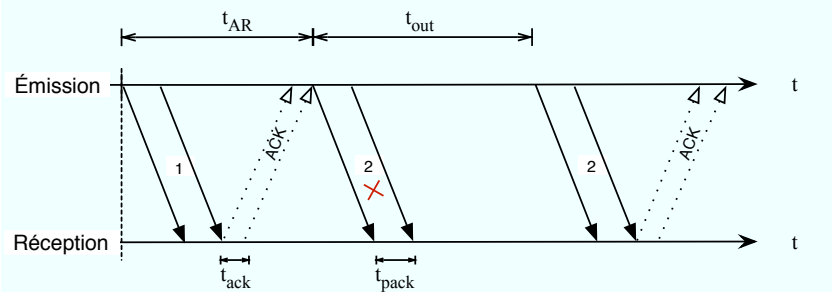


FIG. 2.5: Transmission non anticipée : *Stop & Wait*.

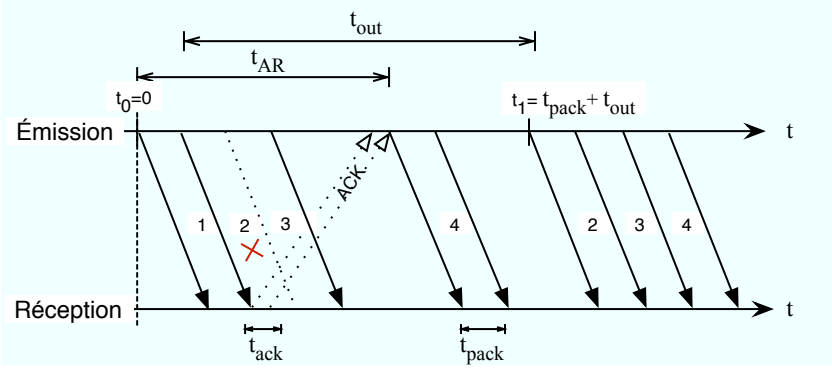


FIG. 2.6: Mécanisme de gestion des retransmissions depuis le p-paquet perdu.

Lorsque les paquets sont émis par anticipation, il est possible de lancer la récupération d'une perte par réémission depuis le paquet incriminé. C'est le cas du mode de retransmission *Go-Back-N*. *Go-Back-N* est illustré sur la figure 2.6 avec un exemple montrant l'émission anticipée de $N = 3$ paquets, les notations utilisées sont explicitées dans le tableau 2.1. Le paquet numéro 3 est émis alors que le second est perdu. Lorsque la perte est détectée par la source après une temporisation, l'émetteur la corrige en émettant de nouveau 3 paquets depuis ce paquet 2 alors que les paquets 3 et 4 n'ont pas forcément été perdus. Cette stratégie de retransmission des paquets nécessite la mémorisation par l'émetteur des N paquets émis par anticipation et susceptibles d'être retransmis. Elle n'est pas optimale dans le sens où le processus de récupération peut conduire à des émissions inutiles (paquets non perdus). Une évolution du mode de récupération des pertes consiste à ne retransmettre que les paquets perdus (mode *Selective Repeat*) aussi défini en option pour TCP. La retransmission sélective implique la détection par l'émetteur de toutes les pertes qui concernent les paquets émis par anticipation, pas uniquement la première. Au niveau du récepteur, cela implique un mécanisme d'acquittement de tous les paquets, ainsi que la mémorisation plus complexe des numéros de séquence des paquets manquants en attente de correction.

Les deux stratégies de retransmissions sont envisageables pour 2CP-ARQ. Cependant, seules les pertes de p-paquets engagent la procédure de retransmission. Si l'on considère la correction par retransmission non sélective, deux approches sont possibles : soit l'émetteur émet à nouveau tous les paquets émis par anticipation, et ce depuis le p-paquet perdu en incluant les éventuels s-paquets qui ont pu lui succéder, soit l'émetteur ne retransmet que les p-paquets émis par anticipation depuis le p-paquet perdu. En considérant la correction sélective, seuls les p-paquets perdus sont retransmis. Du point de vue de la complexité, les deux va-

riantes de retransmissions non sélectives sont semblables. Le récepteur ne nécessite aucune adaptation dans le sens où quoi qu'il arrive, il s'accommode très bien de la non transmission de s-paquets. Au niveau de l'émetteur, la nuance entre les deux variantes est minime puisque si seuls les p-paquets sont retransmis depuis la perte, l'émetteur peut alors se contenter de mémoriser uniquement les p-paquets anticipés. La stratégie de retransmissions sélectives est plus complexe. L'émetteur doit en effet détecter toutes les pertes de p-paquets anticipés, et pas seulement la première. Ceci suppose que le récepteur continue d'acquitter même après la détection d'une perte, il doit également être en mesure de mémoriser les paquets et de reconstruire la séquence après retransmission sélective.

TAB. 2.1: Variables temporelles caractéristiques du type de contrôle Go-Back-N et leurs notations.

Notation	*Définition*
t_{pack}	Temps de l'émission d'un paquet
t_{ack}	Temps de l'émission d'un acquittement
t_{AR}	Temps aller-retour : du début de l'émission d'un paquet à la complète réception de l'ACK correspondant
t_{out}	Durée après laquelle le *timer* expire (timeout). $t_{out} > t_{AR}$

2.1.4 Conception d'un protocole complet

2CP-ARQ emprunte un certain nombre d'éléments à TCP. Tout d'abord il s'agit également d'un protocole de transport orienté connexion au-dessus d'IP. La conception d'un protocole complet nécessite par conséquent la définition des procédures de connexion et de déconnexion. 2CP-ARQ peut employer le même mécanisme de connexion par poignée de main en trois étapes que TCP :

1. client → serveur : `SYN`
2. serveur → client : `SYN/ACK`
3. client → serveur : `ACK`

Procédure à la suite de laquelle les deux parties ont initialisé et échangé leur numéro de séquence. La procédure de déconnexion en quatre étapes de TCP est aussi transposable pour 2CP-ARQ (FIN puis ACK sur chaque extrémité).

2CP-ARQ et TCP diffèrent par contre sur le mode d'acquittement des paquets et sur le marquage de la fiabilité des paquets. En effet, 2CP-ARQ transporte des flux de paquets et non des flux d'octets, tous les paquets doivent donc être acquittés. Du fait de son système de marquage original qui repose sur un couple de compteurs (i, j), 2CP-ARQ nécessite deux champs supplémentaires par rapport à TCP pour stocker les deux compteurs. Comme cela a été précisé en section 2.1.1.2, plus ces champs sont étendus et moins la probabilité qu'un s-paquet se transforme en p-paquet est importante. Une profondeur de 16 bits pour chaque compteur est donc suffisante pour que ce cas reste marginal, surtout que ces compteurs ne sont utilisés qu'à des fins de détection de la classe des paquets et non pas pour en contrôler l'ordre.

L'approche ALF implique qu'un paquet transporte une ADU. La fragmentation des paquets au niveau IP n'est déjà pas recommandée si l'on utilise TCP, mais avec 2CP-ARQ ce serait

encore plus problématique. En cas d'un fragment relatif à un s-paquet, il est en effet inutile de continuer à transporter les autres fragments, cela irait à l'encontre de l'objectif de 2CP-ARQ qui vise à émettre moins de données. À ce titre, il est nécessaire que l'application forme des ADUs dont la taille n'excède pas la taille maximum des segments (MSS), taille qui est directement liée au MTU. La connaissance préalable du MTU est donc une obligation. Pour ce faire, les deux parties doivent rechercher le MTU minimum de tous les nœuds du chemin qui les interconnectent. L'algorithme de découverte du plus petit MTU du chemin (Path MTU discovery, PMTU-D) permet d'y parvenir (Mogul et Deering, 1990).

La régulation de débit est un autre point de divergence entre 2CP-ARQ et TCP. Tandis que TCP régule son débit mécaniquement en fonction du RTT et des congestions, 2CP-ARQ repose sur un mécanisme d'adaptation *TCP-friendly* qui fait intervenir le calcul du débit en fonction des pertes et du RTT (Cf. chapitre 1). Ce calcul du débit est rendu possible du fait de l'acquittement positif de tous les paquets tel que 2CP-ARQ le prévoit. Le taux de pertes et la fluctuation du délai peuvent donc être mesurés, ce qui permet de calculer le débit *TCP-friendly* correspondant.

2.2 Introduction au modèle probabiliste de 2CP-ARQ

L'évaluation des performances d'un protocole comme 2CP-ARQ peut être réalisée de trois manières : par modèle analytique, par simulation ou par expérimentation. L'approche analytique est la plus rapide à faire car elle ne nécessite pas de développer entièrement l'environnement et les scénarios de l'évolution comme cela serait le cas pour les approches par simulation et par expérimentation. En contrepartie, les performances sont exprimées par des espérances mathématiques (*i.e.* des valeurs moyennes basées sur des probabilités) et sont obtenues en posant des hypothèses simplificatrices. Les valeurs de performances que nous allons présenter dans cette section sont donc approximatives, mais elles vont nous permettre d'évaluer théoriquement les gains que l'on peut espérer sur le temps de réponse du service de transport d'image en fonction de l'état du réseau et du niveau de relaxe de la fiabilité. Le tableau 2.2 récapitule les variables et les notations utilisées. Les hypothèses simplificatrices sur lesquelles s'appuie le modèle que nous proposons sont référencées dans la table 2.3.

TAB. 2.2: Notations des différentes variables du modèle de 2CP-ARQ.

Variable	*Définition*
p_{pack}	probabilité de perte pour chaque paquet
p_{ack}	probabilité de perte pour chaque acquittement
P_{succ}	probabilité de succès (paquet et ACK correspondant reçus) $P_{succ} = (1 - p_{pack}) \times (1 - p_{ack})$
τ_F	taux de fiabilité (ratio de p-paquets) probabilité qu'un paquet quelconque soit un p-paquet
t_{AR}	temps aller-retour (ms) de l'émission du paquet jusqu'à réception de l'ACK correspondant
t_{out}	valeur du *timeout* (ms), durée après laquelle le *timer* expire
N_{max}	nombre maximum de paquets pouvant être émis par anticipation pendant t_{AR} $N_{max} = \frac{t_{AR}}{t_{pack}}$
N	nombre de paquets émis par anticipation

TAB. 2.3: Hypothèses du modèle.

	Hypothèses
1	p_{pack} et p_{ack} sont identiques
2	t_{AR} and t_{out} sont constants
3	t_{ack} est négligeable devant t_{pack}
4	L'émetteur a toujours des paquets à émettre

2.2.1 Modélisation des pertes

Parmi les modèles de pertes présentés au premier chapitre à la section 1.1.1, nous avons retenu le processus de Bernoulli pour modéliser les pertes de paquets, plutôt qu'un modèle à mémoire tel que Gilbert. À première vue, cette hypothèse est fausse si l'on considère que les pertes de paquets ne sont pas indépendantes et nécessitent par conséquent un modèle à mémoire tel que Gilbert ou Gilbert étendu. Cependant nous pouvons nuancer cette remarque. D'une part, le modèle de Gilbert n'est pas toujours le plus pertinent pour expliquer les pertes. Yajnik *et al* ont montré dans (Yajnik *et al.*, 1999) que selon la trace étudiée, la modélisation qui correspond le plus aux pertes réelles est soit Bernoulli (7 traces sur 38 totalisant 76 heures), soit Gilbert (10 traces), soit encore Markov d'ordre k variable pour les 21 traces restantes. D'autre part, si nous considérons que les pertes de paquets sont dictées par un modèle à mémoire comme celui de Gilbert à deux états (2 paramètres) alors que Bernouilli (un seul paramètre) est utilisé pour tenter de les expliquer, alors il convient de quantifier l'erreur qui est faite. Dans ce cas nous allons adapter $\hat{p}$ la probabilité moyenne qu'un paquet soit perdu qui est le seul paramètre du modèle de Bernoulli, de façon à correspondre à la probabilité π_1 d'être dans l'état 1 (perte) du modèle de Gilbert qui elle même s'exprime en fonction des deux paramètres p et q :

$$\pi_1 = \frac{p}{p+q} = \hat{p}$$

Selon le modèle de Gilbert, la perte de k paquets en série a une distribution géométrique dont la probabilité est exprimée de la manière suivante :

$$p_k = (1-q)^{k-1} \times q$$

Avec Bernoulli cette probabilité notée $\hat{p}_k$ est définie de la façon suivante :

$$\hat{p}_k = \hat{p}^{k-1} \times (1-\hat{p})$$

On rappelle que le modèle de Gilbert suppose que $p+q<1$ car si $p+q=1$ il se réduit alors à Bernoulli. Les probabilité de perte d'un paquet isolé est alors $p_1 = q$ et $\hat{p}_1 = 1-\hat{p}$, respectivement avec les modèles de Gilbert et Bernoulli. Nous pouvons alors écrire l'égalité suivante :

$$\hat{p}_1 = 1-\pi_1 = 1-\frac{p}{p+q} = \pi_0 = \frac{q}{p+q}$$

Comme $p+q<1$ avec p et q strictement positifs selon la définition du modèle de Gilbert il est alors évident que $\hat{p}_1 > p_1$, c'est-à-dire que Bernoulli surestime la probabilité des pertes uniques.

L'estimation des erreurs causées par le modèle de Bernoulli, si on admet que les pertes réelles sont dictées par un modèle de Gilbert, a été étudiée par Jiang et Schulzrinne dans

(Jiang et Schulzrinne, 2000). LEs auteurs montrent qu'avec un taux de pertes de 0,47% qui correspond à leur première trace de 10039 paquets, Bernoulli évalue le nombre de pertes uniques (sur 10039) à 39,8 alors que Gilbert les évalue à 34. Les pertes de paquets consécutifs sont au contraire sous-évaluées par Bernoulli puisque d'après le même exemple, les pertes doubles s'élèvent à 0,19 pour Bernoulli alors que Gilbert les estime à 5,1.

Nous considérons 2CP-ARQ qui nous le rappelons, corrige les pertes selon un type de contrôle Go-Back-N. Or dans ce cas, qu'il y ait un ou plusieurs paquets consécutivement perdus, la récupération reprendra à partir du premier paquet perdu. Cela signifie que la sous-estimation des pertes en rafales et la surestimation des pertes uniques provoquées par le modèle de Bernoulli (si on admet toutefois que les pertes considérées sont modélisables au mieux par Gilbert) conduisent à une augmentation du nombre de série qui vont provoquer le processus de récupération de pertes. Autrement dit, Bernouilli va dans ce cas surévaluer le nombre de reprises sur perte de paquet et par conséquent surévaluer également le temps moyen nécessaire pour servir un paquet. L'adoption du processus de Bernoulli pour modéliser les pertes ne va donc pas augmenter artificiellement les performances de 2CP-ARQ puisque c'est même l'inverse qui se produit.

Avec ce modèle, chaque paquet est potentiellement perdu indépendamment des autres avec une probabilité notée p_{pack} (voir figure 2.7). Il en va de même pour les acquittements avec une probabilité de perte p_{ack}.

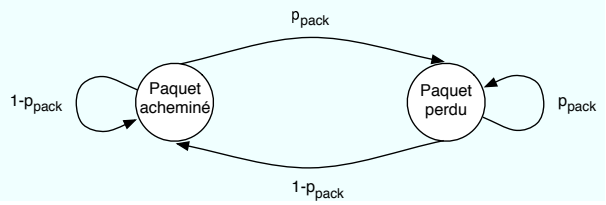


FIG. 2.7: Processus de perte de paquets.

2.2.2 Modélisation de la fiabilité de 2CP-ARQ

Le modèle doit avant tout permettre le calcul du temps de service en fonction des pertes d'une part, mais aussi en fonction du niveau de fiabilité moyen du flux de paquets. Ce niveau de fiabilité global varie entre la non fiabilité lorsque que le flux n'est constitué que de s-paquets à totalement fiable quand tous les paquets sont primaires. Le niveau de fiabilité est donc défini par un taux de fiabilité τ_F tel que $0 \leqslant \tau_F \leqslant 1$ qui correspond à la proportion de p-paquets de la transmission.

La couche transport ne connaissant pas *a priori* la distribution de la classe de fiabilité de chacun des paquets à transmettre, cela dépend de l'application, nous admettrons que la couche transport de l'émetteur marque chaque paquet indépendamment avec la probabilité τ_F que le marquage soit de type p-paquet.

Les processus de décision de la fiabilité des paquets, de perte de paquets et d'acquittements, ainsi que de leur correction par retransmission sont illustrés par la Fig. 2.8. Cette figure ne tient toutefois pas compte de l'émission des paquets par anticipation.

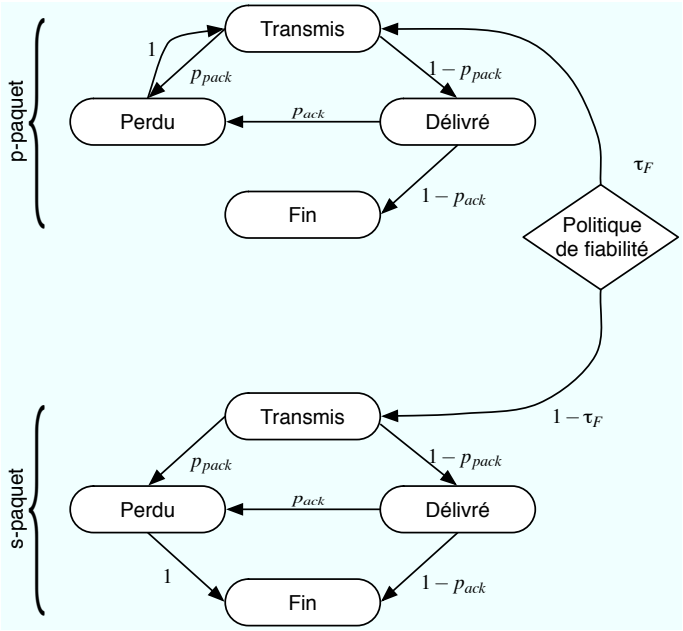


FIG. 2.8: Politique de fiabilité, détection des pertes et retransmission.

2.2.2.1 Probabilité de délivrance des paquets

Cette probabilité, notée P_{deliv}, ne dépend que du taux de fiabilité. Il s'agit de la probabilité qu'un paquet soit délivré à la couche application du récepteur quel que soit le nombre de ses retransmissions. Un p-paquet est toujours délivré par la couche transport du récepteur alors que les s-paquets ne le sont que s'ils n'ont pas été perdus lors de leur unique émission valide. Malgré les éventuelles multiples émissions d'un s-paquet donné consécutivement à la perte d'un p-paquet, seule sa dernière émission est valide puisque les éventuelles réceptions précédentes sont écartées. Cette probabilité notée P_{deliv} peut donc s'écrire :

$$P_{deliv} = \tau_F + (1 - \tau_F)(1 - p_{pack}) \tag{2.1}$$

Parmi les différents paquets de données, certains s-paquets seront manquants du point de vue de la couche transport en réception. P_{deliv} correspond aussi au ratio des paquets reçus utilisables en réception sur le nombre total des différents paquets émis au cours d'une transmission. La surface de la Fig. 2.9 montre évidemment que la probabilité de délivrance d'un paquet est d'autant plus grande que le taux de fiabilité est fort pour des taux de pertes non nuls et strictement inférieurs à 1. La question est de savoir quel est le surcoût temporel de la fiabilité, en d'autres termes, quelle économie permet le relâchement de la contrainte de fiabilité.

2.3 Modèle de 2CP-ARQ

La grandeur qui nous intéresse est la durée du service. Le temps de service dépend du nombre de paquets différents à transmettre, de la politique de fiabilité et des pertes. Si le volume de données à transmettre est tel que le nombre de paquets est grand, on peut considérer que le temps de service est proportionnel à la durée moyenne de la transmission efficace d'un paquet.

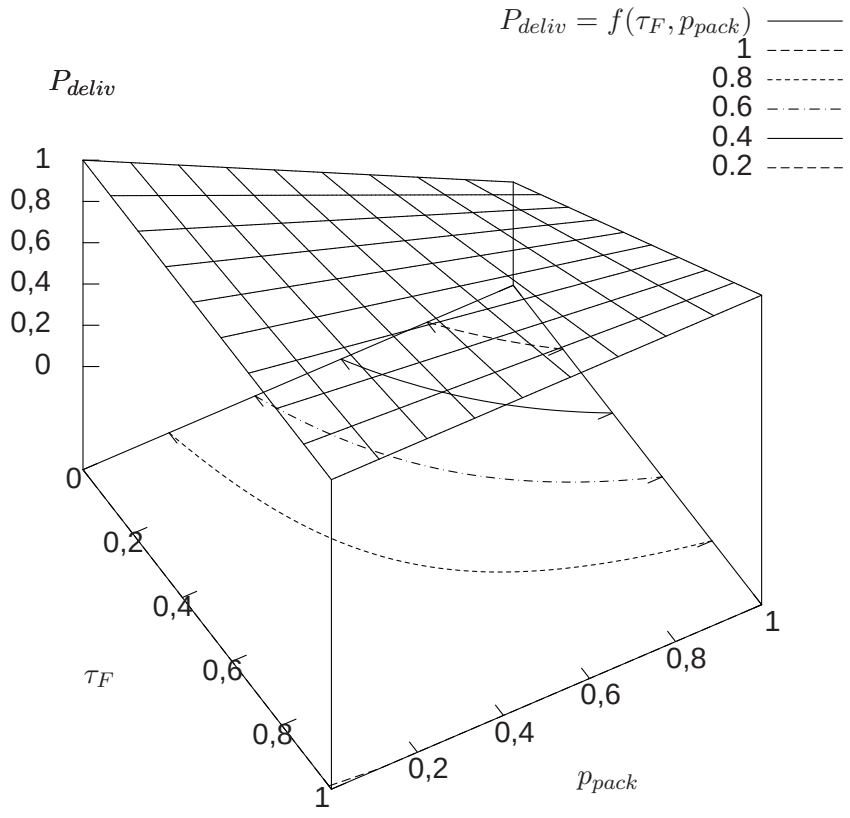


FIG. 2.9: Probabilité qu'un paquet soit délivré à la couche application en réception.

Définitions :

– Nous définissons la durée de transmission effective d'un paquet p_n par le temps qui sépare sa première transmission de la transmission du paquet p_{n+1} telle que le paquet p_n est considéré transmis (acquitté dans le cas d'un p-paquet). Il ne sera plus donc plus émis. Le temps moyen consommé par la transmission effective d'un paquet est noté T_{tep}.
– Une transmission est dite efficace si le paquet dont elle fait l'objet n'est plus jamais retransmis (l'émetteur peut alors écarter ce paquet de son buffer).

2CP-ARQ émet les paquets par série de N pendant t_{AR}. Le temps qui sépare l'émission du premier paquet d'une série au premier de la série suivante varie dans l'intervalle $[t_{AR}, t_{AR} + (N-1) \times t_{pack} + t_{out}]$ selon le rang des pertes éventuelles et leur classe parmi les N paquets de la série (cf. Fig. 2.6). On note T le temps séparant deux séries consécutives. Pour les mêmes raisons, le nombre de transmissions efficaces parmi les N varie dans l'intervalle $[0, N]$: zéro si le premier paquet de la série est un p-paquet perdu, N si aucune perte de p-paquet n'est survenue. Ce nombre de paquets transmis efficacement dans une série est noté $pack$.

On en déduit alors une estimation du temps moyen consommé lors de la transmission effective d'un paquet par le rapport des espérances mathématiques du temps consommé par une série et du nombre de paquets effectivement transmis lors d'une série (voir équation 2.33).

$$T_{ep} = \frac{E[T]}{E[pack]} \tag{2.2}$$

L'analyse de toutes les combinaisons de p-paquets et de s-paquets et de toutes les combinaisons de pertes au sein d'une série de N paquets permet d'exprimer le temps T et le nombre de paquets efficaces $pack$ de chaque série. L'expression de la probabilité d'occurrence de chacune de ces combinaisons permet ensuite d'évaluer les espérances $E[T]$ et $E[pack]$.

Le dénombrement de toutes les configurations de p-paquets et de s-paquets parmi N est immédiat : il en existe 2^N. Chaque paquet peut éventuellement se perdre, il y a donc finalement $2^{2\times N}$séries de N paquets différentes si l'on considère qu'une série peut contenir de 0 à N paquets perdus. Pour chacune de ces séries élémentaires, nous pourrions établir les probabilités d'apparition, puis déterminer le temps consommé et le nombre de paquets efficacement transmis. Toutes les séries ne consomment pas un temps différent car il existe plusieurs configurations qui mènent au même résultat du point de vue temporel (par exemple lorsqu'aucun paquet n'est perdu). Finalement nous obtiendrions un certain nombre de temps consommés différents et il serait possible de calculer la probabilité qu'une série consomme chacune des valeurs de consommation possibles.

Le nombre de séries élémentaires devient rapidement important à mesure que N augmente, il est de plus inutile de s'intéresser à toutes les séries de N paquets émis par anticipation possibles puisque nous cherchons à extraire les possibilités de consommation de temps par une série et les probabilités associées. C'est pourquoi il est plus judicieux de rechercher quelles sont ces possibilités de temps consommé et quelles sont les séries qui conduisent à ce résultat. L'expression des probabilités est ensuite très directe. L'étude des probabilités nécessaire à l'évaluation du temps moyen pour servir un paquet fait l'objet de ce paragraphe, néanmoins de plus amples détails sur l'expression de ces probabilités sont disponibles en annexe A aux paragraphes A.1 et A.2, avec notamment des figures et des explications additionnelles permettant d'illustrer les formules de probabilités présentées respectivement dans les sections 2.3.1 et 2.3.2 de ce chapitre.

Par définition, une série consomme le minimum de temps lorsque la série suivante débute le plus tôt possible et qu'aucune correction de perte n'est engagée. Cela impose deux conditions :

– du point de vue de l'émetteur, aucun p-paquet n'est perdu (les p-paquets et leurs acquittements sont acheminés),
– le premier paquet de la série est correctement acheminé, qu'il s'agisse d'un p-paquet ou non. L'émetteur ne pourra en effet poursuivre avec le premier paquet de la série suivante que si sa fenêtre d'anticipation est créditée par un acquittement.

Le temps minimum consommé par une série correspond au temps aller-retour t_{AR} qui sépare l'émission du premier paquet du retour de son acquittement. Toute série de N paquets dont le premier paquet ainsi que tous les p-paquets sont acquittés consommera le temps t_{AR}. Les N paquets émis lors de cette série sont transmis efficacement puisqu'aucun ne sera émis à nouveau.

Notons qu'en l'absence de perte de p-paquet, les N paquets de la série sont transmis efficacement, mais le temps nécessaire avant que la série suivante ne débute peut varier en fonction du rang du premier paquet dont l'acquittement parviendra à l'émetteur. En effet, toujours sous l'hypothèse selon laquelle aucun p-paquet n'est perdu, si le premier paquet de la série est détecté perdu par l'émetteur (nécessairement un s-paquet) alors il faudra attendre l'acquittement relatif au second paquet pour débuter la série suivante. La série suivante sera donc délayée d'un temps t_{pack} et le temps consommé sera alors $t_{AR} + t_{pack}$ pour transmettre efficacement les N paquets. Par extension, une séquence de k s-paquets consécutifs non acquittés en début de série induira un retard de $k \times t_{pack}$ si le paquet suivant est acquitté. Ce cas de figure est illustré par la figure 2.11(b).

La dernière possibilité qui permet à la série de transmettre N paquets survient lorsqu'elle est constituée de s-paquets uniquement et qu'aucun n'est acquitté. Ce cas peu probable est illustré par la figure 2.11(c). L'absence d'acquittement obligera la source à attendre l'expiration du timeout t_{out} déclenché lors de l'émission du dernier paquet pour débuter l'émission de la série suivante. Ce temps consommé vaudra donc $(N-1) \times t_{pack} + t_{out}$.

Les trois cas généraux exposés ci-dessous ont un point commun : la totalité des paquets de la série sont transmis efficacement, autrement dit aucun d'eux ne sera retransmis, ou encore tous les p-paquets sont acquittés.

Le quatrième et dernier cas général comprend toutes les séries dont un ou plusieurs p-paquets sont détectés perdus (non acquittés). Notons que le temps consommé par la série ainsi que le nombre de paquets transmis efficacement ne dépendent que du rang du premier p-paquet non acquitté. En effet, si le paquet de rang k est le premier p-paquet non acquitté, alors la série suivante débutera avec ce paquet en particulier, et ce quels que soient le nombre et le rang des pertes de p-paquets qui peuvent éventuellement survenir dès le rang $k+1$ avec $k \leqslant N$. Le temps consommé ne dépend donc que du rang k de la première perte de p-paquet et vaut $(k-1) \times t_{pack} + t_{out}$. La même remarque s'applique au nombre de paquets transmis efficacement, une première perte de p-paquet située au rang k implique systématiquement que les $(k-1)$ paquets précédents sont transmis efficacement.

Les probabilités d'occurrence des quatre cas généraux présentés ci-dessus ne peuvent cependant pas être exprimées immédiatement, ceci pour la raison suivante : une perte d'un p-paquet impose obligatoirement que la série suivante débute par l'émission d'un p-paquet. Si tel est le cas, ce premier paquet n'a naturellement pas la probabilité τ_F d'être un p-paquet comme pour tout autre paquet. Il convient par conséquent d'analyser distinctement ces deux cas de figure. Nous noterons NPL la condition « non perte de p-paquet lors de la série précédente » et PL lorsque la série doit commencer par la récupération d'un p-paquet perdu.

Les probabilités de tous les cas possibles sachant la condition NPL ou PL pourront ensuite permettre d'évaluer les probabilités de transitions du modèle à deux états présenté sur la figure

2.10. La perte d'un p-paquet sous la condition NPL (le premier paquet de la série peut être s- ou p-) implique forcément que la probabilité que le premier paquet de la série suivante soit p- vaille 1. De manière analogue, seule l'absence de perte de p-paquet sous la condition PL (le premier paquet est obligatoirement un p-paquet) permet d'enchaîner avec une série en sachant la condition PL.

La connaissance des probabilités qu'une série soit dans l'état NPL ou PL permettra enfin une expression non conditionnelle des probabilités d'occurrence de chaque série, et donc du temps consommé et du nombre de paquets transmis efficacement.

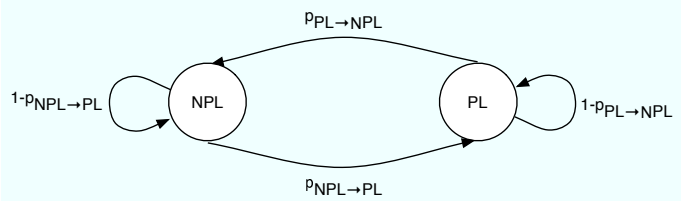


FIG. 2.10: Modèle à deux états : perte d'un p-paquet ou non et transitions $NPL \rightleftharpoons PL$.

La section 2.3.1 commence par analyser les combinaisons possibles sachant la condition NPL et en exprime les probabilités d'occurrence. La section 2.3.2 traite de l'analyse des combinaisons sachant la condition PL. Les résultats sont ensuite confondus dans la section 2.3.3 pour en extraire une expression non conditionnelle des probabilités.

2.3.1 Analyse probabiliste sachant la condition NPL

Nous pouvons classer les différentes combinaisons de p- et s-paquets parmi N selon le temps T_s de la série. Toutes ces combinaisons peuvent ainsi être classées dans l'un des quatre groupes suivants :

groupe i : $T = t_{AR}|NPL$, aucun p-paquet n'est détecté perdu par l'émetteur pendant la série, et au moins l'ACK du premier est retourné à l'émetteur (voir Fig. 2.11(a)).

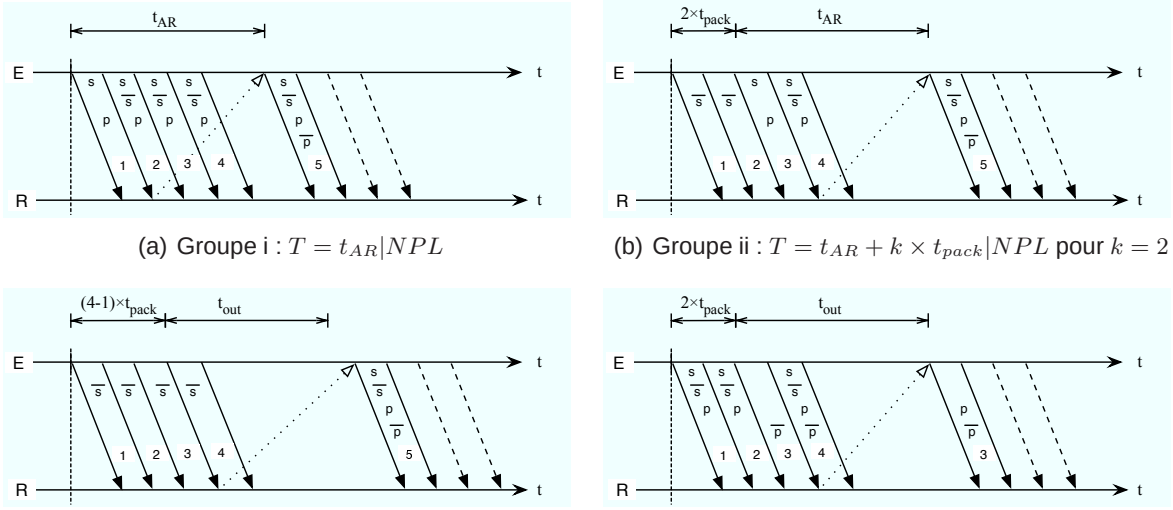
groupe ii : $T = t_{AR}+k\times t_{pack}|NPL$, aucun p-paquet n'est détecté perdu par l'émetteur, mais la série commence par une séquence de k s-paquets pour lesquels aucun ACK ne retourne à l'émetteur. Le paquet $(k+1)$ est cependant correctement acquitté (voir Fig. 2.11(b)).

groupe iii : $T = (N-1)t_{pack} + t_{out}|NPL$, la série n'est constituée que de s-paquets et aucun d'eux n'est acquitté (voir Fig. 2.11(c)).

groupe iv : $T = (k-1)t_{pack} + t_{out}|NPL$, le $k^{\text{ième}}$ paquet est le premier p-paquet détecté perdu de la série, *i.e.* , paquet ou ACK perdu (voir Fig. 2.11(d)).

La Fig. 2.11 montre la différenciation entre ces groupes par un exemple avec $N = 4$. Un paquet peut être un s-paquet acquitté, un s-paquet détecté perdu (paquet ou ACK perdu), un p-paquet acquitté ou un p-paquet détecté perdu. Ces possibilités sont respectivement notées s, $\bar{s}$, p et $\bar{p}$ sur la figure.

Pour chacun de ces quatre groupes, nous exprimons la probabilité d'occurrence ainsi que le nombre de paquets transmis efficacement.



(a) Groupe i : $T = t_{AR}|NPL$

(b) Groupe ii : $T = t_{AR} + k \times t_{pack}|NPL$ pour $k = 2$

(c) Groupe iii : $T = (N-1)t_{pack} + t_{out}|NPL$

(d) Groupe iv : $T = (k-1)t_{pack} + t_{out}|NPL$ pour $k = 3$

FIG. 2.11: Les quatre groupes permettant de classer les différentes séries sachant la condition NPL.

2.3.1.1 Groupe i : $T = t_{AR}|NPL$

Le premier paquet est correctement acquitté quelle que soit sa classe, cet ACK parvient à l'émetteur après $T = t_{AR}$. Tous les acquittements relatifs aux éventuels k p-paquets parmi les $N-1$ restants sont correctement retournés à l'émetteur, *i.e.* , sans *timeout*. Ce cas de figure permet à la série suivante de débuter immédiatement avec un nouveau paquet. Les N paquets sont alors transmis efficacement et la probabilité conditionnelle d'occurrence est donnée par l'équation 2.4.

$$pack_1 = pack[T = t_{AR}|NPL] = N \tag{2.3}$$

$$\begin{aligned} P_1 &= P[T = t_{AR}|NPL] \\ &= P_{succ} \sum_{k=0}^{N-1} \left[\tau_F^k \binom{N-1}{k} P_{succ}^k (1-\tau_F)^{N-1-k} \right] \end{aligned} \tag{2.4}$$

2.3.1.2 Groupe ii : $T = t_{AR} + k \times t_{pack}|NPL$

Une séquence de k s-paquets consécutifs non acquittés en début de série suivie d'un paquet correctement acquitté a pour effet de retarder la série suivante de $k \times t_{pack}$, si toutefois les p-paquets sont tous bien acquittés. Le temps consommé par cette série est alors $T = t_{AR} + k \times t_{pack}$. Comme aucun p-paquet n'est détecté perdu, les N paquets sont transmis efficacement. L'équation 2.6 exprime la probabilité conditionnelle d'occurrence de ce cas de figure en fonction de k, le nombre de s-paquets non acquittés consécutifs en début de série tel que $1 \leqslant k \leqslant N-1$.

$$pack_2(k) = pack[T = t_{AR} + k \times t_{pack}|NPL] = N \tag{2.5}$$

$$\begin{aligned} P_2(k) &= P[T = t_{AR} + k \times t_{pack}|NPL] \\ &= (1-\tau_F)^k (1-P_{succ})^k P_{succ} \times \sum_{j=0}^{N-1-k} \left[\tau_F^j \binom{N-1-k}{j} P_{succ}^j (1-\tau_F)^{N-1-k-j} \right] \end{aligned} \quad (2.6)$$

2.3.1.3 Groupe iii : $T = (N-1)t_{pack} + t_{out}|NPL$

Lorsque la série n'est constituée que de s-paquets tous détectés perdus par l'émetteur, il n'y a par conséquent aucun ACK qui peut annuler les *timers* de s-paquets précédents. La série suivante débutera donc après le *timeout* relatif au dernier paquet, ce qui conduit à un temps consommé $T = (N-1)t_{pack} + t_{out}$. Aucun des N paquets ne sera retransmis puisque ce sont des s-paquets. La probabilité conditionnelle de ce cas rare est exprimée par l'équation 2.8.

$$pack_3 = pack[T = (N-1)t_{pack} + t_{out}|NPL] = N \quad (2.7)$$

$$\begin{aligned} P_3 &= P[T = (N-1)t_{pack} + t_{out}|NPL] \\ &= (1-\tau_F)^N . (1-P_{succ})^N \end{aligned} \quad (2.8)$$

2.3.1.4 Groupe iv : $T = (k-1)t_{pack} + t_{out}|NPL$

La perte d'un p-paquet implique toujours un *timeout*. Selon le rang k de ce p-paquet, la série consomme un temps $T = (k-1)t_{pack} + t_{out}$. Ceci implique que le paquet k est le premier p-paquet détecté perdu, autrement dit tous les p-paquets précédents éventuels doivent être acquittés. Le nombre de paquets efficacement transmis ainsi que la probabilité conditionnelle d'apparition de ce cas sont exprimés en fonction du rang k du premier p-paquet perdu par les équations 2.9 et 2.10 respectivement.

$$pack_4(k) = pack[T = (k-1)t_{pack} + t_{out}|NPL] = k-1 \quad (2.9)$$

$$\begin{aligned} P_4(k) &= P[T = (k-1)t_{pack} + t_{out}|NPL] \\ &= \tau_F(1-P_{succ}) \times \sum_{j=0}^{k-1} \left[\binom{k-1}{j} \tau_F^j P_{succ}^j (1-\tau_F)^{k-1-j} \right] \end{aligned} \quad (2.10)$$

Tous les cas sachant la condition NPL ont été traités puisque nous avons vérifié que la somme de leurs probabilités est unitaire :

$$\sum_i [P_i|NPL] = P_1 + \sum_{k=1}^{N-1} P_2(k) + P_3 + \sum_{k=1}^{N} P_4(k) = 1$$

Le dernier cas étudié implique que la série suivante débute avec le p-paquet perdu, ce qui nous conduit à analyser les différentes possibilités de combinaisons de paquets et pertes en sachant la condition PL.

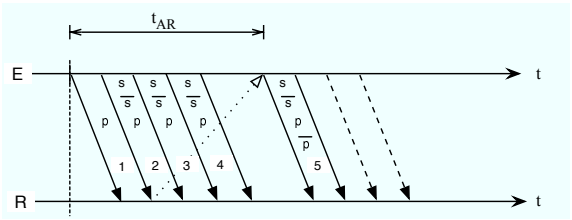
2.3.2 Analyse selon la condition PL

Nous refaisons une analyse analogue à la précédente, en considérant vraie la condition « un p-paquet a été perdu lors de la série précédente ». Dans ce cas, la probabilité que le premier paquet soit p- est de 1. Il n'est donc plus possible de rencontrer une séquence de s-paquets perdus en début de série ou une série totalement constituée de s-paquets. Les groupes à étudier se réduisent donc à deux : tous les p-paquets sont acquittés et la série consomme un temps $T = T_{AR}$ ou alors un p-paquet est perdu avec le retard que cela entraîne. En réalité nous subdivisons le groupe où un p-paquet est perdu en deux groupes pour faciliter l'expression des probabilités dans la section suivante. Toutes les configurations de séries selon le temps consommé peuvent donc être classées selon trois groupes. Représentés sur la figure 2.12, ces groupes sont les suivants :

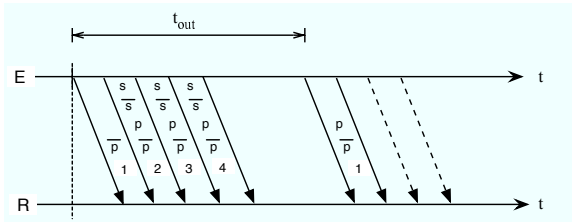
groupe v : $T = t_{AR}|PL$, tous les p-paquets sont acquittés (voir Fig. 2.12(a)).

groupe vi : $T = t_{out}|PL$, le premier paquet est détecté perdu (voir Fig. 2.12(b)).

groupe vii : $(k-1)t_{pack}+t_{out}|PL$, le paquet au rang k avec $2 \leqslant k \leqslant N$ est le premier p-paquet détecté perdu (voir Fig. 2.12(c)).

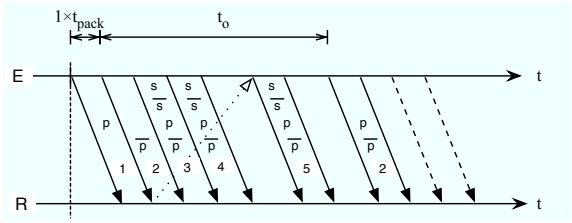


(a) Groupe v : $T = t_{AR}|PL$



(b) Groupe vi : $T = t_{out}$

Groupe vii :



(c) $(k-1)t_{pack} + t_{out}|PL$, avec $k = 2$

FIG. 2.12: Les trois groupes permettant de classer les différentes séries sachant la condition PL.

2.3.2.1 Groupe v : $T = t_{AR}|PL$

Lorsque tous les p-paquets sont acquittés, il n'y a aucun retard introduit. Le nombre de paquets efficacement transmis est naturellement N, la probabilité conditionnelle d'occurrence de ce cas est exprimée par (2.12).

$$pack_5 = pack[T = t_{AR}|PL] = N \tag{2.11}$$

$$\begin{aligned} P_5 &= P[T = t_{AR}|PL] \\ &= P_{succ} \times \sum_{k=0}^{N-1} \left[\tau_F^k \times \binom{N-1}{k} \times P_{succ}^k (1-\tau_F)^{N-1-k} \right] \end{aligned} \tag{2.12}$$

2.3.2.2 Groupe vi : $T = t_{out}|PL$

La perte du premier paquet est détectée puis corrigée après un temps $T = T_{out}$. Aucun des paquets de cette série n'est transmis efficacement puisque la série suivante commencera avec l'envoi du premier paquet.

La probabilité conditionnelle d'apparition de ce cas est donnée par (2.14).

$$pack_6 = pack[T = t_{out}|PL] = 0 \tag{2.13}$$

$$P_6 = P[T = t_{out}|PL] = 1 - P_{succ} \tag{2.14}$$

2.3.2.3 Groupe vii : $T = (k-1)t_{pack} + t_{out}|PL$

Si le premier p-paquet perdu est au rang k, avec $2 \leqslant k \leqslant N$, cette perte implique un retard de $(k-1)t_{pack}$ en plus du *timeout*. La série suivante débutera par l'envoi de ce p-paquet perdu donc $(k-1)$ paquets ne seront jamais retransmis. La probabilité conditionnelle d'apparition de cette configuration est donnée par (2.16).

$$pack_7(k) = pack[T = (k-1)t_{pack} + t_{out}|PL] = k - 1 \tag{2.15}$$

$$\begin{aligned} P_7(k) &= P[T = (k-1) \times t_{pack} + t_{out}|PL] \\ &= P_{succ} \times \tau_F \times (1 - P_{succ}) \times \sum_{j=0}^{k-2} \left[\binom{k-2}{j} \times \tau_F^j \times P_{succ}^j \times (1 - \tau_F)^{k-2-j} \right] \end{aligned} \tag{2.16}$$

Si l'on admet la condition PL, toutes les combinaisons de paquets parmi N et toutes les configurations de pertes font partie de l'un des trois groupes exposées précédemment. Nous avons vérifié que la somme des probabilités conditionnelles sachant PL est unitaire :

$$\sum_i [P_i|PL] = P_5 + P_6 + \sum_{k=2}^{N} P_7(k) = 1$$

En définitive, les cas de figures appartenant aux groupes 1 à 7 supposent la condition NPL ou PL. Par définition, seule la perte d'un p-paquet implique la condition PL pour la série suivante, or les pertes de p-paquets sont toutes incluses dans les groupes 4, 6 et 7 pour lesquels la probabilité est exprimée par P_4, P_6 et P_7. Cs groupes mènent tous obligatoirement à la perte d'un p-paquet et par conséquent à l'état PL, tandis que l'absence de perte de p-paquets exprimée par les groupes 1, 2, 3 et 5 conduit forcément à l'état NPL. Ce point est détaillé en annexe A, au paragraphe A.3 précisément.

2.3.3 Analyse probabiliste non conditionnelle

Les probabilités conditionnelles P_1 à P_7 sont donc les probabilités de transitions du modèle markovien à deux états illustré par la Fig. 2.10. La section A.3 de l'annexe A apporte de plus amples explications et la démonstration des résultats présentés ici.

La transition $NPL \rightarrow PL$ est franchie si un p-paquet est perdu sachant que ce n'était pas le cas lors de la série précédente. Cette transition a pour probabilité $p_{NPL \rightarrow PL}$, voir équation 2.17.

$$p_{NPL \rightarrow PL} = \sum_{k=1}^{N} P_4(k) \tag{2.17}$$

On reste dans l'état NPL si aucun p-paquet n'est perdu sachant la condition NPL, la probabilité associée est $p_{NPL\circlearrowleft}$, voir équation 2.18.

$$\begin{aligned} p_{NPL\circlearrowleft} &= 1 - p_{NPL\to PL} \\ &= P_1 + \sum_{k=1}^{N-1} [P_2(k)] + P_3 \end{aligned} \tag{2.18}$$

De même, la transition $PL \to NPL$ est franchie uniquement pour les configurations de paquets telles qu'aucun p-paquet n'est perdu sachant PL. Cette transition a pour probabilité $p_{PL\to NPL}$ donnée par l'équation 2.19.

$$p_{PL\to NPL} = P_5 \tag{2.19}$$

La dernière possibilité consiste à rester dans l'état PL, ce qui suppose la perte d'un p-paquet sachant que le premier paquet de la série est forcément un p-paquet. Ces pertes de p-paquets conditionnelles ont pour probabilité P_6 et $P_7(k)$ respectivement quand il s'agit du premier paquet ou du paquet k pour $2 \leqslant k \leqslant N$. La probabilité $p_{PL\circlearrowleft}$ de rester dans l'état PL est donnée par l'équation 2.20.

$$\begin{aligned} p_{PL\circlearrowleft} &= 1 - p_{PL\to NPL} \\ &= P_6 + \sum_{k=2}^{N} P_7(k) \end{aligned} \tag{2.20}$$

La matrice de transition P est donc la suivante :

$$\mathbf{P} = \begin{pmatrix} p_{NPL\circlearrowleft} & p_{NPL\to PL} \\ p_{PL\to NPL} & p_{PL\circlearrowleft} \end{pmatrix}$$

Le vecteur ligne π des probabilités stationnaires peut alors s'écrire :

$$\pi = (\pi_{NPL}, \pi_{PL})$$

Les probabilités stationnaires d'être dans l'état NPL ou PL sont données respectivement par π_{NPL} et π_{PL}, voir équations 2.21 et 2.22.

$$\begin{aligned} \pi\mathbf{P} &= \pi \\ \pi_{NPL} &= \frac{1}{1 + \frac{p_{NPL\to PL}}{p_{PL\to NPL}}} \qquad (2.21) \\ \pi_{PL} &= \frac{1}{1 + \frac{p_{PL\to NPL}}{p_{NPL\to PL}}} \qquad (2.22) \end{aligned}$$

En section 2.3.1 nous avons différencié quatre groupes en fonction du temps T consommé à la condition NPL. Trois nouveaux groupes ont été ajoutés à la section suivante (2.3.2) en supposant la condition PL. Ces deux évènements NPL et PL couvrent l'univers des évènements possibles. En appliquant le principe des probabilités totales il est possible d'exprimer les probabilités d'occurrence de chacun de ces sept groupes sans supposer de condition.

Les probabilités d'occurrence des évènements NPL et PL sont exprimée par les probabilités stationnaires π_{NPL} et π_{PL} qui sont à présent connues. De plus, les sept groupes qui ont été étudiés peuvent être fusionnés en fonction du temps T consommé. En effet, un temps T

peut être atteint par plusieurs évènements appartenant à des groupes différents qui supposent parfois la condition NPL ou PL. L'expression non conditionnelle des probabilités signifie que la somme de leurs probabilités est unitaire, ceci a été vérifié numériquement à l'aide de la programmation des équations qui sont présentées dans cette section. Les probabilités d'occurrence de chaque groupe peuvent à présent être additionnées et comparées entre elles, ce qui permet de fusionner ces sept groupes pour obtenir les cinq suivants :

Groupe viii La série consomme un temps $T = t_{AR}$, ce qui signifie qu'aucun p-paquet n'est perdu et que le premier paquet est acquitté.

Groupe ix Le temps consommé par la série est $T = t_{AR} + k \times t_{pack}$. Cela implique d'une part que les k premiers paquets soient s- et perdus, et d'autre part qu'aucun p-paquet ne soit perdu (correspond au groupe ii mais exprimé sans condition).

Groupe x $T = (N-1) \times t_{pack} + t_{out}$ lorsque la série n'est composée que de s-paquets tous perdus (correspond au groupe iii mais cette fois-ci exprimé sans condition).

Groupe xi $T = t_{out}$, le premier paquet est un p-paquet perdu. Ce groupe comprend le groupe vi (condition PL) mais aussi un sous-ensemble du groupe iv (condition NPL) lorsque $k = 1$.

Groupe xii $T = (k-1) \times t_{pack} + t_{out}$, le paquet au rang k avec $2 \leqslant k \leqslant N$ est le premier p-paquet perdu. Ce groupe est composé de groupe iv (condition NPL) pour $2 \leqslant k \leqslant N$ et du groupe vii (condition PL).

2.3.3.1 Groupe viii : $T = t_{AR}$

Une série consomme un temps $T = t_{AR}$ si le premier paquet est correctement acquitté et si aucune perte de p-paquet n'a lieu. D'après les équations 2.4 et 2.12, $P_1 = P_5$ et nous pouvons écrire la probabilité d'occurrence de ce scénario (voir équation 2.23).

$$\begin{aligned} P_8 &= P[T = t_{AR}] \\ &= P_1 \times \pi_{NPL} + P_5 \times \pi_{PL} \\ &= P_1 \end{aligned} \tag{2.23}$$

Auquel cas, le nombre de paquets efficacement transmis est N

$$pack_8 = pack[T = t_{AR}] = N \tag{2.24}$$

2.3.3.2 Groupe ix : $T = t_{AR} + k \times t_{pack}$

Le temps $T = t_{AR} + k \times t_{pack}$ lorsque la série débute par une séquence de k s-paquets non acquittés. Cela peut évidemment n'apparaître que sous la condition NPL. D'où l'équation 2.25 pour exprimer la probabilité d'apparition de ce scénario en fonction de k la longueur de la séquence de s-paquets détectés perdus, pour tout entier naturel k tel que $1 \leqslant k \leqslant N-1$.

$$\begin{aligned} P_9(k) &= P[T = t_{AR} + k \times t_{pack}] \\ &= P_2(k) \times \pi_{NPL} \end{aligned} \tag{2.25}$$

Le nombre de paquets transmis efficacement est alors N, quel que soit k tel que $1 \leqslant k \leqslant N-1$.

$$pack_9 = pack[T = t_{AR} + k \times t_{pack}] = N \tag{2.26}$$

2.3.3.3 Groupe x : $T = (N-1) \times t_{pack} + t_{out}$

La série consomme un temps $T = (N-1) \times t_{pack} + t_{out}$ pour le scénario selon lequel tous les paquets sont des s-paquets et aucun n'est acquitté. Il ne peut apparaître que sous la condition NPL. À partir de l'équation 2.8 l'expression de la probabilité d'apparition de ce cas devient (voir équation 2.27) :

$$\begin{aligned} P_{10} &= P[T = (N-1) \times t_{pack} + t_{out}] \\ &= P_3 \times \pi_{NPL} \end{aligned} \tag{2.27}$$

Comme aucun p-paquet n'est détecté perdu, les N paquets sont considérés transmis.

$$pack_{10} = pack[T = (N-1) \times t_{pack} + t_{out}] = N \tag{2.28}$$

2.3.3.4 Groupe xi : $T = t_{out}$

Le temps de la série $T = t_{out}$ si le premier paquet est un p-paquet détecté perdu. Si la condition PL est vraie, la probabilité d'occurrence est alors P_6 (voir 2.14). Dans le cas contraire, se référer à l'équation de P_4 (2.10) pour $k = 1$. D'où la probabilité P_{11} que la série consomme un temps $T = t_{out}$ donnée par (2.29).

$$\begin{aligned} P_{11} &= P[T = t_{out}] \\ &= P_4(1) \times \pi_{NPL} + P_6 \times \pi_{PL} \end{aligned} \tag{2.29}$$

Dans ce cas aucun paquet n'est transmis puisque la série suivante débutera par le p-paquet perdu.

$$pack_{11} = pack[T = t_{out}] = 0 \tag{2.30}$$

2.3.3.5 Groupe xii : $T = (k-1) \times t_{pack} + t_{out}$

Le temps T consommé s'élève à $k \times t_{pack} + t_{out}$ si le paquet k est le premier p-paquet détecté perdu de la série, quel que soit l'entier $k \in [2, N]$, l'exception $k = 1$ ayant été traitée précédemment. Si la condition PL est vraie, la probabilité de ce scénario est donnée par $P_7(k)$ (cf. 2.16). Inversement, si la condition NPL est vraie il faut se référer à $P_4(k)\ \forall$ entier $k \in [2, N]$ (voir 2.10).

$$\begin{aligned} P_{12}(k) &= P[T = (k-1) \times t_{pack} + t_{out}] \\ &= P_4(k) \times \pi_{NPL} + P_7(k) \times \pi_{PL},\ 2 \leqslant k \leqslant N \end{aligned} \tag{2.31}$$

Le paquet de rang k étant le premier p-paquet perdu, $(k-1)$ paquets sont alors considérés transmis.

$$\begin{aligned} pack_{12} &= pack[T = (k-1) \times t_{pack} + t_{out}] \\ &= k-1,\ \ 2 \leqslant k \leqslant N \end{aligned} \tag{2.32}$$

2.3.4 Évaluation du temps effectif moyen pour servir un paquet

Notre objectif initial est l'évaluation de T_{ep}, le temps moyen nécessaire à la transmission efficace par paquet. Nous rappelons l'équation 2.33 qui exprime ce temps en fonction des espérances $E[T]$ et $E[pack]$:

$$T_{ep} = \frac{E[T]}{E[pack]} \tag{2.33}$$

L'expression de ces espérances est donnée par les équations 2.34 et 2.35 en fonction des classes de temps T présentées précédemment, de $pack_i$ le nombre de paquets efficacement transmis et de la probabilité d'occurrence P_i.

$$\begin{aligned} E[T] &= t_{AR} \times P_8 + \sum_{k=1}^{N-1} [(t_{AR} + k \times t_{pack}) \times P_9(k)] + [(N-1) \times t_{pack} + t_{out}] \times P_{10} \\ &\quad + t_{out} \times P_{11} + \sum_{k=2}^{N} [((k-1) \times t_{pack} + t_{out}) \times P_{12}(k)] \end{aligned} \tag{2.34}$$

$$E[pack] = N \left(P_8 + \sum_{k=1}^{N-1} P_9(k) + P_{10} \right) + \sum_{k=2}^{N} [(k-1) \times P_{12}(k)] \tag{2.35}$$

2.3.5 Régulation de débit TCP-friendly

Pour prendre en compte les effets d'un mécanisme de régulation de débit de type *TCP-friendly* dans notre modèle du protocole 2CP-ARQ, il suffit de faire varier N, le nombre de paquets dans une série pendant un temps t_{AR}. Autrement dit, il s'agit de considérer N dans les équations 2.34 et 2.35 comme une variable dont la valeur est fonction des paramètres du réseau, en particulier le taux de pertes et le RTT, soit $N = f(t_{AR})$. N ne pourra toutefois pas excéder la quantité maximum N_{max} donnée en (2.36).

$$N_{max} = \frac{t_{AR}}{t_{pack}} \tag{2.36}$$

Padhye et al. ont modélisé le débit de TCP dans (Padhye *et al.*, 1998) par l'équation 2.37.

$$B(p) \approx \min \left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)} \right) \tag{2.37}$$

B est exprimé en nombre de paquets par unité de temps, $Wmax$ est le nombre maximum de paquets émis par anticipation, b représente le nombre de paquets acquittés de façon cumulative avec un seul ACK reçu, typiquement deux pour TCP (Padhye *et al.*, 1998). RTT et T_0 sont équivalents à t_{AR} et t_{out} dans notre modèle respectivement. $(1-p)$ étant la probabilité qu'un paquet soit correctement acquitté, pour notre modèle l'équivalence est $p = 1 - P_{succ}$.

En multipliant les termes de l'équation de Padhye par RTT on exprime non plus le nombre de paquets par unité de temps (le débit) mais le nombre de paquets par tranche de temps t_{AR}, ce qui correspond à la profondeur N de la série pour notre modèle. Cette variation du nombre de paquets émis par anticipation dans un intervalle t_{AR} est exprimée en fonction de P_{succ}, qui

est la probabilité qu'un paquet soit détecté correctement reçu du point de vue de l'émetteur. Elle suit désormais l'équation 2.38.

$$\begin{aligned} N(P_{succ}) &\approx \lceil \min\left(N_{max}, N^{\star}\right) \rceil \\ N^{\star} &= \frac{1}{\sqrt{\frac{4(1-P_{succ})}{3}} + t_{out} \times \min\left(1, 3\sqrt{\frac{6\times(1-P_{succ})}{8}}\right)(1-P_{succ})\left(1+32(1-P_{succ})^2\right)} \end{aligned} \quad (2.38)$$

La fonction plafond (notée $\lceil x \rceil$) de l'équation 2.38 exprime le fait que la taille de la fenêtre d'anticipation est exprimée en nombre entier de paquets et que la taille minimale de la fenêtre est de un paquet.

Les effets du niveau de fiabilité sur le temps de service, ainsi qu'une vérification de l'équité de 2CP-ARQ envers TCP sont présentés dans la section suivante. Nous utilisons à cet effet le modèle que nous avons développé, notamment les équations 2.34 et 2.35 en tenant compte des variations de l'anticipation N telle que $N = f(P_{succ})$.

2.4 Évaluation des performances

De manière à évaluer les performances potentielles de 2CP-ARQ, c'est-à-dire indépendamment des applications, nous avons défini deux scénarios de transmission correspondant à des accès Internet avec des produits bande passante×délai courants. Le premier scénario représente un accès par modem 56kbits/s sur une ligne du réseau téléphonique commuté, avec un temps aller-retour constant de 250ms. Le second correspond à une connexion de type DSL avec un débit descendant de 1024kbits/s et un délai aller-retour de 100ms. Les valeurs de RTT choisies sont des moyennes obtenues par *pings* successifs entre une station située à Nancy connectée via un fournisseur d'accès grand public standard et des sites hébergés aux États Unis. De ces caractéristiques nous déduisons les valeurs supposées constantes en entrée du système (voir tableau 2.4). Pour toutes les évaluations, le *timeout* est défini tel que $t_{out} = 1,2 \times t_{AR}$.

TAB. 2.4: Paramètres en entrée du modèle.

Paramètre	*DSL 1024kbits/s*	*RTC 56kbits/s*
MTU (octets)	1500	576
t_{AR} (ms)	100	250
t_{pack} (ms)	11,7	82,3
N_{max} (paquets)	8	3

2.4.1 Vérification du comportement TCP-friendly

2CP-ARQ doit réguler son débit à l'émission en fonction des pertes du réseau de manière similaire à TCP dans la même situation. Les courbes de la Fig. 2.13 comparent les nombres de paquets émis dans un intervalle de temps aller-retour t_{AR} de 2CP-ARQ avec le débit TCP-friendly dans les mêmes conditions selon la formule de Padhye (2.37). La comparaison est faite avec le scénario DSL pour lequel l'émetteur peut émettre huit paquets au maximum dans l'intervalle t_{AR} et les taux de perte de paquets et d'acquittements sont supposés identiques

($p_{pack} = p_{ack}$). Les discontinuités en escalier du débit à l'émission de 2CP-ARQ sont expliquées par le caractère entier du nombre de paquets N émis par anticipation au cours d'une série. Les deux courbes suivent les mêmes tendances pour des taux de pertes de paquets raisonnables, *i.e.* , inférieurs à 0,15, ce qui montre bien que le contrôle de débit à l'émission de 2CP-ARQ est équitable avec celui de TCP. Plus le produit bande passante$\times$délai est grand, plus le nombre maximum de paquets émis au cours d'une série est important, ce qui aura pour effet de réduire les différences entre les deux courbes.

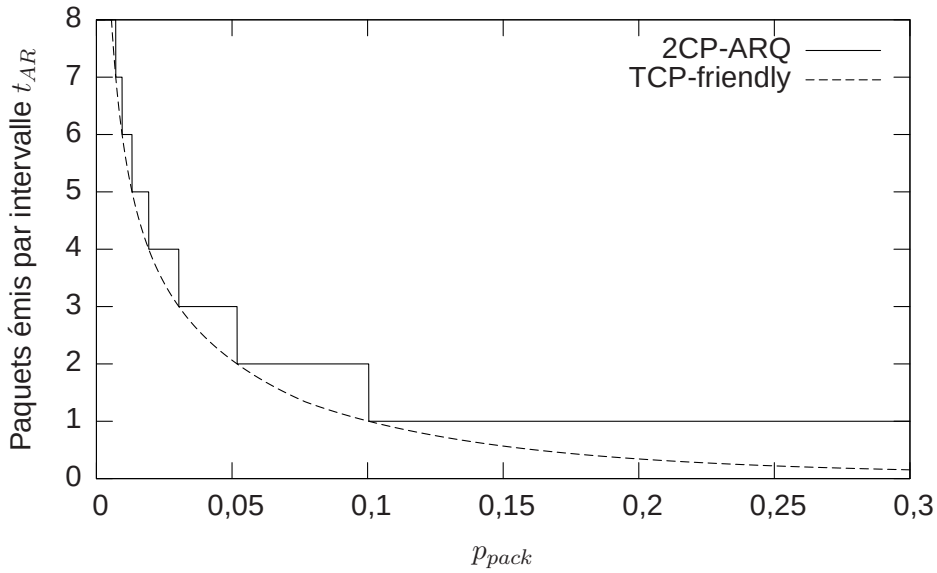


FIG. 2.13: Comportement TCP-friendly de 2CP-ARQ : quantité de données envoyées par anticipation par intervalle de temps t_{AR} (100ms pour la connexion DSL dans le cas présent).

2.4.2 Impact du relâchement de la contrainte de fiabilité

Les courbes présentées dans cette section montrent l'évolution du temps de réponse du service de transport en fonction du niveau de fiabilité du flux de paquets et quantifient le gain de temps obtenu en comparaison d'un service de transport à fiabilité totale. Étudions d'abord T_{ep}, le temps moyen pour transmettre un paquet définitivement, qui est exprimé par l'équation 2.33. La figure 2.14 montre les variations de T_{ep} en fonction du taux de perte de paquets dans le réseau (p_{pack}) et du taux de fiabilité de l'application (τ_F), en considérant le scénario DSL. T_{ep} a été calculé pour des taux de perte allant de 0 à 0,25 (au delà, les résultats n'ont plus de sens pratique) et pour un taux de fiabilité variant de 0 (tous les paquets sont « perdables ») à 1 (tous sont fiables). L'analyse des résultats montre d'abord que le temps de transport par paquet vaut 12,5ms quand le réseau n'est pas sujet à des pertes de paquets. C'est environ le temps nécessaire pour émettre une fenêtre d'anticipation (RTT) divisé par le nombre de paquets de la fenêtre. Lorsque l'application fixe un transport fiable ($\tau_F = 1$), T_{ep} augmente selon une tendance exponentielle. On obtient 18,8ms par paquet avec un taux de perte de 0,01 ; 29,1ms pour un taux de perte de 0,02 ; 44,0ms pour un taux de perte de 0.05 ; 73,8ms pour un taux de perte de 0.1 et 192,8ms pour un taux de perte s'élevant à 0,25. Puisque le service fourni dans ces conditions est totalement fiable, les performances sont du même ordre

que celles qui seraient obtenues avec TCP. Considérons à présent un service partiellement fiable, de façon à vérifier si l'on peut gagner du temps. Pour un taux de perte de 0,01 on obtient un T_{ep} de 18,4ms pour un taux de fiabilité de 0,8 ; de 17,9ms pour un taux de fiabilité de 0,6 ; 17,5ms pour un taux de fiabilité de 0,4 et 17,1ms pour un τ_F de 0,2. De la même manière, on obtient un T_{ep} de 42,0ms ; 39,9ms ; 37,8ms et 35,8ms respectivement pour un taux de fiabilité de 0,8 ; 0,6 ; 0,4 et 0,2.

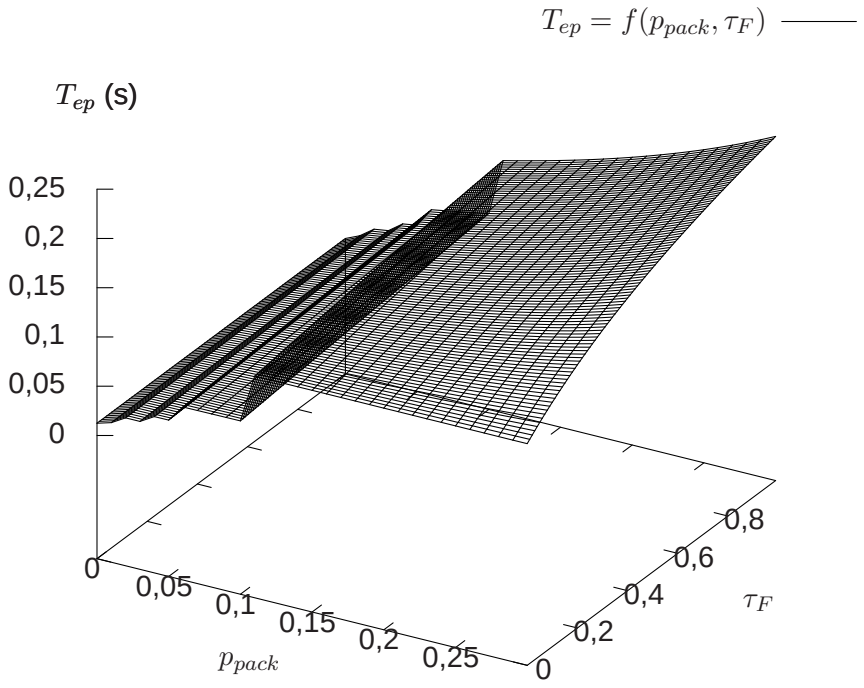
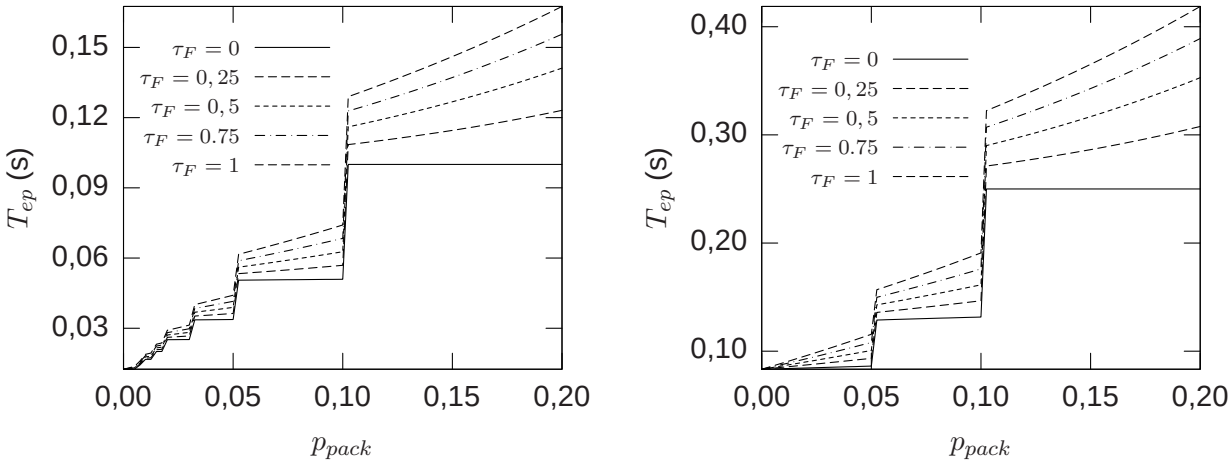


FIG. 2.14: Scénario DSL : évolution du temps de service en fonction des pertes et de la politique de fiabilité.

Les variations du temps de service en fonction des pertes de paquets sont aussi illustrées par les courbes 2.15(a) et 2.15(b), respectivement pour nos scénarios de connexion DSL à 1024kbits/s et RTC à 56kbits/s. Ces courbes exhibent deux caractéristiques majeures. D'une part, elles quantifient la réduction du temps de service qui va de paire avec une moindre fiabilité. D'autre part, elles montrent une augmentation de T_{ep} par paliers. Ces derniers sont expliqués par la variation de la valeur entière du nombre de paquets émis par anticipation pendant un intervalle de temps aller-retour t_{AR}. La courbe 2.15(a) comporte huit paliers qui correspondent à autant de valeurs possibles pour le nombre de paquets émis par anticipation dans un intervalle t_{AR} pour le scénario de connexion DSL. Pour les mêmes raisons, la courbe 2.15(b) relative à la connexion 56kbits/s par modem analogique comporte trois paliers. Quel que soit le scénario de connexion, les courbes montrent les mêmes caractéristiques, avec notamment les mêmes tendances à la baisse dans les variations de T_{ep} au fur et à mesure que le taux de fiabilité diminue.



(a) $T_{ep} = f(p_{pack})$, connexion DSL 1024kbits/s.

(b) $T_{ep} = f(p_{pack})$, connexion RTC 56kbits/s.

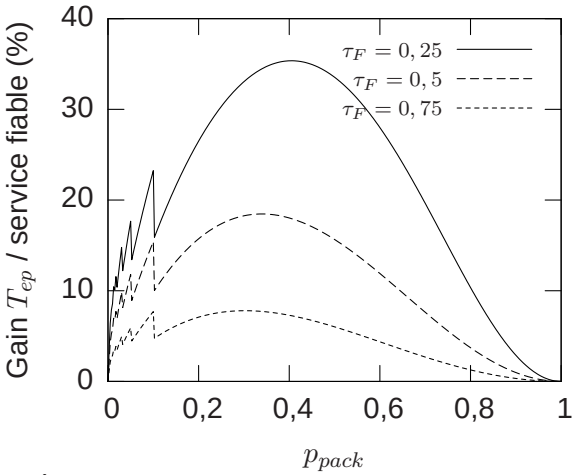
FIG. 2.15: Évolution du temps de service en fonction des pertes de paquets.

La comparaison du temps de réponse entre un service fiable et un service partiellement fiable est présentée sur les courbes 2.16(a) et 2.16(b). La réduction du temps de service relative y est exprimée en pour-cent pour différents niveaux de fiabilité en fonction du taux de pertes de paquets par rapport au service totalement fiable. Bien sûr, le gain de temps est nul lorsque le réseau ne perd aucun paquet, puis le gain augmente jusqu'à atteindre un maximum pour un taux de perte d'autant plus important que la fiabilité est faible (typiquement des taux de pertes de 0,3 à 0,4). Le gain diminue ensuite pour redevenir nul pour un taux de perte total et une fiabilité non nulle (temps de réponse infini). Cette évolution du gain pour des taux de pertes raisonnables en pratique est illustrée par la courbe 2.16(c) pour la connexion DSL à 1024kbits/s et la courbe 2.16(d). Les gains obtenus sont indéniablement intéressants avec par exemple 5,9%, 11,7%, 17,6% et 23,5% pour un taux de pertes de paquets $p_{pack} = 0,05$ respectivement avec un niveau de fiabilité τ_F de 0,75 0,5 0,25 et 0 dans le cas DSL. Avec 10% de pertes, les gains s'élèvent à 7,7%, 15,1% 23,2% et 31,2% pour les mêmes niveaux de fiabilité.

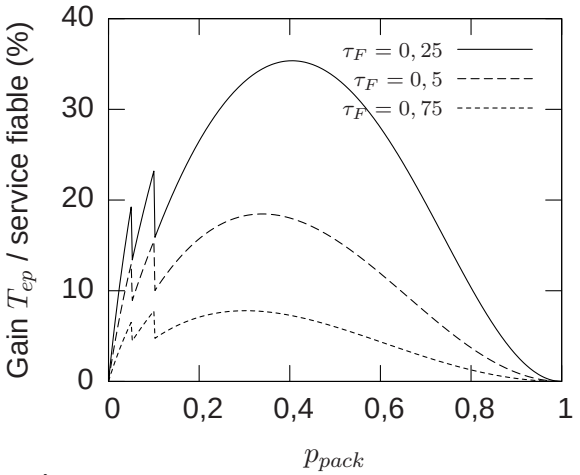
Conclusion

Dans ce chapitre, nous avons proposé un nouveau protocole de transport, 2CP-ARQ, qui différencie deux classes de fiabilité : les paquets primaires qui sont délivrés à tout prix et les paquets secondaires qui ne sont transmis qu'une seule fois, sans garantie. 2CP-ARQ définit un système de numérotation original pour faire un marqueur persistant de la classe des paquets, de sorte que le récepteur soit capable d'identifier la classe des paquets perdus. Les autres mécanismes du protocole sont plus classiques (type de contrôle, ajustement du débit . . .) et il est possible de concevoir plusieurs variantes (retransmissions sélectives ou non). Le protocole 2CP-ARQ a été conçu pour fournir un service à fiabilité partielle qui est adapté au transport d'images fixes, dans le cadre des applications n'ayant pas de contraintes de temps strictes.

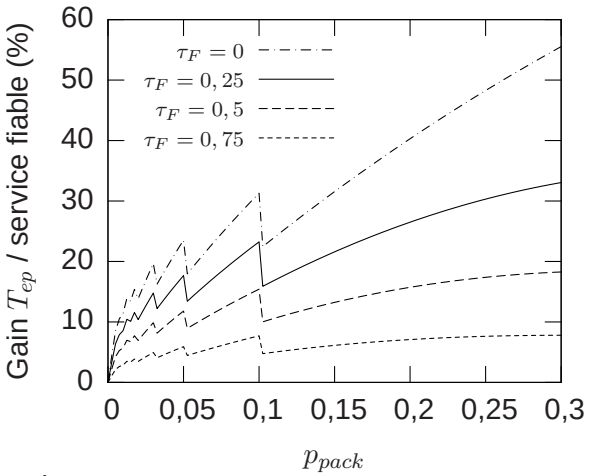
Il fournit les mêmes performances qu'un protocole fiable quand le réseau opère en deçà des conditions de saturation, *i.e.* quand il n'est pas sujet à des pertes de paquets. En présence de pertes de paquets par contre, le relâchement de la contrainte de fiabilité va permettre de limiter



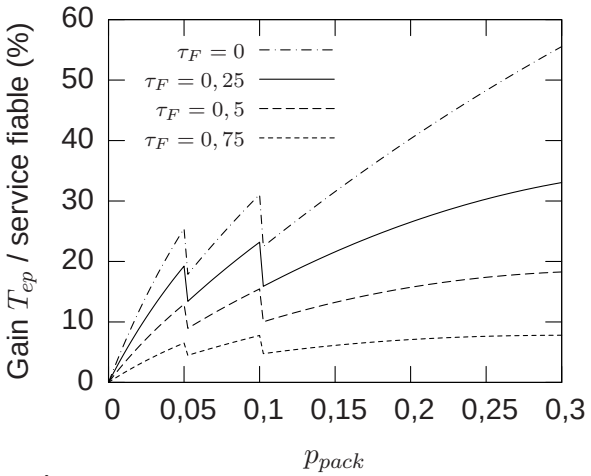
(a) Réduction du temps de service comparativement au service fiable, connexion DSL 1024kbits/s.



(b) Réduction du temps de service comparativement au service fiable, connexion RTC 56kbits/s.



(c) Réduction du temps de service pour les taux de pertes usuels, connexion DSL 1024kbits/s.



(d) Réduction du temps de service pour les taux de pertes usuels, connexion RTC 56kbits/s.

FIG. 2.16: Gain de temps de service permis par une moindre fiabilité.

l'augmentation du temps de réponse puisque toutes les pertes ne seront pas récupérées. 2CP-ARQ assume donc un transport « rapide » des images, le temps gagné se faisant au détriment de la qualité de l'image reconstruite qui subira certaines dégradations. Remarquons que la qualité des images reconstruites peut être contrôlée lorsque la compression des images fait intervenir une décomposition multirésolution puisque l'utilisateur peut spécifier quelles données sont absolument requises. Ces dernières seront transportées par des p-paquets.

Une analyse des performances du protocole a permis de montrer que l'espérance de gain peut aller potentiellement jusqu'à 35% du temps de réponse relativement à un service fiable, jusqu'à 20% si l'on reste dans des conditions raisonnables de l'Internet (moins de 20% de pertes). Ces résultats ne sont pas négligeables mais évidemment les performances réelles dépendent beaucoup du système de codage d'image qui sera utilisé par l'application. Reste à déterminer quels sont les codages appropriés, et avec ceux-ci comment les données peuvent être mises en paquets et ces paquets classifiés pour optimiser les performances du service à fiabilité partielle.

Ces considérations font l'objet de la deuxième partie de ce manuscrit, qui traite naturellement des relations entre l'image et le manque de données liées à la fiabilité partielle. Cette partie va faire l'étude conjointe du système de compression de l'image et du système de transport à fiabilité partielle. Le chapitre 3 fait l'analyse de la compatibilité de 2CP-ARQ avec un standard de compression d'image actuel : JPEG2000. Le chapitre 4 traite le cas d'un système de compression développé au laboratoire.

Deuxième partie

Compatibilité de la compression d’image avec le service de transport partiellement fiable 2CP-ARQ

Chapitre 3

Transport 2CP-ARQ et JPEG2000

Introduction

L'objectif premier de la compression est la réduction du volume des données. Cette diminution de poids est intéressante à deux points de vue : l'archivage et la transmission. La compression avant archivage limite la consommation des ressources de stockage, et par conséquent économise les ressources en bande-passante du fait de la réduction de durée du service de transmission.

La compression peut être destructive ou non. La compression non destructive, appelée aussi compression sans perte, est une opération réversible, c'est-à-dire que les données restituées après compression et décompression seront identiques à celles de départ. Ce type de compression, aussi appelé compactage, est principalement utilisé pour réduire le poids de toutes sortes de documents avec notamment les algorithmes LZ77, LZ78, LZW et tous autres dérivés de Lempel-Ziv qui reposent aussi sur la définition d'un dictionnaire. Ces algorithmes sont notamment employés par les compacteurs *zip*, *7z*, *gzip*, *bzip2* qui sont très largement utilisés.

La compression sans perte est parfois utilisée pour les images. Le format TIFF définit en effet la compression optionnelle LZW en plus du format de codage des informations dans le fichier. L'efficacité de la compression sans perte en termes de taux de compression est cependant moindre en ce qui concerne les images naturelles que pour les fichiers textes ou certaines images de synthèse par exemple. Les images de synthèse (par opposition aux images naturelles) sont en effet souvent constituées de zones unies aux contours très marqués. La répétition de motifs (suites d'échantillons identiques) caractérisant ce type d'image rend la compression sans perte encore intéressante. Les images naturelles numérisées sont quant à elles caractérisées par de fortes corrélations entre les valeurs de coefficients adjacents. Bien que proches, ces valeurs ne sont pas identiques, ce qui limite la recherche de motifs répétés et par conséquent limite aussi l'efficacité des compressions réversibles fondées sur la construction d'un dictionnaire.

Le sens même des images est différent de celui des fichiers textes contenant des informations rigoureuses et objectives. Contrairement aux données objectives, l'image n'a de sens que lorsqu'elle est perçue par la vision humaine. Or la vision est caractérisée par la perception de l'œil qui est interprétée par le cerveau, la vision humaine est donc subjective. Très sensible aux subtils dégradés qui sont caractérisés par de faibles écarts de luminance entre des échantillons proches, la vision humaine est beaucoup plus tolérante en ce qui concerne les amplitudes des coefficients relatifs aux fins détails ainsi qu'aux légères variations de teintes.

Ces caractéristiques expliquent que les images sont souvent compressées de façon destructive, *i.e.* , avec des pertes pour atteindre un taux de compression plus élevé. Les schémas de compression d'images fixes avec pertes intègrent par conséquent une étape de décomposition spatio-fréquentielle de sorte à isoler l'information qui est la moins significative pour la vision humaine : les hautes fréquences. Certains schémas de compression s'appuient ainsi sur une version discrète de la transformée de Fourier : la DCT. C'est par exemple le cas de la compression JPEG avec pertes. D'autres reposent sur la transformée par ondelettes discrète, notamment JPEG 2000 ou SPIHT.

Qu'elle soit réversible ou non, la compression implique un codage ainsi qu'un format de fichier pour contenir les informations codées si l'objectif est l'archivage. La compression en général, et en ce qui concerne les images fixes en particulier, a pour objectif l'élimination des redondances. La compression est par conséquent très adaptée à l'archivage sous forme de fichiers, mais l'élimination des redondances rend en général l'information compressée moins résistante aux pertes et aux erreurs. Il n'est ainsi pas rare que l'altération d'une archive compressée portant sur quelques octets suffise à la rendre invalide ou corrompue au point d'empêcher sa décompression. C'est pourquoi les compacteurs courants proposent des mécanismes optionnels de resynchronisation et l'ajout d'informations de redondances lors de la création de l'archive, cela dans le but d'accroître la robustesse aux erreurs et finalement permettre la récupération totale ou partielle des données originales le cas échéant. La résistance aux erreurs et la compression demeurent cependant antagonistes dans le sens ou la robustesse nécessite un supplément d'information. Les erreurs sont d'autant plus courantes que l'on s'intéresse à la transmission plutôt qu'à l'archivage, ce qui explique que la transmission des archives compressées est en général réalisée par un transport fiable, typiquement avec TCP.

Une nouvelle difficulté se dresse dès que le système de transport adopté n'est pas fiable, c'est-à-dire qui accepte des pertes de paquets. Si la compression ne tient pas compte de cette possibilité, il est possible qu'une partie voire la totalité de l'information reçue ne puisse être décodée. Il est alors nécessaire que le flux dispose au moins de points de resynchronisation de façon à ne pas corrompre la totalité de la transmission des données compressées. Même en présence de pertes de paquets, le récepteur doit toujours pouvoir décoder les données pour restituer une version de l'image. Si l'opération de décodage n'est pas robuste aux pertes de paquets, alors le système de compression ne sera pas compatible à un système de transport à fiabilité partielle.

Dans ce chapitre, nous allons étudier la compatibilité du système de compression standard JPEG2000 avec un système de transport de type 2CP-ARQ. Le choix de JPEG2000 est motivé par deux choses : c'est d'une part le standard de compression d'images le plus récent (et le plus performant) et d'autre part, l'image est codée selon une représentation multirésolution (les données de l'image codée peuvent donc être classifiées en plusieurs réseaux de priorité, condition nécessaire pour utiliser 2CP-ARQ).Nos principales contributions ont consisté à :

- analyser la sensibilité du décodeur JPEG2000 aux erreurs et aux pertes de données pour les différents types de données qui composent l'image codée.
- automatiser la classification des données de l'image codée et leur rangement dans des paquets de transport de type p- et s-.
- mettre en évidence les contradictions qui existent entre le système de compression JPEG2000 et le système de transport 2CP-ARQ.

Le chapitre est organisé selon le plan suivant : la section 3.1 fait une présentation générale de JPEG2000. La section suivante traite de l'organisation du code produit par la compression JPEG2000. Une étude de la tolérance aux pertes de JPEG2000 est présentée en section 3.3. Cette étude conduit à l'analyse de la sensibilité aux pertes selon la nature de l'information et

sur leur classification en section 3.4. Le transport de flux JPEG2000 fait l'objet de la section 3.5, avec notamment la présentation des solutions existantes et notre proposition pour associer 2CP-ARQ à JPEG2000.

3.1 Présentation de JPEG2000

JPEG2000 est un standard ISO en constante évolution depuis 1997. Il est composé de douze volets dont le premier a été finalisé en 2004 (JTC1/SC29, 2004*b*). Le standard JPEG2000 va beaucoup plus loin que la définition d'un schéma de compression d'images fixes. La première partie de la norme définit les bases du schéma de compression en lui-même, ainsi que la syntaxe du *code-stream* et le format de fichier *jp2* prévu pour le stockage. D'autres parties sont venues ensuite (parfois encore en cours d'élaboration ou en voie d'amendement par l'ISO/IEC) pour ajouter des extensions et intégrer la compression dans un cadre plus général qui tient compte aussi bien de la sécurité, des logiciels, du transport interactif ou encore des réseaux sans-fil. La norme JPEG2000 est divisée en 12 parties :

partie 1 : cœur de JPEG2000, cette partie présente une description complète du décodeur, de la syntaxe du *code-stream*, ainsi que du format de fichier portant l'extension JP2, (JTC1/SC29, 2004*b*).

partie 2 : ajout d'extensions, définition d'un format de fichier plus évolué JPX, (JTC1/SC29, 2004*c*).

partie 3 : définition des séquences animées JPEG2000 (motion JPEG2000), (JTC1/SC29, 2004*a*).

partie 4 : définition de procédures pour tester la conformité des codeurs et décodeurs vis-à-vis de la partie 1, (JTC1/SC29, 2004*d*).

partie 5 : implantations logicielles de la partie 1 du standard faisant office de référence (java avec JJ2000 et C avec Jasper), (JTC1/SC29, 2003*b*).

partie 6 : spécification du format de fichier *JPM* pour les documents à pages multiples contenant divers éléments (images, textes, par exemple les documents faxés), (JTC1/SC29, 2003*a*).

partie 7 : cette partie a été abandonnée et n'existe plus.

partie 8 : définition des aspects concernant la sécurité pour les contenus JPEG2000 (JPSEC) tels que cryptage, authentification de la source, intégrité des contenus (projet de norme en 2006).

partie 9 : définition d'un protocole pour les transmissions interactives (JPEG2000 Interactive Protocol, JPIP), (JTC1/SC29, 2005*b*).

partie 10 : traite des images JPEG2000 en trois dimensions (JP3D) (projet de norme en 2006).

partie 11 : traite de la transmission sur les réseaux sans-fil (JPWL) (projet de norme en 2006).

partie 12 : format de fichier standard et universel (ISO Base Media File Format), le volet ISO 15444-11 est équivalent à la norme ISO 14496-12 relative au format MP4, (JTC1/SC29, 2005*a*).

JPEG2000 repose sur le paradigme suivant : compresser une seule fois tout en permettant le décodage de plusieurs manières. Pour ce faire, le standard (partie 1) définit une grande

variété de partitionnements spatiaux, fréquentiels, physiques ou logiques. Ces divers découpages sont utilisés par le codeur pour construire un flux compressé hiérarchique (le *code-stream*). Le *code-stream* est composé de contributions bien délimitées qui apportent chacune leur raffinement en résolution et en qualité sur une zone de l'image et une composante données. JPEG2000 propose différents modes de progression pour le stockage de l'information et son décodage. L'ordre selon lequel le *code-stream* est agencé et selon lequel le décodeur va traiter les informations, influe en effet directement sur la façon dont le rendu de l'image reconstruite est perçu au cours du traitement. JPEG2000 autorise finalement un raffinement du rendu de l'image reconstruite au fur et à mesure que les informations sont traitées selon quatre directions : la qualité (augmentation graduelle du rapport signal sur bruit), la résolution (dimensions de l'image), la composante (luminance ou chrominance) et la position (zone spatiale). Cette aptitude semble *a priori* très intéressante pour un éventuel transport à fiabilité partielle, une des difficultés apparente réside dans le classement des informations requises ou optionnelles pour garantir un décodage satisfaisant.

3.1.1 Schéma général de la compression JPEG2000

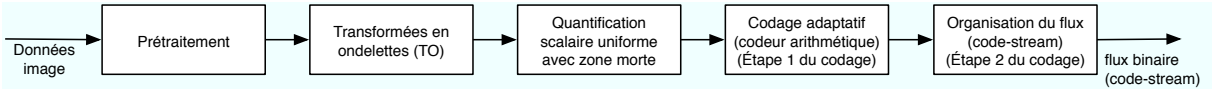


FIG. 3.1: Schéma général de la compression JPEG2000.

La compression JPEG2000 comprend 5 grandes étapes représentées sur la figure 3.1.

- Le prétraitement consiste à identifier les caractéristiques de l'image source (taille, composantes) pour éventuellement la découper en tuiles qui ne se chevauchent pas. Les coefficients de l'image source concernent chacun un pixel et une composante. Ce sont des entiers non signés sur B bits (souvent sur 8 bits). Lors de cette étape, la valeur 2^{B-1} est retranchée à chaque coefficient de façon à les coder sur des entiers signés. Pour finir, la transformation des composantes est effectuée soit de manière réversible (RCT – *Reversible Color Transform*), étape requise pour la compression sans perte, soit de façon irréversible (ICT) bien adaptée pour la compression avec pertes.
- La seconde étape consiste à décomposer l'image en plusieurs résolutions. Plusieurs transformées en ondelettes discrètes sont appliquées à cet effet.
- Les coefficients transformés issus de l'étape 2 sont quantifiés par un quantificateur scalaire avec zone morte. L'objectif de cette étape est de réduire le nombre de valeurs différentes que peuvent prendre les coefficients.
- Le codage entropique des coefficients est effectué à l'étape 4. Deux codeurs interviennent : un codeur de plans de bits et un codeur arithmétique appelé codeur MQ.
- Le code est réorganisé pour produire ce qui est appelé le *code-stream*, puis éventuellement le fichier JP2 prévu pour encapsuler ce *code-stream*.

3.1.2 JPEG2000 et partitionnements

JPEG2000 est un standard conçu pour compresser tous les types d'images naturelles, de toutes les tailles, en couleurs ou monochromes, avec une qualité de reconstruction variable allant du faible débit binaire à la compression sans perte avec un débit binaire élevé.

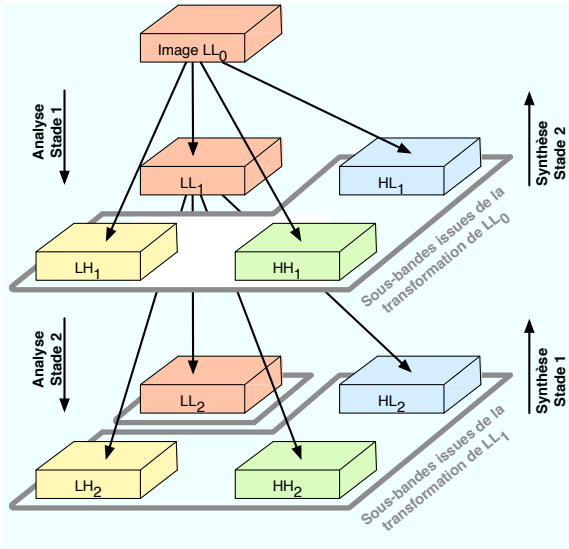


FIG. 3.2: Exemple de décomposition multirésolution de JPEG2000 par deux transformées en ondelettes discrètes. La décomposition compte alors trois résolutions.

3.1.2.1 Partitionnement en tuiles

Lors du pré-traitement, l'image source peut être partitionnée en tuiles (*tiles*). Elle est alors découpée en plusieurs sous-images qui sont chacune traitées indépendamment lors des différentes étapes de la chaîne de compression. Cette fonctionnalité est toutefois peu utilisée pour les images habituelles qui sont de taille raisonnable.

3.1.2.2 Partitionnement en composantes

Une image en niveaux de gris, ou plus généralement monochrome, est définie par une seule composante. Les images couleurs sont par contre généralement projetées dans un espace à trois dimensions. Qu'il s'agisse d'un espace RVB (Rouge Vert Bleu), YCbCr (Luminance Chrominances) ou encore YUV, il est toujours possible de passer d'un espace à l'autre. JPEG2000 peut supporter jusqu'à 16384 composantes mais en pratique ce nombre se réduit à 3.

3.1.2.3 Partitionnement en résolutions

Les échantillons de chaque tuile sont transformés dans le domaine spatio-fréquentiel à l'aide de n (généralement 5) transformées en ondelettes discrètes (TO) dyadiques à deux dimensions. Il en résulte $n+1$ résolutions constituées chacune de trois sous-bandes, sauf pour la plus petite résolution qui est issue d'un filtrage passe-bas dans les deux directions (il s'agit de l'image de départ dans une résolution réduite d'un facteur 2^n dans les deux directions). La figure 3.2 illustre le principe de la décomposition multirésolution utilisée par JPEG2000. De plus amples détails spécifiques à la transformée en ondelettes discrète 2D sont donnés dans la section 4.1.1 du chapitre suivant.

3.1.2.4 Partitionnement en *code-blocks*

Chaque sous-bande est subdivisée en blocs appelés *code-blocks*, généralement de taille 64×64 coefficients. Les *code-blocks* sont les plus petits groupes de coefficients codés indépendamment.

3.1.3 Partitionnement en *precincts*

Un *code-block* correspond donc à une petite zone spatiale de l'image pour une sous-bande et une composante données. La zone couverte par le *code-block* dépend cependant de la résolution (*i.e.* , au niveau de décomposition) considérée. Le *code-blocks* correspondra en effet à une zone de l'image originale 4 fois plus étendue au niveau de décomposition d qu'au niveau $d+1$ (voir figure 3.3).

C'est pourquoi un dernier niveau de partitionnement, logique cette fois, consiste à regrouper les *codes-blocks* relatifs à la même zone spatiale de l'image de toutes les sous-bandes d'une résolution donnée. Les *precincts* ainsi formés correspondent à une zone spatiale de l'image originale quelle que soit la résolution considérée. Le découpage en *code-blocks* et le regroupement de ces derniers en *precincts* sont illustrés sur la figure 3.3.

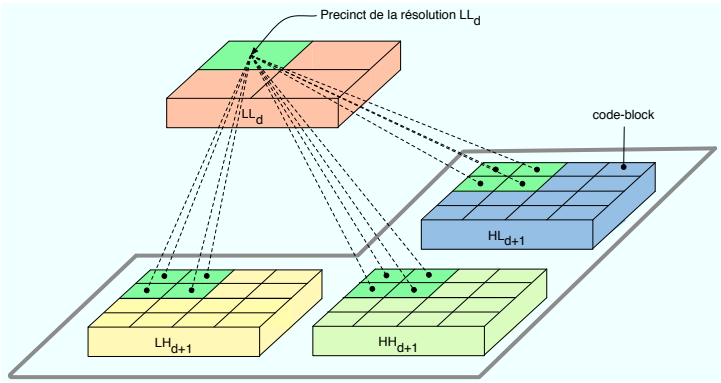


FIG. 3.3: Partitionnement des sous-bandes en *code-blocks* et regroupement en *precincts*.

3.1.3.1 Partitionnement en couches de qualité

Les coefficients transformés sont quantifiés à l'aide d'un quantificateur scalaire embarqué. Contrairement à JPEG qui quantifie chaque coefficient par un seul symbole, dans JPEG2000 ces index de quantification sont encodés bit par bit depuis le MSB jusqu'au LSB. Les bits de même poids des index quantifiés forment les plans de bits, voir figure 3.4.

Les coefficients quantifiés appartenant à chaque *code-block* sont codés indépendamment par un codeur entropique. Cette étape s'appuie notamment sur le principe EBCOT (Embedded Block Coding with Optimal Truncation), (Taubman *et al.*, 2002), (Taubman *et al.*, 2000). Elle utilise la décomposition en plans de bits pour générer un flux binaire codé, appelé *bit-stream*, pour chaque *code-block*. Ces flux binaires ont la particularité d'être fractionnables en plusieurs segments qui chacun ajoutent un niveau de raffinement lorsqu'ils sont décodés dans l'ordre.

Chaque *bit-stream* est donc tronqué à des points précis qui sont dictés par EBCOT. Le flux binaire d'un *code-block* est de ce fait constitué de plusieurs contributions qui apportent chacune leur incrément de qualité au sens de la minimisation de la distorsion. De façon à construire un

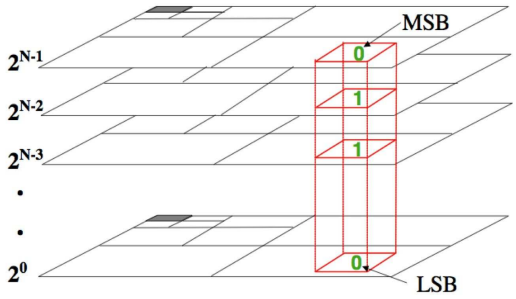


FIG. 3.4: Quantification scalaire embarquée et plans de bits.

flux codé pour la totalité de l'image (ou de la tuile), le codeur mélange les contributions de tous les *code-blocks* en respectant l'ordre d'importance des contributions. Les premières contributions de chaque *code-block* forment un flux qu'il est possible de décoder pour obtenir l'image complète mais avec une qualité faible, il s'agit de la première couche de qualité. Les contributions suivantes de chaque *code-block* forment une seconde couche apportant un incrément de qualité à la totalité de l'image, et ainsi de suite jusqu'à ce que toutes les contributions de *code-blocks* soient incluses dans une couche de qualité. L'image (ou la tuile) peut alors être décodée avec une qualité d'autant plus grande que le nombre de couches pris en compte est important. Les contributions incluses dans une couche sont choisies par EBCOT pour minimiser la distorsion et le volume de données, il est de ce fait possible que certains *code-blocks* ne contribuent pas à toutes les couches.

La totalité du flux binaire codé est ordonnée lors de la deuxième étape du codage (voir Fig. 3.1) pour former le *code-stream*. Ce dernier est ensuite encapsulé dans un fichier au format JP2 selon les spécifications du premier volet de la norme. Ce format de fichier ajoute un certain nombre d'informations comme l'espace colorimétrique (sRGB, adobe RGB, etc) mais aucune de ces informations n'est requise par le décodeur, l'espace RGB standard sera supposé si l'information n'est pas précisée. En effet, JJ2000 qui est l'implémentation java de référence spécifiée dans la cinquième partie de la norme, est en mesure de produire le *code-stream* en l'encapsulant ou non dans un fichier JP2. Réciproquement, le décodeur est capable de reconstruire l'image depuis le *code-stream* seul ou bien depuis un fichier JP2 complet.

3.2 Organisation du *code-stream*

Le *code-stream* contient toutes les informations nécessaires au décodage de l'image compressée. Il contient donc le flux binaire codé relatif aux différents *code-blocks* mais aussi des informations nécessaires au décodage, notamment la taille de l'image, l'organisation du *code-stream*, le nombre de transformées en ondelettes successives, le nombre de couches de qualité, ... Ces informations sont spécifiées dans des en-têtes, tandis que le code en lui même (données relatives aux coefficients codés) est rangé dans des paquets JPEG2000 qui ne doivent pas être confondus avec les paquets de niveau transport qui sont envoyés sur ce réseau. L'ensemble des paquets JPEG2000 est appelé *pack-stream*. Le codeur produit le *code-stream* de manière à satisfaire le débit demandé par l'utilisateur sur l'ensemble du *code-stream*, en-têtes et marqueurs compris, et non pas uniquement sur le code compressé relatif aux coefficients de *code-blocks*. Le *code-stream* est organisé hiérarchiquement comme cela est montré sur la figure 3.5. Il est au moins constitué d'un en-tête principal et d'une partie relative aux tuiles

dans laquelle se situent les paquets JPEG2000. Les informations contenues dans les en-têtes de chaque partie sont accessibles à l'aide de marqueurs spéciaux. Le tableau 3.1 répertorie les marqueurs les plus importants. Il indique le code hexadécimal permettant de les identifier ainsi que leur localisation et le caractère obligatoire ou non de leur présence.

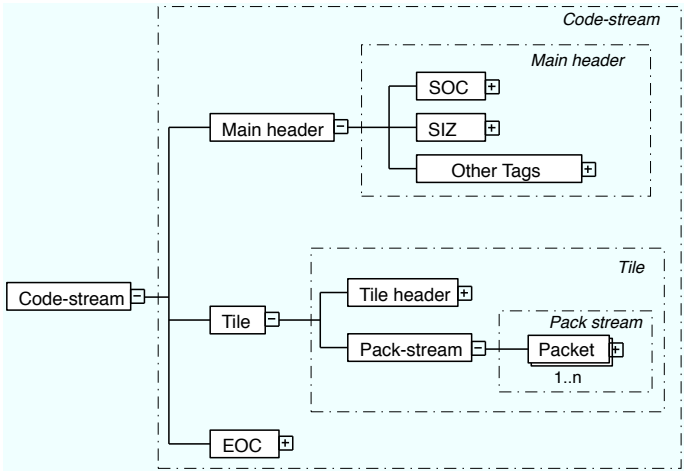


FIG. 3.5: Organisation hiérarchique du *code-stream*, les informations présentes dans les en-têtes sont nécessaire au décodage du code situé dans les paquets.

3.2.1 Ordre de progression du *code-stream*

JPEG2000 définit cinq progressions différentes pour organiser le *code-stream*. La progression utilisée est indiquée par le champ O_p du marqueur COD (coding style default) situé dans l'en-tête principal. Chaque organisation consiste à ordonner les paquets dans un ordre spécifique de façon à permettre au récepteur d'afficher progressivement l'image au fur et à mesure que les données sont décodées. La figure 3.6 montre les progressions par résolution et par qualité qui sont les deux organisations les plus répandues. Les progressions possibles sont les suivantes :

- LRCP (couche de qualité, résolution, composante, position), aussi appelée progression par qualité, c'est le choix par défaut et le plus couramment employé. Les paquets sont alors ordonnés par couche de qualité, dans l'ordre des résolutions croissantes au sein de chaque couche de qualité, dans l'ordre des composantes au sein de chaque résolution, puis les contributions relatives à toutes les positions sont incluses au sein de la composante, ($O_p = 0$). L'ordre d'inclusion des paquets est représenté par l'algorithme suivant :

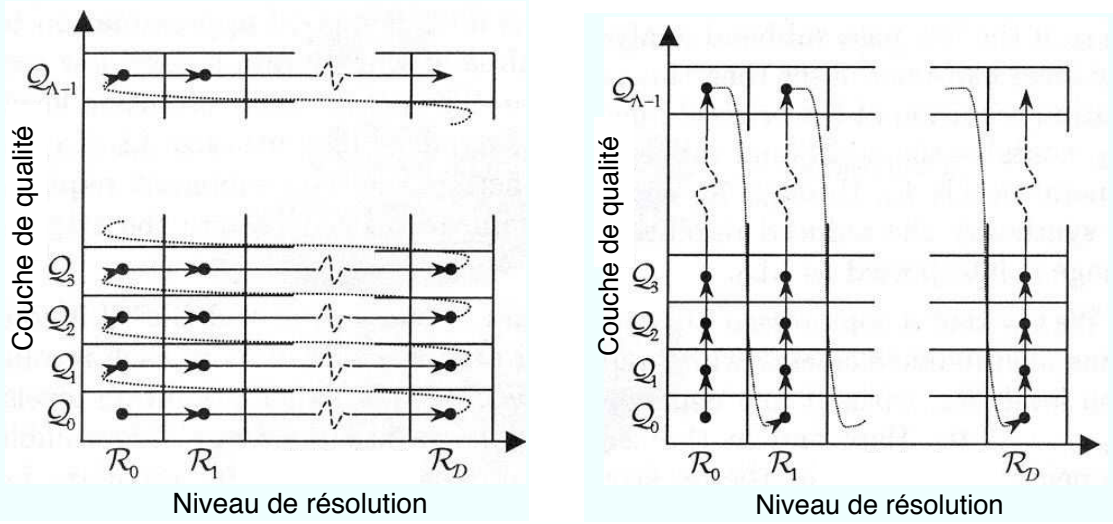
```
pour chaque couche de qualité
   pour chaque résolution
      pour chaque composante
         pour chaque precinct
            inclure paquet[q,r,c,p]
```

- RLCP, aussi appelé progression par résolution, toutes les données relatives à la résolution de base sont suivies des données permettant d'augmenter la résolution d'un facteur 2, et ainsi de suite jusqu'à la résolution native de l'image. Pour chaque résolution les paquets sont rangés selon les couches de qualité croissantes, ($O_p = 1$).
- RPCL, ($O_p = 2$).
- PCRL, progression par position, ($O_p = 3$).

TAB. 3.1: Les principaux marqueurs du *code-stream*. Ceux qui apparaissent isolés (non inclus dans un autre marqueur) sont annotés par *. Ceux dont la valeur dépasse $FF8F_h$ (de façon à ne pas être confondus avec les données compressées du *pack-stream* dans lequel ils peuvent éventuellement résider) sont indiqués par †. Les locations possibles du marqueur sont : M (*Main header*), T (*Tile header*), P (*Tile-part header*), S (dans le *pack-stream*) et E (*End of code-stream*).

Mnemonic	*Valeur*	*Nom*	*Localisation*	*Obligatoire*
SOC^*	$FF4F_h$	Début du *code-stream*	M	oui
$SOT^\dagger$	$FF90_h$	Début d'une tuile	T,P	oui
$SOD^{\dagger *}$	$FF93_h$	Début des données	T,P	oui
$EOC^{\dagger *}$	$FFD9_h$	Fin du *code-stream*	E	oui
SIZ	$FF51_h$	Taille image et tuiles	M	oui
COD	$FF52_h$	Paramètres du codage	M,T	oui
COC	$FF53_h$	Codage des composantes	M,T	non
QCD	$FF5C_h$	Paramètres de quantification	M,T	oui
QCC	$FF5D_h$	Quantification des composantes	M,T	non
RGN	$FF5E_h$	Région d'intérêt	M,T	non
POC	$FF5F_h$	Changement d'ordre de progression	M,T,P	non
TLM	$FF55_h$	Longueurs des *tile-part* lorsque situés dans l'en-tête principal	M	non
PLM	$FF57_h$	Longueurs des paquets lorsque situés dans l'en-tête principal	M	non
PLT	$FF58_h$	Longueur des paquets	T,P	non
PPM	$FF60_h$	En-têtes de paquets lorsque situés dans l'en-tête principal	M	non
PPT	$FF61_h$	En-têtes de paquets dans le *tile-part*	T,P	non
$SOP^\dagger$	$FF91_h$	Début de paquet	S	non
$EPH^{\dagger *}$	$FF92_h$	Fin d'en-tête de paquet	S	non
CRG	$FF63_h$	Enregistrement des composantes	M	non
COM	$FF64_h$	Commentaire	M,T,P	non

– CPRL, progression par ordre des composantes, ($O_p = 4$).



(a) Progression par couches de qualité. (b) Progression par résolutions.

FIG. 3.6: Ordonnancement du flux de paquets selon le type de progression choisi.

3.2.2 Paquets JPEG2000

Les paquets JPEG2000 sont composés de deux parties : un en-tête et un corps. Selon les directives de l'utilisateur lors de la compression, l'en-tête peut précéder directement le corps du paquet (cas par défaut, illustré sur la figure 3.7) ou bien tous les en-têtes peuvent être regroupés dans l'en-tête principal du *code-stream* ou encore dans l'en-tête de la tuile. Les corps des paquets sont alors juxtaposés dans le flux de paquets. Un paquet JPEG2000 contient des contributions des *code-blocks* appartenant à un seul *precinct*, une seule couche de qualité, une seule résolution et une seule composante. Chaque *code-block* inclus dans le paquet contribue pour chacune des trois sous-bandes (sauf si le paquet traite la plus petite résolution R_0, auquel cas chaque *code-block* inclus ne contribue que dans la sous-bande LL).

L'en-tête de paquet renseigne chaque contribution de *code-blocks* incluse dans le paquet dans l'ordre d'apparition dans le corps du paquet. Les informations fournies pour chaque contribution de *code-block* incluse sont les suivantes :

– la première couche de qualité à laquelle le *code-block* a contribué pour la première fois,
– le nombre de plans de bits les plus significatifs à zéro,
– le nombre de passes de codage,
– la longueur de la contribution en octets.

Ces informations sont requises par le décodeur et permettent notamment de calculer l'offset de façon à retrouver les octets de code des différentes contributions au sein d'un même paquet.

L'utilisateur peut spécifier l'utilisation de marqueurs optionnels de façon à délimiter les paquets dans le *code-stream*. Ainsi les marqueurs SOP (Start of Packet) et EPH (End of Packet Header) se trouvent respectivement juste devant l'en-tête du paquet, et juste avant le corps du paquet. Nous avons forcé leur présence de manière à délimiter plus facilement les paquets sans tout décoder. Une autre fonctionnalité de ces marqueurs optionnels consiste à permettre la resynchronisation au décodage sur le paquet suivant en cas d'erreur dans le *code-stream*.

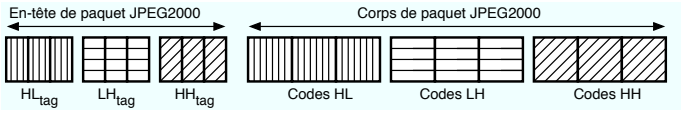


FIG. 3.7: Composition d'un paquet JPEG2000 : l'en-tête renseigne sur les contributions de *code-blocks* présentes, ce qui permet de retrouver le code de ces contributions dans l'ordre des sous bandes HL, LH, HH.

Nous verrons qu'en pratique cette possibilité est loin d'être mise à profit par les décodeurs et qu'une erreur sur un paquet peut conduire au crash du décodeur.

Les informations des en-têtes sont codées pour limiter leur poids, le codage supprime ainsi la plupart des redondances, toujours pour limiter le volume. Cela signifie que des informations relatives à certains *code-blocks* ne sont pas répétées d'un en-tête de paquet à l'autre bien que le dit *code-block* puisse contribuer dans plusieurs paquets à des couches de qualité différentes. La connaissance des informations contenues dans les en-têtes précédents est par conséquent nécessaire à l'interprétation des en-têtes suivants. De plus, le décodeur a besoin de connaître les informations présentes dans les différents marqueurs de l'en-tête principal afin de savoir à quoi s'attendre lorsqu'il parcourt l'en-tête d'un paquet. Ces informations nécessaires qui ne figurent pas dans l'en-tête de paquet mais dans la partie SIZ de l'en-tête principal sont notamment la taille de l'image et des tuiles éventuelles, le nombre de composantes, la taille des *precincts* et des *codes-blocks*.

Prenons un exemple réel : soit l'image source Lena de 512 $\times$ 512 pixels à 256 niveaux de gris. L'image non compressée au format PGM est codée par JJ2000 avec les paramètres utilisateurs suivants : progression LRCP, 3 résolutions et 5 couches de qualité demandées, ainsi que la présence des marqueurs SOP et EPH. Le *code-stream* produit est donc composé de 3 résolutions et 5 couches de qualité. Nous savons en outre que seule une composante est présente (luminance : niveaux de gris). Par défaut le codeur ne divise pas l'image en tuiles, et comme les *precincts* ont une taille par défaut de 2^{15} pixels de côté, il faut s'attendre finalement à la présence d'une seule tuile et un seul *precinct* par résolution.

Lors de l'opération de reconstruction de l'image à partir du *code-stream*, le décodeur a accès à ces informations directement ou non grâce à différents marqueurs. Après avoir déterminé que l'image comporte une seule composante (champ Csiz du marqueur SIZ), une seule tuile, et que la taille maxi des *precincts* est telle qu'aucune sous-bande n'en requiert plusieurs. Les champs du marqueur COD indiquent ensuite que la progression est LRCP (lecture du champ $O_p = 0$), le nombre de niveaux dans la décomposition dyadique par transformées en ondelettes (fourni par le marqueur COD : 2 niveaux donc 3 résolutions), le nombre de couches de qualité (5), que les *code-blocks* ont une taille de $J_x^{t,c} \times J_y^{t,c}$ telle que :

$$J_x^{t,c} = 4 \times 2^{E_2^{CB}}$$

$$J_y^{t,c} = 4 \times 2^{E_1^{CB}}$$

La lecture des champs correspondants du marqueur COD indique $E_1^{CB} = E_2^{CB} = 4$, d'où des *code-blocks* de 64$\times$64 coefficients. À ce stade, le décodeur sait qu'il doit s'attendre à trois résolutions R_0, R_1 et R_2. R_2 et R_1 sont composées de trois sous-bandes tandis que R_0 n'en contient qu'une (LL). Les sous-bandes relatives à R_2 contiennent chacune 256^2 coefficients, soit 16 *code-blocks* ; 128^2 coefficients pour chaque sous-bande de R_1 et R_0, soit 4 *code-blocks*.

Sachant que l'image n'est pas divisée en tuiles, que le nombre de couches de qualité est l, avec r résolutions, c composantes and p *precincts*, alors le décodeur s'attend à trouver $l \times r \times$

$c \times p$ paquets, 15 en l'occurrence. L'accès au paquet N relatif à la couche L_i, la résolution R_j, la composante C_k et le *precinct* P_m est alors déterminé par la formule 3.1 pour une progression LRCP.

$$N = i(r \times c \times p) + j(c \times p) + k(p) + m \tag{3.1}$$

Réciproquement, le décodeur déduit d'après la formule 3.2, les numéros de couche, résolution, composante et *precinct* d'un paquet en fonction de son rang N.

$$\begin{aligned} i &= \left\lfloor \frac{N}{rcp} \right\rfloor \\ j &= \left\lfloor \frac{N - i(r \times c \times p)}{c \times p} \right\rfloor \\ k &= \left\lfloor \frac{N - i(r \times c \times p) - j(c \times p)}{p} \right\rfloor \\ m &= N \bmod p \end{aligned} \tag{3.2}$$

Poursuivons l'exemple précédent du *code-stream* issu de la compression de *Lena* 512×512 pixels, 256 niveaux de gris, 3 résolutions et 4 couches avec le premier paquet relatif à la couche 0, à la résolution 0, à la composante 0 (unique) et au *precinct* 0 (unique). Rappelons que cette résolution ne comprend qu'une seule sous-bande (LL), et quatre *code-blocks*.

L'en-tête du premier paquet (paquet 0) est : E3 F0 FD 13 7E 2F 66 5F 7D 4F 7E 2F 67 20. Le décodage des informations concernant la contribution du premier *code-block* s'effectue de la manière suivante :

$$\underbrace{111}_{LI}\ \underbrace{00011}_{ZBP}\ \underbrace{111100001}_{CP}\ \underbrace{111110}_{L_{Block}}\ \underbrace{1000100110}_{Longueur}$$

LI le *code-block* contribue pour la première fois dans cette couche. Le codage de ce champ fait intervenir l'algorithme *tag-tree* distribué sur toutes contributions de ce *code-block* (à toutes les couches).

ZBP le nombre de plans de bits à zéro (depuis le MSB) est 3. Le codage de cette valeur utilise aussi le *tag-tree*, cependant contrairement à LI, le codage n'est pas distribué ici (le code relatif à ZBP est toujours entièrement présent dans l'en-tête de chacune des contributions du *code-block*).

CP le nombre de passes de codages incluses pour cette contribution de *code-block* est 7 (cf. table du codage présente dans (JTC1/SC29, 2004*b*)).

L_{Block} la valeur de L_{Block} est augmentée de 5, comme il n'y a pas de valeur précédente mémorisée pour ce *code-block*, la valeur par défaut est 3, soit $L_{Block} = 8$ à présent. La longueur de la contribution est codée sur $L_{Block} + \lfloor \log_2(CP) \rfloor$, soit 10 bits.

$Longueur$ codage binaire de la longueur en octets de la contribution, soit 550 octets.

Le décodage de l'en-tête de ce paquet se poursuit pour les contributions des trois autres *codes-blocks* du paquet. L'en-tête se termine éventuellement par un alignement sur l'octet qui se traduit par un bourrage de bits à zéros. L'accès au code proprement dit de chacune des contributions dans le corps du paquet s'effectue grâce à la connaissance de leur longueur.

Dans un premier temps nous avons déterminé le caractère requis ou optionnel des différents composants du *code-stream*. Les différentes informations dont l'impact de la perte sur le décodage doit être évalué sont les en-têtes et leurs différents marqueurs d'une part (en-tête principal, de tuiles, de paquets), et les corps de paquets JPEG2000 d'autre part.

3.3 Tolérance aux pertes du *code-stream*

Dans cette section, nous voulons établir quelles sont les informations du *code-stream* qui sont obligatoires pour la bonne marche du décodage de l'image. Une information est obligatoire si le décodeur n'est pas en mesure de reconstruire une image lorsque l'information est manquante ou erronée. Si le décodeur parvient à reconstruire l'image, deux possibilités se présentent alors : soit tous les décodeurs affichent la même image, l'information manquante ou erronée est alors considérée optionnelle, soit l'image reconstruite est dépendante du décodeur, l'information est alors considérée obligatoire.

Nous avons développé des outils en C qui permettent d'extraire automatiquement les différentes informations du *code-stream*. Ces outils détectent les différents marqueurs, les en-têtes et les paquets et lisent les champs des en-têtes. Les pertes de données sont simulées à l'aide d'un outil capable de supprimer une partie précise à l'intérieur du *code-stream* ou bien de remplacer ces données par d'autres selon les directives de l'utilisateur.

3.3.1 *Code-stream* et latitude de perte

L'objectif étant de distinguer les parties du code JPEG2000 qui sont obligatoires (donc devant absolument être rangées dans des p-paquets) de celles optionnelles, donc susceptibles d'être rangées dans des s-paquets avec le protocole 2CP-ARQ, nous avons étudié les effets des pertes de données avec des *code-streams* réels. Le codeur que nous avons utilisé est JJ2000, (EPFL, 2002), inclus dans l'implantation logicielle java de référence développée conjointement par l'École Polytechnique Fédérale de Lausanne, Canon et Ericsson. Le type de progression spécifié pour la compression est la progression par qualité (LRCP) qui est la plus courante. Nous avons utilisé l'image de référence *Lena*, en 512×512 pixels et 256 niveaux de gris. L'image est compressée par JJ2000 en faisant varier le débit, le nombre de couches et de résolutions, les marqueurs SOP et EPH ont été forcés. Deux versions du produit de la compression sont construites à chaque itération : un fichier qui contient le *code-stream* pur, mais aussi un fichier JP2 compatible avec le standard. Le fichier JP2 est en effet requis par les applications compatibles JPEG2000 commerciales. JJ2000 et Jasper étant les applications logicielles de référence, ce sont des outils qui intègrent toutes les fonctionnalités offertes par le standard et sont de plus en mesure de produire et décoder le *code-stream* pur. Les applications compatibles JPEG2000 utlisées pour le décodage sont JJ2000 et Jasper, mais aussi deux applications commerciales qui sont Preview (le lecteur multi-format intégré au système OS X Apple) et Irfan View sur Microsoft Windows.

Nous avons procédé à la suppression de chacun des en-têtes du *code-stream*, des en-têtes de paquets et des corps de paquets avant de tenter le décodage et la reconstruction de l'image. Les résultats sont présentés dans le tableau 3.2. Il s'avère que la suppression de l'en-tête principal rend le *code-stream* invalide et conduit à l'échec du décodage. Aucun des décodeurs utilisés n'a été en mesure de reconstruire une image. Ce résultat était attendu puisque les informations de tailles et les paramètres (nombre de couches, résolutions...), nécessaires à l'interprétation du code compressé proprement dit, se situent à cet endroit. Le code de l'image compressée se situe quant à lui dans les paquets JPEG2000. L'analyse de leur perte révèle que les décodeurs peuvent se passer d'une partie de ces informations sous certaines conditions très strictes.

Comme c'est le cas pour l'en-tête principal, la suppression des en-têtes de paquets mène à l'échec du décodage. Ce comportement s'explique parce que les informations concernant le contenu du paquet, son numéro ainsi que sa taille sont alors manquantes, et le décodeur ne

peut pas interpréter le paquet. Du fait du codage *tag-tree* des informations qui vise à distribuer les informations relatives aux contributions de *code-blocks* dans les différents en-têtes de paquets auxquels ces contributions participent, il est également possible que les informations d'un paquet ne puissent être entièrement décodées même lorsque son en-tête est présent car une information située dans un en-tête précédent est manquant. Cependant, cela montre aussi que les décodeurs ne mettent pas à profit la resynchronisation possible grâce à la présence des marqueurs SOP et EPH. Par contre le contenu des paquets peut subir certaines pertes. Si un ou plusieurs corps de paquets sont purement supprimés, certaines applications peuvent parfois parvenir à reconstruire une image (Preview et Irfan View), mais les décodeurs officiels échouent. Lorsque les applications parviennent à reconstruire une image, on remarque que le résultat est variable d'un décodeur à l'autre pour un *code-stream* identique.

La perte d'information n'est cependant pas obligatoirement synonyme de suppression. Il est possible, sous certaines conditions, de remplacer les données perdues par d'autres données issues d'une reconstruction intelligente (dissimulation d'erreur) ou non. C'est pourquoi nous avons substitué les différents en-têtes et des paquets par des données fausses avant de procéder au décodage. En ce qui concerne les corps de paquets JPEG2000, cette opération est toujours réalisable en cas de perte du moment que l'en-tête correspondant est disponible puisque la longueur des informations manquantes y figure. Les données de substitution peuvent consister en toute suite d'octets tant que deux octets consécutifs n'ont pas une valeur supérieure à 0xFF8F. Ceci est du au mécanisme de « bit-stuffing » du codeur JPEG2000, l'objectif étant de réserver les valeurs supérieures ou égales à 0xFF90 pour les marqueurs du *code-stream*, (Taubman et Marcellin, 2002). Pour cette expérimentation, le corps de paquet JPEG2000 a été substitué par des bits à 0. Dans ces conditions les quatre décodeurs utilisésreconstruisent une image identique. L'image est certes dégradée, et cette dégradation dépend du paquet substitué et des données de substitution, mais pas du décodeur. Le *code-stream* JPEG2000 supporte par conséquent les pertes de paquets, à condition que des données soient substituées aux corps de paquets perdus, que la longueur des données de substitution respecte celle des données originelles, et que ces données de substitution ne contiennent pas de sous-chaînes correspondant à des marqueurs ($>$0xFF8F).

Les corps de paquets peuvent donc faire l'objet de pertes par le réseau du moment que le récepteur est capable de reconstruire un substitut ayant la même longueur. Ces pertes ne peuvent cependant pas être acceptées à n'importe quel prix, il est donc nécessaire de s'intéresser à la dégradation engendrée par cette substitution. La section suivante traite par conséquent de l'étude de l'impact de la substitution de corps de paquets sur la qualité de reconstruction des images.

3.3.2 Impact de la substitution de données sur la qualité de l'image décodée

Selon la définition de la progression par qualité (LRCP), les paquets sont ordonnés par ordre croissant des couches. Chaque couche apporte une amélioration de la qualité au sens du rapport signal sur bruit. Tout porte donc à penser que lorsque le *code-stream* est organisé selon la progression par qualité LRCP, les informations sont ordonnées selon leur importance décroissante. Autrement dit, cela signifierait que les données les plus critiques du point de vue de la qualité de l'image reconstruite (au sens du SNR) sont situées dans les paquets JPEG2000 de tête, les derniers paquets n'apportant qu'un raffinement de qualité beaucoup moins sensible. En réalité, nos résultats vont montrer que l'importance des données d'un paquet JPEG2000 ne dépend pas seulement de sa position dans le *pack-stream*. D'autres facteurs vont devoir être pris en considération. Cela est présenté ci-dessous.

TAB. 3.2: Impact des erreurs dans le *code-stream* selon l'information touchée et le décodeur utilisé.

Information erronée	Décodage et reconstruction			
	JJ2000	Jasper	Preview	Irfan View
En-tête principal (effacement)	impossible	impossible	impossible	impossible
En-tête principal (substitution)	impossible	impossible	impossible	impossible
En-tête de paquet (effacement)	impossible	impossible	impossible	impossible
En-tête de paquet (substitution)	impossible	impossible	impossible	impossible
Corps de paquet (effacement)	impossible	impossible	variable	variable
Corps de paquet (substitution)	OK	OK	OK	OK

3.3.2.1 Impact de la valeur de substitution des corps de paquets

Les données des paquets JPEG2000 sont toujours alignées à l'octet, nous les substituons par des octets tous identiques. Deux octets consécutifs ne doivent pas excéder la valeur 0xFF8F pour ne pas être confondus avec un marqueur (cf. section 3.3.1). Pour satisfaire cette contrainte, les octets de remplacement que nous choisissons sont compris dans l'intervalle [0x00, 0xFE].

L'image de référence utilisée est Lena 512×512 pixels à 256 niveaux de gris. La compression a été effectuée avec JJ2000 pour obtenir un débit binaire de 1b/pixel pour le *code-stream*, une progression LRCP avec la commande suivante :

```
java JJ2KEncoder -i lena512.pgm -rate 1 -Psop -Peph -Aptype layer -file_format
-o lena1bpp.jp2
```

Le *code-stream* résultant est composé de 6 résolutions, 19 couches de qualité, 1 composante, soit au total 114 paquets. L'image issue de sa reconstruction a un PSNR[1] de 40,47dB.

La figure 3.8 montre la dégradation de l'image en terme de PSNR lorsqu'un seul paquet est substitué, en fonction du paquet concerné et de la valeur de substitution des octets de données. La figure 3.9 est une coupe de la courbe précédente, elle correspond au paquet 87. Nous remarquons que la valeur des octets de substitution a une influence imprévisible sur la reconstruction de l'image. Comme le montrent les figures 3.8 et 3.9, aucune valeur ne semble préférable *a priori*. La figure 3.9 est un résultat typique, transposable aux autres paquets bien que la variance puisse être différente.

Les valeurs qui permettent les meilleures reconstructions du point de vue PSNR varient en fonction du paquet et de l'image considérée. Pour cette raison, nous avons choisi la valeur

[1]Peak Signal to Noise Ratio (rapport de signal sur bruit écrêté). Pour des images codées avec 8 bits/pixel :

$$PSNR = 10 \log_{10} \frac{255^2}{\frac{1}{N_x . N_y} \sum_{x=0}^{N_x - 1} \sum_{y=0}^{N_y - 1} \left(I_{x,y} - \widehat{I}_{x,y} \right)^2}$$

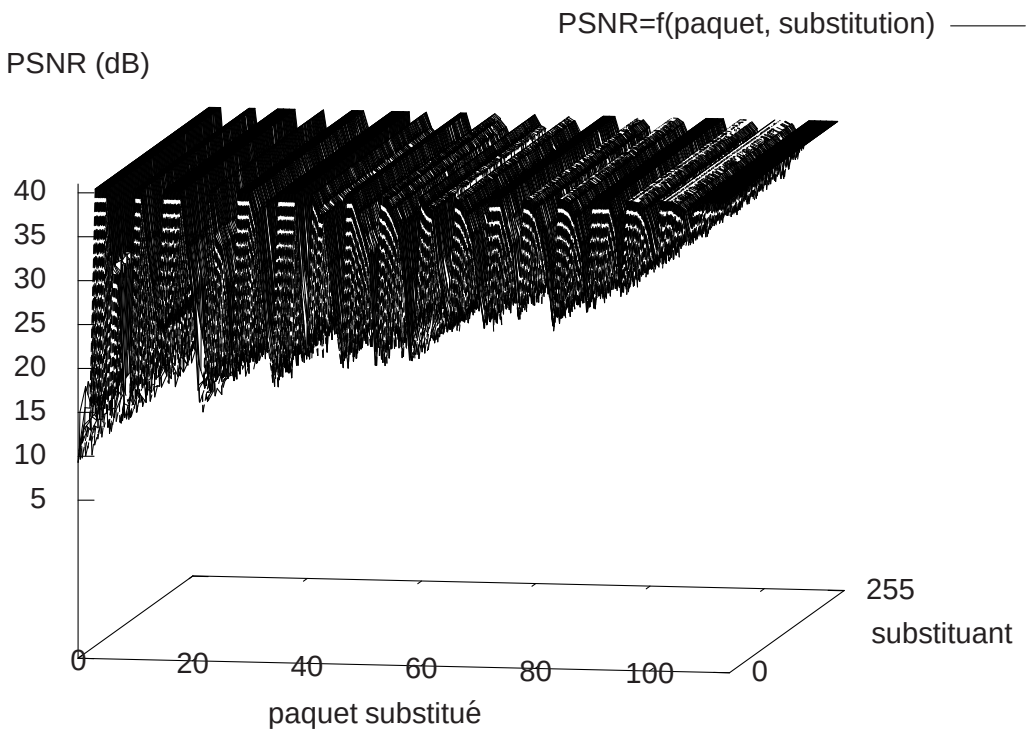


FIG. 3.8: PSNR de l'image reconstruite en fonction du paquet substitué et de la valeur de substitution.

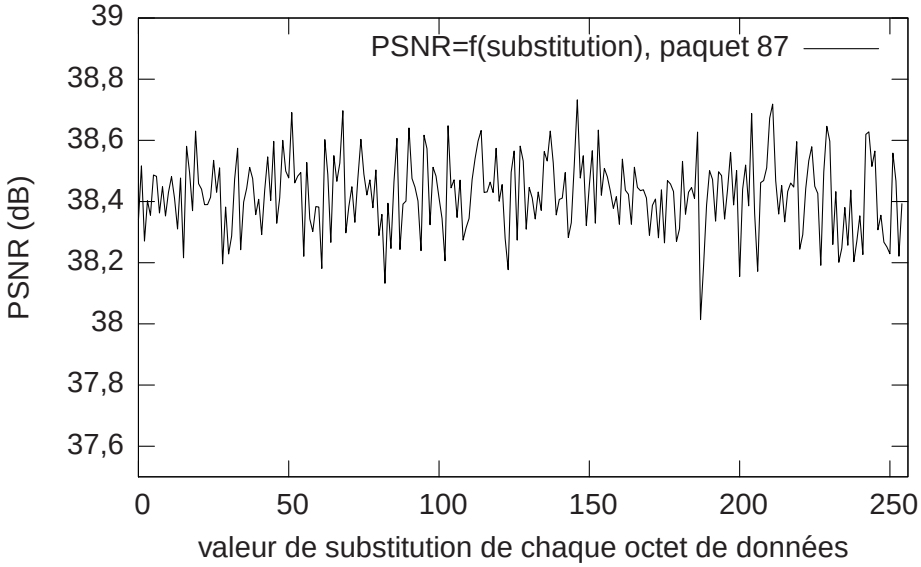


FIG. 3.9: PSNR de l'image reconstruite en fonction de la valeur de substitution des octets de données du paquet numéro 87.

de substitution 0x00 arbitrairement. Du point de vue du récepteur qui ne connaît pas l'image qu'il reçoit, il n'est donc pas possible de choisir une valeur de substitution optimale *a priori*. Par contre, une reconstruction du code perdu en fonction des données restantes selon un procédé de dissimulation d'erreur serait envisageable. Or une telle dissimulation d'erreur demande une étude complète du codeur MQ de manière à prévoir quelle serait la reconstruction en fonction des données fausses introduites déborde du cadre de ce travail.

Notons que si la valeur de substitution ne semble pas affecter le PSNR dans de grandes proportions et de façon prévisible, il en va tout autrement concernant le paquet substitué.

3.3.2.2 Impact de la substitution des paquets JPEG2000 sur la qualité de l'image décodée

Un paquet contient des contributions de *code-blocks* qui concernent chacune une couche, une résolution et un *precinct* uniques. La longueur du paquet peut être nulle, dans ce cas il ne comporte pas de donnée et la substitution n'affecte pas le PSNR. Sur la figure 3.8 on remarque que le PSNR de l'image reconstruite est de 40,47dB (le PSNR original) lors de certaines substitutions de paquets, il s'agit alors typiquement de paquets de longueur nulle.

La substitution des paquets contenant des données implique systématiquement une baisse du PSNR. Ce dernier varie de 9,3 à 40,44dB selon le paquet remplacé (voir figure 3.11). Il est intéressant de remarquer qu'en progression LRCP, la dégradation de PSNR tend à se réduire au fur et à mesure que le paquet substitué (reconstruit avec des octets à zéro) est situé en avant dans le *code-stream*. On ne peut cependant pas pour autant affirmer que les paquets sont ordonnés par ordre d'importance décroissante en regard de la dégradation occasionnée par leur perte, la courbe 3.11 comportant suffisamment de contre-exemples pour corroborer cette affirmation. Par exemple la substitution du paquet 13 (55 octets) entraîne un PSNR de 30,28 dB (Fig. 3.10(b)) alors que la substitution du paquet 40 (21 octets) entraîne un PSNR de 21,66 dB (Fig. 3.10(c)). Une substitution portant sur un volume plus important de données n'entraîne pas non plus une dégradation plus forte de la qualité, les figures 3.10(d), 3.10(e) et

3.10(f) en témoignent.

(a) Reconstruction à partir du *code-stream* intact, PSNR=40,46 dB.

(b) Substitution du paquet 13 (55 octets), PSNR=30,28 dB.

(c) Substitution du paquet 40 (21 octets), PSNR=21,66 dB.

(d) Substitution du paquet 52 (195 octets), PSNR=21,14 dB.

(e) Substitution du paquet 82 (1111 octets), PSNR=32,82 dB.

(f) Substitution du paquet 101 (2816 octets, PSNR=33,43 dB.

FIG. 3.10: Effets de la substitution d'un paquet sur les qualités visuelles objective et subjective de la reconstruction de l'image *Lena* JP2 1 bit/pixel (6 résolutions, 19 couches, 1 composante, 114 paquets).

Notons que la tendance qu'a la progression LRCP à organiser les paquets par ordre décroissant des impacts de leur perte mais de façon non stricte n'est pas liée à l'image *Lena*, au débit binaire moyen de sortie ou au nombre de paquets. La courbe 3.12(a) montre en effet les mêmes tendances pour l'image *Peppers* encodée à 2 bits/pixel, toujours avec une composante, 6 résolutions mais 21 couches au lieu de 19, de même pour l'image couleur *Rome* (tirée du CDROM fourni avec l'ouvrage de Taubman et Marcellin (Taubman et Marcellin, 2002) encodée à 2bpp avec 17 couches, 3 composantes et 6 résolutions, soit 306 paquets (voir courbe 3.12(b)).

La progression LRCP est prévue pour optimiser l'apport en SNR de chaque couche de qualité, et donc de chaque paquet supplémentaire. Si elle ne permet pas exactement de déduire la dégradation engendrée par la perte d'un paquet en fonction de son rang, la progression LRCP est néanmoins la plus régulière à cet égard. La progression par résolution (RLCP) modifie l'ordre des paquets de manière à regrouper toutes les couches d'une même résolution avant de traiter la résolution suivante. Les résultats en termes de dégradation en fonction du rang des paquets perdus sont en effet encore moins prévisibles (voir courbe 3.13).

Rappelons que les informations perdues et non récupérées lors du transport à fiabilité par-

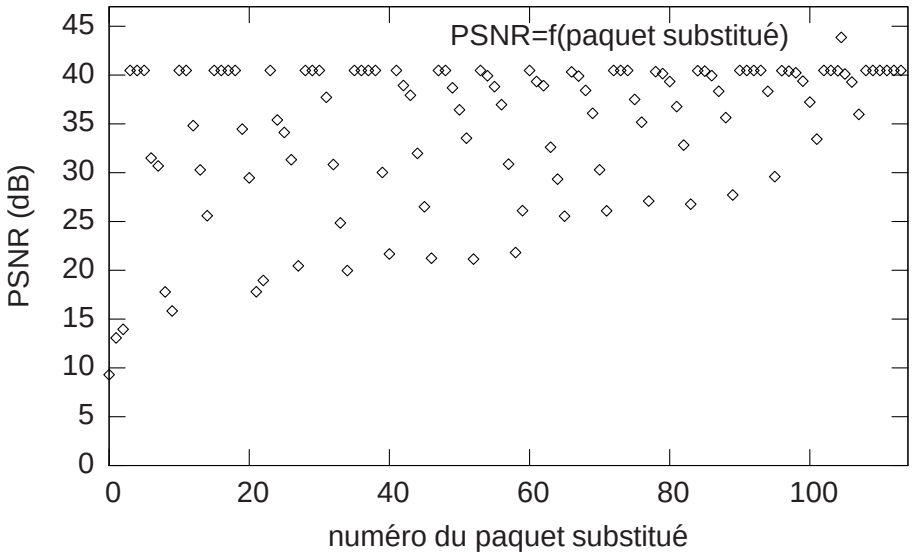
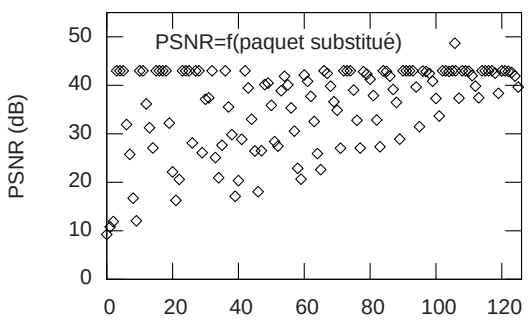
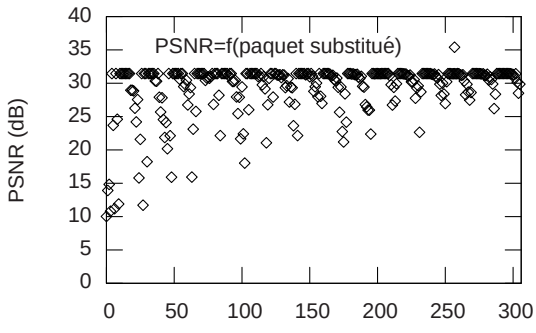


FIG. 3.11: Progression LRCP : PSNR en fonction du numéro du paquet substitué de l'image *Lena* à 1bit/pixel (6 résolutions, 19 couches, 1 composante, 114 paquets).



(a) PSNR en fonction du numéro du paquet substitué de l'image *Peppers* à 2bits/pixel (6 résolutions, 21 couches, 1 composante, 126 paquets).



(b) PSNR en fonction du numéro du paquet substitué de l'image *Rome* à 2bits/pixel (6 résolutions, 17 couches, 3 composantes, 306 paquets).

FIG. 3.12: Avec la progression LRCP, la dégradation de qualité produite de la perte d'un paquet n'est pas strictement décroissante lorsque la progression par qualité est utilisée, et ce quels que soient l'image, le débit, le nombre de couches et composantes.

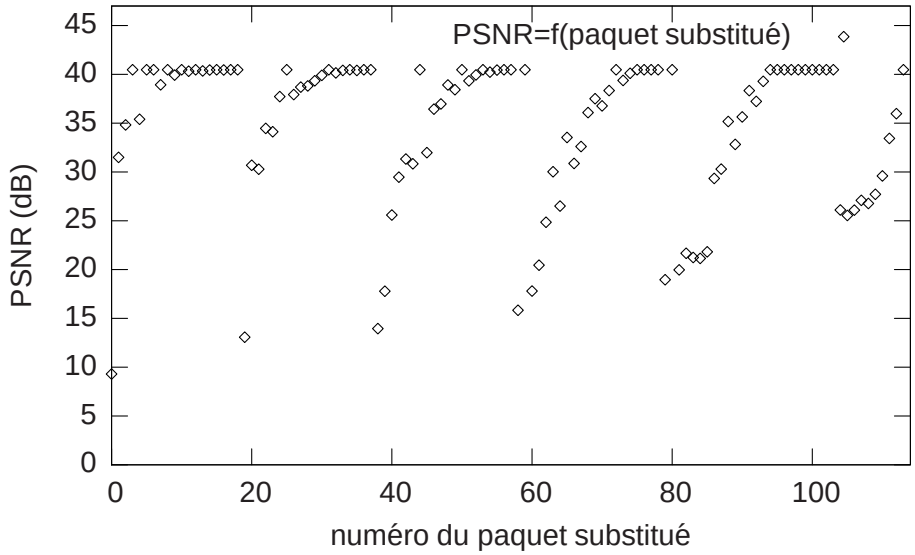


FIG. 3.13: Progression RLCP : PSNR en fonction du numéro du paquet substitué de l'image *Lena* à 1bit/pixel (6 résolutions, 19 couches, 1 composante, 114 paquets).

tielle seront remplacées par de fausses données de substitution. Il convient donc de choisir les données non fiabilisées (transportées par les s-paquets de 2CP-ARQ) en fonction de la dégradation que leur perte provoque. Il nous faut donc trouver un critère plus pertinent que le rang du paquet dans un *code-stream* organisé LRCP pour sélectionner les informations à protéger.

Pour ce faire, la section suivante présente une étude de la dégradation impliquée par la substitution des données en fonction des variables et connaissances dont nous disposons malgré les données manquantes.

3.4 Analyse de la sensibilité aux pertes

Nous avons déjà montré que les pertes d'information ne peuvent porter que sur le contenu des paquets JPEG2000. Le préjudice des pertes de paquets JPEG2000 en termes de PSNR n'est pas strictement décroissant en fonction du rang croissant des paquets en LRCP. Nous cherchons maintenant à déterminer la dégradation qu'entraîne la perte d'un paquet. Cette étape devra être réalisée rapidement par l'émetteur de manière à classer les paquets JPEG2000 selon le niveau de dégradation de qualité provoquée par leur perte. Une fois cette classification effectuée, l'émetteur est en mesure de réaliser une mise en paquets adéquate de manière à minimiser le nombre de p-paquets tout en autorisant une reconstruction de l'image avec une dégradation mesurée en réception. Cette opération doit être rapide, il est donc inconcevable pour l'émetteur de mesurer la dégradation introduite par la perte de chaque paquet JPEG2000 de l'image pour des raisons évidente de coût du calcul des multiples décodages. Par contre, les en-têtes de paquets JPEG2000 sont toujours disponibles et les informations contenues peuvent être extraites à moindre coût.

La difficulté consiste à établir un lien entre les informations contenues dans les en-têtes des paquets JPEG2000 et l'importance des paquets en termes de dégradation de PSNR que leur perte (leur substitution plus précisément) introduirait. Si ce lien est établi, il sera alors possible

de rendre automatique la classification des paquets JPEG2000 en p- et s-.

3.4.1 Analyse des données des en-têtes de paquets JPEG2000

Dans cette section nous proposons de déterminer le niveau d'importance des paquets JPEG2000 en termes de dégradations qu'entraîne leur perte/substitution. Pour ce faire, nous analysons le contenu des paquets au niveau accessible le plus fin défini par le standard : le *code-block*. Chaque paquet est en effet constitué de contributions de *code-blocks* pour une couche et une résolution données. Pour connaître cette composition, il n'est pas nécessaire de décoder le paquet en totalité, seul le décodage de son en-tête est requis. Ce travail a été effectué en utilisant l'outil en ligne de décodage des en-têtes JPEG2000 « Ludovico JP2 Parser » proposé par le *Multimedia Communications Lab* (MCLab) de l'université de Cagliari. Bien qu'il ne soit plus en ligne à cet instant (novembre 2006), cet outil nous a néanmoins permis de retrouver les paramètres de chaque contribution et paquet.

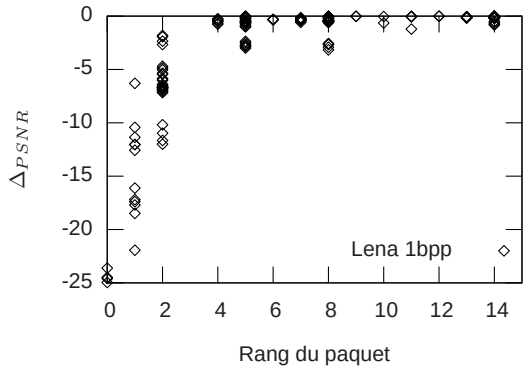
3.4.1.1 Analyse préparatoire

Dans un premier temps, nous avons recueilli les informations présentes dans les en-têtes de chaque paquet de l'image *Lena* 512×512 pixels en niveaux de gris encodée avec un débit cible de 1 bit/pixel, 3 résolutions et 5 couches de qualité. Pour chacune des contributions de *code-block*, les informations lues directement ou reconstruites indirectement sont :

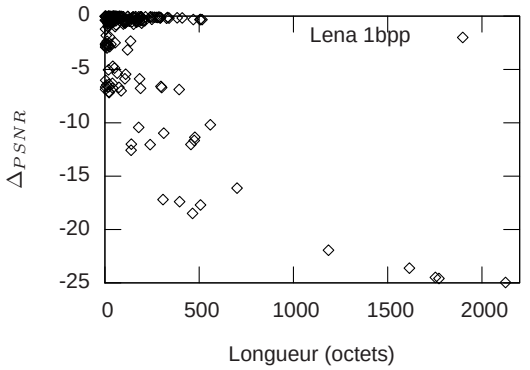
- le paquet auquel la contribution appartient,
- la taille de la contribution en octets,
- la situation dans le fichier (offset),
- la résolution,
- la couche de qualité,
- la sous-bande,
- le nombre de plans de bits les plus significatifs à zéro,
- le nombre de passes de codage,
- la couche à laquelle le *code-block* contribue pour la première fois.

Les figures 3.14 et 3.15 exposent les relations éventuelles pouvant lier la dégradation de PSNR enregistrée lorsqu'une contribution est substituée en fonction de ses différents paramètres que nous avons recueillis dans les en-têtes. La courbe 3.14(a) montre que l'importance d'une contribution n'est pas étroitement liée au rang du paquet auquel elle appartient. Ceci appuie les constatations qui ressortent de l'analyse de la substitution de paquets entiers effectuée en section 3.3.2.2. La longueur de la contribution en octets n'est pas révélatrice de la dégradation de qualité engendrée par sa perte, il semble même que les contributions les plus volumineuses sont les moins importantes (voir Fig. 3.14(b)). Il en va de même pour la résolution (Fig. 3.14(c)) et le nombre de plans de bits les plus significatifs à zéro. Bien qu'une tendance générale donnant plus de poids aux contributions de la plus petite résolution (basse fréquence LL) semble se profiler, la figure 3.14(c) ne montre pas qu'une contribution de la résolution $\mathcal{R}_i$ est nécessairement plus porteuse d'information que la résolution $\mathcal{R}_{i+1}$.

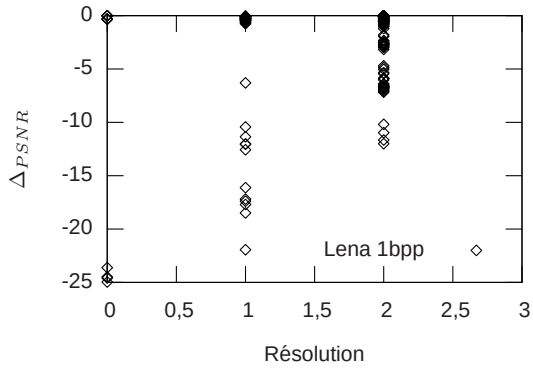
Par contre, la couche de qualité, le nombre de plans de bits à zéro et surtout le nombre de passes de codages semblent beaucoup plus pertinents quant à l'impact entraîné par la perte de la contribution sur la qualité de l'image. Comme le montre la figure 3.14(d), les contributions des couches de qualité les plus basses sont à quelques exceptions près celles dont la perte est la plus préjudiciable du point de vue du PSNR. Sur le même principe, les contributions dont le nombre de plans de bits les plus significatifs à zéro est élevé (ils sont de ce fait non



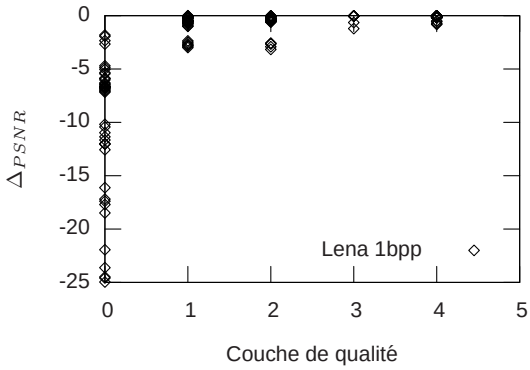
(a) Δ_{PSNR} en fonction du rang du paquet auquel appartient la contribution.



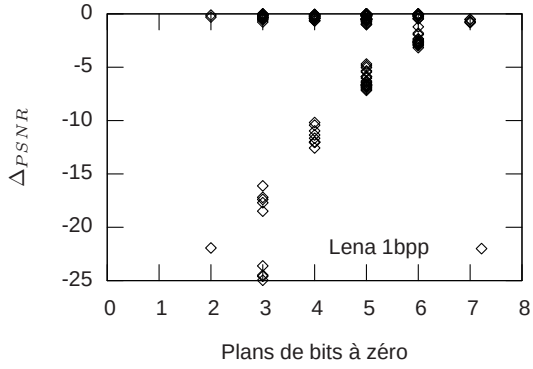
(b) Δ_{PSNR} en fonction de la longueur de la contribution.



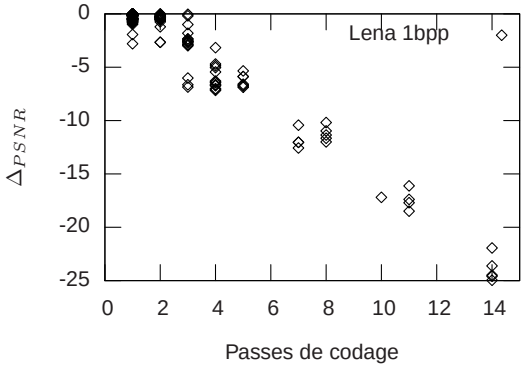
(c) Δ_{PSNR} en fonction de la résolution.



(d) Δ_{PSNR} en fonction de la couche de qualité.

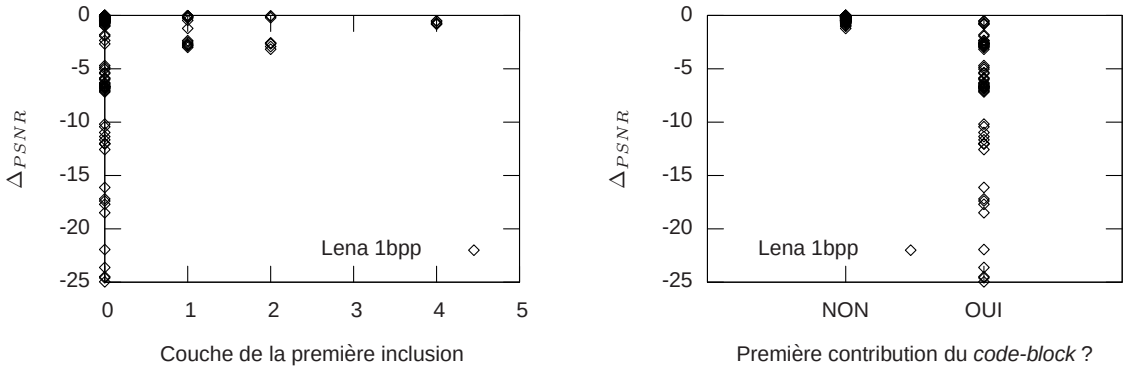


(e) Δ_{PSNR} en fonction du nombre de plans de bits à zéro.



(f) Δ_{PSNR} en fonction du nombre de passes de codage.

FIG. 3.14: Dégradation du PSNR entraînée par la substitution de chacune des 318 contributions de *code-block* en fonction de ses caractéristiques qui sont indiquées dans l'en-tête de paquet. Image *Lena* à 1bits/pixel (3 résolutions, 5 couches, 1 composante, 15 paquets).



(a) Δ_{PSNR} en fonction la couche à laquelle la contribution contribue pour la première fois (*layer inclusion*).

(b) Δ_{PSNR} selon que le *code-block* contribue pour la première fois dans la couche courante ou non.

FIG. 3.15: Dégradation du PSNR entraînée par la substitution de chacune des 318 contributions de *code-block* en fonction de la première couche à laquelle le *code-block* contribue (information de *layer inclusion*). Image *Lena* à 1bits/pixel (3 résolutions, 5 couches, 1 composante, 15 paquets).

significatifs) tend à réduire la dégradation provoquée par leur perte (voir figure 3.14(e)). Le nombre de passes de codages est encore plus déterminant puisque les exceptions sont encore plus rares que lorsque l'on considère la couche de qualité ou le nombre de plans de bits à zéro de la contribution. La courbe 3.14(f) montre en effet une tendance franche de l'augmentation de la dégradation provoquée par la perte d'une contribution avec la croissance du nombre de passes de codage qu'elle contient.

Un *code-block* peut évidemment contribuer dans plusieurs couches de qualité. Il semble logique que plus il contribue tôt, plus il est important. C'est précisément ce qui est indiqué par l'information de *layer inclusion* : la première couche dans laquelle le *code-block* contribue. La courbe 3.15(b) montre en effet cette tendance à la diminution de la dégradation au fur et à mesure que le *code-block* contribue tard dans les couches de qualité. Cela est compréhensible car par définition, plus un *code-block* contribue tard, plus ses contributions seront dans les couches élevées dont la perte est moins préjudiciable nous l'avons vu. L'information de *layer inclusion* permet surtout de savoir pour une contribution donnée si c'est la première fois qu'elle contribue et dans le cas contraire, à quelle point elle est éloignée de la première couche où elle est apparu. Lorsqu'une contribution est la première d'un *code-block* donné, son importance est bien sûr capitale, c'est ce que montre la courbe 3.15(b) en calculant la différence entre la couche de la contribution et la *layer inclusion*. Les premières contributions de *code-blocks* sont donc clairement les plus déterminantes.

Pour aller plus loin nous avons pratiqué une analyse des données recueillies sur l'image *Lena* 512×512 pixels en niveaux de gris pour les débits 0,25bpp, 0,5bpp, 1bpp, 1,5bpp et 2bpp. L'encodage est toujours réalisé par JJ2000 de façon à produire des images compressées composées de 15 paquets 5 couches et 3 résolutions.

3.4.1.2 Analyse en composantes principales

L'objectif est ici d'expliquer la variable Δ_{PSNR} en fonction des variables que nous connaissons et que nous considérons comme des variables explicatives. Les mêmes informations qu'à la section 3.4.1.1 ont été relevées pour chaque contribution de *code-block* incluse dans les

paquets. Les données recueillies sont constituées de 852 contributions non nulles pour l'ensemble des 5 débits qui correspondent aux 5 *code-streams* collectés. L'analyse a été effectuée à l'aide de la plate-forme de calcul scientifique Scilab développée conjointement par l'Institut National de Recherche en Informatique et en Automatique (INRIA) et l'École Nationale des Ponts et Chaussées (ENPC).

Nous notons Y la variable à expliquer et X_i les variables explicatives selon la désignation du tableau 3.3.

TAB. 3.3: Désignation des variables à expliquer et explicatives.

Variable	*Désignation*
Y	Δ_{PSNR}, la dégradation de qualité (variable à expliquer)
X_1	Débit moyen en bits/pixel de la compression
X_2	Rang de la couche à laquelle participe la contribution
X_3	Rang de la résolution à laquelle participe la contribution
X_4	Sous-bande à laquelle appartient la contribution (0, 1, 2 ou 3 pour LL, HL, LH, HH respectivement)
X_5	*Layer inclusion*, couche à laquelle le *code-block* de la contribution apparaît pour la première fois
X_6	Nombre de passes de codage inclus dans la contribution
X_7	Nombre de plans de bits les plus significatifs à zéro
X_8	Longueur de la contribution en octets

Soit D la matrice de données à 852 lignes et 9 colonnes dont les colonnes correspondent aux variables X_1 à X_8 puis Y. On note M la matrice des corrélations associée à D.

$$M = \begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \\ Y \end{array} \begin{array}{c} \begin{array}{ccccccccc} X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 & Y \end{array} \\ \left(\begin{array}{ccccccccc} 1 & & & & & & & & \\ -0,12 & 1 & & & & & & & \\ 0,24 & 0,07 & 1 & & & & & & \\ 0,19 & 0,09 & 0,36 & 1 & & & & & \\ -0,39 & 0,36 & 0,19 & 0,25 & 1 & & & & \\ 0,04 & -0,55 & -0,25 & -0,17 & -0,10 & 1 & & & \\ 0.29 & 0,10 & 0,76 & 0,55 & 0,33 & -0,26 & 1 & & \\ 0,16 & -0,08 & -0,43 & -0,29 & -0,25 & 0,63 & -0,36 & 1 & \\ 0,03 & 0,63 & 0,28 & 0,19 & 0,09 & \mathbf{-0,90} & 0,29 & -0,53 & 1 \end{array} \right) \end{array}$$

Le pouvoir explicatif du nombre de passes de codages de la contribution (*i.e.* , X_6) sur la dégradation de qualité (*i.e.* , Y) est déjà remarquable dans la matrice de corrélation. La corrélation entre Y et X_6 est indiquée en gras dans la matrice M. La valeur -0,90 est en effet le terme le plus élevé en valeur absolue de M et s'il est négatif c'est parce que la dégradation est exprimée de manière négative, par conséquent Y est d'autant plus négative que le nombre de passes de codage est grand.

Lorsque nous nous intéressons aux valeurs propres de la matrice des corrélations, nous pouvons conclure que les deux premières composantes principales expliquent 59% de l'inertie (voir tableau 3.4. Le calcul de l'inertie liée à la i-ème composante principale, elle même associée à la i-ème plus grande valeur propre, est effectué avec la formule $\frac{\lambda_i}{\sum_{j=1}^{p}\lambda_j}$, p étant le nombre de composantes principales ou encore le nombre de variables.).

TAB. 3.4: Valeurs propres et inertie expliquée par les composantes principales.

Axe	*Valeur propre*	*Inertie*	*Inertie cumulée*
1	3,46	38,4%	38,4%
2	1,86	20,6%	59,0%
3	1,36	15,1%	74,1%
4	0,93	10,4%	84,6%
5	0,64	7,1%	91,7%
6	0,32	3,6%	95,3%
7	0,22	2,5%	97,8%
8	0,13	1,4%	99,2%
9	0,08	0,8%	100%

Les pourcentages d'inertie liée à chaque axe sont représentés sur la figure 3.16. Comme leur décroissance n'est pas régulière puisqu'un palier est visible au passage de l'axe 1 à l'axe 2, nous pouvons déduire que les données sont structurées, c'est-à-dire qu'elles sont caractérisées par l'existence de corrélations et l'analyse factorielle peut fournir des résultats intéressants. Pour des raisons de lisibilité et de facilité de lecture, les représentations sont effectuées dans le plan des deux premiers axes.

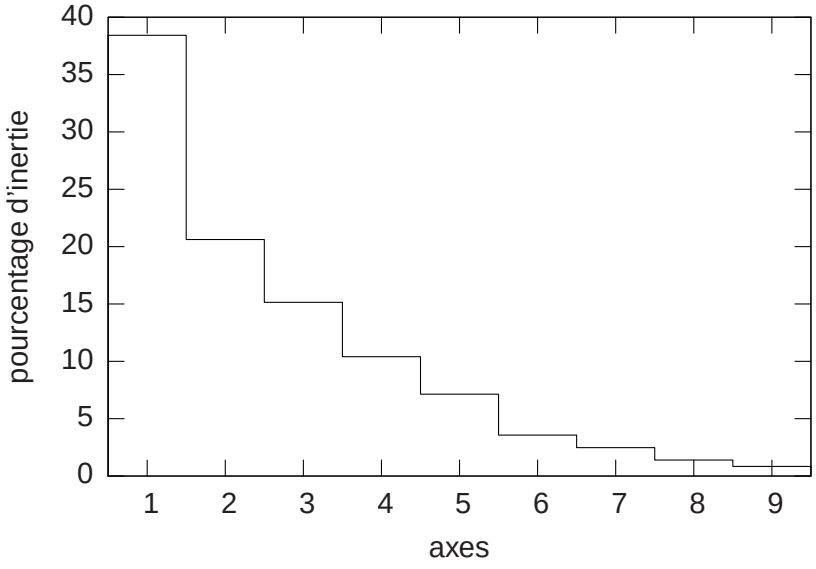


FIG. 3.16: Éboulis des valeurs propres.

Le cercle des corrélations fait apparaître des corrélations, notamment concernant la variable à expliquer Y (Δ_{PSNR}) qui nous intéresse ici (voir Fig. 3.17(a)). Y apparaît positivement

corrélée avec X_5 (couche de la première inclusion) mais aussi et surtout avec X_2 (couche de qualité). Cela s'explique car plus la couche de première inclusion est élevée, moins le *code-block* concerné est important du point de vue de l'apport en qualité. De même, plus la couche de la contribution est élevée, moins le raffinement de qualité est sensible. Dans les deux cas une valeur élevée pour X_2 ou X_5 tend à induire un Y fort (*i.e.* , la variation de PSNR sera moins négative). La corrélation entre le nombre de passes de codage et la variation de PSNR apparaît également, négative cette fois. En effet, un X_6 important entraîne une variation de PSNR plus importante (*i.e.* , Y de plus en plus négatif) comme cela a été vu dès la section 3.4.1.1. Notons que la corrélation positive entre X_6 et X_8 est due au codeur JPEG2000, un nombre de passes de codage plus important a de fortes chances de générer une contribution plus volumineuse. Cela n'apparaît pas sur la courbe 3.14(b) car le codeur JJ2000 a tendance à créer des contributions volumineuses dans la dernière couche de qualité sans forcément recourir à un nombre de passes de codage élevé. Or cette couche n'est par définition pas riche en incrément de qualité.

La projection des 852 points individus sur le même plan des axes 1 et 2 révèle une forte concentration au niveau de l'axe 1 et à gauche. Comme Y est projeté sensiblement parallèlement à cet axe, cela corrobore les corrélations positives entre les variables Y, X_2 et X_5. Ces points sont probablement caractérisés par des valeurs faibles pour ces variables et fortes pour X_6, l'axe 1 est donc un facteur taille. Un autre petit groupe d'individus se situe à droite sur cet axe, il correspond aux points qui sont caractérisés par des fortes valeurs pour Y, X_2 et X_5 et des X_6 faibles.

Les deux premiers axes de l'analyse en composantes fournissent près de 57% de l'inertie. Bien que d'interprétation délicate, nous pouvons espérer qu'un modèle linéaire avec peu de variables puissent expliquer Y. La section suivante traite donc naturellement de l'ajustement d'un modèle linéaire.

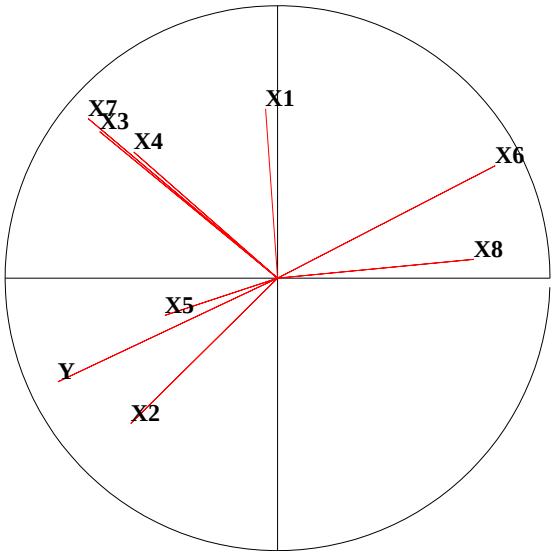
3.4.1.3 Explication de la dégradation par un modèle linéaire

Cette section vise à définir un modèle linéaire de la forme $y = X\alpha + \epsilon$ d'après notre jeu de données. Ce jeu de données est constitué de $n = 852$ échantillons eux-même composés de la lecture de Y et des $p = 8$ variables explicatives X_1 à X_8. On peut donc en extraire le vecteur y des observations de la variable à expliquer et la matrice (n, p) des observations des variables explicatives X_i. La finalité du modèle est bien entendu l'estimation de Y connaissant les variables X_i lues dans les en-têtes de paquets JPEG2000 de façon à pouvoir déterminer quelles sont les informations les plus critiques du point de vue de la dégradation de qualité potentielle.

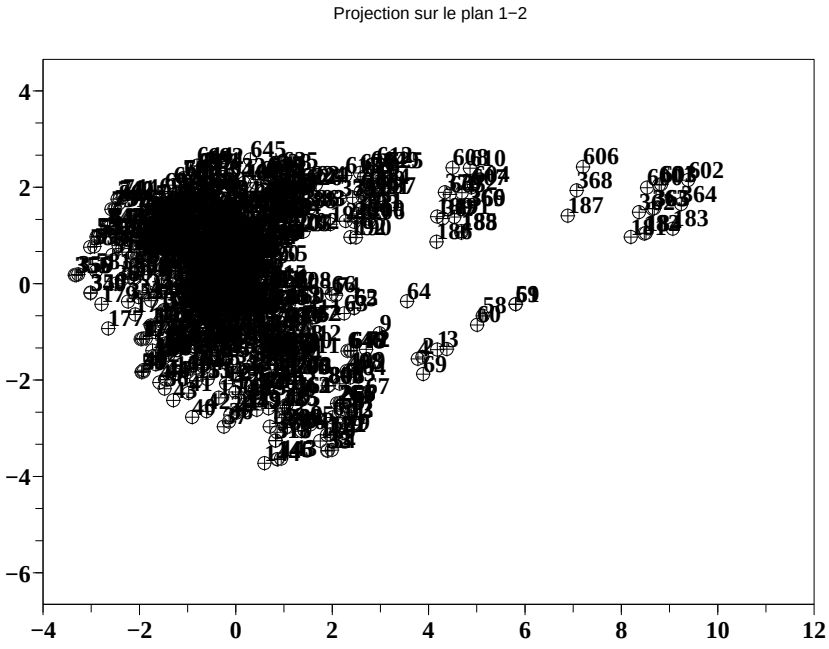
Nous cherchons un ajustement au sens des moindres carrés du modèle linéaire, c'est-à-dire, le vecteur de coefficients a solution de l'équation matricielle (3.3) qui minimise la somme des carrés des résidus $e^T e$. La matrice X contient les p colonnes relatives aux variables X_i précédées par une colonne de 1 qui correspond à une constante $X_0 = 1$. Le vecteur a est formé des $p + 1$ coefficients relatifs au terme constant et aux variables explicatives, *i.e.* , (a_0) détermine le terme constant (offset) et a_1 à a_8 sont les coefficients qui correspondent aux variables explicatives (X_1 à X_8 en l'occurrence).

$$\underset{(n,1)}{y} = \underset{(n,p+1)}{X} \underset{(p+1,1)}{a} + \underset{(n,1)}{e} \tag{3.3}$$

Le vecteur a qui correspond à l'ajustement minimisant l'erreur quadratique moyenne (EQM)



(a) Cercle pour le jeu de données complet.



(b) Projection des individus.

FIG. 3.17: Cercles des corrélations et projection des individus dans le plan factoriel défini par les axes 1 et 2.

est défini par :

$$a = (X^T X)^{-1} X^T y \tag{3.4}$$

Le coefficient de corrélation multiple R donne idée de la qualité de l'ajustement. Il s'agit du coefficient de corrélation entre les valeurs y initiales (observées dans le jeu de données) et les valeurs $\tilde{y}$ ajustées comme le montre l'équation (3.5). En pratique, c'est son carré qui est évalué selon (3.6).

$$R = cor(y, \tilde{y}) = cor(y, Xa) \tag{3.5}$$

$$R^2 = \frac{cov^2(y,\tilde{y})}{var(y)var(\tilde{y})} = \frac{var(\tilde{y})}{var(y)} = \frac{\sum(\tilde{y}_i)^2}{\sum(y_i)^2} = \frac{\text{variance expliquée}}{\text{variance totale}}$$

$$R^2 = \frac{a^T X^T X a}{y^T y} = \frac{y^T X (X^T X)^{-1} X^T y}{y^T y} \tag{3.6}$$

La meilleure régression prend obligatoirement en compte toutes les variables explicatives dont nous disposons, le vecteur des coefficients associé est le suivant :

$$a = \left[\begin{array}{cccccccc} -2,270 & 0,433 & 0,943 & 0,369 & 0,095 & -0,469 & -1,646 & 0,123 & 0,001 \end{array} \right]^T$$

Le modèle linéaire complet qui correspond le mieux à notre jeu de données au sens des moindres carrés est représenté par l'équation 3.7. Il est caractérisé par $R^2 = 0,897$, soit une variance de l'erreur $var(e) = 4,04$.

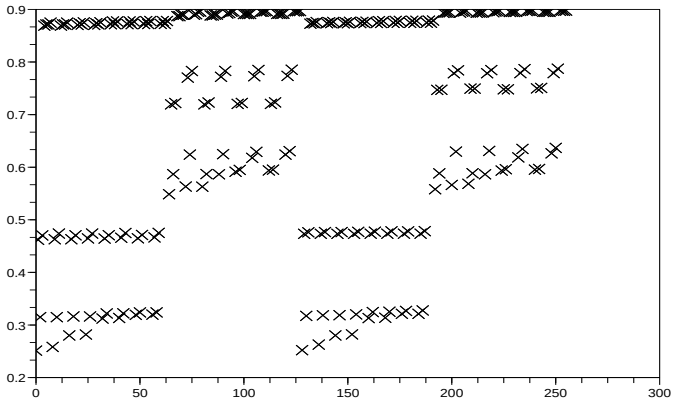
$$\begin{aligned} Y = {} & -2,270 + 0,433 \times X_1 + 0,943 \times X_2 + 0,369 \times X_3 + 0,095 \times X_4 - 0,469 \times X_5 \\ & -1,646 \times X_6 + 0,123 \times X_7 + 0,001 \times X_8 + e \end{aligned} \tag{3.7}$$

Il est possible de proposer un ajustement qui tient compte d'un nombre réduit de variables au prix d'un accroissement de l'erreur, à condition que ce dernier soit faible. La courbe 3.18(a) montre l'évolution de R^2 pour les 256 régressions possibles en tenant compte des combinaisons de une à huit variables parmi les huit dont nous disposons. Cette courbe révèle un nombre relativement élevé de régressions telles que $R^2 > 0,85$, il y en a effectivement 128. En pratique ces régressions sont les 128 meilleures et font toutes intervenir le nombre de passes de codage X_6. L'évolution de R^2 en fonction du nombre de variables pris en compte est mise en évidence sur la figure 3.18(b). R^2 excède 0,89 lorsque l'on tient compte de trois variables explicatives. Le meilleur des 56 ajustements à trois variables se rapproche fortement du modèle complet du point de vue de la variance des résidus (4,22 pour le meilleur ajustement à trois variables contre 4,04 avec les huit variables). Un nombre de variables supérieur à trois n'apporte ensuite qu'une augmentation très faible de la variance expliquée (*i.e.* , une réduction faible de la variance de l'erreur). Les performances des ajustements à trois variables sont représentées sur la courbe 3.18(c).

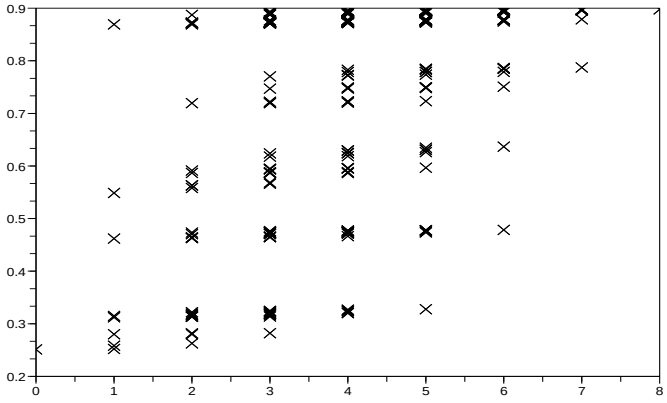
La meilleure régression à trois variables au sens des moindres carrés fait intervenir X_1, $X2$ et X_6. L'ajustement associé est défini par l'équation 3.8. La variance de l'erreur est alors de 4,22dB^2, c'est-à-dire que l'erreur moyenne de la prévision de la dégradation d'une contribution JPEG2000 est de $\pm$2,05dB.

$$Y = -1,115 + 0,844 X_1 + 0,751 \times X_2 - 1,797 \times X_6 \tag{3.8}$$

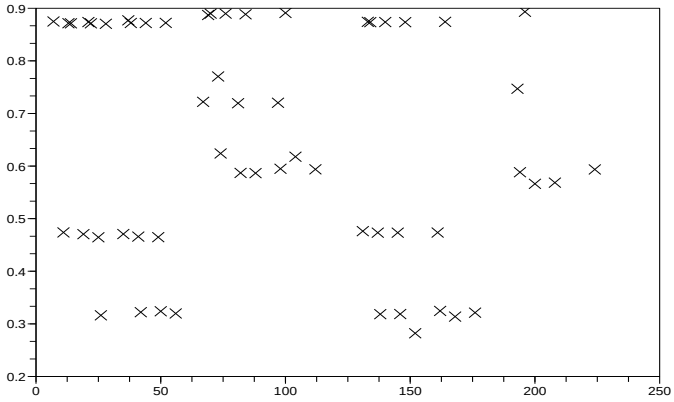
Le nombre de passes de codage X_6, intervient dans cet ajustement, ce qui confirme l'importance de cette information quant à la dégradation de qualité provoquée par la suppression



(a) R^2 pour chacune des 256 régressions tenant compte de toutes les variables explicatives ou non.



(b) R^2 en fonction du nombre de variables explicatives prises en compte.



(c) R^2 pour toutes les régressions qui tiennent compte de trois variables explicatives.

FIG. 3.18: Réduction du nombre de variables explicatives du modèle linéaire.

d'une contribution. Le numéro de la couche (X_2) apparaît également : plus il est faible et plus la dégradation sera importante (plus négative), ce qui correspond aux conclusions de l'analyse en composantes principales puisque Y et X_2 sont proches sur le cercle des corrélations (voir Fig.3.17(a)). La projection de la variable X_1 (le débit) est presque orthogonale à celles de Y, X_2 et X_6. Son intégration permet donc d'améliorer légèrement l'ajustement par compensation de l'écart d'alignement engendré par X_6 qui n'est pas parfaitement colinéaire avec Y (voir 3.17(a)). Le coefficient de corrélation multiple dans ce cas est tel que $R^2 = 0,887$ et la variance de l'erreur est $var(e) = 4,22$, c'est-à-dire très proches des résultats obtenus avec le modèle complet qui prend en compte la totalité des variables. Cette analyse, publiée dans (Holl *et al.*, 2006), permet donc de se faire une idée assez précise sur la dégradation de qualité objective (en termes de PSNR) que la perte d'une information élémentaire provoque sur la reconstruction.

À ce stade, l'ajustement ne tient compte que de l'image *Lena* 512×512 pixels à 256 niveaux de gris. Nous avons montré qu'un tel ajustement permet d'avoir une connaissance *a priori* de l'importance des contributions uniquement d'après la lecture des en-têtes des paquets JPEG2000. Nous avons également montré que pour une image donnée, il était possible d'établir un ajustement valable pour divers taux de compression. Afin d'étendre sa validité sur d'autres images, il faudrait prendre en compte un volume de données beaucoup plus important qui inclurait une grande variétés d'images à des tailles, des taux de compression, des paramètres de compression (nombre de couches, de tuiles, de composantes, codeur employé ...) divers. Nous savons en effet que la situation des informations les plus importantes après compression de l'image dépend naturellement de l'image source. La recherche d'un ajustement générique visant à prédire quelles sont les données dont la perte est la plus tolérable n'est peut-être pas le moyen le plus efficace pour y parvenir.

Toutefois s'il est suffisamment précis pour ne pas confondre une contribution essentielle avec une prédiction trop optimiste sur la dégradation engendrée par sa perte, nous pouvons proposer une classification des contributions selon leur niveau d'importance afin d'ajuster la fiabilité de leur transport. En l'état, nous observons les performances de l'ajustement quand il est utilisé comme critère pour la classifier des données de l'image *Lena* selon leur importance.

3.4.2 Classification des contributions et des paquets JPEG2000

Plusieurs voies sont envisageables pour classer les données du *code-stream* JPEG2000. Le caractère requis ou non peut être défini en fonction d'un seuil de qualité au delà duquel la dégradation de l'image n'est plus acceptable pour l'utilisateur. Ce caractère requis ou non peut être défini au niveau de la contribution élémentaire (*i.e.* , sur les contributions de *code-block*) ou bien du paquet JPEG2000.

En adoptant une approche par contribution, les plus importantes nécessitent alors un transport fiable tandis que l'on s'accommode de la perte éventuelle des contributions les moins significatives. Si l'on s'appuie sur un ajustement pour décider du caractère requis ou non de l'information en fonction du seuil de dégradation permis, le danger réside dans la possible mauvaise classification par l'ajustement. Finalement, certaines contributions peuvent être supposées significatives en regard du seuil fixé alors qu'en réalité la dégradation que leur perte entraîne est en deçà du seuil et inversement, d'autres contributions peuvent être considérées à tort non significatives par l'ajustement. Cette dernière éventualité est la plus gênante puisqu'elle autorise une dégradation excédant le niveau autorisé, *i.e.* la non sélection de contributions JPEG2000 qui provoquent pourtant une dégradation supérieure au seuil qui constitue un paramètre d'entrée de notre ajustement défini par l'équation 3.8. Le tableau 3.5 présente les

quantités des erreurs potentiellement graves qui sont liées à la sous-évaluation par l'ajustement défini par l'équation 3.8 de la dégradation associée à la perte de contributions. Ces erreurs sont exprimées en termes de contributions et de quantité de données exprimées en octets. La proportion des données jugées importantes par l'ajustement est également indiquée. Ce tableau montre que le taux de sous-évaluation des contributions varie de 0 à 15,7% selon le seuil et le débit de la compression, cela correspond à une erreur portant sur 0 à 18,7% du nombre total d'octets du *code-stream* (hors en-têtes de paquets).

Comme le *code-stream* est composé de paquets JPEG2000 dont chacun est décrit par un en-tête, il est judicieux de conserver le même niveau de granularité pour la décision du niveau d'importance et de fiabilité. Travailler au niveau du paquet permet en effet de ne pas multiplier les descriptions qui deviendraient nécessaires dans le cas d'une spécification de la fiabilité au niveau des contributions élémentaires (en particulier pour détecter leurs pertes). L'importance des paquets en fonction du seuil de dégradation autorisée peut être définie soit d'après la somme des dégradations élémentaires que le paquet contient, soit sur la base de la plus forte dégradation provoquée par la perte d'une des contributions de *code-block* qu'il contient. Cette dernière approche est la plus logique puisqu'elle classera comme important un paquet contenant une contribution élémentaire. Or comme chaque *code-block* présent dans le paquet concerne une zone distincte de l'image, une forte erreur sur un *code-block* serait alors plus remarquable visuellement que des erreurs faibles sur de multiples *code-blocks*. Cette approche implique qu'un paquet JPEG2000 contenant au moins une contribution classée importante devient lui-même requis dans sa globalité, ce qui a pour effet d'augmenter le rapport de la quantité de données à fiabiliser sur la quantité totale (voir tableau 3.6). Ce tableau rassemble pour chaque seuil de classification et débit de compression, le nombre de paquets JPEG2000 requis ainsi que le nombre total d'octets que cela représente en incluant les en-têtes de paquets JPEG2000 et tous les autres marqueurs éventuellement situés en dehors du *code-stream*. Notons que certains paquets JPEG2000 peuvent être vides, c'est en particulier le cas avec les fortes compressions et cela explique qu'il est possible que la totalité des données soit requise mais pas la totalité des paquets (cf. colonne 0,25bpp).

La section suivante présente une simulation des pertes des paquets jugés non fiables par l'ajustement. Cette analyse vise à déterminer si l'ajustement reste intéressant quant à la décision des paquets à fiabiliser malgré la sous-évaluation de la dégradation qu'entraîne la perte de certaines contributions.

3.4.3 Performances du système de classification

Nous avons déterminé les dégradations provoquées par l'absence des données des paquets qui contiennent uniquement des contributions jugées facultatives en regard du seuil de dégradation admissible pour chacune d'elles considérées indépendamment. Ces expérimentations permettent d'observer les effets du cumul des pertes de plusieurs contributions jugées peu importantes en regard du seuil de dégradation admissible. Le cumul de ces pertes produit évidemment une dégradation supérieure à celle engendrée par l'absence d'une contribution unique mais nous montrons que selon le niveau d'exigence de l'application, il est possible d'obtenir un niveau de qualité acceptable en agissant sur le seuil du classificateur.

Les courbes de la figure 3.19 montrent les variations du PSNR en fonction du seuil pris en compte par le classificateur. Il s'agit du pire cas, c'est-à-dire lorsque tous les paquets non fiabilisés sont manquants. En regard du PSNR, l'indication du niveau de fiabilité met en relation la dégradation de qualité et la quantité de données perdues. Ce niveau de fiabilité est exprimé relativement au nombre d'octets contenus dans des paquets jugés importants par le classifica-

TAB. 3.5: Contributions sous-évaluées par l'ajustement et quantité de données cruciales en fonction du seuil de dégradation autorisée.

Débit / Seuil	0,25bpp	0,5bpp	1bpp	1,5bpp	2bpp
-0,5dB (cont. SE)	9 (15,7%)	5 (4,0%)	6 (3,3%)	10 (4,2%)	23 (9,0%)
(octets SE)	451 (5,6%)	772 (4,8%)	293 (0,9%)	600 (1,2%)	4017 (6,2%)
(p-octets/total)	91,4%	75,8%	75,5%	57,4%	67,1%
-1dB (cont. SE)	7 (12,2%)	5 (4,0%)	0	8 (3,3%)	7 (2,7%)
(octets SE)	259 (3,2%)	629 (3,9%)	0	333 (0,6%)	297 (0,4%)
(p-octets/total)	91,4%	72,0%	69,1%	56,8%	60,6%
-2dB (cont. SE)	5 (8,7%)	2 (1,6%)	1 (0,5%)	2 (0,8%)	5 (1,9%)
(octets SE)	420 (5,3%)	405 (2,5%)	4 (0,0%)	28 (0,0%)	115 (0,1%)
(p-octets/total)	79,6%	67,6%	65,8%	52,6%	56,4%
-3dB (cont. SE)	2 (3,5%)	5 (4,0%)	0	2 (0,8%)	3 (1,1%)
(octets SE)	289 (3,6%)	103 (0,6%)	0	7 (0,0%)	31 (0,0%)
(p-octets/total)	76,9%	51,7%	52,2%	39,9%	30,9%
-4dB (cont. SE)	4 (7,0%)	1 (0,8%)	0	1 (0,4%)	3 (1,1%)
(octets SE)	668 (8,4%)	15 (0,1%)	0	3 (0,0%)	17 (0,0%)
(p-octets/total)	46,4%	50,4%	51,0%	38,8%	30,9%
-5dB (cont. SE)	8 (14,0%)	0	0	0	8 (3,1%)
(octets SE)	630 (7,9%)	0	0	0	176 (0,2%)
(p-octets/total)	23,6%	40,5%	50,4%	35,1%	25,9%
-6dB (cont. SE)	7 (12,2%)	6 (4,8%)	3 (1,6%)	3 (1,2%)	4 (1,5%)
(octets SE)	526 (6,6%)	182 (1,1%)	11 (0,0%)	11 (0,0%)	131 (0,2%)
(p-octets/total)	22,6%	35,0%	49,5%	33,8%	24,6%
-7dB (cont. SE)	7 (12,2%)	8 (6,5%)	0	3 (1,2%)	17 (6,7%)
(octets SE)	842 (10,6%)	219 (1,3%)	0	11 (0,0%)	687 (1,0%)
(p-octets/total)	12,9%	32,0%	49,5%	32,8%	23,8%
-8dB (cont. SE)	6 (10,5%)	7 (5,6%)	0	9 (3,7%)	15 (5,9%)
(octets SE)	655 (8,2%)	296 (1,8%)	0	250 (0,5%)	526 (0,8%)
(p-octets/total)	12,9%	26,6%	47,8%	31,7%	23,8%
-9dB (cont. SE)	7 (12,2%)	4 (3,2%)	0	3 (1,2%)	19 (7,5%)
(octets SE)	1170 (14,7%)	145 (0,9%)	0	338 (0,6%)	1953 (3,0%)
(p-octets/total)	6,0%	21,1%	47,8%	28,5%	21,4%
-10dB (cont. SE)	5 (8,7%)	5 (4,0%)	0	0	11 (4,3%)
(octets SE)	1484 (18,7%)	815 (5,1%)	0	0	1126 (1,7%)
(p-octets/total)	0,8%	16,9%	43,0%	28,5%	21,4%

TAB. 3.6: Quantités de données contenues dans les paquets classés comme requis.

Seuil \ Débit	0,25bpp	0,5bpp	1bpp	1,5bpp	2bpp
-0,5dB (JP2pkt requis/15)	11	13	10	11	11
(octets requis en-têtes inclus)	8268	15171	29973	45436	52498
(p-octets/total)	100%	92,5%	91,5%	92,7%	80,6%
-1dB (JP2pkt requis/15)	11	13	9	11	11
(octets requis en-têtes inclus)	8268	15171	25058	45436	52498
(p-octets/total)	100%	92,5%	76,4%	92,7%	80,6%
-2dB (JP2pkt requis/15)	11	12	8	10	8
(octets requis en-têtes inclus)	8268	14987	24741	42329	41490
(p-octets/total)	100%	91,4%	75,5%	86,3%	63,7%
-3dB (JP2pkt requis/15)	11	10	6	8	6
(octets requis en-têtes inclus)	8268	12306	21047	34077	35340
(p-octets/total)	100%	75,0%	64,2%	69,5%	54,2%
-4dB (JP2pkt requis/15)	8	10	5	6	6
(octets requis en-têtes inclus)	4919	12306	20342	31830	35340
(p-octets/total)	59,4%	75,0%	62,1%	64,9%	54,2%
-5dB (JP2pkt requis/15)	5	8	4	4	4
(octets requis en-têtes inclus)	3471	9922	19449	19705	20353
(p-octets/total)	41,9%	60,5%	59,4%	40,2%	31,2%
-6dB (JP2pkt requis/15)	5	6	3	4	3
(octets requis en-têtes inclus)	3471	9295	16767	19705	16778
(p-octets/total)	41,9%	56,6%	51,2%	40,2%	25,7%
-7dB (JP2pkt requis/15)	4	5	3	3	3
(octets requis en-têtes inclus)	3184	7620	16767	16739	16778
(p-octets/total)	38,5%	46,4%	51,2%	34,1%	25,7%
-8dB (JP2pkt requis/15)	4	5	3	3	3
(octets requis en-têtes inclus)	3184	7620	16767	16739	16778
(p-octets/total)	38,5%	46,4%	51,2%	34,1%	25,7%
-9dB (JP2pkt requis/15)	3	4	3	3	3
(octets requis en-têtes inclus)	2607	5908	16767	16739	16778
(p-octets/total)	31,5%	36,0%	51,2%	34,1%	25,7%
-10dB (JP2pkt requis/15)	1	4	3	3	3
(octets requis en-têtes inclus)	439	5908	16767	16739	16778
(p-octets/total)	5,3%	36,0%	51,2%	34,1%	25,7%

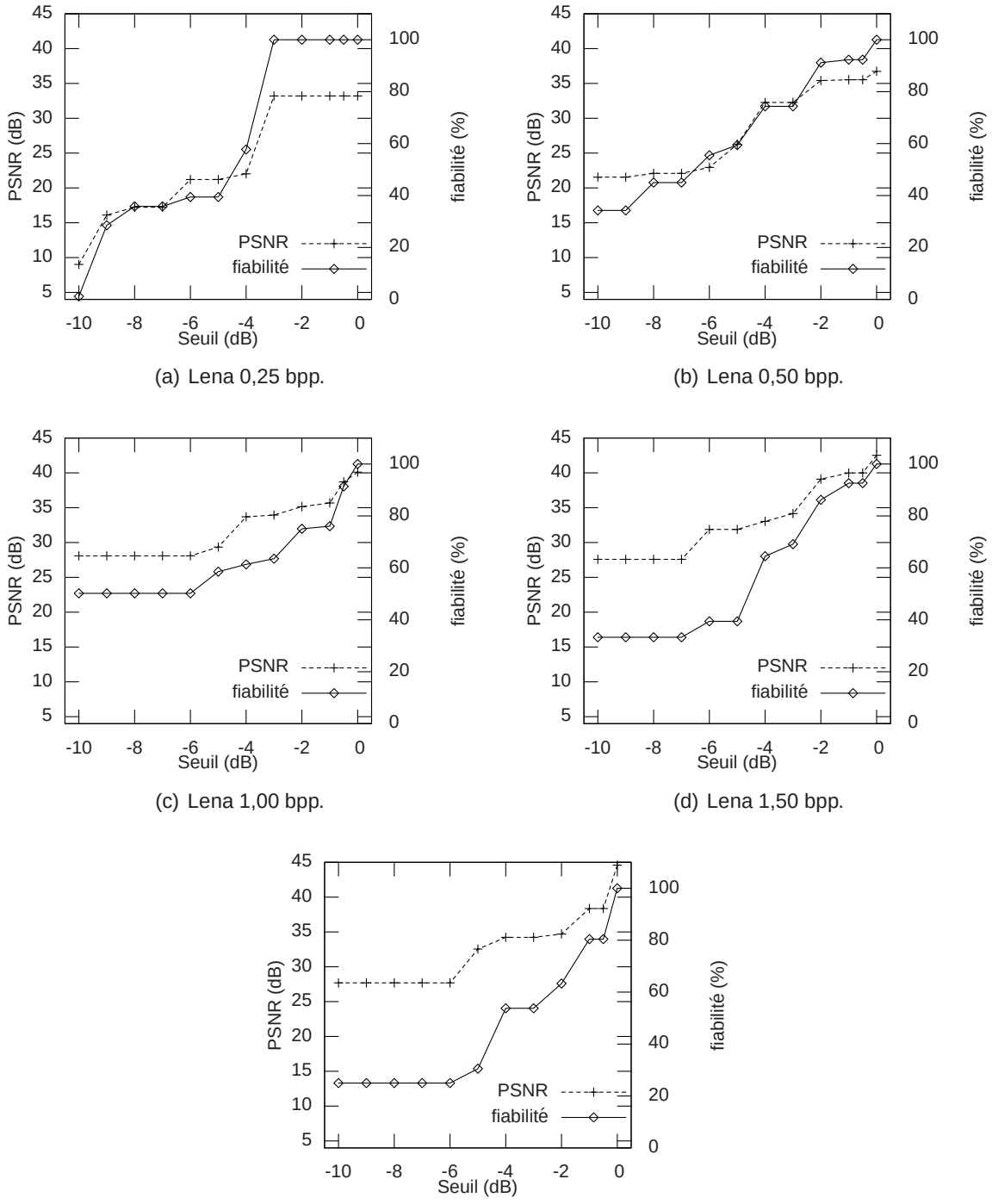
teur en fonction du seuil de dégradation admissible fixé, et du nombre total d'octets inclus dans tous les paquets JPEG2000.

La dégradation exprimée par la chute du PSNR ne rend pas compte des types de défauts visibles sur les images reconstruites, d'où l'intérêt des images présentées sur les figures 3.20 à 3.24 selon le débit de la compression pour l'image *Lena* 512×512 pixels à 256 niveaux de gris. Ces images montrent que les effets de la perte des paquets choisis par le classificateur ne sont pas du même ordre que ceux produits par une compression plus forte par exemple. La dégradation obtenue se présente sous la forme d'un bruit qui peut être localisé (si les paquets manquants ne contiennent pas les *code-blocks* permettant de couvrir toute la surface de l'image).

Ces expérimentations montrent que la plus forte compression (0,25 bit/pixel) est celle qui tolère le moins les pertes d'informations (voir Fig. 3.19(a) et 3.20). Cette remarque s'explique simplement du fait qu'une compression génère par définition des données d'autant plus porteuses d'informations cruciales qu'elle est forte. Pour les autres taux de compression de l'image *Lena*, nous observons qu'il est possible de supprimer une quantité non négligeable d'information tout en maîtrisant la dégradation de PSNR. Par exemple pour la compression à 0,5 bit/pixel, un seuil fixé à -3dB permet au classificateur de sélectionner 26% des données comme « facultatives ». Le PSNR résultant de la perte de ces données est alors de 32,2dB et les dégradations perceptibles visuellement sont peu sensibles comme en témoigne l'image 3.21(c). Un taux de compression plus faible augmente évidemment la latitude de perte, que ce soit en volume absolu ou relatif. Les figures 3.19(d) et 3.19(e) vont dans ce sens. La qualité de l'image issue de la compression sans suppression additionnelle est naturellement plus élevée (PSNR de 42,5dB et 44,5dB respectivement à 1,5 et 2 bits/pixels). Le volume de données est également proportionnel au taux de compression et le moindre incrément de qualité apporté par ces données expliquent ces résultats. Les figures 3.23 et 3.24 prouvent que la dégradation est visuellement peu sensible, que ce soit à l'impression (l'échantillonnage utilisé ne camoufle pourtant pas la dégradation : 73pixels/cm) ou à l'écran.

Bien sûr nous pourrions argumenter qu'une compression plus forte est plus efficace en termes de qualité qu'une compression plus faible qui subit des pertes additionnelles *a posteriori*, mais cela est vrai lorsque l'on considère la compression comme un moyen de stocker, archiver et reconstruire ensuite. Lorsque l'on adopte une approche transmission via un réseau comme c'est le cas ici, il faut garder à l'esprit que les pertes de paquets existent et que leur correction est coûteuse en temps et en bande passante. Notre proposition peut s'apparenter à une seconde compression appliquée par le réseau. Cette compression est alors aléatoire mais avec une latitude de perte restreinte par le marquage des informations sur lesquelles elle est tolérée.

Ces résultats sont donc intéressants et ce sans compter que la probabilité que toutes les données non fiabilisées soient perdues sur un réseau reste faible, donc la qualité de l'image reconstruite ne peut qu'être supérieure aux cas exposés ici. C'est là une différence essentielle entre la transmission fiable d'une image compressée sur un réseau à commutation de paquets avec pertes et la transmission partiellement fiable de la même image mais compressée moins fortement : il est possible d'obtenir une qualité satisfaisante dans les deux cas lorsque le réseau perd des paquets. Le coût en termes de temps et de consommation des ressources réseau peut être comparable, par contre dès que les conditions réseaux sont bonnes (*i.e.* , pas de pertes) la transmission partiellement fiable de l'image plus faiblement compressée autorisera une qualité supérieure au prix du surcoût de temps associé. Cependant si le réseau ne perd pas de paquets c'est qu'il n'y a pas de congestions, et par définition les débits peuvent alors augmenter et la durée de transport s'en voit réduite.



(a) Lena 0,25 bpp.

(b) Lena 0,50 bpp.

(c) Lena 1,00 bpp.

(d) Lena 1,50 bpp.

(e) Lena 2,00 bpp.

FIG. 3.19: Pire cas pour la reconstruction de l'image après perte de tous les paquets JPEG2000 non fiabilisés par le classificateur.

(a) Seuil = 0dB, fiabilité = 100%, PSNR = 33,2dB.

(b) Seuil = -4dB, fiabilité = 57,7%, PSNR = 22,0dB.

(c) Seui = -9dB, fiabilité = 28,5%, PSNR = 16,1dB.

FIG. 3.20: Pire des cas pour la reconstruction de *Lena* 512×512 à 0,25 bpp après que tous les paquets décidés facultatifs en fonction du seuil par le classificateur ont été supprimés (substituées par 0).

(a) Seuil = 0dB, fiabilité = 100%, PSNR = 36,7dB.

(b) Seuil = -1dB, fiabilité = 92,3%, PSNR = 35,5dB.

(c) Seuil = -3dB, fiabilité = 74,3%, PSNR = 32,2dB.

(d) Seuil = -9dB, fiabilité = 34,2%, PSNR = 21,5dB.

FIG. 3.21: Pire des cas pour la reconstruction de *Lena* 512×512 à 0,5 bpp après que tous les paquets décidés facultatifs en fonction du seuil par le classificateur ont été supprimés (substituées par 0).

(a) Seuil = 0dB, fiabilité = 100%, PSNR = 40,1dB.

(b) Seuil = -1dB, fiabilité = 91,3%, PSNR = 35,6dB.

(c) Seuil = -3dB, fiabilité = 63,5%, PSNR = 33,9dB.

(d) Seuil = -9dB, fiabilité = 50,1%, PSNR = 28,0dB.

FIG. 3.22: Pire des cas pour la reconstruction de *Lena* 512×512 à 1 bpp après que tous les paquets décidés facultatifs en fonction du seuil par le classificateur ont été supprimés (substituées par 0).

(a) Seuil = 0dB, fiabilité = 100%, PSNR = 42,5dB.

(b) Seuil = -1dB, fiabilité = 92,6%, PSNR = 39,9dB.

(c) Seuil = -3dB, fiabilité = 69,1%, PSNR = 34,1dB.

(d) Seuil = -9dB, fiabilité = 33,2%, PSNR = 27,5dB.

FIG. 3.23: Pire des cas pour la reconstruction de *Lena* 512×512 à 1.5 bpp après que tous les paquets décidés facultatifs en fonction du seuil par le classificateur ont été supprimés (substituées par 0).

(a) Seuil = 0dB, fiabilité = 100%, PSNR = 44,5dB.

(b) Seuil = -1dB, fiabilité = 80,4%, PSNR = 38,3dB.

(c) Seuil = -3dB, fiabilité = 53,7%, PSNR = 34,2dB.

(d) Seuil = -9dB, fiabilité = 24,9%, PSNR = 27,6dB.

FIG. 3.24: Pire des cas pour la reconstruction de *Lena* 512×512 à 2 bpp après que tous les paquets décidés facultatifs en fonction du seuil par le classificateur ont été supprimés (substituées par 0).

3.5 Transport d'images codées selon le standard JPEG2000

3.5.1 Adaptation du transport à fiabilité partielle pour les flux JPEG2000

Puisque les sections précédentes ont montré que le *code-stream* JPEG2000 est caractérisé par une certaine tolérance aux pertes de paquets, l'utilisation d'un protocole de transport à fiabilité partielle est intéressante *a priori*. Notre proposition consiste naturellement à transmettre les paquets JPEG2000 en utilisant 2CP-ARQ avec une fiabilité décidée par notre classificateur en fonction d'une consigne qui est le seuil de dégradation autorisé.

Une solution simple consisterait à transporter un paquet JPEG2000 dans un paquet 2CP-ARQ. Un paquet JPEG2000 décidé requis par le classificateur serait encapsulé dans un p-paquet 2CP-ARQ et inversement, un paquet JPEG2000 facultatif en regard du seuil serait transporté dans un s-paquet. Un problème se pose alors du fait de la longueur extrêmement variable des paquets JPEG2000 et des contraintes de taille des paquets en couche transport. La longueur des paquets JPEG2000 varie en effet de quelques octets à plusieurs milliers, ils ne peuvent donc pas être embarqués dans un paquet 2CP-ARQ unique à tous les coups. Les exigences de tailles admissibles pour les unités de transport (MSS, lui-même lié au MTU) imposent en effet de ne pas dépasser une certaine valeur, typiquement 1460 octets pour TCP/IP/Ethernet. Inversement, les petits paquets JPEG2000 dégradent fortement l'efficacité du transport du fait de la surcharge relative des divers en-têtes.

Pour pallier le manque d'efficacité de l'encapsulation des petits paquets JPEG2000, une solution consiste à transporter plusieurs paquets JPEG2000 par paquet 2CP-ARQ dans la limite du MSS sous réserve que ces paquets JPEG2000 appartiennent à la même classe de fiabilité (p- ou s-). En pratique, l'alternance dans *le code-stream* des paquets JPEG2000 classés requis ou facultatifs en fonction du seuil fixé rend la tâche délicate. Cela reste néanmoins possible, mais nécessite de réordonner les paquets JPEG2000, ce qui implique l'ajout d'informations supplémentaires dans les paquets 2CP-ARQ pour réordonner différemment en couche application.

L'alternance des données facultatives et requises est encore accentuée par la présence des en-têtes de paquets JPEG2000 qui par défaut se situent juste avant le paquet concerné. Or, quelle que soit la nature du paquet JPEG2000, son en-tête est toujours requis pour réaliser le décodage d'une part et pour reconstruire les paquets manquants de bonne longueur d'autre part. Ces en-têtes doivent donc naturellement être transportées dans des p-paquets 2CP-ARQ bien que leur petite taille et leur présence avant chaque paquet du *pack-stream* ne favorise pas la tâche. Cette difficulté est cependant facilement contournable puisqu'il est possible de spécifier lors de la compression que tous les en-têtes de paquets JPEG2000 soient regroupés dans l'en-tête principal ou l'en-tête de tuile. Ce regroupement des en-têtes nécessaires limite la multiplication des segments requis et facultatifs au sein du *code-stream*, sans pour autant supprimer l'alternance des paquets JPEG2000 classés importants ou non.

Pour récapituler, les difficultés liées au transport à fiabilité partielle des images codées JPEG2000 sont les suivantes :

- la taille des paquets JPEG2000 est extrêmement variable, de zéro à plusieurs milliers d'octets. Les paquets de grande taille nécessitent par conséquent plusieurs paquets de transport. Des petits paquets doivent au contraire être rassemblés dans un même paquet 2CP-ARQ pour optimiser les performances.
- les paquets JPEG2000 de même caractéristique (requis ou non) ne sont pas forcément groupés au sein du *code-stream*. Il est de ce fait difficile de regrouper les petits paquets JPEG2000 dans un paquet 2CP-ARQ de même classe.

– les divers en-têtes JPEG2000 doivent être transportés dans des p-paquets. Dans l'éventualité d'un groupement de paquets JPEG2000 facultatifs, la seule présence de leurs en-têtes dans le *code-stream* empêche la possibilité de les rassembler dans un même s-paquet. Les en-têtes de paquets JPEG2000 étant très courtes, cela va conduire à de petits paquets 2CP-ARQ alors que c'est justement ce qu'il faut éviter.
– les en-têtes de paquets JPEG20000 peuvent être relocalisés à l'extérieur du *pack-stream* (en-tête principal ou en-tête de tuile). Bien que ce ne soit pas le choix par défaut des codeurs JPEG2000, cette option est plus adaptée aux caractéristiques du protocole 2CP-ARQ.

Prenons un exemple concret pour illustrer les difficultés liées au transport à fiabilité partielle des images codées JPEG2000 avec l'image *Lena* encodée à 1,5bpp, 512×512 pixels en niveaux de gris, trois résolutions et cinq couches de qualité avec la progression LRCP. La compression conduit à un *code-stream* de quinze paquets JPEG2000 qui peut être encapsulé dans un fichier JP2. Si l'on désire transporter cette image à l'aide de 2CP-ARQ, nous devons sélectionner les données qui peuvent être considérées facultatives. Nous allons faire appel au classificateur en lui donnant une valeur de seuil pour cela. Prenons par exemple un seuil à -3dB permettant d'obtenir une qualité intéressante dans le pire des cas (voir Fig. 3.23(c)). Les caractéristiques des paquets JPEG2000 correspondants sont rassemblées dans le tableau 3.7.

TAB. 3.7: Classification et transport des paquets JPEG2000 de l'image *Lena* 1,5 bit/pixel.

Paquet	longueur (octets)	contrib. classées requises	classification	classe 2CP-ARQ
0	7265	4	requis	p-paquet
1	5094	12	requis	p-paquet
2	3725	29	requis	p-paquet
3	973	0	optionnel	s-paquet
4	2348	4	requis	p-paquet
5	2966	16	requis	p-paquet
6	1117	1	requis	p-paquet
7	286	0	optionnel	s-paquet
8	1130	1	requis	p-paquet
9	20	0	optionnel	s-paquet
10	3107	0	optionnel	s-paquet
11	7279	0	optionnel	s-paquet
12	0	0	optionnel	s-paquet
13	3261	0	optionnel	s-paquet
14	9777	1	requis	p-paquet

Le classificateur sélectionne les paquets optionnels lorsqu'aucune des contributions incluses ne provoque une dégradation supérieure au seuil fixé (en valeur absolue). Dans notre exemple les paquets 3, 7, 9, 10, 11, 12 et 13 sont dans ce cas. En considérant le cas le plus simple où tous les en-têtes de paquets sont situés avant les paquets et un MSS de 1460 octets nous avons dans l'ordre :

– trois paquets JPEG2000 requis qui nécessiteront chacun plusieurs paquets de transport ou qui peuvent être regroupés dans onze p-paquets de 1460 octets de charge utile plus un p-paquet de 51 octets.
– un paquet optionnel transporté dans un s-paquet de 973 octets de charge utile.

– trois paquets JPEG2000 requis qui peuvent être combinés dans quatre p-paquets de 1460 octets de charge utile et un paquet contenant les 591 octets utiles restants.
– un s-paquet de 286 octets (rempli à 19%).
– un p-paquet de 1130 octets.
– cinq paquets JPEG2000 optionnels qui peuvent être transportés à l'aide de neuf s-paquets complets et un s-paquet de 527 octets.
– un paquet requis qui nécessitera sept p-paquets dont six complets plus un de 1017 octets de charge utile.

Au total il y aura donc 37 paquets 2CP-ARQ sans compter les divers en-têtes qui ajouteront encore un p-paquet de 555 octets, soit 38 paquets dont 27 sont fiabilisés (71%). On remarquera la création d'unités de transport très peu chargées dont la surcharge des divers en-têtes peut représenter jusqu'à 113% de la partie utile pour le paquet de 51 octets de charge utile si l'on considère un MSS de 1460 octets pour 2CP-ARQ sur IP et Ethernet. Sur les 38 paquets que nécessitera le transport de l'image, 7 ne sont pas remplis, ce qui représente 18% des paquets.

Les contenus JPEG2000 ne présentent donc pas les caractéristiques favorables pour optimiser leur transport à fiabilité partielle, notamment en termes de tailles et importance des blocs indépendants. Pour ce faire, il faudrait intégrer la finalité du transport lors de l'étape de compression ce qui peut aller à l'encontre des performances recherchées par la compression, à savoir la réduction maximale de l'espace nécessaire au stockage de l'information pour une qualité donnée.

Notons que l'approche que nous adoptons pour le transport de contenus JPEG2000 n'est pas celle qui est souvent abordée dans la littérature. Dans la section suivante nous montrons en effet que les motivations des travaux de recherche dans ce domaine sont essentiellement liées à rendre le transport interactif ou bien associés à un contexte réseau particulier, notamment les réseaux sans-fil.

3.5.2 Positionnement par rapport aux travaux extérieurs

Les différentes propositions de transmission associées à JPEG2000 présentées dans la littérature s'appuient soit sur un transport fiable (*e.g.* , TCP), soit sur la nécessité de corriger les pertes lorsque l'environnement considéré a un fort taux d'erreur (*e.g.* , environnement radio et sans-fil en général). Dans le premier cas, les solutions proposées visent à rendre interactive la transmission. Il s'agit alors d'afficher à distance l'image sur un client, le serveur devant fournir la quantité minimale d'informations à partir d'un unique fichier source pour satisfaire les requêtes des différents clients pour une image donnée. S'il s'agit de réseaux à forts taux d'erreurs ou de pertes qui sont considérés, alors les propositions s'appuient essentiellement sur le concept de FEC (Forward Error Correction) pour les corriger, avec notamment les codes à effacement (erasure codes), (Natu *et al.*, 2005), (Thie et Taubman, 2004), ou sur une protection inégale des données en accord avec leur importance (Unequal Error Protection, UEP), (Thomos *et al.*, 2004) et (Pu *et al.*, 2005).

La transmission d'images JPEG2000 est traitée dans la neuvième partie de la norme, (JTC1/SC29, 2004*b*), avec la définition d'un protocole pour un transfert interactif. Cette partie repose de nouveau sur les travaux de David Taubman, (Taubman, 2002) : JPEG2000 Interactive Protocol (JPIP). JPIP est interactif dans le sens où les données transmises par le serveur d'images vont dépendre des exigences du client. Ce dernier peut nécessiter uniquement une résolution réduite s'il s'agit d'un téléphone ou d'un assistant personnel (PDA), ou bien il peut demander un niveau de qualité inférieur à ce que permet la totalité des données du fichier JP2 source. Le serveur transmet alors la totalité ou une partie seulement des données, dans un

ordre qui peut différer de l'organisation du *code-stream* dans le fichier stocké. L'application de réception effectue les requêtes nécessaires à l'affichage en fonction de ses besoins. Il peut s'agir d'une zone bien définie (certains *precincts*), d'une résolution ou bien d'un niveau de qualité (un certain nombre de couches de qualité). Pour répondre finement aux attentes du client alors que le serveur dispose d'un unique fichier compressé par image, ce fichier doit disposer de certaines caractéristiques en termes de nombre de couches, de *precincts*. Cela implique l'ajout de directives particulières lors de la compression. Il ne s'agit donc pas de fichiers JP2 standard que les applications compatibles créent par défaut. Typiquement, JPIP demande que les *precincts* soient petits pour que le niveau de finesse de la réponse puisse satisfaire les exigences précises du client (et ne pas transmettre des données inutilement). Or par défaut les images sont constituées d'un unique *precinct* dont la taille maximum peut atteindre 2^{15} pixels par défaut. Ainsi aucune image courante ne nécessite plusieurs *precincts*.

Les recherches dans le domaine de la transmission de flux JPEG2000 ne s'appuient pas uniquement sur le partitionnement en *precincts*. Gormish et al. ont proposé dans (Gormish et Banerjee, 2003) de transmettre les images JPEG2000 sur la base d'un découpage en tuiles (*tiles*) plutôt qu'en *precincts* comme c'est le cas de JPIP. Le découpage en tuiles revient à fragmenter l'image en plusieurs images plus petites et indépendantes. D'apparence plus simple que le partitionnement en *precincts*, cette technique impose là encore que les fichiers images à servir soient compressés en vue de posséder les caractéristiques voulues. Notamment avec un nombre de tuiles important, or la grande majorité des images ne comportent qu'une seule tuile.

Le transport de flux JPEG2000 ne s'arrête pas aux images fixes, la société Sony a dans ce sens proposé des solutions pour transporter les séquences animées JPEG2000 (Futemma *et al.*, 2005). Ce draft IETF s'appuie sur RTP (contraintes temporelles de la vidéo), mais propose de transmettre d'abord les informations contenues dans les différents en-têtes du *code-stream* JPEG2000 avant de transmettre le contenu des paquets JPEG2000. Nous avons constaté en section 3.4 que les en-têtes étaient absolument requis pour que la reconstruction de l'image soit possible. La transmission en premier lieu de ces informations semble donc légitime puisque si elles ne sont pas disponibles dans les temps, le récepteur ne pourra pas décoder l'image. Après la transmission des en-têtes, cette proposition recommande un ordonnancement des contenus de paquets JPEG2000 selon leur degré d'importance pour la qualité de la reconstruction. Bien que recommandé dans le cas des séquences animées, notre proposition repose en partie sur ce principe puisque les différents en-têtes sont transportés de manière fiable. Nous allons cependant plus loin au niveau de l'assignation d'un niveau d'importance à chaque paquet JPEG2000. En effet, tandis que les travaux présentés dans (Futemma *et al.*, 2005) supposent que les paquets JPEG2000 sont ordonnés dans l'ordre décroissant de leur importance pour une image en progression LRCP, nous avons montré que cela n'était pas si évident. Cette constatation nous a conduit à la recherche d'un classificateur permettant de sélectionner les informations dont la perte est la plus tolérable du point de vue de la qualité de la reconstruction et nous avons montré que la dégradation de qualité pouvait être maîtrisée.

Conclusion

Dans ce chapitre, nous avons présenté les caractéristiques du code issu de la compression JPEG2000 d'une image fixe. Une analyse de la tolérance aux pertes du *code-stream* a été présentée, ce qui a permis d'en extraire un ajustement visant à évaluer la dégradation entraînée par la perte des données en ayant juste la connaissance des informations présentes dans

les en-têtes de paquets JPEG2000. Cet ajustement permet de classer les contributions et les paquets JPEG2000 selon leur niveau d'importance en fonction d'un seuil de dégradation admissible.

La classification de l'importance des données est le préalable à l'ajustement du niveau de fiabilité lorsque l'on considère un transport à fiabilité partielle. Nous nous heurtons cependant à des difficultés de mise en paquets liées aux spécificités du système de compression étudié, en l'occurrence JPEG2000 mais tout système de compression poserait ce type de difficulté lorsqu'il est conçu et optimisé pour offrir des fonctionnalités parmi lesquelles le décodage de l'information partielle ne figure pas. Les systèmes de compression couramment employés pour les images fixes sont avant tout conçus pour réduire le volume nécessaire au stockage sous forme de fichiers, le corollaire est que le transport de ces fichiers est plus rapide que celui des images brutes mais il suppose un transport fiable (TCP). Cette condition de fiabilité du transport est naturellement satisfaite pour l'ensemble du trafic web utilisant le protocole HTTP, mais si l'on veut autoriser des pertes alors des mécanismes de résistance aux erreurs et/ou de resynchronisation deviennent nécessaires. JPEG2000 dispose bien de tels mécanismes de resynchronisation, notamment avec l'utilisation optionnelle des marqueurs SOP (Start of Packet) et EPH (End of Packet Header). Néanmoins nous avons mis en évidence que tous les décodeurs JPEG2000 ne réagissent pas correctement face à l'absence de paquets JPEG2000, certains quittent inopinément lorsqu'ils détectent que le *code-stream* est invalide alors qu'il serait possible de resynchroniser le décodage au paquet suivant qui est marqué. Pour la vidéo le problème est différent puisque les possibilités naturelles de resynchronisation entre deux images ou deux groupes d'images rendent possible le transport non fiable.

Finalement les difficultés de mise en paquets et de conciliation du décodage avec le caractère partiellement fiable du transport auxquelles nous sommes confrontés justifient l'approche dite « Network Conscious » de Amer *et al.* que les auteurs ont appliquée à différents systèmes de compression comme GIF (Amer *et al.*, 1999) ou SPIHT (Iren et Amer, 2000) bien que leur objectif soit la progressivité de l'affichage avec l'emploi d'un transport partiellement ordonné et non pas partiellement fiable. Leur approche consiste à modifier le système de compression de manière à le rendre pleinement compatible avec l'ordre partiel et à offrir les caractéristiques propices à la progressivité de l'affichage.

Dans ce chapitre, nous avons cherché à conserver JPEG2000 en l'état. L'idéal aurait été une couche 2CP-ARQ générique utilisée directement quelle que soit la compression envisagée mais nous avons été contraints à élaborer une couche intermédiaire entre le système de compression et la couche transport de façon à assurer une compatibilité relative et à prévenir les impossibilités de décodage qui apparaissent lorsque JPEG2000 est directement couplé à 2CP-ARQ. Notons que cette couche intermédiaire doit être adaptée au système de compression envisagé, ce qui implique le développement d'une couche pour chaque système de compression. Cette approche n'est donc pas plus simple ou rapide à mettre en œuvre que celle qui consiste à modifier la compression pour offrir les caractéristiques nécessaires au transport considéré, de plus, une couche intermédiaire ne peut certainement pas garantir une compatibilité et surtout toutes les caractéristiques propices au transport qu'un système de compression expressément conçu dans ce but peut offrir.

Ces difficultés peuvent être adressées en adoptant une approche qui lie compression et transport. Il est évident qu'un transport à fiabilité partielle tel que nous le présentons doit avoir conscience de la nature des données transportées, notamment en ce qui concerne l'importance de la dégradation générée en cas de perte. Consécutivement aux travaux concernant JPEG2000 que nous avons présentés dans ce chapitre, il semble judicieux de penser au transport à fiabilité partielle dès l'étape de compression. Une telle approche implique des contraintes

supplémentaires qui rendent certainement moins performante la compression (en termes de PSNR pour un taux donné) mais ce serait oublier les avantages d'une compression orientée transport qui par définition n'est pas à vocation de stockage/archivage. Cela permettrait de créer un flux ayant des caractéristiques intéressantes en ce qui concerne la taille des unités de données, leur ordonnancement et leur tolérance aux pertes. Notons que cette approche est courante pour la compression des flux vidéo à vocation de diffusion sur les réseaux. Des protocoles comme H323 rassemblent aussi bien la partie compression que le protocole de transport (RTP), la compression doit alors produire des unités de données qui correspondent à un intervalle temporel dont la taille doit être maîtrisée.

Dans le chapitre suivant, nous allons donc adopter une approche pour la compression en gardant à l'esprit les impératifs liés au transport à fiabilité partielle reposant sur deux classes de paquets.

Chapitre 4

Décomposition multirésolution et compression d'images associée à 2CP-ARQ

Introduction

Les systèmes de compression d'images couramment utilisés comme JPEG et à plus forte raison JPEG2000 fournissent de bons rapports entre le taux de compression et la qualité des images mais ces codages n'ont pas été conçus pour être robustes aux pertes de données. Ils ne sont donc pas adaptés avec un système de transport non fiable des images sur un réseau à commutation de paquets. Dans le cadre du transport d'images à fiabilité partielle, rappelons les deux conditions pour que les systèmes de compression et de transport soient compatibles :

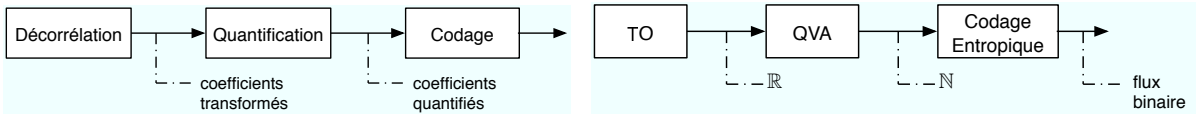
- le *code-stream* de l'image doit être divisible en unités de données décodables indépendamment,
- les unités de données doivent pouvoir être classées selon différents niveaux de priorité (au moins deux niveaux).

Évidemment, le respect de ces deux conditions risque de réduire la « performance » du système de compression, mais en contrepartie, cela permettra d'augmenter celle du système de transport. Dans ce chapitre, nous proposons un système de compression qui remplit ces conditions. Il est fondé sur la décomposition par transformée en ondelette discrète, un principe aujourd'hui communément admis en compression d'images, puis sur la quantification vectorielle algébrique. Au-delà de ces techniques, nous proposons une organisation du *code-stream* en unités de données traitables indépendamment les unes des autres. La présentation de ce système de compression fait l'objet de la section 4.1. Un mécanisme de mise en paquets original est présenté en sections 4.2 et 4.3 afin de satisfaire les contraintes liées au transport. Les performances de l'association compression et transport sont ensuite présentées en section 4.4.

4.1 Présentation générale du schéma de compression

Le domaine de la compression d'images a fait l'objet de recherches étendues ces deux dernières décennies (Bovik, 2000), (Taubman et Marcellin, 2002), (Gersho et Gray, 1992). Ces recherches ont prouvé l'efficacité des schémas de compression composés de trois opérations :

la décorrélation des données, la quantification des coefficients transformés, et enfin l'encodage des coefficients issus de la quantification (voir Fig. 4.1(a)).



(a) Schéma général de la compression. (b) Schéma de la compression utilisée.

FIG. 4.1: Principes de compression.

Notre proposition d'un transport partiellement fiable des images fixes compressées est issu de la collaboration avec le thème *Identification, Restauration, Images et Signaux* (IRIS) du CRAN et plus particulièrement avec l'équipe Signaux et Information chargée des recherches en compression d'images, animée par Jean-Marie Moureaux. Le schéma de compression d'images fixes pour lequel nous étudions la proposition d'un transport partiellement fiable repose sur la transformée en ondelettes discrète (TO) et une décomposition multirésolution que nous détaillons en section 4.1.1. L'étape suivante est assurée par la quantification vectorielle algébrique explicitée en section 4.1.2 (voir Fig. 4.1(b)).

4.1.1 Transformée en ondelettes discrète et décomposition multirésolution

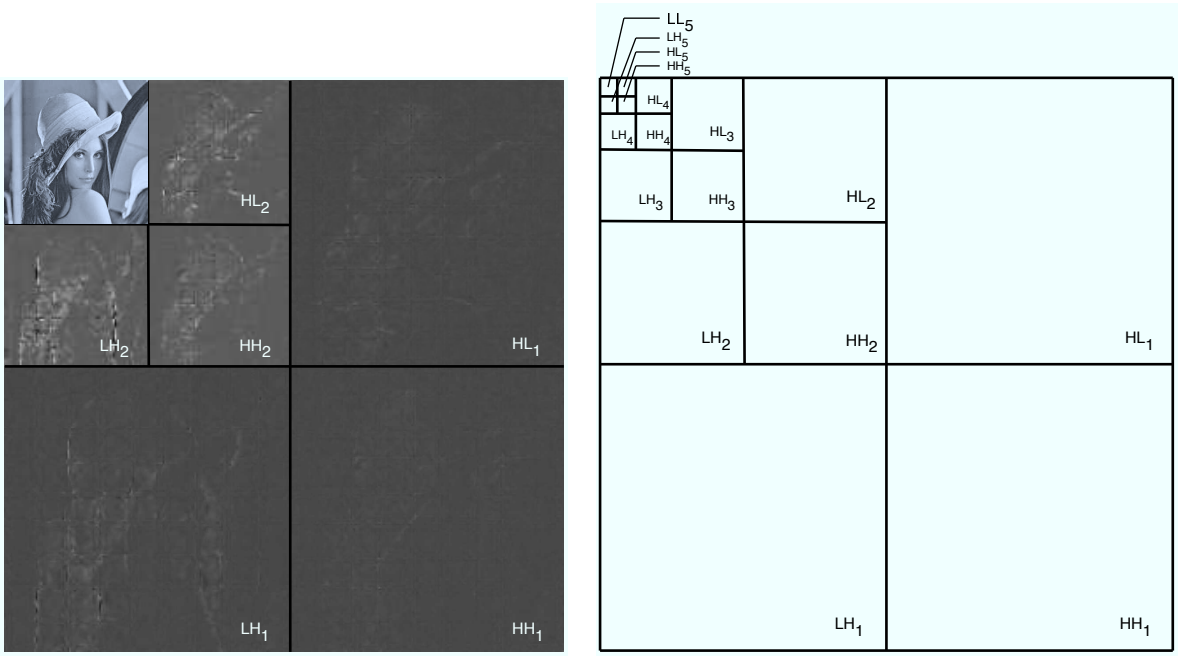
Il est à présent bien établi que la transformée en ondelettes discrète (TO) est l'une des transformées dans le domaine spatio-fréquentiel les plus appropriées à la compression d'images (Antonini *et al.*, 1992). Cette transformée constitue d'ailleurs d'un élément clé de la compression SPIHT (Saïd et Pearlman, 1996) ainsi que du standard JPEG2000, (JTC1/SC29, 2004*b*) qui a été étudié au chapitre précédent. La TO a notamment les propriétés de décorréler les coefficients de l'image et de fournir des coefficients transformés tels que l'énergie soit distribuée essentiellement sur un nombre restreint de coefficients. La distribution de l'énergie des coefficients de l'image source est en effet extrêmement dépendante de l'image, et peu prédictible. Il suffit d'observer les histogrammes fournis par les appareils photos numériques du commerce pour s'en convaincre. Après transformation, la distribution de l'énergie des coefficients transformés est assimilable à une distribution gaussienne généralisée beaucoup plus prédictible qui favorise la compression.

La TO en deux dimensions est appliquée sur la matrice des pixels de l'image. Elle fait appel à des filtres d'analyse pour la décomposition, et à des filtres de synthèse pour la reconstruction. Selon les filtres employés, la décomposition/reconstruction peut être réversible strictement (nécessaire à une compression sans perte) ou bien réversible aux erreurs d'arrondis près. La TO de notre compression est réversible aux erreurs d'arrondis près, elle utilise les filtres de Cohen-Daubechies-Fauveau (CDF) 9-7 à coefficients réels (Antonini *et al.*, 1992) largement reconnus pour leurs performances dans le domaine de la compression d'images. Ceci est d'ailleurs confirmé par l'adoption de ces mêmes filtres dans le standard JPEG2000 pour la compression avec pertes.

Il en résulte une décomposition en quatre sous-bandes dans le domaine fréquentiel. Ces sous-bandes sont des matrices dont les dimensions sont réduites d'un facteur deux. La sous-bande LL est issue d'un filtrage basse-bas dans les directions horizontale et verticale. Puisqu'ils correspondent aux basses fréquences, ses coefficients sont donc les plus significatifs. Il s'agit de l'image source dont la résolution est réduite de moitié dans les deux directions. Les trois autres sous-bandes sont issues des combinaisons restantes de filtrage passe-haut et

passe-bas dans les deux directions. Elles révèlent les contours horizontaux, verticaux et diagonaux, respectivement avec les combinaisons de filtrage horizontal-vertical LH, HL et HH. L'opération inverse consiste à recréer la matrice image dans sa résolution d'origine à partir de ces quatre sous-bandes et les filtres de synthèse.

La décomposition nécessite en général plusieurs niveaux de résolution. Ainsi la sous-bande LL issue de la première TO subit une nouvelle TO, ce qui produit quatre nouvelles sous-bandes, et ainsi de suite. La décomposition finale compte donc une résolution de plus que le nombre de TO successives. La figure 4.2(a) donne un exemple visuel de l'image *Lena* décomposée en trois résolutions, soit deux TO successivement. Les décompositions à base de TO habituellement employées en compression comportent cinq étapes. C'est notamment le cas de notre schéma de compression ou encore de JPEG2000. Ce cas général est illustré par la Fig. 4.2(b). Les coefficients ainsi transformés sont répartis en seize sous-bandes et six résolutions. Ces coefficients sont d'autant plus porteurs de sens du point de vue de la qualité de la reconstruction de l'image que la résolution à laquelle ils appartiennent est petite (produits par les dernières TOs). Il s'agit donc d'une décomposition multirésolution et c'est précisément cette caractéristique que nous allons utiliser pour adapter le niveau de fiabilité du transport en fonction de l'importance des données.



(a) Exemple de décomposition avec deux TO dyadiques.

(b) Décomposition usuelle avec cinq TOs successives.

FIG. 4.2: Décomposition multirésolution à l'aide de la transformée en ondelettes discrète.

4.1.2 Quantification vectorielle algébrique

Comme le montre la figure 4.3, les coefficients issus de la transformée décrite précédemment constituent la source qui fait l'objet d'une quantification. La compression que nous utilisons intègre une Quantification Vectorielle Algébrique (QVA). La QVA est largement détaillée dans la littérature, par exemple dans (Barlaud *et al.*, 1994), (Fisher, 1986) , (Voinson *et al.*, 2002),

(Gersho et Gray, 1992), (Conway et Sloane, 1988). La QVA quantifie les coefficients par groupes, contrairement à la quantification scalaire utilisée par exemple par JPEG2000 qui traite les coefficients individuellement. La QVA utilisée par la compression que nous utilisons ici est particulièrement détaillée par Ludovic Guillemot dans son manuscrit de thèse (Guillemot, 2004).

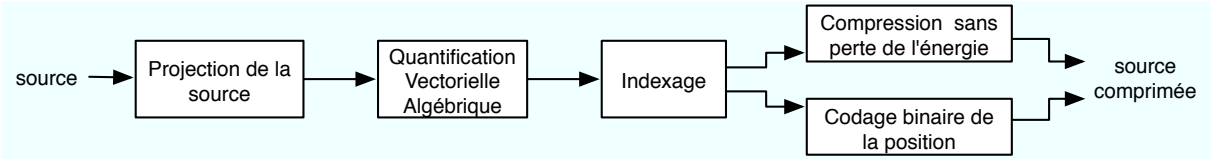


FIG. 4.3: Codage de la source par Quantification Vectorielle Algébrique.

D'après les propriétés de la transformée en ondelette, les coefficients d'ondelettes (hors sous-bande LL) à quantifier sont distribués selon une loi gaussienne généralisée. Une conséquence de cette distribution est que le nombre de coefficients de forte énergie est faible, il en va de même pour les vecteurs de forte énergie. La QVA employée utilise un dictionnaire pyramidal, l'indexage se fait alors selon la norme $L1$ (Guillemot, 2004). Les vecteurs de la source ne sont pas forcément à l'intérieur du dictionnaire, or les vecteurs de forte énergie sont les plus significatifs et la qualité de la reconstruction dépendra de leur bonne quantification. C'est pourquoi une étape de normalisation se charge préalablement de satisfaire cette contrainte en faisant intervenir un facteur de projection γ. La figure 4.4 illustre le dictionnaire pyramidal de norme $L1$ avec l'exemple du réseau $\mathbb{Z}^2$.

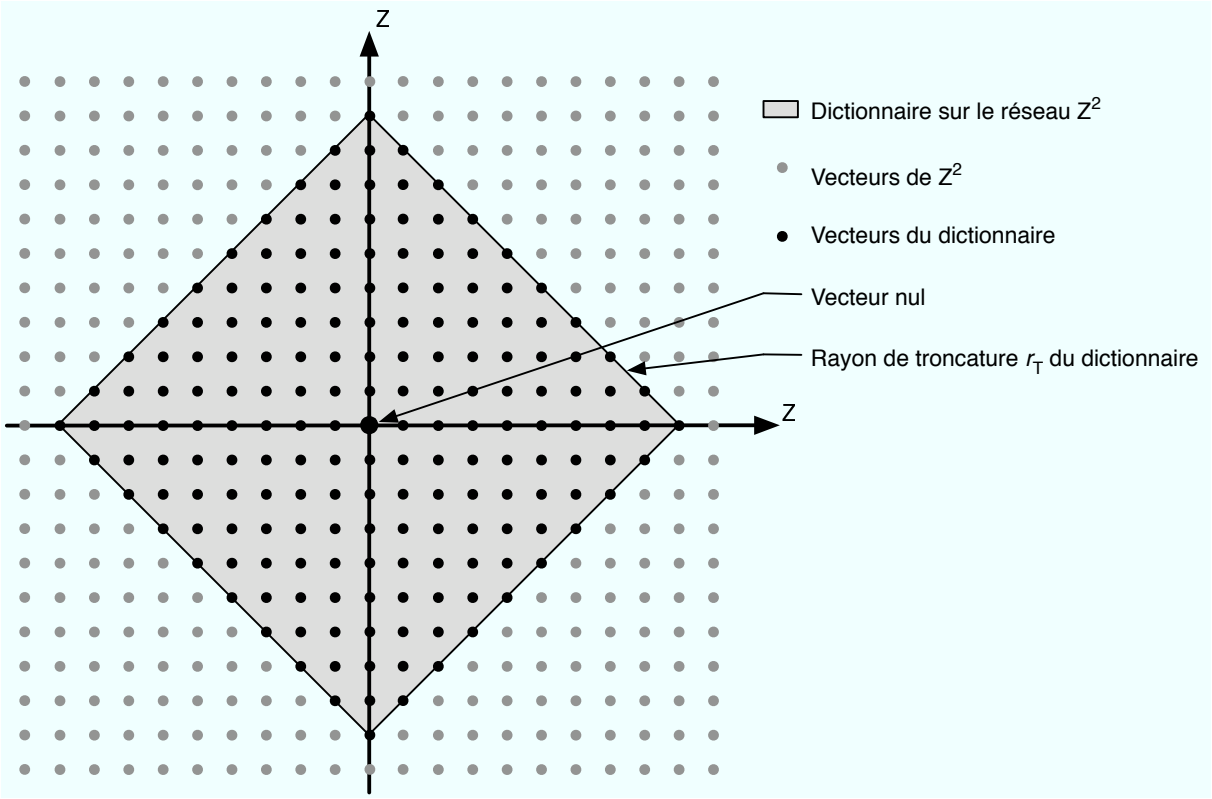


FIG. 4.4: Dictionnaire pyramidal sur le réseau $\mathbb{Z}^2$.

L'approche que nous adoptons est dite intra-bandes, une procédure d'allocation de débit assigne un débit pour chaque sous-bande de sorte à minimiser la distorsion globale (Raffy *et al.*, 1998) et satisfaire le débit moyen final demandé pour la compression.

Les sous-bandes sont décomposées en un certain nombre de vecteurs qui sont formés par des coefficients adjacents. Lorsque les vecteurs sont orientés orthogonalement à la direction du filtre (*i.e.* , parallèlement aux contours), il y a une grande concentration de vecteurs de faible énergie. Cette orientation favorise le regroupement des coefficients significatifs dans un plus petit nombre de vecteurs significatifs eux aussi, ce qui permet un codage plus concis et efficace. Concrètement, les dimensions que nous avons adoptées pour les vecteurs sont les suivantes :

- (2×4) dans les sous-bandes relatives aux contours horizontaux (LH) des résolutions $\mathcal{R}_3$, $\mathcal{R}_4$ et $\mathcal{R}_5$.
- (4×2) dans les sous-bandes relatives aux contours verticaux et diagonaux pour les mêmes résolutions $\mathcal{R}_3$ à $\mathcal{R}_5$.
- (2×2) pour tous les vecteurs des résolutions $\mathcal{R}_1$ et $\mathcal{R}_2$ (issues des deux dernières TOs TO_4 et TO_5) quelle que soit la sous-bande.
- La résolution $\mathcal{R}_0$, *i.e.* , la sous-bande LL issue de TO_5, n'est pas décomposée en vecteurs puisqu'il s'agit de la plus petite résolution $\mathcal{R}_6$ issue d'un filtrage basse fréquence dans les deux directions qui mène directement à une réduction de l'image source d'un facteur 2^5 dans chaque direction.

La QVA fait correspondre à chaque vecteur de coefficients d'ondelettes à quantifier, le vecteur du dictionnaire le plus proche (au sens de l'erreur quadratique moyenne). Plus précisément, c'est l'étiquette du vecteur le plus proche qui est utilisée. Cette étiquette est elle-même composée de deux parties : un préfixe l_n pour la norme du vecteur, et un suffixe l_p pour la position de ce vecteur du dictionnaire parmi l'ensemble des vecteurs ayant la norme l_n (Loyer *et al.*, 2003).

4.1.3 Codage de l'image

Les deux composantes (norme et position) des étiquettes de chaque vecteur sont codées différemment (voir figure 4.5). Comme la distribution des coefficients d'ondelettes n'est pas uniforme (il s'agit d'une distribution gaussienne généralisée), il est judicieux d'encoder la norme l_n à l'aide d'un codeur entropique, tandis qu'un codage binaire direct de la position l_p est approprié car le nombre de positions possibles est déterminé par l'énergie l_p. En d'autres termes, le préfixe l_n est codé avec une longueur variable, contrairement au suffixe l_p dont la longueur du code, qui dépend de la norme, est fixe pour tout vecteur situé sur la sphère de rayon l_n. Les détails de cette opération sont disponibles dans (Voinson *et al.*, 2002).

Du fait de l'approche intra-bande, les vecteurs sont codés indépendamment dans chaque sous-bande. Le code entropique nécessite une table de correspondance et il est décodable sans ambiguïté, la norme peut alors être décodée. Connaissant la norme et les caractéristiques de la QVA (le rayon de la zone morte, le rayon de troncature, le facteur de projection, les dimensions des vecteurs), le décodeur connaît alors la longueur du suffixe du vecteur. Une fois le décodage complet du vecteur effectué, les coefficients sont extraits et l'application doit les placer au bon endroit dans la sous-bande de la résolution appropriée avant de procéder aux transformations inverses qui mènent à la reconstruction de l'image.

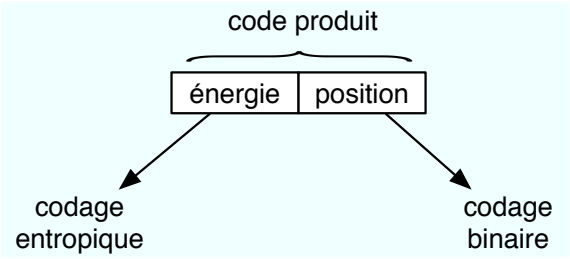


FIG. 4.5: Codages des vecteurs produits.

4.2 Organisation du code et mise en paquets

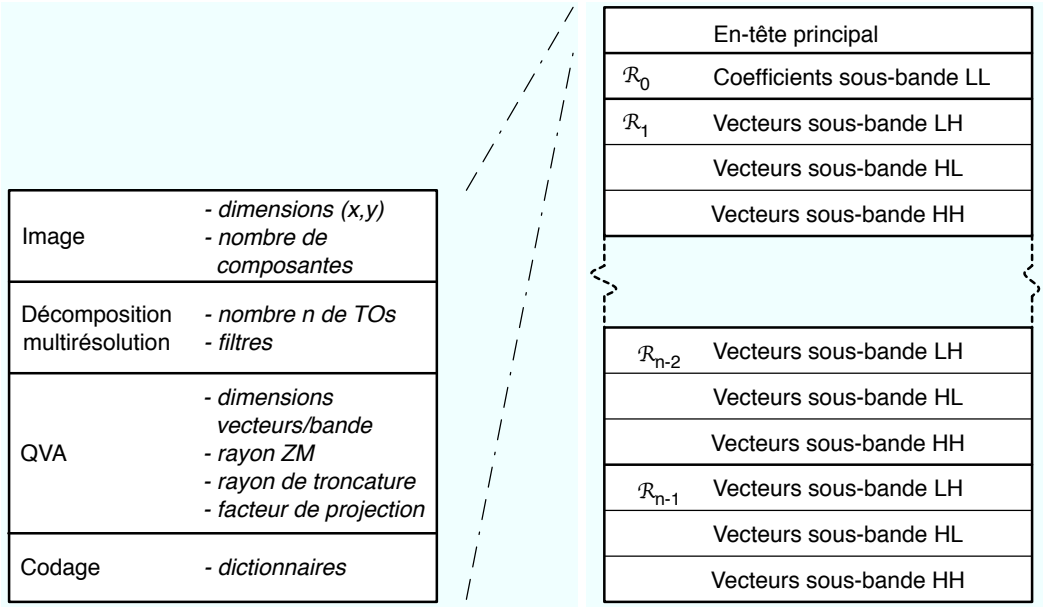
Le code compressé (appelé *code-stream* dans le standard JPEG2000) doit être organisé, autant à des fins d'archivage sous forme de fichier que pour être transmis sur un réseau. L'utilisation de transformées en ondelettes dyadiques , et par conséquent de la décomposition multirésolution, ouvre naturellement la voie à une progression par résolutions. Nous avons vu au chapitre précédent concernant JPEG2000 que lorsque les paquets JPEG2000 étaient ordonnés dans l'ordre croissant des résolutions (RLCP), c'est-à-dire de la basse fréquence LL à la plus grande résolution issue de la première TO, la dégradation de PSNR consécutive à la perte d'un paquet était beaucoup moins liée au rang de ce paquet dans le *code-stream* que lorsque le *code-stream* était organisé par qualité (LRCP).

Ce résultat présenté en section 3.3.2.2, notamment avec les figures 3.11 et 3.13 n'est cependant pas transposable à la compression que nous utilisons. La compression que nous employons n'intègre pas de troncature du code compressé des *code-blocks* de façon à affiner la qualité comme c'est le cas avec JPEG2000. Or c'est cette division du code relatif à chaque bloc de données indépendant (*i.e.* , les *code-blocks*) et leur étalement sur différentes couches qui est à l'origine du comportement mentionné ci-dessus. En progression RLCP, l'absence d'un paquet JPEG2000 relatif à une résolution, une couche de qualité, une composante et un certain nombre de *code-blocks* donnés retire leur sens aux incréments de qualité de ces mêmes *code-blocks* présents à des couches plus fines.

Cet effet n'apparaît donc pas avec la compression que nous utilisons puisque l'unité de traitement élémentaire est le vecteur. Du fait de l'approche intra-bande, un vecteur est un regroupement de coefficients adjacents d'une unique sous-bande, il affecte donc une zone de l'image qui dépend de sa taille et de sa position dans la sous-bande. Par conséquent, la perte d'un vecteur n'affecte ni les autres sous-bandes de la résolution considérée, ni les autres résolutions, et encore moins les autres zones de l'image.

Le *code-stream* complet de l'image compressée devra contenir toutes les informations nécessaires au décodage et à la reconstruction de l'image. C'est-à-dire non seulement les codes des vecteurs de toutes les sous-bandes mais aussi toutes les informations nécessaires à leur interprétation et au décodage. Ces informations concernent les dimensions de l'image, la profondeur de la décomposition multirésolution, la table de correspondance employée pour coder les normes des vecteurs de chaque sous-bande ainsi que les paramètres de la QVA . En dehors des codes des vecteurs, les informations additionnelles requises pour le décodage et la reconstruction de l'image constituent l'en-tête principal du *code-stream* illustré par la figure 4.6(a).

À la suite de l'en-tête principal, le *code-stream* est organisé dans l'ordre croissant des résolutions. Les sous-bandes qui composent une résolution étant indépendantes, il n'y a pas



(a) Constitution de l'en-tête principal. (b) Organisation du *code-stream*.

FIG. 4.6: Constitution et organisation du *code-stream*.

d'ordre particulier imposé. La sous-bande issue du filtrage passe-haut dans les deux directions étant cependant moins porteuse d'informations que les autres, elle sera située au dernier rang de la résolution. Cette organisation est représentée sur la figure 4.6(b). L'exemple de l'organisation du *code-stream* pour les images 512×512 pixels en niveaux de gris décomposées en six résolutions est présenté ci-dessous.

– **En-tête :** Informations nécessaires pour donner un sens à l'ADU (situation des vecteurs contenus, dans quelle résolution, quelle sous-bande ?).

– $\mathcal{R}_0$ **:** Données de la première résolution :
  – les 16×16 coefficients de la sous-bande LL issue de TO_5 ordonnés de gauche à droite et de bas en haut

– $\mathcal{R}_i$ **:** Données de la $i^{ème}$ résolution avec i entier tel que $1 \leqslant i \leqslant 5$:
  – les vecteurs de la sous-bande LH issue de la $TO_{(6-i)}$ ordonnés de gauche à droite et de bas en haut. Les $\left(\frac{512}{2^{6-i}}\right)^2$ coefficients de la sous-bande sont regroupés en vecteurs. Selon la taille de ces derniers, cela représente 64 vecteurs pour $\mathcal{R}_1$, 256 pour $\mathcal{R}_2$, 512 pour $\mathcal{R}_3$, 2048 pour $\mathcal{R}_4$ et 8192 pour $\mathcal{R}_5$.
  – idem pour la sous-bande HL.
  – idem pour la sous-bande HH.

4.2.1 Construction des ADUs

Lorsque l'application souhaite initier la transmission d'une image en utilisant 2CP-ARQ, elle doit former ses unités de données (ADUs) à partir du *code-stream* de l'image. Une ADU doit pouvoir être décodée et traitée par l'application indépendamment des autres. Or comme l'unité élémentaire du code est le vecteur, il est absolument impossible qu'un vecteur soit réparti sur deux ADUs distinctes. Une ADU contiendra donc un nombre entier de vecteurs.

Le niveau de fiabilité est associé à chaque ADU. Selon le principe de la progression par résolution, une résolution donnée est nécessairement transportée par des paquets de même classe, cela s'applique donc à toutes les sous-bandes d'une même résolution. L'ADU est alors constituée d'un nombre entier de vecteurs appartenant à une même résolution dans la limite de la charge admissible. Par convention, les ADUs affectées à une résolution donnée sont remplies avec les vecteurs ordonnés de gauche à droite puis de haut en bas au sein des sous-bandes LH, HL et HH prises dans cet ordre. L'algorithme de formation des ADUs est présenté par la figure 4.8.

Les vecteurs sont codés indépendamment mais l'application doit être en mesure de replacer les coefficients dans leur contexte (résolution, sous-bande et situation dans la sous-bande). En considérant un transport fiable comme TCP, l'application de réception dispose de toutes les ADUs dans l'ordre où elles ont été transmises. Cela revient à disposer du fichier *code-stream* à distance et dans ce cas aucun problème d'interprétation des données ne se pose puisque le *code-stream* n'a subi aucune modification au cours de la transmission.

Si par contre le transport envisagé n'est pas fiable, certains vecteurs peuvent se perdre et l'application pourrait être incapable de resituer les vecteurs consécutifs à une perte alors qu'ils sont pourtant décodables. C'est pourquoi il est nécessaire de préciser le contexte de chaque ADU de manière qu'elles conservent leur sens indépendamment des autres. Comme les vecteurs sont organisés dans le *code-stream* de gauche à droite et de haut en bas dans chaque sous-bande, une possibilité consiste à préciser la situation du premier vecteur inclus dans l'ADU. Cette caractérisation peut être assurée simplement en indiquant la résolution à laquelle l'ADU est liée, ainsi que le numéro du premier vecteur contenu.

Les informations nécessaires à la bonne interprétation des données contenues dans l'ADU sont reportées dans un en-tête pour chacune des ADUs comme le montre la figure 4.7. La résolution est codée sur un champ de 3 bits qui permet un maximum de 8 résolutions, ou encore 7 transformées en ondelettes, ce qui est amplement suffisant. Le numéro du premier vecteur est codé sur 29 bits, soit un maximum de 536 millions de vecteurs par résolution, ce qui suffit à différencier tous les vecteurs des sous-bandes les plus étendues en considérant une image 23000×23000 pixels et des « vecteurs » de dimension 1×1. La surcharge qu'entraîne ce supplément de données s'élève à 32 bits par ADU, soit 0,27% de l'unité de données si la taille de l'ADU est de 1460 octets au total.

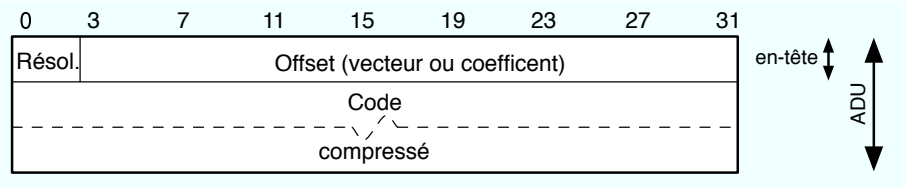


FIG. 4.7: Format des unités de données de l'application (ADUs).

Un tel ordonnancement des informations par résolution pose cependant un problème si l'on considère un transport à fiabilité partielle fondé sur 2CP-ARQ. Tous les p-paquets du flux à transmettre seraient alors transmis en premier, puis les s-paquets suivraient. Cela reviendrait à transmettre la première partie protégée du flux avec TCP, puis utiliser une variante d'UDP (avec l'addition d'un contrôle de congestion à la manière de DCCP) pour transmettre la seconde partie facultative. Ce n'est pas ce que nous souhaitons parce que l'objectif du relâchement de la fiabilité est bien de réduire le temps de service et le trafic en diminuant le nombre de points bloquants (les pertes de p-paquets et leur retransmission). Or si tous les p-paquets

```
/*Initialisation du numéro d'ADU en cours, du compteur de vecteur, du premier vecteur
de l'ADU en cours, du nombre de vecteurs de l'ADU en cours et de la taille de l'ADU en
cours */
adu←0, cpt_vect←0, vect_1←0, n_vect←0, taille_ADU←0
création d'une nouvelle ADU et attribution de sa classe de fiabilité

Formation de l'ADU 1 avec l'en-tête du code-stream

Pour chaque coefficient de la résolution R_0
  Si (taille_coeff<MSS-taille_ADU)
    inclure le coefficient courant dans le corps de l'ADU
    taille_ADU←taille_ADU+taille_coeff
  Sinon
    création d'une nouvelle ADU et attribution de sa classe de fiabilité
    adu←adu+1
    inclure le vecteur courant dans le corps de l'ADU
    taille_ADU←taille_ADU+taille_coeff

Pour chaque résolution R_i avec i entier de 1 à 5
  création d'une nouvelle ADU et attribution de sa classe de fiabilité
  adu←adu+1
  n_vect←0
  vect_1←cpt_vect
  Pour chaque sous-bande
    Pour chaque vecteur de la sous-bande
      Si (taille_vecteur<MSS-taille_ADU)
        inclure le vecteur courant dans le corps de l'ADU
        taille_ADU←taille_ADU+taille_vecteur
        n_vect←n_vect+1
        cpt_vect←cpt_vect+1
      Sinon
        création d'une nouvelle ADU et attribution de sa classe de fiabilité
        adu←adu+1
        n_vect←0
        vect_1←cpt_vect
        inclure le vecteur courant dans le corps de l'ADU
        taille_ADU←taille_ADU+taille_vecteur
        n_vect←n_vect+1
        cpt_vect←cpt_vect+1
```

FIG. 4.8: Algorithme de formation des paquets 2CP-ARQ à partir des vecteurs de coefficients de l'image.

sont contigus, il y aura inévitablement plus de paquets anticipés inefficacement et le trafic sera supérieur. Le modèle de 2CP-ARQ présenté au chapitre 2 suppose que les paquets des deux classes soient mélangés, et nous proposons une méthode pour rendre vraie cette hypothèse dans la section suivante.

4.3 Mélangeur de paquets

L'objectif du mélangeur de paquets est d'ordonnancer le flux de paquets à l'émission, de façon à intercaler p- et s-paquets le plus efficacement possible. En sortie du mélangeur, les séquences d'ADUs consécutives de même classe doivent être minimisées. Cet ordonnancement doit être effectué en fonction du taux de fiabilité τ_F.

Ce type de problématique est typiquement celui de la recherche en ordonnancement des tâches. C'est particulièrement le cas de l'ordonnancement (m,k)-firm dont l'un des objectifs est de déterminer un ordonnancement permettant de satisfaire la contrainte (m,k)-firm, c'est-à-dire satisfaire l'exécution dans les temps de m instances de la tâche sur k. Autrement dit, il s'agit de déterminer quelles sont les m instances d'une tâche qui sont critiques parmi k, les $k-m$ restantes sont classées optionnelles et peuvent à ce titre ne pas être exécutées de façon à laisser libres les ressources pour le traitement d'une instance requise d'une autre tâche, (Jia *et al.*, 2005).

La similitude entre la contrainte (m,k)-firm et la fiabilité selon 2CP-ARQ est évidente : avec 2CP-ARQ, il s'agit de garantir le transport de m p-paquets parmi les k paquets (différents) que compte le transfert.

Ramanathan a présenté dans (Ramanathan, 1999) un algorithme d'ordonnancement (m,k)-firm qui qualifie d'obligatoire l'instance a d'une tâche τ si la condition suivante est satisfaite, dans le cas contraire l'instance est optionnelle :

$$a = \left\lfloor \left\lceil \frac{a \times m}{k} \right\rceil \times \frac{k}{m} \right\rfloor$$

Cet algorithme repose sur le principe de la construction de mots mécaniques (mots de Sturm). Énoncé différemment, il s'agit de construire un mot binaire Π de longueur k bits qui comporte m bits à 1 selon la formule 4.1. Ce mot est aussi appelé (m,k)-motif, la position de chaque bit à 1 correspond aux instances obligatoires de la tâche.

$$\Pi(n) = \begin{cases} 1 \text{ si } n = \left\lfloor \left\lceil n \times \frac{m}{k} \right\rceil \times \frac{k}{m} \right\rfloor \quad n = 0,1,\ldots,k-1 \\ 0 \text{ sinon} \end{cases} \tag{4.1}$$

Cette construction de mots mécaniques peut être utilisée par l'application source pour mélanger les ADUs. Les bits à 1 du mot binaire obtenu correspondent au rang des ADUs qui seront encapsulées dans des p-paquets et réciproquement, les 0 indiquent des ADUs transportées dans des s-paquets. La source peut ainsi fournir les ADUs à la couche transport en piochant alternativement dans les files des unités cruciales et optionnelles. L'exemple suivant illustre ce mécanisme.

Exemple

On suppose que l'application source dispose de 50 ADUs dont 20 sont requises. Le flux de paquets correspondant est donc constitué de 50 paquets différents dont 20 sont des p-paquets ($\tau_F = 0,4$). Le mot binaire construit selon la formule 4.1 est le suivant :

`10100101001010010100101001010010100101001010010100`
L'application source devra donc fournir les ADUs à la couche transport dans cet ordre : la première ADU fournie sera une ADU requise (un p-paquet au niveau transport), la seconde sera une ADU optionnelle, puis une requise et ainsi de suite conformément au mot binaire comme le montre la figure 4.9.

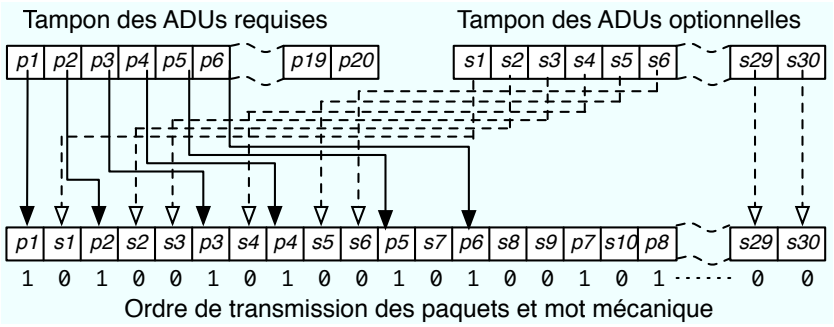


FIG. 4.9: Ordre d'émission selon le mot mécanique construit en fonction du taux de fiabilité.

Les ADUs seront donc remontées dans le désordre à l'application réceptrice. Ce désordre ne représente pas un problème car selon le principe ALF, les ADUs peuvent être traitées indépendamment par l'application. C'est en l'occurrence le cas puisque les informations relatives à la résolution et la sous-bande, ainsi que le numéro du premier vecteur inclus dans l'ADU, renseignent la signification des données reçues à l'application de réception.

4.3.1 Construction des paquets

Chez l'émetteur, la fonction principale de la couche transport, (2CP-ARQ en l'occurrence), consiste à demander les ADUs à la couche supérieure à l'aide de la primitive `DATA_REQUEST`, les encapsuler dans des paquets, les passer à la couche réseau, analyser les acquittements et retransmettre en cas de perte de p-paquets.

En réception, la couche 2CP-ARQ doit analyser les en-têtes de paquets, acquitter les paquets reçus et passer les données valides à la couche application à l'aide de la primitive `DATA_INDICATION` (ADUs non consécutives à la perte d'un p-paquet).

L'application fournit les ADUs à 2CP-ARQ en spécifiant pour chacune le niveau de fiabilité. Puis 2CP-ARQ encapsule les ADUs dans l'ordre selon lequel elles lui sont données. Cette encapsulation dans des paquets nécessite de traduire le niveau de fiabilité demandé (ADU nécessaire ou optionnelle) à l'aide du couple de compteurs (i, j) comme nous l'avons présenté au chapitre 2. Ces compteurs sont par conséquent inclus dans l'en-tête de paquet 2CP-ARQ. Cet en-tête contient également les champs correspondant au numéro de séquence des paquets (ou acquittements), la longueur et une somme de contrôle permettant de vérifier l'intégrité du paquet, ainsi que les numéros de ports nécessaires à toute application qui utilise IP (voir figure 4.10).

Le couple de compteurs (i, j) permet d'indiquer la classe de fiabilité. Le numéro de séquence est utile pour la remise en ordre des paquets à l'arrivée. Il a en effet été montré dans les sections 2.1.1.1 et 2.1.1.2 que les compteurs (i, j) ne pouvaient pas assurer cette fonction en toutes circonstances. Un numéro d'acquittement permet au récepteur d'acquitter un paquet particulier. Une communication bidirectionnelle peut être envisagée avec un paquet qui acquitte tout en transportant des données utiles. La longueur et la somme de contrôle sur 16 bits cha-

cune permettent à 2CP-ARQ de détecter les corruptions de données qui ne l'auraient pas été par les couches inférieures.

Un tel en-tête 2CP-ARQ nécessite 16 octets (voir figure 4.10), c'est-à-dire le double de celui associé aux datagrammes UDP mais 4 de moins que TCP. Avec ces informations, 2CP-ARQ est en mesure de détecter les pertes de paquets, piloter les corrections de p-paquets et de remonter la charge des paquets (les ADUs) à la couche application. Tous comptes faits, en-têtes, 2CP-ARQ et ADUs compris, la surcharge totale est de 20 octets, soit autant qu'avec TCP.

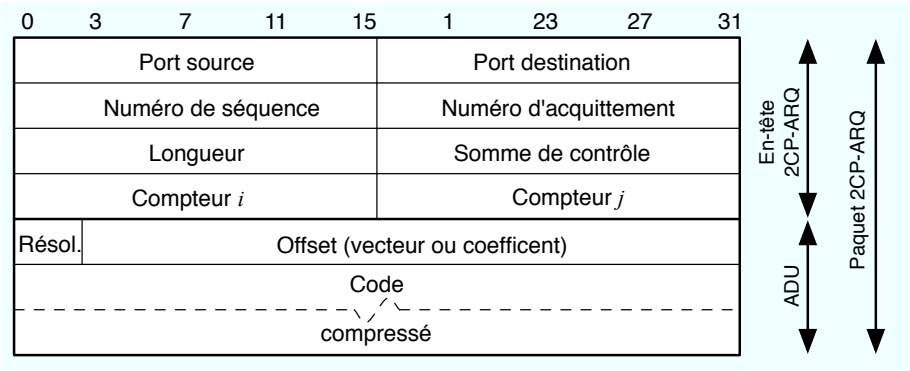


FIG. 4.10: Format des paquets 2CP-ARQ.

Les procédures de mise en paquets étant définies, les questions auxquelles nous allons répondre concernent les performances d'un tel transport associé aux images compressées selon notre schéma de compression.

4.4 Évaluation de performances

L'évaluation des performances du transport à fiabilité partielle des images sur un réseau en utilisant 2CP-ARQ comprend deux aspects : l'étude du gain de temps que permet le relâchement de la contrainte de fiabilité d'une part, et la dégradation de qualité que la fiabilité partielle provoque sur les images d'autre part.

Les aspects temporels liés à 2CP-ARQ ont déjà été traités au chapitre 2. Le temps du service est étroitement lié au taux de fiabilité du transport, lequel dépend de la politique de fiabilité sélectionnée par l'application.

Ce niveau de fiabilité doit être réglé de façon que l'image reconstruite ait une qualité satisfaisante. La définition de « qualité satisfaisante » dépend de l'utilisateur mais surtout de l'application visée. Il est en effet évident que la transmission des images et des éléments graphiques, dont le but est d'agrémenter les pages web, est moins critique que la transmission de photographies artistiques, d'images médicales à vocation de diagnostic, ou encore des images militaires. Les éléments graphiques qui ne sont pas primordiaux peuvent donc s'accommoder d'une politique de fiabilité moins stricte tout en satisfaisant les exigences de l'application (lisibilité et non introduction de fausse information).

Nous considérons ici le contexte où les images n'ont qu'une fonction d'agrément graphique, comme c'est par exemple la cas sur la plupart des pages web. La qualité de la reconstruction peut être dégradée dans la mesure où la lisibilité est conservée (caractères alphanumériques éventuels) et que cette diminution de qualité ne soit pas évidente pour l'utilisateur. La section

4.4.1 présente les expérimentations qui ont été menées afin de déterminer la sensibilité aux pertes de la reconstruction d'image. Ce travail permet de choisir une politique de fiabilité en section 4.4.2, puis d'analyser les performances en termes de dégradation de qualité et de gain de temps du service, en fonction du taux de pertes de paquets et du taux de compression dans les sections suivantes.

4.4.1 Analyse expérimentale

Dans le chapitre précédent, nous avons vu avec JPEG2000 qu'il était difficile de concilier la compression et les pertes additionnelles post-compression, et ce d'autant plus que la compression était forte. Les faibles taux de compression permettent évidemment une plus grande tolérance aux pertes. Or en général, lorsque la finalité de la compression est le transfert d'images sur des réseaux, le taux de compression est de préférence élevé. C'est pourquoi nous proposons d'étudier la sensibilité aux pertes pour des taux de compression allant d'un niveau moyen à élevé.

Les expérimentations consistent à simuler le transport 2CP-ARQ des images à l'aide de l'outil logiciel MATLAB. Les différentes étapes qui constituent la simulation sont les suivantes :

1. compression de l'image source selon le taux de compression visé (décomposition en six résolutions, allocation de débit, QVA),
2. formation des ADUs dans la limite du MSS (cf. 4.2.1),
3. marquage des ADUs conformément à la politique de fiabilité souhaitée (*i.e.* , les résolutions à protéger),
4. pertes de certains paquets non protégés en accord avec le taux de pertes de paquets (équiprobabilité de perte quel que soit le paquet, conformément à la section 2.2.1),
5. reconstruction de l'image. Pour ce faire, les pertes de paquets sont simulées avec la substitution des vecteurs concernés par des vecteurs nuls,
6. analyse de la qualité en termes de PSNR et par contrôle visuel.

4.4.1.1 Caractéristiques des images avant transmission

Les expérimentations ont été réalisées sur trois images sources bien connues de la communauté du traitement d'images : *Lena*, *Boat* et *Peppers*. Il s'agit d'images 512$\times$512 pixels à 256 niveaux de gris dont les sujets et le type de détails diffèrent (voir Fig. 4.11).

Les cinq taux de compression réalisés C_1 à C_5 ont pour valeur $8:1$, $16:1$, $46:1$, $60:1$ et $80:1$ respectivement. Les images *Lena* et *Peppers* supportent mieux les taux de compression élevés que l'image *Boat* en raison du nombre important de détails perceptibles que la compression tend à gommer sur l'image *Boat*. C'est pourquoi nous avons restreint les expérimentations aux taux de compressions C_1 à C_3 pour cette image. La table 4.1 indique le PSNR résultant de la compression des images pour les taux considérés. Ces valeurs de PSNR, comme toutes les valeurs que nous indiquons, sont calculées par rapport à l'image source non compressée. Il en va donc de même pour les valeurs de PSNR calculées après perte d'information post-compression.

4.4.1.2 Influence du réseau sur les paramètres de simulation

Nous avons étudié les paramètres correspondant à deux types de connexion réseau largement répandues qui sont la connexion par modem sur ligne analogique classique (RTC) et une

(a) Image *Lena*. (b) Image *Boat*. (c) Image *Peppers*.

FIG. 4.11: Images de référence : *Lena, Boat* et *Peppers*

TAB. 4.1: PSNR en fonction de l'image et du taux de compression.

Compression	C_1	C_2	C_3	C_4	C_5
Débit réalisé (bits/pixel)	1,04	0,52	0,17	0,12	0,10
PSNR boat (dB)	36,97	33,36	28,34	—	—
PSNR peppers (dB)	36,61	34,75	29,99	29,78	28,92
PSNR Lena (dB)	38,60	36,25	31,31	30,58	29,46

connexion ADSL 1Mbit/s. Lorsque les conditions réseau sont bonnes pour une communication donnée (pas de congestion, hôtes en mesure de traiter les données lorsqu'elles arrivent), la connexion de l'abonné à l'Internet est un facteur limitant. La technologie et les caractéristiques de la connexion vont directement influer sur la taille maximale des unités de transport et leur débit, mais aussi sur les retards qui sont introduits et donc sur le nombre de paquets qui pourront être émis par anticipation.

Les paramètres requis pour l'analyse du temps de service sont les suivants :

– $t_{pack} = \frac{MTU \times 8}{BW}$
– N_{max}, voir eq. (2.36)

Les processus de formation des ADUs et de leur mise en paquets requièrent la connaissance du MSS. Considérons une connexion au réseau par Ethernet avec un MTU valant 1500 octets. Compte-tenu des vingt octets des en-têtes d'ADU et de paquets 2CP-ARQ (cf. 4.3.1) auxquels s'ajoutent les vingt octets d'en-tête IP ainsi qu'éventuellement les huit octets d'encapsulation PPPOE, la taille maximale de la charge utile des ADU s'élève alors à 1452 octets.

Le délai nécessaire à l'acheminement d'un paquet est un paramètre d'entrée du modèle, tout comme le taux de pertes de paquets. Le délai caractéristique a été évalué d'après la mesure du demi délai aller-retour. Ces mesures ont été effectuées depuis une machine connectée alternativement via une ligne DSL 1Mbits/s puis une connexion RTC 56kbits/s avec l'envoi d'un paquet de la plus grande taille possible dans la limite du MTU à destination de sites variés français et étrangers. L'évaluation du temps t_{AR} suppose donc que les chemins aller et retour sont identiques. Les paramètres du modèle présenté au chapitre 2 sont indiqués dans le tableau 4.2.

TAB. 4.2: Paramètres en entrée du modèle de 2CP-ARQ.

	DSL 1024kbits/s	*RTC 56kbits/s*
MTU (octets)	1492	576
t_{AR} (ms)	100	250
t_{pack} (ms)	11,7	82.3
N_{max} (paquets)	8	3

4.4.2 Choix d'une politique de fiabilité

Généralement, que ce soit avec JPEG2000 ou notre compression, la décomposition multirésolution compte cinq transformées en ondelettes qui donnent lieu à six résolutions de taille croissante et d'importance décroissante : $\mathcal{R}_0$ à $\mathcal{R}_5$. $\mathcal{R}_0$ est l'image réduite d'un facteur 2^5 suite à un filtrage passe-bas, elle est donc la résolution la plus importante. Sa transmission sera donc naturellement assurée par des p-paquets, mais quelle frontière entre deux résolutions successives doit-on choisir pour constituer le seuil à partir duquel la transmission sera réalisée par des s-paquets ?

En tenant compte d'une décomposition en six résolutions, il existe sept politiques de fiabilité de niveaux croissants F_0 à F_6 pour transporter une image avec 2CP-ARQ.

F_0 **:** aucune résolution n'est protégée, le transport n'est pas fiable (UDP).

F_1 **:** seule la résolution issue de la sous-bande basse fréquence LL ($\mathcal{R}_0$) est protégée.

F_2 **:** $\mathcal{R}_0$ à R_1 sont fiabilisées (Fig. 4.12(a)).

F_3 **:** $\mathcal{R}_0$ à R_2 sont fiabilisées (Fig. 4.12(b)).

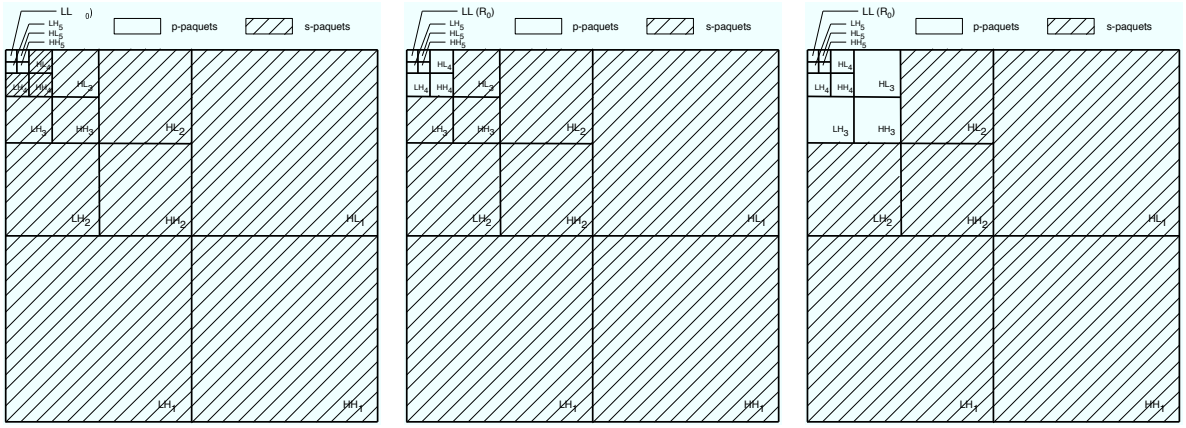
F_4 **:** $\mathcal{R}_0$ à R_3 sont fiabilisées (Fig. 4.12(c)).

F_5 **:** $\mathcal{R}_0$ à R_4 sont fiabilisées.

F_6 **:** le transport est alors totalement fiable (TCP).

Nous cherchons la politique de fiabilité qui permet d'obtenir une qualité de reconstruction acceptable. Cette définition dépend de l'application visée. Nous rappelons que l'objectif est ici de permettre une reconstruction dont la baisse de qualité n'est pas perceptible par l'utilisateur destinataire, ce qui conduit à écarter la politique de fiabilité F_0 qui n'offre aucune garantie. Le transport doit être partiellement fiable. Il est par conséquent exclu de protéger toutes les résolutions sinon cela reviendrait à un transport totalement fiable. Le niveau de fiabilité F_6 est écarté pour cette raison. F_1 et F_5 ont également été écartées parce que la qualité minimale garantie est évidemment trop basse dans le premier cas, et le taux de fiabilité associé à F_5 est tel que le gain de temps potentiel est très faible. Nous avons par conséquent focalisé l'analyse sur les trois politiques de fiabilité intermédiaires F_2, F_3 et F_4 illustré sur la figure 4.12.

Une fois la mise en paquets réalisée, il est possible de calculer le taux de fiabilité des paquets τ_F qui correspond. Les nombres de p-paquets, de s-paquets et le taux de fiabilité τ_F par image sont indiqués dans la table 4.3 en fonction de la combinaison du taux de compression, de la valeur du MTU et de la politique de fiabilité. Il est important de remarquer que les valeurs sont identiques quelle que soit l'image (*Lena*, *Peppers* ou *Boat*). Deux raisons à cela : d'une part, les débits alloués à chaque sous-bande varient dans des proportions faibles pour ces trois images, d'autre part les paquets sont rarement tous remplis. Les différences de volumes de données selon l'image sont en effet suffisamment faibles pour être absorbées par la taille de la charge maximale des paquets. Les paquets sont en effet rarement tous chargés au



(a) Niveau de fiabilité F_2 : seules les deux plus petites résolutions sont fiabilisées.

(b) Niveau de fiabilité F_3 : les trois résolutions les plus importantes sont protégées.

(c) Niveau de fiabilité F_4 : les quatre résolutions les plus importantes sont protégées.

FIG. 4.12: Trois politiques de niveaux de fiabilité différents sont évaluées, les informations fiabilisées sont toujours les plus importantes au sens de la progression par résolutions.

maximum de leur capacité, c'est en particulier le cas du dernier paquet relatif à une résolution donnée qui peut de ce fait amortir les différences de volumes.

TAB. 4.3: Volumes de paquets et taux de fiabilité (p-/s-/p-ratio) en fonction de la politique de fiabilité, du taux de compression et de la valeur du MTU.

Taux de compression/MTU	F_2	F_3	F_4
C_1/1492 (DSL)	2/25/8%	3/24/12%	6/21/23%
C_1/576 (RTC)	2/65/3%	5/62/8%	12/55/18%
C_2/1492 (DSL)	2/13/14%	3/12/20%	6/9/40%
C_2/576 (RTC)	2/34/6%	5/31/14%	12/24/34%
C_3/1492 (DSL)	2/5/29%	3/4/43%	5/2/72%
C_3/576 (RTC)	2/11/16%	4/9/31%	8/5/62%
C_4/1492 (DSL)	2/4/34%	3/3/50%	5/1/84%
C_5/1492 (DSL)	2/3/40%	3/2/60%	4/1/80%

La qualité de la reconstruction des images est évaluée pour chaque combinaison de politique de fiabilisation des résolutions, d'images et de taux de compression. Cette mesure de qualité est effectuée de manière objective en termes de PSNR, mais aussi de manière subjective avec un contrôle visuel de l'image reconstruite.

4.4.2.1 Dégradation de qualité objective à des taux de pertes de paquets usuels

Les mesures de PSNR ont été réalisées sur 2000 images reconstruites pour chaque combinaison d'image, de taux de compression, de taux de pertes de paquets et de politique de fiabilité. Les paquets perdus sont choisis parmi ceux qui ne sont pas protégés par la politique de fiabilité en vigueur. Le choix des pertes parmi ces s-paquets est réalisé selon une loi uniforme, tous ont la même probabilité d'être perdus.

Il est évident que plus la politique de fiabilité est forte (nombre important de résolutions fiabilisées), plus la reconstruction est proche de la version sans perte de paquets. En termes

de PSNR, cela signifie une variance inférieure lorsque la politique de fiabilité est forte. La politique de fiabilité doit donc être choisie en fonction des tolérances de l'application vis-à-vis de la dégradation. Il est également évident que le taux de pertes de paquets, qui peut être vu comme une caractéristique du réseau, a aussi un effet sur le PSNR. Plus le taux est élevé et plus la dégradation est importante en termes de moyenne et de variance. La politique de fiabilité doit donc être choisie non seulement en fonction des tolérances de l'application, mais aussi en fonction des caractéristiques du réseau, notamment du taux de pertes de paquets.

Les PSNRs des reconstructions sont présentés sur les figures 4.13, 4.14 et 4.15 sous forme d'histogrammes, respectivement pour les politiques de fiabilité F_2, F_3 et F_4. Pour chaque politique de fiabilité, nous montrons l'évolution du PSNR pour les trois images à trois taux de compression différents et deux taux de pertes de paquets (faible : 1% et élevé : 10%). Toutes les combinaisons d'images/taux de compression/taux de pertes ne sont pas présentées pour des raisons évidentes d'espace mais aussi parce que toutes choses égales par ailleurs, l'image influe très peu sur la variance du PSNR, même remarque quant au taux de compression ou au taux de pertes.

L'augmentation de la variance du PSNR sur l'ensemble des 2000 images reconstruites est visible lorsque l'on compare l'histogramme correspondant à une combinaison d'image/taux de compression/taux de pertes pour différentes politiques de fiabilité. La figure 4.13(a) montre la distribution des 2000 PSNR pour l'image *Lena* faiblement compressée (1bit/pixel) lorsque le taux de pertes de paquets est faible (1%) et en considérant le niveau de fiabilité faible F_2. Le resserrement du PSNR dans les hautes valeurs est visible sur les figures 4.14(a) et 4.15(a) respectivement pour les niveaux de fiabilité $F3$ et F_4. Dans l'ordre croissant du niveau de fiabilité demandé, le PSNR moyen est de 38,08dB pour F_2, 38,12dB pour F_3 et 38,36dB pour F_4. Si le PSNR augmente peu dans ce cas précis lorsque la politique de fiabilité devient plus restrictive, le gain est par contre plus sensible en termes de variance. La variance du PSNR pour l'image *Lena* au taux de compression C_1 avec 1% de pertes est de 3,010dB^2 pour F_2, 2,320dB^2 pour F_3 et enfin 0,567dB^2 pour F_4.

La même conclusion s'applique pour toutes nos combinaisons d'images et de taux de compression lorsque les pertes de paquets sont faibles (1%). Voir les figures 4.13(c), 4.15(c), 4.15(c) en ce qui concerne l'image *Boat* au taux de compression moyen C_3 (46 :1) et les figures 4.13(e), 4.14(e), 4.15(e) pour l'image *Peppers* avec un fort taux de compression (80 :1). Notons que lorsque le taux de pertes de paquets est faible, la distribution du PSNR est située vers les fortes valeurs quelle que soit la politique de fiabilité. Par exemple dans le cas de *Lena* C_1 et de la fiabilité F_2, près de 1600 des 2000 reconstructions ne sont pas altérées par des pertes de paquets par le réseau et très peu de reconstructions ont un PSNR très dégradé entre 25 et 25dB (voir figure 4.13(a)). Avec une politique de fiabilité très forte, l'amélioration est peu sensible puisqu'à peine plus de 1600 reconstructions ont le PSNR maximum (voir figure 4.15(a)). Nous pouvons donc dire qu'une politique de fiabilité peu protectrice comme F_2 est adaptée aux réseaux qui sont fortement perturbés par les pertes de paquets. Bien sûr, un taux de pertes faible n'est pas le plus intéressant pour 2CP-ARQ puisque que les pertes autorisées sont par définition peu nombreuses et le gain de temps possible est lui aussi très faible (cf. chapitre 2).

La variance du PSNR suit aussi le même chemin que le taux de pertes de paquets. Si on reprend le même exemple que précédemment avec *Lena* C_1, la comparaison des histogrammes pour différents taux pertes et une politique de fiabilité donnée met en évidence un élargissement du PSNR vers la gauche. Voir par exemple les figures 4.13(a) et 4.13(b) lorsque la fiabilité est faible. Dans cet exemple, la variance de 3,01dB^2 pour 1% de pertes de paquets augmente à 17,48dB^2 lorsque les pertes s'élèvent à 10%. L'histogramme 4.13(b) montre qu'à

10% de pertes, le PSNR est extrêmement variable, il y a même plus de 50% des 2000 reconstructions qui enregistrent une dégradation supérieure à 2dB comparés à moins de 10% lorsque le taux de pertes vaut 1%.

Une telle variance avec un taux de pertes n'est pas compatible avec une application de transfert d'images, même si les contraintes sont faibles. Dans le cas de *Lena*, F_2 et 10% de pertes de paquets, il y a 200 reconstructions sur 2000 qui se situent dans la tranche des 22-26dB, soit une diminution de plus de 12dB ! Si une politique de fiabilité peu protectrice est utilisable pour transporter des images lorsque le réseau perd peu de paquets, ce n'est plus le cas à mesure que le taux de pertes de paquets augmente. À 10% de pertes et en considérant les images faiblement compressées à 1bit/pixel, il faut déjà recourir à F_4 pour endiguer l'élargissement vers la gauche de l'histogramme (voir figure 4.15(b)) et retrouver une variance de 3dB^2 équivalente à celle obtenue avec F_2 et 1% de pertes.

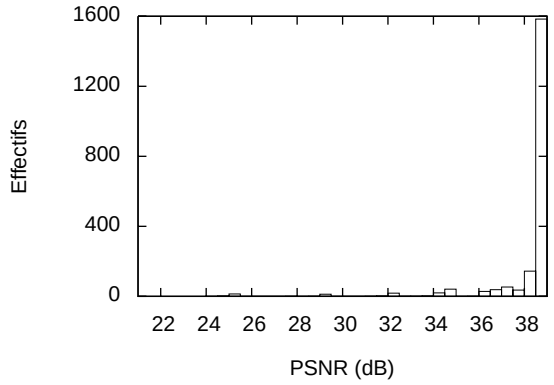
Le taux de compression est un paramètre qu'il faut prendre en considération en présence de pertes de paquets significatives : plus le taux de compression est élevé et moins la variance est importante. Cette remarque est vérifiable quel que soit le taux de pertes en comparant les histogrammes. La compression explique cela car un taux de compression élevé implique une réduction du volume des données surtout dans les grandes résolutions. Ces grandes résolutions sont alors moins significatives et comme ce sont celles qui sont par définition non protégées, les pertes sont également peu significatives relativement à un taux de compression plus faible.

Les figures 4.13(d) et 4.14(d) montrent que la politique F_3 permet de réduire la variance de 3,73dB^2 à 1,77dB^2. La fiabilisation F_3 apparaît donc suffisante pour que la variance passe sous les 3dB^2 lorsque le taux de compression est C_3 (46 :1) alors que pour le taux C_1 il faut recourir à F_4. Compte-tenu de la remarque du paragraphe précédent, la même politique de fiabilité F_2 est donc également suffisante lorsque la compression est plus forte encore. Les histogrammes 4.13(f) et 4.14(f) le prouvent avec une évolution de la variance qui passe de 4,45dB^2 à 1,30dB^2 en changeant la fiabilisation F_2 par F_3.

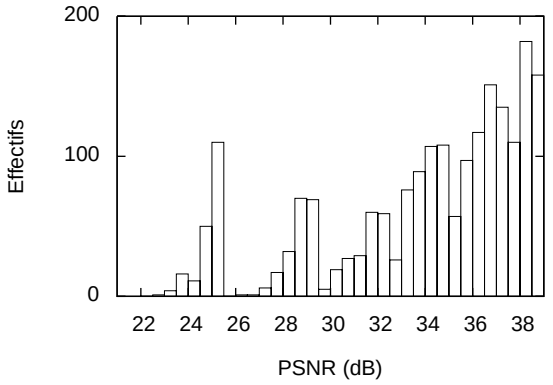
L'étude d'un critère de qualité objective comme le PSNR n'est pas suffisant pour statuer sur la satisfaction ou non des exigences de l'application par une politique de fiabilité. Il est en effet délicat de traduire les exigences de l'application par un PSNR moyen ou une variance tolérée. Pour ce faire, il est nécessaire de contrôler visuellement les images.

4.4.2.2 Dégradation de qualité subjective

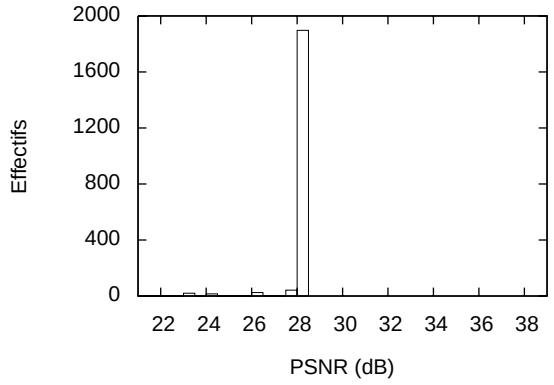
Une fiabilisation moindre semble assurer des reconstructions plus homogènes avec les taux de compression élevés que les taux plus faibles. La planche d'images de la figure 4.16 montre que ce n'est pas si évident. Cette planche est constituée d'images *Lena* compressées avec un taux de 80 :1. Les images de la première ligne 4.16(a), 4.16(b) et 4.16(c) sont des échantillons caractéristiques obtenus avec un taux de pertes de paquets de 10%, respectivement avec les résolutions protégées selon F_4, F_3 puis F_2. Il est évident que la dégradation de qualité est immédiatement sensible, et ce même sans comparer à l'image sans perte de paquet 4.16(d). La dégradation correspond à une chute de PSNR de 5,4dB pour F_2 et 3,8dB pour F_3, mais la dégradation perçue visuellement n'est pas en rapport avec ces chiffres. La perte de détails et surtout l'apparition d'artefacts sont évidentes et ne sont n'est pas retranscrites par le PSNR ou la chute de PSNR. Il est en effet difficile de décider si la qualité de la reconstruction est satisfaisante du point de vue des exigences de l'application avec l'unique connaissance du PSNR ou de la chute de PSNR qu'implique le relâchement de la fiabilité. Après examen visuel des images de la planche 4.16, nous pouvons donc conclure que les politiques de fiabilité



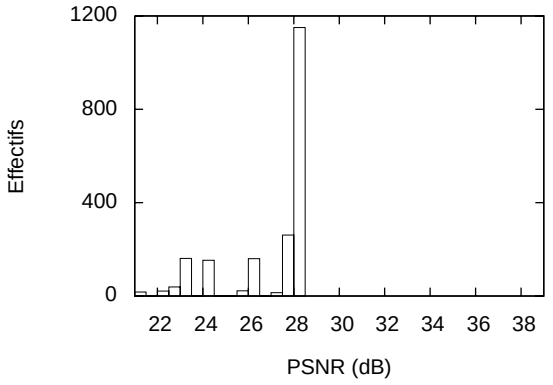
(a) *Lena*, C_1, $p_{pack} = 1\%$, $\overline{PSNR} = 38,08dB$, $var(PSNR) = 3,010dB^2$.



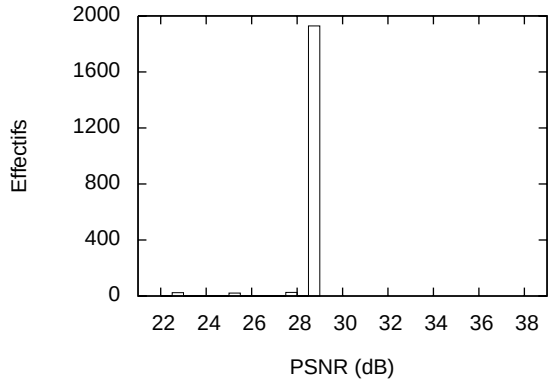
(b) *Lena*, C_1, $p_{pack} = 10\%$, $\overline{PSNR} = 33,99dB$, $var(PSNR) = 17,480dB^2$.



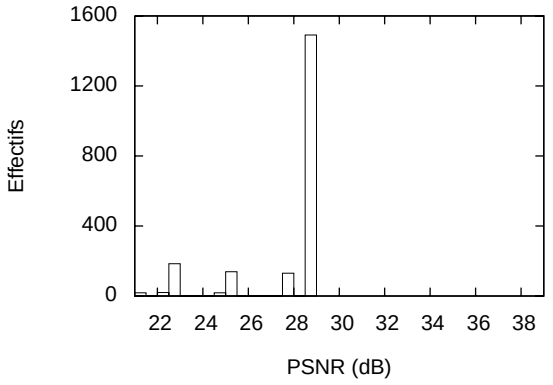
(c) *Boat*, C_3, $p_{pack} = 1\%$, $\overline{PSNR} = 28,21dB$, $var(PSNR) = 0,455dB^2$.



(d) *Boat*, C_3, $p_{pack} = 10\%$, $\overline{PSNR} = 27,11dB$, $var(PSNR) = 3,739dB^2$.

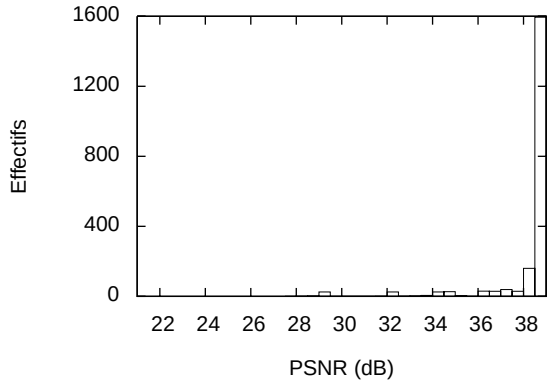


(e) *Peppers*, C_5, $p_{pack} = 1\%$, $\overline{PSNR} = 28,79dB$, $var(PSNR) = 0,600dB^2$.

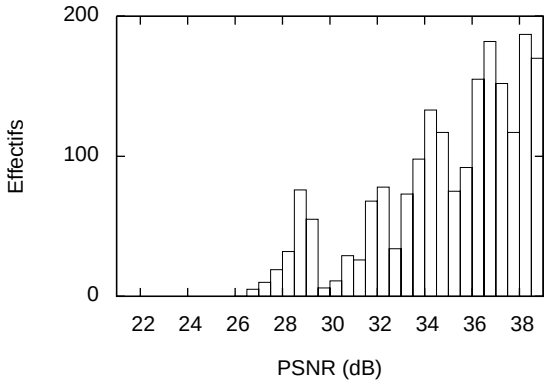


(f) *Peppers*, C_5, $p_{pack} = 10\%$, $\overline{PSNR} = 27,84dB$, $var(PSNR) = 4,455dB^2$.

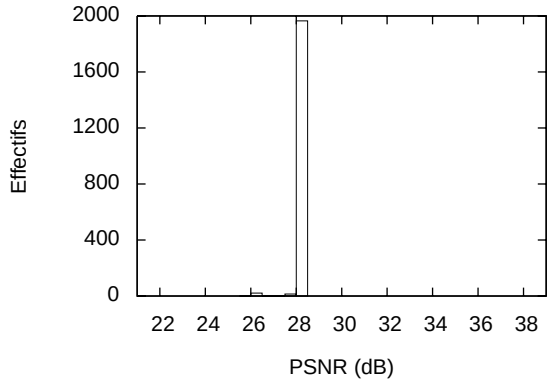
FIG. 4.13: Distribution du PSNR de l'image reconstruite avec la politique de fiabilité F_2.



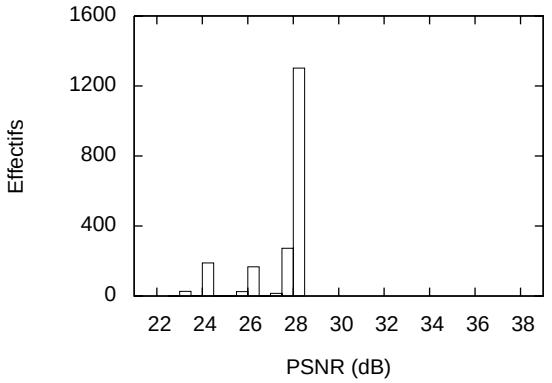
(a) *Lena*, C_1, $p_{pack} = 1\%$, $\overline{PSNR} = 38,12dB$, $var(PSNR) = 2,320dB^2$.



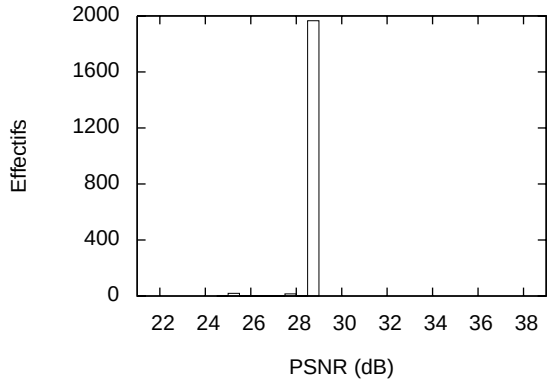
(b) *Lena*, C_1, $p_{pack} = 10\%$, $\overline{PSNR} = 35,01dB$, $var(PSNR) = 9,070dB^2$.



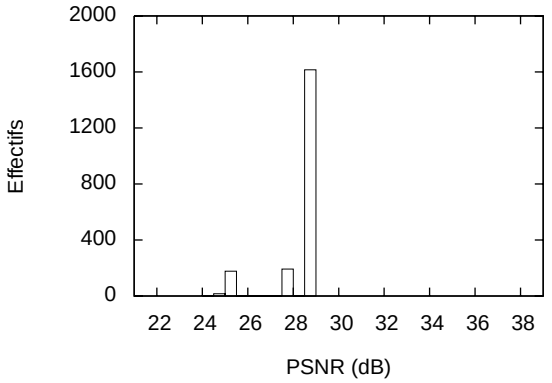
(c) *Boat*, C_3, $p_{pack} = 1\%$, $\overline{PSNR} = 28,26dB$, $var(PSNR) = 0,213dB^2$.



(d) *Boat*, C_3, $p_{pack} = 10\%$, $\overline{PSNR} = 27,61dB$, $var(PSNR) = 1,770dB^2$.

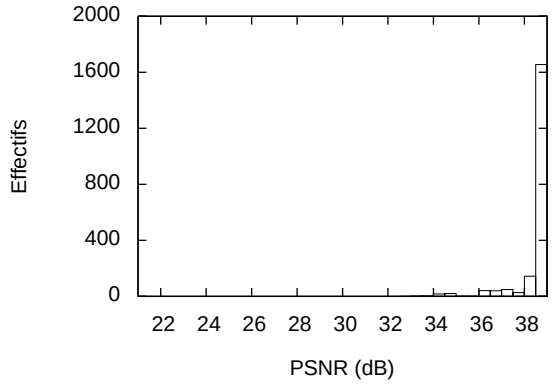


(e) *Peppers*, C_5, $p_{pack} = 1\%$, $\overline{PSNR} = 28,87dB$, $var(PSNR) = 0,142dB^2$.

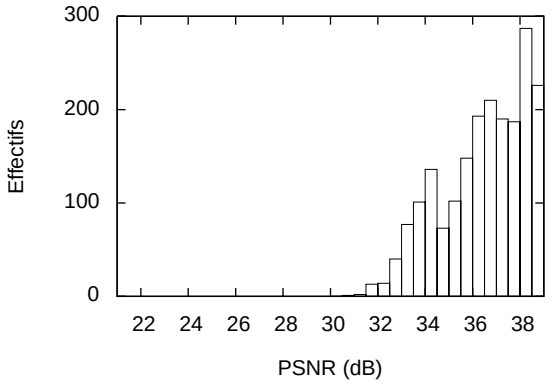


(f) *Peppers*, C_5, $p_{pack} = 10\%$, $\overline{PSNR} = 28,43dB$, $var(PSNR) = 1,303dB^2$.

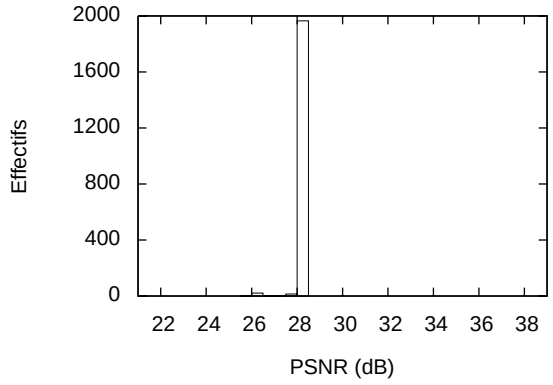
FIG. 4.14: Distribution du PSNR de l'image reconstruite avec la politique de fiabilité F_3.



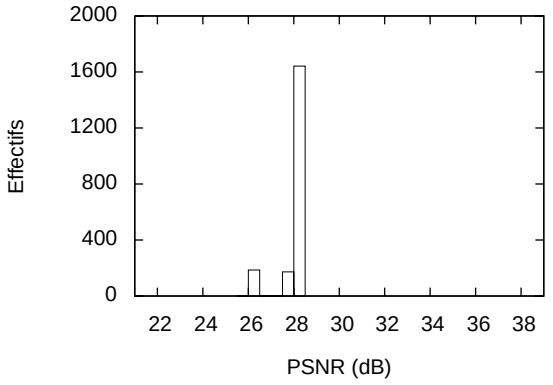
(a) *Lena*, C_1, $p_{pack} = 1\%$, $\overline{PSNR} = 38,36dB$, $var(PSNR) = 0,567dB^2$.



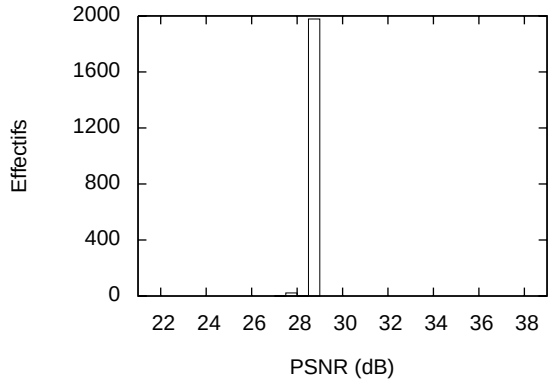
(b) *Lena*, C_1, $p_{pack} = 10\%$, $\overline{PSNR} = 36,44dB$, $var(PSNR) = 3,104dB^2$.



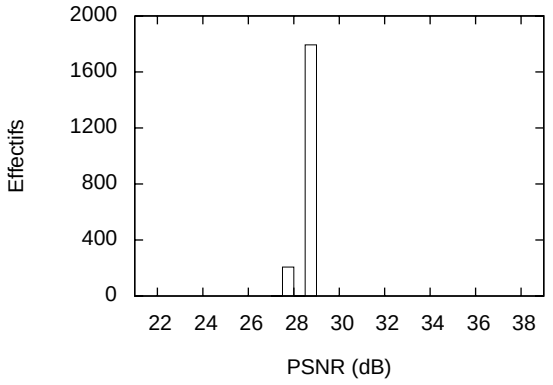
(c) *Boat*, C_3, $p_{pack} = 1\%$, $\overline{PSNR} = 28,31dB$, $var(PSNR) = 0,041dB^2$.



(d) *Boat*, C_3, $p_{pack} = 10\%$, $\overline{PSNR} = 28,11dB$, $var(PSNR) = 0,338dB^2$.

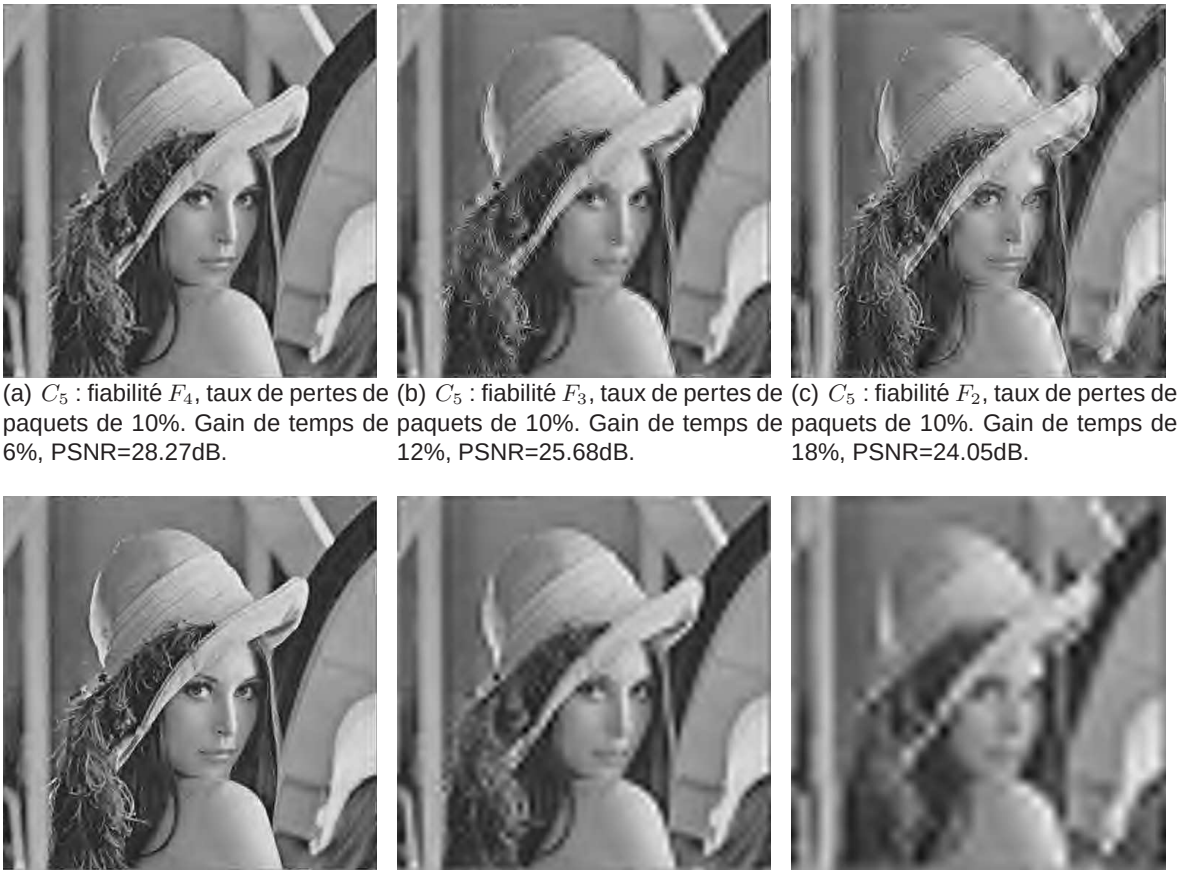


(e) *Peppers*, C_5, $p_{pack} = 1\%$, $\overline{PSNR} = 28,90dB$, $var(PSNR) = 0,018dB^2$.



(f) *Peppers*, C_5, $p_{pack} = 10\%$, $\overline{PSNR} = 28,78dB$, $var(PSNR) = 0,154dB^2$.

FIG. 4.15: Distribution du PSNR de l'image reconstruite avec la politique de fiabilité F_4.



(a) C_5 : fiabilité F_4, taux de pertes de paquets de 10%. Gain de temps de 6%, PSNR=28.27dB.

(b) C_5 : fiabilité F_3, taux de pertes de paquets de 10%. Gain de temps de 12%, PSNR=25.68dB.

(c) C_5 : fiabilité F_2, taux de pertes de paquets de 10%. Gain de temps de 18%, PSNR=24.05dB.

(d) C_5 : aucune perte de paquet (29.47dB).

(e) C_5 : fiabilité F_3, s-paquets tous perdus (25.08dB).

(f) C_5 : fiabilité F_2, s-paquets tous perdus (22.19dB).

FIG. 4.16: Taux de compression élevé C_5 : images *Lena* et tolérance aux pertes de paquets en fonction de la politique de fiabilité.

F_2 et F_3 sont insuffisantes pour garantir une qualité d'image suffisante pour l'application (la contrainte, nous le rappelons, est de fournir une image dont la dégradation de qualité n'attire pas immédiatement l'attention). Le niveau de protection offert par F_4 est par contre compatible avec ces exigences. L'image 4.16(a) montre une dégradation beaucoup moins évidente au premier regard, le PSNR est de 28,27dB, soit une baisse de 1,2dB par rapport à l'image sans perte additionnelle.

Les images 4.16(e) et 4.16(f) sont obtenues lorsque tous les s-paquets sont perdus, il s'agit donc des pires reconstructions possibles du point de vue de la qualité. Ces cas extrêmes sont rares mais les reconstructions dont le PSNR est inférieur à 23dB sont suffisamment fréquentes (200 sur 2000 avec F_2, voir figure 4.13(f)) pour confirmer le choix de la politique F_4, y compris avec les forts taux de compression.

La planche d'images 4.17 présente la meilleure (aucun paquet n'est perdu) et la pire reconstruction possible (tous les s-paquets sont perdus), respectivement sur les lignes 1 et 2 pour l'image *Boat* aux taux de compressions faibles à intermédiaires. La dégradation de qualité est sensible, notamment au niveau des cordages, mais pas autant que la baisse de PSNR ne le laisserait présager. L'image compressée à 1 bit/pixel (C_1) enregistre la baisse la plus importante : 10,5dB entre la meilleure et la plus mauvaise reconstruction. Cela est expliqué par la grande quantité des informations situées dans les deux résolutions les plus grandes. Leur

(a) C_1 : aucune perte de paquet (36.97dB). (b) C_2 : aucune perte de paquet (33.36dB). (c) C_3 : aucune perte de paquet (28.34dB).

(d) C_1 : s-paquets tous perdus (26.50dB). (e) C_2 : s-paquets tous perdus (26.48dB). (f) C_3 : s-paquets tous perdus (26.08dB).

FIG. 4.17: Tolérance de l'image *Boat* face aux pertes de paquets lorsque la politique de fiabilité est F_4.

perte prive alors la reconstruction d'un incrément de qualité important. La qualité visuelle est néanmoins acceptable dans le sens où il est difficile de remarquer la dégradation sans disposer de l'original. Il s'agit cependant des pires reconstructions possibles. Il faut garder à l'esprit que leurs fréquences d'apparition sont très faibles au point que sur les 2000 reconstructions, aucune ne correspond au pire cas, que ce soit pour les images *Boat*, *Lena* ou *Peppers* à 1 bit/pixel et un taux de pertes de paquets de 10%.

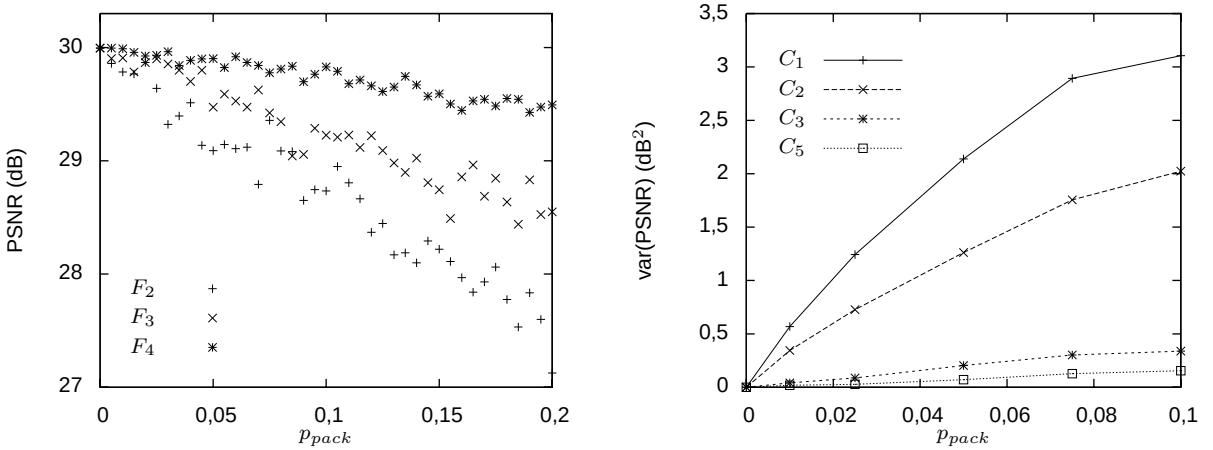
La politique de fiabilité F_4 permet une reconstruction dont la dégradation n'est pas sensible visuellement au premier abord, et ce quel que soit l'agencement des pertes. Avec cette politique de fiabilité, la dégradation est maîtrisée, même lorsque tous les s-paquets sont perdus. La protection des quatre premières résolutions selon la politique F_4 est par conséquent le premier niveau de protection en mesure de satisfaire les contraintes de l'application considérée en toutes circonstances. C'est donc aussi la première politique qui permet une réduction du temps de service, c'est pourquoi nous considérons ce niveau de fiabilité F_4 pour la suite de ce chapitre.

4.4.3 Compromis entre le temps de service et la dégradation de l'image

Ce choix de la politique de fiabilité qui consiste à protéger les quatre résolutions principales a été effectué après examen visuel des images. Dans cette section nous allons mettre en relation la qualité de la reconstruction avec le temps que le transport à fiabilité partielle permet de gagner par rapport à un transport fiable.

4.4.3.1 Dégradation de qualité maîtrisée par la protection des quatre résolutions principales

L'analyse du PSNR tend à confirmer le choix de la politique de fiabilité F_4 selon laquelle les quatre résolutions principales sont protégées. La courbe 4.18(a) présente le PSNR moyen obtenu en fonction du taux de pertes de paquets pour l'image *Peppers* compressée au taux moyen C_3 (46 :1) et en considérant un MSS de 1452 octets. Le niveau de protection F_4 est incontestablement plus performant que F_3 du point de vue du PSNR mesuré quand les pertes augmentent. À 10% de pertes de paquets, la baisse de PSNR moyenne est inférieure à 0,3dB pour une image compressée à un taux de 46 :1, alors que la politique F_3 réduit le PSNR de 1dB dans les mêmes conditions. Ces chiffres sont intéressants en ce qui concerne les taux de compression moyens à élevés mais comme nous l'avons conclu dans la section précédente, les taux de compression faibles subissent une diminution de PSNR plus importante (voir figure 4.18(b)) bien que la qualité visuelle reste satisfaisante.



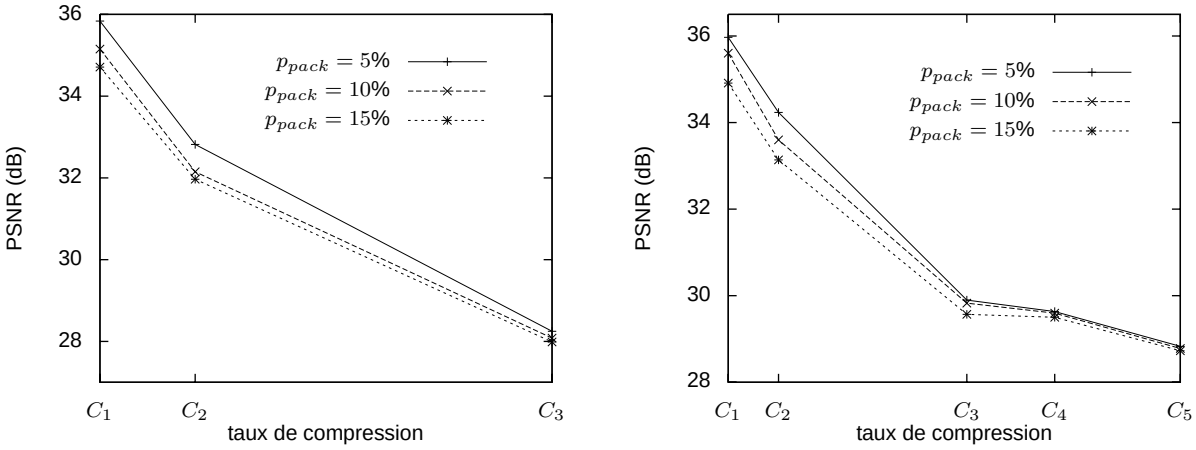
(a) C_3 : DSL, PSNR en fonction du taux de pertes de paquets pour l'image *peppers*.

(b) Variance du PSNR en fonction du taux de pertes de paquets en considérant la protection F_4.

FIG. 4.18: Dégradation de la qualité de l'image en fonction du taux de pertes de paquets.

La réduction de la dégradation du rapport signal/bruit avec l'augmentation du taux de compression est montrée sur la figure 4.19. La courbe 4.19(a) présente les variations du PSNR moyen en fonction des taux de compression et de pertes de paquets pour l'image *Boat* en considérant des paquets de 1452 octets de charge utile (DSL). Les résultats pour l'image *Peppers* sont présentés sur la courbe 4.19(b), sachant que l'image *Lena* produit des résultats qui sont très proches à une composante continue près. Les taux de compression élevés, en particulier C_4 et C_5 ne sont pratiquement plus perturbés par le transport à fiabilité partielle (voir 4.19(b)). Cela vient de la très faible quantité de données des résolutions non protégées. Leur perte est à la fois rare et peu importante pour la qualité de reconstruction. La résolution la

plus grande ne contient d'ailleurs pas de données pour ces taux de compression, les pertes ne peuvent de ce fait concerner la résolution $\mathcal{R}_4$ uniquement.



(a) *Boat* : PSNR *vs.* taux de compression.

(b) *Peppers* : PSNR *vs.* taux de compression.

FIG. 4.19: Influences du taux de compression sur la dégradation de qualité entraînée par la fiabilité partielle F_4.

La diminution de PSNR en fonction des pertes assurée par la protection F_4 est relativement douce par rapport à celle offerte par F_2 ou F_3. Si les images présentées sur les figures 4.16 et 4.17 sont satisfaisantes du point de vue des contraintes de l'application, alors F_4 permet à la fois de satisfaire les besoins de l'application et de gagner du temps par rapport à un service fiable.

4.4.3.2 Gain de temps potentiel

Nous avons quantifié la dégradation de qualité des images qui est associée au transport à fiabilité partielle dans la section précédente. Les paramètres nécessaires à l'analyse du temps de service selon le modèle présenté au chapitre 2 étant connus, nous pouvons évaluer le gain de temps que permet en contrepartie le transport à fiabilité partielle, en particulier par rapport au transport fiable.

La figure 4.20 présente les aspects temporels du service à fiabilité partielle 2CP-ARQ. Une spécification très protectrice de la fiabilité augmente inévitablement le temps nécessaire à la transmission d'un paquet. La durée maximale correspond au service fiable (F_6) pour lequel toutes les résolutions de l'image sont protégées. La courbe 4.20(a) montre le temps de service moyen pour un paquet 2CP-ARQ lors du transport des images compressées au taux C_3 pour une connexion DSL 1Mbits/s en fonction de la politique de fiabilité et du taux de pertes de paquets. Cette courbe montre que la politique de fiabilité F_4 est, en termes de temps, significativement plus efficace que la fiabilité totale F_6. Le temps de service par paquet est en effet de 36ms, 38ms, 41ms et 44ms, respectivement pour F_2, F_3, F_4 et F_6 et un taux de pertes de paquets de 5%. Lorsque le taux de pertes est de 10%, les temps de service pour un paquet augmentent respectivement à 57ms, 61ms, 67ms et 74ms. Rapporté aux images compressées au taux C_3 qui sont constituées de 7 paquets dont la charge maximale est de 1452 octets avec une connexion à 1Mbits/s et la protection F_4, cela correspond à un temps de service moyen par image de 87ms, 287ms et 469ms pour un taux de pertes nul, de 5% et de 10% respectivement.

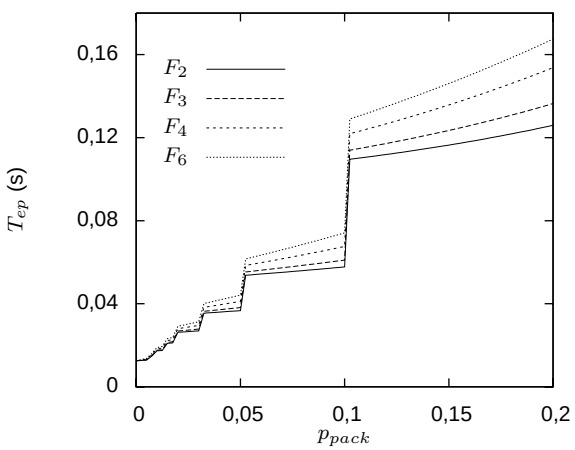
En considérant la fiabilité totale, le temps de service pour une image est identique lorsque les pertes sont nulles, puis il passe à 308ms et 518ms pour un taux de pertes de paquets de 5% et 10% respectivement. Si l'on compare les 287ms/469ms que l'on peut espérer avec F_4 aux 308ms/518ms dans le cas totalement fiable, le gain semble négligeable pour un utilisateur isolé. Il faut néanmoins rapporter ces chiffres aux millions d'images transmises à chaque instant sur l'Internet pour mesurer le temps gagné, et donc les ressources réseau économisées puisque le nombre de paquets transmis ne change pas.

Comparé au service fiable, le gain de temps que l'on peut espérer est présenté sur la courbe 4.20(b) pour les trois politiques de fiabilité envisagées, en fonction du taux de pertes de paquets pour la connexion 1Mbits/s DSL et le taux de compression moyen C_3. Cette courbe confirme le gain, certes plus faible mais cependant significatif, apporté par F_4 par rapport à une protection moindre. Très logiquement, le gain est nul en cas d'absence de perte ou au contraire lorsque tous les paquets sont perdus. Le gain peut atteindre 9,1% pour un taux de pertes de paquets de 30% mais ce qui est intéressant en pratique, ce sont les taux de pertes réalistes, *i.e.* , inférieurs à 20%.

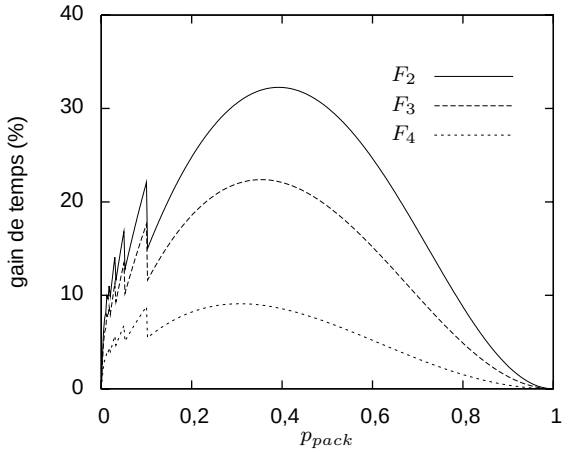
Le gain de temps pour ces taux de pertes est mis en évidence sur les courbes 4.20(c) et 4.20(d), toujours pour un accès 1Mbits/s et pour le niveau de fiabilité F_4 en fonction du taux de pertes de paquets et pour différents taux de compressions. La tendance générale selon laquelle un taux de compression faible autorise un gain plus important qu'une compression forte se confirme. En considérant 5% de pertes de paquets, le gain de temps sur le service fiable s'élève à 18,3% pour le taux de compression C_1, 14,1% pour C_2, 6,7% pour C_3, 3,2% pour C_4 et 4,7% pour C_5. Le gain de temps potentiel supérieur pour le taux de compression C_5 (80 :1) que pour le taux C_4 (60 :1) est une singularité. Cette « anomalie » est due à la manière dont l'information est mise en paquets, notamment parce qu'il y aura toujours au minimum un paquet par résolution non vide. Dans le cas de C_4, la répartition p-/s- est 5/1. Avec la compression C_5, la quantité de données protégées diminue. Cette moindre quantité de données a pour effet de créer un p-paquet de moins pour atteindre la limite basse de quatre p-paquets (4 résolutions). Mais il n'est pas possible de créer un s-paquet de moins car il doit y en avoir au moins un qui correspond à la résolution $\mathcal{R}_4$ qui est la dernière résolution non vide de données. Le rapport du nombre de p-paquets sur le nombre total de paquets augmente alors légèrement, ce qui explique la contre-performance observée à ce niveau de compression.

4.4.3.3 Influence des caractéristiques du réseau sur les performances de 2CP-ARQ

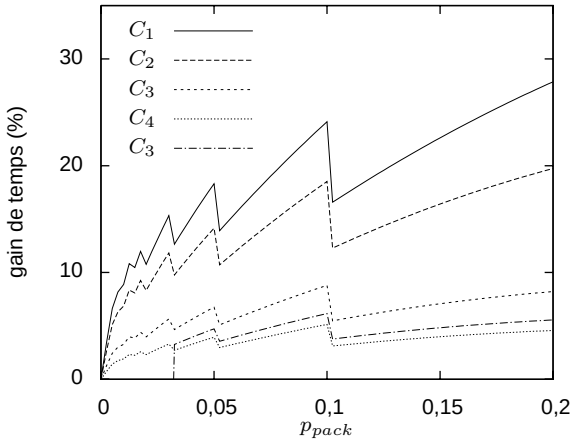
Les différences essentielles entre les deux types de connexion au réseau que nous avons prises en exemple résident dans le débit maximum de la transmission, la différence de taille maximale des paquets et les délais aller et retour. La courbe 4.4.3.3 montre que le gain de temps que permet 2CP-ARQ par rapport au service totalement fiable, est supérieur dans le cas d'une connexion RTC 56kbits que pour la connexion plus performante à 1Mbits/s. Cette différence sur le gain relatif de temps de transmission est expliquée uniquement par la taille maximale des paquets inférieure dans le cas RTC. Des paquets plus petits favorisent le rapport du nombre de p-paquets sur le nombre total de paquets, notamment pour les petites images dont la quantité de données dans les petites résolutions sont faibles. Des paquets de grande taille ne servent à rien dans la mesure où les données dans les petites résolutions sont suffisamment peu nombreuses pour être transportées dans un seul paquet par résolution. Cela a même pour effet d'augmenter le rapport des p-paquets, c'est-à-dire le taux de fiabilisation et par conséquent le temps de transmission. D'où le gain relativement plus important dans le cas des paquets de 536 octets de charge utile comparé au gain dans le cas des paquets de 1452



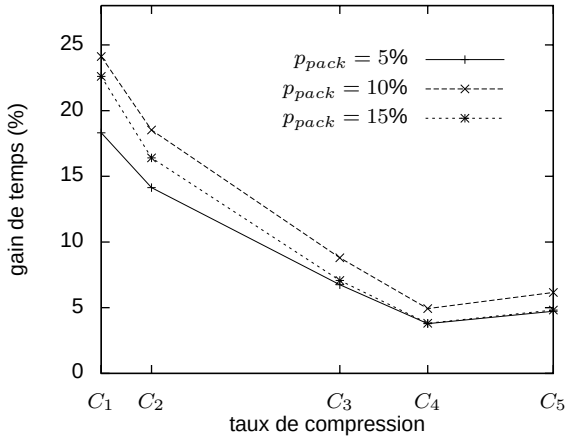
(a) C_3 : DSL, D_{ip} *vs.* taux de pertes de paquets.



(b) C_3 : DSL, gain de temps en comparaison du service totalement fiable.



(c) Gain de temps permis par F_4 par rapport au service fiable.



(d) Gain de temps par rapport au service fiable F_6 en fonction du taux de compression pour l'image *Lena*.

FIG. 4.20: Transport des images partiellement fiable et temps de service.

octets de charge utile. Si les images étaient de plus grande taille et/ou le taux de compression moindre, cette différence de gain tendrait à disparaître et ce gain n'en serait que plus important encore.

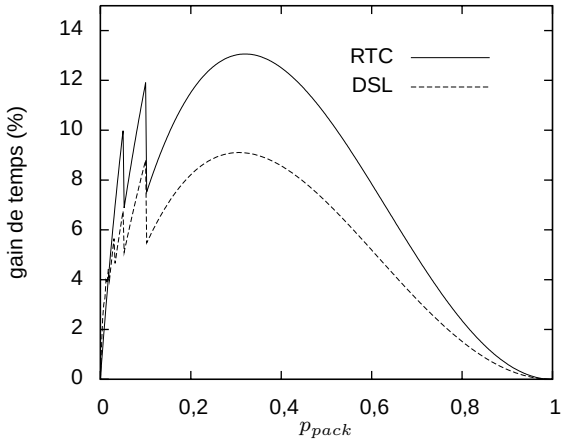


FIG. 4.21: Gain de temps relativement à un service totalement fiable selon le type de connexion réseau. Taux de compression C_3 et fiabilité F_4.

Conclusion

Nous avons proposé dans ce chapitre une approche qui développe conjointement la compression et le transport à fiabilité partielle des images par 2CP-ARQ. Cette approche peut être vue comme une compression en deux étapes : la première consiste à compresser l'image au sens habituel du terme. Elle fournit des données qui peuvent accepter certaines pertes supplémentaires occasionnées par le réseau de communication. Ces pertes additionnelles peuvent être considérées comme un deuxième étage de compression qui est aléatoire puisque le réseau en est la cause, mais pas totalement puisque le premier étage de la compression décide sur quel ensemble de données la compression additionnelle peut avoir lieu.

Nous avons montré que la décomposition multirésolution réalisée par les TOs permettait un ajustement de la fiabilité par la protection progressive des résolutions. Dans le cas des images à 256 niveaux de gris décomposées en six résolutions, la protection des quatre résolutions les plus importantes permet d'assurer une reconstruction satisfaisante pour les taux de compression et les applications courants.

Par rapport à un service fiable comme TCP, la qualité de la reconstruction est inférieure ou égale, mais en contrepartie la durée des transmissions est elle aussi inférieure ou égale. Le transport des images par 2CP-ARQ autorise une réduction du temps de transmission compris entre 3,2% et 18,3% par rapport au service totalement fiable, quand le réseau perd 5% des paquets et pour des taux de compression s'échelonnant de 80 à 8 :1. Ce gain de temps est le résultat d'une réduction du nombre de paquets transmis, il traduit donc une moindre utilisation des ressources réseau.

Bien sûr on peut rétorquer qu'une compression avec un débit de 3 à 18% inférieur permet une transmission dont la durée est identique et consommera autant de ressources réseau tout en garantissant une qualité constante de la reconstruction. Le transport à fiabilité partielle garde cependant l'avantage de permettre une reconstruction de meilleure qualité lorsque les

conditions du réseau sont bonnes tout en s'adaptant lorsque des congestions apparaissent. Finalement c'est bien lorsque le réseau éprouve des difficultés à acheminer le trafic global que l'économie des ressources est la plus bénéfique.

Certains aspects du travail présenté dans ce chapitre peuvent cependant faire l'objet de perspectives. Les pertes additionnelles produites par le réseau sont ici substituées par des vecteurs nuls. De façon à améliorer la qualité de reconstruction, une solution consiste par exemple à réaliser de la dissimulation d'erreur (*error concealment*) au niveau de l'application réceptrice. Les vecteurs peuvent en effet être reconstruits de façon plus vraisemblable, notamment à partir des vecteurs voisins.

Les paquets manquants ne provoquent pas une dégradation localisée dans une zone compacte comme c'est le cas avec la perte des contributions de *code-blocks* JPEG2000. Les paquets sont néanmoins construits à partir de vecteurs adjacents dans l'ordre des lignes dans chaque sous-bande d'une résolution donnée (dans la limite de la taille imposée par le réseau). Cette mise en paquet provoque donc nécessairement la pertes de données contiguës et une dégradation localisée. Si la résolution considérée est petite, il est possible qu'elle soit en totalité transportée par un seul paquet, mais si ce n'est pas le cas il serait alors plus judicieux de construire les paquets à partir de vecteurs non adjacents. Une telle approche favoriserait l'idée précédente qui consiste à dissimuler les erreurs au niveau de l'application réceptrice.

Enfin, même si la transposition de ce travail aux images couleurs paraît assez directe, surtout en considérant l'espace luminance/chrominances qui sont de mise en compression, il conviendrait de traiter ce cas.

Conclusions

Les travaux présentés dans cette thèse se situent à la rencontre des deux domaines des sciences de l'information que sont les réseaux et le traitement d'images. Ils portent sur la transmission des images fixes compressées dans des réseaux à commutation de paquets sujets à des pertes de paquets comme c'est le cas de l'Internet. Une difficulté réside alors dans la non possibilité d'agir directement sur les réseaux, en raison du facteur d'échelle e de l'hétérogénéité des « tiyaux » , puisque les métriques de qualité de service (bande-passante, retards, pertes) sont liées aux caractéristiques intrinsèques des liens et des technologies en présence. Il est donc nécessaire de s'accommoder de la qualité de service offerte par le réseau, c'est-à-dire de la qualité d'un service « best-effort » , et d'agir où on le peut, notamment au niveau de la couche transport. Le contrôle de la QoS est donc considéré de bout en bout.

Nous avons mis à profit la relative tolérance aux pertes des contenus multimédias et des images fixes en particulier, pour relâcher la contrainte de fiabilité lors de la transmission de ce type de données. Nous avons dans ce but proposé un nouveau protocole à fiabilité partielle, appelé 2CP-ARQ, qui comble l'espace existant entre le transport totalement fiable assuré par TCP pour la quasi totalité des flux, et le transport non fiable offert par UDP. Ce protocole distingue deux classes de paquets pour assurer un contrôle d'erreurs différencié.

Le principal objectif de notre proposition consiste à réduire la charge du réseau en autorisant une reconstruction altérée des images transmises. Le niveau d'acceptation concernant cette dégradation de la qualité des images reconstruites dépend uniquement de l'application visée, il doit néanmoins être satisfait par action sur le niveau de fiabilité variable que 2CP-ARQ permet. Ce type de transport réduit le nombre des corrections de pertes par retransmission. Finalement cela réduit le nombre de paquets transmis, ce qui a pour effet de diminuer le temps d'utilisation du réseau pour une transmission donnée, ce qui conduit finalement à une économie des ressources du réseau. Nous avons montré que lorsque le taux de pertes de paquets est raisonnable (inférieur à 10%), 2CP-ARQ peut permettre un gain de temps pouvant atteindre 9% tout en garantissant une qualité minimale de l'image reconstruite compatible avec les applications courantes.

2CP-ARQ autorise la transmission des informations selon deux classes de fiabilité. Les informations doivent donc pouvoir être classées en deux catégories, la première rassemble les données cruciales qui seront transportées de manière fiable, la seconde est constituée des données optionnelles dont l'application peut s'accommoder de la perte. La transmission des images compressées par 2CP-ARQ peut alors être vue comme une seconde compression effectuée par le réseau cette fois. Cette seconde compression n'est certes pas optimisée comme l'est la compression avec pertes définie par les différents standards, mais elle n'est pas totalement aléatoire non plus puisqu'une partie de l'information est protégée, cette protection étant en effet décidée avant la transmission.

Cela suppose que ces informations puissent être regroupées et segmentées dans des paquets indépendants. L'étude du standard JPEG2000 conduit à la recherche d'une méthode

pour trier les informations codées et les classer selon les deux catégories requise ou optionnelle nécessaires à 2CP-ARQ. Notre proposition de tri s'appuie sur les informations contenues dans les en-têtes de paquets JPEG2000, c'est-à-dire que le décodage complet de l'image n'est pas nécessaire pour classer les différentes contributions qui la composent. Cette étape préalable à la mise en paquets permet le transport des images JPEG2000 par 2CP-ARQ. Le prix à payer se situe dans le regroupement en zones de la dégradation. La perte d'information porte en effet nécessairement sur certains *codeblocks*, ce qui peut conduire à une dégradation immédiatement sensible dans les zones carrées correspondantes puisque les zones de l'image qui correspondent aux *codeblocks* reçus en intégralité ne sont pas altérées. En considérant une application peu contraignante comme par exemple la transmission d'objets graphiques qui n'ont pour vocation que d'agrémenter des pages web, alors la dégradation peut passer inaperçue, ce qui n'est pas le cas des images porteuses de sens et qui vont de ce fait accaparer toute l'attention des utilisateurs. À cette difficulté s'en ajoutent deux autres. D'une part les données compressées altérées par les pertes doivent permettre une reconstruction du reste de l'image quelles que soient les pertes et d'autre part, 2CP-ARQ nécessite des unités indépendantes dont la taille est fixée par les caractéristiques du réseau pour fonctionner au mieux. Or il est difficile de concilier ces deux contraintes avec JPEG2000 sans devoir remettre en cause la décomposition en paquets JPEG2000 ou sans nécessiter un nouveau codage avec des paramètres de compression différents de ceux utilisés couramment, ces derniers ne délivrant pas des unités indépendantes dont la taille est très favorable à l'encapsulation directe par 2CP-ARQ, notamment à cause des contraintes de taille des paquets.

Les décompositions multirésolution et en couches de qualité qui caractérisent JPEG2000 sont intéressantes pour différencier les données à protéger de celles qui peuvent être considérées facultatives. Cependant le partitionnement complexe en résolution, couches de qualités, *codeblocks* et *precincts* et l'organisation des données dans le *code-stream* JPEG2000 rendent délicate la formation d'unités de données applicatives propices à une encapsulation directe par 2CP-ARQ. Avec sa proposition de transport interactif adapté à JPEG2000 (JPIP), Taubman lui-même a dû recourir à des paramètres de compression particuliers de façon à produire un *code-stream* caractérisé par un grand nombre de *precincts*. Le transport des images JPEG2000 codées avec les paramètres habituels (précision du débit à atteindre et du type de progression par qualité) qui correspondent à l'immense majorité des images JPEG2000 ne sont donc pas compatibles dès que TCP n'est plus la solution de transport envisagée.

Les standards de compression d'images habituels supposent le stockage ou la transmission totalement fiable. Cela revient à faire abstraction des problèmes inhérents à la transmission puisque TCP s'occupe d'assurer une connexion fiable dont la seule visibilité du point de vue de l'application réside dans l'apparition de retard comparativement à des accès locaux aux données. Dès lors que le transport envisagé n'est plus totalement fiable, deux options sont envisageables : soit le service de transport est étudié spécifiquement pour les images issues d'un standard de compression particulier, c'est l'approche adoptée par ITP par exemple (Raman *et al.*, 2000), soit le schéma de compression est développé avec les propriétés liées à la fiabilité et/ou l'ordre partiels en tête, c'est le concept de GIF-NC par exemple (Amer *et al.*, 1999).

Il est donc difficile de proposer un service de transport à fiabilité partielle indépendant de tout standard de compression, c'est pourquoi nous avons proposé un schéma de compression fondé sur la décomposition multirésolution par transformées en ondelettes discrètes qui permet d'obtenir des unités de données dont la longueur est modulable et peut s'adapter aux besoins de la couche transport. Nous avons montré que pour des débits courants et une décomposition en six résolutions, la protection des quatre résolutions principales permet une reconstruction dont la dégradation de qualité est compatible avec une application non critique telle que la

transmission des objets graphiques décorant les pages web, et ce quelle que soit la propension du réseau à perdre des paquets. Lorsque le taux de pertes de paquets est inférieur au taux déjà très élevé de 10%, la dégradation de PSNR mesurée est typiquement inférieure à 1dB lorsque le taux de compression est faible (8 :1), et plus faible à mesure que le taux de compression s'élève. Visuellement, la dégradation est dans le même temps difficile à percevoir. Par rapport à JPEG2000, notre proposition ne s'appuie plus sur des *codeblocks* indivisibles. Le regroupement en zones de la dégradation est par conséquent moins sensible puisque les données perdues ne correspondent plus à des zones carrées compactes.

Perspectives

Les travaux menés tout au long de cette thèse ainsi que les résultats obtenus permettent de dégager plusieurs perspectives de recherche. Nous avons envisagé le transport des images compressées par 2CP-ARQ comme une alternative au transport d'images par TCP dans le but de contribuer à l'économie des ressources du réseau dans un environnement best-effort. Puisqu'il fournit une connexion fiable, TCP permet à l'application de se concentrer uniquement sur l'aspect compression, avec la recherche des meilleurs performances en termes de rapport signal sur bruit/volume. La qualité de restitution est alors constante mais le temps de transmission, extrêmement variable, est tributaire des ressources disponibles tout au long du chemin. À l'inverse, 2CP-ARQ autorise des taux de compression plus faibles en présence de pertes de paquets pour un temps de transmission équivalent au service fiable, donc une qualité meilleure lorsque les ressources du réseau sont suffisantes. En contrepartie, 2CP-ARQ oblige à intégrer la possibilité de pertes lors de la conception de l'application et de la compression d'images en particulier.

Ces travaux peuvent être enrichis avec l'intégration de techniques dans le but de s'adapter plus finement aux conditions du réseau, notamment avec un taux de fiabilité dynamique tout au long de la transmission. La qualité des images reconstruites peut également faire l'objet d'optimisations. Deux Deux voies devraient y contribuer : une sélection plus fine des données les moins pertinentes du point de vue de la qualité de l'image qui peuvent être perdues d'une part, et une reconstruction adaptative des données manquantes en réception d'autre part. La proposition du système de compression adapté au transport à fiabilité partielle présentée au chapitre 4 doit pouvoir être améliorée, principalement en ce qui concerne la sélection des données à fiabiliser. La sélection présentée se fonde sur la résolution à laquelle appartiennent les vecteurs constituant les données, or tous les vecteurs ne contribuent pas autant à l'amélioration de la qualité de l'image au sein d'une même résolution. Les vecteurs situés sur des contours de l'image (hautes fréquences) sont naturellement plus important que ceux qui sont relatifs à une zone peu détaillée. Il est donc envisageable que la sélection de vecteurs relatifs à des contours situés dans les résolutions les plus grandes au détriment de certains vecteurs qui appartiennent à des résolutions *a priori* cruciales, puisse améliorer la qualité de la reconstruction.

Un deuxième axe de recherche qui permettrait d'améliorer la qualité des images reconstruites concerne la dissimulation d'erreur. Que ce soit avec JPEG2000 ou notre proposition de compression adaptée au transport à fiabilité partielle, lorsque des données sont manquantes, le récepteur crée des données de longueur équivalentes dont le contenu est arbitraire, sans prise en compte du contexte. Des techniques de dissimulation d'erreurs permettraient la création de données de substitution plus vraisemblable en tenant compte des données présentes. Les techniques de dissimulation ne peuvent cependant pas être intégrées au niveau transport car

elles sont extrêmement dépendantes du système de compression, elles doivent donc être définies et mises en œuvre au niveau de l'application de réception.

Enfin, ces travaux peuvent faire l'objet de perspectives avec la mise en œuvre d'applications de transport d'images fixes, mais le domaine d'application pourraient être étendu au streaming vidéo (non temps réel donc), notamment pour des flux continus tels que motion JPEG2000 dont les techniques de codage et de mise en forme des données exhibent des similitudes avec la variante du standard dédiée aux images fixes.

Au chapitre des champs d'application possibles, nous pensons aux transmissions multicast. La problématique de ces transmissions lorsque le transport fiable est envisagé réside dans l'explosion des acquittements et la difficulté qu'ont les différents routeurs qui constituent l'arbre de maintenir la fiabilité avec des destinataires mal servis qui tendent à pénaliser ceux qui bénéficient de meilleures conditions sur leur chemin. Parce qu'il réduit les écarts du temps de transmission par rapport au service fiable quelles que soient les pertes de paquets, 2CP-ARQ peut contribuer à limiter les problèmes de maintien de la fiabilité dans le cadre des transmissions d'images multicast.

2CP-ARQ aide à réduire la consommation des ressources du réseau, à plus forte raison lorsque celles-ci font défaut. Or la réduction du nombre de paquets émis peut aussi contribuer à la sauvegarde des ressources des hôtes, notamment au niveau de l'énergie consommée. Cette problématique de l'énergie est très préoccupante pour certains réseaux particuliers, notamment les réseaux de capteurs (WSN : Wireless Sensor Networks). La transmission des images sur les réseaux de capteurs fait actuellement l'objet de recherches, y compris au CRAN, en particulier avec la thèse de Cristian Duran-Faundez qui vise à transmettre des images sur de tels réseaux dans l'optique de réduire la consommation d'énergie car la source d'énergie d'un capteur est synonyme de sa durée de vie.

2CP-ARQ adapte sa fiabilité en fonction des ressources que le réseau met à disposition pour une transmission donnée. Cette adaptation est dépendante du taux de pertes de paquets, indicateur mesuré indirectement par le jeu des acquittements positifs, ces derniers permettant aussi d'évaluer les retards aller-retour. Une possibilité d'évolution consiste à mettre en œuvre des mesures actives ou non des caractéristiques de la qualité de service offerte par le réseau, particulièrement la bande-passante. La métrologie réseau et la problématique de la mesure de la bande passante constituent le sujet de thèse d'Ahmed Ait-Ali, toujours au sein de l'équipe SCR au CRAN. L'intégration des outils de mesure proposés dans sa thèse sera très utile, pour 2CP-ARQ mais aussi pour tout protocole de transport adaptatif.

* * *

Annexe

A Détails de l'expression des probabilités liées au modèle de 2CP-ARQ

Cette annexe apporte un complément d'informations visant à clarifier les différentes formules du modèle de 2CP-ARQ présentées au chapitre 2. Nous rappelons que ce modèle se décompose selon les quatre étapes suivantes :

1. Calcul des probabilités d'occurrence conditionnelles des séries sachant la condition NPL (aucun p-paquet n'a été perdu lors de la série précédente). Des détails sur ce point sont donnés au paragraphe A.1 de la présente annexe.
2. Calcul des probabilités d'occurrence conditionnelles sachant la condition PL (au moins un p-paquet a été perdu lors de la série précédente). Ce point est traité au paragraphe A.2.
3. Calcul de la distribution stationnaire de la chaîne de Markov à 2 états NPL et PL (équivalente à un modèle de Gilbert).
4. La connaissance des probabilités d'occurrence de ces deux états permet d'exprimer les probabilités d'occurrence des séries (avec la mise en relation du temps T consommé ainsi que du nombre $pack$ de paquets efficacement transmis) de manière inconditionnelle. Le paragraphe A.3 apporte des explications visant à justifier ces deux derniers points.

A.1 Expression des probabilités conditionnelles sachant que le premier paquet peut être un s-paquet ou un p-paquet

Si aucun p-paquet n'a été perdu lors de la série précédente alors il n'y a aucune raison que le premier paquet de la série courante soit un p-paquet plutôt qu'un s-paquet. La probabilité que ce premier paquet soit un p-paquet est la même que celle que tout autre paquet de la série soit un p-paquet, c'est-à-dire τ_F. Cette condition est appelée NPL (No P-Loss). Selon le temps T consommé par série, nous avons montré que quatre grandes familles d'évènements contiennent tous les évènements (*i.e.* , les séries possibles) qui vérifient la condition NPL.

Nous montrons ici la démarche permettant d'exprimer la probabilité d'occurrence du premier groupe, l'expression des probabilités d'occurrence des autres groupes sachant que la condition NPL est vraie suit le même principe. Le groupe 7 est traité au paragraphe A.2 sous la condition PL, il aide cependant la compréhension des groupes 2 et 4 qui sont aussi caractérisés par une variable k entière.

Groupe i : $T = t_{AR}|NPL$

Pour atteindre le temps minimum $T = t_{AR}$, 2 conditions sont nécessaires.

– Le premier paquet doit être correctement acquitté quelle que soit sa classe, cet ACK parvient à l'émetteur après $T = t_{AR}$.
– Tous les acquittements relatifs aux éventuels k p-paquets parmi les $N-1$ restants sont correctement retournés à l'émetteur, *i.e.* , sans *timeout*.

Soit S_n une série composée de N paquets. Lorsqu'aucun p-paquet n'est perdu, la série S_{n+1} débute par un nouveau paquet. Le nombre $pack_1$ de paquets efficacement transmis est alors évidemment N.

$$pack_1 = pack[T = t_{AR}|NPL] = N$$

La probabilité conditionnelle d'occurrence P_1 donnée par l'équation (2.4) du chapitre 2 est rappelée ci-dessous avec des explications notées E1, E2 et E3 facilitant la compréhension.

E1 Quelle que soit sa classe, le premier paquet est acquitté avec une probabilité de succès notée P_{succ} (ni le paquet, ni son acquittement ne sont perdus, $P_{succ} = (1-p_{pack})(1-p_{ack})$).

E2 Parmi les $(N-1)$ paquets suivants, il peut exister un nombre entier k de p-paquets avec $0 \leqslant k \leqslant (N-1)$. La probabilité que ces paquets soient des p-paquets est τ_F. Les combinaisons de k p-paquets parmi $(N-1)$ sont au nombre de $\binom{N-1}{k}$ et quel que soit le nombre k de p-paquets, tous doivent être acquittés.

E3 Les $(N-k-1)$ paquets restants sont des s-paquets, peu importe le succès ou non de leur transmission.

$$\begin{aligned} P_1 &= P[T = t_{AR}|NPL] \\ &= \underbrace{P_{succ}}_{\texttt{E1}} \sum_{k=0}^{N-1} \left[\underbrace{\tau_F^k \binom{N-1}{k} P_{succ}^k}_{\texttt{E2}} \underbrace{(1-\tau_F)^{N-1-k}}_{\texttt{E3}} \right] \end{aligned}$$

A.2 Expression des probabilités conditionnelles sachant que le premier paquet ne peut être qu'un p-paquet

Lorsqu'un p-paquet a été perdu lors de la série précédente, la série courante commencera forcément par ce p-paquet. Ce premier paquet est donc un p-paquet avec une probabilité de 1 au lieu de τ_F. Les $(N-1)$ paquets suivants de la série ont cependant de nouveau la probabilité τ_F d'être des p-paquets.

Nous rappelons que le groupe 7 comprend les séries dont au moins un p-paquet est perdu à la condition NPL, d'autre part ces pertes ne doivent pas concerner le premier paquet (qui est forcément un p-paquet), ce cas étant déjà intégré par le groupe 6.

Ce groupe 7 est choisi comme exemple dans cette annexe parce qu'il suppose la condition PL et intègre donc la nuance du premier paquet par rapport à la condition NPL. Ce groupe 7 est aussi caractérisé par la variable entière k qui détermine le temps T de la série, caractéristique également partagée par le groupe 4 sachant la condition NPL. Cela signifie que pour chaque rang de première perte d'un p-paquet (pour chaque valeur de k), la probabilité $P_i(k)$ doit être évaluée.

Groupe vii : $T = (k-1)t_{pack} + t_{out}|PL$

Les séries qui composent ce groupe contiennent un ou plusieurs p-paquets perdus (hormis le premier paquet). Or, quel que soit le nombre de p-paquets perdus, seul le premier p-paquet perdu est déterminant puisque la série suivante commencera par lui.

La première perte de p-paquet sera détectée après expiration du *timeout* déclenché lors de son émission. Comme le premier p-paquet perdu est au rang k, avec $2 \leqslant k \leqslant N$, cette perte implique un retard de $(k-1)t_{pack}$ en plus du *timeout*. La série suivante débutera par l'envoi de ce p-paquet perdu donc $(k-1)$ paquets ont été transmis efficacement et ne seront jamais émis à nouveau. La probabilité conditionnelle d'apparition de cette configuration est donnée par (2.16) au chapitre 2, elle est rappelée ci-dessous munie des explications nécessaires.

E4 Le premier paquet doit être acquitté, il s'agit d'une condition d'appartenance à ce groupe. Il est acquitté avec la probabilité P_{succ} (sa probabilité d'être un p-paquet est 1).

E5 Le paquet au rang k est LE premier p-paquet en échec. Il s'agit d'un p-paquet avec une probabilité τ_F et la probabilité de sa perte (paquet ou acquittement) est $(1 - P_{succ})$.

E6 Si la première perte de p-paquet a lieu au rang k alors il faut que tous les p-paquets précédents soient acquittés. Le premier paquet étant par définition acquitté, il y a $(k-2)$ paquets précédant la perte du p-paquet au rang k parmi lesquels il y a potentiellement des p-paquets qui le cas échéant doivent être acquittés. Il y a donc potentiellement j p-paquets précédant la perte du p-paquet au rang k tels $0 \leqslant j \leqslant (k-2)$. Les combinaisons de j p-paquets parmi les $(k-2)$ sont au nombre de $\binom{k-2}{j}$, ces j paquets doivent être des p-paquets et acquittés.

E7 Les $(k-2-j)$ paquets précédant la perte au rang k sont donc des s-paquets. Notons que les paquets qui suivent la perte au rang k importent peu, quelle que soit leur configuration cela ne change pas la valeur de T ou de $pack$. L'ensemble des évènements possibles pour les paquets suivants a par définition une probabilité unitaire d'occurrence et n'apparaît pas dans la formule.

$$
\begin{aligned}
pack_7(k) &= pack[T = (k-1)t_{pack} + t_{out}|PL] = k-1 \\
P_7(k) &= P[T = (k-1) \times t_{pack} + t_{out}|PL] \\
&= \underbrace{P_{succ}}_{\text{E4}} \times \underbrace{\tau_F \times (1 - P_{succ})}_{\text{E5}} \times \sum_{j=0}^{k-2} \left[\underbrace{\binom{k-2}{j} \times \tau_F^j \times P_{succ}^j}_{\text{E6}} \times \underbrace{(1-\tau_F)^{k-2-j}}_{\text{E7}} \right]
\end{aligned}
$$

A.3 Transitions de la chaîne à deux états NPL et PL

Les sept groupes présentés au paragraphe précédent ainsi qu'au chapitre 2 sont disjoints, c'est pour cette raison qu'ils ont été choisis de cette manière. L'union des groupes 1 à 4 forment une partition de l'univers des évènements sachant que la condition NPL est vraie. La somme des probabilités des évènements constituant chacun de ces groupes est donc unitaire. De manière analogue, les groupes 5 à 7 sont disjoints et leur union forme une partition de l'univers des évènements sachant la condition PL. Ces partitions sont illustrées par la figure 22. Cette figure montre également la composition détaillée du modèle à deux états NPL et PL déjà

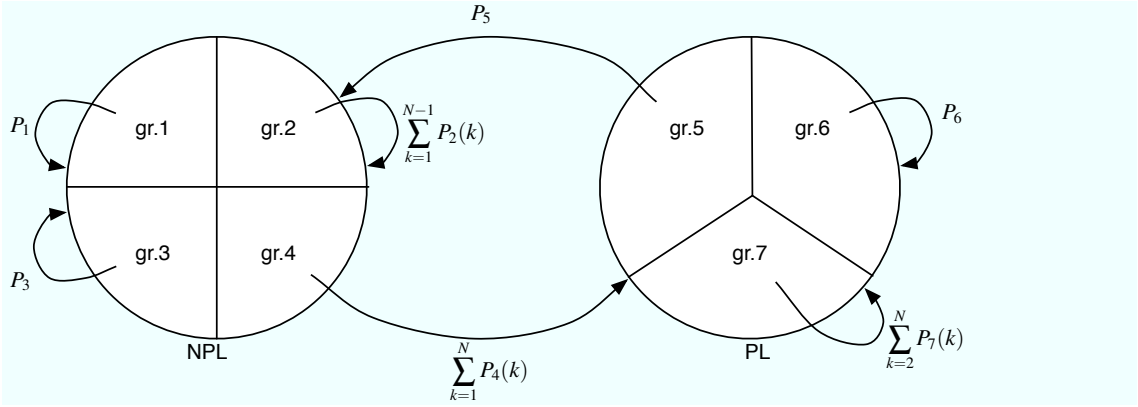


FIG. 22: Transitions détaillées du modèle à deux états NPL et PL.

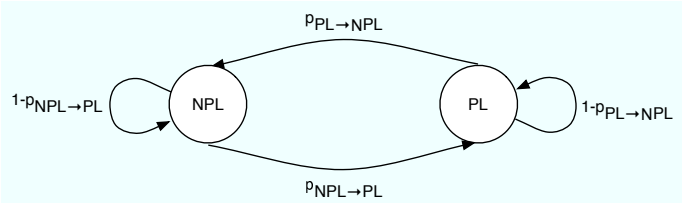


FIG. 23: Transitions $NPL \rightleftharpoons PL$ du modèle à deux états.

présenté au chapitre 2, plus précisément par la figure 2.10 qui est reproduite dans cette annexe en figure 23.

Les notations utilisées sont les suivantes :

– $p_{NPL\to PL}$ est la probabilité de transition de l'état NPL à l'état PL,
– $p_{PL\to NPL}$ est la probabilité de transition de l'état PL à l'état NPL,
– $p_{NPL\circlearrowleft} = 1 - p_{NPL\to PL}$ est la probabilité de rester dans l'état NPL,
– $p_{PL\circlearrowleft} = 1 - p_{PL\to NPL}$ est la probabilité de rester dans l'état PL.

L'expression de ces probabilités peut s'effectuer par une lecture de la figure 22, ce qui conduit aux résultats suivants illustrés par la figure 24.

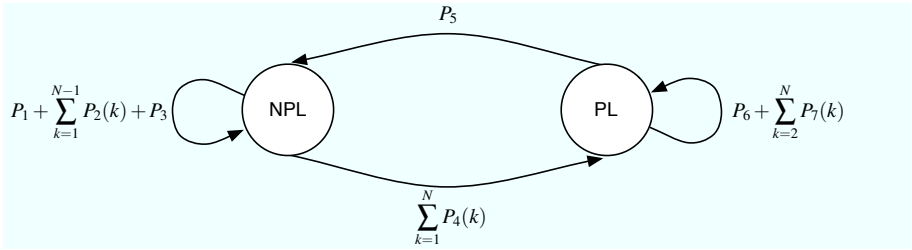


FIG. 24: Expression des probabilités de transitions $NPL \rightleftharpoons PL$ du modèle à deux états.

La matrice de transition P du modèle NPL-PL à deux états est donc la suivante :

$$\mathbf{P} = \begin{pmatrix} 1 - p_{NPL\to PL} & p_{NPL\to PL} \\ p_{PL\to NPL} & 1 - p_{PL\to NPL} \end{pmatrix}$$

Cette chaîne de Markov à deux états est irréductible puisque chaque état peut être atteint

à partir de l'autre, elle est de plus récurrente puisque les deux états vont être traversés un nombre infini de fois lorsque le temps tend vers l'infini. Il existe alors une distribution stationnaire, la probabilité d'être dans chacun des états indépendamment de l'état de départ peut être calculée.

Soient π_{NPL} et π_{PL} les probabilités stationnaires d'être respectivement dans l'état NPL et l'état PL. Le vecteur ligne π des probabilités stationnaires peut alors s'écrire :

$$\pi = (\pi_{NPL}, \pi_{PL})$$

Ce vecteur π de la distribution stationnaire doit par définition vérifier l'équation 2.

$$\pi \mathbf{P} = \pi \tag{2}$$

Le résultat est bien connu pour les chaînes ergodiques à deux états comme celle qui est étudiée ici. Sous réserve que $p_{NPL\to PL} + p_{PL\to NPL} > 0$, or cette condition est vraie pour notre modèle, ce résultat est donné par la formule 3.

$$(\pi_{NPL}, \pi_{PL}) = \left(\frac{p_{PL\to NPL}}{p_{NPL\to PL} + p_{PL\to NPL}}, \frac{p_{NPL\to PL}}{p_{NPL\to PL} + p_{PL\to NPL}} \right) \tag{3}$$

Ce résultat peut aussi s'écrire sous la forme des équations 2.21 et 2.22 du chapitre 2 que nous rappelons ci-dessous :

$$\begin{aligned} \pi_{NPL} &= \frac{1}{1 + \frac{p_{NPL\to PL}}{p_{PL\to NPL}}} \\ \pi_{PL} &= \frac{1}{1 + \frac{p_{PL\to NPL}}{p_{NPL\to PL}}} \end{aligned}$$

La preuve du résultat donné par la formule (3) est rapide :

Preuve

À partir de la définition donnée par (2), nous pouvons écrire :

$$\begin{pmatrix} \pi_{NPL} \\ \pi_{PL} \end{pmatrix} = \begin{pmatrix} \pi_{NPL}(1 - p_{NPL\to PL}) + \pi_{PL} \times p_{PL\to NPL} \\ \pi_{NPL} \times p_{NPL\to PL} + \pi_{PL}(1 - p_{PL\to NPL}) \end{pmatrix} \tag{4}$$

Comme la somme des probabilités stationnaires d'être dans l'un ou l'autre des états est unitaire, on peut écrire :

$$\pi_{NPL} + \pi_{PL} = 1$$

En substituant π_{PL} par $1 - \pi_{NPL}$ dans la première composante du membre de droite de l'équation 4 et π_{NPL} par $1 - \pi_{PL}$ dans la seconde composante, il vient :

$$\begin{aligned} \begin{pmatrix} \pi_{NPL} \\ \pi_{PL} \end{pmatrix} &= \begin{pmatrix} \pi_{NPL}(1 - p_{NPL\to PL}) + (1 - \pi_{NPL}) p_{PL\to NPL} \\ (1 - \pi_{PL}) p_{NPL\to PL} + \pi_{PL}(1 - p_{PL\to NPL}) \end{pmatrix} \\ &= \begin{pmatrix} \pi_{NPL}(1 - p_{NPL\to PL} - p_{PL\to NPL}) + p_{PL\to NPL} \\ \pi_{PL}(1 - p_{PL\to NPL} - p_{NPL\to PL}) + p_{NPL\to PL} \end{pmatrix} \\ \begin{pmatrix} \pi_{NPL}(p_{NPL\to PL} + p_{PL\to NPL}) \\ \pi_{PL}(p_{NPL\to PL} + p_{PL\to NPL}) \end{pmatrix} &= \begin{pmatrix} p_{PL\to NPL} \\ p_{NPL\to PL} \end{pmatrix} \\ \begin{pmatrix} \pi_{NPL} \\ \pi_{PL} \end{pmatrix} &= \begin{pmatrix} \frac{p_{PL\to NPL}}{p_{NPL\to PL} + p_{PL\to NPL}} \\ \frac{p_{NPL\to PL}}{p_{NPL\to PL} + p_{PL\to NPL}} \end{pmatrix} \end{aligned}$$

L'équation (3) est donc vérifiée. Il est à présent possible d'écrire les probabilités stationnaires en fonction des probabilités d'occurrence P_1 à P_7 définies au chapitre 2.

$$\begin{aligned} \pi_{NPL} &= \frac{P_5}{P_5 + \sum_{k=1}^{N} P_4(k)} \\ \pi_{PL} &= \frac{\sum_{k=1}^{N} P_4(k)}{P_5 + \sum_{k=1}^{N} P_4(k)} \end{aligned}$$

Les évènements NPL et PL sont disjoints et forment une partition de l'univers des cas possibles (leur intersection est nulle et la somme de leurs probabilités est unitaire). En vertu du principe des probabilités totales, lorsque B_i est un système complet d'évènements, la probabilité $P(A)$ de l'évènement A peut s'écrire :

$$P(A) = \sum_i P(A \bigcap B_i) = \sum_i P(B_i) P(A|B_i)$$

Appliqué à notre modèle, ce principe nous permet d'exprimer les probabilités sans supposer de condition. $P(G_n)$ étant la probabilité d'occurrence du groupe G_n, ce principe s'énonce de la manière suivante :

$$\begin{aligned} P(G_n) &= P(G_n \bigcap NPL) + P(G_n \bigcap PL) \\ &= \pi_{NPL} \times P(G_n|NPL) + \pi_{PL} \times P(G_n|PL) \end{aligned} \quad (5)$$

Le calcul des probabilités des groupes 8 à 12 présentés à la section 2.3.3 du chapitre 2 est une application directe du résultat (5).

Bibliographie

Akyildiz, Ian F., Giacomo Morabito et Sergio Palazzo (2001). Tcp-peach : a new congestion control scheme for satellite ip networks.. *IEEE/ACM Transactions on Networking*, **9**(3), 307–321.

Allman, M., V. Paxson et W. Stevens (1999). Tcp congestion control. *RFC 2581*.

Amer, Paul D., Christophe Chassot, Thomas J. Connolly et Michel Diaz (1993). Partial order quality of service to support multimedia connections : Unreliable channels. In : *International Networking Conference (INET'93)*. San-Francisco (USA).

Amer, Paul D., Christophe Chassot, Thomas J. Connolly, Michel Diaz et Phillip Conrad (1994). Partial-order transport service for multimedia and other applications. *IEEE/ACM Transactions on Networking (TON)*, **2**(5), 440–456.

Amer, Paul D., Sami Iren, Gul E. Sezen, Phillip T. Conrad, Mason Taube et Armando Caro (1999). Network-conscious GIF image transmission over the Internet. *Computer Networks, Special issue on high-performance protocol architectures*, **31**(7), 693–708.

Andrews, Michael J. (1997). Xudp : A real-time multimedia networking protocol. Master's thesis. Faculty of the Worcester Polytechnic Institute.

Antonini, Marc, Michel Barlaud, Pierre Mathieu et Ingrid Daubechies (1992). Image coding using wavelet transform. *IEEE Transactions on Image Processing*, **1**(2), 205–220.

Barlaud, Michel, Patrick Sol'e, Thierry Gaidon, Marc Antonini et Pierre Mathieu (1994). Pyramidal lattice vector quantization for multiscale image coding. *IEEE Transactions on Image Processing*, **3**(4), 367–381.

Blake, S., D. Black, M. Carlson, E. Davies, Z. Wang et W. Weiss (1998). An architecture for differentiated services. *RFC 2475*.

Bovik, Al (2000). *Handbook of Image and Video processing*. Academic Press.

Braden, R., D. Clark et S. Shenker (1994). Integrated services in the internet architecture : an overview. *RFC 1633*.

Braden, R., L. Zhang, S. Berson, S. Herzog et S. Jamin (1997). Resource reservation protocol (rsvp). *RFC 2205*.

Brakmo, L. et L. Peterson (1995). Cp vegas : End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communication*, **13**(8), 1465–1480.

Brakmo, L., S. O'Malley et L. Peterson (1994). Tcp vegas : New techniques for congestion detection and avoidance.

Cen, Shanwei, Calton Pu et Jonathan Walpole (1998). Flow and congestion control for internet media streaming applications. In : *Multimedia Computing Networking (MMCN98)*.

Chen, Zhigang, See-Mong Tan, Roy H. Campbell et Yongcheng Li (1995). Real time video and audio in the world wide web. In : *Fourth International World Wide Web Conference (WWW)*. Massachusetts, USA.

Clark, David D. et David L. Tennenhouse (1990). Architectural considerations for a new generation of protocols. In : *ACM symposium on Communications architectures & protocols*. pp. 200–208.

Connolly, T., P. Amer et P. Conrad (1994). An extension to tcp : Partial order service. *RFC 1693*.

Conrad, P. T., E. Golden, P.D. Amer et R. Marasli (1996). A multimedia document retrieval system using partially-ordered/partially reliable transport service. In : *Multimedia Computing and Networking 1996*. San Jose, CA, USA.

Conway, J. H. et N. J. A. Sloane (1988). *Sphere packing, lattices and group*. Springer Verlag.

Danskin, John M., Geoffrey M. Davis et Xiyong Song (1995). Fast lossy internet image transmission. In : *Proceedings of ACM Multimedia*.

Deering, S. et R. Hinden (1998). Internet protocol, version 6 (ipv6). *RFC 2460*.

Dempsey, B., T. Strayer et A. Weaver (1992). Adaptive error control for multimedia data transfer. In : *IWACA 92*. Munich, allemagne. pp. 279–289.

Dempsey, Bert J., Jörg Liebeherr et Alfred C. Weaver (1993). A new error control scheme for packetized voice over high-speed local area networks. In : *18th Conference on Local Computer Networks (LCN)*. Minneapolis, USA. pp. 91–100.

Diaz, M., K. Drira, A. Lozes et C. Chassot (1995). On the definition and representation of the quality of service for multimedia systems. In : *6th IFIP international conference on high performance networking (HPN'95)* (Chapman & Hall Ed. R.Puigjaner, Ed.). pp. 116–128.

EPFL (2002). Jj2000 an implementation of the jpeg2000 standard in java http ://jj2000.epfl.ch/.

Feamster, Nicholas G. (2001). Adaptive delivery of real-time streaming video. Master's thesis. Massachusetts Institute of Technology.

Fielding, R., J. Gettys, J. Mogul, H. Frystyk et T. Berners-Lee (1997). Profile for datagram congestion control protocol (dccp) congestion control id 3 : Tcp-friendly rate control (tfrc). *RFC 2068*.

Fisher, Thomas R. (1986). A pyramid vector quantizer. *IEEE Transactions on Information Theory*, **32**(4), 568–583.

Floyd, Sally, E. Kohler et J. Padhye (2006). Profile for datagram congestion control protocol (dccp) congestion control id 3 : Tcp-friendly rate control (tfrc). *RFC 4342*.

Floyd, Sally et E. Kohler (2006). Profile for datagram congestion control protocol (dccp) congestion control id 2 : Tcp-like congestion control. *RFC 4341*.

Floyd, Sally et Kevin Fall (1999). Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, **7**(4), 458–472.

Floyd, Sally, Mark Handley, Jitendra Padhye et Jorg Widmer (2000). Equation-based congestion control for unicast applications. In : *SIGCOMM 2000*. Stockholm, Sweden. pp. 43–56.

Futemma, Satoshi, Eisaburo Itakura et Andrew Leung (2005). Rtp payload format for jpeg 2000 video streams, draft-ietf-avt-rtp-jpeg2000-08. Technical report. Sony Corporation, IETF Internet Draft.

Gersho, Allen et Robert M. Gray (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Gilbert, Edgar N. (1960). Capacity of a burst-noise channel. *Bell systems technical journal*, **39**(5), 1253–1265.

Gong, F. et G. Parulkar (1992). An application-oriented error control scheme for high-speed networks. Technical Report WUCS-92-37. Washington University.

Gormish, Michael et Serene Banerjee (2003). Tile-based transport of jpeg2000 images. In : *VLVB03, Madrid, Spain*.

Gribble, Steven D. et Eric A. Brewer (1997). System design issues for Internet middleware services : Deductions from a large client trace. In : *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*. Monterey, CA.

Grieco, Luigi A. et Saverio Mascolo (2004). Performance evaluation and comparison of westwood+, new reno, and vegas tcp congestion control. *ACM SIGCOMM Computer Communication Review*, **34**(2), 25–38.

Grinnemo, K.-J. et A. Brunstrom (2001). Enhancing TCP for applications with soft real-time constraints. In : *Proc. SPIE Vol. 4518, p. 18-31, Multimedia Systems and Applications IV, Andrew G. Tescher ; Bhaskaran Vasudev ; V. Michael Bove ; Eds.* (A. G. Tescher, B. Vasudev et V. M. Bove, Eds.). pp. 18–31.

Grinnemo, Karl-Johan, Johan Garcia et Anna Brunstrom (2004). Taxonomy and survey of retransmission-based partially reliable transport protocols.. *Computer Communications*, **27**(15), 1441–1452.

Guillemot, Ludovic (2004). Une approche vectorielle pour exploiter le contenu de l'image en compression et tatouage. PhD thesis. Université Henri Poincaré (Nancy Université).

Handley, M., S. Floyd, J. Padhye et J. Widmer (2003). Tcp friendly rate control (tfrc) : Protocol specification. *RFC 3448*.

Hess, C. (1998). Media streaming protocol : An adaptive protocol for the delivery of audio and video over the internet. Master's thesis. College of Computer Science at the University of Illinois at Urbana-Champaign.

Holl, Thomas, Vincent Lecuire et Jean-Marie Moureaux (2004). évaluation de performances d'un transport à fiabilité partielle d'images fixes compressées sur l'internet. In : *Journées Doctorales en Informatique et Réseaux (JDIR)*. pp. 229–239.

Holl, Thomas, Vincent Lecuire et Jean-Marie Moureaux (2005). Transport à fiabilité partielle d'images compressées sur internet. In : *CFIP'05*. pp. 201–216.

Holl, Thomas, Vincent Lecuire et Jean-Marie Moureaux (2006). Error sensitivity of jpeg2000 codestream for efficient data protection on unreliable network. In : *EUSIPCO'06, European Signal Processing Conference*. Florence, Italie.

Iren, Sami et Paul Amer (2000). SPIHT-NC : Network-conscious zerotree encoding. *Proc. Data Compression Conference (DCC '2000), Snowbird, USA*, pp. 313–322.

Jacobson, Van (1988). Congestion avoidance and control. In : *ACM SIGCOMM '88*. Stanford, CA. pp. 314–329.

Jacobson, Van (1990). Modified tcp congestion avoidance algorithm. *end2end-interest mailing list*.

Jacobson, Van et R. Braden (1988). Tcp extensions for long-delay paths. *RFC 1072*.

Jia, Ning, Emmanuel Hyon et Ye-Qiong Song (2005). Ordonnancement sous contraintes (m, k)-firm et combinatoire de mots. In : *RTS embedded systems, 13e édition*. Paris, France. pp. 345–367.

Jiang, Wenyu et Henning Schulzrinne (2000). Modeling of packet loss and delay and their effect on real time multimedia service quality. In : *10th International workshop on Network and Operating Systems Support for Digital Audio and Video(NOSSDAV'2000)*. Chapel Hill, North Carolina, USA.

JTC1/SC29 (2003*a*). Technologies de l'information – système de codage d'image jpeg 2000 – partie 6 : Format de fichier d'image de composant. In : *ISO/IEC 15444-1 :2004*.

JTC1/SC29 (2003*b*). Technologies de l'information – système de codage d'image jpeg 2000 : Logiciel de référence. In : *ISO/IEC 15444-1 :2004*.

JTC1/SC29 (2004*a*). Système de codage d'image jpeg 2000 – partie 3 : Motion jpeg 2000. In : *ISO/IEC 15444-1 :2004*.

JTC1/SC29 (2004*b*). Technologies de l'information – système de codage d'image jpeg 2000 : Système de codage noyau. In : *ISO/IEC 15444-1 :2004*.

JTC1/SC29 (2004*c*). Technologies de l'information – système de codage d'images jpeg 2000 : Extensions. In : *ISO/IEC 15444-1 :2004*.

JTC1/SC29 (2004*d*). Technologies de l'information – système de codage d'images jpeg 2000 : Tests de conformité. In : *ISO/IEC 15444-1 :2004*.

JTC1/SC29 (2005*a*). Technologies de l'information – système de codage d'image jpeg 2000 – partie 12 : Format iso de base pour les fichiers médias. In : *ISO/IEC 15444-9*. ISO.

JTC1/SC29 (2005*b*). Technologies de l'information – système de codage d'images jpeg 2000 : Outils d'interactivité, interfaces de programmes d'application et protocoles. In : *ISO/IEC 15444-9*. ISO.

Kohler, E., M. Handley et S. Floyd (2006). Datagram congestion control protocol (dccp). *RFC 4340*.

Kohler, Eddie, Mark Handley et Sally Floyd (2003). Designing dccp : Congestion control without reliability. Technical report. ICSI Center for Internet Research.

Li, J-R, Dane Dwyer et V. Bharghavan (1999). A transport protocol for heterogeneous packet flows.. In : *In IEEE Infocom'99*. New York, USA.

Liu, J. et S. Singh (2001). ATCP : TCP for mobile ad hoc networks. *IEEE J–SAC*, **19**(7), 1300–1315.

Loyer, Pierre, Jean-Marie Moureaux et Marc Antonini (2003). Lattice codebook enumeration for generalized gaussian source. *IEEE Transactions on Information Theory*, **49**(2), 521–529.

Marasli, Rahmi (1997). Partially ordered and partially reliable transport protocols : Performance analysis. PhD thesis. University of Delaware.

Marasli, Rahmi, Paul D. Amer et Pillip T. Conrad (1996). Retransmission-based partially reliable transport service : An analytic model. In : *INFOCOM (2)*. pp. 621–629.

Mascolo, Saverio, Claudio Casetti, Mario Gerla, M. Y. Sanadidi et Ren Wang (2001). Tcp westwood : Bandwidth estimation for enhanced transport over wireless links. In : *7th annual international conference on Mobile computing and networking*. ACM Press. Rome, Italie. pp. 287–297.

Mathis, M., J. Madhavi, S. Floyd et A. Romanow (1996). Tcp selective acknowledgment options. *RFC 2018*.

Mathis, Matthew, Jeffrey Semke et Jamshid Mahdavi (1997). The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM SIGCOMM computer Communication Review*, **27**(3), 67–82.

Mogul, J. et S. Deering (1990). Path mtu discovery. *RFC 1191*.

Moraru, Bogdan, Flavius Copaciu, Gabriel Lazar et Virgil Debrota (2003). Practical analysis of tcp implementations : Tahoe, reno, newreno. In : *2nd RoEduNet International Conference, Networking in Education and Research*. Iasi, Romania. pp. 125–130.

Mukherjee, Biswaroop (2000). Time-lined TCP : a transport protocol for delivery of streaming media over the internet. Master's thesis. University of Waterloo, Ontario, Canada.

Mukherjee, Biswaroop et Tim Brecht (2000). Time-lined tcp for the tcp-friendly delivery of streaming media over the internet. In : *8th International Conference on Network Protocols*. Osaka, Japon. pp. 165–176.

Natu, Ambarish, Maria Fresia et Fabio Lavagetto (2005). Transmission of jpeg2000 codestreams over mobile radio channels. In : *IEEE ICIP05 International Conference in Image Processing*. Vol. 1. IEEE Signal Processing Society. pp. 785–788.

Olivier, Philippe, Philippe Owezarski et Kavé Salamatian (2003). Quelques éléments caractéristiques du trafic internet. In : *colloque national "Mesures de l'Internet"*. Nice, France.

Owezarski, Philippe, Nicolas Larrieu, Laurent Bernaille, Walid Saddi, Fabrice Guillemin, Augustin Soule et Kavé Salamatian (2004). Distribution of traffic among applications as measured in the french metropolis project. Technical report. LAAS-CNRS, LIP6-CNRS, France Telecom CNET.

Padhye, J., V. Firoiu, D. Towsley et J. Krusoe (1998). Modeling TCP throughput : A simple model and its empirical validation. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 303–314.

Padhye, Jitendra et Sally Floyd (2001). On inferring tcp behavior.. In : *SIGCOMM*. pp. 287–298.

Piecuch, Mike, Ken French, George Oprica et Mark Claypool (2000). A selective retransmission protocol for multimedia on the internet. In : *SPIE International Symposium on Multimedia Systems and Applications*. Boston, MA, USA.

Postel, Jon (1980). User datagram protocol. *RFC 768*.

Postel, Jon (1981*a*). Internet protocol. *RFC 791*. DARPA Internet Program Protocol Specification.

Postel, Jon (1981*b*). Transmission control protocol. *RFC 793*. DARPA Internet Program Protocol Specification.

Pu, Lingling, Michael W. Marcellin, Bane Vasic et Ali Bilgin (2005). Unequal error protection and progressive decoding for jpeg2000. In : *IEEE ICIP05 International Conference in Image Processing*. Vol. 3. IEEE Signal Processing Society. pp. 896–899.

Raffy, P., Marc Antonini et Michel Barlaud (1998). Optimal subband bit allocation procedure for very low bit rate image coding. *IEE Electronic Letters*, **34**(7), 647–648.

Raman, Suchitra, Hari Balakrishnan et Murari Srinivasan (2000). ITP : An image transport protocol for the internet. *Proc. Intl. Conference on Network Protocols, Japan*.

Ramanathan, Parameswaran (1999). Overload managemennt in real-time control applications using $(m - k)$-firm guarantee. *IEEE Transactions on parallel and distributed systems*, **10**(6), 549–559.

Saïd, A. et W. Pearlman (1996). A new, fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits, Systems for Video Technology*, **6**(3), 243–250.

Sanneck, H. et G. Carle (2000). A framework model for packet loss metrics based on loss runlengths. In : *SPIE/ACM SIGMM Multimedia Computing and Networking Conference*.

Schulzrinne, H., S. Casner, R. Frederick et V. Jacobson (2003). Rtp : A transport protocol for real-time applications. *RFC 3550*.

Smith, Brian C. (1994). Cyclic-udp : A priority-driven best-effort protocol. Computer Science Department Cornell University.

Stewart, R., M. Ramalho, Q. Xie, M. Tuexen et P. Conrad (2004). Stream control transmission protocol (sctp), partial reliability extension. *RFC 3758*.

Taubman, David (2002). Remote browsing of jpeg2000 images. In : *Proceedings of the IEEE International Conference on Image Processing (ICIP), volume 1*. pp. 249–252.

Taubman, David, Erik Ordentlich, Gadiel Seroussi et Ikuro Ueno (2000). Embedded block coding in jpeg2000. In : *IEEE International Conference on Image Processing*. Vol. 2. pp. 33–36.

Taubman, David, Erik Ordentlich, Marcelo Weinberger et Gadiel Seroussi (2002). Embedded block coding in jpeg2000. *Signal Processing - Image Communication*, **17**(1), 49–72.

Taubman, David S. et Michael W. Marcellin (2002). *JPEG2000 Image Compression Fundamentals Standards and Practice*. Kluwer Academic Publishers.

Thie, Johnson et David Taubman (2004). Optimal erasure protection assignment for scalable compressed data with small channel packets and short channel codewords. *EURASIP Journal on Applied Signal Processing*.

Thomos, Nikolaos, Nikolaos V. Boulgouris et Michael G. Strintzis (2004). Wireless transmission of images using jpeg2000. In : *ICIP04 International Conference on Image Processing*. Vol. 4. IEEE Signal Processing Society. pp. 2523–2526.

UIT-T (2003). Temps de transmission dans un sens. *Recommandation G.114*.

Voinson, Teddy, Ludovic Guillemot et Jean-Marie Moureaux (2002). Image compression using lattice vector quantization with code book shape adapted thresholding. In : *IEEE International Conference on Image Processing*. ICIP 2002.

Wang, Ren, Kenshin Yamada, M. Y. Sanadidi et Mario Gerla (2005). Tcp with sender-side intelligence to handle dynamic, large, leaky pipes. *IEEE Journal on Selected Areas in Communications*, **23**(2), 235–248.

Wang, Rn, Massimo Valla, M.Y. Sanadidi et Mario Gerla (2002). Using adaptive rate estimation to provide enhanced and robust transport over heterogeneous networks. In : *10th IEEE International Conference Network Protocols*. Paris, France. pp. 206–215.

Yajnik, Maya, Sue B. Moon, James F. Kurose et Donald F. Towsley (1999). Measurement and modeling of the temporal dependence in packet loss. In : *INFOCOM (1)*. pp. 345–352.

Zhang, Yin (2001). Characterizing End-To-End Internet Performance. PhD thesis. Faculty of the graduate school of Cornell University.

Résumé

Bien que les données qui transitent sur l'Internet et les contraintes inhérentes à leur transmission soient de natures variées, ces transmissions reposent presque toujours sur TCP ou UDP. L'adoption d'une approche de bout en bout impose d'agir sur le service de transport pour s'adapter aux besoins des données en fonction de la qualité de service offerte par le réseau. Les données multimédias en général sont tolérantes aux pertes, elles peuvent alors faire l'objet d'une compression et d'un transport avec pertes tout en conservant une bonne qualité de l'information reconstruite si les informations perdues sont parfaitement contrôlées. Cela suppose que les applications multimédias puissent compter sur un service de transport adapté à leurs besoins, associant les principes de fiabilité partielle et le contrôle d'erreur déterministe.
Dans cette thèse, nous définissons un service de transport à fiabilité partielle de ce type, utilisé pour la transmission d'images fixes compressées sur les réseaux de type « best-effort ». Les travaux portent sur la conciliation du transport à fiabilité partielle avec les données compressées ainsi que la recherche d'un gain de temps de service et d'une dégradation acceptable. Un protocole de transport appelé 2CP-ARQ est proposé à cet effet. Sa compatibilité avec le transport d'images codées selon le standard JPEG2000 est d'abord étudiée. Les résultats de cette étude nous conduisent à élaborer un schéma de compression et d'organisation des données plus approprié à l'utilisation d'un système de transport à fiabilité partielle basé sur 2CP-ARQ. Les résultats montrent que le réseau peut bénéficier de l'utilisation de ce service de transport, qui se traduit par une moindre sollicitation des ressources du réseau, tout en satisfaisant les contraintes de l'application en termes de qualité des images reconstruites.

Mots clés

Qualité de service de bout en bout, protocole de transport, fiabilité partielle, compression d'images, JPEG2000, décomposition multirésolution.

Abstract

Although data can have various and multiple constraints, their transport over the Internet are based on TCP and UDP almost exclusively. The end-to-end approach enables the transport service adaptation according to the type of data as a function of the quality of service provided by networks. Generally speaking, multimedia data are loss-tolerant, so it can afford lossy compression without compromising further human interpretation after reconstruction if the lost piece of information are well controlled. This requires a transport service well suited for the needs of the multimedia applications, with the association of the partial reliability concept and deterministic error recovery mechanisms.
In this thesis we proposes a partially reliable transport service intended to the transmission of compressed still images over best effort networks. We aim to find a tradeoff between the saving of time and the degradation enabled by the partial reliability. A partially reliable service called 2CP-ARQ is presented for that purpose. First, the compliance of 2CP-ARQ with the JPEG2000 standard has been studied. It shows the need to develop a compression scheme that produces independent units suited for packet encapsulation by the transport layer. It leads to the development of a compression scheme 2CP-ARQ conscious in order to satisfy the requirements of the partially reliable service but also that benefits from it. Results show that 2CP-ARQ transport service reduces the network resources consumption while the quality degradation for the reconstructed images can be bounded.

Keywords

End-to-end quality of service, transport protocol, partial reliability, image compression, JPEG2000, multiresolution decomposition.

Université Henri Poincaré, Centre de Recherche en Automatique de Nancy

Faculté des Sciences, B.P. 239, 54506 Vandœuvre lès Nancy